UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA ELÉTRICA
PROJETO DE GRADUAÇÃO

SUELLEN OLIVEIRA COLLI

LUVA COM SENSORES FLEX PARA DETECÇÃO DAS CONFIGURAÇÕES DE MÃO “B”, “E” E “Y” DA LÍNGUA BRASILEIRA DE SINAIS

VITÓRIA – ES
AGOSTO/2017

SUELLEN OLIVEIRA COLLI

LUVA COM SENSORES FLEX PARA DETECÇÃO DAS CONFIGURAÇÕES DE MÃO “B”, “E” E “Y” DA LÍNGUA BRASILEIRA DE SINAIS

Parte manuscrita do Projeto de Graduação da aluna **Suellen Oliveira Colli**, apresentada ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para aprovação na disciplina “ELE08553 – Projeto de Graduação 2”.

_____
Prof. Dr. Paulo J. M. Menegáz
Professor da disciplina

_____
Prof. Dr. Evandro O. T. Salles
Orientador

_____
Suellen Oliveira Colli
Aluno

VITÓRIA – ES
AGOSTO /2017

AGRADECIMENTOS

Toda a glória e honra, dou ao meu Deus, a quem sirvo, pois foi Ele quem me deu vitória, capacitou-me e sustentou-me para chegar até aqui. A Deus, toda glória, honra e louvor!

Agradeço à minha família pelo apoio, carinho, amor e ajuda.

Agradeço aos meus amigos, muito mais que irmãos para mim. Em especial, às meninas, minhas irmãs em Cristo. Amigas maravilhosas para toda vida!

Agradeço às amigas e irmãs que o Senhor me deu. Amizade que vem de Deus!

Agradeço à equipe de Libras, valentes, servos voluntários e amados. Em especial, à equipe voluntária do Maanaim.

Agradeço aos amados irmãos da minha igreja pelas orações, pelo amor e pelo apoio.

Agradeço aos meus irmãos em Cristo que me ajudaram e oram pela minha vida.

Agradeço por aqueles que, com paciência, foram voluntários para as realizações dos testes. Ficaram comigo durante horas. Sou grata ao Senhor por esses meus irmãos em Cristo. Família que Deus me deu.

Agradeço ao meu orientador pelas sugestões, pela disposição e pela paciência. Agradeço, também, aos meus colegas e professores de curso, em especial, aqueles que contribuíram neste projeto, pois com paciência me auxiliaram neste trabalho. Cada código feito, houve uma contribuição! Obrigada meus colegas! E, também, pelos sensores emprestados, agradeço ao professor que me confiou esses materiais!

Epígrafe

"(...) mas o povo que conhece ao seu Deus se esforçará e fará proezas." – Daniel 11:32b

"Não to mandei eu? Esforça-te, e tem bom ânimo; não temas, nem te espantes; porque o Senhor teu Deus é contigo, por onde quer que andares." – Josué 1:9

Bíblia Sagrada – Traduzida em português por João Ferreira de Almeida – Edição revista e corrigida – 1969

RESUMO

Observa-se que existem diversos tradutores digitais, os quais fornecem a tradução de várias línguas. Por exemplo, inglês para espanhol ou japonês para francês. Um exemplo deste tipo de ferramenta é o Google Tradutor. Há, também, tradutores digitais de português para Libras (Língua Brasileira de Sinais). A tecnologia de um tradutor de língua de sinais para outra língua, por exemplo, Libras para português, está sendo desenvolvida atualmente. O objetivo deste projeto de graduação é a construção e desenvolvimento da tecnologia que pode ser usada para a realização desse tipo de tradutor. Assim, propõe-se a construção de uma luva, na qual são alocados dispositivos eletrônicos, por exemplo: sensores flex, módulo Wi-Fi e conversor A/D. Por ela, poderá ser identificado algumas configurações de mãos da Libras. Assim, após realização dos testes com novos usuários, pôde-se observar a viabilidade dos procedimentos realizados neste projeto obtendo-se identificações corretas.

Palavras-chave: Libras, luva sensoreada, configurações de mão, sistemas embarcados.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS, SIGLAS E TERMOS

A/D	Analógico/Digital
ASL	*American Sign Language* (Língua de sinais americana)
BD	Banco de Dados
BIM	*Bahasa Isyarat Malasya* (Língua de sinais da Malásia)
BSL	*Brazilian Sign Language* (Língua de sinais brasileira)
GND	*Ground* (Terra)
I2C	*Inter-Integrated Circuit* (Circuito Inter-Integrado)
IBGE	Instituto Brasileiro de Geografia e Estatística
ICM	Igreja Cristã Maranata
IDE	*Integrated Development Environment* (Ambiente de desenvolvimento integrado)
IoT	*Internet of Things* (Internet das Coisas)
Libras	Língua Brasileira de Sinais
LS	Língua(s) de Sinais
LSB	*Brazil Sign Language* (Língua Brasileira de Sinais)
IEEE	*Institute of Electrical and Electronic Engineers* (Instituto de Engenheiros Elétricos e Eletrônicos)
UFES	Universidade Federal do Espírito Santo
USB	*Universal Serial Bus* (Barramento serial universal)

LISTA DE TABELAS

SUMÁRIO

1. INTRODUÇÃO

A Libras é uma língua falada por surdos e intérpretes no Brasil. Resumidamente, é a língua dos surdos brasileiros. A língua de sinais (LS) são línguas usadas pelas comunidades surdas. A língua usada por essas comunidades possuem estruturas gramaticais. LS é visual, composta por sinais e expressões corporais (ICM, [20--], p.18, grifo nosso).

No livro de Ronice Quadros e Lodenir Karnopp: "Língua de sinais brasileira – Estudos linguísticos" relata que as línguas de sinais são consideradas línguas naturais (ou seja, simplesmente um idioma ou uma língua). Essas autoras mencionam que LS tem características específicas, a língua de sinais se distingue da estrutura da língua falada (2004, p.30). Um trecho desse livro demonstra que língua de sinais é um idioma (2004, p.30):

> As línguas de sinais são, portanto, consideradas pela linguística como línguas naturais ou como um sistema linguístico legítimo e não como um problema do surdo ou como uma patologia da linguagem. Stokoe, em 1960, percebeu e comprovou que a língua dos sinais atendia a todos os critérios linguísticos de uma língua genuína, no léxico, na sintaxe e na capacidade de gerar uma quantidade infinita de sentenças.

No Brasil, segundo dados do IBGE, há cerca de 0,9% de brasileiros que ficaram surdos devido a alguma doença ou acidente e 0,2% nasceu surdo (VILLELA, 2015). Importante mencionar que, até o ano de 2015, o IBGE mostrou uma estimativa da população brasileira de mais de 204 milhões (UOL, 2015).

Em 24 de abril de 2002, houve a oficialização da língua brasileira de sinais, conforme a lei nº 10.436. Assim, no Brasil, há duas línguas oficializadas: português e Libras. O artigo de número 3 dessa lei determina que as empresas concessionárias de serviçoes públicos de saúde devem proporcionar a acessibilidade ao surdo aos seus serviços (BRASIL, 2002).

Mesmo com as estimativas relatadas anteriormente acerca da população com deficiência auditiva no Brasil e sobre a lei nº 10.436, ainda há dificuldades quanto à acessibilidade dos surdos em órgãos públicos e particulares. Na Universidade Federal do Espírito Santo, observa-se entre os estudantes e professores, a dificuldade de implementação no que vigora os artigos dessa lei. Sendo que, não há intérpretes para todo o contingente do campus universitário, por exemplo. Essas argumentações são provenientes da observação do cotidiano no campus universitário.

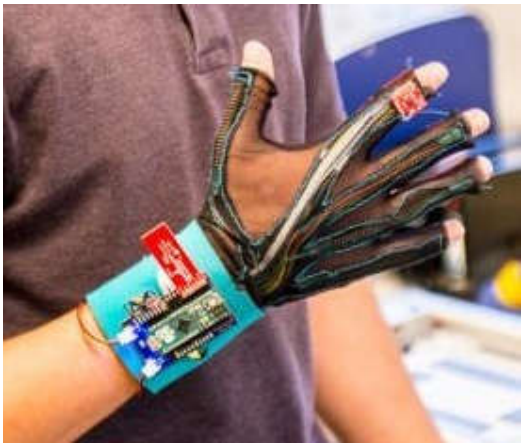
Portanto, mediante essas questões mencionadas inicialmente, é importante a construção de um tradutor de Libras para português. Nesse sentido, quando o sinal em Libras for realizado, a tradução para o português será simultânea. De modo a melhorar a

acessibilidade do surdo e construir um equipamento para maior aprendizagem da Libras. O foco deste projeto está na identificação de determinadas configurações de mãos da Libras por intermédio de uma luva na qual contém sensores (*sensor flex*) para obtenção dos dados a fim de serem feitas análises para validação dessas configurações.

Acerca do tema apresentado neste trabalho quanto a pesquisas e projetos realizados para melhor acessibilidade do surdo, há projetos já desenvolvidos como, por exemplo, as luvas *SignAloud* (sinal em voz alta), projeto desenvolvido pelos estudantes Navid Azodi e Thomas Pryor da Universidade de Washington, nas quais os sinais são reconhecidos e traduzidos automaticamente na língua falada, em tempo real. Os autores do projeto das luvas *SignAloud* desejam aplicá-las em hospitais para monitoramento dos movimentos dos pacientes que sofreram derrame ou sofreram algum acidente que prejudique os movimentos dos mesmo (D'ANGELO, 2016). Contudo, o projeto de Azodi e Pryor está relacionado com a tradução da Língua de Sinais Americana (*ASL*) para o inglês. Isto mostra que há projetos desenvolvidos para tradução de LS de um país para a língua falada desse mesmo país.

A Figura 1 mostra o protótipo do projeto feito por Navid Azodi e Thomas Pryor da Universidade de Washington.

Figura 1 – Luva *SignAloud*



Fonte: D'ANGELO, (2016).

Em 2012, foi publicado o artigo "*American Sign Language Interpreter*" (Intérprete da língua de sinal americana), no qual é proposto a utlização de luvas com sensores para implementação de um programa de ensino da LS (KADAM et al., 2012, p. 157). A luva, proposta por Kadam e colegas, possibilita a prática e aperfeiçoamento da aprendizagem das LS e, também, proporcionar a comunicação com os surdos.

Kadam e colegas criaram uma luva que contêm sensores flex (variam sua resistência de acordo com a sua dobradura) montados sobre os dedos da luva. Estes sensores mudam o valor de sua resistência em função da dobradura dos dedos. Um display LCD é usado para indicar o quanto o usuário necessita dobrar os dedos corretamente dependendo da letra que deseja fazer em *ASL* (língua de sinais usada pela comunidade surda e intérpretes dos Estados Unidos). Há um microcontrolador para armazenar os sinais realizados. (KADAM et al., 2012, p. 157).

A luva tem dois tipos de operação: "modo ensinar" e "modo aprender". No primeiro caso, foi criado um banco de dados para inserir diferentes sinais, correspondentes ao alfabeto da *ASL*. O usuário faz o sinal durante um tempo específico por algumas vezes para que seja gravado no banco de dados. Já no segundo caso, "modo aprender", o usuário faz o sinal de forma a corresponder com o banco de dados existente. Se este não fizer o sinal corretamente, o *firmware* embarcado no microcontrolador indicará que o sinal é inválido e é necessária mais precisão por parte do usuário ao realizar o sinal. (KADAM et al., 2012, p. 157).

A conclusão dos estudos no projeto de (Kadam et al. 2012) foi que os resultados obtidos podem ser reconhecidos em 86% do tempo, dos 14 sinais selecionados. Mas, para a língua de sinais, ainda devem ser considerados outros parâmetros como as expressões faciais, ou seja, expressões não-manuais. Logo, para um tradutor de *ASL* para o inglês, é necessária a extensão dos estudos para este aspecto utilizando "câmeras (ou sensores) e processamento de imagens para capturar as expressões faciais" (KADAM et al., 2012, p. 158).

A Figura 2 mostra o protótipo da luva realizada no projeto (Kadam et al. 2012) descrita em seu artigo.

Figura 2 – Protótipo da luva realizada por (Kadam et al. 2012)

Fonte: KADAM et al. (2012).

Há diversas pesquisas já realizadas e outras que estão sendo feitas acerca da construção de um tradutor de LS para a língua falada. Em (Swee et al. 2007), propõe-se um sistema de reconhecimento da LS por uma luva com comunicação sem fio via *Bluetooth* (para processamento dos dados obtidos na realização do sinal feito pelo usuário), possibilitando a comunicação de surdos com ouvintes. O sistema reconhece 25 palavras da língua de sinais malasiana, conhecida como *BIM* (*Bahasa Isyarat Malasya*).

Swee e colegas introduzem uma explanação sobre a LS em seu texto semelhante a que é feita para caracterização da Libras (2007, p. 1, grifo nosso):

> A língua de sinais é uma habilidade de comunicação que utiliza gestos (*sinais*) em vez de som para transmitir um significado. Combinando configurações de mão, orientação e movimento das mãos, braços ou corpo e expressões faciais (*expressões não-manuais*) para *se expressar* [...].

A língua de sinais pode ser definida dessa forma, contudo, ela não é universal. Ou seja, cada país tem sua LS própria. Por isso, há *ASL* (Estados Unidos), Libras (Brasil), *BIM* (Malásia), por exemplo.

Resumidamente o projeto descrito em (Swee et al. 2007) utiliza: um par de luvas com sensores flex e acelerômetros (estes últimos colocados no ombro, cotovelo e braço) que obtêm os dados para captura da posição dos dedos do usuário e do movimento do braço para

se ter o sinal em *BIM*. Quando o usuário executa um sinal, os sensores (sensores flex e acelerômetros) fornecem medições em sinais analógicos que, após conversão analógico/digital, serão processados em um microcontrolador. Em seguida, o microcontrolador envia os dados remotamente via *Bluetooth*. Swee e colegas informam que os sensores flex têm uma faixa de 10 kΩ (totalmente esticado) e 30 kΩ (totalmente dobrado). Esse projeto conseguiu identificar aproximadamente 25 palavras da *BIM*.

A Figura 3 mostra os sensores utilizados em (Swee et al. 2007) e a localização destes no usuário.

Figura 3 – Disposição dos sensores flex e do acelerômetro no protótipo a serem utilizados pelo usuário do projeto (Swee et al. 2007).

Fonte: SWEE et al. (2007); legenda: própria autora. (2016).

O interesse nesses estudos no que tange este projeto de graduação é como poderia ser construído um tradutor de Libras para português. Para a obtenção de tal tradutor, é necessário tecnologias que identifiquem os 5 parâmetros dessa língua.

Sendo assim, o objetivo desse projeto de graduação é construir uma das tecnologias integrantes para a composição de tal tradutor, o qual possa identificar um desses parâmetros, nesse caso: a configuração de mão.

No livro de Ronice Quadros e Lodenir Karnopp: "Língua de sinais brasileira – Estudos linguísticos" mostra que o sinal de Libras é caracterizado por cinco parâmetros (2004, p. 53-61):

- configuração de mãos;
- movimento;
- localização;

- orientação de mão e
- expressões não-manuais.

Para a construção de um tradutor de Libras para o português, é necessária desenvolvimento de tecnologias cuja aplicação seja a identificação desses parâmetros. Para um ponto de partida nesses estudos, propõe-se a detecção de determinadas configurações de mãos da Língua Brasileira de Sinais. Estudos desta natureza podem contribuir para, futuramente, facilidade e tornar pervasivo o meio de comunicação eletrônico com os surdos e, também, uma futura ferramenta de aprendizagem de Libras, semelhantemente como é empregado quanto ao projeto proposto por Kadam e colegas, por exemplo.

Portanto, o objetivo deste projeto é a construção de uma luva que contém sensores para identificação de três determinadas configurações de mãos, mostradas na Figura 4 da Língua Brasileira de Sinais. Ou seja, a configuração de mão é feita por um usuário vestindo a luva e, em seguida, identificada.

Figura 4 – Configurações usadas como parâmetros para testes e validação no protótipo

Fonte: QUADROS, R.; KARNOPP, L. (2004).

Assim, os objetivos específicos deste projeto são a identificação e validação dos dados obtidos a partir da luva com os sensores flex para confirmação da detecção correta de determinada configuração de mão, a detecção dos dados de cada dedo da mão feita por sensores flex, a transmissão dos dados obtidos via Wi-Fi, a criação de um banco de dados para armazenamento dos valores obtidos e calculados para posterior análise acerca de cada configuração de mão realizada e desenvolvimento de uma interface para a análise e validação dos dados obtidos por testes realizados em usuários de Libras.

Assim, o projeto apresenta em seu capítulo 2, o embasamento teórico, destacando os materiais e métodos utilizados quanto à aplicação do trabalho. No capítulo 3, mostra a metodologia de desenvolvimento usada para a realização dos testes. No capítulo 4, apresentam-se os materiais utilizados para a confecção do protótipo. No capítulo 5, apresentam-se os resultados obtidos dos testes realizados. No capítulo 6, têm-se discussões finais, conclusões e propostas de trabalhos futuros. No capítulo 7, mostram-se as referências usadas neste projeto. No apêndice A, encontra-se o algoritmo desenvolvido e implementado

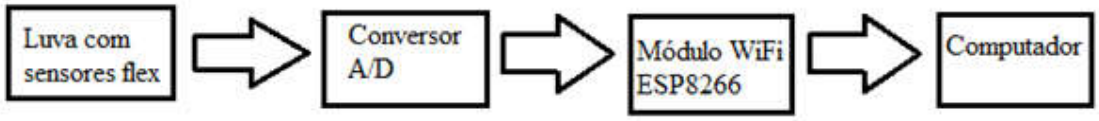
no módulo Wi-Fi ESP8266. No apêndice B, apresenta-se o algoritmo desenvolvido no programa Visual Studio. Por fim, no apêndice C, mostra-se o algoritmo implementado no banco de dados MySQL.

2. EMBASAMENTO TEÓRICO

2.1 Materiais e Métodos

O diagrama de blocos do sistema desenvolvido pode ser visualizado na Figura 5. Como se observa, esse projeto caracteriza-se por uma aplicação da eletrônica analógica, conversão A/D, comunicação de rede sem fio e manipulação dos dados via softwares do computador.

Figura 5 – Disposição das etapas do projeto segundo os equipamentos a serem usados



Fonte: própria autora. (2016).

Sobre a Figura 5:

- luva com sensores flex: disponibilização dos sinais;
- conversor A/D: converte os sinais dos sensores (analógico para digital);
- módulo Wi-Fi ESP8266: envio remoto do sinal digital para o computador;
- computador: obtenção das amostras e valores base em relação aos usuários I, L e S, obtenção das amostras de novos usuários, cálculo de variáveis desejadas, análise e validação dos dados obtidos através do programa Visual Studio e armazenamento dos dados no banco de dados MySQL.

No capítulo 3, as Figuras 17 e 19 mostram o projeto construído e a disposição dos dispositivos eletrônicos para montagem em uma placa, respectivamente.

Nas próximas seções, segue uma explanação desses elementos:

2.1.1 Sensores flex

Os sensores flex são resistores que variam sua resistência em função da flexão que neles é empregada. A **Erro! Fonte de referência não encontrada.** apresenta o aspecto físico desse sensor.

Figura 6 – Sensor flex

Fonte: própria autora. (2016).

O sensor da **Erro! Fonte de referência não encontrada.** é de 4,5 polegadas. É um dos sensores usado neste projeto. Sua resistência quando está plano (não é feita nenhuma dobradura) é de 10 kΩ, sendo sua faixa de resistência da curvatura entre 60 a 110 kΩ (SPECTRA SYMBOL, [20--?], p.1).

2.1.2 Módulo Wi-Fi ESP8266 NodeMCU

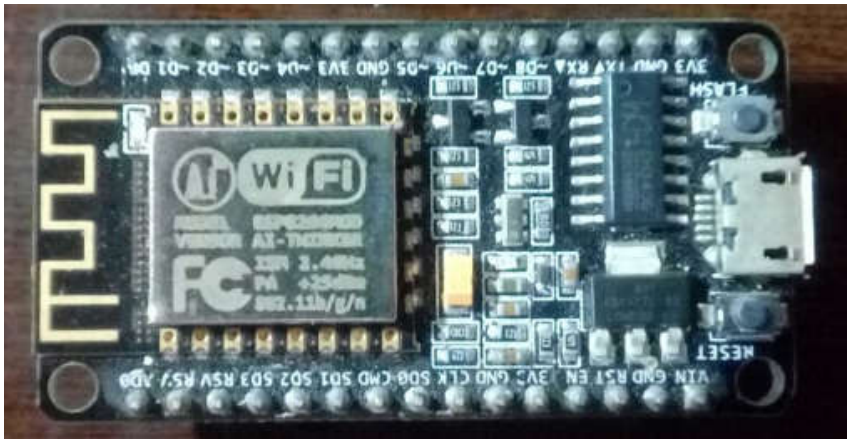
O ESP8266 está relacionado com a Internet das coisas, conhecida como *IoT* (*Internet of Things*). Soluções *IoT* surgiram naturalmente com o crescimento da teconologia relacionada à conexão via Wi-Fi de dispositivos eletrônicos. O módulo em questão integra o conceito de *IoT*, pois convém ao ESP8266 à conectividade e à mobilidade da luva com os sensores flex e conversor A/D. O módulo também é responsável pelo envio remoto dos dados obtidos para o computador, ou qualquer dispositivo microprocessado, conectado com a rede disponibilizada por esse módulo. Esta é apenas uma das aplicações possíveis de ser empregada com o uso do ESP8266.

O ESP8266 NodeMCU é um módulo Wi-Fi. Este dispositivo pode funcionar como um ponto de acesso, ou seja, como um servidor que provê acesso a uma rede criada pelo usuário. No caso deste projeto, é criada uma rede chamada de “rede_luva” no ESP8266, o qual envia os dados convertidos pelo ADS7828 dos sensores flex embutidos na luva para o computador que está conectado a essa rede. As ligações entre o conversor e esse módulo podem ser vistas na Figura 19.

Este componente pode trabalhar como uma Estação, ou seja, é quando o dispositivo pode se comunicar com outro roteador, o qual está conectado à Internet. O ESP8266 pode trabalhar também como um Ponto de Acesso, ou seja, o próprio ESP8266 pode ter uma rede criada e um usuário acessá-la (ROBOCORE, 2015). No caso deste projeto, o ESP8266 será utilizado como Ponto de Acesso.

A Figura 7 mostra o aspecto físico do módulo ESP8266 utilizado no projeto.

Figura 7 – Módulo Wi-Fi ESP8266 NodeMcu



Fonte: própria autora. (2017).

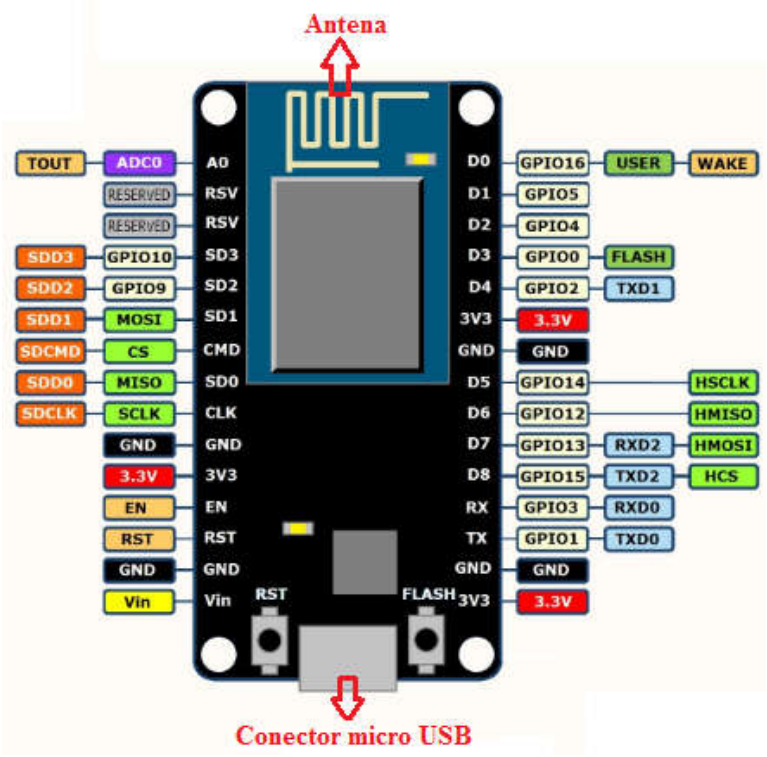
Quanto ao hardware do módulo Wi-Fi ESP8266 NodeMCU (THOMSEN, 2016):

- há 2 botões: FLASH (para gravação do *firmware*) e RST (*Reset*);
- entre esses botões, há um conector micro-*USB* para alimentação e conexão com o computador;
- nas laterais, existem os pinos de entrada e saída (GPIO), alimentação externa, comunicação, etc;
- existe uma antena embutida.

Observação: *firmware* "é basicamente parecido com o *software* (programa), ou seja, ele é um aplicativo que pode ser instalados em hardwares", segundo Lucas Tedeschi (2011).

A Figura 8 mostra o módulo ESP8266 em relação aos seus pinos.

Figura 8 – Módulo ESP8266 em relação aos seus pinos



Fonte: THOMSEN, A. (2017); modificado pela própria autora (2017).

No capítulo 3, “Metodologia de Desenvolvimento”, será descrito a aplicação desse módulo no projeto.

2.1.3 ADS7828

2.1.3.1 Apresentação do conversor

Foi necessária a utilização deste conversor, pois o ESP8266 dispõe apenas de uma entrada para conversão analógico/digital. Sendo que, no caso deste projeto, são usadas cinco entradas para cada sensor flex disposto em cada dedo da mão.

O ADS7828 é um conversor A/D. Seu *datasheet* (ficha de dados) o descreve como sendo “um dispositivo de aquisição de dados de 12 bits de fornecimento simples, de baixa potência, que possui uma interface serial *I2C* [...].” (TEXAS INSTRUMENTS, 2001, p. 1).

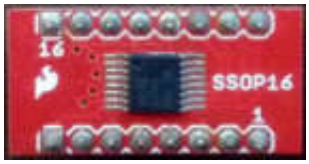
O conversor ficará disposto entre os sensores flex e o ESP8266. Fazendo a aquisição de dados analógicos dos sensores, convertendo estes sinais para sinais digitais e transmitindo-os para o módulo ESP8266. A Figura 9 apresenta o conversor ADS7828.

Figura 9 – Conversor utilizado no projeto acoplado em um adaptador de 16 pinos

Fonte: TEXAS INSTRUMENTS, (2016).

A Figura 10 mostra o módulo do conversor utilizado no projeto acoplado em um adaptador de 16 bits.

Figura 10 – Conversor utilizado no projeto acoplado em um adaptador de 16 pinos

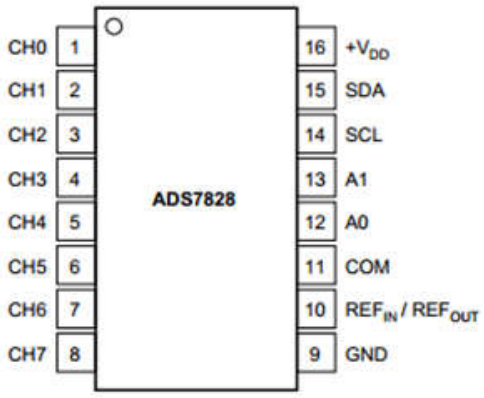


Fonte: própria autora. (2017).

O adaptador se faz necessário para realização de testes preliminares no *protoboard*, pois o conversor é muito pequeno (largura, espessura e comprimento possuem valor menor que dezena de milímetros).

A Figura 11 apresenta um esquemático do conversor com descrição dos pinos.

Figura 11 – Conversor ADS7828, configuração dos pinos e descrição destes



Fonte: Texas Instruments. (2017).
Nota:
CH0 a CH7 (pinos de 1 a 8): canal de entrada analógica;
Pino 9: terra (*ground*) analógico;
Pino 10: referência de interna (2.5 V) / referência de entrada externa;
Pino 11: comum para canal de entrada analógico;
Pino 12 e 13: bit de endereço de escravo;
Pino 14: *serial clock*;
Pino 15: *serial data* (dados);
Pino 16: fonte de alimentação nominal de 3.3 V.

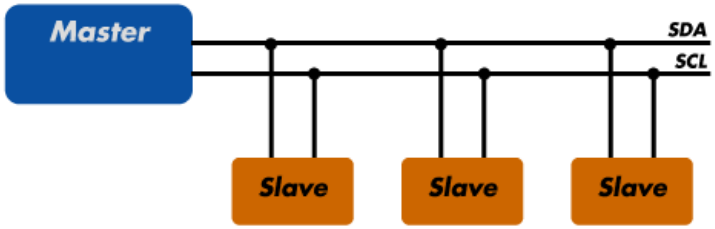
2.1.3.2 Comunicação entre o ADS7828 e o ESP8266

O *datasheet* do dispositivo, disponibilizado pela *Texas Instruments*, mostra a forma de endereçamento de bytes, uma vez que, é necessária a correta programação em relação ao conversor para a manipulação dos valores que este estará recebendo dos sensores e transmitirá para o ESP8266. Neste último, será implementado o algoritmo desenvolvido, pois o ESP8266 atuará como mestre e o conversor como escravo.

O próprio *datasheet* relata que o ADS7828 tem interface *I2C*. Essa interface possui um protocolo de comunicação entre um dispositivo "mestre" e dispositivos "escravos". O "mestre" comanda o acesso ao barramento e os "escravos" o acessam conforme comando do "mestre". (REIS, 2014)

A Figura 12 ilustra a conexão utilizando *I2C*.

Figura 12 – Ilustração da forma de conexão utilizando *I2C*



Fonte: REIS, V. (2014).

Nota: *Master* – Mestre; *Slave* – Escravo.

Neste projeto, o mestre é o ESP8266 que comanda o escravo, o qual é o ADS7828. O mestre gera o sinal SCL (relógio serial), que controlará o acesso ao barramento (TEXAS INSTRUMENTS, 2001, p. 9).

SDA (*Serial Data*) é o pino no qual são transferidos ou recebidos os dados e SCL (*Seria Clock*) é o pino que faz a temporização entre os dispositivos escravos, ou seja, cada um irá acessar o barramento de acordo com a temporização feita por SCL (REIS, 2014).

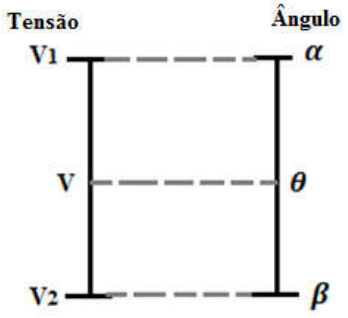
Portanto, resumidamente, quanto à implementação do conversor ADS7828 no projeto: será feita uma programação no ESP8266, o qual atuará como mestre. O algoritmo implementado disporá do correto endereçamento de bytes do conversor para aquisição e manipulação dos dados provenientes dos sensores para transmissão destes para o módulo ESP8266, após conversão do sinal feita pelo ADS7828.

A forma de como foram feitas as conexões entre os pinos no projeto e o desenvolvimento do algoritmo para tratamento dos dados recebidos pelos sensores e transmitidos ao conversor e enviados por este ao ESP8266 estão no capítulo 3, "Metodologia de Desenvolvimento".

2.1.3.3 Sinais de entrada

Os sinais de entrada são obtidos através de um divisor de tensão para coleta de dados em relação aos sensores flex (veja Figura 19). A Figura 13 apresenta um esquema para a compreensão de como são tratados os sinais de entrada.

Figura 13 – Relação entre variáveis (tensão e ângulo)



Fonte: própria autora. (2017).

Na referida figura, tem-se:

- $V_1 = V_{cc} = 3{,}3$ V (Tensão de alimentação do circuito);

- $V_2 = GND$ (Ground – Terra);
- V – Tensão de entrada (fornecida pelos sensores);
- $\alpha = 0$ – Dedo totalmente esticado;
- $\beta = 90°$ – Dedo totalmente dobrado;
- θ – Ângulo correspondente à tensão de entrada.

Assim, obtêm-se a seguinte equação com base no esquema da Figura 13:

$$\frac{V - V_2}{V_1 - V_2} = \frac{\theta - \beta}{\alpha - \beta}$$

$$(V - V_2)(\alpha - \beta) = (V_1 - V_2)(\theta - \beta)$$

$$\therefore \theta = \frac{(V - V_2)(\alpha - \beta)}{(V_1 - V_2)} + \beta$$

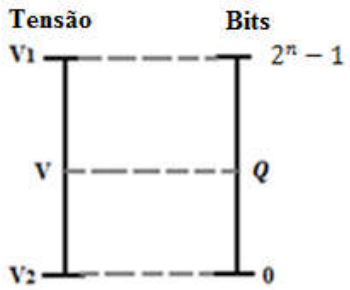
$$\theta = \frac{(V - 0)(\alpha - \beta)}{(3{,}3 - 0)} + \beta$$

$$\theta = \frac{V(\alpha - \beta)}{3{,}3} + \beta. \quad (1)$$

2.1.3.4 Relação entre tensão e quantidade de bits

O ESP8266 manda os dados dos sensores, via Wi-Fi, para um computador. O algoritmo em que isso é implementado se encontra no Apêndice B. Os dados estão contidos em um vetor de cinco posições (chamado de "x" no algoritmo). Cada posição do vetor é o valor referente a cada sensor flex. A Figura 14 apresenta um esquema em que é demonstrado como é obtido o sinal de saída do conversor, para valor em tensão, em relação ao sinal de entrada fornecido pelo sensor flex.

Figura 14 – Relação entre variáveis (tensão e bits)



Fonte: própria autora. (2017)

Na figura referida, tem-se:

- V – Tensão de entrada (divisor de tensão, veja Figura 19), em V;

- $V_1 = 2{,}5$ V (pois, utiliza-se a referência interna do ADS7828, o qual tem um circuito interno que fornece esse valor de tensão) (TEXAS INSTRUMENTS, 2001, p. 3);
- $V_2 = GND$;
- Q – Valor em bits em relação da tensão obtida;
- n – Número de bits do conversor. Pelo seu *datasheet*, é 12 bits (TEXAS INSTRUMENTS, 2001, p. 1).

A relação entre tensão obtida pelo sensor e os dados fornecidos (em bits) pela saída do conversor é a seguinte:

$$\frac{V - V_2}{V_1 - V_2} = \frac{Q - 0}{(2^n - 1) - 0}$$

$$\frac{V - 0}{2{,}5 - 0} = \frac{Q}{2^{12} - 1}$$

$$Q = 1638V$$

$$V \cong 0{,}0006105Q. \qquad (2)$$

A equação 2 é utilizada no algoritmo para conversão dos dados e é mostrada na interface apresentada nas Figuras 21 e 22 (onde são apresentados os valores, em tensão, dos sensores em relação à cada dedo).

2.2 Recursos de software

2.2.1 *Softwares* utilizados

Para realização do projeto, foi necessária a utilização dos programas: Esplorer, Visual Studio e MySQL. No Esplorer, realiza-se a programação do ESP8266. O Visual Studio faz a manipulação e análise dos dados que o módulo Wi-Fi está transmitindo ao computador.

Para armazenamento das variáveis calculadas no Visual Studio foi criado um BD. Assim, com esses valores, pode-se construir um gráfico para análise dos limites de valores de cada configuração. Além da possibilidade da identificação da configuração de mão realizada por um novo usuário.

O BD utilizado foi o MySQL, o qual é um banco de dados de código aberto, com o objetivo de se obter tabelas para armazenamento das amostras e das variáveis calculadas a partir destas. Assim, com esse software, consegue-se gerar um arquivo csv com as variáveis

que foram incluídas em uma tabela criada, a qual pertence ao banco de dados também criado. Através de um tutorial disponibilizado por Jussimar Leal, "Conectando C# a um Banco de Dados MySql", conseguiu-se fazer a criação do BD (LEAL, 2017).

A partir do arquivo csv, pode-se convertê-lo para arquivo xls (Excel da Microsoft Office). Logo, consegue-se manipular os dados de tal maneira a apresentá-lo em gráficos (CASSIUSMG, 2010). E, também, usá-los para fazer a comparação com os novos dados do novo usuário e os demais dados que foram calculados e armazenados no banco de dados para fazer a validação da configuração de mão do novo usuário. A forma como isso é feito está demonstrado no capítulo 3, "Metodologia de Desenvolvimento".

2.2.2 Cálculos realizados no Visual Studio

No âmbito do presente trabalho, foram selecionadas três configurações de mão ("B", "E" e "Y") para um estudo inicial para a obtenção de uma correta validação da configuração feita por um novo usuário. Para efeitos de identificação de uma das três configurações de mãos estudadas, foi pedido aos usuários que repetissem cada configuração diversas vezes. Assim, uma análise estatística se fez necessária.Os dados obtidos e calculados no algoritmo construído no Visual Studio foram: amostras coletadas, valor médio, desvio padrão limites inferior e superior de cada configuração de mão para cada usuário.

O cálculo do valor médio é feito para cada configuração de mão e para cada dedo. Por exemplo, para configuração de mão B e dedo polegar. Assim, de maneira geral, a fórmula matemática é:

$$\bar{x} = \frac{\sum_{i=1}^{n} x}{n} \tag{3}$$

Sendo:

- $\bar{x}$: valor médio;
- x: valor da amostra em tensão;
- n: quantidade de amostras.

Cálculo da variância (para obtenção do desvio padrão, o qual é a raiz quadrada da variância):

$$\sigma^2 = \frac{\sum_{i=1}^{n} |x - \bar{x}|^2}{n} \tag{4}$$

Sendo σ^2 a variância.

Cálculo do desvio padrão (σ):

$$\sigma = \sqrt{\sigma^2} = \sqrt{\frac{\sum_{i=1}^{n} |x - \bar{x}|^2}{n}} \quad (5)$$

Cálculo do limite inferior (li) e limite superior (ls):

$$li = \bar{x} - \sigma \quad (6)$$

$$ls = \bar{x} + \sigma \quad (7)$$

Esses cálculos foram realizados para cada configuração de mão e dedo. Por exemplo: configuração B e dedo polegar, configuração B e dedo indicador e assim por diante até a última configuração e dedo (Y e mínimo).

Também foram feitos cálculos dos valores absolutos da diferença entre o valor médio base de cada configuração de mão para cada dedo e o valor médio de cada dedo do novo usuário.

$$d = \bar{x}_{base} - \bar{x}_{nu} \quad (8)$$

Sendo:

- d: diferença;
- $\bar{x}_{base}$: valor médio base;
- $\bar{x}_{nu}$: valor médio do novo usuário.

Essa variável é usada para observar qual configuração mais próxima que o novo usuário está fazendo. Por exemplo, tomando o valor da diferença para o dedo polegar do novo usuário e se faz o cálculo da diferença para todas as configurações de mão para aquele dedo:

$$d_{B\_Pol} = |\bar{x}_{base_{B\_Pol}} - \bar{x}_{nu_{B\_Pol}}|$$

$$d_{E\_Pol} = |\bar{x}_{base_{E\_Pol}} - \bar{x}_{nu_{E\_Pol}}|$$

$$d_{Y\_Pol} = |\bar{x}_{base_{Y\_Pol}} - \bar{x}_{nu_{Y\_Pol}}|$$

Sendo:

- d_{B_Pol}: diferença da configuração B quanto ao dedo polegar;
- $\bar{x}_{base_{B\_Pol}}$: valor médio base da configuração B quanto ao dedo polegar;
- $\bar{x}_{nu_{B\_Pol}}$: valor médio do novo usuário da configuração B quanto ao dedo polegar;
- d_{E_Pol}: diferença da configuração E quanto ao dedo polegar;
- $\bar{x}_{base_{E\_Pol}}$: valor médio base da configuração E quanto ao dedo polegar;

- $\bar{x}_{nu_{E\_Pol}}$: valor médio do novo usuário da configuração E quanto ao dedo polegar;
- d_{Y_Pol}: diferença da configuração Y quanto ao dedo polegar;
- $\bar{x}_{base_{Y\_Pol}}$: valor médio base da configuração Y quanto ao dedo polegar;
- $\bar{x}_{nu_{Y\_Pol}}$: valor médio do novo usuário da configuração Y quanto ao dedo polegar.

O menor valor é selecionado como o que caracteriza ser mais próximo daquela configuração de mão. Esta análise é feita para todos os dedos. O conjunto de valores do novo usuário que apresenta a maior proximidade com o conjunto de valores de uma das configurações B, E ou Y é caracterizado como sendo a configuração de mão mais próxima que o novo usuário realizou. Na subseção 3.5, é explanado como isto é feito no código no Visual Studio.

3. METODOLOGIA DE DESENVOLVIMENTO

3.1 Introdução à metodologia de desenvolvimento do projeto

O projeto se resume em uma ferramenta eletrônica. A luva é composta por cinco sensores flex, dispostos em cada dedo (veja Figura 17 na qual é mostrada a luva com os sensores flex dispostos nos dedos).

Para a identificação da configuração de mãos, pode ser usada apenas uma luva. Porém, quanto à identificação de certos sinais, seriam necessárias duas luvas. Pois, há sinais que são feitos com as duas mãos. A Figura 15 mostra um exemplo disto.

Figura 15 – Sinal de "casa" em Libras

Fonte: PRODEAF, (2016).

Contudo, para a identificação de um sinal de Libras são necessários mais recursos tecnológicos para a identificação dos demais parâmetros desta língua.

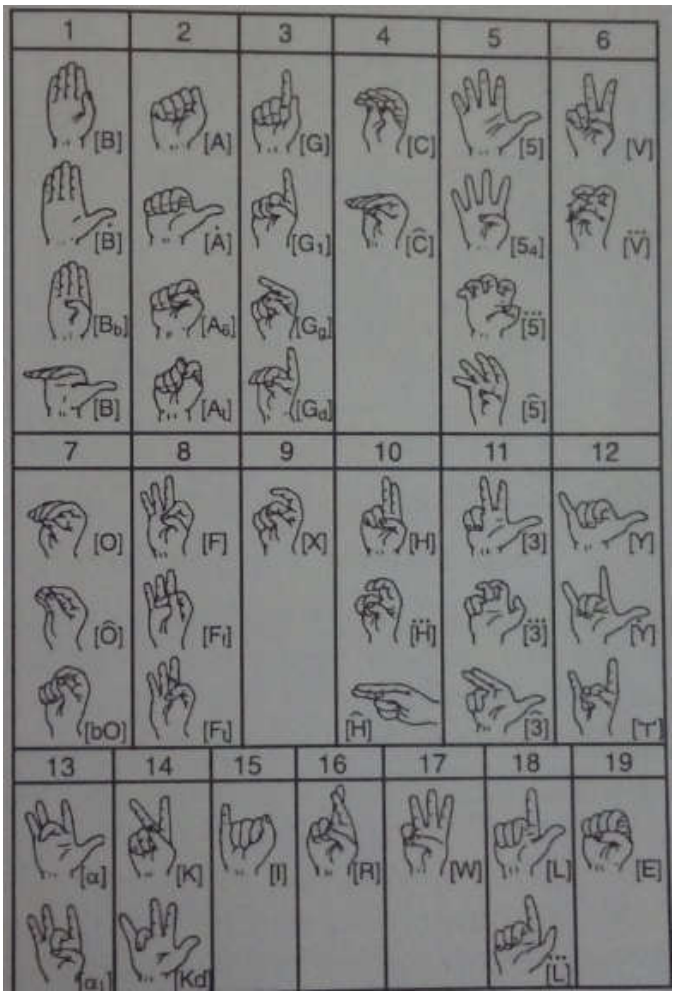
Para o processamento dos sinais obtidos pelos sensores flex, utiliza-se o módulo Wi-Fi ESP8266, o qual faz o acesso remoto ao computador. Nesse módulo, há um algoritmo desenvolvido para este projeto para tratamento dos sinais obtidos. Esses sinais são enviados remotamente para o computador. Neste, é analisado e validado, pelo programa Visual Studio, a configuração de mão realizada. A alimentação do módulo se dá diretamente pela conexão *USB* com o computador e a alimentação do conversor se dá por uma tensão de 3.3 V fornecida pelo próprio ESP8266.

A programação é feita pela linguagem de programação Lua, utilizando a comunicação via cabo micro-*USB* (computador → módulo Wi-Fi ESP8266).

No caso do módulo ESP8266 utilizado, há apenas um pino usado para conversão de analógico para digital. Logo, é usado um conversor A/D para se ter como entrada os sinais analógicos provenientes dos sensores flex e saída digital.

Ferreira-Brito e Langevin (1995, citado por QUADROS; KARNOPP, 2004, p. 53) indicam 46 configurações de mãos da Libras. A Figura 16 mostra tais configurações de mãos.

Figura 16 – Configurações de mãos da Libras

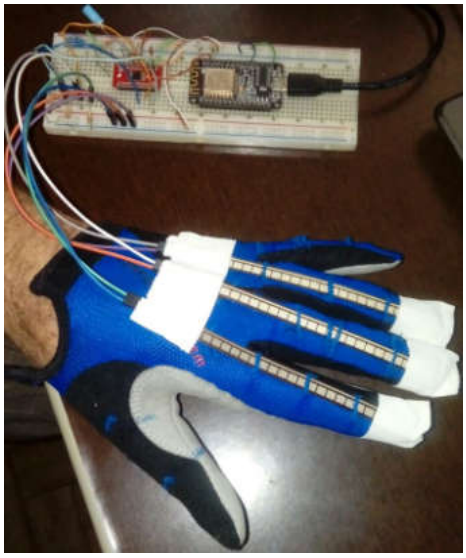


Fonte: QUADROS, R.; KARNOPP, L. (2004).

Dentre as 46 configurações de mãos apresentadas, foram escolhidas as configurações “B”, “E” e “Y” (mostradas na Figura 4) para questão do emprego da metodologia de desenvolvimento para verificação da correta validação das configurações de mão realizadas durante os testes na luva com os novos usuários.

A Figura 17 apresenta uma foto tirada do projeto construído.

Figura 17 – Projeto construído



Fonte: própria autora. (2017).

Na Figura 17, mostra-se a luva apenas com três sensores flex (nos dedos indicador, médio e anelar). Entretanto, o desejável é serem cinco sensores flex dispostos em todos os dedos da mão. A limitação deve-se à quantidade de sensores flex disponíveis para a realização dos experimentos, que foram apenas três. Contudo, para efeitos de identificação de uma das três configurações de mãos estudadas (“B”, “E” ou “Y”), foi pedido aos usuários que repetissem cada configuração vinte vezes. A coleta de amostras foi feita em duas etapas:

- Os sensores flex ficaram dispostos nos dedos indicador, médio e anelar. Foram coletadas as amostras dos sensores, em especial, desses três dedos e feito o armazenamento no BD;
- Os sensores flex ficaram dispostos nos dedos polegar, médio e mínimo. Foram coletadas as amostras dos sensores, em especial, dos dedos polegar e mínimo e feito o armazenamento no BD.

Os sensores flex fornecem valores que são convertidos pelo conversor ADS7828. Este fornece tais valores para o ESP8266, o qual enviará os dados para cálculos e armazenamento em um banco de dados para cada dedo e para cada configuração de mão realizada da Libras.

3.2 Procedimento

Para validação dos dados obtidos para confirmação correta da configuração de mão realizada, foram feitos testes com três usuários:

- usuário S: foi classificado como usuário S, pois se trata de uma pessoa surda;
- usuário I: foi classificado como usuário I, pois se trata de um intérprete de Libras, certificado pelo Prolibras, o qual é um certificado de proficiência da Libras (MINISTÉRIO DA EDUCAÇÃO, [2016 ou 2017]);
- usuário L: foi classificado como usuário L, pois se trata de uma pessoa leiga no assunto concernente à Libras, ou seja, não conhece essa língua.

Todos os voluntários fizeram as configurações de mãos propostas (veja Figura 4). Foram coletadas 20 amostras de cada dedo (polegar, indicador, médio, anelar e mínimo) para cada configuração de mão de cada usuário. A coleta das amostras de todos os dedos para cada configuração de mão foi feita em duas etapas: inicialmente, coletavam-se os dados, em especial, dos dedos indicador, médio e anelar; em seguida, retiravam-se os sensores dos dedos indicador e anelar e colocava-os sobre os dedos polegar e mínimo para, assim, fazer a coleta dos dados, em especial, destes dois dedos. Nessas duas etapas, o usuário repetia a configuração de mão vinte vezes, pois foram coletadas vinte amostras de cada dedo para cada configuração de mão realizada.

Quanto ao usuário L, foi-lhe ensinado as configurações de mãos da Libras por um intérprete dessa língua. Este usuário realizou as configurações conforme aprendeu.

A proposta quanto a esses três tipos de voluntários serve para a obtenção de dados para realização de um banco de dados, o qual conterá dados das configurações médias de cada dedo referente à cada configuração de mão feita. Podendo, assim, proporcionar para outro usuário (além dos 3 mencionados: I, L e S) a possível qualificação de qual configuração de mão ele está fazendo enquanto veste a luva.

A proposta de fazer um BD com valores base usando esses três usuários (I, L e S) é para obterem-se valores de referência que possam abranger qualquer futuro usuário da luva. Ou seja, a luva será útil tanto para uma pessoa que não conhece Libras, mas tem o interesse em aprender até a uma pessoa com profundo conhecimento desta (intérpretes e surdos conhecedores dessa língua), permitindo que os dados coletados do novo usuário sejam comparados com os valores base para validação.

Após a realização dos testes com os 3 usuários, os seguintes dados são armazenados no banco de dados:

- amostras coletadas;
- valores médios;
- desvio padrão e
- limites.

Todos os valores são para cada dedo e cada configuração de mão realizada. Somente no primeiro item é que são armazenados os valores para cada usuário. Já nos três itens seguintes, são valores calculados com base nas amostras coletadas nos testes realizados nos usuários I, L e S. Estes últimos são chamados valores base.

Com esses dados, os valores base de cada configuração de mão, pode-se criar um gráfico mostrando a faixa de valores que cada configuração feita tem, ou seja, limites mínimo e máximo de valores para cada configuração de mão (veja capítulo 5, “Resultados obtidos”).

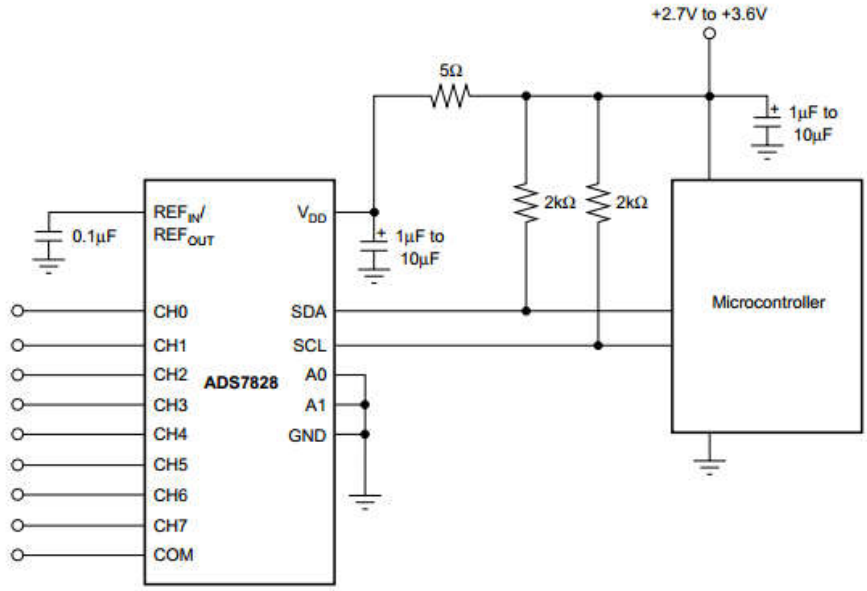
Depois, foram realizados mais dois testes com dois usuários: um novo usuário e outro novamente com o usuário I, com a finalidade de fazer a validação. Ou seja, esses usuários realizam uma das três configurações de mão proposta, verificam-se os valores coletados destes comparando-se com os valores base para fazer a validação. Observação: a validação é feita através de testes com novos usuários um de cada vez.

Isto foi realizado através da interação do circuito montado com os programas Visual Studio, MySQL Workbrench e USBWebserver. O USBWebserver faz com que ocorra o link entre o Visual Studio e o banco de dados MySQL (LEAL, 2017). Ou seja, os dados, provenientes dos sensores flex, são convertidos pelo ADS7828, passados para o ESP8266, enviados via Wi-Fi para o computador, capturados pelo Visual Studio, através de um algoritmo construído para essa finalidade e, também, para os cálculos e armazenamento de dados no BD.

3.3 Conversor A/D

Trabalhou-se no conversor, baseando-se em seu *datasheet*, ou seja, suas ligações com o microcontrolador, no caso o ESP8266, foram feitas conforme seu *datasheet* mostra (TEXAS INSTRUMENTS, 2001, p. 9). Essa ligação pode ser visualizada na Figura 18.

Figura 18 – Ligação das entradas e saídas do ADS7828



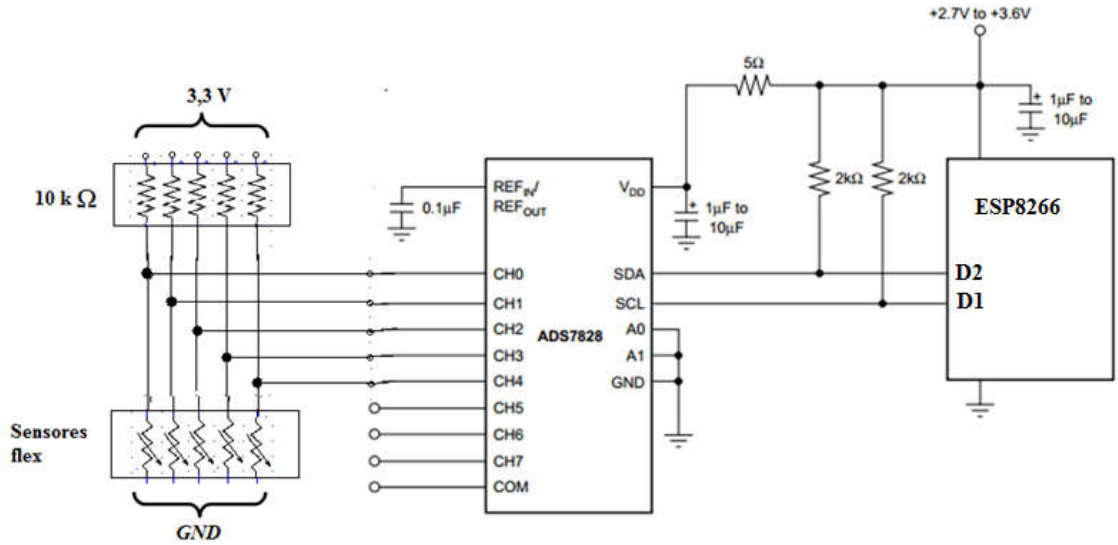
Fonte: TEXAS INSTRUMENTS. (2001).

No caso deste projeto, foram usados os resistores de 2.2 kΩ ao invés de 2 kΩ e resistor de 10 Ω ao invés de 5 Ω.

No caso da Figura 18, o microcontrolador (*microcontroller*) mostrado corresponde ao ESP8266 neste projeto.

São utilizados 5 pinos de entrada analógica e 5 pinos de saída digital do conversor ADS7828. As entradas são para os sensores flex e as saídas (SDA e SCL) serão conectadas ao módulo ESP8266 (nos pinos D1 e D2). Na Figura 19, ilustram-se as disposições das entradas e saídas no conversor.

Figura 19 – Disposições das entradas e saídas no ADS7828 e no ESP8266



Fonte: TEXAS INSTRUMENTS. (2001); modificado pela própria autora. (2017).
Nota:

CH0 – Dedo polegar;
CH1 – Dedo indicador;
CH2 – Dedo médio;
CH3 – Dedo anelar;
CH4 – Dedo mínimo;

Observe que, para as entradas no conversor, é feito um divisor de tensão. Utilizam-se cinco resistores de 10 kΩ e 5 sensores flex.

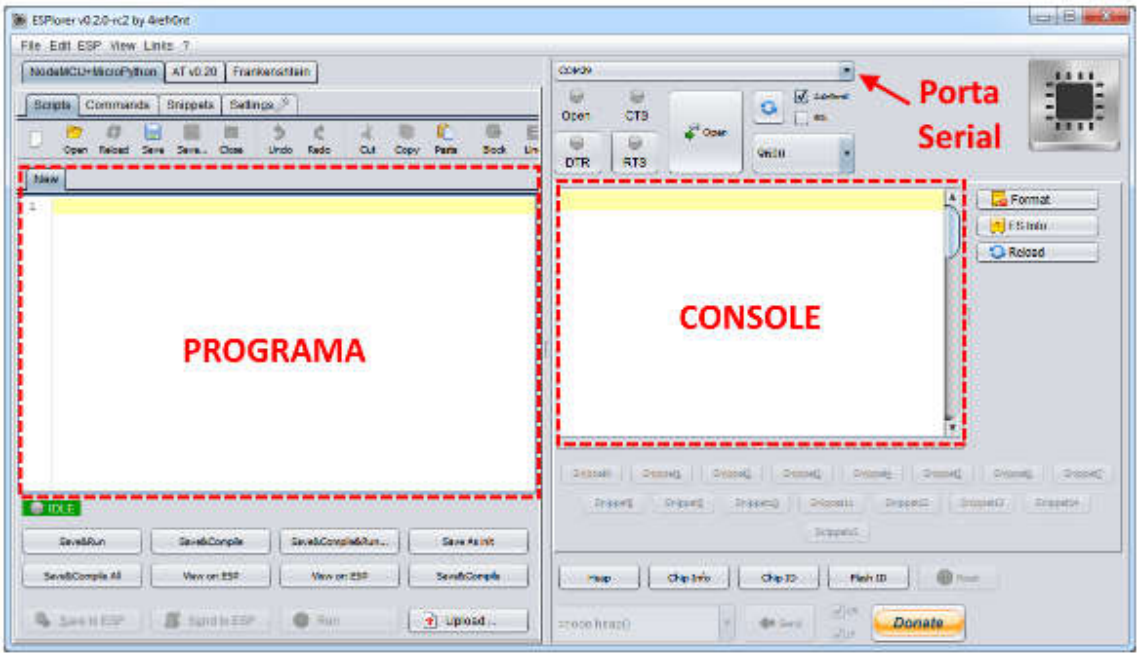
Quanto à comunicação entre o ADS7828 e ESP8266, este assunto está explanado na subseção 2.1.3.2. O protocolo de comunicação é o *I2C*. O dispositivo atuando como mestre é o ESP8266 e o escravo é o ADS7828. Portanto, o ESP8266 controlará o acesso ao barramento.

Foi feito um algoritmo que possibilita a comunicação desses dispositivos. Esse algoritmo encontra-se no Apêndice A e está comentado para melhor compreensão. A linguagem de programação usada foi Lua.

3.4 Módulo Wi-Fi ESP8266

A programação implementada no módulo foi realizada usando o *software* ESPlorer cuja interface é apresentada na Figura 20. O *download* e a explicação para instalação se encontram no site da 14core (14CORE, [2016 ou 2017]).

Figura 20 – Interface do ESPlorer



Fonte: THOMSEN, A. [2016]

Na Figura 20, o espaço onde está escrito “PROGRAMA” é onde se pode escrever o algoritmo para implementação no ESP8266. O espaço onde está escrito “CONSOLE” é onde se pode observar, em tempo real, a comunicação com o módulo, verificando se o algoritmo está sendo executado corretamente (THOMSEN, [2016]).

Foi feito um algoritmo que possibilita o envio de dados do ESP8266 para o computador. Esse algoritmo, implementado em linguagem Lua, encontra-se no Apêndice A e está comentado para melhor compreensão.

3.5 Programa desenvolvido no Visual Studio

No programa Visual Studio, é feita a programação desde a captura dos dados enviados pelo ESP8266 via Wi-Fi até seu processamento, tomando-se valores médios, desvio padrão e limites (inferior e superior) de cada dedo conforme a configuração de mão era solicitada.

As etapas da programação foram:

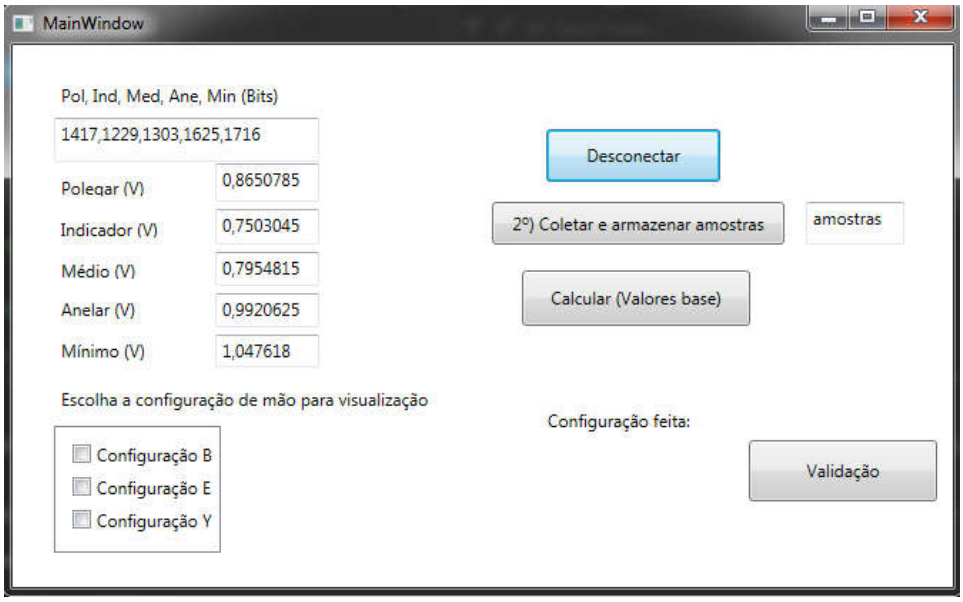
- recepção dos dados enviados via Wi-Fi: neste caso, os dados estão em formato *string*, logo, é necessária uma conversão para *double* para realização dos cálculos desejados;

- conversão de *string* para *double*;
- conversão dos valores de bits para tensão (a demonstração de como isso é feito está no subseção 2.1.3.4);
- cálculos para obtenção dos valores médios, desvio padrão e limites de cada dedo para cada configuração de mão.
- armazenamento desses valores em um BD.

Observação: *string* representa uma cadeia de caracteres e *double* representa um número de 8 bits. São tipos de variáveis utilizadas na programação.

Por fim, a interface criada para a realização do projeto pode ser visualizada na Figura 21.

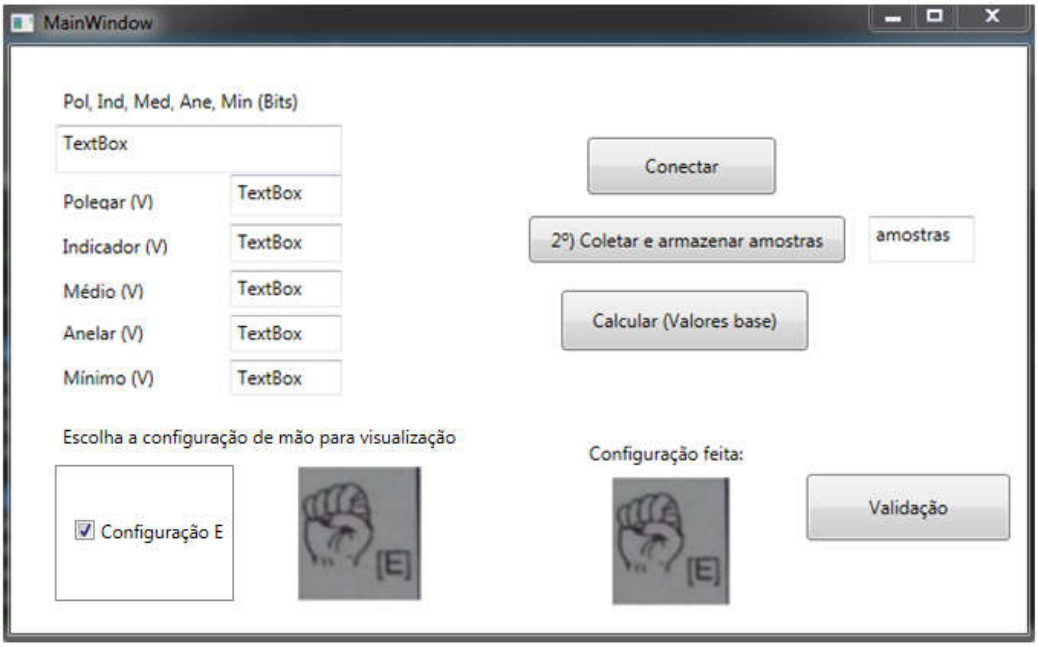
Figura 21 – Interface no Visual Studio



Fonte: própria autora. (2017)

No espaço onde está escrito "Escolha a configuração de mão para visualização", quando o usuário clica em alguma *checkBox* (caixa de checagem), aparece a figura da configuração escolhida para visualização (à esquerda). Quando o botão "Validação" é selecionado, a figura da configuração de mão feita por novo usuário é mostrada (à direita). A Figura 22 mostra isso.

Figura 22 – Interface no Visual Studio (Visualização das configurações de mão)



Fonte: própria autora. (2017)

A seguir uma explanação acerca dos elementos da Figura 21:

- "Pol, Ind, Med, Ane, Min (Bits)": onde são mostrados os valores dos sensores, em bits, enviados via Wi-Fi através do ESP8266.
- Pol – Dedo polegar;
- Ind – Dedo indicador;
- Med – Dedo médio;
- Ane – Dedo anelar;
- Min – Dedo mínimo;
- Os quadros ao lado de "Polegar (V)", "Indicador (V)", "Médio (V)", "Anelar (V)" e "Mínimo (V)" mostram os valores dos sensores flex convertidos para tensão. A explicação teórica de como é realizada esta conversão está na subseção 2.1.3.4, "Relação entre tensão e quantidade de bits".
- O botão onde está escrito "Desconectar" é o que faz a conexão com a rede criada pelo ESP8266. Quando o programa do criado no Visual Studio não está conectado com esta rede, no botão fica escrito "Conectar". Caso contrário, o botão fica escrito "Desconectar", conforme mostrado na Figura 21.

- O botão “2º) Coletar e armazenar amostras” é o segundo passo a ser feito, após o acionamento do botão “Conectar”. Esse botão dispara a captura dos dados coletados introduzindo-os em uma tabela do banco de dados. A cada clique do botão, um dado é armazenado no BD. Esses dados são mostrados nos quadros “Polegar (V)”, “Indicador (V)”, “Médio (V)”, “Anelar (V)” e “Mínimo (V)”.
- Ao lado do botão “2º) Coletar e armazenar amostras”, há uma *textbox* (caixa de texto) cujo nome está como “amostras”. Esta *textBox* mostra a quantidade de amostras coletadas.
- O botão “Calcular (Valores base)” é onde se tem a função na qual são pegas as amostras coletadas e, a partir delas, são feitos os cálculos desejados para obtenção dos valores base, as quais são: valor médio dos dedos, desvio padrão e limites (inferior e superior) para cada configuração de mão e para cada dedo (obtidos através das amostras nos testes realizados nos usuário L, I e S). Estes dados, também, são armazenados em uma tabela no BD.
- O campo “Escolha a configuração de mão para visualização” é onde um novo usuário poderá escolher qual configuração de mão (B, E ou Y) deseja visualizar para fazer conforme sua respectiva figura. Veja Figura 22.
- O botão “Validação” é onde se tem a função na qual são pegas as amostras coletadas e, a partir delas, são feitos os cálculos desejados para obtenção dos variáveis a serem analisadas, as quais são: valor médio dos dedos, desvio padrão e limites (inferior e superior) para cada configuração de mão e para cada dedo (obtidos através das amostras nos testes realizados por novo usuário). Estes dados, também, são armazenados em uma tabela no BD. Este botão é usado para verificar qual configuração de mão o novo usuário fez. No caso apresentado na Figura 22, foi analisada qual configuração de mão que o novo usuário “NU_1” fez. No teste realizado, este usuário fez a configuração E. A interface apresentou corretamente a configuração que este fez, conforme mostrado na Figura 22.
- Abaixo da *label* (rótulo) “Configuração feita”, é mostrada a configuração de mão feita pelo novo usuário após clique no botão “Validação”. Veja Figura 22.

Observação: os dados obtidos através dos testes realizados nos usuários L, I e S são usados como referência para as configurações de mãos propostas (B, E e Y) para posterior comparação com os dados obtidos com qualquer novo usuário que usar a luva.

A validação é realizada no algoritmo construído no Visual Studio. Para isso, calcula-se a diferença entre os valores médios de cada dedo do novo usuário em relação aos valores base. Depois, é feita uma comparação entre as diferenças entre os valores médios obtidos de cada dedo para cada configuração de mão. Utilizam-se contadores para sinalizar a menor diferença. O maior contador é o que apontará qual configuração de mão o novo usuário fez. A seguir, uma parte do código como exemplo para melhor compreensão de como a validação é feita:

```
difPol_B = Math.Abs(vmPol_B - configMedia_Pol);
if (difPol_B < difPol_E && difPol_B < difPol_Y) { contB++; }
else if (difPol_E < difPol_B && difPol_E < difPol_Y) { contE++; }
else if (difPol_Y < difPol_B && difPol_Y < difPol_E) { contY++; }
```

Sendo:

- difPol_B: diferença dos valores médios do dedo polegar da configuração B;
- difPol_E: diferença dos valores médios do dedo polegar da configuração E;
- difPol_Y: diferença dos valores médios do dedo polegar da configuração Y;
- Math.Abs: Valor absoluto;
- vmPol_B: valor médio base do dedo polegar da configuração B (obtido através das amostras nos testes realizados nos usuário L, I e S). Na subseção 2.2.2, explica-se como é feito o cálculo para obtenção do valor médio;
- configMedia_Pol: valor médio do dedo polegar do novo usuário;
- contB: contador para configuração B;
- contE: contador para configuração E;
- contY: contador para configuração.

A lógica para o cálculo das demais diferenças entre os valores médios é a mesma.

Observe que, caso difPol_B for menor que difPol_E e difPol_Y, então, contB é incrementado. Caso contrário, verificam-se os demais. Esses cálculos e verificações são feitas para todos os dedos e configurações de mão.

Por fim, o contador que for maior ou igual a 3, será considerado a configuração de mão realizada pelo novo usuário. Por exemplo, se contB for maior ou igual a 3, então, conclui-se que a configuração de mão realizada pelo novo usuário será a configuração B.

Para visualização dessa lógica empregada, veja o algoritmo que se encontra no Apêndice B.

Observação: para todo novo usuário é necessária criação de novas tabelas no BD (a lógica para criação está apresentada no Apêndice C). É modificações no início do código do Visual: colocação do nome da tabela criada no código (o local está evidenciado no próprio código apresentado no Apêndice B).

3.6 Banco de dados MySQL

Conforme relatado no capítulo 2, subseção 2.2.1, "*Softwares* utilizados", foi criado um BD para armazenamento dos dados obtidos nos testes. O algoritmo feito no Visual Studio insere os dados calculados no banco de dados MySQL criado e, também, trabalha com esses valores, pegando-os nas tabelas do banco para realizarem os cálculos desejados (explicação de como é feito os cálculos e a lógica para essa finalidade no programa estão nos itens 4.5, 5.6 e no Apêndice B, no qual o código é apresentado). O BD é usado para armazenamento dos dados para posterior análise destes. Logo, um algoritmo foi feito para essa finalidade. Este se encontra no Apêndice C.

4. ALOCAÇÃO DE RECURSOS

A Tabela 1 apresenta uma descrição dos equipamentos adquiridos para a construção do protótipo:

Tabela 1 – Equipamentos utilizados, especificações e preço

Quantidade (Unidade)	**Dispositivo**	**Descrição**	**Preço (R$)**
1	Luva	Ciclismo	1 par: 60,00
5	Sensor flex	4,5 polegadas	*40,53/unidade
1	Módulo Wi-Fi ESP8266	Módulo NodeMcu ESP-12E.	*49,90
1	Computador	Acer Aspire 5750Z-4633.	*
1	ADS7828	Conversor A/D, de baixa potência. Possui 8 canais.	1 unidade: 23,00
1	Resistor de 10 Ω	Potência de ¼ W.	*2,00
2	Resistor 2.2 kΩ	Potência de ¼ W.	*2,00/unidade
10	Resistor 10 kΩ	Potência de ¼ W.	*2,00/unidade
1	Capacitor 0.1 µF		*3,00 (5 unidades)
2	Capacitor 1 µF		*0,17/unidade

Fonte: própria autora. (2016).
Nota: os quadros destacados com um asterisco (*) referem-se a elementos que foram fornecidos por alunos e por um professor do departamento de engenharia elétrica da UFES ou já pertencentes à própria autora antes da realização do projeto.

Acerca desses elementos usados no projeto, como o objetivo é a detecção de determinadas configurações de mão da Libras, é necessária apenas uma luva. Pois, a configuração de mão da Língua Brasileira de Sinais pode ser feito apenas com uma mão. A Figura 23 apresenta uma foto da luva usada no projeto, na qual foi feito alguns ajuste feitos (costura em azul) para acoplamento dos sensores flex.

Figura 23 – Luva utilizada no projeto

Fonte: própria autora. (2017).

Os sensores flex são manufaturados pela empresa Spectra Symbol. A especificação descrita no site Digi-Key Eletronics, uma das revedendoras deste sensor, mostra que o sensor flex de 4.5 polegadas possui resistência de 10 kΩ (resistência plana a 180°), uma tolerância da resistência de ± 30%, potência de 0.5 W, largura de 6,35 mm, altura 0.43mm, entre outras especificações técnicas (DIGI-KEY, [20--]).

Foram utilizadas algumas resistências de valores aproximados dos apresentados na Figura 18, como: resistores de 2.2 kΩ ao invés de 2 kΩ e resistor de 10 Ω ao invés de 5 Ω. Isto não provoca nenhuma alteração significativa para a aplicação do projeto.

5. RESULTADOS OBTIDOS

Os dados calculados foram armazenados em tabelas no BD MySQL. Os valores armazenados no BD são os valores de referência, chamados também de valores base (Configuração B, E e Y), obtidos através dos testes realizados nos usuários I, L e S, e os valores dos novos usuários para cada configuração de mão que estes fizeram. No caso dos testes realizados, os novos usuários são "NU_1" e "NU_2". As Tabelas de 2 a 5 apresentam os valores calculados através dos testes realizados para o caso do dedo polegar. A forma como foram calculados os valores apresentados nessas tabelas está explanada na subseção 2.2.2.

Tabela 2 – Valor médio de cada dedo em relação às configurações e usuários

	$\bar{x}_{pol}$	$\bar{x}_{ind}$	$\bar{x}_{med}$	$\bar{x}_{ane}$	$\bar{x}_{min}$
Configuração B	0,94823877	0,99823872	0,86431537	0,85348917	0,91173087
Configuração E	1,1232793	1,26669592	1,22272975	1,2615779	1,19440255
Configuração Y	0,97601652	1,21478307	1,15149457	1,18650675	0,94873735
NU_1 B	0,94429087	0,99651915	0,8589735	0,84972442	0,8770443
NU_1 E	0,99310035	1,27078627	1,2538449	1,2109878	1,1536008
NU_1 Y	0,99273405	1,21217827	1,18800247	1,21767277	0,96022492
NU_2B	0,94654972	1,01785612	0,91654365	0,89514562	0,87783795
NU_2E	1,05509662	1,24035285	1,22826495	1,21968742	1,1494494
NU_2Y	0,97200757	1,20335655	1,13940667	1,1461527	1,00692817

Fonte: própria autora. (2017).

Tabela 3 – Desvio padrão de cada dedo em relação às configurações e usuários

	σ_{pol}	σ_{ind}	σ_{med}	σ_{ane}	σ_{min}
Configuração B	0,17220552	0,01942058	0,03480539	0,03748796	0,07046108
Configuração E	0,08782935	0,03112089	0,02985196	0,03771609	0,0869278
Configuração Y	0,05701735	0,01969954	0,02152501	0,03778916	0,04565162
NU_1 B	0,10889254	0,01248334	0,10857069	0,01389377	0,02864565
NU_1 E	0,09234689	0,01821754	0,02099853	0,02735653	0,0432962
NU_1 Y	0,08577707	0,02357468	0,01378902	0,02536219	0,03546068
NU_2B	0,10080988	0,01119377	0,09651271	0,01348344	0,12366534
NU_2E	0,12972297	0,01295799	0,02164986	0,02638834	0,04791417
NU_2Y	0,08277944	0,02589981	0,03713038	0,02060145	0,04702802

Fonte: própria autora. (2017).

Tabela 4 – Limite inferior de cada dedo em relação às configurações e usuários

	li_{pol}	li_{ind}	li_{med}	li_{ane}	li_{min}
Configuração B	0,77603325	0,97881814	0,82950998	0,81600121	0,84126978
Configuração E	1,03544994	1,23557502	1,19287778	1,2238618	1,10747474
Configuração Y	0,91899917	1,19508353	1,12996955	1,14871758	0,90308572
NU_1 B	0,83539833	0,9840358	0,7504028	0,83583065	0,84839864
NU_1 E	0,90075345	1,25256873	1,23284636	1,18363126	1,11030459
NU_1 Y	0,90695697	1,18860358	1,17421345	1,19231057	0,92476424
NU_2B	0,84573983	1,00666235	0,82003093	0,88166217	0,7541726
NU_2E	0,92537365	1,22739485	1,20661508	1,19329907	1,10153522
NU_2Y	0,88922812	1,17745673	1,10227629	1,12555124	0,95990014

Fonte: própria autora. (2017).

Tabela 5 – Limite superior de cada dedo em relação às configurações e usuários

	ls_{pol}	ls_{ind}	ls_{med}	ls_{ane}	ls_{min}
Configuração B	1,12044429	1,0176593	0,89912076	0,89097713	0,98219196
Configuração E	1,21110865	1,0176593	1,25258171	1,29929399	1,28133035
Configuração Y	1,03303387	1,23448261	1,17301959	1,22429591	0,99438897
NU_1 B	1,05318341	1,00900249	0,96754419	0,86361819	0,90568995
NU_1 E	1,08544724	1,28900381	1,27484343	1,23834433	1,196897
NU_1 Y	1,07851112	1,23575296	1,20179149	1,24303497	0,9956856
NU_2B	1,04735961	1,02904989	1,01305636	0,90862907	1,00150329
NU_2E	1,18481959	1,25331084	1,24991481	1,24607577	1,19736357
NU_2Y	1,05478702	1,22925636	1,17653705	1,16675415	1,0539562

Fonte: própria autora. (2017).

As variáveis mostradas nas Tabelas 2 a 5 são:

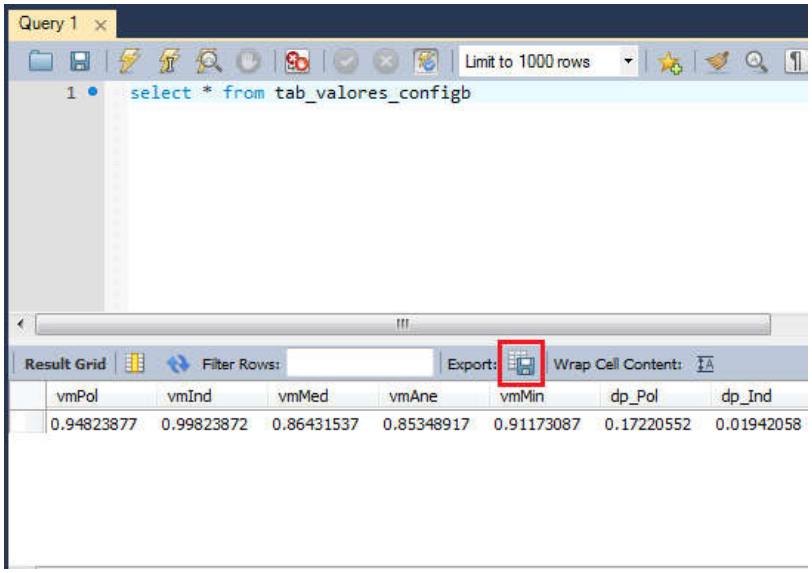
- $\bar{x}$: valor médio;
- σ: desvio padrão;
- li: limite inferior;
- ls: limite superior;
- Subscrtio pol, ind, med, ane e min: polegar, indicador, médio, anelar e mínimo, respectivamente;
- Configuração B: valores base correspondentes à configuração B;
- Configuração E: valores base correspondentes à configuração E;
- Configuração Y: valores base correspondentes à configuração Y;
- NU_1 B: valores correspondentes ao novo usuário 1 quando este realizou a configuração B;

- NU_1 E: valores correspondentes ao novo usuário 1 quando este realizou a configuração E;
- NU_1 Y: valores correspondentes ao novo usuário 1 quando este realizou a configuração Y;
- NU_2 B: valores correspondentes ao novo usuário 2 quando este realizou a configuração B;
- NU_2 E: valores correspondentes ao novo usuário 2 quando este realizou a configuração E;
- NU_2 Y: valores correspondentes ao novo usuário 2 quando este realizou a configuração Y.

As Tabelas de 2 a 5 foram obtidas após a execução do programa Visual Studio em relação ao algoritmo construído para obtenção dessas variáveis. Esse algoritmo está no Apêndice B. Esses dados estão armazenados em tabelas no banco de dados MySQL. Para aquisição das tabelas, é necessária apenas a importação dos dados para um arquivo csv (o próprio BD fornece esta opção) e do arquivo csv para Excel (CASSIUSMG, 2010).

A Figura 24 mostra o ícone (circundado em vermelho) que possibilita fazer a importação dos dados da tabela do BD criada para um arquivo csv.

Figura 24 – Importação dos dados para um arquivo csv



Fonte: própria autora. (2017).

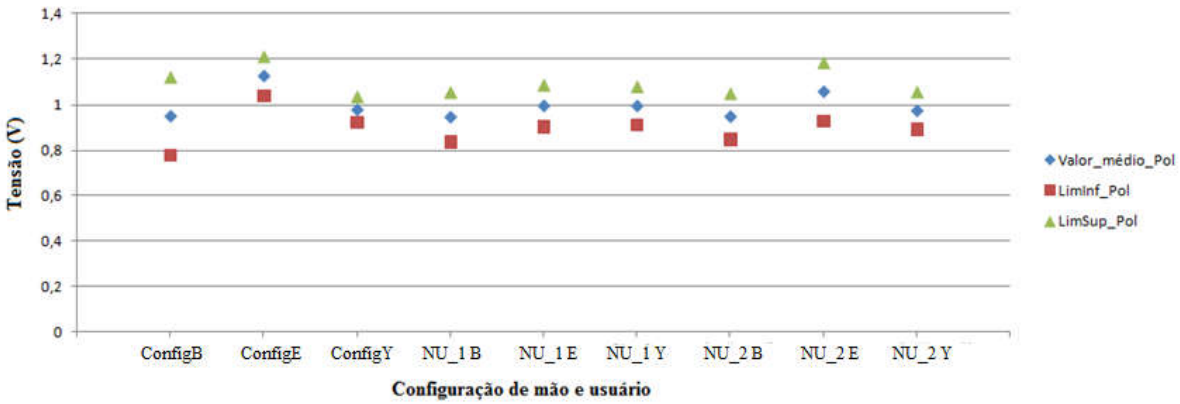
Uma observação importante é que NU_1 é o segundo teste feito por este usuário, pois, NU_1 também é o usuário I (para obtenção dos valores base). Foram realizados novos testes

com esse usuário para verificação da correta validação. O usuário NU_1 fez uma das três configurações de mão analisadas. Após a coleta das amostras de cada dedo desse usuário, foi feita a validação. Importante salientar essas validações foram corretas, ou seja, o novo usuário vestiu a luva, fez uma configuração de mão, coletou-se os dados, depois, foi feita a validação e o programa indicou a configuração de mão que o novo usuário realmente fez.

Na subseção 2.2.2, mostra a lógica dos cálculos realizados para obtenção destas variáveis. No Apêndice B, está a aplicação dessa lógica.

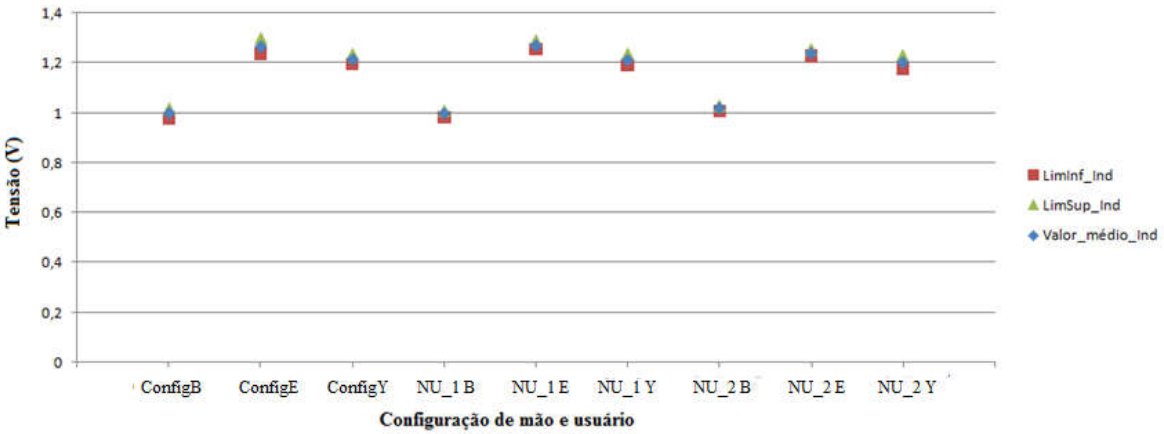
Os Gráficos de 1 a 5 foram obtidos com base nos valores adquiridos e calculados.

Gráfico 1 – Tensão x Configuração de mão e usuário (Dedo polegar)



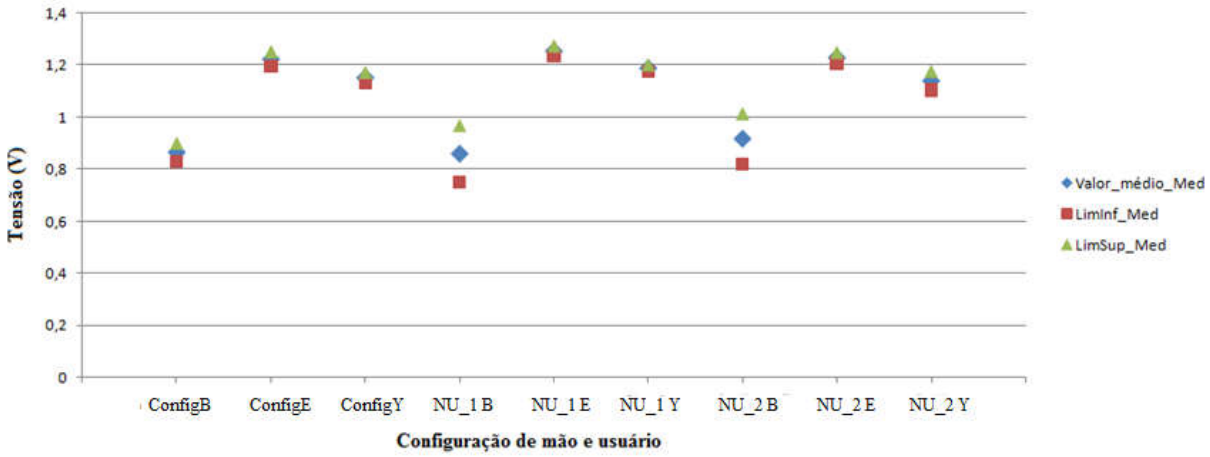
Fonte: própria autora. (2017).

Gráfico 2 – Tensão x Configuração de mão e usuário (Dedo indicador)



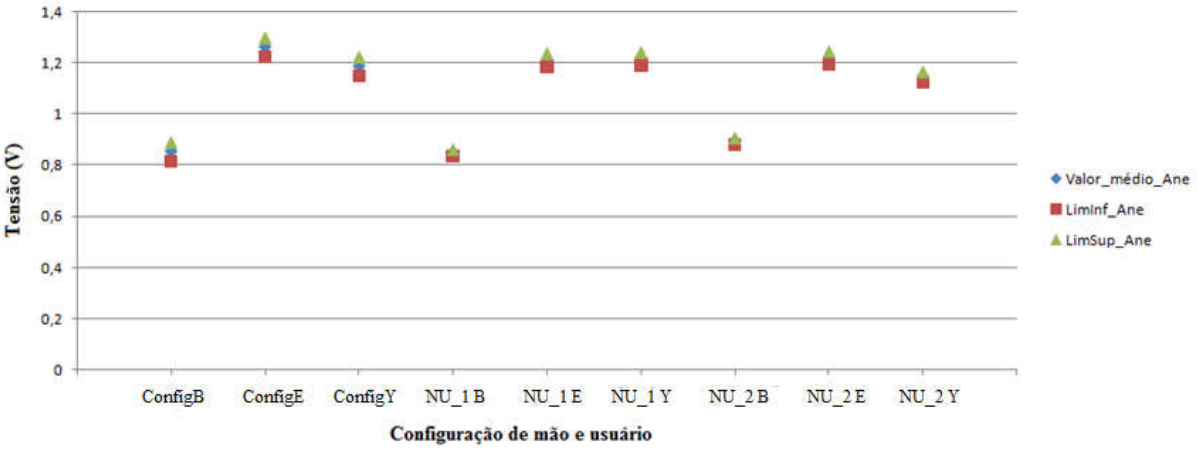
Fonte: própria autora. (2017).

Gráfico 3 – Tensão x Configuração de mão e usuário (Dedo médio)



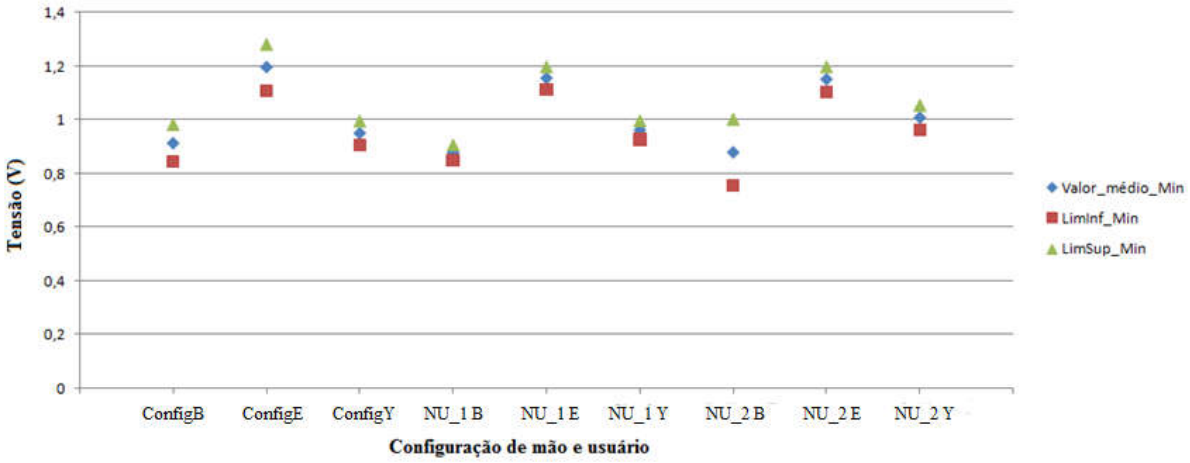
Fonte: própria autora. (2017).

Gráfico 4 – Tensão x Configuração de mão e usuário (Dedo anelar)



Fonte: própria autora. (2017).

Gráfico 5 – Tensão x Configuração de mão e usuário (Dedo mínimo)

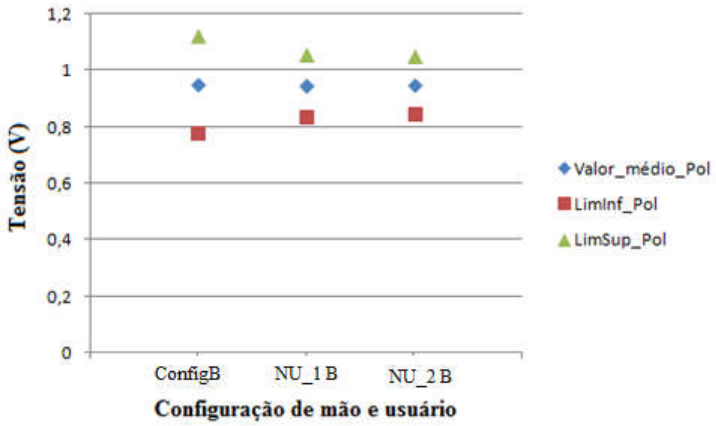


Fonte: própria autora. (2017).

Com base nesses gráficos, pode-se analisar se os valores dos novos usuários (“NU_1” e “NU_2”) para cada configuração de mão e cada dedo estão dentro dos valores base, os quais são representados por “Configuração B”, “Configuração E” e “Configuração Y”.

Dos Gráficos de 1 a 5, observa-se que os valores dos novos usuário (“NU_1” e “NU_2”) estão dentro da faixa de valores dos “valores base”. Por exemplo, comparando-se o valor do dedo polegar para configuração B dos novos usuários com relação à “Configuração B”, observa-se que esses valores estão dentro dos valores base. O Gráfico 6 apresenta os mesmos valores que o Gráfico 1, contudo, apenas para a configuração B:

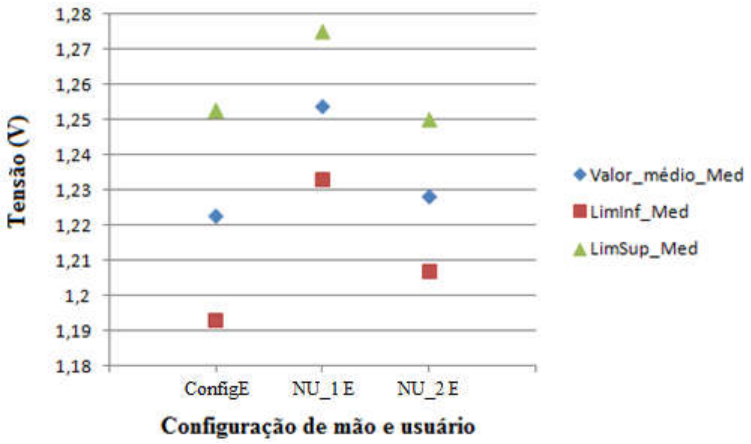
Gráfico 6 – Tensão x Configuração de mão B e usuário (Dedo polegar)



Fonte: própria autora. (2017).

O Gráfico 7 apresenta o resultado de se fazer esta mesma análise, mas agora para outra configuração de mão e outro dedo.

Gráfico 7 – Tensão x Configuração de mão E e usuário (Dedo médio)

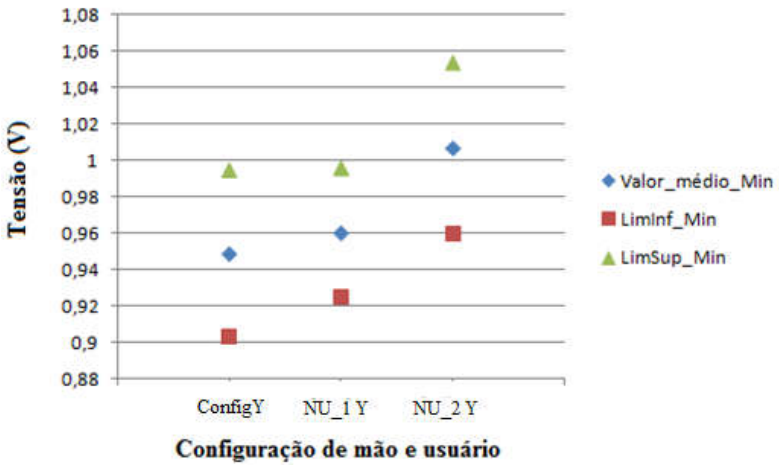


Fonte: própria autora. (2017).

Note que, os valores de "NU_1 E" e "NU_2 E" estão dentro dos limites de "ConfigE".

Por fim, tomando um exemplo para a análise quanto à configuração Y e dedo mínimo, os resultados obtidos estão presentes no Gráfico 8.

Gráfico 8 – Tensão x Configuração de mão Y e usuário (Dedo mínimo)



Fonte: própria autora. (2017).

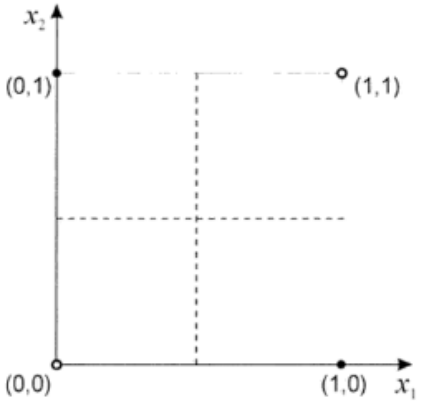
Observe que, da mesma forma quanto aos Gráficos 6 e 7, o Gráfico 8 mostra que os valores dos novos usuários estão dentro da configuração de mão e dedo determinados.

É importante ressaltar ainda a análise dos dados para a validação, pois pode haver valores que não são muito distintos entre si. Por exemplo, o valor médio do dedo indicador de um novo usuário pode estar dentro da faixa do valor base que representa o dedo indicador da

configuração B e da configuração Y (observe o Gráfico 5). Assim, tem-se como análise o problema do OU-Exclusivo e Representação em alta dimensão nesse caso. Logo, para uma qualificação ainda mais precisa de qual configuração de mão um novo usuário estaria fazendo, tendo como referência os valores base, seria necessária uma representação em alta dimensão.

Segundo (MESQUITA e PIMENTA, 2013, p.1), "o problema do OU-Exclusivo (XOR) pertence à classe de problemas que envolvem padrões não linearmente separáveis.". Na Figura 25, pode-se visualizar isso para a exemplificação do assunto.

Figura 25 – Visualização do problema OU-Exclusivo



Fonte: REZENDE, S. (2005).

Observe que não há uma reta que faça a separação entre os conjuntos de dados, um conjunto representado pelo círculo não hachurado (nas coordenadas: (0,0) e (1,1)) e outro representado pelo círculo hachurado (nas coordenadas: (0,1) e (1,0)).

Segundo (Rezende, 2015, p.207), a estratégia de solução para esse tipo de problema "consiste em transformar o espaço de entrada em outro espaço de maior dimensão, por meio de uma transformação não-linear, de forma que o problema torne-se linearmente separável nesse novo espaço.".

Assim, de modo a melhorar a qualificação das configurações realizadas e no intuito de aumentar a qualidade de configurações detectadas, é necessária a inclusão de uma nova medida. Isto pode ser feito introduzindo-se um novo sensor para a coleta de dados da configuração de mão, por exemplo, sensores de contato, os quais verificariam se os dedos estão encostando entre si e quais seriam esses dedos.

O que se deseja mostrar com este exemplo é que tendo mais variáveis a serem medidas, mais fidedigna é a qualificação da configuração de mão. Contudo, surge o aumento da complexidade e do custo do projeto.

Com a representação em alta dimensão, pode-se fazer mais distinção entre os valores a serem analisados para a validação da configuração de mão realizada. Portanto, isso pode ser inserido como um aperfeiçoamento para este projeto.

Essa dicotomia entre aumentar a dimensão dos dados de entrada para melhorar os índices de acerto de configuração com o indesejado aumento da complexidade (que pode impactar no custo computacional) é conhecida na literatura como maldição da dimensionalidade (DUDA; HART, 2000).

6. CONCLUSÃO E DISCUSSÕES FINAIS

6.1 Conclusões finais

No presente projeto, desenvolveu-se uma luva sensoreada que traduz configuração de mãos em sinais elétricos. O projeto consta de duas partes: a primeira consiste no desenvolvimento da eletrônica e dos programas, tanto embarcados como residentes em um computador, e a segunda reside na análise nos dados coletados para fins de validação.

Os sinais elétricos oriundos de uma configuração de mão são capturados por cinco canais de um conversor A/D. Após a digitalização dos sinais, os dados obtidos são enviados para um módulo Wi-Fi ESP8266, o qual os envia para o computador, conectado à rede criado no ESP8266, para processamento das informações através das ferramentas computacionais. Nesse interim, utilizou-se Visual Studio (manipulação dos dados) e MySQL (armazenamento dos dados em um BD). Para os testes, selecionou-se três configurações de mão da Língua Brasileira de Sinais.

Dos resultados apresentados no capítulo 5, verifica-se que a luva funciona conforme desejado, apresentando resultados satisfatórios para as três configurações de mão escolhidas. Pois, todas as validações feitas, em relação aos testes com novos usuários, foram corretas, ou seja, é identificado corretamente qual a configuração de mão (B, E ou Y) o novo usuário fez. Pode-se observar também que todos os dados dos novos usuários (NU_1 e NU_2) estão dentro da faixa dos valores base correspondentes (veja os gráficos de 1 a 8).

A forma como foram realizados os testes com os novos usuários seguiu a metodologia implementada e descrita no capítulo 3 (coleta de 20 amostras, 20 repetições feitas quanto à configuração de mão escolhida dentre as três propostas, e armazenamento no BD para cada dedo quanto à cada configuração de mão que o novo usuário fez). Quando foi feita a validação, todos esses testes realizados com os novos usuários, apresentaram o resultado correto correspondente à configuração de mão. Ou seja, quando o novo usuário vestiu a luva e fez a configuração Y, por exemplo, a interface apresentou o resultado correto (a figura que representa a configuração Y), mostrando que a metodologia implementada corresponde corretamente à finalidade da implementação da luva: a detecção de determinadas configurações de mão da Libras. É importante salientar que se usou como referência os valores base obtidos através dos dados coletados de três usuários, aqui designados por I, L e S.

6.2 Trabalho futuros

6.2.1 Biblioteca de valores de cada configuração de mão

Com os resultados obtidos e a inclusão destes em um banco de dados, pode-se obter uma "biblioteca de valores para cada configuração de mão". Ou seja, pode-se obter valores base das demais configurações de mão apresentadas por Quadros e Karnopp (2004, p. 53) e mostradas na Figura 16. Assim, utilizando-se da mesma metodologia apresentada (capítulo 3) e realizando testes com usuários I, L e S, é possível coletar as amostras, realizar cálculos e armazenar as amostras e as variáveis calculadas no BD. Entretanto, é importante salientar que, para isso, é necessário modificações no algoritmo feito no Visual Studio, além da adição de novas tabelas no BD para armazenar esses novos dados. Contudo, a lógica implementada pode permanecer, ou seja, para a adição de novos parâmetros de referência (valores base para outras configurações de mão), pode-se seguir a lógica implementada para a configuração B, E e Y, mas serão necessários mais testes para sua avaliação. Isso possibilitará uma forma de aprendizagem, por exemplo, para um usuário da luva que esteja aprendendo determinada configuração de mão da Libras.

6.2.2 Mobilidade do protótipo

Para maior mobilidade, o circuito ilustrado na Figura 19 pode ser confeccionado em uma placa de circuito impresso. O circuito pode ser colocado na base superior da luva. A fonte de alimentação pode ser uma bateria de 3.3 V.

6.2.3 Implementação da lógica Fuzzy

Para validação dos dados de qualquer novo usuário da luva, pode-se utilizar uma abordagem baseada na lógica Fuzzy. Lógica Fuzzy é geralmente empregada para modelar sistemas com imprecisão e/ou sistemas não lineares complexos (FERREIRA, 2009, p. 5).

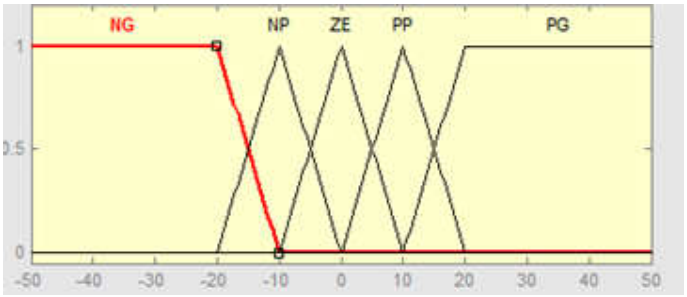
Resumidamente, a lógica Fuzzy é composta por dados de entrada, saída e um banco de regras (definidas pelo projetista do sistema). Os sinais capturados nas entradas deixam de ser tratados como na álgebra de Boole e passam a ter mensurados suas pertinências (variando de 0 a 1) a conjuntos que exprimem a incerteza das entradas, conjuntos esses construídos baseadas na expertise do projetista do sistema. Baseado nas regras fuzzy e um motor

(mecanismo) de inferência, o sistema gera, na sua saída, o processamento dos dados pelas regras fuzzy. Como Fuzzy pode ser considerado como um aproximador universal de funções, assim como as redes neurais (HAYKIN, 2000), para o nosso caso, é possível criar conjuntos fuzzy e regras para classificar as configurações de mãos.

O sistema Fuzzy pode ser usado para implementar uma representação lógica para analisar os dados que procedem de um novo usuário, podendo determinar qual é a configuração de mão realizada por esse novo usuário. A lógica construída no sistema Fuzzy será baseada nos dados coletados nos testes realizados nos usuário L, I e S. Estes valores chamados como valores base para a implementação da lógica Fuzzy.

A Figura 26 é apresentada para exemplificar a aplicação da lógica Fuzzy para a validação da configuração de mão para novo usuário.

Figura 26 – Exemplo para visualização da validação da configuração de mão usando lógica Fuzzy



Fonte: COLLI, S. MATTOS, J. (2016).

Considerando que essa figura mostra um gráfico da configuração B e que NG representa dedo polegar, NP dedo indicar, ZE dedo médio, PP dedo anelar e PF dedo polegar, pode-se criar uma lógica Fuzzy (um banco de regras) para verificar se os dados dos do novo usuário se enquadram aos dados da configuração B e, também, para as demais configurações.

6.2.4 O tradutor de Libras para português

Sabe-se que um sinal da Libras é composto por 5 parâmetros, já discutido no capítulo 1. Este projeto focou-se na identificação de um desses parâmetros: a configuração de mãos. Logo, para trabalhos futuros, pode-se desenvolver tecnologias para identificar os demais parâmetros da Língua Brasileira de Sinais, podendo-se criar um tradutor de Libras para português. Os parâmetros e o que é necessário identificar estão dispostos na Tabela 6:

Tabela 6 – Relação dos parâmetros da Libras e o que identificar

Parâmetros da Libras	**O que identificar**
Configuração de mãos	Já proposto e explanado neste trabalho.
Movimento	O movimento da mão e braço.
Localização	Onde, no espaço, a mão está localizada. Exemplo: no rosto, em frente ao corpo, etc.
Orientação de mão	Para onde a palma da mão aponta. Exemplo: para cima, para baixo, etc.
Expressões não-manuais	Expressão da face. Exemplo: expressão de tristeza, de alegria, etc.

Fonte: própria autora. (2017).

Portanto, compondo o desenvolvimento de tecnologias que identifiquem os parâmentro apresentados na Tabela 6, pode-se construir um tradutor de Libras para o português, possivelmente. Esse tipo de tecnologia pode melhorar a acessibilidade de pessoas com deficiência auditiva. Também pode melhorar a comunicação entre ouvinte e surdo, utilizando esse tipo de ferramenta. Além de também ser um instrumento de aprendizagem para as pessoas que desejam estudar LS.

7. REFERÊNCIAS BIBLIOGRÁFICAS

14CORE. *How to install ESPlorer IDE in multiple platform*. [S.l.]. [2016 ou 2017]. Disponível em: <http://www.14core.com/how-to-install-esplorer-ide-in-multiple-platform/>. Acesso em: 15 de fevereiro de 2017.

BRASIL. Lei nº 10.436, de 24 de abril de 2002. Brasília. 24 de abril de 2002. Disponível em: <http://pesquisa.in.gov.br/imprensa/jsp/visualiza/index.jsp?jornal=1&pagina=23&data=25/04/2002>. Acesso em: 27 de maio de 2016.

CASSIUSMG. Transformar CSV em XLS. [S.l.]. 20 de outubro de 2010. Disponível em: <https://social.msdn.microsoft.com/Forums/pt-BR/bccf43ec-a754-49b1-b793-2d0819802f47/transformar-csv-em-xls?forum=excelpt>. Acesso em: 02 de julho de 2017.

COLLI, S. MATTOS, J. Controlador Fuzzy para ponte rolante. Vitória: [s.n.]. 2016.

D'ANGELO, H. Luvas inteligentes traduzem linguagem de sinais em tempo real. Disponível em: <http://super.abril.com.br/tecnologia/luvas-inteligentes-traduzem-linguagem-de-sinais-em-tempo-real>. Atualizado em: 26 de abril de 2016. Acesso em: 27 de maio de 2016.

DIGI-KEY ELETRONICS. Spectra Symbol FS-L-0095-103-ST. [S.l.]. [20--]. Disponível em: <https://www.digikey.com/product-detail/en/spectra-symbol/FS-L-0095-103-ST/905-1000-ND/2175377>. Acesso em: 05 de abril de 2017.

DUDA, O. R; HART, P. Pattern Classification. 2ª ed. [S.l.]: Ed. Wiley, 2000.

FERREIRA, E. Controle Inteligente & Fundamentos e Aplicações da Lógica Fuzzy ao Controle de Sistemas. Vitória: [s.n.]. 2009.

FERREIRA-BRITO, L.; LAGEVIN, R. Sistema Ferreira Brito-Langevin de Transcrição de Sinais. In: FERREIRA-BRITO, L. *Por uma gramática de língua de sinais*. Rio de Janeiro: Tempo Brasileiro. 1995.

HAYKIN, S. Neural Networks and Learning Machines. 2ª ed. [S.l.]: Ed. Prentice Hall, 2000.

IGREJA CRISTÃ MARANATA. Libras: Apostila do professor. [20--]. Apostila da oficina de Libras da Igreja Cristã Maranata. Presbitério Espírito Santense. Espírito Santo.

KADAM, K. et al. *American Sign Language Interpreter*. IEEE *Fourth International Conference on Technology for Education*, Haiderabade, p.157-159, jul. 2012.

LEAL, J. Conectando C# a um Banco de Dados MySql. Youtube, 9 de janeiro de 2017. Disponível em: <https://www.youtube.com/watch?v=ldY7B9CRFuc>. Acesso em: 02 de julho de 2017

MESQUITA, M. E.; PIMENTA, M. A. Perceptrons Morfológico de Camada Única. Instituto de Matemática, Estatística e Computação Científica, Unicamp, 09 de maio de 2013. Disponível em: <http://www.ime.unicamp.br/~valle/PDFfiles/SLMP_report.pdf>. Acesso em: 07 de julho de 2017.

MINISTÉRIO DA EDUCAÇÃO. Prolibras. [S.l.]. [2016 ou 2017]. Disponível em: <http://portal.mec.gov.br/pnaes/194-secretarias-112877938/secad-educacao-continuada-223369541/17436-prolibras-programa-nacional-para-a-certificacao-de-proficiencia-no-uso-e-ensino-da-lingua-brasileira-de-sinais-libras-e-para-a-certificacao-de-proficiencia-em-traducao-e-interpretacao-da-libraslingua-portuguesa-novo>. Acesso em: 13 de julho de 2017.

QUADROS, R; KARNOPP, L. Língua de sinais brasileira: Estudos linguísticos. 1. ed. Rio Grande do Sul: Ed. Artmed Editora S.A. 2004.

REIS, V. *I2C* – Protocolo de Comunicação. [S.l]. 17 de dezembro de 2014. Disponível em: <http://www.arduinobr.com/arduino/i2c-protocolo-de-comunicacao/>. Acesso em: 01 de março de 2017.

REZENDE, S. Sistemas Inteligente: Fundamentos e Aplicações. 1. ed. São Paulo: Ed. Manole Ltda. 2005.

ROBOCORE. ESP8266 #1 / Crie um WebServer sem Arduino. Produção de Robocore. Edição de Luke Morales. Youtube, 27 de agosto de 2015. Disponível em: <https://www.youtube.com/watch?v=1jA9pFYxPQ0&t=21s>. Acesso em: 08 de fevereiro de 2017.

SPECTRA SYMBOL. *Flex sensor*. [S.l.]. [S.n]. [20--?]. Disponível em: <https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf>. Acesso em: 04 de agosto de 2017.

SWEE, T. T. et al. *Wireless Data Gloves Malay Sign Language Recognition System*. IEEE *6th International Conference on Information, Communications & Signal Processing*, Singapura, p. 1-4, dez. 2007.

TEDESCHI, L. Entenda o que é *firmware*. [S.l.]. 09 de julho de 2011. Disponível em: <http://manualdatecnologia.com/anuncios-mt/entenda-o-que-e-firmware/>. Acesso em: 14 de fevereiro de 2017.

TEXAS INSTRUMENTS. *Datasheet ADS 7828: 12-Bit, 8-Channel ampling ANALOG-TO-DIGITAL CONVERTER with I2C™ Interface*. Novembro de 2001. Revisado em março de 2005.

THOMSEN, A. Como programar o módulo ESP8266 NodeMCU. 2016. [S.l.]. [2016]. Disponível em: <http://blog.filipeflop.com/wireless/esp8266-nodemcu-como-programar.html>. Acesso em: 14 de fevereiro de 2017.

UOL. Brasil tem mais de 204 milhões de habitantes, diz IBGE. Rio de Janeiro. 28 de agosto de 2015. Disponível em: <http://noticias.uol.com.br/cotidiano/ultimas-noticias/2015/08/28/brasil-tem-mais-de-204-milhoes-de-habitantes-diz-ibge.htm>. Acesso em: 27 de maio de 2016.

VILLELA, F. IBGE: 6,2% da população têm algum tipo de deficiência. Rio de Janeiro. 21 de agosto de 2015. Disponível em: <http://agenciabrasil.ebc.com.br/geral/noticia/2015-08/ibge-62-da-populacao-tem-algum-tipo-de-deficiencia>. Acesso em: 27 de maio de 2016.

APÊNDICE A

Mais adiante, mostra-se o algoritmo feito e implementado no ESP8266 para a realização da obtenção dos sinais analógicos dos sensores flex, convertidos pelo ADS7828 e enviados para o computador.

Este algoritmo foi feito com o auxílio dos colegas do departamento da engenheira elétrica da UFES Letícia Munhoz e Anderson Borba.

Observação para quem for utilizar este código: caso deseje salvá-lo no módulo ESP8266, é necessário tirar os comentários. Pois, ocorre erro ao salvar o código no ESP8266. O salvamento correto do código no ESP8266 ocorre quando grande parte dos comentários é retirada.

Segue o algoritmo:

```
----- Comuninicacao ADS7828 e ESP8266 ----

-- i2c.setup(ID,SDA,SCL,velocidade da trasmissÃ£o)
-- Inicializa o modulo I2C.
-- ID do ESP e' zero.
-- Comunicação entre ADS7828 e ESP8266: SDA é por onde passará os sinais de comando e respostas; SCL vai determinar quando um bit do SDA começa e termina.
i2c.setup(0,2,1,i2c.SLOW)
-- Protocolo de leitura do ADS7828
function ler_sensor(sensor)

    -- Envia uma condicao de inicio do protocolo I2C.
    i2c.start(0)

    -- i2c.adress: configura o endereco I2C e o modo de leitura/escrita para a proxima transferencia.
    -- i2c.address(ID, endereco do dispositivo, direcao)
    -- i2c.TRANSMITTER: para modo de escrita.
    -- Datasheet - Pag. 11: 72 = (2^6) + (2^3)
    -- Verificando se conversor esta' no modo escrita.
    -- Esta funcao retorna true (verdadeiro) ou false (falso). Significando se alcancou (true) ou nao (false) o endereco.
    -- “adress” é de D2 (ESP8266) para SDA (ADS7828) . Write é de D2 para SDA. E read é de SDA para D2.
    if i2c.adress(0,72,i2c.TRANSMITTER) == true then

        -- i2c.write: gravar dados no barramento I2C.
        --  Itens de dados podem ter varios numeros, strings ou tabelas lua.
        -- == 1, pois retorna 1 byte
        -- Deseja-se ler o sensor identificado pela variavel "sensor" usando a referencia interna do ADS7828.
        -- 128+(sensor*16)+12:
        --     sensor: C2  C1  C0
        --       16: Deslocamento ate C0 (SD C2  C1  C0  PD1 PD0 X  X) >>> Deslocar 16 bits >>> sensor*16
        --     12: PD0 = 1 e PD1 = 1 >>> 2^2 + 2^3 = 12
        --     128: SD = 1 >>> 2^7 = 128
        --     128+(sensor*16)+12 equivale a "SD  C2  C1  C0  PD1 PD0 X  X"
        if i2c.write(0,128+(sensor*16)+12) == 1 then
```

```
            -- Cada instrucao I2C ("read" ou "write") tem que ser iniciada com um "start".
            i2c.start(0)

        -- Verificando se conversor esta' no modo leitura.
        if i2c.adress(0,72,i2c.RECEIVER) == true then

            -- Le dados quanto ao numero de bytes da variavel.
            -- c = i2c.read(ID,SCL)
            -- SCL: Numero de bytes de dados que se deseja ler.
            -- Copia para variavel "c" o conteudo da porta 2 do endereco 0.
            c = i2c.read(0,2)
            return c
        end
    end
  end
  return nil
end

-- Seta o timer 1 do ESP8266 para disparar a cada 100 milissegundos e executar a funcao "function".
tmr.alarm(1,100,1,function()

  -- Inicializa variavel sensorX = nil (nulo).
  sensor1 = nil
  sensor2 = nil
  sensor3 = nil
  sensor4 = nil
  sensor5 = nil

  -- sensorX vai receber o valor da funcao read_sen(X), onde X e' o index para o sensor (ch1).
  -- CH0 - Sensor no dedo polegar -   0   0   0   - 0*(2^2) + 0*(2^1) + 0*(2^0) = 0
  --                                      C2        C1          C0
  sensor1 = ler_sensor(0)     -- CH0 - Sensor no dedo polegar - 000
  sensor2 = ler_sensor(4)     -- CH1 - Sensor no dedo indicador - 100
  sensor3 = ler_sensor(1)     -- CH2 - Sensor no dedo medio - 001
  sensor4 = ler_sensor(5)     -- CH3 - Sensor no dedo anelar - 101
  sensor5 = ler_sensor(2)     -- CH4 - Sensor no dedo minimo - 010

  -- Se algo der errado e nao for possi-vel ler ch1, a funcao "read_sen" retorna nil.
  if sensor1 == nil then

    -- No caso de sensorX ainda ser nil, ele recebe dois bytes de valor nulo.
    valor_ads1 = {0,0}
  else

    -- Caso sensor1 tenha um valor diferente de nil,
    --  significa que ch1 foi lido com sucesso, portanto,
    --  a variavel valor_ads1 vai receber o valor resultante do
    --  primeiro byte de sensor1 deslocado (shiffitado) para a direita
    --  em 8 bits (string.byte(sensor1)*128) e adicionado do
    --  valor do segundo byte (string.byte(sensor1,2)).
    valor_ads1 = string.byte(sensor1)*128 + string.byte(sensor1,2)
  end

  if sensor2 == nil then
    valor_ads2 = {0,0}
  else
```

```
        valor_ads2 = string.byte(sensor2)*128 + string.byte(sensor2,2)
    end

    if sensor3 == nil then
        valor_ads3 = {0,0}
    else
        valor_ads3 = string.byte(sensor3)*128 + string.byte(sensor3,2)
    end

    if sensor4 == nil then
        valor_ads4 = {0,0}
    else
        valor_ads4 = string.byte(sensor4)*128 + string.byte(sensor4,2)
    end

    if sensor5 == nil then
        valor_ads5 = {0,0}
    else
        valor_ads5 = string.byte(sensor5)*128 + string.byte(sensor5,2)
    end

    -- Vetor a ser enviado para o computador.
    x = (valor_ads1 .. "," .. valor_ads2 .. "," .. valor_ads3 .. "," .. valor_ads4 .. "," ..
valor_ads5)
end)

------------ Envio de dados ----------

-- Configurando a rede
cfg = {}
cfg.ssid = "rede_luva"      -- Nome da rede
cfg.pwd = "123123123"       -- Senha da rede

wifi.ap.config(cfg)

cfg = {}
cfg.ip = "192.168.0.1"
cfg.netmask = "255.255.255.0"
cfg.gateway = "192.168.0.1"

wifi.ap.setip(cfg)

wifi.setmode(wifi.SOFTAP)

print(wifi.ap.getip())

-- Preparando para enviar dados
i = 0
ippc = "192.168.0.2"
porta = 12345

-- Cria um cliente.
-- net.createConnection(type, secure) -> (tipo, seguro)
--  No caso, UDP, nao e' requerido nenhum tipo de conexao (type).
-- Para net.UDP -> net.udpsocket sub modulo.
sk = net.createConnection(net.UDP)
```

```
sk:connect(porta,ippc)
tmr.delay(30)   -- Tempo para conexao

-- Enviando dados para computador
-- Existem 6 funcoes timer (tmr) disponi-veis. A primeira usada mais acima e a segunda utilizada agora.
-- Por isso, "tmr.alarm(2,..."
tmr.alarm(2,200,1,function()
    if (wifi.ap.getip() == nil) then
        print("Conectando...\n")
        else
            sk:send(x)
            print("Mandei dado: " .. i .. "\n")
            i = i + 1
    end
end)
```

APÊNDICE B

Mais adiante, mostra-se o algoritmo feito no Visual Studio para captura dos sinais enviados pelo ESP8266, cálculos realizados, armazenamento no banco de dados e validação da configuração de mão.

Este algoritmo foi feito com o auxílio da colega do departamento da engenheira elétrica da UFES Letícia Munhoz.

Segue o algoritmo:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Windows.Threading;
using MySql.Data.MySqlClient;
using System.Drawing;
using System.IO;

namespace PG
{
    /// <summary>
    /// Interação lógica para MainWindow.xam
    /// </summary>
    public partial class MainWindow : Window
    {
        ///////// Configurações iniciais a serem colocadas pelo usuário.
        static string tabelaNU = "tab_nu_2_e";          // Tabela onde será armazenado valores calculados do novo usuário dos testes.
                                                        // Indicador, médio e anelar.
                                                        // Fica no botão ("2º)...")
        static string tabelaNUU = "tab_nu_2_ee";        // Tabela onde será armazenado valores calculados do novo usuário dos testes.
                                                        // Polegar e mínimo.
                                                        // Fica no botão ("2º)...")
        static string tabela_bd_nu = "tab_valores_teste";       // Tabela onde será armazenado valores calculados do novo usuário.

        static string tabela_base = "tab_teste";                // Tabela do banco de dados para usuários base (L, I e S)
```

```
        // Para cálculo da diferença dos valores médios de cada configuração e cada dedo
        int contB = 0, contE = 0, contY = 0;                        // Contadores
        double difPol_B, difInd_B, difMed_B, difAne_B, difMin_B;   // Diferença
        double difPol_E, difInd_E, difMed_E, difAne_E, difMin_E;
        double difPol_Y, difInd_Y, difMed_Y, difAne_Y, difMin_Y;
        double vmPol_B, vmInd_B, vmMed_B, vmAne_B, vmMin_B;             // Valores médios do
banco de dados.
        double vmPol_E, vmInd_E, vmMed_E, vmAne_E, vmMin_E;
        double vmPol_Y, vmInd_Y, vmMed_Y, vmAne_Y, vmMin_Y;

        // Configuração média para configurações B, E e Y de cada dedo.
        double  configMediaB_Pol,  configMediaB_Ind,  configMediaB_Med,  configMediaB_Ane,
configMediaB_Min;
        double  configMediaE_Pol,  configMediaE_Ind,  configMediaE_Med,  configMediaE_Ane,
configMediaE_Min;
        double  configMediaY_Pol,  configMediaY_Ind,  configMediaY_Med,  configMediaY_Ane,
configMediaY_Min;

        int cont;       // Contador

        /////// Variância
        // Para configuração B
        double[] varB_Pol = new double[tam];
        double[] varB_Ind = new double[tam];
        double[] varB_Med = new double[tam];
        double[] varB_Ane = new double[tam];
        double[] varB_Min = new double[tam];
        double variancB_Pol = 0, variancB_Ind = 0, variancB_Med = 0, variancB_Ane = 0,
variancB_Min = 0;
        double varianciaB_Pol, varianciaB_Ind, varianciaB_Med, varianciaB_Ane, varianciaB_Min;
        // Para configuração E
        double[] varE_Pol = new double[tam];
        double[] varE_Ind = new double[tam];
        double[] varE_Med = new double[tam];
        double[] varE_Ane = new double[tam];
        double[] varE_Min = new double[tam];
        double variancE_Pol = 0, variancE_Ind = 0, variancE_Med = 0, variancE_Ane = 0,
variancE_Min = 0;
        double varianciaE_Pol, varianciaE_Ind, varianciaE_Med, varianciaE_Ane, varianciaE_Min;
        // Para configuração Y
        double[] varY_Pol = new double[tam];
        double[] varY_Ind = new double[tam];
        double[] varY_Med = new double[tam];
        double[] varY_Ane = new double[tam];
        double[] varY_Min = new double[tam];
        double variancY_Pol = 0, variancY_Ind = 0, variancY_Med = 0, variancY_Ane = 0,
variancY_Min = 0;
        double varianciaY_Pol, varianciaY_Ind, varianciaY_Med, varianciaY_Ane, varianciaY_Min;

        // Desvio padrão
        double dpB_Pol, dpB_Ind, dpB_Med, dpB_Ane, dpB_Min;     // Para configuração B
        double dpE_Pol, dpE_Ind, dpE_Med, dpE_Ane, dpE_Min;     // Para configuração E
        double dpY_Pol, dpY_Ind, dpY_Med, dpY_Ane, dpY_Min;     // Para configuração Y

        ///////// Limites inferior e superior
        // Configuração B
        double limInfB_Pol, limInfB_Ind, limInfB_Med, limInfB_Ane, limInfB_Min;
```

```
double limSupB_Pol, limSupB_Ind, limSupB_Med, limSupB_Ane, limSupB_Min;
// Configuração E
double limInfE_Pol, limInfE_Ind, limInfE_Med, limInfE_Ane, limInfE_Min;
double limSupE_Pol, limSupE_Ind, limSupE_Med, limSupE_Ane, limSupE_Min;
// Configuração Y
double limInfY_Pol, limInfY_Ind, limInfY_Med, limInfY_Ane, limInfY_Min;
double limSupY_Pol, limSupY_Ind, limSupY_Med, limSupY_Ane, limSupY_Min;

////////////////// Listas que guardam valores dos sensores para cada usuário (L, I e S)
//////// Usuário L
// Configuração B
List<double> listPolL_configB = new List<double>();     // Polegar
List<double> listIndL_configB = new List<double>();     // Indicador
List<double> listMedL_configB = new List<double>();     // Médio
List<double> listAneL_configB = new List<double>();     // Anelar
List<double> listMinL_configB = new List<double>();     // Mínimo
// Configuração E
List<double> listPolL_configE = new List<double>();     // Polegar
List<double> listIndL_configE = new List<double>();     // Indicador
List<double> listMedL_configE = new List<double>();     // Médio
List<double> listAneL_configE = new List<double>();     // Anelar
List<double> listMinL_configE = new List<double>();     // Mínimo
// Configuração Y
List<double> listPolL_configY = new List<double>();     // Polegar
List<double> listIndL_configY = new List<double>();     // Indicador
List<double> listMedL_configY = new List<double>();     // Médio
List<double> listAneL_configY = new List<double>();     // Anelar
List<double> listMinL_configY = new List<double>();     // Mínimo
//////// Usuário I
// Configuração B
List<double> listPolI_configB = new List<double>();     // Polegar
List<double> listIndI_configB = new List<double>();     // Indicador
List<double> listMedI_configB = new List<double>();     // Médio
List<double> listAneI_configB = new List<double>();     // Anelar
List<double> listMinI_configB = new List<double>();     // Mínimo
// Configuração E
List<double> listPolI_configE = new List<double>();     // Polegar
List<double> listIndI_configE = new List<double>();     // Indicador
List<double> listMedI_configE = new List<double>();     // Médio
List<double> listAneI_configE = new List<double>();     // Anelar
List<double> listMinI_configE = new List<double>();     // Mínimo
// Configuração Y
List<double> listPolI_configY = new List<double>();     // Polegar
List<double> listIndI_configY = new List<double>();     // Indicador
List<double> listMedI_configY = new List<double>();     // Médio
List<double> listAneI_configY = new List<double>();     // Anelar
List<double> listMinI_configY = new List<double>();     // Mínimo
//////// Usuário S
// Configuração B
List<double> listPolS_configB = new List<double>();     // Polegar
List<double> listIndS_configB = new List<double>();     // Indicador
List<double> listMedS_configB = new List<double>();     // Médio
List<double> listAneS_configB = new List<double>();     // Anelar
List<double> listMinS_configB = new List<double>();     // Mínimo
// Configuração E
List<double> listPolS_configE = new List<double>();     // Polegar
List<double> listIndS_configE = new List<double>();     // Indicador
```

```
        List<double> listMedS_configE = new List<double>();     // Médio
        List<double> listAneS_configE = new List<double>();     // Anelar
        List<double> listMinS_configE = new List<double>();     // Mínimo
        // Configuração Y
        List<double> listPolS_configY = new List<double>();     // Polegar
        List<double> listIndS_configY = new List<double>();     // Indicador
        List<double> listMedS_configY = new List<double>();     // Médio
        List<double> listAneS_configY = new List<double>();     // Anelar
        List<double> listMinS_configY = new List<double>();     // Mínimo

        ////////// Novo usuário
        List<double> listPol = new List<double>();     // Polegar
        List<double> listInd = new List<double>();     // Indicador
        List<double> listMed = new List<double>();     // Médio
        List<double> listAne = new List<double>();     // Anelar
        List<double> listMin = new List<double>();     // Mínimo
        // Configuração média
        double configMedia_Pol, configMedia_Ind, configMedia_Med, configMedia_Ane,
configMedia_Min;
        // Variância
        double[] var_Pol = new double[tam];
        double[] var_Ind = new double[tam];
        double[] var_Med = new double[tam];
        double[] var_Ane = new double[tam];
        double[] var_Min = new double[tam];
        double varianc_Pol = 0, varianc_Ind = 0, varianc_Med = 0, varianc_Ane = 0,
varianc_Min = 0;
        double variancia_Pol, variancia_Ind, variancia_Med, variancia_Ane, variancia_Min;
        // Desvio padrão
        double dp_Pol, dp_Ind, dp_Med, dp_Ane, dp_Min;
        // Limites inferior e superior
        double limInf_Pol, limInf_Ind, limInf_Med, limInf_Ane, limInf_Min;
        double limSup_Pol, limSup_Ind, limSup_Med, limSup_Ane, limSup_Min;

        // Diretório das imagens das configurações de mão
        string dir = "D:\\SUELLEN\\Atualizar\\UFES\\PG\\Projeto de Graduação II\\Visual
Studio\\Pictures box\\";

        int amostras = 0;               // Para ser mostrado na tela (TextBox).

        // Passa a string de conexão.
        MySqlConnection objcon = new MySqlConnection("server=localhost;port=3307;User
Id=root;database=bd_pg;password=usbw");

        // Relacionado à conexão e manipulação dos dados recebidos via rede.
        IPEndPoint PC;
        Socket SocketComunicacao;
        Thread Receptor;
        int TamanhoRX;
        string x;
        string[] splits;
        int i = 0;

        static int tam = 20;            // Limitação para as iterações

        // Valores dos dados dos sensores em bits (enviados pelo ESP8266)
        string sen_b1, sen_b2, sen_b3, sen_b4, sen_b5;
```

```
        // Valores dos dados dos sensores em bits (enviados pelo ESP8266) - Para conversão de string para double
        double sen_b1D, sen_b2D, sen_b3D, sen_b4D, sen_b5D;

        // Valores provenientes dos sensores convertidos em tensão (V)
        double polV, indV, medV, aneV, minV;

        // Constante para a conversão dos valores das entradas para tensão (bits -> V)
        double cte = 0.0006105;

        // Valores provenientes dos sensores convertidos para string (de double para string) para apresentação na interface
        string polS, indS, medS, aneS, minS;

        public DispatcherTimer TimerRawSignal = new DispatcherTimer();

        public MainWindow()
        {
            InitializeComponent();

            PC = new IPEndPoint(IPAddress.Parse("192.168.0.2"), 12345);
            SocketComunicacao = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);
            Receptor = new Thread(Conectar);      // Thread: permite execução de mais de uma operação ao mesmo tempo.
        }

        // Função que atualiza informações da interface de tempos em tempos.
        void AtualizaJanela(object sender, EventArgs e)
        {
            textBox_x.Text = x;
            textBox_Pol.Text = polS;
            textBox_Ind.Text = indS;
            textBox_Med.Text = medS;
            textBox_Ane.Text = aneS;
            textBox_Min.Text = minS;
        }

        // Carregamento da janela de tempos em tempos.
        private void Window_Loaded(object sender, RoutedEventArgs e)
        {
            TimerRawSignal.Tick += new System.EventHandler(AtualizaJanela);
            TimerRawSignal.Interval = new System.TimeSpan(0, 0, 0, 0, 200);   // Seta evento de timer a cada 200 ms.
        }

        private void Window_Closing(object sender, System.ComponentModel.CancelEventArgs e)
        {
            Receptor.Abort();
            Environment.Exit(0);
            TimerRawSignal.Stop();
        }

        // Quando botão for clicado. Faz conexão e deconexão.
        private void button_Conectar_Click(object sender, RoutedEventArgs e)
```

```
        {
            if (button_Conectar.Content.ToString() == "Conectar")
            {
                if (i == 0)
                {
                    try
                    {
                        SocketComunicacao.Bind(PC);
                        TimerRawSignal.Start();
                        button_Conectar.Content = "Desconectar";

                        Receptor.Start();
                        MessageBox.Show("Conectado!");
                        i++;
                    }
                    catch (Exception exc) { MessageBox.Show("Não foi possível conectar. Tente
novamente!"); }

                }

                else
                {
                    TimerRawSignal.Start();
                    button_Conectar.Content = "Desconectar";
                }
            }

            else
            {
                TimerRawSignal.Stop();
                button_Conectar.Content = "Conectar";
            }
        }

        // Conexão e tratamento dos dados recebidos via rede.
        public void Conectar()
        {
            byte[] BytesRX = new byte[512];

            while (true)
            {
                TamanhoRX = SocketComunicacao.Receive(BytesRX);
                x = System.Text.Encoding.UTF8.GetString(BytesRX, 0, TamanhoRX);

                splits = x.Split(',');

                // Valores do ESP7828, os quais vem do conversor ADS7828
                sen_b1 = splits[0];       // Polegar
                sen_b2 = splits[1];       // Indicador
                sen_b3 = splits[2];       // Médio
                sen_b4 = splits[3];       // Anelar
                sen_b5 = splits[4];       // Mínimo

                // Convertendo string para double
                sen_b1D = Convert.ToDouble(sen_b1);
                sen_b2D = Convert.ToDouble(sen_b2);
```

```
            sen_b3D = Convert.ToDouble(sen_b3);
            sen_b4D = Convert.ToDouble(sen_b4);
            sen_b5D = Convert.ToDouble(sen_b5);

            // Valores em tensão (V)
            polV = sen_b1D * cte;
            indV = sen_b2D * cte;
            medV = sen_b3D * cte;
            aneV = sen_b4D * cte;
            minV = sen_b5D * cte;

            // Convertendo double para string - Para apresentação na interface
            polS = Convert.ToString(polV);
            indS = Convert.ToString(indV);
            medS = Convert.ToString(medV);
            aneS = Convert.ToString(aneV);
            minS = Convert.ToString(minV);
        }
    }

    // Iamgem da configuração de mão escolhida pelo novo usuário.
    // Configuração B
    private void checkBox_configB_Checked(object sender, RoutedEventArgs e)
    {
        image_config.Visibility = Visibility.Visible;
        Uri imageUri = new Uri(dir + "B.png");
        image_config.Source = new BitmapImage(imageUri);
        checkBox_configE.Visibility = Visibility.Hidden;
        checkBox_configY.Visibility = Visibility.Hidden;
    }
    private void checkBox_configB_Unchecked(object sender, RoutedEventArgs e)
    {
        image_config.Visibility = Visibility.Hidden;
        checkBox_configE.Visibility = Visibility.Visible;
        checkBox_configY.Visibility = Visibility.Visible;
    }
    // Configuração E
    private void checkBox_configE_Checked(object sender, RoutedEventArgs e)
    {
        image_config.Visibility = Visibility.Visible;
        Uri imageUri = new Uri(dir + "E.png");
        image_config.Source = new BitmapImage(imageUri);
        checkBox_configB.Visibility = Visibility.Hidden;
        checkBox_configY.Visibility = Visibility.Hidden;
    }
    private void checkBox_configE_Unchecked(object sender, RoutedEventArgs e)
    {
        image_config.Visibility = Visibility.Hidden;
        checkBox_configB.Visibility = Visibility.Visible;
        checkBox_configY.Visibility = Visibility.Visible;
    }
    // Configuração Y
    private void checkBox_configY_Checked(object sender, RoutedEventArgs e)
    {
        image_config.Visibility = Visibility.Visible;
        Uri imageUri = new Uri(dir + "Y.png");
        image_config.Source = new BitmapImage(imageUri);
```

```
            checkBox_configB.Visibility = Visibility.Hidden;
            checkBox_configE.Visibility = Visibility.Hidden;
        }
        private void checkBox_configY_Unchecked(object sender, RoutedEventArgs e)
        {
            image_config.Visibility = Visibility.Hidden;
            checkBox_configB.Visibility = Visibility.Visible;
            checkBox_configE.Visibility = Visibility.Visible;
        }

        // Validação da configuração de mão feita por um novo usuário com base nos valores base (dos usuários L, I e S).
        private void button_validar_Click(object sender, RoutedEventArgs e)
        {
            // Limpa listas.
            listPol.Clear();
            listInd.Clear();
            listMed.Clear();
            listAne.Clear();
            listMin.Clear();

            // Preenche listas de valores para os dedos indicador, médio e anelar.
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from " + tabelaNU + "";

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
                MySqlDataReader tab = cmd.ExecuteReader();      // Instância
                while (tab.Read())
                {
                    // Para cada iteração, é adicionado o valor ao listbox.
                    listInd.Add(Convert.ToDouble(tab["indicador"]));
                    listMed.Add(Convert.ToDouble(tab["medio"]));
                    listAne.Add(Convert.ToDouble(tab["anelar"]));
                }
                objcon.Close();     // Fecha a conexão com o banco de dados.
            }
            catch
            {
                MessageBox.Show("Erro: " + tabelaNU + "!");
            }

            // Preenche listas de valores para os dedos polegar e mínimo.
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from " + tabelaNUU + "";

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);
```

```
    objcon.Open();       // Abre conexão com o banco de dados.
    MySqlDataReader tab = cmd.ExecuteReader();       // Instância
    while (tab.Read())
    {
        // Para cada iteração, é adicionado o valor ao listbox.
        listPol.Add(Convert.ToDouble(tab["polegar"]));
        listMin.Add(Convert.ToDouble(tab["minimo"]));
    }
    objcon.Close();      // Fecha a conexão com o banco de dados.
}
catch
{
    MessageBox.Show("Erro: " + tabelaNUU + "!");
}

////////////// Cálculos para validação
// Configuração média
for (cont = 0; cont < tam; cont++)
{
    configMedia_Pol = configMedia_Pol + (listPol[cont]);
    configMedia_Ind = configMedia_Ind + (listInd[cont]);
    configMedia_Med = configMedia_Med + (listMed[cont]);
    configMedia_Ane = configMedia_Ane + (listAne[cont]);
    configMedia_Min = configMedia_Min + (listMin[cont]);
}
configMedia_Pol = configMedia_Pol / (cont);
configMedia_Ind = configMedia_Ind / (cont);
configMedia_Med = configMedia_Med / (cont);
configMedia_Ane = configMedia_Ane / (cont);
configMedia_Min = configMedia_Min / (cont);
// Somatório para encontrar a variância
for (cont = 0; cont < tam; cont++)
{
    var_Pol[cont] = (Math.Pow(listPol[cont] - configMedia_Pol, 2));
    var_Ind[cont] = (Math.Pow(listInd[cont] - configMedia_Ind, 2));
    var_Med[cont] = (Math.Pow(listMed[cont] - configMedia_Med, 2));
    var_Ane[cont] = (Math.Pow(listAne[cont] - configMedia_Ane, 2));
    var_Min[cont] = (Math.Pow(listMin[cont] - configMedia_Min, 2));
}
for (cont = 0; cont < tam; cont++)
{
    varianc_Pol = varianc_Pol + var_Pol[cont];
    varianc_Ind = varianc_Ind + var_Ind[cont];
    varianc_Med = varianc_Med + var_Med[cont];
    varianc_Ane = varianc_Ane + var_Ane[cont];
    varianc_Min = varianc_Min + var_Min[cont];
}
// Variância
variancia_Pol = varianc_Pol / (cont);
variancia_Ind = varianc_Ind / (cont);
variancia_Med = varianc_Med / (cont);
variancia_Ane = varianc_Ane / (cont);
variancia_Min = varianc_Min / (cont);
// Desvio padrão
dp_Pol = Math.Sqrt(variancia_Pol);
dp_Ind = Math.Sqrt(variancia_Ind);
```

```
            dp_Med = Math.Sqrt(variancia_Med);
            dp_Ane = Math.Sqrt(variancia_Ane);
            dp_Min = Math.Sqrt(variancia_Min);
            /////// Limites
            // Limite inferior
            limInf_Pol = configMedia_Pol - dp_Pol;
            limInf_Ind = configMedia_Ind - dp_Ind;
            limInf_Med = configMedia_Med - dp_Med;
            limInf_Ane = configMedia_Ane - dp_Ane;
            limInf_Min = configMedia_Min - dp_Min;
            // Limite superior
            limSup_Pol = configMedia_Pol + dp_Pol;
            limSup_Ind = configMedia_Ind + dp_Ind;
            limSup_Med = configMedia_Med + dp_Med;
            limSup_Ane = configMedia_Ane + dp_Ane;
            limSup_Min = configMedia_Min + dp_Min;

            /////////// Diferença (Para verificação de qual configuração de mão foi realizada)
            //// Pegando valores base do banco de dados para fazer os cálculos
            // Para configuração B
            try
            {
                string sql = "SELECT * from tab_valores_configb";       // String com o comando
a ser executado

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
                MySqlDataReader tab = cmd.ExecuteReader();      // Instância
                while (tab.Read())
                {
                    // Para cada iteração, é adicionado o valor ao listbox.
                    vmPol_B = Convert.ToDouble(tab["vmPol"]);
                    vmInd_B = Convert.ToDouble(tab["vmInd"]);
                    vmMed_B = Convert.ToDouble(tab["vmMed"]);
                    vmAne_B = Convert.ToDouble(tab["vmAne"]);
                    vmMin_B = Convert.ToDouble(tab["vmMin"]);
                }
                objcon.Close();     // Fecha a conexão com o banco de dados.
            }
            catch
            {
                MessageBox.Show("Erro: tab_valores_configb!");
            }
            // Para configuração E
            try
            {
                string sql = "SELECT * from tab_valores_confige";       // String com o comando
a ser executado

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
```

```
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                vmPol_E = Convert.ToDouble(tab["vmPol"]);
                vmInd_E = Convert.ToDouble(tab["vmInd"]);
                vmMed_E = Convert.ToDouble(tab["vmMed"]);
                vmAne_E = Convert.ToDouble(tab["vmAne"]);
                vmMin_E = Convert.ToDouble(tab["vmMin"]);
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_valores_confige!");
        }
        // Para configuração Y
        try
        {
            string sql = "SELECT * from tab_valores_configy";       // String com o comando
a ser executado.

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                vmPol_Y = Convert.ToDouble(tab["vmPol"]);
                vmInd_Y = Convert.ToDouble(tab["vmInd"]);
                vmMed_Y = Convert.ToDouble(tab["vmMed"]);
                vmAne_Y = Convert.ToDouble(tab["vmAne"]);
                vmMin_Y = Convert.ToDouble(tab["vmMin"]);
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_valores_configy!");
        }
        // Configuração B
        difPol_B = Math.Abs(vmPol_B - configMedia_Pol);
        difInd_B = Math.Abs(vmInd_B - configMedia_Ind);
        difMed_B = Math.Abs(vmMed_B - configMedia_Med);
        difAne_B = Math.Abs(vmAne_B - configMedia_Ane);
        difMin_B = Math.Abs(vmMin_B - configMedia_Min);
        // Configuração E
        difPol_E = Math.Abs(vmPol_E - configMedia_Pol);
        difInd_E = Math.Abs(vmInd_E - configMedia_Ind);
        difMed_E = Math.Abs(vmMed_E - configMedia_Med);
        difAne_E = Math.Abs(vmAne_E - configMedia_Ane);
        difMin_E = Math.Abs(vmMin_E - configMedia_Min);
        // Configuração Y
        difPol_Y = Math.Abs(vmPol_Y - configMedia_Pol);
```

```
difInd_Y = Math.Abs(vmInd_Y - configMedia_Ind);
difMed_Y = Math.Abs(vmMed_Y - configMedia_Med);
difAne_Y = Math.Abs(vmAne_Y - configMedia_Ane);
difMin_Y = Math.Abs(vmMin_Y - configMedia_Min);
// Polegar
if (difPol_B < difPol_E && difPol_B < difPol_Y)
{ contB++; }
else if (difPol_E < difPol_B && difPol_E < difPol_Y)
{ contE++; }
else if (difPol_Y < difPol_B && difPol_Y < difPol_E)
{ contY++; }
// Indicador
if (difInd_B < difInd_E && difInd_B < difInd_Y)
{ contB++; }
else if (difInd_E < difInd_B && difInd_E < difInd_Y)
{ contE++; }
else if (difInd_Y < difInd_B && difInd_Y < difInd_E)
{ contY++; }
// Médio
if (difMed_B < difMed_E && difMed_B < difMed_Y)
{ contB++; }
else if (difMed_E < difMed_B && difMed_E < difMed_Y)
{ contE++; }
else if (difMed_Y < difMed_B && difMed_Y < difMed_E)
{ contY++; }
// Anelar
if (difAne_B < difAne_E && difAne_B < difAne_Y)
{ contB++; }
else if (difAne_E < difAne_B && difAne_E < difAne_Y)
{ contE++; }
else if (difAne_Y < difAne_B && difAne_Y < difAne_E)
{ contY++; }
// Mínimo
if (difMin_B < difMin_E && difMin_B < difMin_Y)
{ contB++; }
else if (difMin_E < difMin_B && difMin_E < difMin_Y)
{ contE++; }
else if (difMin_Y < difMin_B && difMin_Y < difMin_E)
{ contY++; }

// Mostrar na tela
if (contB >= 3)
{
    image_validar.Visibility = Visibility.Visible;
    Uri imageUri = new Uri(dir + "B.png");
    image_validar.Source = new BitmapImage(imageUri);
}
else if (contE >= 3)
{
    image_validar.Visibility = Visibility.Visible;
    Uri imageUri = new Uri(dir + "E.png");
    image_validar.Source = new BitmapImage(imageUri);
}
else if (contY >= 3)
{
    image_validar.Visibility = Visibility.Visible;
    Uri imageUri = new Uri(dir + "Y.png");
```

```
            image_validar.Source = new BitmapImage(imageUri);
        }

        // Armanenando valor médio, desvio padrão e limites do novo usuário no banco de dados.
        // Entrada no código: "tabela_bd_nu".
        try
        {
            objcon.Open();          // Abre conexão com o banco de dados.

            // Query SQL
            MySqlCommand cmd = new MySqlCommand("INSERT INTO " + tabela_bd_nu +
"(vmPol, vmInd, vmMed, vmAne, vmMin, " +
                "dp_Pol, dp_Ind, dp_Med, dp_Ane, dp_Min, " +
                "limInf_Pol, limInf_Ind, limInf_Med, limInf_Ane, limInf_Min, " +
                "limSup_Pol, limSup_Ind, limSup_Med, limSup_Ane, limSup_Min)" +
                "    VALUES    ('"    +    Convert.ToString(configMedia_Pol)    +    "','"    +
Convert.ToString(configMedia_Ind) + "','" +
                Convert.ToString(configMedia_Med) + "','" + Convert.ToString(configMedia_Ane)
+ "','" +
                Convert.ToString(configMedia_Min) + "','" +
                Convert.ToString(dp_Pol) + "','" + Convert.ToString(dp_Ind) + "','" +
                Convert.ToString(dp_Med) + "','" + Convert.ToString(dp_Ane) + "','" +
                Convert.ToString(dp_Min) + "','" +
                Convert.ToString(limInf_Pol) + "','" + Convert.ToString(limInf_Ind) + "','" +
                Convert.ToString(limInf_Med) + "','" + Convert.ToString(limInf_Ane) + "','" +
                Convert.ToString(limInf_Min) + "','" +
                Convert.ToString(limSup_Pol) + "','" + Convert.ToString(limSup_Ind) + "','" +
                Convert.ToString(limSup_Med) + "','" + Convert.ToString(limSup_Ane) + "','" +
                Convert.ToString(limSup_Min) + "')", objcon);

            cmd.ExecuteNonQuery();      //Executa a Query SQL
            objcon.Close();             // Fecha a conexão com o banco de dados.

            //MessageBox.Show("Conectado com o banco de dados!");
        }
        catch
        {
            MessageBox.Show("Não conectou com o banco de dados!");
        }
    }

    // Botão "2º) Coletar e armazenar amostras" para todo aquele que usar e testar a luva.
    private void button_amostras_Click(object sender, RoutedEventArgs e)
    {
        amostras++;
        textBox_amostras.Text = Convert.ToString(amostras);

        if (amostras == 20)     // Máximo de 20 coletas. Para controle.
        {
            amostras = 0;
        }

        // Armazenando dados no banco de dados
        try
        {
            objcon.Open();          // Abre conexão com o banco de dados.
```

```
            // Query SQL
            MySqlCommand cmd = new MySqlCommand("INSERT INTO " + tabela_base + "
(polegar, indicador, medio, anelar, minimo)" +
                " VALUES ('" + polS + "','" + indS + "','" + medS + "','" + aneS + "','" +
minS + "')", objcon);

            cmd.ExecuteNonQuery();      //Executa a Query SQL
            objcon.Close();             // Fecha a conexão com o banco de dados.

            //MessageBox.Show("Conectado com o banco de dados!");
        }
        catch
        {
            MessageBox.Show("Não conectou com o banco de dados!");
        }
    }

    // Botão "Calcular (Valores base)": faz o cálculo para obtenção dos valores base, os quais
    // serão usados como dados de referência para comparação com testes com outros usuários.
    // Dados obtidos a partir dos testes realizados nos usuários L, I e S.
    // Depois do cálculos feitos, armazená-os em tabelas no banco de dados.
    private void button_calcular_Click(object sender, RoutedEventArgs e)
    {
        /////// Limpa listas.
        //// Usuário I
        // Configuração B
        listPolI_configB.Clear();
        listIndI_configB.Clear();
        listMedI_configB.Clear();
        listAneI_configB.Clear();
        listMinI_configB.Clear();
        // Configuração E
        listPolI_configE.Clear();
        listIndI_configE.Clear();
        listMedI_configE.Clear();
        listAneI_configE.Clear();
        listMinI_configE.Clear();
        // Configuração Y
        listPolI_configY.Clear();
        listIndI_configY.Clear();
        listMedI_configY.Clear();
        listAneI_configY.Clear();
        listMinI_configY.Clear();
        //// Usuário L
        // Configuração B
        listPolL_configB.Clear();
        listIndL_configB.Clear();
        listMedL_configB.Clear();
        listAneL_configB.Clear();
        listMinL_configB.Clear();
        // Configuração E
        listPolL_configE.Clear();
        listIndL_configE.Clear();
        listMedL_configE.Clear();
        listAneL_configE.Clear();
        listMinL_configE.Clear();
```

```
            // Configuração Y
            listPolL_configY.Clear();
            listIndL_configY.Clear();
            listMedL_configY.Clear();
            listAneL_configY.Clear();
            listMinL_configY.Clear();
            //// Usuário S
            // Configuração B
            listPolS_configB.Clear();
            listIndS_configB.Clear();
            listMedS_configB.Clear();
            listAneS_configB.Clear();
            listMinS_configB.Clear();
            // Configuração E
            listPolS_configE.Clear();
            listIndS_configE.Clear();
            listMedS_configE.Clear();
            listAneS_configE.Clear();
            listMinS_configE.Clear();
            // Configuração Y
            listPolS_configY.Clear();
            listIndS_configY.Clear();
            listMedS_configY.Clear();
            listAneS_configY.Clear();
            listMinS_configY.Clear();

            ////////////////////////////// Usuário I
            //// Usuário I - Configuração B
            // Preenche listas de valores para os dedos indicador, médio e anelar.
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from tab_nu_1_b";

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
                MySqlDataReader tab = cmd.ExecuteReader();      // Instância
                while (tab.Read())
                {
                    // Para cada iteração, é adicionado o valor ao listbox.
                    listIndI_configB.Add(Convert.ToDouble(tab["indicador"]));
                    listMedI_configB.Add(Convert.ToDouble(tab["medio"]));
                    listAneI_configB.Add(Convert.ToDouble(tab["anelar"]));
                }
                objcon.Close();     // Fecha a conexão com o banco de dados.
            }
            catch
            {
                MessageBox.Show("Erro: tab_nu_1_b!");
            }
            //// Usuário I - Configuração B
            // Preenche listas de valores para os dedos polegar e mínimo.
            try
            {
```

```
            // String com o comando a ser executado
            string sql = "SELECT * from tab_nu_1_bb";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listPolI_configB.Add(Convert.ToDouble(tab["polegar"]));
                listMinI_configB.Add(Convert.ToDouble(tab["minimo"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_nu_1_bb!");
        }
        //// Usuário I - Configuração E
        // Preenche listas de valores para os dedos indicador, médio e anelar.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_nu_1_e";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listIndI_configE.Add(Convert.ToDouble(tab["indicador"]));
                listMedI_configE.Add(Convert.ToDouble(tab["medio"]));
                listAneI_configE.Add(Convert.ToDouble(tab["anelar"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_nu_1_e!");
        }
        //// Usuário I - Configuração E
        // Preenche listas de valores para os dedos polegar e mínimo.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_nu_1_ee";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);
```

```
            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listPolI_configE.Add(Convert.ToDouble(tab["polegar"]));
                listMinI_configE.Add(Convert.ToDouble(tab["minimo"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_nu_1_ee!");
        }
        //// Usuário I - Configuração Y
        // Preenche listas de valores para os dedos indicador, médio e anelar.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_nu_1_y";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listIndI_configY.Add(Convert.ToDouble(tab["indicador"]));
                listMedI_configY.Add(Convert.ToDouble(tab["medio"]));
                listAneI_configY.Add(Convert.ToDouble(tab["anelar"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_nu_1_y!");
        }
        //// Usuário I - Configuração Y
        // Preenche listas de valores para os dedos polegar e mínimo.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_nu_1_yy";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
```

```
                listPolI_configY.Add(Convert.ToDouble(tab["polegar"]));
                listMinI_configY.Add(Convert.ToDouble(tab["minimo"]));
            }
            objcon.Close();      // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_nu_1_yy!");
        }
        ////////////////////////////// Usuário L
        // Usuário L - Configuração B
        // Preenche listas de valores para os dedos indicador, médio e anelar.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_usuarioL_b";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();       // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();       // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listIndL_configB.Add(Convert.ToDouble(tab["indicador"]));
                listMedL_configB.Add(Convert.ToDouble(tab["medio"]));
                listAneL_configB.Add(Convert.ToDouble(tab["anelar"]));
            }
            objcon.Close();      // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_usuarioL_b!");
        }
        // Usuário L - Configuração B
        // Preenche listas de valores para os dedos polegar e mínimo.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_usuarioL_bb";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();       // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();       // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listPolL_configB.Add(Convert.ToDouble(tab["polegar"]));
                listMinL_configB.Add(Convert.ToDouble(tab["minimo"]));
            }
            objcon.Close();      // Fecha a conexão com o banco de dados.
        }
```

```
            catch
            {
                MessageBox.Show("Erro: tab_usuarioL_bb!");
            }
            // Usuário L - Configuração E
            // Preenche listas de valores para os dedos indicador, médio e anelar.
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from tab_usuarioL_e";

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
                MySqlDataReader tab = cmd.ExecuteReader();      // Instância
                while (tab.Read())
                {
                    // Para cada iteração, é adicionado o valor ao listbox.
                    listIndL_configE.Add(Convert.ToDouble(tab["indicador"]));
                    listMedL_configE.Add(Convert.ToDouble(tab["medio"]));
                    listAneL_configE.Add(Convert.ToDouble(tab["anelar"]));
                }
                objcon.Close();     // Fecha a conexão com o banco de dados.
            }
            catch
            {
                MessageBox.Show("Erro: tab_usuarioL_e!");
            }
            // Usuário L - Configuração E
            // Preenche listas de valores para os dedos polegar e mínimo.
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from tab_usuarioL_ee";

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
                MySqlDataReader tab = cmd.ExecuteReader();      // Instância
                while (tab.Read())
                {
                    // Para cada iteração, é adicionado o valor ao listbox.
                    listPolL_configE.Add(Convert.ToDouble(tab["polegar"]));
                    listMinL_configE.Add(Convert.ToDouble(tab["minimo"]));
                }
                objcon.Close();     // Fecha a conexão com o banco de dados.
            }
            catch
            {
                MessageBox.Show("Erro: tab_usuarioL_ee!");
            }
            // Usuário L - Configuração Y
            // Preenche listas de valores para os dedos indicador, médio e anelar.
```

```
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from tab_usuarioL_y";

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
                MySqlDataReader tab = cmd.ExecuteReader();      // Instância
                while (tab.Read())
                {
                    // Para cada iteração, é adicionado o valor ao listbox.
                    listIndL_configY.Add(Convert.ToDouble(tab["indicador"]));
                    listMedL_configY.Add(Convert.ToDouble(tab["medio"]));
                    listAneL_configY.Add(Convert.ToDouble(tab["anelar"]));
                }
                objcon.Close();     // Fecha a conexão com o banco de dados.
            }
            catch
            {
                MessageBox.Show("Erro: tab_usuarioL_y!");
            }
            // Usuário L - Configuração Y
            // Preenche listas de valores para os dedos polegar e mínimo.
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from tab_usuarioL_yy";

                // Instância do comando recebendo como parâmetro a string com o comando (sql) e a conexão (objcon).
                MySqlCommand cmd = new MySqlCommand(sql, objcon);

                objcon.Open();      // Abre conexão com o banco de dados.
                MySqlDataReader tab = cmd.ExecuteReader();      // Instância
                while (tab.Read())
                {
                    // Para cada iteração, é adicionado o valor ao listbox.
                    listPolL_configY.Add(Convert.ToDouble(tab["polegar"]));
                    listMinL_configY.Add(Convert.ToDouble(tab["minimo"]));
                }
                objcon.Close();     // Fecha a conexão com o banco de dados.
            }
            catch
            {
                MessageBox.Show("Erro: tab_usuarioL_yy!");
            }
            ////////////////////////////// Usuário S
            // Usuário S - Configuração B
            // Preenche listas de valores para os dedos indicador, médio e anelar.
            try
            {
                // String com o comando a ser executado
                string sql = "SELECT * from tab_usuarioS_b";
```

```
            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listIndS_configB.Add(Convert.ToDouble(tab["indicador"]));
                listMedS_configB.Add(Convert.ToDouble(tab["medio"]));
                listAneS_configB.Add(Convert.ToDouble(tab["anelar"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_usuarioS_b!");
        }
        // Usuário S - Configuração B
        // Preenche listas de valores para os dedos polegar e mínimo.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_usuarioS_bb";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listPolS_configB.Add(Convert.ToDouble(tab["polegar"]));
                listMinS_configB.Add(Convert.ToDouble(tab["minimo"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_usuarioS_bb!");
        }
        // Usuário S - Configuração E
        // Preenche listas de valores para os dedos indicador, médio e anelar.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_usuarioS_e";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
```

```
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listIndS_configE.Add(Convert.ToDouble(tab["indicador"]));
                listMedS_configE.Add(Convert.ToDouble(tab["medio"]));
                listAneS_configE.Add(Convert.ToDouble(tab["anelar"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_usuarioS_e!");
        }
        // Usuário S - Configuração E
        // Preenche listas de valores para os dedos polegar e mínimo.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_usuarioS_ee";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listPolS_configE.Add(Convert.ToDouble(tab["polegar"]));
                listMinS_configE.Add(Convert.ToDouble(tab["minimo"]));
            }
            objcon.Close();     // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_usuarioS_ee!");
        }
        // Usuário S - Configuração Y
        // Preenche listas de valores para os dedos indicador, médio e anelar.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_usuarioS_y";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listIndS_configY.Add(Convert.ToDouble(tab["indicador"]));
                listMedS_configY.Add(Convert.ToDouble(tab["medio"]));
                listAneS_configY.Add(Convert.ToDouble(tab["anelar"]));
```

```
            }
            objcon.Close();      // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_usuarioS_y!");
        }
        // Usuário S - Configuração Y
        // Preenche listas de valores para os dedos polegar e mínimo.
        try
        {
            // String com o comando a ser executado
            string sql = "SELECT * from tab_usuarioS_yy";

            // Instância do comando recebendo como parâmetro a string com o comando (sql) e a
conexão (objcon).
            MySqlCommand cmd = new MySqlCommand(sql, objcon);

            objcon.Open();      // Abre conexão com o banco de dados.
            MySqlDataReader tab = cmd.ExecuteReader();      // Instância
            while (tab.Read())
            {
                // Para cada iteração, é adicionado o valor ao listbox.
                listPolS_configY.Add(Convert.ToDouble(tab["polegar"]));
                listMinS_configY.Add(Convert.ToDouble(tab["minimo"]));
            }
            objcon.Close();      // Fecha a conexão com o banco de dados.
        }
        catch
        {
            MessageBox.Show("Erro: tab_usuarioS_yy!");
        }

        /////////// Cálculos dos valores base: valor médio e desvio padrão (baseado nos valores dos
usuários L, I e S)
        // Configuração média
        for (cont = 0; cont < tam; cont++)
        {
            // Obs.: cont * 3: é pelo fato de que média = somatório de amostras / qtd de amostras.
            //      listPolL_configB[cont] equivale a 3 amostras e assim para as outras listas.
            //      Totalizando 3 + 3 + 3 = 9, no total (considerando tam = 3).
            // Para configuração B
            configMediaB_Pol      =      configMediaB_Pol      +      (listPolL_configB[cont]      +
listPolI_configB[cont] + listPolS_configB[cont]);
            configMediaB_Ind      =      configMediaB_Ind      +      (listIndL_configB[cont]      +
listIndI_configB[cont] + listIndS_configB[cont]);
            configMediaB_Med      =      configMediaB_Med      +      (listMedL_configB[cont]      +
listMedI_configB[cont] + listMedS_configB[cont]);
            configMediaB_Ane      =      configMediaB_Ane      +      (listAneL_configB[cont]      +
listAneI_configB[cont] + listAneS_configB[cont]);
            configMediaB_Min      =      configMediaB_Min      +      (listMinL_configB[cont]      +
listMinI_configB[cont] + listMinS_configB[cont]);

            // Para configuração E
            configMediaE_Pol      =      configMediaE_Pol      +      (listPolL_configE[cont]      +
listPolI_configE[cont] + listPolS_configE[cont]);
            configMediaE_Ind      =      configMediaE_Ind      +      (listIndL_configE[cont]      +
```

```
listIndI_configE[cont] + listIndS_configE[cont]);
            configMediaE_Med = configMediaE_Med + (listMedL_configE[cont] +
listMedI_configE[cont] + listMedS_configE[cont]);
            configMediaE_Ane = configMediaE_Ane + (listAneL_configE[cont] +
listAneI_configE[cont] + listAneS_configE[cont]);
            configMediaE_Min = configMediaE_Min + (listMinL_configE[cont] +
listMinI_configE[cont] + listMinS_configE[cont]);

            // Para configuração Y
            configMediaY_Pol = configMediaY_Pol + (listPolL_configY[cont] +
listPolI_configY[cont] + listPolS_configY[cont]);
            configMediaY_Ind = configMediaY_Ind + (listIndL_configY[cont] +
listIndI_configY[cont] + listIndS_configY[cont]);
            configMediaY_Med = configMediaY_Med + (listMedL_configY[cont] +
listMedI_configY[cont] + listMedS_configY[cont]);
            configMediaY_Ane = configMediaY_Ane + (listAneL_configY[cont] +
listAneI_configY[cont] + listAneS_configY[cont]);
            configMediaY_Min = configMediaY_Min + (listMinL_configY[cont] +
listMinI_configY[cont] + listMinS_configY[cont]);
        }
        configMediaB_Pol = configMediaB_Pol / (cont * 3);
        configMediaB_Ind = configMediaB_Ind / (cont * 3);
        configMediaB_Med = configMediaB_Med / (cont * 3);
        configMediaB_Ane = configMediaB_Ane / (cont * 3);
        configMediaB_Min = configMediaB_Min / (cont * 3);

        // Para configuração E
        configMediaE_Pol = configMediaE_Pol / (cont * 3);
        configMediaE_Ind = configMediaE_Ind / (cont * 3);
        configMediaE_Med = configMediaE_Med / (cont * 3);
        configMediaE_Ane = configMediaE_Ane / (cont * 3);
        configMediaE_Min = configMediaE_Min / (cont * 3);

        // Para configuração Y
        configMediaY_Pol = configMediaY_Pol / (cont * 3);
        configMediaY_Ind = configMediaY_Ind / (cont * 3);
        configMediaY_Med = configMediaY_Med / (cont * 3);
        configMediaY_Ane = configMediaY_Ane / (cont * 3);
        configMediaY_Min = configMediaY_Min / (cont * 3);

        // Somatório para encontrar a variância
        for (cont = 0; cont < tam; cont++)
        {
            // Para configuração B
            varB_Pol[cont] = (Math.Pow(listPolL_configB[cont] - configMediaB_Pol, 2)) +
(Math.Pow(listPolI_configB[cont] - configMediaB_Pol, 2)) + (Math.Pow(listPolS_configB[cont] -
configMediaB_Pol, 2));
            varB_Ind[cont] = (Math.Pow(listIndL_configB[cont] - configMediaB_Ind, 2)) +
(Math.Pow(listIndI_configB[cont] - configMediaB_Ind, 2)) + (Math.Pow(listIndS_configB[cont]
- configMediaB_Ind, 2));
            varB_Med[cont] = (Math.Pow(listMedL_configB[cont] - configMediaB_Med, 2)) +
(Math.Pow(listMedI_configB[cont] - configMediaB_Med, 2)) +
(Math.Pow(listMedS_configB[cont] - configMediaB_Med, 2));
            varB_Ane[cont] = (Math.Pow(listAneL_configB[cont] - configMediaB_Ane, 2)) +
(Math.Pow(listAneI_configB[cont] - configMediaB_Ane, 2)) +
(Math.Pow(listAneS_configB[cont] - configMediaB_Ane, 2));
```

```
            varB_Min[cont] = (Math.Pow(listMinL_configB[cont] - configMediaB_Min, 2)) +
(Math.Pow(listMinI_configB[cont] - configMediaB_Min, 2)) + (Math.Pow(listMinS_configB[cont]
- configMediaB_Min, 2));

            // Para configuração E
            varE_Pol[cont] = (Math.Pow(listPolL_configE[cont] - configMediaE_Pol, 2)) +
(Math.Pow(listPolI_configE[cont] - configMediaE_Pol, 2)) + (Math.Pow(listPolS_configE[cont] -
configMediaE_Pol, 2));
            varE_Ind[cont] = (Math.Pow(listIndL_configE[cont] - configMediaE_Ind, 2)) +
(Math.Pow(listIndI_configE[cont] - configMediaE_Ind, 2)) + (Math.Pow(listIndS_configE[cont]
- configMediaE_Ind, 2));
            varE_Med[cont] = (Math.Pow(listMedL_configE[cont] - configMediaE_Med, 2)) +
(Math.Pow(listMedI_configE[cont] - configMediaE_Med, 2)) +
(Math.Pow(listMedS_configE[cont] - configMediaE_Med, 2));
            varE_Ane[cont] = (Math.Pow(listAneL_configE[cont] - configMediaE_Ane, 2)) +
(Math.Pow(listAneI_configE[cont] - configMediaE_Ane, 2)) + (Math.Pow(listAneS_configE[cont]
- configMediaE_Ane, 2));
            varE_Min[cont] = (Math.Pow(listMinL_configE[cont] - configMediaE_Min, 2)) +
(Math.Pow(listMinI_configE[cont] - configMediaE_Min, 2)) + (Math.Pow(listMinS_configE[cont]
- configMediaE_Min, 2));

            // Para configuração Y
            varY_Pol[cont] = (Math.Pow(listPolL_configY[cont] - configMediaY_Pol, 2)) +
(Math.Pow(listPolI_configY[cont] - configMediaY_Pol, 2)) + (Math.Pow(listPolS_configY[cont] -
configMediaY_Pol, 2));
            varY_Ind[cont] = (Math.Pow(listIndL_configY[cont] - configMediaY_Ind, 2)) +
(Math.Pow(listIndI_configY[cont] - configMediaY_Ind, 2)) + (Math.Pow(listIndS_configY[cont]
- configMediaY_Ind, 2));
            varY_Med[cont] = (Math.Pow(listMedL_configY[cont] - configMediaY_Med, 2)) +
(Math.Pow(listMedI_configY[cont] - configMediaY_Med, 2)) +
(Math.Pow(listMedS_configY[cont] - configMediaY_Med, 2));
            varY_Ane[cont] = (Math.Pow(listAneL_configY[cont] - configMediaY_Ane, 2)) +
(Math.Pow(listAneI_configY[cont] - configMediaY_Ane, 2)) + (Math.Pow(listAneS_configY[cont]
- configMediaY_Ane, 2));
            varY_Min[cont] = (Math.Pow(listMinL_configY[cont] - configMediaY_Min, 2)) +
(Math.Pow(listMinI_configY[cont] - configMediaY_Min, 2)) + (Math.Pow(listMinS_configY[cont]
- configMediaY_Min, 2));
        }

        for (cont = 0; cont < tam; cont++)
        {
            // Para configuração B
            variancB_Pol = variancB_Pol + varB_Pol[cont];
            variancB_Ind = variancB_Ind + varB_Ind[cont];
            variancB_Med = variancB_Med + varB_Med[cont];
            variancB_Ane = variancB_Ane + varB_Ane[cont];
            variancB_Min = variancB_Min + varB_Min[cont];

            // Para configuração E
            variancE_Pol = variancE_Pol + varE_Pol[cont];
            variancE_Ind = variancE_Ind + varE_Ind[cont];
            variancE_Med = variancE_Med + varE_Med[cont];
            variancE_Ane = variancE_Ane + varE_Ane[cont];
            variancE_Min = variancE_Min + varE_Min[cont];

            // Para configuração Y
```

```
    variancY_Pol = variancY_Pol + varY_Pol[cont];
    variancY_Ind = variancY_Ind + varY_Ind[cont];
    variancY_Med = variancY_Med + varY_Med[cont];
    variancY_Ane = variancY_Ane + varY_Ane[cont];
    variancY_Min = variancY_Min + varY_Min[cont];
}

// Variância
// Para configuração B
varianciaB_Pol = variancB_Pol / (cont * 3);
varianciaB_Ind = variancB_Ind / (cont * 3);
varianciaB_Med = variancB_Med / (cont * 3);
varianciaB_Ane = variancB_Ane / (cont * 3);
varianciaB_Min = variancB_Min / (cont * 3);

// Para configuração E
varianciaE_Pol = variancE_Pol / (cont * 3);
varianciaE_Ind = variancE_Ind / (cont * 3);
varianciaE_Med = variancE_Med / (cont * 3);
varianciaE_Ane = variancE_Ane / (cont * 3);
varianciaE_Min = variancE_Min / (cont * 3);

// Para configuração Y
varianciaY_Pol = variancY_Pol / (cont * 3);
varianciaY_Ind = variancY_Ind / (cont * 3);
varianciaY_Med = variancY_Med / (cont * 3);
varianciaY_Ane = variancY_Ane / (cont * 3);
varianciaY_Min = variancY_Min / (cont * 3);

// Desvio padrão
// Para configuração B
dpB_Pol = Math.Sqrt(varianciaB_Pol);
dpB_Ind = Math.Sqrt(varianciaB_Ind);
dpB_Med = Math.Sqrt(varianciaB_Med);
dpB_Ane = Math.Sqrt(varianciaB_Ane);
dpB_Min = Math.Sqrt(varianciaB_Min);

// Para configuração E
dpE_Pol = Math.Sqrt(varianciaE_Pol);
dpE_Ind = Math.Sqrt(varianciaE_Ind);
dpE_Med = Math.Sqrt(varianciaE_Med);
dpE_Ane = Math.Sqrt(varianciaE_Ane);
dpE_Min = Math.Sqrt(varianciaE_Min);

// Para configuração Y
dpY_Pol = Math.Sqrt(varianciaY_Pol);
dpY_Ind = Math.Sqrt(varianciaY_Ind);
dpY_Med = Math.Sqrt(varianciaY_Med);
dpY_Ane = Math.Sqrt(varianciaY_Ane);
dpY_Min = Math.Sqrt(varianciaY_Min);

// Limites
// Configuração B
// Limite inferior
limInfB_Pol = configMediaB_Pol - dpB_Pol;
limInfB_Ind = configMediaB_Ind - dpB_Ind;
limInfB_Med = configMediaB_Med - dpB_Med;
```

```
            limInfB_Ane = configMediaB_Ane - dpB_Ane;
            limInfB_Min = configMediaB_Min - dpB_Min;
            // Limite superior
            limSupB_Pol = configMediaB_Pol + dpB_Pol;
            limSupB_Ind = configMediaB_Ind + dpB_Ind;
            limSupB_Med = configMediaB_Med + dpB_Med;
            limSupB_Ane = configMediaB_Ane + dpB_Ane;
            limSupB_Min = configMediaB_Min + dpB_Min;

            // Configuração E
            // Limite inferior
            limInfE_Pol = configMediaE_Pol - dpE_Pol;
            limInfE_Ind = configMediaE_Ind - dpE_Ind;
            limInfE_Med = configMediaE_Med - dpE_Med;
            limInfE_Ane = configMediaE_Ane - dpE_Ane;
            limInfE_Min = configMediaE_Min - dpE_Min;
            // Limite superior
            limSupE_Pol = configMediaE_Pol + dpE_Pol;
            limSupE_Ind = configMediaE_Ind + dpE_Ind;
            limSupE_Med = configMediaE_Med + dpE_Med;
            limSupE_Ane = configMediaE_Ane + dpE_Ane;
            limSupE_Min = configMediaE_Min + dpE_Min;

            // Configuração Y
            // Limite inferior
            limInfY_Pol = configMediaY_Pol - dpY_Pol;
            limInfY_Ind = configMediaY_Ind - dpY_Ind;
            limInfY_Med = configMediaY_Med - dpY_Med;
            limInfY_Ane = configMediaY_Ane - dpY_Ane;
            limInfY_Min = configMediaY_Min - dpY_Min;
            // Limite superior
            limSupY_Pol = configMediaY_Pol + dpY_Pol;
            limSupY_Ind = configMediaY_Ind + dpY_Ind;
            limSupY_Med = configMediaY_Med + dpY_Med;
            limSupY_Ane = configMediaY_Ane + dpY_Ane;
            limSupY_Min = configMediaY_Min + dpY_Min;

            ////////// Armanenando valor médio, desvio padrão e limites no banco de dados
            //// Configuração B
            try
            {
                objcon.Open();          // Abre conexão com o banco de dados.

                // Query SQL
                MySqlCommand cmdB = new MySqlCommand("INSERT INTO
tab_valores_configB (vmPol, vmInd, vmMed, vmAne, vmMin, " +
                    "dp_Pol, dp_Ind, dp_Med, dp_Ane, dp_Min, " +
                    "limInf_Pol, limInf_Ind, limInf_Med, limInf_Ane, limInf_Min, " +
                    "limSup_Pol, limSup_Ind, limSup_Med, limSup_Ane, limSup_Min)" +
                    "    VALUES    ('"    +    Convert.ToString(configMediaB_Pol)    +    "','"    +
Convert.ToString(configMediaB_Ind) + "','" +
                    Convert.ToString(configMediaB_Med)              +              "','"              +
Convert.ToString(configMediaB_Ane) + "','" +
                    Convert.ToString(configMediaB_Min) + "','" +
                    Convert.ToString(dpB_Pol) + "','" + Convert.ToString(dpB_Ind) + "','" +
                    Convert.ToString(dpB_Med) + "','" + Convert.ToString(dpB_Ane) + "','" +
                    Convert.ToString(dpB_Min) + "','" +
```

```
                Convert.ToString(limInfB_Pol) + "','" + Convert.ToString(limInfB_Ind) + "','" +
                Convert.ToString(limInfB_Med) + "','" + Convert.ToString(limInfB_Ane) + "','"
+
                Convert.ToString(limInfB_Min) + "','" +
                Convert.ToString(limSupB_Pol) + "','" + Convert.ToString(limSupB_Ind) + "','"
+
                Convert.ToString(limSupB_Med) + "','" + Convert.ToString(limSupB_Ane) + "','"
+
                Convert.ToString(limSupB_Min) + "')", objcon);

            cmdB.ExecuteNonQuery();      //Executa a Query SQL
            objcon.Close();              // Fecha a conexão com o banco de dados.

            //MessageBox.Show("Conectado com o banco de dados!");
        }
        catch
        {
            MessageBox.Show("Não conectou com o banco de dados! - Configuração B");
        }
        // Configuração E
        try
        {
            objcon.Open();          // Abre conexão com o banco de dados.

            // Query SQL
            MySqlCommand cmdE = new MySqlCommand("INSERT INTO
tab_valores_configE (vmPol, vmInd, vmMed, vmAne, vmMin, " +
                "dp_Pol, dp_Ind, dp_Med, dp_Ane, dp_Min, " +
                "limInf_Pol, limInf_Ind, limInf_Med, limInf_Ane, limInf_Min, " +
                "limSup_Pol, limSup_Ind, limSup_Med, limSup_Ane, limSup_Min)" +
                "    VALUES    ('"    +    Convert.ToString(configMediaE_Pol)    +    "','"    +
Convert.ToString(configMediaE_Ind) + "','" +
                Convert.ToString(configMediaE_Med)              +              "','"              +
Convert.ToString(configMediaE_Ane) + "','" +
                Convert.ToString(configMediaE_Min) + "','" +
                Convert.ToString(dpE_Pol) + "','" + Convert.ToString(dpE_Ind) + "','" +
                Convert.ToString(dpE_Med) + "','" + Convert.ToString(dpE_Ane) + "','" +
                Convert.ToString(dpE_Min) + "','" +
                Convert.ToString(limInfE_Pol) + "','" + Convert.ToString(limInfE_Ind) + "','" +
                Convert.ToString(limInfE_Med) + "','" + Convert.ToString(limInfE_Ane) + "','"
+
                Convert.ToString(limInfE_Min) + "','" +
                Convert.ToString(limSupE_Pol) + "','" + Convert.ToString(limSupB_Ind) + "','"
+
                Convert.ToString(limSupE_Med) + "','" + Convert.ToString(limSupE_Ane) + "','"
+
                Convert.ToString(limSupE_Min) + "')", objcon);

            cmdE.ExecuteNonQuery();      //Executa a Query SQL
            objcon.Close();              // Fecha a conexão com o banco de dados.

            //MessageBox.Show("Conectado com o banco de dados!");
        }
        catch
        {
            MessageBox.Show("Não conectou com o banco de dados! - Configuração E");
```

```
        }
        // Configuração Y
        try
        {
            objcon.Open();          // Abre conexão com o banco de dados.

            // Query SQL
            MySqlCommand cmdY = new MySqlCommand("INSERT INTO
tab_valores_configY (vmPol, vmInd, vmMed, vmAne, vmMin, " +
                "dp_Pol, dp_Ind, dp_Med, dp_Ane, dp_Min, " +
                "limInf_Pol, limInf_Ind, limInf_Med, limInf_Ane, limInf_Min, " +
                "limSup_Pol, limSup_Ind, limSup_Med, limSup_Ane, limSup_Min)" +
                "    VALUES    ('"    +    Convert.ToString(configMediaY_Pol)    +    "','"    +
Convert.ToString(configMediaY_Ind) + "','" +
                Convert.ToString(configMediaY_Med)              +              "','"              +
Convert.ToString(configMediaY_Ane) + "','" +
                Convert.ToString(configMediaY_Min) + "','" +
                Convert.ToString(dpY_Pol) + "','" + Convert.ToString(dpY_Ind) + "','" +
                Convert.ToString(dpY_Med) + "','" + Convert.ToString(dpY_Ane) + "','" +
                Convert.ToString(dpY_Min) + "','" +
                Convert.ToString(limInfY_Pol) + "','" + Convert.ToString(limInfY_Ind) + "','" +
                Convert.ToString(limInfY_Med) + "','" + Convert.ToString(limInfY_Ane) + "','" +
                Convert.ToString(limInfY_Min) + "','" +
                Convert.ToString(limSupY_Pol) + "','" + Convert.ToString(limSupY_Ind) + "','"
+
                Convert.ToString(limSupY_Med) + "','" + Convert.ToString(limSupY_Ane) + "','"
+
                Convert.ToString(limSupY_Min) + "')", objcon);

            cmdY.ExecuteNonQuery();      //Executa a Query SQL
            objcon.Close();             // Fecha a conexão com o banco de dados.

            //MessageBox.Show("Conectado com o banco de dados!");
        }
        catch
        {
            MessageBox.Show("Não conectou com o banco de dados! - Configuração Y");
        }
    }
  }
}
```

APÊNDICE C

Segue o algoritmo feito quanto à criação e manipulação do banco de dados através do software MySQL Workbench:

```
-- Criacao do banco de dados
create database bd_pg
default character set utf8
default collate utf8_general_ci;

-- Usando bd_pg
use bd_pg;

-- Criando as tabelas
-- Valores corretos de indicador, medio e anelar
create table tab_nomedousuario_configB (        -- Configuracao B
polegar varchar(10),
indicador varchar(10),
medio varchar(10),
anelar varchar(10),
minimo varchar(10)
);
-- Valores corretos de polegar e minimo
create table tab_nomedousuario_configBB (       -- Configuracao B
polegar varchar(10),
indicador varchar(10),
medio varchar(10),
anelar varchar(10),
minimo varchar(10)
);

-- Valor medio, desvio padrao, limites inferior e superior de cada dedo
create table tab_valores_config (
vmPol varchar(10),
vmInd varchar(10),
vmMed varchar(10),
vmAne varchar(10),
vmMin varchar(10),
dp_Pol varchar(10),
dp_Ind varchar(10),
dp_Med varchar(10),
dp_Ane varchar(10),
dp_Min varchar(10),
limInf_Pol varchar(10),
limInf_Ind varchar(10),
limInf_Med varchar(10),
limInf_Ane varchar(10),
limInf_Min varchar(10),
limSup_Pol varchar(10),
limSup_Ind varchar(10),
limSup_Med varchar(10),
limSup_Ane varchar(10),
limSup_Min varchar(10)
);
```

```
-- Ver valores da tabela
select * form tab_nomedousuario_configB;
```