Anderson Henrique Dantas Borba

Proposta de luva háptica wi-fi para interação com ambientes virtuais

Vitória - ES

Julho de 2017

Anderson Henrique Dantas Borba

Proposta de luva háptica wi-fi para interação com ambientes virtuais

Parte manuscrita do Projeto de Graduação do aluno Anderson Henrique Dantas Borba, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Universidade Federal do Espírito Santo – UFES

Departamento de Engenharia Elétrica

Programa de Graduação

Orientador: Prof. Dr. Anselmo Frizera Neto

Vitória - ES

Julho de 2017

Anderson Henrique Dantas Borba

Proposta de luva háptica wi-fi para interação com ambientes virtuais

Parte manuscrita do Projeto de Graduação do aluno Anderson Henrique Dantas Borba, apresentado ao Departamento de Engenharia Elétrica do Centro Tecnológico da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do grau de Engenheiro Eletricista.

Trabalho aprovado. Vitória - ES, Julho de 2017:

Prof. Dr. Anselmo Frizera Neto
Orientador

Prof. Dr. André Ferreira
Departamento de Eng. Elétrica
UFES

Prof. Dr. Thiago Oliveira dos Santos
Departamento de Informática
UFES

Vitória - ES
Julho de 2017

Este trabalho é dedicado aos que arriscam sabendo dos riscos

Agradecimentos

Agradeço à oportunidade de ter começado e concluído este trabalho e a todos que me emprestaram imenso auxílio em chegar a este ponto. Agradeço à banca por aceitar o convite, a meus pais por não só me conceberem, mas por me apoiarem durante todo o caminho. Agradeço também à Instituição da Universidade Federal do Espírito Santo por me proporcionar conhecimento e despertar minha curiosidade.

Agradeço, especialmente a meu orientador, Anselmo Frizera Neto, pela paciência e disposição em me orientar; a Thomaz Bothelho pelas conversas, toques e dicas; às minhas amigas, Paula Sarmenghi e Letícia Munhoz, pela força e companheirismo nas matérias que nos fizeram passar noites estudando; a Filipe Demuth pelas opiniões e auxílio com questões práticas e mecânicas; a Franco Vieira por me ensinar sobre o filtro capacitivo digital on-line (Filtro linear de primeira ordem).

"And on the pedestal these word appear:
"My name is Ozymandias, king of kings:
Look on my works, ye Mighty, and despair"
Nothing beside [the pedestal] remains. Round the decay
Of that colossal wreck, boundless and bare.
The lone and level sands stretch far away."
Percy Sherlley - Ozymandias

Resumo

Há um crescente interesse industrial e social no desenvolvimento de dispositivos relacionados à realidade virtual. Esse interesse pode ser justificado pelas aplicações de tais sistemas em pesquisa e desenvolvimento tecnológico em diversas áreas como a teleoperação, o aprendizado com reforço sensorial, reabilitação robótica, engenharia biomédica, desenvolvimento lúdico, prototipagem de produtos e desenvolvimento arquitetônico. Entretanto, para alcançar o objetivo de explorar aplicações em realidade virtual é necessário estabelecer uma infraestrutura de mecanismos vestíveis que possibilitem uma interação entre o usuário e o ambiente virtual. Este trabalho teve por objetivo desenvolver um protótipo de uma luva de realidade virtual, seus mecanismos e funcionamento. O sistema foi validado qualitativamente com testes de funcionalidade de cada um de seus subsistemas, sendo necessário estudos futuros para levantamento de características quantitativas de interesse e métodos padrões para teste destas características.

Palavras-chave: *Flexsensors*. Realidade Virtual. *Data Glove*. *Feedback* háptico

Abstract

There's an increasing social and industrial interest in the development of devices related to virtual reality. This interest can be justified by the application of these systems in technological research and development in many fields such as teleoperation, sensorially enhanced learning, robotic rehabilitation, biomedical engineering, game development, product prototyping and architectonical development. But in order to achieve the objective of exploring applications in virtual reality it is necessary to establish an infrastructure of wearable devices to enable the interaction between the user and the virtual environment. This graduation project had as the objective the development of a virtual reality glove prototype, its mechanisms and functionalities. The system as a whole was validated by testing each of its subsystems qualitatively, with further research being necessary in order to determine quantitative characteristics of interest and standard methods to test these characteristics.

Keywords: Flexsensors. Virtual Reality. Data Glove. Haptic Feedback.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ERM	Motor de eixo excêntrico
VR	Realidade Virtual
HMD	Display montado na cabeça

Sumário

1 Introdução

1.1 Motivação

O ano de 2016 foi um marco no contexto do tema Realidade Virtual (*VR*). Eventos tais quais a *Oculus Connect 3*, o *Steam Dev Days* e a *Vision Summit 2016*, foram ocasiões em que os grandes nomes da indústria, juntamente com a comunidade de desenvolvedores e entusiastas, puderam se reunir e expor a que ponto a realidade virtual chegou, quais são as dificuldades e pesquisas que estão sendo abordadas e qual é a visão de cada segmento para o futuro.

Dentre os diversos tópicos discutidos nas conferências houve a apresentação dos dispositivos de interação com ambientes virtuais desenvolvidos pelas empresas de maior destaque no campo de desenvolvimento comercial de aparelhos de realidade virtual (*Oculus*, *Samsung*, *HTC*, *Sony*). Os aparelhos apresentados são sistemas de interação baseados em alavancas e gatilhos com rastreamento de posição e orientação. A empresa *Oculus* disponibilizou comercialmente em dezembro de 2016 o *Oculus Touch* (Figura 1), um par de *joysticks* rastreados por câmeras infravermelho e algoritmos de visão computacional. A *HTC* disponibilizou comercialmente o sistema Vive em abril de 2016, que apresentava um *Head mounted Display* (*HMD*), um par de *joystics* e seu sistema de rastreamento chamado de *Light House* (Figura 2), que consiste em um par de pequenos aparelhos cuja única função é emitir sinais em infravermelho de tal forma que o *joystick*, equipado com sensores infravermelho, consegue processar os disparos de cada aparelho e determinar sua posição em relação a cada aparelho e enviá-la ao computador conseguindo assim se posicionar no ambiente. Já a *Sony* utlizou o *playstation move* (Figura 3) como aparelho de *input* para sua aplicações em realidade virtual. Este aparelho consiste em um par de *joystick* com uma esfera de borracha contendo um led RGB em seu centro, permitindo que a esfera apresente cores diferentes. Uma câmera utiliza visão computacional para posicionar o *playstation move* no mundo.

Estas grandes empresas tiveram como foco inicial o desenvolvimento de *Head Mounted Displays* e portanto focam em chegar em um ponto de desenvolvimento aceitável destes aparelhos que por si só apresentam grandes desafios de desenvolvimento. Entretanto empresas menores estão se dedicando a tornar luvas de interação com ambientes virtuais seu carro chefe. Esse é o caso das empresas *Neurodigital Technologies*, *Manus VR* e a *Dextra Robotics*. Cada uma destas empresas está trabalhando em disponibilizar ou já disponibilizou um aparelho comercial (Figuras 4, 5 e 6), porém estes aparelhos ainda são muito caros e novos, apresentando eventuais problemas não detectados em testes internos, mas relatados pelos *early adopter* e desenvolvedores de aplicações.

Figura 1 – Sistema de realidade virtual *Oculus Rift* e *Oculus Touch*.

fonte: https://www.octopusrift.com/mounting-touch-cameras/

Figura 2 – Sistema de realidade virtual *HTC Vive* com o *Light house*.

fonte: https://www.roadtovr.com/htc-vive-weight-15-percent-lighter-than-original-headset-vs-oculus-rift-comparison/

A área de realidade virtual é extensa, podendo ser segmentada em subáreas. Para efeitos deste trabalho, o autor a segmentou por seus produtos modulares:

- *HDMs* (*Head Mounted Displays*) são o equivalente aos atuais monitores para computadores pessoais. Estes são os equipamentos responsáveis pela imersão visual do usuário em ambientes virtuais.
- *Headsets* ou fones de ouvido são os aparelhos responsáveis pela imersão auditiva do usuário.
- *Input Devices* são os aparelhos pelos quais um usuário é capaz de agir sobre o ambiente virtual, podendo ser caracterizados como elementos simplesmente ativos

Figura 3 – Sistema de realidade virtual *Playstation VR*.

fonte: https://pt.wikipedia.org/wiki/PlayStation_VR

Figura 4 – Luva de dados *Glove One* da *Neurodigital Technologies*.

fonte: https://www.neurodigital.es/

ou interativos (onde há algum tipo de *feedback* do aparelho para o usuário).

- *Infraestrutura* São todos os sistemas que interligam o *HMD*, *Headset* e *Input devices* com as aplicações em *VR*.
- *Aplicações*, que se utilizam das demais subáreas acrescida de uma programação específica para realizar uma função que seja de serventia a um usuário.

A subárea de *Input Devices* para *VR* tem sua importância no molde de diversas aplicações, dentre as quais estão reabilitação biomédica (HODA; HAFIDH; SADDIK, 2013; JACK et al., 2001), arquitetura, teleoperação de equipamentos (ARIYANTO et al., 2017; FANG et al., 2016; LOW et al., 2016; WANG; ZHANG, 2016; PITITEERAPHAB, 2015; JAIJONGRAK; CHANTASUBAN; THIEMJARUS, 2009), educação, área de desenvolvimento lúdico (JOFRE et al., 2016), desenvolvimento de ambientes colaborativos de prototipagem, modelagem tridimensional, ambientes de telepresença, tecnologias assistenciais (MAJEAU et al., 2011), entre outras. Portanto, estes aparelhos devem ser desenvolvidos

Figura 5 – Luva de dados da empresa *Manus VR*.

fonte: https://manus-vr.com/press/

Figura 6 – Exoesqueleto *Dexmo* da *Dextra Robotics*.

fonte: http://www.dextarobotics.com/

para atender estas diferentes áreas de aplicações, cumprindo requisitos como, por exemplo, critérios de vestibilidade (facilidade de por e remover o aparelho), mensuração de diferentes níveis de *Input* e atender a requisitos específicos da aplicação.

Um ponto interessante a ressaltar é a existência de dispositivos de *feedback* vestíveis, como luvas puramente hápticas, por exemplo. Enquanto que estes dispositivos podem apresentar utilidades em diferentes aplicações, tais quais diagnóstico médico (RINDERKNECHT et al., 2015), tecnologia assistencial (KEYES; D'SOUZA; POSTULA, 2015) e complemento de experiências em Realidade Virtual, estes aparelhos não podem ser classifi-

cados como *Input Devices*, uma vez que sua função é de possibilitar a ação de um sistema sobre o usuário, independente do *Input* do usuário sobre o sistema.

1.2 Objetivos Gerais

Este trabalho teve por objetivo o desenvolvimento de um protótipo de um sistema de sensores vestível para aquisição de dados referente ao posicionamento e orientação da mão direita do usuário, a flexão de cada dedo desta mão e possibilitar o fornecimento de *feedback* háptico para cada falange distal. O protótipo deveria realizar a troca de dados com o computador por um sistema sem fio.

1.3 Objetivos Específicos

Os objetivos específicos deste projeto foram:

- O hardware do dispositivo deve estar com seu módulo principal (placa de circuito impresso com conversor analógico-digital e microcontrolador) construído, permitindo o acoplamento de uma bateria externa e com pelo menos um sensor e um atuador funcionais.
- O microcontrolador deve receber os dados do conversor analógico-digital, realizar quaisquer pré-processamentos que forem necessários e enviar os dados pré-processados para a máquina que tenha estabelecido conexão com o sistema. A forma de envio dos dados deve ocorrer mediante uma requisição do sistema. A rede a qual o microcontrolador e o computador irão usar para comunicação é uma rede Wi-Fi. A comunicação será feita utilizando o protocolo *UDP*. O firmware também deve permitir a recepção de comandos para acionar ou desativar os atuadores.
- A API do sistema deve estar implementada de forma que um programador tenha acesso aos dados: posição e orientação da mão do usuário e grau de flexão dos dedos cujos sensores estão implementados. A API deve disponibilizar ao programador a possibilidade de acionar os atuadores que estiverem implementados.
- A aplicação exemplo irá fazer uso dos dados do sistema vestível e poderá acionar os atuadores do sistema apenas se o protótipo estiver sendo reconhecido pelo sensor de rastreamento de mãos.

1.4 Justificativa

Esse projeto tem a importância de apresentar e disponibilizar um dispositivo vestível para membros superiores com a finalidade de adquirir, processar, e utilizar dados relativos

à configuração espacial das mãos do usuário para interação dentro de um ambiente virtual com a possibilidade de realização de *feedback* háptico sobre as falanges distais do usuário.

Pesquisas sobre o tema de tecnologias vestíveis estão sendo realizadas dentro da instituição UFES e abrangem diversas áreas de pesquisa, tais quais redes de sensores, redes de atuadores, sistemas embarcados, processamento de dados, computação gráfica, realidade virtual e ergonomia; apresentando também possibilidades de aplicação em projetos de pesquisa futuros como telerreabilitação, reabilitação assistida por máquinas, teleoperação, desenvolvimento lúdico, estudos de cinemática e dinâmica do movimento. O protótipo desenvolvido pode vir a servir como ferramenta de auxílio à pesquisa.

1.5 O Conteúdo Deste Trabalho

O capítulo 2 apresenta o referencial teórico a respeito de luvas de dados e sistemas hápticos desenvolvidos desde seu surgimento no final da década de 70 até os dias atuais com base na literatura pesquisada e na revisão realizada por Sturman e Zeltzer (1994).

O capítulo 3 se refere aos materiais e métodos utilizados para a confecção do sistema de interação, bem como os processos utilizados para a criação de sua aplicação exemplo.

O capítulo 4 expõe o sistema criado, suas variáveis mensuráveis e o resultado dos testes realizados sobre o sistema em conjunto com sua aplicação, ressaltando que os testes não foram exaustivos e o propósito de cada teste é apresentar um ponto de referência sobre o qual o sistema pode ser comparado em trabalhos futuros.

O capítulo 5 discute sobre o trabalho bem como quais seriam as possíveis pesquisas futuras que poderiam lhe dar continuidade.

O capítulo 6 apresenta as conclusões do autor sobre o estado final do sistema.

Por fim, são apresentados os elementos pós-textuais. Estes elementos são as referências bibliográficas do trabalho e os apêndices, onde estão dispostos os esquemáticos e códigos desenvolvidos.

2 Referencial Teórico

A literatura apresenta diversos trabalhos sobre *inputs* baseados em luvas de dados e sistemas hápticos, não havendo consenso ou padrões sobre a infraestrutura de sistema a ser implementado devido majoritariamente aos requisitos variantes de aplicações a que se destina um sistema de sensores ou outro.

2.1 Análise Tecnológica

Uma análise das tecnologias utilizadas nos últimos dezessete anos por pesquisadores revelou que o uso de *flexsensors* para rastreamento da posição das falanges com relação à palma é relativamente comum (ARIYANTO et al., 2017; LOW et al., 2016; PITITEERAPHAB, 2015; FLORES et al., 2014; KUMAR; RAUTARAY; AGRAWAL, 2012; JAIJONGRAK; CHANTASUBAN; THIEMJARUS, 2009). No entanto outros meios de mensuração da posição das falanges são usados. Dentro da literatura pesquisada está o uso de *IMU's* (*Inertial Measurement Units*) (FANG et al., 2016; WANG; ZHANG, 2016), sensores ópticos (MAJEAU et al., 2011), elastômeros eletrocondutores (TOGNETTI et al., 2006) e *strain gauges* (TARCHANIDIS; LYGOURAS, 2003). Outro sensor que foi estudado como uma alternativa para realizar o mesmo papel é o *strech sensor*, cuja resistencia varia de acordo com o tensionamento do filamento. O trabalho de Sbernini, Pallotti e Saggio (2016) compara *flex sensors* com *strech sensors* (ambos os sensores podem ser utilizados para mensurar posicionamento das falanges) e apresenta como resultados que o *strech sensor* apresenta variações de resistências menores, com histerese mais pronunciada, porém suas curvas de variação de resistência em relação à deformação são mais lineares que as do outro sensor. Por fim, um segundo estudo (SHEN et al., 2016) propõe o uso de um elastômero como um sensor de flexão deformável, podendo ser esticado e comprimido para se adequar às deformações da pele quando uma junta flexiona ou extende-se.

Com relação aos sistemas de processamento embarcados em luvas, há um uso comum de microcontroladores *Arduino* (ARIYANTO et al., 2017; ALVES; SANTOS, 2016; PITITEERAPHAB, 2015; HODA; HAFIDH; SADDIK, 2013) e similares (*Zigduino* e *gisDuino*) (KEYES; D'SOUZA; POSTULA, 2015; FLORES et al., 2014). Porém Fang et al. (2016) utilizou um *STM32F4* e Tarchanidis e Lygouras (2003) implementou seu sistema com dois *PIC16C74A*. Não foram encontrados registros de uso do microcontrolador *ESP8266* para implementação de dispositivos relacionados com realidade virtual.

Quanto à interface háptica, há uma diversidade de aparatos mecânicos que foram desenvolvidos com o intuito de prover sensações tácteis ou de restrição de movimentos. Dentre estes aparatos estão atuadores pneumáticos (LOW et al., 2016), estimulação

elétrica funcional (*FES*) (WANG; ZHANG, 2016), pistões pneumáticos (BOUZIT et al., 2002; JACK et al., 2001), motores de massa de rotação excêntrica (KEYES; D'SOUZA; POSTULA, 2015; RINDERKNECHT et al., 2015; HODA; HAFIDH; SADDIK, 2013) e atuadores cujo princípio de funcionamento é o *Jamming principle* (princípio em que um material particulado tal qual areia ou material pulverizado apresentam comportamento similar a fluidos, porém quando submetidos a forças externas, seu comportamento torna-se sólido.) (ZUBRYCKI; GRANOSIK, 2015).

Nenhum dos sistemas *wireless* da literatura pesquisada ((ALVES; SANTOS, 2016; HODA; HAFIDH; SADDIK, 2013; KEYES; D'SOUZA; POSTULA, 2015; FANG et al., 2016; FLORES et al., 2014; JAIJONGRAK; CHANTASUBAN; THIEMJARUS, 2009; KUMAR; RAUTARAY; AGRAWAL, 2012; PITITEERAPHAB, 2015)) apresentou um protocolo ou uma proposta de protocolo de padronização dos dados trocados entre o sistema *wireless* e o computador.

2.2 Análise Histórica

Analisando a literatura estudada, observa-se que o trabalho mais antigo nela contido sobre luvas de dados data de 1994 (o trabalho mais antigo sobre sistemas hápticos estudado data de 2001), enquanto que o trabalho mais antigo citado sobre o tema data de 1977, referindo-se à primeira luva de dados patenteada: a *Sayre Glove*, que utilizava tubos flexíveis, LEDs e fotocélulas para mensurar a flexão dos dedos. Isto significa que a campo de estudo e desenvolvimento de luvas como meio de *input* de sistemas computacionais existe há quarenta anos e, de acordo com Sturman e Zeltzer (1994), muitos dos atuais focos da área tais quais teleoperação, desenvolvimento lúdico, apoio à saúde e visualização científica também eram áreas de interesse desde o final da década de 70. Entre 1977 e 1994 tecnologias de sensoriamento e rastreamento foram desenvolvidas e utilizadas em diferentes luvas de dados. Sturman e Zeltzer (1994) subdividem essas tecnologias em dois subconjuntos: (I) Tecnologias de rastreamento de posição da mão, que poderia ser realizado por meio óptico, magnético ou acústico; e (II) "Tecnologias de luva", que seria todo o aparelho eletromecânico posicionado sobre mãos e dedos usado para determinar a configuração da mão.

O rastreamento por meio óptico envolve utilização de marcadores infravermelhos tais quais LEDs ou adesivos que reflitam luz infravermelha e câmeras espalhadas pelo ambiente. Em 1994 ainda não existiam algoritmos de visão computacional que conseguissem fazer processamentos on-line de captura de movimentos (STURMAN; ZELTZER, 1994), a taxa de captura de uma câmera convencional chegava a 60 quadros por segundo, uma taxa insuficiente para rastrear movimentos rápidos de dedos e mãos; a resolução da câmera comum ainda era baixa para realizar dissociação de dedos e abranger o campo de

movimentação das mãos. Outro problema da utilização de métodos ópticos é a oclusão, especialmente de dedos.

O rastreamento magnético não é restrito por campo de visão, sua taxa de aquisição de dados era maior que sistemas ópticos, mas essas soluções eram cabeadas e a existência de outros objetos metálicos ou a interferência de campos eletromagnéticos externos causavam leituras errôneas nos aparelhos.

Os sistemas acústicos consistiam em emissores e sensores de ultrassom espalhados pelo ambiente enviando pulsos de ultrassom para localizar objetos no espaço.

Oito luvas de dados foram abordadas no trabalho Sturman e Zeltzer (1994), dentre elas está a já mencionada *Sayre Glove*, a primeira luva de dados; a *Dextrous HandMaster*, um exoesqueleto para as mãos, sendo o mais preciso e recente dentre as luvas contidas no estudo, com capacidade de medir 20 graus de liberdade da mão (quatro graus para cada dedo) com precisão de dados de flexão de um grau a uma taxa de amostragem de 200 amostras por segundo; e a *Power Glove*, a luva de dados comercialmente menos custosa (seu preço chegava a ser 100 vezes menor que as demais) e a menos precisa, com apenas dois bits de resolução para cada um de seus seis *flexsensors* (dois deles eram posicionados sobre o polegar) e uma taxa de amostragem de 30 amostras por segundo.

A partir dos anos 2000 houve um avanço em sistemas embarcados e popularização dos microcontroladores *PIC*, que posteriormente vieram a ter seu mercado disputado por outros microcontroladores como a plataforma de desenvolvimento *Arduino* e as famílias de microcontroladores da *Texas instruments* (*MSP* e *Tiva* dentre outros). Em 2003 o trabalho Tarchanidis e Lygouras (2003) dissertou sobre uma luva operada por dois microcontroladores *PIC16C74A* (um mestre e um escravo), que utilizava *strain gauges* ligados a amplificadores operacionais resultando em sistema cabeado que requer calibração e cuja finalidade é a medição da força aplicada pelo dedo sobre a luva, apresentando uma resolução de 0,38N e uma sensitividade de 0,05V/N e leitura máxima de até 100N.

Em 2006 Tognetti et al. (2006) deu continuidade a um estudo anterior do mesmo grupo aplicando uma malha de um elastômero eletrocondutivo com propriedades piezoresistivas desenvolvido pelo grupo em uma proposta de luva de dados onde foram impressos 20 sensores para medir 10 graus de liberdade dos dedos da mão (2 graus para cada dedo). No entanto, dos 20 sensores impressos sobre a luva apenas 15 foram utilizados devido a uma restrição na quantidade de canais de conversão A/D que o sistema do grupo possuía. A deformação que os sensores da luva sofrem não está associada a um único grau de liberdade e por isso um mapeamento de uma função que correlacione a deformação em cada sensor com cada um dos graus de liberdade foi necessária. Para realizar o mapeamento foi utilizada uma luva de dados comercial (a *Cyber Glove*) sobre a luva com o elastômero para fazer uma correlação entre os dados de ambas as luvas. Os métodos usados para realizar a correlação foram o método dos mínimos quadrados e o treinamento de uma rede

neural de múltiplas camadas *feedforward* e *back propagation* com 15 camadas de entrada tan-sigmóide, 30 camadas intermediárias (ocultas) tan-sigmóide e 10 camadas lineares de saída. Os resultados com a rede neural aparentam ser pouco melhores que os obtidos pelo método dos mínimos quadrados e ambos são considerados satisfatórios pelos autores do artigo no sentido de que ambos geram resultados próximos às medidas obtidas com a *Cyber Glove*. Entretanto, uma questão que pode ser levantada sobre esse trabalho é o uso de uma luva de dados para calibrar outra luva de dados, o que restringe a performance do protótipo gerado à luva já existente.

Em 2009 foi publicado o trabalho Jaijongrak, Chantasuban e Thiemjarus (2009) sobre redes de sensores para o corpo (*Body Sensor Network* ou *BSN*) como meio de interface gestual para comandar aplicações em casas. O sistema proposto consiste em um par luvas com um *flexsensor* posicionado sobre cada dedo ligados a um sistema de transmissão sem fio de dados (não especificado) para um computador. Também é apresentado o sistema que atua sobre o ambiente inteligente, que consiste na placa Lambda Nu SC-800XP (Um circuito com oito canais de acionamento por relé).

Em 2011 o trabalho Majeau et al. (2011) tratou de uma luva de dados sem rastreamento posicional com sensores baseados em fibra óptica plástica. O protótipo apresentado utilizava 13 sensores por perda de acoplamento óptico (onde o deslizamento da fibra em um tubo guia varia a distância de sua ponta ao emissor, implicando em menores tensões lidas na base do fotodiodo a medida que a ponta da fibra se distancia do LED emissor) e um sensor inercial resultando em um sistema capaz de monitorar 16 graus de liberdade (dois graus de flexoextenção para cada dedo, mais um de adução e abdução para o polegar, 2 graus de liberdade para o pulso e 3 graus indicando a orientação do sistema no espaço) com precisão de um grau a um preço de produção de 73,27 dólares canadenses.

Em 2012 Kumar, Rautaray e Agrawal (2012) apresentou uma proposta para mapeamento de uma luva de dados comercial (A *Vhand* 2.0 da empresa *DGTech*) para realizar comandos de *mouse* e desta forma facilitar a interação do usuário com aplicações de criação de conteúdo multimidiático. O produto comercial em questão apresenta uma *IMU*, cinco *flexsensors* e comunica-se com o computador via *Bluetooth*. Os pesquisadores utilizaram um algoritmo de classificação k-NN para reconhecer os gestos de apontar, clique com botão esquerdo do mouse, clique com o botão direito do mouse, arrastar e rotacionar.

Em 2014 Flores et al. (2014) apresentou uma proposta de uma luva de dados utilizando um microcontrolador *Arduino*, um sensor inercial e *flexsensor* manufaturados com fitas de alumínio, espuma condutora e fita isolante, reduzindo o custo de fabricação da luva em 90%. O grupo de pesquisa criou duas luvas, uma com *flexsensors* comerciais e outra com os manufaturados. Comparações entre ambas as luvas foram feitas em uma aplicação utilizando a API de *Phython* no software *Blender*. A luva contendo os sensores comerciais apresenta um sinal significativamente menos ruidoso e pode ser usada para aplicações mais

precisas, enquanto que a outra luva, apesar de apresentar um sinal de qualidade inferior, foi capaz de manipular a aplicação, embora maior ênfase nos movimentos fosse necessária para que a aplicação reconhecesse os gestos. A redução no preço pode compensar a perda na qualidade do sinal fornecido pelos *flexsensors*.

Em 2015 foi publicado o trabalho Pititeeraphab (2015). Este foi um trabalho sobre teleoperação de braços robóticos utilizando um exoesqueleto de braços e um par luvas de dados. Cada luva utiliza 5 *flexsensors*, um de 2,2 polegadas para o polegar e 4 de 4,5 polegadas para os demais dedos. O exoesqueleto possui 12 potenciômetros, detectando 6 graus de liberdade em cada braço. O sistema todo é controlado por um microcontrolador *Arduino* e possui interface com os braços robóticos via protocolo de comunicação sem fio *Zigbee*. O sistema robótico a ser controlado são os braços do robô *InMoov* (*InMoov*, França), constituído de 20 servo motores, pesando aproximadamente 30 kg e controlado por um microcontrolador *Arduino*.

Em 2016 alguns trabalhos sobre luvas de dados foram publicados em conferências. Houveram trabalhos sobre a novas tecnologias de sensoriamento como o uso de *strech sensors* como sensores para flexão das falanges (SBERNINI; PALLOTTI; SAGGIO, 2016). Outro novo sensor proposto foi um sensor de flexão maleável (SHEN et al., 2016), cujas características elétricas são similares ao *flexsensor*, entretanto este sensor tem a propriedade de ser mais maleável podendo ser posicionado diretamente sobre uma articulação e ser esticado e contraído acompanhando a dinâmica da pele nas proximidades da articulação. Outros trabalhos que foram apresentados neste mesmo ano se referiam a tecnologias de teleoperação de aparelhos robóticos por meio de luvas de dados (ARIYANTO et al., 2017; WANG; ZHANG, 2016; FANG et al., 2016), cada um utilizando sensores diferentes para operar um braço robótico, sendo que apenas o trabalho Ariyanto et al. (2017) apresenta um sistema de luva de dados bidirecional (implementando um *feedback* háptico por meio de sensores).

Uma das conclusões que se pode inferir sobre as análises dos diversos aparelhos apresentados e trabalhos publicados é que dos sensores apresentados, o que é mais utilizado e há mais tempo é o *flexsensor*. Seu uso comum não é evidência suficiente para atestar sua eficácia, mas aparenta implicar que sua implementação é a mais simples. Novos sensores que já estão sendo utilizados são *IMU's* e observa-se que há dois casos pontuais na literatura em que a fibra óptica é usada com resultados interessantes. Sensores que ainda estão sendo pesquisados são os elastômeros, *Stretch Sensors* e sensores de flexão maleáveis. Acrescido ao surgimento de estudos sobre novos sensores, a capacidade de processamento embarcado aumentou consideravelmente desde o início das pesquisas sobre luvas de dados e o surgimento de formas de comunicação sem fio só incrementou a atratividade destes sistemas de interação, entretanto o estado atual das pesquisas ainda é muito incipiente, e não foram encontrados trabalhos cujo foco seja uma luva de dados como um fim e não

como um meio. Este é um campo de pesquisa vasto e com grande potencial.

3 Materiais e Métodos

Neste capítulo será explanado o *Hardware* implementado no sistema, o *firmware* que implementa as funcionalidades do protótipo, o protocolo de comandos utilizado para comunicação entre o microcontrolador e a luva (o protocolo *DRES*), as implementações de software, que incluem a a biblioteca que implementa o protocolo de comunicações, o filtro utilizado para reduzir o ruído dos sinais amostrados a partir dos sensores de flexão e a aplicação. O último tópico deste capítulo se refere à implementação física do protótipo e aos testes realizados.

3.1 *Hardware*

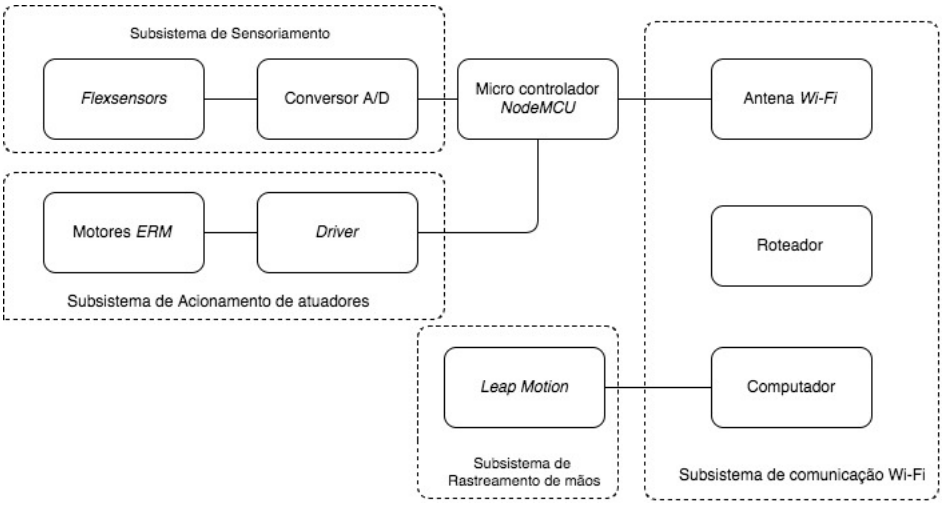
O protótipo foi desenvolvido com o intuito de suportar quatro subsistemas: (I) Um sistema de acionamento de motores de massa de rotação excêntrica (*ERM motors*) para atuar como mecanismo de *feedback* háptico. (II) Um sistema de sensoriamento de flexão das falanges, que realiza a aquisição de dados de flexão dos dedos. (III) Um sistema de Comunicação Wi-Fi para servir como interface de comunicação entre o protótipo e a aplicação. (IV) Um sistema de rastreamento de mãos para realizar a aquisição de dados da posição e orientação da mão e do polegar no espaço. O diagrama da Figura 7 mostra o protótipo e a segmentação dos subsistemas. Para integrar os três primeiros subsistemas foi utilizado o kit de desenvolvimento de *firmware NodeMCU* (*NodeMCU*, China)(Figura 8), que possui em seu centro um microcontrolador *ESP8266 12E* (*Espressif*, China)[1], que por sua vez é operado por um microprocessador de 32bits *Xtensa L106* (*Tensilica*,EUA), com *clock* de $80MHz$, com comandos da pilha *TCP* já integrada e uma memória de até 4Mb. A programação desta plataforma foi realizada utilizando o ambiente de desenvolvimento próprio do *NodeMCU*, o *ESPlorer*, cuja linguagem de programação suportada é *Lua* [2].

3.1.1 Subsistema de Sensoriamento

O propósito deste subsistema é a aquisição dos dados de flexão de todos os dedos, exceto o polegar. O subsistema de sensoriamento consiste em quatro *Flex Sensors* (*Spectra Symbol*, EUA)[3] (Figura 9), cuja resistência elétrica varia de acordo com sua flexão. Cada um destes sensores foi ligado em série com um resistor de 10kΩ, formando um divisor de tensão. Os valores analógicos da saída do divisor de tensão são traduzidos por um

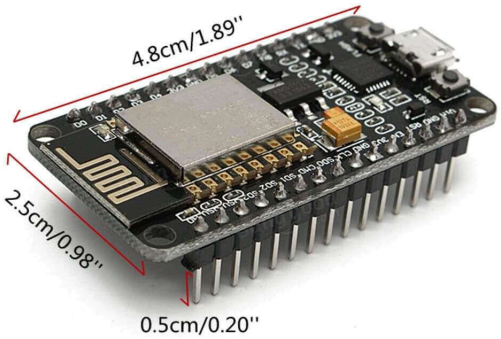
[1] Datasheet: http://espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf
[2] https://www.lua.org
[3] Datasheet: https://cdn.sparkfun.com/datasheets/Sensors/ForceFlex/FLEXSENSORREVA1.pdf

Figura 7 – Diagrama do sistema desenvolvido.



fonte: O autor

Figura 8 – plataforma de desenvolvimento de aplicações, *NodeMCU*, contendo ESP8266 e suas dimensões



fonte: http://irving.com.br/esp8266/nodemcu-esp8266-o-modulo-que-desbanca-o-arduino-e-facilitara-a-internet-das-coisas/

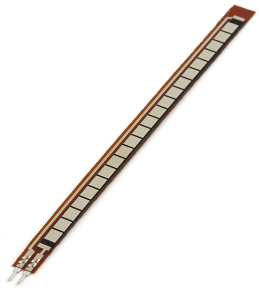
conversor analógico-digital. A tensão de saída é inversamente proporcional ao valor da resistência do sensor.

Considerando que o *NodeMCU* possui somente um canal de conversão A/D, optou-se pela utilização do conversor *ADS7828* (*Texas Instruments*, EUA)[4], que possui oito canais para conversão analógico-digital, com uma resolução de 12bits, taxa de amostragem de $100kHz$ em modo de operação normal e utiliza o protocolo de comunicação I^2C (*inter-integrated circuit bus*) para transmitir dados digitalizados ao microcontrolador.

O *Datasheet* do *ADS7828* especifica que sua tensão de referência interna é $2,5V$. Isto implica que sua resolução teórica é de aproximadamente $0,61mV$. Com essa informação

[4] Datasheet: http://www.ti.com/lit/ds/symlink/ads7828.pdf

Figura 9 – *Flex Sensor*



fonte: https://www.sparkfun.com/products/8606

é possível inferir a resolução do sensor, entretanto é importante levar em consideração que a correlação entre a tensão lida pelo conversor e o valor da resistência não é linear (apesar de ser próxima do linear). Uma análise da função do divisor de tensão revela que a pior resolução está atrelada aos maiores valores de resistência do sensor (onde, para uma mesma variação na tensão de saída do divisor, a variação de resistência é maior). De acordo com o *Datasheet* dos sensores: sua resistência possui valores na faixa de $10k\Omega \pm 30\%$ quando não flexionado e o dobro desta resistência quando totalmente flexionado. Assumindo que foi escolhido um resistor de $10k\Omega$ para a ligação em série, que o sensor em questão tem o maior valor de repouso possível: $13k\Omega$ e, consequentemente, $26k\Omega$ quando totalmente flexionado e que valor de tensão que alimenta o circuito apresenta valor de $3,6V$, obtém-se que o valor de saída do circuito, quando o sensor está totalmente flexionado equivale á $1V$. Entretanto, o valor lido pelo conversor seria de $1,0004V$, uma vez que este valor deve ser um múltiplo da resolução do CI por um número inteiro. O valor lido pelo conversor é o primeiro valor disponível acima do valor real, consequentemente o valor da resistência do sensor lido pelo conversor seria de: $25,9856k\Omega$, apresentando um erro de discretização de $0,0554\%$. O próximo valor de tensão que o conversor conseguiria interpretar seria de: $1,0010V$, significando uma resistência de: $25,9637k\Omega$. A diferença entre as resistências interpretadas pelo conversor representam a maior zona de insensibilidade do conversor e é uma faixa de $21,9233\Omega$ (aplicando o mesmo processo em torno do valor de repouso de um sensor padrão de $10k\Omega$, a variação obtida seria de $6,7\Omega$ para a faixa inferior de resistência e $6,8\Omega$ para a faixa superior).

Assumindo que a medida de interesse seja a angulação da ponta do sensor e que essa angulação varie de 0^o a 180^o e que essa variação seja linearmente correlata com a variação da resistência do sensor temos a equação 3.2:

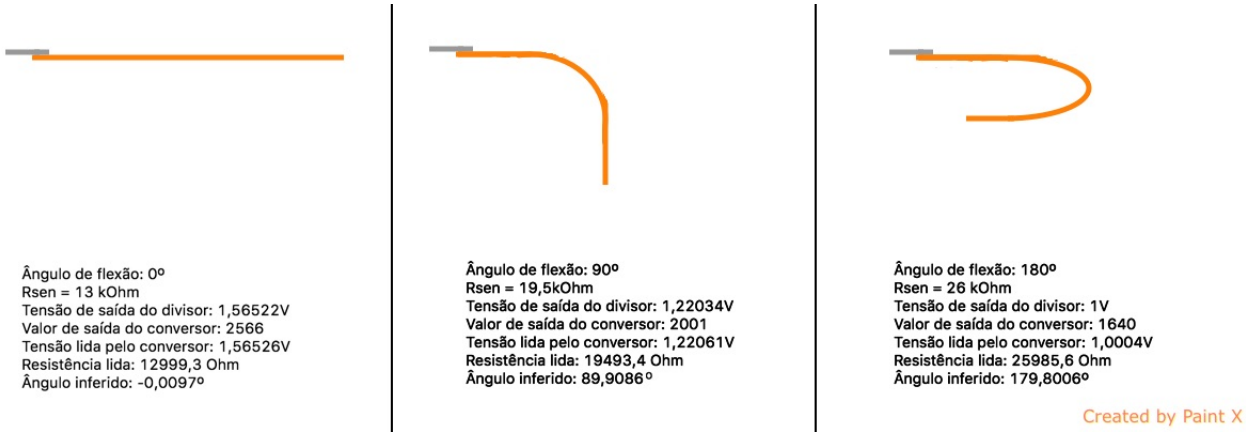
$$\frac{\Theta - 0}{180 - 0} = \frac{R - 13}{26 - 13} \tag{3.1}$$

A equação 3.1 resulta em:

$$\Theta = 180\frac{R - 13}{13} \tag{3.2}$$

Onde Θ é o valor do ângulo em graus para cada valor válido de R, que varia entre 13 e 26, em $k\Omega$. Para que Θ tenha um incremento em um grau, R precisa ser incrementado em $72,22\Omega$. Uma variação em R de 21.92Ω implica em uma variação angular de $0,3035^o$. A Figura 10 ilustra três configurações para o *Flexsensor*, apresentando valores teóricos de flexão e resistência e demonstrando os valores de que seriam inferidos pelo sistema devido aos erros de discretização.

Figura 10 – diferentes configurações teóricas de flexão para o *flexsensor* com maior resistência teórica. À esquerda uma representação do sensor não flexionado. Ao centro está a flexão do sensor a 90 graus. À direita, flexão do sensor a 180 graus



fonte: O autor

O conversor é capaz de transmitir os dados coletados ao microcontrolador através do barramento de comunicação entre circuitos integrados I^2C, um canal por vez, conforme especificado no *datasheet* do conversor. O dado que é enviado é um número, que uma vez multiplicado por $0,61mV$ resultará no valor de tensão medido em V_o.

3.1.2 Subsistema de acionamento

O subsistema de acionamento é constituído por cinco motores DC com massa de rotação excêntrica (*ERM*) (Figura 11), um meio de acionamento e comandos de acionamento. Ele é a implementação física do meio pelo qual *feedback* háptico pode ser fornecido ao usuário. A princípio foi considerado que o meio de acionamento seria através um transistor NPN e um diodo para descarga de corrente indutiva para cada motor, entretanto esse sistema teria uma dimensão indesejável para o propósito de acoplamento a

uma luva, portanto foi substituído por um circuito integrado de acionamento de cargas, mais especificamente o CI *ULN2003* (*Diodes Inc.*, EUA) [5], que possui sete canais de acionamento e que opera pelos mesmos princípios básicos da primeira forma de acionamento considerada.

Figura 11 – *Motores ERM*

fonte: http://produto.mercadolivre.com.br/MLB-695329123-mini-motor-lg-vibra-vibracall-arduino-robotica-original-_JM

Os comandos de acionamento são realizados através de *Pulse Width Modulation* (*PWM*), já que esta forma de acionamento permite não somente acionar e desligar os atuadores com um único comando de alteração do *Duty-Cicle*, mas também permite regular a intensidade de vibração dos motores com o mesmo comando.

3.1.3 Subsistema de Comunicação Wi-Fi

Este é o subsistema de interface de comunicação entre o protótipo e aplicações. O microcontrolador *ESP8266* implementa as especificações definidas pelo protocolo *IEEE 802.11*, mais especificamente os protocolos *IEEE 802.11* b/g/n/e/i e a plataforma *NodeMCU* faz uso apenas dos protocolos b/g/n, que são o conjunto de implementações físicas e especificações para controle de acesso que permitem comunicação sem fio *wi-fi* em redes locais (*WLAN*s). Adicionalmente as bibliotecas criadas pela equipe de desenvolvimento do *NodeMCU* permitem a implementação rápida de comunicação nos protocolos *TCP/IP* e *UDP*. Uma vez que estes sistemas estavam disponíveis a questão se resume a escolher um protocolo de comunicação e desenvolver um conjunto de comandos que pudessem ser trocados utilizando o protocolo escolhido. Para esse projeto o protocolo de comunicação escolhido foi o *UDP*, por ter um *header* menor, ocupando uma banda menor do canal

[5] Datasheet: https://www.diodes.com/assets/Datasheets/ULN200xA.pdf

Figura 12 – *Leap Motion*

fonte: https://apps.leapmotion.com

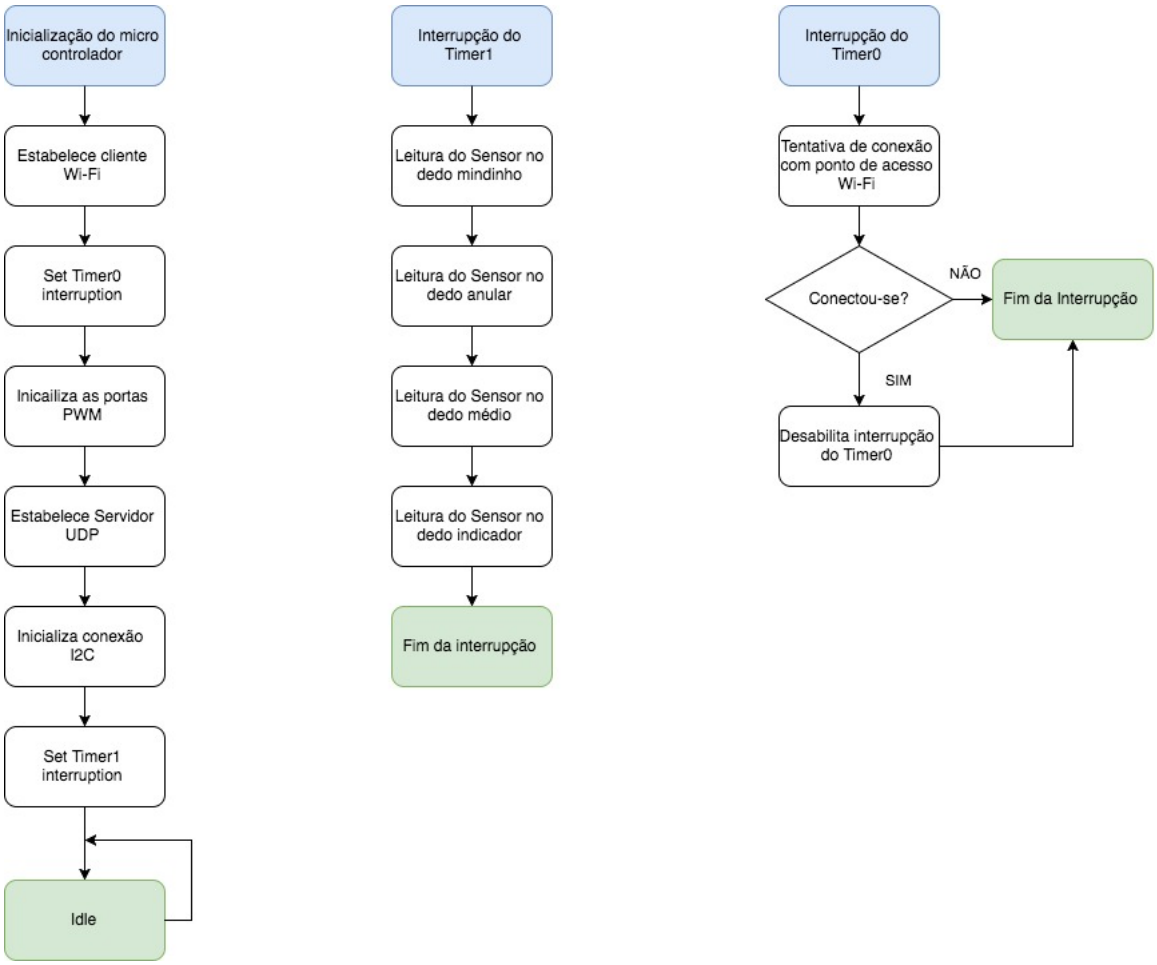
de comunicações, além de não implementar maneiras de garantir chegada de pacotes, o que poderia ser considerado uma desvantagem, se não fosse levado em consideração que o sistema de garantias de transmissão de dados do protocolo *TCP/IP* implicasse em um aumento da latência da comunicação, já que há atrasos inerentes à espera de confirmação de envio de pacotes e reenvio de pacotes que não receberam confirmação. Como a perda de um eventual pacote de informações não inviabiliza o sistema, tendo em vista que o protocolo de controle utilizado mantém toda a informação de comando e resposta contida em pacotes unitários e que os comandos serão constantemente requisitado em intervalos de tempo curtos (significando que um pacote perdido será reposto, sendo o equivalente a um reenvio de pedido por perda de confirmação), pode se inferir que o protocolo *UDP* é adequado ao sistema proposto.

O protocolo de comandos unitários desenvolvido (de nome provisório: *DRES*) será explanado mais adiante.

3.1.4 Subsistema de rastreamento de mãos

Para realizar o rastreamento de posição e orientação do protótipo optou-se por uma solução comercial: o *Leap Motion* (*LEAP MOTION inc.*, EUA) (Figura 12). Essa solução consiste em um par de câmeras infravermelhas e três *LED's* Infravermelhos operando em uma taxa de aquisição de até 115 imagens por segundo, construída especificamente para o rastreamento de mãos. Como se trata de uma solução comercial, os detalhes específicos do seu funcionamento não são fornecidos, mas sabe-se que o produto utiliza um algoritmo de visão computacional associado a um modelo gráfico para realizar sua função. O sistema possui uma documentação concisa e permite sua agregação a sistemas maiores. Mais detalhes sobre a forma como esse subsistema foi integrado na sessão de Aplicação.

Figura 13 – Fluxogramas do funcionamento do *firmware*. A esquerda está a *thread* principal do programa. O fluxograma do centro descreve a interrupção por *timer* para a aquisição de dados de flexão dos dedos. A direita está a interrupção por *Timer* para tentativa de conexão com o ponto de acesso *wi-fi*

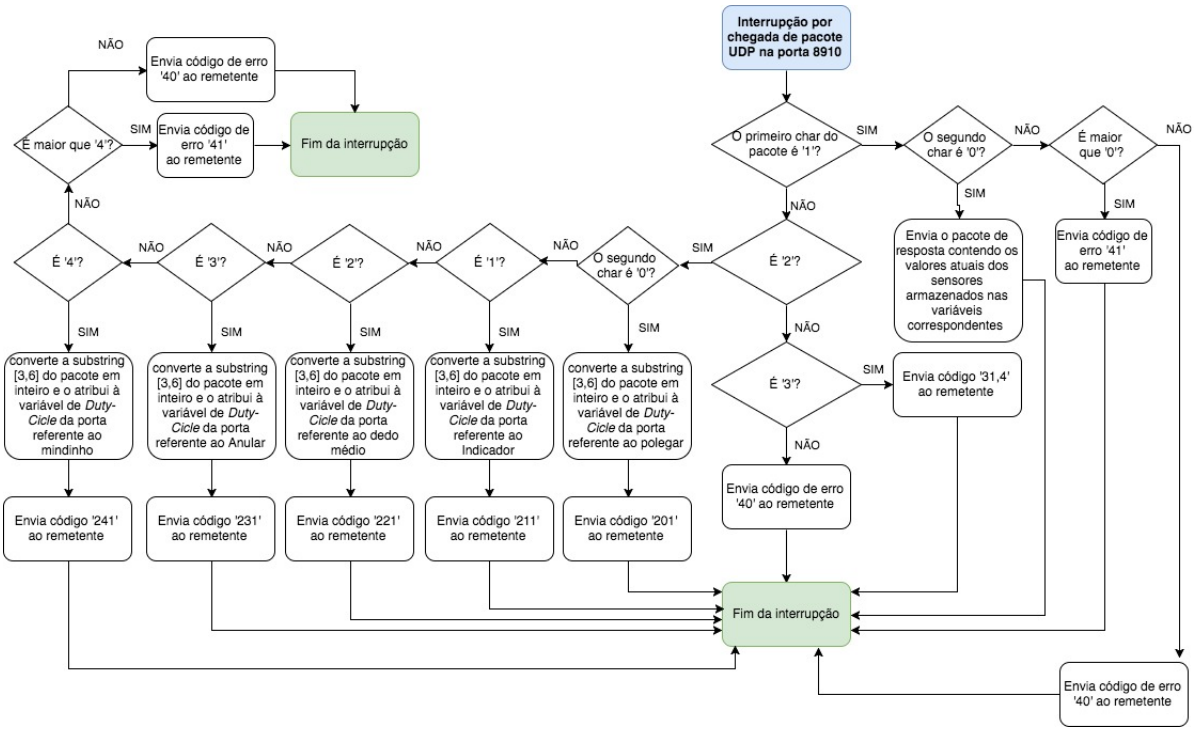


fonte: O autor

3.2 Firmware

O código do *firmware* desenvolvido para que o microcontrolador desempenhe suas funções está disponível no apêndice C e opera segundo as Figuras 13 e 14. Foram implementadas duas interrupções por *Timer* (Figura 13) e uma interrupção por chegada de pacotes *UDP* (Figura 14). Uma das interrupções por *timer* irá realizar uma tentativa de acesso à rede *wi-fi* especificada no código do *firmware* a cada segundo. Uma vez que o sistema se conecta a rede, a interrupção é desativada. A segunda interrupção por *timer* é responsável pela amostragem dos valores digitalizados de tensão relativa a cada sensor de flexão. A interrupção por chegada de pacote *UDP* constitui o servidor *UDP* que escuta a porta (*socket*) 8910 e seu propósito é responder aos comandos contidos nos pacotes que causam a interrupção.

Figura 14 – Fluxograma da interrupção por recepção de pacote UDP



fonte: O autor

Para o escopo deste projeto, a conexão do microcontrolador a uma rede é realizada via código (significando que para conectá-lo a uma rede diferente é necessário alterar variáveis específicas do código fonte, mais especificamente no arquivo ***init.lua***).

No arquivo ***main.lua***, são definidas variáveis para armazenar o valor de *duty cicle* do *PWM* de cada um dos cinco atuadores e também há variáveis que armazenam o valor lido de cada um dos sensores. Logo em seguida é declarado o servidor *UDP* com todas as respostas possíveis a comandos válidos do protocolo *DRES*. Por fim é implementada uma amostragem dos 5 sensores através de comandos no protocolo I^2C utilizando uma interrupção por timer, que é acionada a cada $10ms$ (culminando em um taxa de amostragem de pouco menos que 100 amostras por segundo) onde a cada interrupção é realizado sequencialmente um pedido do valor de cada sensor de flexão ao conversor *ADS7828* e cada valor retornado é armazenado nas variáveis apropriadas.

3.3 Protocolo DRES

O protocolo DRES pode ser considerado um metaprotocolo, no sentido de que ele define funções, mas não especifica quais dados ou qual o formato dos dados a serem enviados nos pacotes. Este protocolo consiste em um conjunto simples de quatro comandos criados para requerer informações da unidade escrava ou alterar uma variável interna da

Tabela 1 – Protocolo DRES (Em *ASCII*)

Comando	Código	Resposta	Descrição
Leia (Read)	1X	1XYYYY, ZZZZ,AAAA...	Comando de requisição de dados à unidade escrava. Ele consiste do caractere '1' e de um identificador para um conjunto de dados de interesse. A resposta a essa requisição é um eco do comando seguido por um conjunto de caracteres numéricos separados por vírgula que represente o conjunto de dados indexados.
Faça (do)	2XYYYY, ZZZZ,AAAAA...	2X	Comando para modificação de um conjunto de variáveis internas do escravo. Ele consiste no caractere numérico '2' seguido pelo identificador do conjunto de variáveis e por fim por um conjunto de caracteres numéricos separados por vírgulas significando um grupo ordenado de valores que irão ser atribuídos a variáveis.
Status	30	3X,Y	Comando para a aquisição de índices de grupos de leitura e variáveis internas. Ele consiste do caractere numérico '3' seguido do caractere numérico '0' e sua resposta é o caractere numérico 3, seguido pelo número de indices de leitura, uma vírgula e por fim o número de índices de conjunto de variáveis configuráveis.
Erro (Error)	-	40,41,42,....	Esse comando é reservado a respostas de erro. A resposta 40 significa datagrama inválido. A resposta 41 significa que o índice do grupo de sensores/variáveis está acima do conjunto implementado no escravo. As demais respostas podem ser configuradas pelo implementador para se adaptarem a detalhes de sua implementação

unidade que esteja ligada a uma rede que suporte protocolo *UDP* ou *TCP/IP*. A sigla já define os quatro comandos: *Do* (Faça), *Read* (Leia), *Error* (Erro) e *Status* (Estado). O datagrama do protocolo não envolve endereçamento uma vez que esta questão já é resolvida pelos protocolos *UDP* e *TCP/IP* através de conexões orientadas a *sockets* e por resolução de *IP* respectivamente. Os detalhe do protocolo estão explicitados na tabela 1.

E mais especificamente para a luva, o protocolo é mapeado segundo a tabela 2.

3.4 Software

Para integração do protótipo com uma aplicação em ambiente de programação, foi desenvolvida uma classe em *C#* que gerencia o protocolo *DRES* implementado para a luva (que está disponível no apêndice D). Esta classe contem funções que montam os pacotes de envio, sendo do encargo do programador a montagem de um cliente *UDP* para realizar a comunicação com o protótipo e enviar pacotes. O código exemplo no apêndice E demonstra como implementar o protocolo em uma aplicação simples de troca de dados. Foi desenvolvido também um *script* em *C#* para a plataforma de desenvolvimento Unity permitindo a comunicação com o protótipo através do mesmo protocolo *DRES*.

Adicionalmente foi implementado um filtro passa-baixas para amenizar ruídos de alta frequências relativos aos artefatos de implementação física tais quais maus contatos e ruídos mecânicos presentes nos sinais proveniente dos sensores de flexão. Devido às limitações de espaço físico e principalmente à facilidade de ajuste, optou-se pela implementação do filtro em *software* em oposição a implementação do filtro em *Hardware*. O filtro utilizado apresenta as mesmas características de um filtro capacitivo RC e é regido pela equação 3.3.

$$V_o[n] = (1-k)V_i[n] + (k)V_o[n-1]) \tag{3.3}$$

Onde V_o são valores de saída no instante amostral n para valores de entrada $V_i[n]$ acrescido de uma parcela do valor anterior de saída $V_o[n-1]$. k é a constante de tempo cujos valores possíveis e viáveis estão contidos no intervalo real entre 0 e 1 e é regida pela equação 3.4.

$$k^N = e^{-1} \tag{3.4}$$

Esta equação apresenta uma forma de determinar o valor de k com base no valor de N (modificando a equação 3.4, resultando na equação 3.5). Para explicar o significado de N, deve se considerar a situação hipotética em que o filtro recebe como entrada um degrau unitário no instante $n = 0$, neste caso N é a quantidade de amostras a partir do ponto $n = 0$ necessárias para que V_o atinja o valor de $0,63$ $(1 - e^{-1})$, ou seja $V_o[N] = 0,63V_i[N]$.

$$k = e^{-\frac{1}{N}} \tag{3.5}$$

É necessário é determinar um valor apropriado de amostras para definir um valor para k. Se a quantidade de amostras for muito pequena, o filtro continuará sensível a

Tabela 2 – Mapeamento de comandos da luva para o protocolo *DRES*

Comando	**Descrição**	**Resposta**	**Descrição**
10	Pedido de leitura dos *flexsensors*	10WWWW,XXXX, YYYY,ZZZZ	Retorna, além do eco do comando, quatro valores separados por vírgulas. Cada valor é representado em quatro caracteres numéricos, representando o valor de tensão lida pelo *ADS7828* relativo respectivamente aos dedos indicador, médio anelar e mínimo.
20XXXX	Comando para modificar o *duty-cycle* do *PWM* do atuador no polegar. O valor enviado deve estar contido no intervalo [0000 1023]	20	Resposta de reconhecimento positivo e execução do comando
21XXXX	Comando para modificar o *duty-cycle* do *PWM* do atuador no indicador. O *Range* de valores é [0000 1023]	21	Resposta de reconhecimento positivo e execução do comando
22XXXX	Comando para modificar o *duty-cycle* do *PWM* do atuador no dedo médio. O valor enviado deve estar contido no intervalo [0000 1023]	22	Resposta de reconhecimento positivo e execução do comando
23XXXX	Comando para modificar o *duty-cycle* do *PWM* do atuador no dedo anelar. O valor enviado deve estar contido no intervalo [0000 1023]	23	Resposta de reconhecimento positivo e execução do comando
24XXXX	Comando para modificar o *duty-cycle* do *PWM* do atuador no dedo mínimo. O valor enviado deve estar contido no intervalo [0000 1023]	24	Resposta de reconhecimento positivo e execução do comando
30	Comando para requisição da quantidade de indices de leitura e variáveis configuráveis	31,5	Existem um índice de leitura e cinco índices de variáveis configuráveis
-	-	40	Erro: Comando inválido
-	-	41	Erro: O índice de leitura/escrita excede os índices do sistema

ruídos de alta frequência, se a quantidade de amostras for muito grande, o filtro fica insensível a qualquer variação no seu sinal de entrada, mantendo o sinal constante. Ambos os extremos do filtro podem ser exemplificados se o valor de k na equação for substituído por 0 e por 1. Para o caso em que $k = 0$, a saída dos sistema é igual à entrada; para o caso em que $k = 1$, a saída dos sistema independe da entrada. Efetivamente o valor de k é inversamente proporcional à frequência de corte deste filtro passa-baixas.

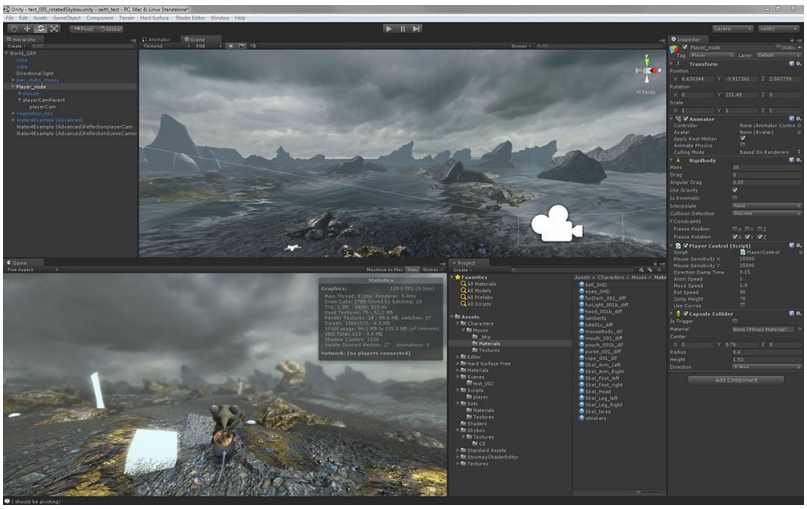
Para definir uma frequência de corte apropriada ao filtro seria necessário realizar uma análise das frequências do sinal de entrada, definir quais são as frequências do ruído do sinal, realizar a transformada z da equação do filtro e estabelecer a relação de k com a frequência de corte. Outra opção é o método empírico: a taxa de amostragem que deve ser levada em consideração é a da aplicação, uma vez que é menor que a taxa de amostragem do microcontrolador. A aplicação desenvolvida por meio da plataforma *Unity* (*Unity Technologies*,EUA) tem uma taxa de quadros pré-configurada para 30 quadros por segundo. Esta taxa é adequada para aplicações desenvolvidas na plataforma *Unity*, já evita a sobrecarga do processador, permite que a plataforma compute toda a parte de simulação física e processamento gráfico e não obstrui a percepção humana. Realizar uma amostragem de valores de flexão a uma taxa superior é desnecessário para os propósitos de uma aplicação de interação com ambientes virtuais. É aceito que o limiar de percepção de controle humano e atuação humana é de 300 ms (significando que um humano consegue controlar um sistema que apresente atrasos de até 300ms) (SHAN; LINGJIANG, 2009), isto implica que um valor aceitável para a constante de tempo de decaimento deva ser inferior a este limite. A taxa de amostragem estabelecida é de 30 amostras por segundo (10 vezes maior que o limite proposto por Shan e Lingjiang (2009)), portanto o filtro deve atingir o valor de entrada em até 10 amostras. O ponto de partida escolhido foi $N = 7$. Aplicando este valor à equação 3.5, temos que k vale $0,8668779$, que é um ponto de partida apropriado para começar a realizar testes com o filtro. Caso ainda exista ruído de frequências mais elevadas, deve se elevar o valor de N, elevando assim o valor de k e reduzindo o valor da frequência de corte, caso o sistema se mostre pouco sensível ou apresente atrasos na resposta, é necessário reduzir o valor de N para elevar o valor da frequência de corte.

3.5 Aplicação

Por fim foi criada uma aplicação na plataforma de desenvolvimento Unity (Figura 15) que permitisse testar o protótipo. A Figura 16 mostra a tela da aplicação, onde o usuário pode manusear um cursor tridimensional. Os componentes gráficos da aplicação foram modelados em Blender (*Blender Foundation*) (17).

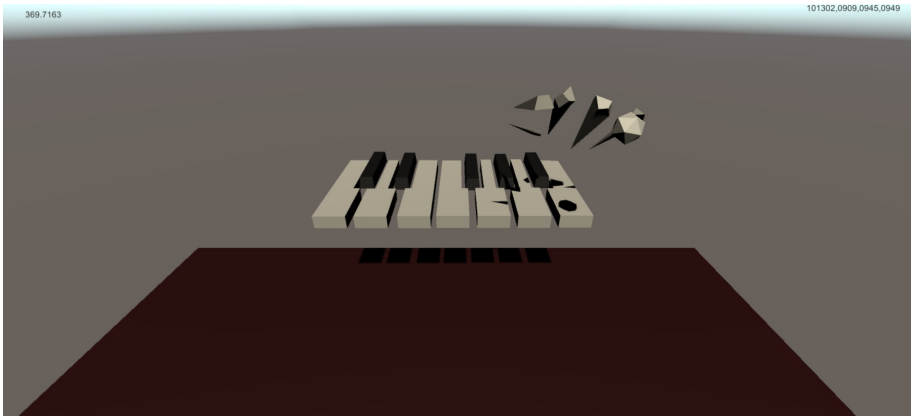
Neste ponto é válido se estender um pouco mais a respeito dos modelos gráficos que

Figura 15 – Ambiente de desenvolvimento lúdico Unity 5



Fonte: http://www.wraithkal.com/unity-5-multiplayer/

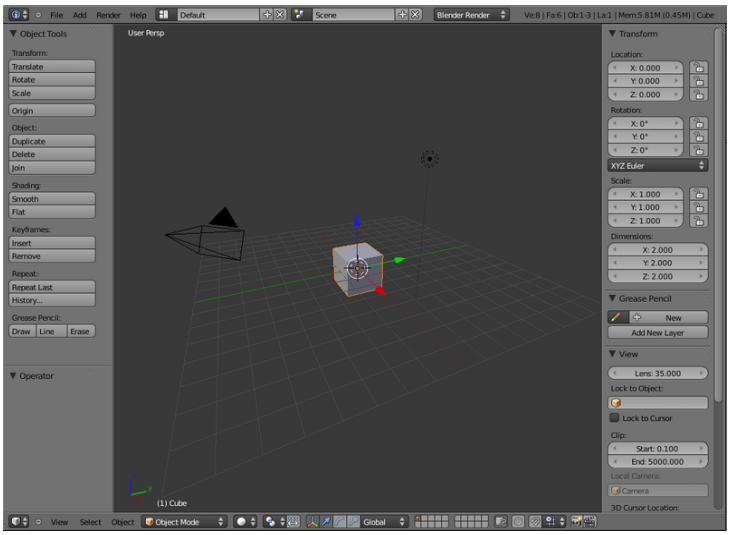
Figura 16 – tela da aplicação exemplo



Fonte: O autor

o usuário utilizará como meio para interagir com o ambiente virtual: seu *avatar*. Note-se, porém, que o termo *avatar* está sendo utilizado como o componente que liga o usuário ao ambiente virtual gráfico. Desta forma uma analogia válida é que o elemento gráfico do cursor do *mouse* na interface gráfica de sistemas operacionais seria o *avatar* do usuário dentro do ambiente de interface gráfica de um sistema operacional. O ponto de partida da concepção deste *avatar*, a principio, parece ser a representação fidedigna da interface mecânica (a luva) dentro do ambiente virtual e, claro, esta concepção está correta para uma aplicação de prova de conceito, já que bastaria deformar a representação virtual da mesma forma que a interface mecânica para demonstrar a funcionalidade da interface mecânica. Entretanto, tal conceito deve ser abandonado quando se considera a criação de

Figura 17 – Plataforma de modelagem 3D Blender



Fonte: https://en.wikibooks.org/wiki/Blender_3D:_Noob_to_Pro/Printable_Version

uma interface para uma aplicação com um propósito específico, neste caso a modelagem da ferramenta deve tomar uma forma que desempenhe um papel melhor que a representação realista do modelo, tal qual o cursor é uma representação do *mouse*.

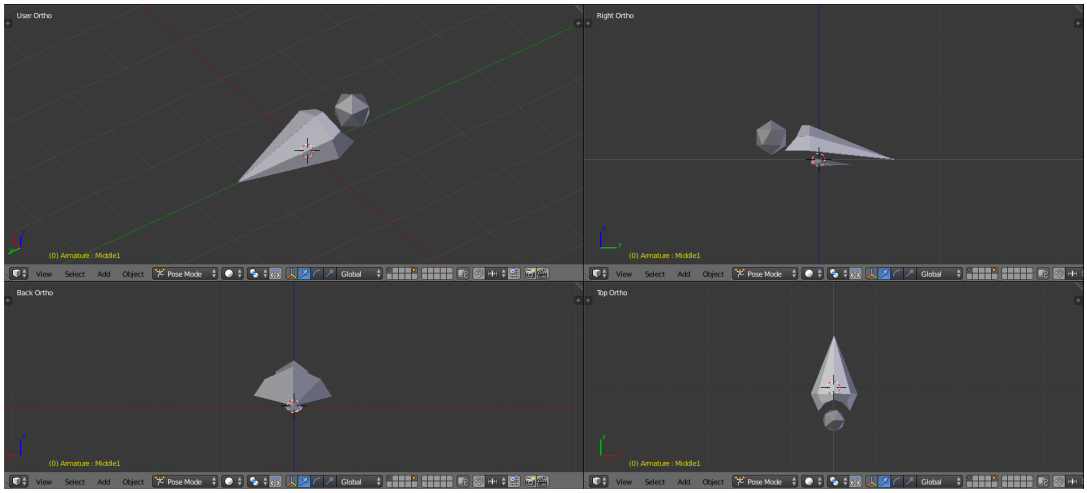
De acordo com o manual do software *Blender*[6], a ferramenta trata seus diferentes tipos de dados armazenando-os em *Data Blocks* (Blocos de dados), que funcionam com a dupla função de conter um conjunto de dados e estarem relacionados a outros *Data Blocks*, existindo exatamente 33 tipos diferentes de *Data blocks*, cada um contendo uma estrutura de dados específica para representar elementos do ambiente. Estes blocos de dados são usados em conjunto para representar objetos, efeitos visuais, câmeras, iluminação, audio e outros aspectos de imagens e modelos gerados digitalmente dentro do ambiente da plataforma. Por exemplo um modelo tridimensional de uma esfera inerte dentro do programa é representada por um *Data Type* do tipo *Object* que contem atrelado a ele numa relação pai-filho um *Data type* do tipo *Mesh*, que pode estar implícita ou explicitamente atrelado a um *Data Type* do tipo *Material*.

Para o *avatar* do usuário dentro da aplicação exemplo foi criado um objeto do tipo *Object* contendo um *Mesh* (que corresponde a malha de vértices, arestas, faces e normais que definem a forma de um objeto gráfico no espaço virtual) do cursor conforme a Figura 18. Este *Mesh* foi modelado de forma a permitir uma segunda forma mostrada na Figura 19 cujo objetivo é mimetizar uma mão. O objeto foi atrelado a um *Data Block* do tipo *Armature*, contendo elementos chamados de *bones*(ossos) necessários para animação (Figura 20). Finalmente, foram criadas quatro animações: uma do cursor virando uma

[6] https://docs.blender.org/manual/en/dev/data_system/data_blocks.html

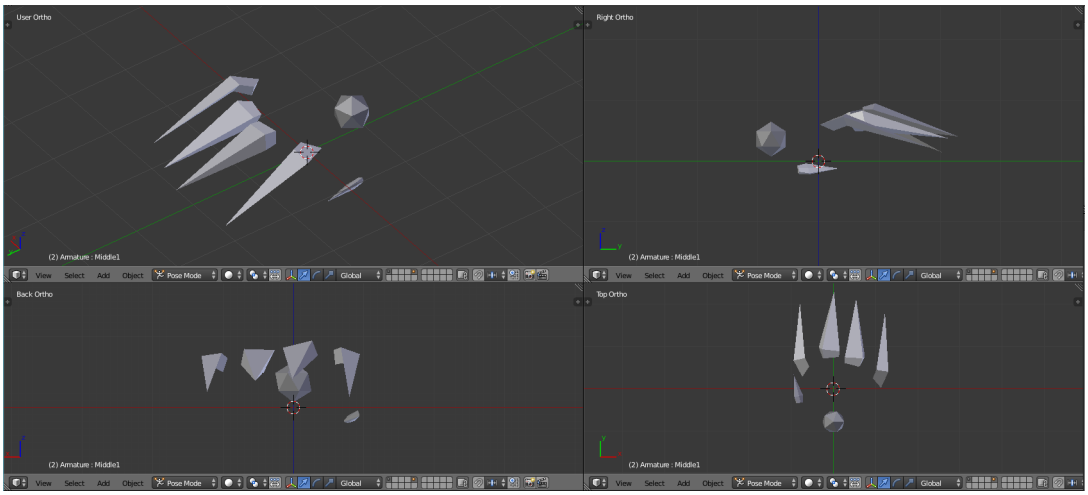
mão[7], outra que faz exatamente o contrário[8], e duas animações de dez frames com cada forma estática.

Figura 18 – Cursor 3D no ambiente Blender



Fonte: o Autor

Figura 19 – Segunda forma Cursor 3D no ambiente Blender



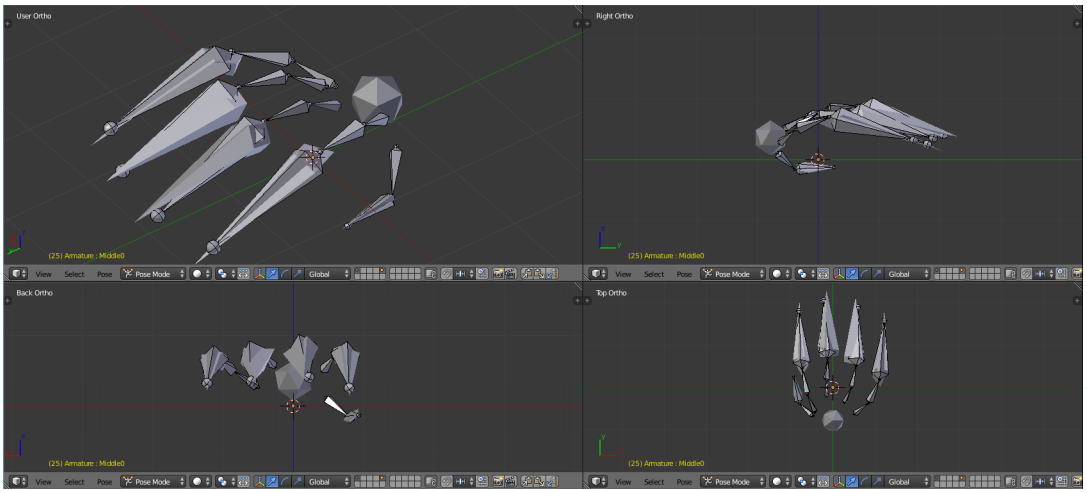
Fonte: o Autor

O contexto concebido para a aplicação seria um espaço com teclas de piano que geram sons ao receberem uma colisão e retornam a sensação de colisão por meio dos motores *ERM*.

Os dados dos *flexsensors* deveriam ser utilizados para complementar os dados colhidos pelo *Leap Motion*, entretanto por motivos de simplicidade o sensor *Leap Motion* e

[7] video da animação: https://www.youtube.com/watch?v=aNmlbPOALUc
[8] video da animação: https://www.youtube.com/watch?v=QLrxOWfBTBY

Figura 20 – Armadura do Cursor 3D no ambiente Blender



Fonte: o Autor

os sensores de flexão realizam tarefas diferentes: O primeiro é responsável pelo rastreamento de posição e orientação da mão, bem como do polegar, enquanto que os últimos mensuram a flexão das demais falanges.

A forma escolhida para converter o valor de tensão lido pelo conversor analógico-digital em ângulos de flexão envolve um fator importante: o modelo de mão. Por simplicidade o modelo escolhido para representar as falanges do dedo indicador ao dedo mínimo é o de articulações no formato de dobradiça, significando que, para efeitos de modelo, os únicos valores de interesse seriam os ângulos de flexo-extensão das falanges. No entanto, cada sensor de flexão é responsável por inferir a rotação das três articulações do dedo, então para realizar uma conversão funcional, simplificou-se o modelo por conta das limitações do sistema. A implementação mais simples foi a de que todas as juntas do dedo flexionam-se com a mesma angulação de tal forma que o ângulo da falange distal coincida com o ângulo de orientação da ponta do sensor de flexão no plano sargital. Essa conversão pode ser expressa pela equação paramétrica 3.6 derivadas do teorema de Tales.

$$\frac{\Theta - \beta}{\alpha - \beta} = \frac{V - MIN}{MAX - MIN} \tag{3.6}$$

Que pode ser reescrita como:

$$\Theta = \frac{(\alpha - \beta)(V - MIN)}{MAX - MIN} - \beta \tag{3.7}$$

Estas equações apresentam quatro constantes: MAX é o valor numérico máximo medido pelo sensor, MIN é o valor mínimo do sensor, α é o ângulo de articulação correspondente a MAX, β é o ângulo de articulação correspondente ao valor de MIN.

As equações possuem uma variável: V que corresponde ao valor medido pelo sensor em um dado instante; e resultam em Θ, que corresponde ao ângulo de uma articulação em um dado instante.

Aplicando-se a equação 3.7 a cada *bone* da armadura (*Armature*) do modelo da figura 20 (exceto os do polegar) e ajustando-se os valores das constantes, é possível contornar a variação de valores entre sensores, permitindo assim uma tradução dos valores medidos pelos sensores para valores de flexão da articulação.

O sensor *Leap Motion* foi utilizado para complementar as demais funções que não puderam ser implementadas de outra forma com os recursos já empregados no desenvolvimento do protótipo, tais quais o posicionamento da mão e do polegar no espaço.

Em termos de interface da aplicação com o protótipo, foi estabelecido a principio, que a cada ciclo do programa seria enviado um pacote *Read* do protocolo *DRES* para leitura dos quatro sensores montados no protótipo. Para o acionamento e desligamento dos sensores foi considerado, também em primeira instância, o envio de um pacote *Do* a cada colisão detectada para ativação dos atuadores, configurando o atuador apropriado com um *Duty-cicle* de 50%, depois de 100 milissegundos, um segundo pacote *Do* é enviado para desativar o atuador (configurando o *Duty-cile* para 0%). Uma vez que a aplicação receba os dados puros de flexão dos dedos, eles são então filtrados e então utilizados para determinar o valor dos ângulos de flexão.

Para o cursor, os ângulos de flexão são utilizados para reconhecer o gesto que inicia a animação de transição para o formato de mão, este gesto consiste na flexão de todos os sensores e só pode ser repetido após a aplicação detectar que os sensores voltaram ao estado de estendido. Já no formato de mão, os valores de flexão de cada dedo são aplicados aos ossos da armadura do modelo fazendo com que cada dedo assuma o valor de flexão desejado e também são utilizados para reconhecer o gesto para retornar ao formato de cursor.

3.6 Protótipo e testes

A integração de todos os subsistemas juntamente com a parte mecânica teve três iterações, onde o ponto principal foi o posicionamento adequado tanto dos sensores quanto dos atuadores.

A primeira iteração foi feita fixando a ponta do sensor de flexão no dorso da luva com fita adesiva, mais especificamente sobre a área correspondente ao centro da falange distal e suportes guia foram fixados nos dedos da luva para permitir um deslizamento ao longo do dorso do dedo por parte do sensor para acompanhar movimentos de flexão e extensão sem tensionar o sensor. Os atuadores foram fixados sob o centro das falanges

digitais com cola plástica.

Na segunda iteração o sensor teve sua base (ponto de contato do sensor com os fios) fixada na luva um pouco antes da junta metacarpofalangeana através de fios de tecido costurados na luva. Guias plásticas para o sensores foram costuradas no dorso dos dedos das luvas para permitir que os sensores ainda se mantivessem sobre a superfície da luva e pudessem deslizar para acompanhar o movimento do dedo e se adequar à deformação da pele. Suportes plásticos foram desenvolvidos e impressos em impressora 3D para prover maior suporte aos atuadores. Estes suportes plásticos foram costurados sob o centro da falange distal e os atuadores foram fixados sobre os suportes com cola quente.

A última iteração envolveu substituir a costura dos sensores sobre a luva pela fixação dos sensores em um suporte plástico impresso em impressora para impedir que o sensor flexione em sua base, evitando a separação entre os contatos metálicos e o pigmento condutivo, reduzindo assim a possibilidade de maus contatos ocorrerem, ao mesmo tempo em que a peça mantém os sensores fixos pela base.

Uma vez que a integração do sistema foi feita de forma a que o sistema se torne funcional seriam necessários realizar testes para sua validação. O objetivo deste trabalho foi construir um protótipo de uma luva de dados rastreada dentro de um espaço limitado com possibilidade de *feedback* háptico que pudesse interagir com uma aplicação exemplo, portanto os testes projetados foram direcionados de tal forma que o protótipo pudesse atender ao seu objetivo. Suas especificações quantitativas, apesar de importantes, foram deixadas em aberto devido à incerteza quanto à qualidade do produto resultante. O imperativo era que o protótipo funcionasse, suas especificações sendo consequência de seu funcionamento.

Para realizar os testes de subsistemas de fato foi primeiro necessário fazer com que todos os subsistemas funcionassem e interagissem com a aplicação. Então ajustes foram feitos, os valores de taxa de amostragem da aplicação e a constante k do filtro digital foram alterados, o sistema foi ajustado aos valores dos sensores e problemas mecânicos referentes ao mau contato de sensores e rompimento dos fios dos atuadores foram tratados para poder possibilitar testes em condições em que o protótipo fosse usável.

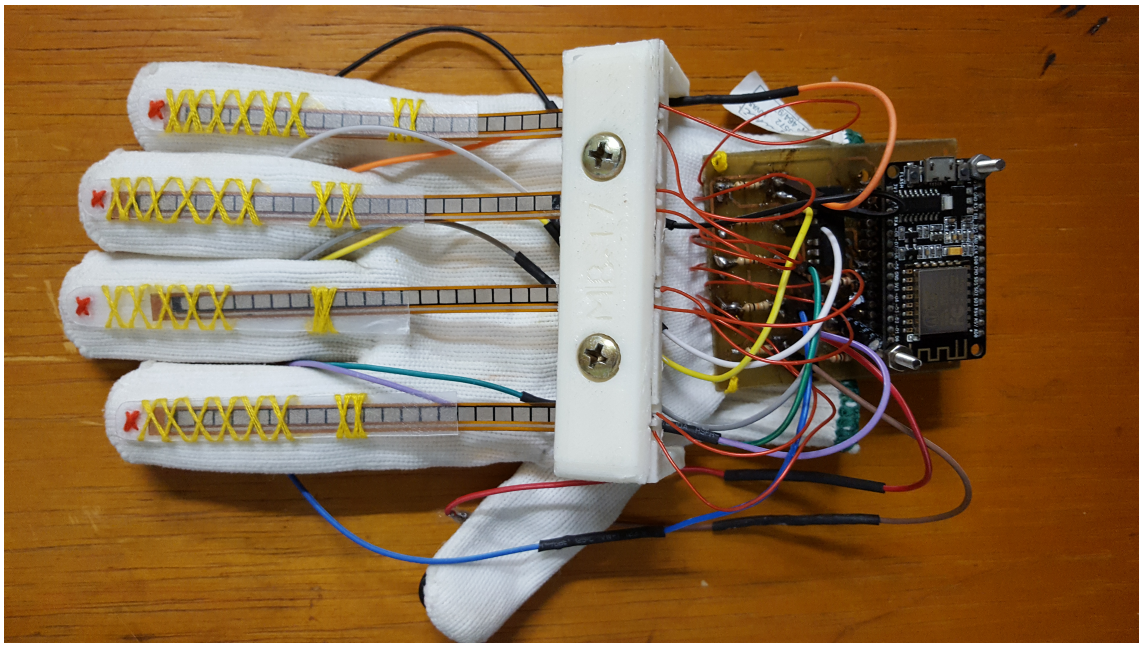
Para testar o sistema de aquisição, foram medidos valores de tensão do sinal de entrada no conversor analógico-digital com um multímetro e estes valores foram comparados com os valores entregues pelo conversor para levantar a correlação experimental entre os dados digitais e os valores analógicos e poder de fato atribuir um valor físico aos números emitidos pelo conversor. O código da aplicação foi modificado para gravar os valores de flexão obtidos pela aplicação em um arquivo txt. Neste arquivo também é salvo o instante de tempo em que os dados foram obtidos em relação ao momento em que a aplicação começou a enviar pacotes de requisição, juntamente com os valores dos dados após a filtragem.

Os sistemas de acionamento foram testados através do manuseio da aplicação, cada atuador foi acionado dez vezes por meio da colisão do *Avatar* do usuário com uma tecla do teclado virtual, na ordem do polegar ao dedo mínimo.

4 Resultados e Discussão

O protótipo resultante está apresentado na Figura 21. Ele é alimentado por uma entrada *microUSB*, que pode ser conectada com um carregador na tomada ou a uma bateria com entrada *USB*. Uma vez alimentado por uma fonte de tensão o protótipo automaticamente entra em funcionamento. O esquemático de hardware do protótipo, bem como o design de sua placa, encontram-se no apêndice A e B. É possível mensurar do sistema os valores de flexão dos dedos indicador, médio, anelar e mínimo, bem como posição e orientação da mão e do polegar. É possível enviar comandos para alterar o *duty-cicle* de cada atuador fixado na ponta de cada dedo.

Figura 21 – Protótipo *MB-17*



Fonte: O autor

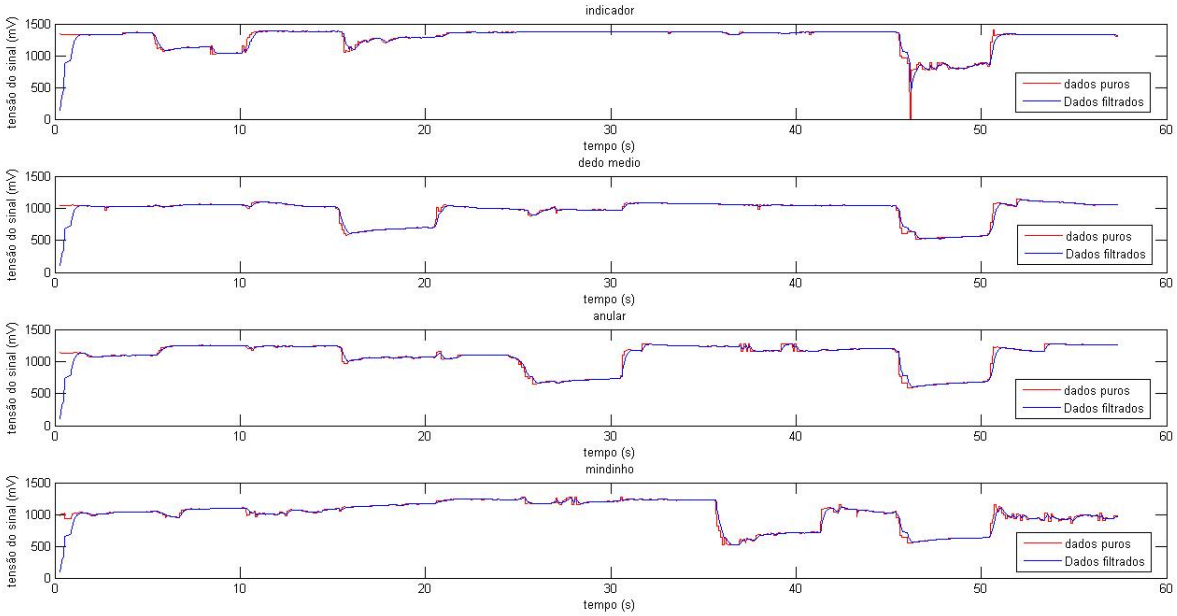
Verificou-se que, utilizando o protocolo de comunicações proposto, foi possível obter informações dos sensores da luva via *wi-fi* dentro de todo o campo de visão do sensor *Leap Motion*. Foi possível acionar todos os atuadores do sistema quando interagindo com os elementos da aplicação. Observou-se que o *Avatar* do usuário movimentava-se de acordo com a movimentação da mão do usuário. Verificou-se que, quando o *Avatar* apresenta o formato de mão, cada dedo do *avatar* que possui seu correspondente sensoriado por um *flexsensor* na luva respondeu à flexão do dedo do usuário [1].

[1] Video de demonstração do protótipo: https://www.youtube.com/watch?v=M296rRcrxho

Durante os ajustes do protótipo foi verificado que a uma taxa de aquisição de 30 amostras por segundo, a latência de dados provenientes da luva aumentava com o passar do tempo. Ao reiniciar a aplicação, com a luva ainda ligada, a latência voltava a seu valor inicial e voltava a crescer para valores em que a resposta para a flexão do dedo poderia demorar entre um e dois segundos. Ao reduzir a taxa de amostragem da aplicação para 10 amostras por segundo a latência se estabilizou nos valores iniciais da aplicação. O valor de latência após os ajustes não é perceptível ao usuário.

Como ocorreu um ajuste na taxa de amostragem também se fez necessário um ajuste do valor da constante k do filtro. Utilizando o valor base previamente calculado verificou-se a presença de ruídos perceptíveis dentro da aplicação (Os dedos do *avatar* do usuário tremiam de forma esporádica e em alta velocidade). Um ajuste cego para o valor de k foi feito antes de recálculo de um novo valor. O valor de k foi ajustado para $0,9$ e testado dentro da aplicação, resultando na redução do ruído. A eficácia da filtragem pode ser observada na Figura 22, mais testes são necessários para definir um valor ótimo de k.

Figura 22 – Gráficos de tensão sobre os sensores lida pelo conversor analógico-digital e após a filtragem ao longo do tempo

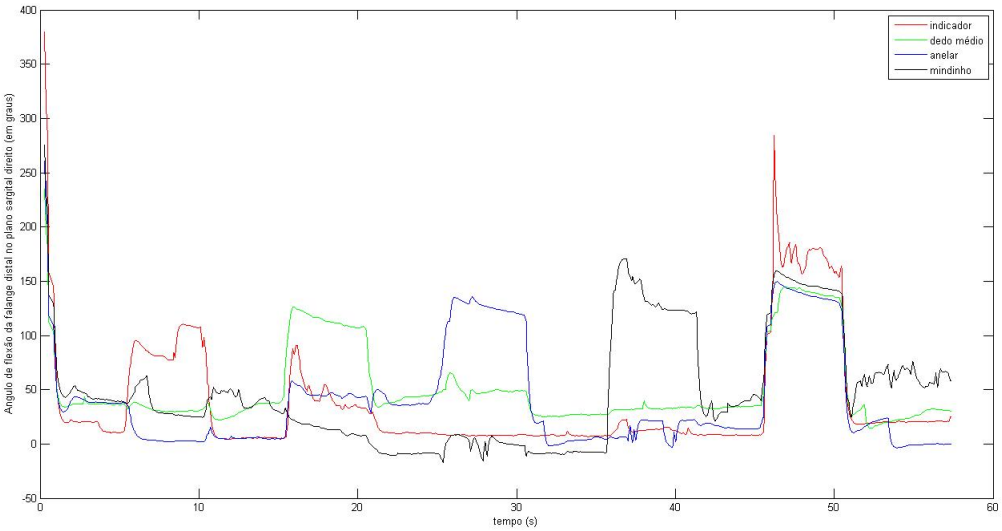


Fonte: O autor

Os valores da Figura 22 foram convertidos em ângulos conforme proposto no capítulo anterior. Foram medidos os valores aproximados máximo e mínimo de tensão (numa escala de 5mV) do sinal filtrado de cada sensor e eles foram correlacionados com os valores máximos de flexão de cada dedo da mão (aproximadamente 90 graus para cada

junta, significando que a falange distal consegue flexionar até aproximadamente 270 graus em relação à palma da mão). Mas ao fazer testes com a aplicação os movimentos dos dedos do avatar eram muito maiores do que o movimento da mão. Os fatores envolvidos nesta distorção são o fato do sensor de flexão não acompanhar a flexão da falange distal ao ponto de não se encontrar posicionado sobre ela quando a mão está fechada em punho e o *avatar* do usuário ser uma forma caricata de uma mão, com dedos longos significando que uma pequena rotação na base do dedo implica em um arco maior da ponta. Não foi estabelecido um método para se ajustar o valor de angulação máxima da aplicação de tal forma a achar o valor ótimo que mimetize o movimento da mão. Foram testados valores entre 90 e 30, múltiplos de cinco que tornassem o movimento da mão adequado (no sentido de mimetizar o movimento da mão): o valor final utilizado foi 50. Utilizando este valor para a conversão dos valores dos sensores em ângulo obtém-se o os resultados apresentados na Figura 23. Independentemente do ângulo máximo, a Figura 23 ilustra que o método de conversão de valor de tensão utilizando valores máximos e mínimos dos sensores aproxima os valores de flexão, limitando-os em valores aproximados máximos e mínimos de angulação, mesmo com o sensor sobre o dedo indicador tendo valores de tensão muito maiores que os demais dedos.

Figura 23 – Gráfico do ângulo de flexão dos dedos sensoriados ao longo do tempo



Fonte: o Autor

A verificação de perda de pacotes para a leitura dos sensores não foi feita e não se observou efeitos negativos de eventuais perdas de pacotes para a leitura dos sensores. No entanto, foram observadas perdas de pacotes no envio de comandos de acionamento dos atuadores, resultando que em algumas ocasiões o atuador não era acionado quando um

comando era enviado e em outras ocasiões o atuador não desligava quando a aplicação enviava um comando para configurar o *Duty-cicle* do atuador como zero. Estes efeitos foram observados apenas em três momentos no teste com os atuadores: uma única vez o atuador do indicador não foi acionado, o atuador do dedo anelar também deixou de ser acionado uma única vez e o atuador do dedo mínimo permaneceu acionado após o comando de desligamento ter sido enviado, entretanto o sistema respondeu ao reenvio dos comandos perdidos. Os prováveis motivos para perda de pacotes são erro no reconhecimento de bits de sinais enviados e recebidos pela antena do microcontrolador e descarte de pacote devido ao preenchimento do *Buffer* do microcontrolador. Uma possível solução para a perda de comandos de atuação é o envio sequencial de múltiplos comandos idênticos.

Os dados de rastreamento da mão e polegar conseguiram ser empregados com um grau parcial de sucesso na aplicação, respondendo aos movimentos da luva, apesar de o polegar ser facilmente suscetível à oclusão. Os dados de orientação tanto do polegar quanto da mão não puderam ser aplicados diretamente ao *avatar*, uma vez que os sistemas de coordenadas do *Leap Motion* e *Unity* diferem. Ajustes foram feitos e o *avatar* segue a orientação da mão para a maioria dos gestos, mas existem posições em que o *avatar* responde de forma diferente à orientação da mão do usuário. Por exemplo, começando com a palma da mão voltada para baixo, se o usuário supinar a mão e flexioná-la de forma que a mão fique na vertical com o dorso voltado para o sensor o *avatar* seguirá o movimento de supinação, mas ao final do movimento de flexão o *avatar* se encontrará na vertical com o dorso voltado para a direita. A causa desta divergência está provavelmente relacionada com a forma com que os dados de orientação são tratados, já que os quaternions enviados pelo sensor são traduzidos para ângulos de *Euler*, tem um de seus eixos de rotação invertido e são convertidos novamente para quaternions para serem aplicados à orientação do avatar. Este tipo de erro está relacionado com posições específicas da mão e não interferem com o funcionamento da aplicação.

Outras análises quantitativas do sistema devem ser feitas mediante o estabelecimento de um conjunto de parâmetros e especificações de interesse, criação de métodos padronizados para realizar a medida dos valores e determinação de valores alvo para comparação e qualificação dos resultados obtidos.

Uma observação final é que o *avatar* do usuário, apesar de possibilitar a demonstração do sistema, é pouco apropriado para interagir com o teclado virtual, no sentido de que a interação com as teclas é possível, o *feedback* háptico foi verificado como funcional, mas ainda não é possível de fato tocar sequencialmente as teclas do teclado virtual de forma a produzir música. Considerações e investigações no desenho do modelo e funcionalidade desenvolvidas são necessárias para gerar uma proposta que possibilite uma melhor interação com a aplicação.

5 Conclusão

A implementação física do conjunto dos subsistemas de acionamento e leitura de dados via *wi-fi* foi realizada com sucesso. Uma grande quantidade de tempo, esforço, criatividade e determinação foram necessários para determinar uma forma de fixar sensores, atuadores e uma placa de circuito impresso a uma luva de forma a gerar uma plataforma de sensoriamento e atuação háptica funcional. Muito esforço foi direcionado ao desenvolvimento de *firmware* e outra grande parcela de empenho foi empregada na criação da aplicação exemplo, seus elementos gráficos, lógica de funcionamento e forma de comunicação com a luva de dados. Uma proposta de protocolo de comunicação em camada de aplicação foi gerada e pode ser expandida para se tornar um padrão de comunicação entre sistemas embarcados e um controlador. Um estudo tanto da literatura, quanto de produtos comerciais foi realizado e um ano inteiro de desenvolvimento está compilado neste trabalho.

O protótipo resultante deste projeto apresentou resultados promissores, sendo capaz de manipular um *avatar* em ambiente virtual, com amplo potencial para propostas de melhorias. Dito isto, pode-se inferir que o trabalho atingiu o objetivo de criar uma luva de dados para futuras investigações, testes e possibilidade de ser utilizada em aplicações.

Trabalhos que poderão dar seguimento a este são a proposição de um conjunto de parâmetros para luvas de dados e testar o protótipo deste trabalho no intuito de levantar suas especificações dentro dos parâmetros propostos; a proposição de testes padrão para levantamento de características de luvas de dados e levantar as características da luva resultante deste trabalho. Desenvolver uma forma de monitorar os movimentos do polegar incluindo flexão das falanges e adução/abdução da falange metacarpal através de sensores mecânicos atrelados a uma estrutura de suporte que não obstrua os movimentos do dedo; a atualização dos componentes de *Hardware* do protótipo e criar aplicações.

Referências

ALVES, G.; SANTOS, G. A. Haptic Glove Development for Passive Control of a Robotic Arm. p. 1–6, 2016. Citado 2 vezes nas páginas 27 e 28.

ARIYANTO, M. et al. Development of a low cost anthropomorphic robotic hand driven by modified glove sensor and integrated with 3D animation. *IECBES 2016 - IEEE-EMBS Conference on Biomedical Engineering and Sciences*, p. 341–346, 2017. Citado 3 vezes nas páginas 23, 27 e 31.

BOUZIT, M. et al. The Rutgers Master II - New design force-feedback glove. *IEEE/ASME Transactions on Mechatronics*, v. 7, n. 2, p. 256–263, 2002. ISSN 10834435. Citado na página 28.

FANG, B. et al. A robotic hand-arm teleoperation system using human arm/hand with a novel data glove. *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, p. 2483–2488, 2016. Citado 4 vezes nas páginas 23, 27, 28 e 31.

FLORES, M. B. H. et al. User-oriented finger-gesture glove controller with hand movement virtualization using flex sensors and a digital accelerometer. *2014 International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management, HNICEM 2014 - 7th HNICEM 2014 Joint with 6th International Symposium on Computational Intelligence and Intelligent In*, p. 4–6, 2014. Citado 3 vezes nas páginas 27, 28 e 30.

HODA, M.; HAFIDH, B.; SADDIK, A. E. Vibro -Tactile Actuator. 2013. Citado 3 vezes nas páginas 23, 27 e 28.

JACK, D. et al. Virtual reality-enhanced stroke rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, v. 9, n. 3, p. 308–318, 2001. ISSN 15344320. Citado 2 vezes nas páginas 23 e 28.

JAIJONGRAK, V.-R.; CHANTASUBAN, S.; THIEMJARUS, S. Towards a BSN-based gesture interface for intelligent home applications. *Iccas-Sice, 2009*, p. 5613–5617, 2009. Citado 4 vezes nas páginas 23, 27, 28 e 30.

JOFRE, L. N. et al. Non-verbal communication interface using a data glove. *2016 IEEE Congreso Argentino de Ciencias de la Informática y Desarrollos de Investigación (CACIDI)*, p. 1–6, 2016. Disponível em: <http://ieeexplore.ieee.org/document/7786008/>. Citado na página 23.

KEYES, A.; D'SOUZA, M.; POSTULA, A. Navigation for the blind using a wireless sensor haptic glove. *Proceedings - 2015 4th Mediterranean Conference on Embedded Computing, MECO 2015 - Including ECyPS 2015, BioEMIS 2015, BioICT 2015, MECO-Student Challenge 2015*, p. 361–364, 2015. ISSN 2377-5475. Citado 3 vezes nas páginas 24, 27 e 28.

KUMAR, P.; RAUTARAY, S. S.; AGRAWAL, A. Hand data glove: A new generation real-time mouse for human-computer interaction. *2012 1st International Conference on*

Recent Advances in Information Technology, RAIT-2012, p. 750–755, 2012. Citado 3 vezes nas páginas 27, 28 e 30.

LOW, J. H. et al. A Hybrid Tele-manipulation System using a Sensorized 3D-printed Soft Robotic Gripper and a Soft Fabric-Based Haptic Glove *. v. 2, n. 2, p. 1–8, 2016. ISSN 2377-3766. Citado 2 vezes nas páginas 23 e 27.

MAJEAU, L. et al. Dataglove for consumer applications Low cost dataglove using optical fiber sensor. *7th Workshop on Fibre and Optical Passive Components (WFOPC)*, p. 1–4, 2011. Citado 3 vezes nas páginas 23, 27 e 30.

PITITEERAPHAB, Y. The Movement of The Robot Arm. *The 2015 Biomedical Engineering International Conference (BMEiCON-2015)*, p. 3–6, 2015. Citado 4 vezes nas páginas 23, 27, 28 e 31.

RINDERKNECHT, M. D. et al. Objective assessment of vibrotactile mislocalization using a Haptic Glove. *IEEE International Conference on Rehabilitation Robotics*, v. 2015-Septe, p. 145–150, 2015. ISSN 19457901. Citado 2 vezes nas páginas 24 e 28.

SBERNINI, L.; PALLOTTI, A.; SAGGIO, G. Evaluation of a Stretch Sensor for its inedited application in tracking hand finger movements. *2016 IEEE International Symposium on Medical Measurements and Applications, MeMeA 2016 - Proceedings*, 2016. Citado 2 vezes nas páginas 27 e 31.

SHAN, Y.; LINGJIANG, K. Research on characteristic extraction of human gait. In: *3rd International Conference on Bioinformatics and Biomedical Engineering, iCBBE 2009*. [S.l.: s.n.], 2009. p. 2–5. ISBN 9781424429028. Citado na página 44.

SHEN, Z. et al. A soft stretchable bending sensor and data glove applications. *Robotics and Biomimetics*, v. 3, n. 1, p. 22, 2016. ISSN 2197-3768. Disponível em: <http://jrobio.springeropen.com/articles/10.1186/s40638-016-0051-1>. Citado 2 vezes nas páginas 27 e 31.

STURMAN, D. J.; ZELTZER, D. A survey of glove-based input. *Computer Graphics and Applications, IEEE*, v. 14, n. 1, p. 30–39, 1994. ISSN 02721716. Citado 3 vezes nas páginas 26, 28 e 29.

TARCHANIDIS, K. N.; LYGOURAS, J. N. Data glove with a force sensor. *IEEE Transactions on Instrumentation and Measurement*, v. 52, n. 3, p. 984–989, 2003. ISSN 00189456. Citado 2 vezes nas páginas 27 e 29.

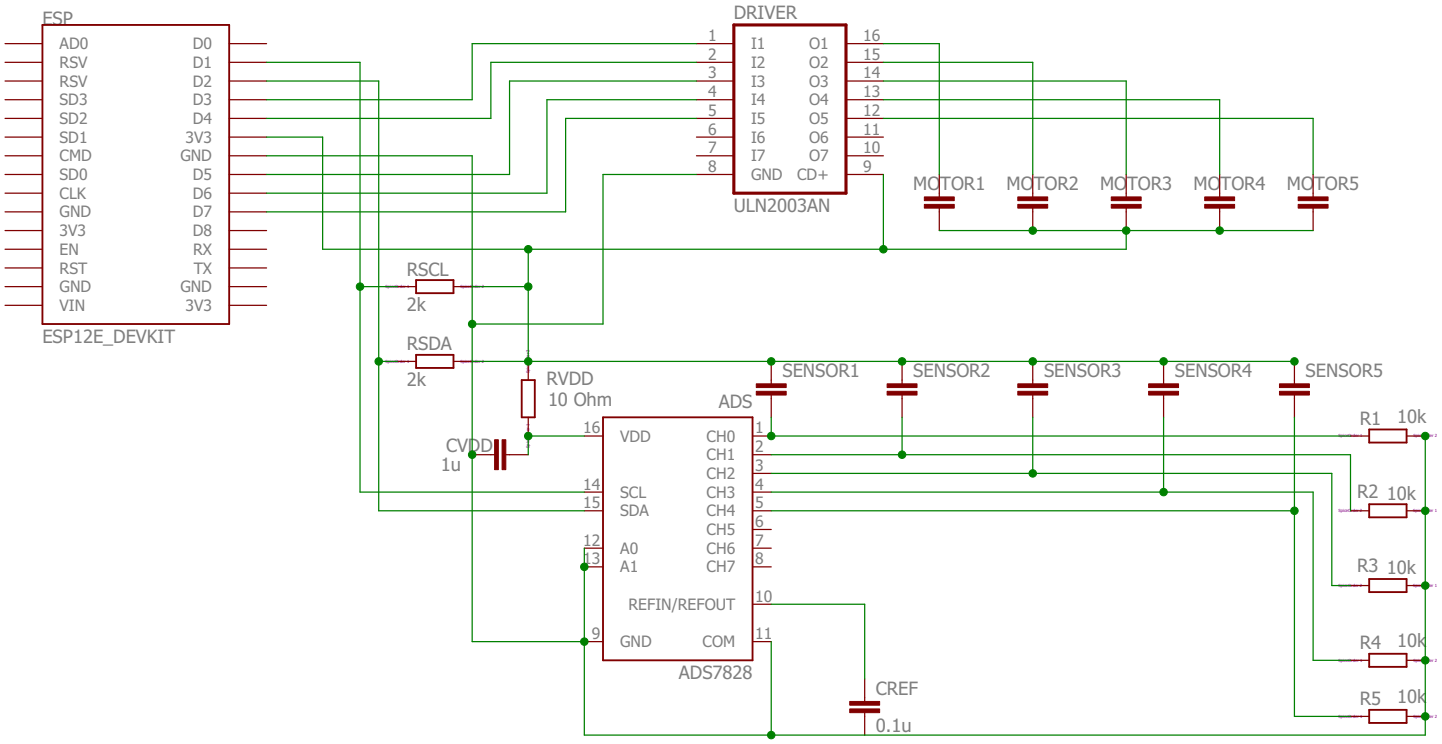
TOGNETTI, A. et al. Characterization of a novel data glove based on textile integrated sensors. *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, p. 2510–2513, 2006. ISSN 05891019. Citado 2 vezes nas páginas 27 e 29.

WANG, Y.; ZHANG, W. Data glove control of robot hand with force telepresence. *2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015*, p. 314–319, 2016. Citado 4 vezes nas páginas 23, 27, 28 e 31.

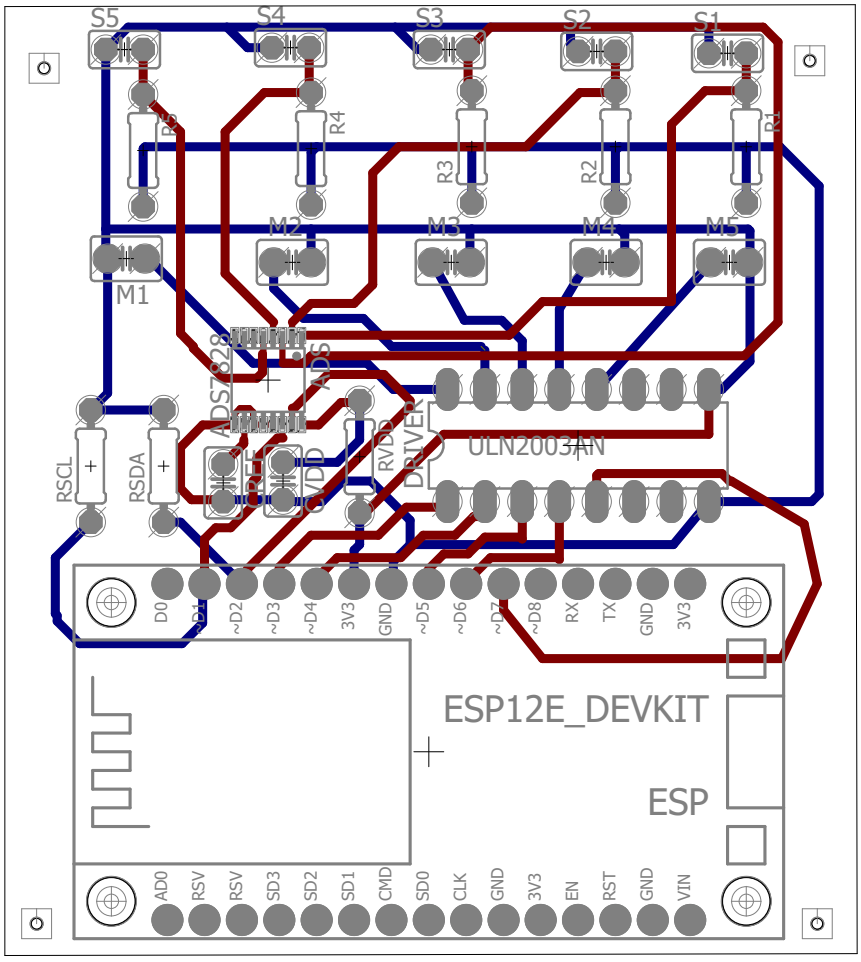
ZUBRYCKI, I.; GRANOSIK, G. Novel haptic glove-based interface using jamming principle. *2015 10th International Workshop on Robot Motion and Control, RoMoCo 2015*, p. 46–51, 2015. Citado na página 28.

Apêndices

APÊNDICE A – Esquemático do circuito


ESP
AD0
RSV
RSV
SD3
SD2
SD1
CMD
SD0
CLK
GND
3V3
EN
RST
GND
VIN
D0
D1
D2
D3
D4
3V3
GND
D5
D6
D7
D8
RX
TX
GND
3V3
ESP12E_DEVKIT
DRIVER
I1
I2
I3
I4
I5
I6
I7
GND
O1
O2
O3
O4
O5
O6
O7
CD+
ULN2003AN
MOTOR1
MOTOR2
MOTOR3
MOTOR4
MOTOR5
RSCL
2k
RSDA
2k
RVDD
10 Ohm
CVDD
1u
ADS
VDD
SCL
SDA
A0
A1
GND
CH0
CH1
CH2
CH3
CH4
CH5
CH6
CH7
REFIN/REFOUT
COM
ADS7828
CREF
0.1u
SENSOR1
SENSOR2
SENSOR3
SENSOR4
SENSOR5
R1 10k
R2 10k
R3 10k
R4 10k
R5 10k

APÊNDICE B – Modelo da placa de circuito impresso


S5
S4
S3
S2
S1
R5
R4
R3
R2
R1
M1
M2
M3
M4
M5
ADS1828
RSCL
RSDA
RVDD
DRIVER
ULN2003AN
D0
~D1
~D2
~D3
~D4
3V3
GND
~D5
~D6
~D7
~D8
RX
TX
GND
3V3
ESP12E_DEVKIT
ESP
AD0
RSV
RSV
SD3
SD2
SD1
CMD
SD0
CLK
GND
3V3
EN
RST
GND
VIN

APÊNDICE C – Códigos de Firmware

```
-- init.lua --
-- Código de inicialização do micro controlador
-- Criado por: Anderson Henrique Dantas Borba

-- Descrição: Este código realiza a conexão do NodeMCU com
-- uma rede Wi-fi com ssid: nomeDaRede e senha: senha
-- Logo em seguida ele executa o arquivo main.lua.

-- O código foi segmentado em duas partes pois é possível
-- alterar este código para realizar o setup da rede, mas
-- devido a restrições de memória do NodeMCU isto não foi
-- feito

-- Obs: IMPORTANTE!! o sistema do NODEMCU armazena comentários!
-- Para conseguir rodar este código pelo ESPlorer é necessário
-- apagar comentários.

-- Obs2: Para maiores informações referente a cada função do
-- NodeMCU consultar a documentação do NodeMCU.
-- (https://nodemcu.readthedocs.io/en/master/).

--init.lua--

ssid="nomeDaRede"
pass= "senha"

--configurar internet wireless
wifi.setmode(wifi.STATION)

--Cinfigurar WiFi
wifi.sta.config(ssid,pass)

--entra na parte principal do código
dofile("main.lua")
```

```
-- main.lua --
```

```
-- Rotina principal do micro controlador
-- Criado por: Anderson Henrique Dantas Borba

-- Layout de Hardware: A comunicação I2C com o ADS7828 é feita
-- através dos pinos D1 e D2 do NodeMCU (Sendo o pino D1 o SCL
-- e o pino D2 o SDA). O acionamento dos atuadores em cada dedo
-- é feito através dos pinos D3 ao D7. Todos os pinos são portas
-- de propósito geral digitais.

-- Descrição: Este código utiliza duas funções de interrupção
-- por tempo (tmr.alarm), uma função de interrupção por recep-
-- ção de pacote UDP (serv:on) e uma subrotina para a leitura
-- do ADS7828 (readsen).

-- A interrupção por tempo de índice 0 é uma função de debug
-- que permite observar se o micro controlador está conecta-
-- do com a rede, quando conectado mostra as informações: IP
-- do micro controlador, máscara de rede e Gatewy padrão e
-- encerra a interrupção.

-- A interrupção por tempo de índice 1 é uma função para lei-
-- tura repetida de 4 canais do conversor analógico-digital
-- ADS7828 a cada 100 milissegundos (em adição ao tempo neces-
-- sário para executar a função) e armazenar os valore lidos
-- nas variáveis adc0_value, adc1_value, adc2_value, adc3_value.

-- A interrupção por recepção de pacote UDP implementa o
-- protocolo DRESS, ao receber um pacote contendo uma instrução
-- DRESS, o micro controlador irá executar a instrução e
-- responder ao remetente.

-- A subrotina de leitura do ADS7828 implementa as instruções
-- I2C para leitura de um canal do conversor. O valor de entrada
-- da função é o valor C (de 3 bits) da tabela II do datasheet
-- do conversor.

-----------------------------------
-- Verificação da conexão Wi-fi --
-----------------------------------
```

```
tmr.alarm(0, 1000, 1, function()
    if wifi.sta.getip() == nil then
        print("Conecting to AP...\n")
    else
        ip, nm, gw= wifi.sta.getip()
        print("IP Info: \nIP: ", ip)
        print("Netmask: ", nm)
        print("Gateway: ",gw,'\n')
        tmr.stop(0)
    end
end)


-----------------------------------
--- Variaveis Globais -------------
-----------------------------------

adc0_value= 512 -- Valor referente ao sensor no dedo mindinho
adc1_value= 512 -- Valor referente ao sensor no dedo anular
adc2_value= 512 -- Valor referente ao sensor no dedo médio
adc3_value= 512 -- Valor referente ao sensor no dedo indicador

-----------------------------------
--- Setting PWM -------------------
-----------------------------------
pwm.setup(3,500,0) -- PWM do polegar
pwm.setup(4,500,0) -- PWM do dedo indicador
pwm.setup(5,500,0) -- PWM do dedo médio
pwm.setup(6,500,0) -- PWM do dedo anular
pwm.setup(7,500,0) -- PWM do dedo mindinho

-- Inicio do PWM
pwm.start(3)
pwm.start(4)
pwm.start(5)
pwm.start(6)
pwm.start(7)

-----------------------------------
```

```
-- UDP Server ------------------------
-------------------------------------

print("starting server...")                    -- DEBUG indica
   o inicio do servidor UDP
serv = net.createServer(net.UDP,1000)          -- Declaração do
    servidor
print("server created!")                       -- DEBUG indica
   que o servidor foi criado
    serv:on("receive", function(serv, payload) -- Função de
       interrupção por recepção de pacote
        print(payload)                         -- DEBUG para
           mostrar o conteúdo do pacote
        if payload:sub(1,1)== "1" then         -- Pacote é
           comando para ler sensores
            if payload:sub(2,2) == "0" then --comdndo para ler o
                adc
                serv:send('10'..string.format("%04d",adc3_value)
                    ..','.. string.format("%04d",adc2_value) ..'
                   ,' .. string.format("%04d",adc1_value) ..','
                   .. string.format("%04d",adc0_value))
            elseif payload:sub(2,2)> "5" then
                serv:send("401")
            else
                serv:send("402")
            end
        elseif payload:sub(1,1)== "2" then     -- Comando para
            acionar um periferico
            if payload:sub(2,2)== "0" then     -- Comando para
                acionar o pwm0
                value= payload:sub(3,6)
                pwm.setduty(3,value)
                serv:send("201")
            elseif payload:sub(2,2)== "1" then -- Comando para
                acionar o pwm1
                value= payload:sub(3,6)
                pwm.setduty(4,value)
                serv:send("211")
            elseif payload:sub(2,2)== "2" then -- comando para
```

```
            acionar o pwm2
            value= payload:sub(3,6)
            pwm.setduty(5,value)
            serv:send("221")
        elseif payload:sub(2,2)== "3" then   -- comando para
            acionar o pwm3
            value= payload:sub(3,6)
            pwm.setduty(6,value)
            serv:send("231")
        elseif payload:sub(2,2)== "4" then   -- comando para
            acionar o pwm4
            value= payload:sub(3,6)
            pwm.setduty(7,value)
            serv:send("241")
        elseif payload(2,2)> 4 then
            serv:send("401")
        else
            serv:send("402")
        end
    elseif payload:sub(1,2) == "30" then
        serv:send("31,5")
    else
        serv:send("402")                     -- Resposta
           para comando inválido
    end
end)                                         -- Fim da
    função.

serv:listen(8910)  -- O servidor UDP escuta a porta 8910

i2c.setup(0,2,1,i2c.SLOW)  -- declaração para inicializar a
   comunicação I2C

-- protocolo de leitura I2C do ADS7828 (segundo o datasheet do
   conversor)
function read_sen(sensor)
    i2c.start(0)
    if i2c.address(0,72,i2c.TRANSMITTER) == true then -- Comando
        para escrever no ADS
```

```
        if i2c.write(0,128+(sensor*16)+12) == 1 then -- Escreveu
            um comando de um byte com sucesso no ADS
            i2c.stop(0)
            i2c.start(0)
            if i2c.address(0,72,i2c.RECEIVER) -- Comando para
               ler dados do ADS
                c=i2c.read(0,2) -- Ler dois bytes do ADS
                i2c.stop(0)
                return c
            end
        end
    end
    return nil
end

tmr.alarm(1,10,1,function() -- Função de leitura repetida (A
   cada 100 ms) de 4 canais do ADS7828
    sen1= read_sen(0) -- leitura do valor do sensor sobre o
       mindinho
    if sen1== nil then
        sen1 = 0
    else
       adc0_value= string.byte(sen1)*128 + string.byte(sen1,2)
    end
    sen1= read_sen(4) -- leitura do valor do sensor sobre o
       anular
    if sen1== nil then
        sen1 = 0
    else
       adc1_value= string.byte(sen1)*128 + string.byte(sen1,2)
    end
    sen1= read_sen(1) -- leitura do valor do sensor sobre o
       médio
    if sen1== nil then
        sen1 = 0
    else
       adc2_value= string.byte(sen1)*128 + string.byte(sen1,2)
    end
    sen1= read_sen(5) -- leitura do valor do sensor sobre o
```

```
        indicador
    if sen1== nil then
        sen1 = 0
    else
        adc3_value= string.byte(sen1)*128 + string.byte(sen1,2)
    end
end)
```

APÊNDICE D – Código de implementação do protocolo DRES

```
/*
-- Conjunto de funções em C# que implementam o conjunto de
   comandos de Mestre do protocolo DRES
-- Criado por: Anderson Henrique Dantas Borba

-- Descrição: Conjunto de funções para montar um comando DRES.
   Essas funções são: Do, Read e Status.

-- A função Do possui como entrada o índice da variável ou
   conjunto de variáveis mapeado dentro do protocolo, o número
   de valores a ser enviado e valores a serem enviados para
   alterar valores dentro do micro controlador. Ela retorna um
   vetor de bytes prontos para serem enviados via UDP (ou TCP/IP
   ).

-- A função Read possui como entrada o índice mapeado dentro do
   protocolo DRESS de valor ou valores a serem lidos. A função
   retorna um vetor de bytes prontos para serem enviados via UDP
    (ou TCP/IP)

-- A função Status não possui entrada (O comando para requisição
    de status é sempre o mesmo) e retorna um vetor de bytes
   prontos para serem enviados.
*/


using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace DRES
{
```

```
    public class ProtocoloDRES
    {
        public static byte[] Read(int index)
        {
            byte[] comando = new byte[2];
            comando[0] = (byte)'1';
            comando[1] = (byte)(index.ToString())[0];
            return comando;
        }

        public static byte[] Do(int atuador, int infos, int[]
            valores)
        {
            byte[] comando = new byte[6];
            comando[0] = (byte)'2';
            comando[1] = (byte)atuador.ToString()[0];
            for(int i=0; i<infos; i++){
              for(int j=0; i<4; i++)
              {
                comando[2+5*i+j] = (byte)valores[i].ToString("
                    0000")[j];
              }
              if(i!=infos-1){
                comando [2+5*i+5]=','
              }
            }
            return comando;
        }

        public static byte[] Status()
        {
            byte[] comando = new byte[2];
            comando[0] = (byte)'3';
            comando[1] = (byte)'0';
            return comando;
        }
    }
}
```

APÊNDICE E – *Script* exemplo de implementação de um servidor *UDP* para *Unity* utilizando o protocolo *DRES*

```
/*
-- Script para Unity em C# que implementa um cliente UDP para
   comunicar-se com a luva via protocolo DRESS e implementa
   funções para tratamento dos dados colhidos.

-- Criado por: Anderson Henrique Dantas Borba

-- Descrição: Este script apresenta 9 funções: Conect(), Start()
   , Disconect(), Set_Actuator(int index value, int pwmvalue),
   isConnected(), Update(), OnApplicationQuit(), lowPassFilter(
   float[] inputs) e getFlexion_Values().

-- Por herdar da classe MonoBehaviour, a classe TouchIt
   necessariamente precisa implementar as funções Start() e
   Update(). A função Start() é chamada uma única vez quando o
   programa se inicia, já a função update é chamada a cada
   quadro (Frame).

-- A função Connect() faz a conecção do cliente UDP com a luva e
    faz o setting para que o comando client.Recieve() não
   bloqueie o fluxo da aplicação.

-- A função start() inicializa as variaveis, setta a taxa de
   quadros da aplicação para 30 quadros por segundo e chama a
   função Connect().

-- A função Disconect() Encerra a conecção com o servidor UDP.

-- A função Set_Actuator(int index, int pwmvalue) envia um
   comando DRESS do tipo Do para modificar o duty-cicle do
   atuador correspondente ao index para o pwmValue (o valor do
```

```
    duty-cicle deve ser escrito entre 0 e 1023, sendo 512 o ponto
     médio) SE a aplicação estiver conectada com o servidor UDP.

— A função IsConnected() retorna true se a aplicação estiver
    conectada com o servidor UDP e false em caso contrario.

— A função Update() é a responsável pelo envio constante de
    requisições DRESS do tipo Read ao servidor UDP (Ela envia um
    request de dados a cada 50 ms aproximadamente) e aguarda de
    forma a não bloquear o fluxo da aplicação por respostas aos
    comandos de leitura enviados. A função sobrescreve as
    respostas na string recievedData.

— OnApplicationQuit() é uma sobrecarga do método homônimo da
    classe Application, que é chamado quando a aplicação é
    finalizada. Neste caso essa função chama a função Disconect,
    encerrando a conexão com o cliente UDP.

— A função lowPassFilter implementa o filtro digital passa
    baixas (um filtro linear de primeira ordem) on-line, onde a
    entrada é um vetor de até 4 valores referentes a uma amostra
    de até quatro sinais a serem filtrados e a sáida do filtro é
    sinal um vetor com 4 valores contendo a amostra dos sinais
    filtrados.
— IMPORTANTE: Cada espaço do vetor a ser filtrado deve conter
    amostras de um mesmo sinal

— A função getFlexion_Values() separa a string recievedData em
    substrings contendo os valores individuais das funções
*/


using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Text;
```

```
using System.Collections.Generic;
using DRES;// Importante!
using System.IO;


public class TouchIt : MonoBehaviour {

    //Variaveis associadas com a taxa de envio de requisição
    private float sampling_Time = 0.05f;
    private float timer=0;

    //Variaveis de UI e debug
    public Text recievedMessages;
    public Text filteredData;
    public InputField IP_enter;
    public string recievedData= "0,0,0,0,0";

    //Variaveis associadas com o Cliente UDP
    System.Net.IPEndPoint IP { get; set; }
    public bool connected=false;
    private static UdpClient client;
    private float timeCounter;

    //Variaveis do filtro passa baixas
    float[] acumulator = new float[4];
    float k = 0.9f;//0.8668779f;

    public void Connect()
    {
        IP = new System.Net.IPEndPoint(IPAddress.Parse(IP_enter.
            text), 8910);
        client = new UdpClient(8910);
        client.Connect(IP);
        client.Client.Blocking = false;
        connected = true;
    }

    void Start()
    {
```

```
        IP_enter.text = "192.168.0.99"; //ultimo IP da luva
            quando
        recievedData = "  started the Tread";
        for(int i=0; i<acumulator.Length; i++)
        {
            acumulator[i] = 0;
        }
        Application.targetFrameRate = 30;
        Connect();
    }

    public void Disconnect()
    {
        client.Close();
        connected = false;
    }

    public void Set_Actuator(int index_value, int pwm_value)
    {
        System.Net.IPEndPoint EP = IP;//new System.Net.
            IPEndPoint(IP.Address, IP.Port);
        client.Send(ProtocoloDRES.Faca(index_value, pwm_value)
            ,6);
        Debug.Log("Command Sent: " + index_value.ToString());
    }

    public bool isConnected()
    {
        return connected;
    }

    void Update()
    {
        if (connected)
        {
            timer = timer + Time.deltaTime;
            timeCounter = timeCounter + Time.deltaTime;
            filteredData.text = timeCounter.ToString();
```

```
            System.Net.IPEndPoint EP = IP;//new System.Net.
                IPEndPoint(IP.Address, IP.Port);
            if (timer > sampling_Time)
            {
                timer = 0;
                client.Send(ProtocoloDRES.Leia(0),2);
            }
            try
            {
                byte[] data = client.Receive(ref EP);
                recievedData = Encoding.ASCII.GetString(data);
            }
            catch
            {

            }
            recievedMessages.text = recievedData;
        }
    }

    void OnApplicationQuit()
    {
        client.Close();
    }

    private float[] lowPassFilter(float[] inputs)
    {
        for(int i=0; i<inputs.Length; i++)
        {
            acumulator[i] = k * acumulator[i] + (1-k)*inputs[i];
        }
        return acumulator;
    }

    public float[] getFlexion_Values()
    {
        if (connected && recievedData.Length>5)
        {
            string rData = recievedData.Substring(2);
```

```
            string[] individualData;
            individualData = rData.Split(',');
            if (individualData.Length != 4)
            {
                return null;
            }
            float[] integerData= new float[4];
            for(int i=0; i<4; i++)
            {
                if(!float.TryParse(individualData[i],out
                   integerData[i]))
                {
                    return null;
                }
            }
            integerData = lowPassFilter(integerData); //Á pesar
               do nome, neste ponto o valor é de ponto flutuante
               .
            return integerData;
        }
        else
        {
            return null;
        }
    }
}
```