UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
ESCOLA DE ENGENHARIA
ENG. DE CONTROLE E AUTOMAÇÃO

ANTÔNIO DAI PRÁ AIROLDI - 00252877

SISTEMA DE MONITORAMENTO DE SENSORES BIOMÉDICOS PARA ESTUDO DE CORRELAÇÃO ENTRE O EQUILÍBRIO CORPORAL E A DOENÇA DE PARKINSON

Porto Alegre
2020

ANTÔNIO DAI PRÁ AIROLDI - 00252877

SISTEMA DE MONITORAMENTO DE SENSORES BIOMÉDICOS PARA ESTUDO DE CORRELAÇÃO ENTRE O EQUILÍBRIO CORPORAL E A DOENÇA DE PARKINSON

Trabalho de Conclusão de Curso (TCC-CCA) apresentado à COMGRAD-CCA da Universidade Federal do Rio Grande do Sul como parte dos requisitos para a obtenção do título de *Bacharel em Eng. de Controle e Automação* .

ORIENTADOR:
Prof. Dr. Mário Roland Sobczyk Sobrinho

Porto Alegre
2020

ANTÔNIO DAI PRÁ AIROLDI - 00252877

SISTEMA DE MONITORAMENTO DE SENSORES BIOMÉDICOS PARA ESTUDO DE CORRELAÇÃO ENTRE O EQUILÍBRIO CORPORAL E A DOENÇA DE PARKINSON

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção dos créditos da Disciplina de TCC do curso *Eng. de Controle e Automação* e aprovado em sua forma final pelo Orientador e pela Banca Examinadora.

Orientador: _____
Prof. Dr. Mário Roland Sobczyk Sobrinho, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Banca Examinadora:

Prof. Dr. Mário Roland Sobczyk Sobrinho, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Tiago Becker, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Marcelo Götz, UFRGS
Doutor pela Universidade Federal do Rio Grande do Sul – Porto Alegre, Brasil

Prof. Dr. Alexandre Severo de Pinho, UFCSPA
Doutor pela Universidade Federal de Ciências da Saúde de Porto Alegre – Porto Alegre, Brasil

_____
Marcelo Götz
Coordenador de Curso
Eng. de Controle e Automação

Porto Alegre, novembro de 2020.

AGRADECIMENTOS

À Universidade Federal do Rio Grande do Sul, UFRGS, pela oportunidade de realização de estudos.

À Empresa Miotec pelo compartilhamento dos protocolos de comunicação com o dispositivo Miotool 400.

Ao amigo Helton Moraes, ex-desenvolvedor da Miotec, que auxiliou no entendimento das rotinas de aquisição do Miotool 400.

RESUMO

O presente trabalho tem por objetivo apresentar o desenvolvimento de um dispositivo de aquisição de dados de um conjunto de sensores biomédicos, no estudo das características do controle postural de pacientes com doença de Parkinson. Foram utilizados um Eletromiógrafo comercial da marca Miotec e os sensores inerciais de um telefone celular. O dispositivo de aquisição tem por função sincronizar os sinais provenientes destes dispositivos, permitindo uma análise coerente de um conjunto maior de dados em uma única interface amigável ao usuário. Este trabalho foi realizado em uma parceria entre as universidades UFRGS e UFCSPA. A ferramenta será utilizada nos próximos estudos sobre a Doença de Parkinson, permitindo uma análise mais ampla dos ensaios.

Palavras-chave: Automação e Controle, Processamento de Sinais, Aquisição de Dados, Doença de Parkinson, Controle Motor, Equilíbrio Postural.

ABSTRACT

This work presents the development of a supervisory device for data acquisition using a set of biomedical sensors and IMU sensors from a phone device to study the body balance characteristics on patients with Parkinson's disease. The supervisory device synchronizes the signals coming from these devices in real time, allowing a coherent analysis of a larger dataset in a single user-friendly interface. This work is a partnership between the universities UFRGS and UFCSPA. Finally, the system was able to synchronize the sensors and aims to be used on the next studies about Parkinson Disease.

Keywords: Automation and Control, Signal Processing, Data Acquisition, Parkinson Disease, Motor Control, Postural Balance.

SUMÁRIO

LISTA DE ILUSTRAÇÕES 8

LISTA DE TABELAS 9

LISTA DE ABREVIATURAS 10

1 INTRODUÇÃO 11
1.1 Objetivos 11
1.2 Justificativa 12

2 REVISÃO DA LITERATURA 13
2.1 Doença de Parkinson 13
2.1.1 Parkinsonismo 13
2.1.2 Diagnóstico da doença 14
2.1.3 Causas 14
2.1.3.1 Deficiência de Dopamina 14
2.1.3.2 Fatores Genéticos 14
2.1.3.3 Fatores Ambientais 15
2.2 Correlação entre a propriocepção e o equilíbrio com a DP 15
2.2.1 Instrumentação clínica para o diagnóstico de DP 16
2.2.1.1 Ressonância Magnética do Cérebro 16
2.2.1.2 Eletromiografia de Superfície 16
2.2.1.3 Análise química do transportador de Dopamina 17
2.2.1.4 Dinamômetro Isocinético 18
2.2.1.5 Baropodômetro 18
2.2.1.6 *IMU* 18
2.3 *SCADA* 20

3 DESENVOLVIMENTO 22
3.1 Sensores utilizados 23
3.1.1 Atividade Muscular 23
3.1.2 Equilíbrio Corporal 23
3.2 Algoritmo de execução 24
3.2.1 Rotina de aquisição do EMG 24
3.2.2 Rotina de aquisição dos sensores do telefone celular 26
3.2.3 Rotina de Interface de Usuário 27
3.3 Sistema de Aquisição 27

4 RESULTADOS 30
4.1 Testes de execução 30
4.2 Ruído de medição 31
4.3 Testes planejados 33

5 CONCLUSÃO 35

APÊNDICE A - CÓDIGO-FONTE DO SISTEMA DE AQUISIÇÃO . . 36

APÊNDICE B - CÓDIGO-FONTE DE INTERFACE GRÁFICA DO SISTEMA DE AQUISIÇÃO 55

ANEXO A - PROTOCOLO DE MENSAGENS DO MIOTOOL 400 . . 70

REFERÊNCIAS 75

LISTA DE ILUSTRAÇÕES

1 Representação esquemática dos sintomas motores da DP em Diagrama de Blocos. Fonte: Autor 15
2 Diagrama esquemático de um dispositivo de EMG Fonte: (LUCA, 2002) 17
3 Comparativo entre exames DaTScan normal (Esquerda) e anômalo (direita). Fonte: (ASSOCIATION, 2019) 17
4 Aspectos construtivos de um acelerômetro utilizado em dispositivos móveis. Fonte: (GUJARATI, 2020) 19
5 Aspectos construtivos de um giroscópio utilizado em dispositivos móveis. Fonte: (ST. LOUIS, 2018) 20
6 Sala de controle central da usina hidrelétrica Itaipu Binacional. Fonte: (CBIE, 2019) 21
7 Diagrama de funcionamento da aplicação. Fonte: Autor 22
8 EMG Miotool 400 Fonte: (MENEZES ARRIAL, s.d.) 23
9 Fluxograma da Thread de aquisição do EMG. Fonte: Autor 25
10 Fluxograma da Thread de aquisição da IMU. Fonte: Autor 26
11 Aplicativo *Open-Source SensorStreamer* Fonte: (JAN MRÁZEK, 2020) 27
12 Interface gráfica do Sistema de aquisição Fonte: Autor 28
13 Dados obtidos do ensaio com o acelerômetro Fonte: Autor 30
14 Dados obtidos do ensaio com o giroscópio Fonte: Autor 31
15 Dados obtidos do ensaio com o EMG Fonte: Autor 31
16 Dados obtidos do ensaio com o EMG utilizando o software Miotec Suite Fonte: Autor 32
17 FFT do sinal amostrado utilizando o software Miotec Suite Fonte: Autor 33

LISTA DE TABELAS

LISTA DE ABREVIATURAS

DP	Doença de Parkinson
EMG	Eletromiografia
FFT	*Fast Fourier Transform*
IMU	*Inercial Measurement Unit*
UFCSPA	Universidade Federal de Ciências da Saúde de Porto Alegre
UFRGS	Universidade Federal do Rio Grande do Sul

1 INTRODUÇÃO

A Doença de Parkinson é um dos transtornos neurológicos mais comuns na atualidade e ainda não possui uma cura definitiva. No entanto, quando diagnosticada nos estágios iniciais da doença, é possível, através de tratamento farmacológico, retardar os efeitos da doença no corpo do paciente.

Existe hoje uma gama de procedimentos diagnósticos, além de testes que identificam alguns dos sintomas de forma desconexa, porém não existem ferramentas que correlacionem esses métodos.

Em uma parceria entre a UFRGS, representada pelo estudante Antônio Airoldi e o professor Mário Sobczyk, e a UFCSPA, representada pelo professor Alexandre do Pinho, este trabalho foi idealizado para resolver um problema pré-existente nos estudos sobre o equilíbrio, do laboratório de Fisioterapia da UFCSPA, onde as aquisições dos dados de sensores biomédicos eram realizadas de forma não síncrona, dificultando a análise dos resultados pelos pesquisadores.

1.1 Objetivos

O objetivo principal deste trabalho consiste no desenvolvimento de um sistema de aquisição, um caso particular de sistema *SCADA* que possa sincronizar dados, obtidos de dispositivos biomédicos, com o objetivo de correlacionar dados e permitir um diagnóstico simplificado e barato em vias dos mecanismos existentes hoje no mercado.

Além disso, com o desenvolvimento da ferramenta, novos avanços no estudo do equilíbrio poderão ser realizados, possibilitando desta forma, um melhor entendimento dos mecanismos de atuação da Doença de Parkinson no corpo humano.

Os objetivos específicos do trabalho são os seguintes:

- Integrar múltiplos sensores que permitam uma análise aprofundada do equilíbrio;
- Desenvolver uma aplicação simples, no entanto completa, que atenda às necessidades do laboratório de Fisioterapia da UFCSPA;
- Gerar relatórios completos das medições, bem como permitir uma análise coesa dos dados sincronizados;
- Disponibilizar um método alternativo para o estudo do controle postural de indivíduos com doença de Parkinson

Ao longo deste trabalho, serão apresentados, resumidamente, aspectos relacionados à Doença de Parkinson, bem como alguns conceitos-chave para facilitar o entendimento da solução escolhida para o desenvolvimento da aplicação.

Também será descrito o funcionamento da aplicação, tanto na forma de algoritmo de execução, quanto na utilização pelo usuário final.

Por fim, serão apresentados os resultados obtidos experimentalmente, comparando os sinais com ferramentas comerciais existentes no mercado, e analisando a coerência desses.

1.2 Justificativa

O departamento de fisioterapia da Universidade Federal de Ciências da Saúde de Porto Alegre está realizando um estudo que analisa a correlação entre a DP e o equilíbrio corporal.

Este trabalho visa disponibilizar uma ferramenta para auxiliar neste estudo, sendo um método alternativo para o estudo do controle postural de indivíduos com doença de Parkinson, permitindo diagnosticar a doença de forma antecipada e com baixo custo, ampliando o acesso à triagem, além de promover a pesquisa sobre a doença que, apesar de popularmente conhecida, ainda não são claros os mecanismos de sua atuação.

2 REVISÃO DA LITERATURA

Este capítulo tem por objetivo contextualizar o leitor a respeito da Doença de Parkinson, bem como os métodos existentes na atualidade para o diagnóstico da enfermidade.

2.1 Doença de Parkinson

A Doença de Parkinson (DP) é uma condição neurológica progressiva que causa problemas no cérebro e apresenta agravamento do quadro com o passar dos anos. Estima-se que em torno de 930 mil pessoas sofrem da doença nos Estados Unidos (MARRAS et al., 2018), e mais de 10 milhões de casos diagnosticados pelo mundo inteiro (FOUNDATION, 2020).

A doença é causada pela disfunção de células cerebrais, as quais param de produzir dopamina - um hormônio e neurotransmissor conhecido como o hormônio do prazer - responsável por diversas funções cerebrais e corporais, como o controle motor, redução da produção de insulina, entre outras funções primordiais. Os sintomas têm início quando o cérebro para de produzir dopamina em quantidades suficientes para permitir o controle adequado dos movimentos. As manifestações mais comuns da doença são as seguintes:

- Tremores
- Lentidão dos movimentos
- Perda de equilíbrio
- *Freezing*
- Distonia
- Cãimbras

2.1.1 Parkinsonismo

O Parkinsonismo é uma síndrome que é caracterizada principalmente por tremores, rigidez dos músculos e articulações, hipocinesia e lentidão nos movimentos (UK, 2019). Apesar de ser característico da DP, denomidado Parkinsonismo Idiopático, o Parkinsonismo pode ter outras causas:

- Parkinsonismo Vascular: Também conhecido como Parkinsonismo Arteriosclerótico, afeta pessoas com fluxo sanguíneo para o cérebro restrito, normalmente ocorrendo em casos de pacientes que sofreram derrames leves.

- Parkinsonismo induzido por fármacos: Medicamentos neurolépticos, utilizados para tratamento de esquizofrenia e transtornos psicóticos, bloqueiam a ação da dopamina no cérebro. É uma manifestação rara de reação medicamentosa e reversível após a interrupção da utilização do medicamento.

2.1.2 Diagnóstico da doença

Uma vez que existem diversas manifestações de Parkinsonismo e os sintomas entre elas são muito semelhantes, o diagnóstico para a DP não é simples. Além disso, na maioria dos casos, os sintomas são desenvolvidos gradualmente e de formas diversas para cada tipo de pessoa.

Segundo (GRIMES et al., 2019), a forma mais comum para o diagnóstico é através da análise da reação do paciente ao tratamento com remédios como Levodopa. Após a utilização deste medicamento, quando a administração é diminuída ou interrompida, é esperado um retorno dos sintomas para casos positivos de Parkinsonismo Idiopático.

Além do diagnóstico farmacológico, alguns tipos de exames podem ser executados para análise dos sintomas:

- Ressonância magnética do cérebro;
- Eletromiografica muscular;
- Análise química do transportador de Dopamina;
- Teste de funções autonômicas.

Nenhum destes testes, de forma isolada, consegue diagnosticar a DP de forma definitiva. Esses testes são utilizados de forma combinada com históricos médicos e resultados de examinação médica, para a elaboração de um diagnóstico mais preciso.

2.1.3 Causas

A DP é uma doença ocasionada por diversos fatores, não havendo ainda um consenso sobre qual ou quais desses são decisivos para o desenvolvimento da doença (UK, 2020b). Cientistas do mundo todo acreditam que uma combinação de fatores é a causa definitiva dessa enfermidade.

2.1.3.1 Deficiência de Dopamina

O fator definitivo para o aparecimento dos sintomas da doença é a perda de neurônios dopaminérgicos na área do cérebro denominada *substantia nigra*. Isto faz com que a produção de Dopamina não atenda às necessidades do corpo, acarretando o aparecimento dos sintomas da doença. Uma vez que a enfermidade é progressiva, com o passar do tempo, os sintomas tendem a se agravar, além do surgimento de novos sintomas.

Em estágios avançados da doença, o impacto na atividade motora pode acarretar complicações sistêmicas, causando quedas ou desenvolvendo pneumonia pela inabilidade de controlar a respiração de forma adequada, aspirando alimentos ou líquidos e dificultando a expulsão do material pela tosse.

2.1.3.2 Fatores Genéticos

De acordo com os estudos de (UK, 2020a), cerca de 10% a 15% dos casos de DP estão relacionados com a predisposição genética, e mesmo nos casos onde há esta predisposição, não é uma garantia de que o paciente desenvolverá a doença.

2.1.3.3 Fatores Ambientais

Existem evidências de que algumas toxinas, com destaque para o uso de pesticidas e herbicidas, estejam relacionados com a perda de neurônios dopaminérgicos. No entanto, não existem afirmações conclusivas que confirmem esta correlação.

2.2 Correlação entre a propriocepção e o equilíbrio com a DP

Em (ARTIGAS et al., 2016), foi realizado um estudo que confirmou que mudanças na propriocepção podem contribuir para instabilidade postural, influenciando no equilíbrio corporal. Foram analisados um total de 40 pacientes, sendo 20 destes com diagnóstico positivo para DP e 20 deles do grupo de controle, de ambos os sexos e com idade acima de 45 anos.

Os testes fizeram uso de um dinamômetro isocinético (Biodex Multi-Joint System 4 Pro), para analisar a propriocepção angular dos joelhos e um baropodômetro (FootWork IST/AM3 Intermetique), para verificar as variações no centro de pressão e analisar o equilíbrio estático corporal.

Neste estudo foram avaliados pacientes em diversos estágios da DP, comparando com um grupo de controle onde foi possível identificar que a doença, principalmente nos estágios mais avançados, influencia diretamente na propriocepção, na habilidade cognitiva e nos sintomas motores. Além de estar presente nas *Guidelines* sobre a DP, é possível verificar que são características conhecida, o atraso no tempo de reação nervosa e a deficiência na sensibilidade no controle fino do equilíbrio estático.

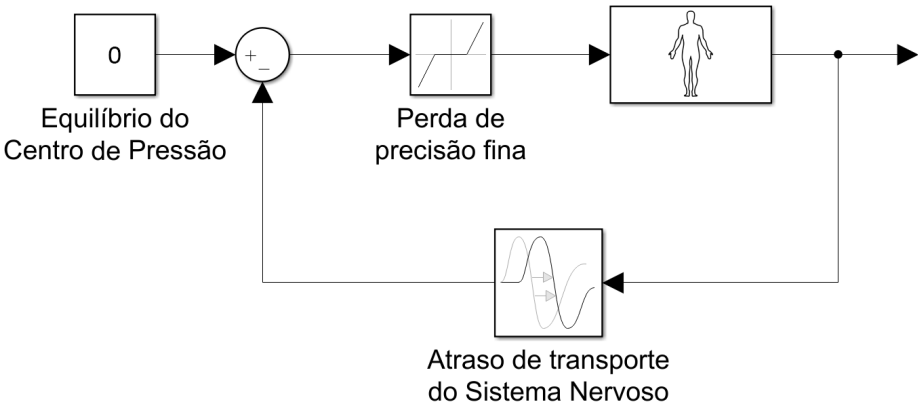

Figura 1: *Representação esquemática dos sintomas motores da DP em Diagrama de Blocos. Fonte: Autor*

Com o objetivo de contextualizar ao leitor a representação matemática dos sintomas motores da DP, foi elaborado o diagrama de blocos presente na Figura 1. Um exemplo palpável para o entendimento, é a análise da caminhada.

A caminhada pode ser interpretada como um pêndulo invertido (MORASSO et al., 2019), onde o corpo é inclinado para a direção do deslocamento e as pernas, como elemento atuador, busca reequilibrar o corpo, reposicionando o centro de gravidade a um ponto de equilíbrio. A DP insere uma série de dificuldades para o controle motor do paciente, uma vez que existe um atraso para a reação ao desequilíbrio provocado pelo movimento, além de apresentar perda do controle fino do equilíbrio, representada no diagrama pela zona-morta, ambos sintomas relacionados com a deficiência de dopamina.

2.2.1 Instrumentação clínica para o diagnóstico de DP

Conforme apresentado nas seções 2.1.2 e 2.2, uma série de abordagens podem ser utilizadas para diagnosticar sintomas relacionados ao Parkinsonismo. Esta seção elucidará de forma sucinta o princípio de funcionamento de alguns destes sensores e métodos de forma a contextualizar a abordagem para o desenvolvimento deste trabalho.

2.2.1.1 Ressonância Magnética do Cérebro

A ressonância magnética é uma das formas mais eficientes na detecção de anormalidades no cérebro do paciente acometido por DP. Segundo (BELL et al., 2020), é possível perceber através deste tipo de exame uma perda sensível do volume cerebral.

O diagnóstico de DP pode ser realizado sem a utilização de Ressonância Magnética, no entanto, é através deste tipo de exame que podem ser detectados diversos aspectos da doença, bem como avaliar a evolução desta no paciente. São exemplos de avaliações que podem ser realizadas por este método:

- Avaliar a perda de tecido e a atrofia do cérebro;
- Detectar mudanças na região do cérebro denominada Gânglio Basal, associada com diversas funções motoras, cognitivas e de aprendizado;
- Detectar depósitos anormais de ferro que estão relacionadas diretamente com a *substantia nigra*;
- Excluir casos reversíveis de Parkinsonismo, descartando a possibilidade de DP.

2.2.1.2 Eletromiografia de Superfície

Eletromiografia de superfície, ou EMG, é uma ferramenta que mede os impulsos elétricos oriundos da ativação do músculo esquelético. Este equipamento pode ser utilizado para mensurar a ativação involuntária dos músculos, quando o paciente é acometido por tremores. Através do padrão da onda obtida, é possível verificar através de análises frequenciais, o diagnóstico de Parkinsonismo. Além disso, é possível avaliar também a eficácia de medicamentos como o Levodopa, além de poder ser utilizado para realizar um ajuste fino na dosagem deste medicamento, uma vez que pela sua forma de atuação, este medicamento pode trazer efeitos indesejados como a contração demasiada dos músculos, causando cãimbras e dores musculares.

Esta técnica é utilizada massivamente nos estudos de controle postural (BERG; STRANG, 2012), em conjunto com metodologias experimentais, técnicas de processamento e computação estatística, permitem além do estudo de reações à perturbações externas, analisar objetivos específicos de estudos científicos, como é o caso deste trabalho.

O princípio de funcionamento de um dispositivo de EMG é basicamente a amplificação do sinal elétrico nervoso de forma diferencial com um ganho elevado, amplificando e condicionando o sinal de forma a permitir a visualização do impulso nervoso atuando na contração muscular. Este mecanismo é ilustrado na Figura 2, onde se representa um circuito esquemático simplificado do funcionamento de um eletromiógrafo. m_1 e m_2 representam as tensões em cada ponto de aplicação dos eletrodos, geralmente em μV, enquanto n representa o ruído. Assumindo que o ruído é igual nos dois pontos de medição, é possível anular a influência do ruído no sinal. Na prática, é sabido que é impossível anular completamente estas perturbações e como, tanto o sinal, quanto o ruído são da mesma ordem de grandeza, ao amplificar o sinal, amplificamos também o ruído.

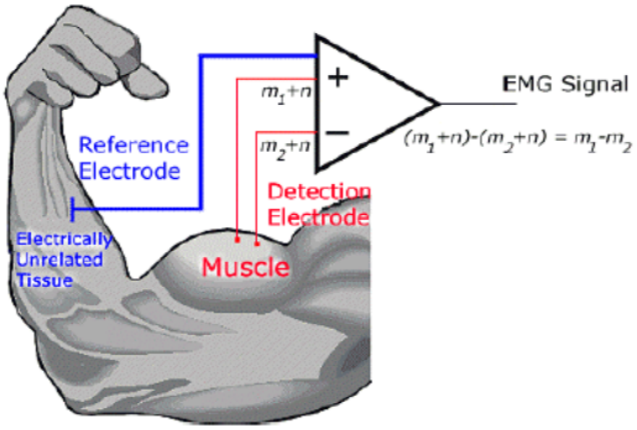

Figura 2: *Diagrama esquemático de um dispositivo de EMG Fonte:* (LUCA, *2002)*

2.2.1.3 Análise química do transportador de Dopamina

Também denomidado DaTScan, esse método de análise é uma espécie de tomografia, onde um rastreador radioativo denominado Ioflupano é injetado na corrente sanguínea. O composto se atrela aos transportadores de dopamina e, após algumas horas, acumula-se nos neurônios dopaminérgicos do cérebro do paciente.

Com a utilização de equipamentos não invasivos de detecção de radiação, é possível ter uma imagem do cérebro do paciente, onde as regiões ricas em dopamina são ressaltadas. Na Figura 3, é possível ver um comparativo entre um exame normal e um exame que apresenta uma anomalia nos níveis de dopamina.

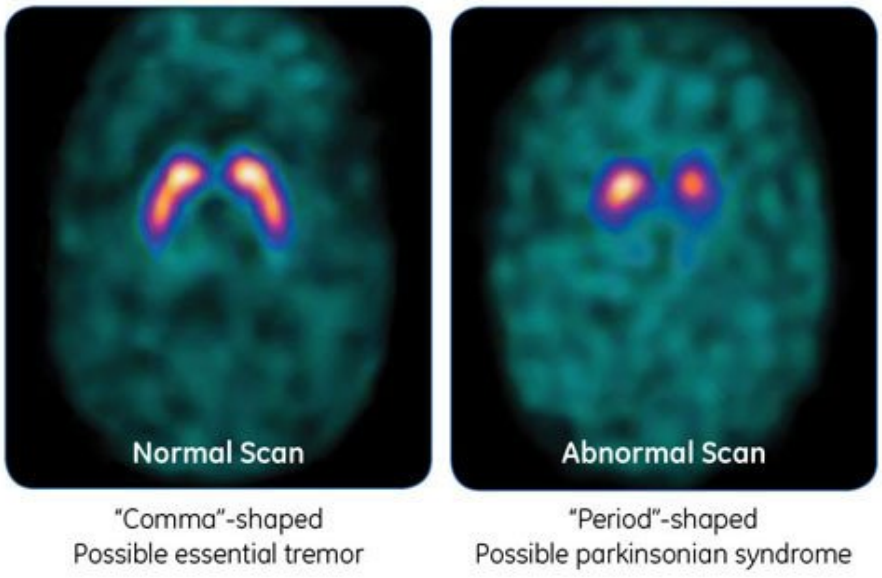

Figura 3: *Comparativo entre exames DaTScan normal (Esquerda) e anômalo (direita). Fonte:* (ASSOCIATION, *2019)*

2.2.1.4 Dinamômetro Isocinético

O dinamômetro isocinético é um equipamento de medição de posição angular utilizado para avaliar com precisão a propriocepção corporal de membros inferiores. Esse equipamento é composto por uma cadeira acoplada a um dispositivo que mensura a posição angular, podendo também avaliar grandezas físicas como força, trabalho, potência, tempo de reação e a resistência de grupos musculares.

Alguns estudos, como o que foi apresentado em 2.2, correlacionam a deficiência do controle e do senso de posicionamento articular.

2.2.1.5 Baropodômetro

O baropodômetro é um tipo de sensor utilizado para identificar pontos de pressão dos pés, podendo assim analisar a distribuição de peso do corpo, permitindo a análise do controle postural.

Esse equipamento é largamente utilizado no estudo da pisada ao realizar tarefas como caminhar, mas também pode ser utilizado para outros tipos de análise (ARKIPELAGO, 2020), como por exemplo:

- Postura do paciente em equilíbrio estático;
- Disfunções funcionais de equilíbrio e estabilidade;
- Divisão das cargas corporais em condições ortostáticas;
- Detecção de alterações biomecânicas do corpo;
- Identificação de assimetrias de comprimento dos membros inferiores.

2.2.1.6 IMU

Uma Unidade de Medida Inercial ou *IMU - Inertial Measurement Unit* é um dispositivo que é composto por sensores capazes de detectar grandezas dinâmicas como posição, velocidade e aceleração. Este conjunto de sensores são unidos, pois possuem limitações de atuação em alguns cenários, e a utilização da fusão de sensores minimiza o impacto dos erros de medição.

Em geral, *IMU's* são compostas por um acelerômetro e um giroscópio, podendo também contar com a presença de um magnetômetro. Cada sensor mensura uma grandeza diferente. Acelerômetros medem a aceleração absoluta em um determinado corpo. Giroscópios são sensíveis a perturbações na orientação. Magnetômetros avaliam a posição angular em relação ao eixo magnético terrestre (PAULA, 2015).

Estes sensores são largamente utilizados em *Drones*, com o objetivo de identificar a posição e a orientação do veículo. Nas aplicações para a área da saúde, são utilizados para mensurar o comportamento dinâmico do corpo humano em situações estáticas e dinâmicas. Com o auxílio destes sensores, é possível avaliar a orientação do corpo ou de seus membros, de forma a adicionar novas variáveis aos estudos de movimento e equilíbrio.

Unidades Inerciais também são largamente utilizadas na área da saúde, tendo resultados muito promissores no estudo do equilíbrio. Em (NOUREDANESH; TUNG, 2019), foi identificado que IMU's tem uma precisão na identificação de ajustes posturais compensatórios de 99,33%, enquanto utilizando eletromiografia de superfície foi possível alcançar a precisão de 90,5%.

Acelerômetros de dispositivos móveis são construtivamente inspirados em um sistema massa-mola. Na Figura 4 temos uma representação gráfica de como este sensor consegue mensurar a aceleração em um eixo. O dispositivo baseia-se em uma massa que pode oscilar livremente. Esta massa é fixada em pontos de ancoragem do dispositivo, que funcionam como molas, fixando a massa porém permitindo o seu deslocamento. O deslocamento é medido indiretamente pela variação da capacitância com as hastes de controle. Utilizando a Lei de Hooke, presente na equação 1, pode-se isolar o termo de aceleração, calculando-o através do deslocamento que é função da capacitância, conforme a equação 2.

$$F = kx \tag{1}$$

$$a = \frac{k}{m}x(C) \tag{2}$$

Nas equações apresentadas, F é a força em N, k é a constante de mola em N/m, x é o deslocamento axial em m, m é a massa oscilante em kg e a é a aceleração da massa, em $m/2^2$.

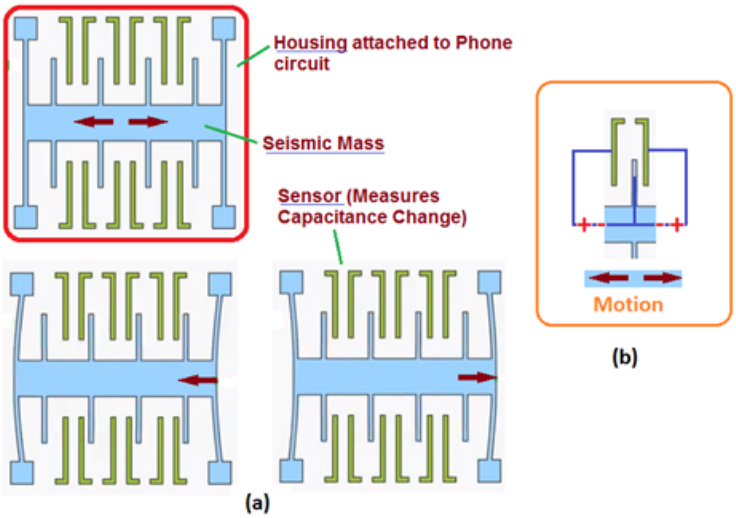

Figura 4: *Aspectos construtivos de um acelerômetro utilizado em dispositivos móveis. Fonte:* (GUJARATI, *2020)*

Já giroscópios utilizados em IMU's baseiam-se no princípio de coriolis. Este princípio define que uma perturbação no movimento rotacional em torno de um eixo promove uma força em um outro eixo. Assim como no caso do acelerômetro, o giroscópio possui uma massa suspensa por molas. Esta massa possui hastes que ficam vibrando continuamente sob a excitação de dispositivos piezoelétricos, conforme podemos ver na Figura 5. Quando há um deslocamento rotacional incidindo sobre a massa, surge uma reação vertical nas hastes que podem ser medidas pelos braços sensoriais. Se acordo com a intensidade da deformação destes é possível mensurar variações na orientação da massa.

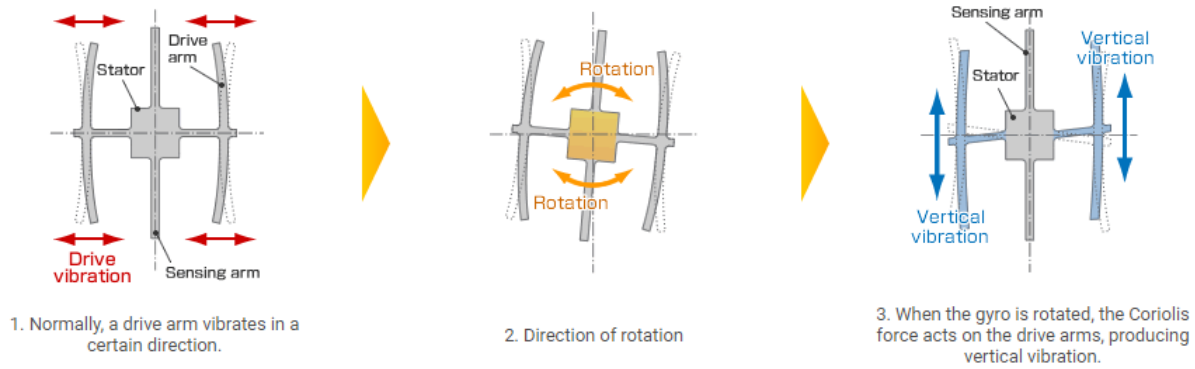

Figura 5: *Aspectos construtivos de um giroscópio utilizado em dispositivos móveis. Fonte:* (ST. LOUIS, *2018)*

2.3 *SCADA*

SCADA, ou Sistema Supervisório de Controle e Aquisição de Dados, é um acrônimo que vem do termo inglês *Supervisory control and data acquisition*. Esta tecnologia utiliza recursos de hardware e software que permitem o controle local ou remoto de processos, a aquisição de dados em tempo real, a interação com um ou mais sensores e atuadores, além de oferecer uma interface amigável interativa ao usuário denominada IHM (Interface Homem-Máquina).

Sistemas SCADA são utilizados nas mais diversas aplicações de engenharia, pois possibilitam um acesso ao controle de equipamentos e dados de sensoriamento de forma simplificada e reunidas em um único dispositivo. São alguns exemplos de casos reais de sistemas SCADA:

- Monitoramento do status de funcionamento de servidores;
- *Dashboard* de estatísticas de linha de produção;
- Salas de controle de geração de energia;
- Controle de temperatura e umidade em câmaras climáticas.

Na Figura 6 podemos verificar um exemplo de sistema supervisório implementado na sala de controle central da usina hidrelétrica Itaipu Binacional.

O conceito de um sistema SCADA é muito abrangente e pode ser utilizado para denominar diversas formas de aplicações, conforme citado anteriormente. O caso em estudo deste trabalho consiste em uma situação específica onde somente é realizada a aquisição de dados, sendo denominado Sistema de Aquisição ou de Monitoramento.

Figura 6: *Sala de controle central da usina hidrelétrica Itaipu Binacional. Fonte: (*CBIE*, 2019)*

3 DESENVOLVIMENTO

Este capítulo tem por objetivo apresentar a aplicação que foi desenvolvida ao longo deste trabalho. Na seção 3.1, são apresentados os sensores utilizados no estudo, enquanto na 3.2, discutem-se os algoritmos em alto nível que são executados pelo software. Por fim, a seção 3.3 é dedicada à interface gráfica do sistema de aquisição, sendo descritos todos os recursos inseridos na ferramenta para a análise do operador.

A ferramenta funciona como uma ferramenta de integração de sensores, apresentando os dados adquiridos em tela de forma a facilitar a análise pelo operador. Ela estabelece a troca de informações com 2 sensores independentes, utilizando os protocolos de comunicação adequados. Na Figura 7 temos uma representação esquemática que correlaciona os sinais de cada equipamento e sua forma de comunicação com a aplicação.

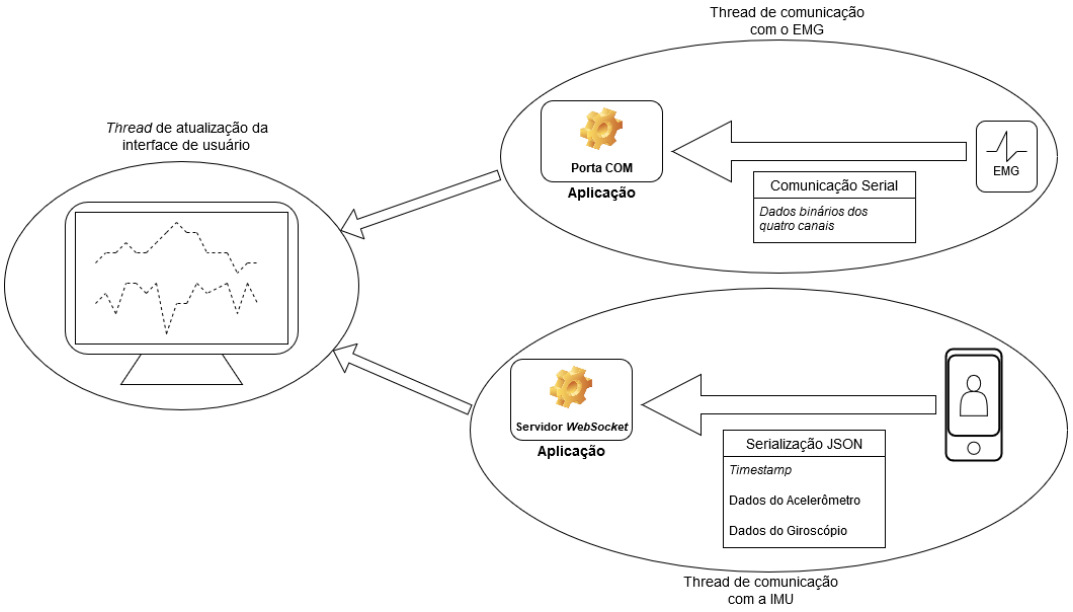

Figura 7: *Diagrama de funcionamento da aplicação. Fonte: Autor*

Os sensores utilizados foram os seguintes: Um eletromiógrafo de superfície e uma unidade de medida inercial. Estes dispositivos foram selecionados para capturar grandezas relacionadas à atividade muscular dos membros, sendo possível verificar o atraso temporal entre o estímulo elétrico do músculo e a sua resposta no movimento.

O EMG comunica-se via protocolo serial pois é a única forma disponível de comunicação com este dispositivo comercial. Já o IMU tomou-se a decisão de utilizar os sensores de um telefone celular, para permitir uma aplicação mais maleável e sem fios.

3.1 Sensores utilizados

O objetivo original deste trabalho era integrar dois equipamentos comerciais do laboratório de fisioterapia da UFCSPA: um EMG, para adquirir dados da atividade elétrica dos músculos e um baropodômetro, para capturar informações relativas ao equilíbrio corporal.

3.1.1 Atividade Muscular

A UFCSPA possui um dispositivo de EMG denominado Miotool 400. O equipamento da empresa Miotec possui 4 canais de aquisição independentes, além de um canal de referência.

Figura 8: *EMG Miotool 400 Fonte: (*MENEZES ARRIAL*, s.d.)*

Apesar de ser um equipamento comercial e com protocolos de comunicação proprietários, contamos com o auxílio do setor de pesquisa e desenvolvimento da empresa para viabilizar a comunicação do dispositivo com o sistema de aquisição e, desta forma, a integração foi possível de ser realizada.

3.1.2 Equilíbrio Corporal

A proposta inicial para este trabalho consistia na utilização de um baropodômetro comercial de propriedade da UFCSPA. O equipamento permite uma avaliação do equilíbrio através da distribuição de pressão dos pés do paciente em uma plataforma. Em decorrência do isolamento social, não foi possível ter acesso ao equipamento, pois a retirada, bem como o acesso ao laboratório em que este encontra-se alocado, não foram autorizados.

Como alternativa a este aparato, foi proposta a utilização de uma *IMU*, pois através de sensores inerciais seria possível também avaliar o equilíbrio corporal através de outras inferências.

Optou-se pela utilização de sensores de telefones celulares, uma estratégia que está tornando uma tendência na área médica (ROSARIO et al., 2015), principalmente em razão da praticidade e da logística de equipamentos, pois a comunicação com esses dispositivos pode ser feita sem fios, o que permitiria uma execução de testes de forma facilitada.

Uma vez que este sensoriamento depende do modelo de telefone utilizado para a análise, não serão dados detalhes de configurações de *hardware* para a *IMU*.

3.2 Algoritmo de execução

O sistema de aquisição proposto realiza a integração de 3 sensores de forma sincronizada, para a análise do equilíbrio em pacientes com DP:

- Um dispositivo de EMG, que tem por objetivo capturar a atividade elétrica de ativação dos músculos;
- Um acelerômetro e um giroscópio, que têm como função detectar o movimento do paciente.

Uma vez que estes sensores são sincronizados, é possível realizar uma análise correlacionada desses fenômenos pelo operador durante a execução do programa, ou fazendo uso dos dados exportados pela aplicação.

O programa foi desenvolvido na linguagem *C#*, suportada pelo *.NET Framework* que é a plataforma de desenvolvimento da *Microsoft* para aplicações principalmente em ambiente *Windows*.

O dispositivo de EMG comunica-se com o computador por comunicação Serial, enquanto o acelerômetro e o giroscópio, proveninentes de sensores de um telefone celular, comunicam-se via protocolo TCP/IP.

O programa desenvolvido possui basicamente 3 subdivisões de rotinas, sendo executadas por 3 *Threads* independentes: a rotina de aquisição do EMG, a rotina de aquisição dos sensores do telefone celular e a rotina de atualização de formulário e interação com o usuário. Esta divisão em 3 *Threads* é necessária para que cada rotina possa ser executada de forma independente, sem interferir na outra. Desta forma, por exemplo, impedimos que o acionar de um botão possa interferir no intervalo de aquisição de um dos equipamentos.

A aplicação armazena os dados adquiridos de forma ininterrupta, com taxas de aquisição independentes (2kHz para o EMG e uma taxa fixa definida no telefone celular, que varia de 5 Hz a 50 Hz) devido às particularidades dos equipamentos e apresenta os dados em tela com uma taxa de atualização de 10 Hz. Além disso, a aplicação permite criar sub-conjuntos de dados para a exportação, com o objetivo de facilitar a análise posterior por parte do operador.

3.2.1 Rotina de aquisição do EMG

Esta *Thread* é responsável por gerir a comunicação do sistema de aquisição com o dispositivo de EMG. É apresentado na Figura 9 um fluxograma que explica o funcionamento da *Thread* do EMG.

A rotina de execução inicia estabelecendo uma comunicação Serial com o dispositivo de EMG. A seguir, o *buffer* da comunicação é limpo e, por fim, utilizando um protocolo de comunicação proprietário, que é apresentado nos anexos deste trabalho, é configurado o modo de aquisição para envio contínuo de dados do EMG para o computador.

Uma vez realizadas estas configurações, o programa entra em um laço de duração infinita, onde o computador continuamente lê os dados disponíveis na porta *Serial*, enviados pelo EMG, sendo estes amostrados à uma taxa de 2 kHz.

Uma vez que o foco deste sistema de aquisição é permitir a correlação de fenômenos mensuráveis por mais de um dispositivo, é necessária uma sincronização dos dados adquiridos. Desta forma, essa rotina é executada em uma *Thread* independente para a aquisição dos dados juntamente com o instante de tempo em que a amostra foi feita.

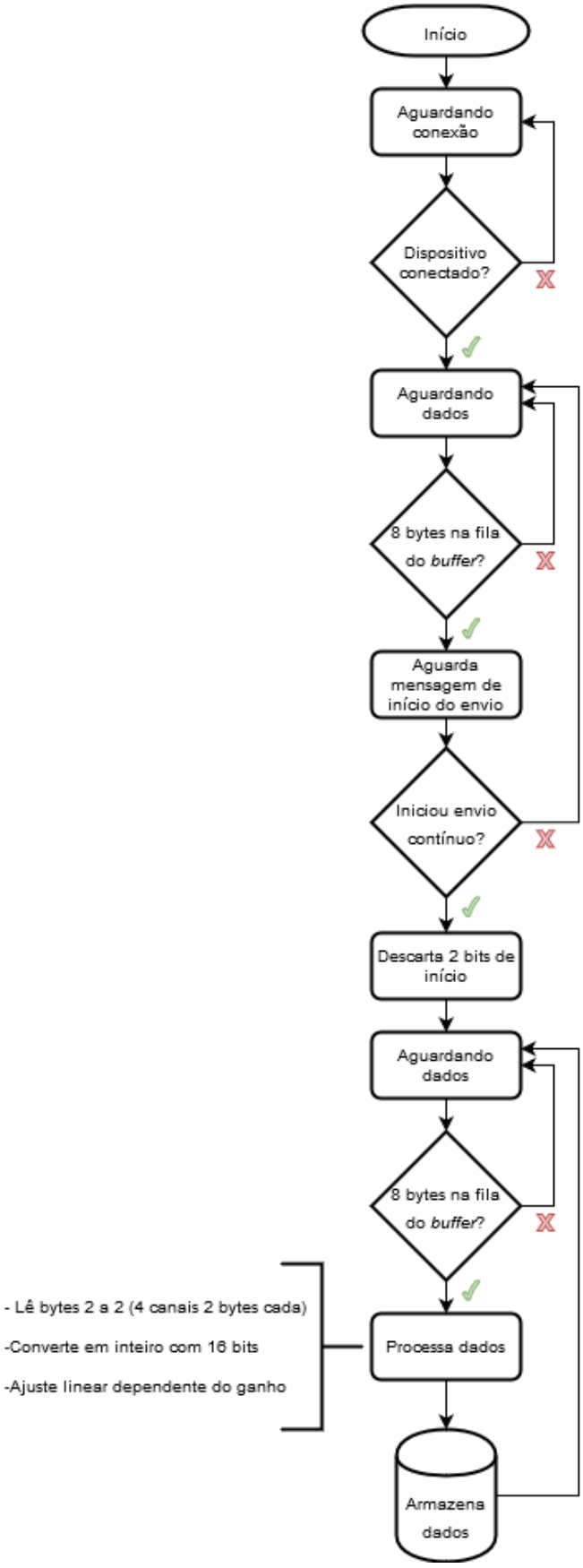

Figura 9: *Fluxograma da Thread de aquisição do EMG. Fonte: Autor*

3.2.2 Rotina de aquisição dos sensores do telefone celular

Ao inicializar esta rotina, um servidor *WebSocket* é estabelecido na máquina que executa o sistema de aquisição. É através deste servidor que a aplicação consegue receber os dados dos sensores do telefone celular. O acesso pelo dispositivo cliente é realizado pelo IP da máquina na rede local na porta 80. Pode-se verificar na Figura 10 um fluxograma que explica o funcionamento da *Thread* de aquisição de dados da IMU.

Figura 10: *Fluxograma da Thread de aquisição da IMU. Fonte: Autor*

Para o envio das informações dos sensores, foi utilizado um aplicativo pré-existente e de código aberto denominado *SensorStreamer* (JAN MRÁZEK, 2020). Este aplicativo desenvolvido para a plataforma Android, basicamente, adquire dados de alguns sensores do telefone celular de forma configurável e envia com uma latência convigurável, conforme é apresentado na Figura 11, via modelo de transmissão JSON.

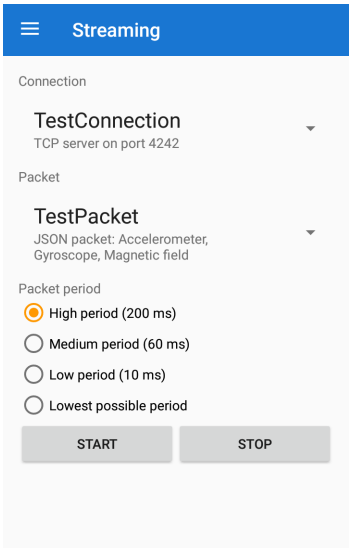

Figura 11: *Aplicativo Open-Source SensorStreamer Fonte: (*JAN MRÁZEK, *2020)*

O protocolo JSON resumidamente consiste na estruturação de um conjunto dados em objetos para troca destes entre múltiplos sistemas. Estes objetos possuem atributos, que por sua vez possuem valores. A serialização das informações no formato JSON define algumas regras simples para estruturar dados, permitindo uma troca de dados organizada entre múltiplas linguagens de programação, conforme (INTERNATIONAL, 2017).

Como se trata de um dado que necessita ser sincronizado com o EMG, a aquisição é realizada novamente em uma *Thread* independente em um laço de execução infinito e armazenando juntamente com os dados, o instante de tempo em que a amostra ocorre.

3.2.3 Rotina de Interface de Usuário

Esta rotina é responsável pela responsividade da interface gráfica da aplicação, diante dos comandos do usuário.

Esta rotina administra a execução de timers de atualização dos gráficos, permite a interação do usuário com os gráficos durante a execução, além de possibilitar ações como a criação de aquisições para posterior exportação dos dados no armazenamento do computador.

3.3 Sistema de Aquisição

Esta seção tem por objetivo apresentar a interface gráfica do programa bem como descrever as formas com que o usuário pode interagir com a aplicação.

Figura 12: *Interface gráfica do Sistema de aquisição Fonte: Autor*

Na Figura 12 é apresentada a interface com que o usuário consegue interagir com o programa de aquisição. No título da janela do programa é apresentado, com a precisão de 100 ms, a data e hora de execução do programa, com o objetivo de auxiliar o usuário a documentar os testes realizados.

No canto superior direito são informados os parâmetros de conexão para os dispositivos.

Para a conexão com a IMU somente é informado o endereço de IP do computador que executa a aplicação, para ser realizada a configuração no telefone celular cliente.

Para a conexão com o EMG, é necessário informar a Porta Serial em que o dispositivo encontra-se conectado. A informação de Ganho é relacionada a sensibilidade necessária do dispositivo, para adquirir o sinal elétrico. Esta informação é diretamente relacionada com o grupo muscular onde são aplicados os eletrodos, portanto é uma informação de conhecimento do usuário final. Por fim, é possível selecionar quais canais do eletromiógrafo serão apresentados no gráfico do dispositivo.

Ao centro do painel lateral direito, possuímos duas ferramentas utilizadas para a exportação dos dados da seção, para posterior análise: Intervalo de Aquisição e Destacar Dados. Os intervalos de aquisição são formas de criar sub-conjuntos de dados e salvá-los de forma externa em um arquivo de Excel. Esta necessidade foi levantada em decorrência de que os ensaios que são realizados com o paciente duram algumas dezenas de segundos, podendo, no máximo, alcançar alguns minutos de duração. Desta forma, foram incluídos intervalos de aquisição pré-definidos, além de permitir que o usuário possa iniciar a

gravação dos dados por tempo indefinido, controlando a aquisição manualmente.

Este processo de aquisição é executado simultaneamente com as *Threads* de aquisição, porém geram pacotes de dados menores, que facilitam a análise posterior.

A ferramenta de destacar os dados tem como objetivo marcar *ranges* dados que possuem um comportamento identificado pelo operador e em que há o interesse de análise posterior.

Ainda no painel de operação lateral, podemos definir o diretório de exportação dos dados adquiridos, onde serão salvos de forma independente cada sensor com data e hora, intervalo de aquisição em ms e os dados brutos de cada sensor.

Também é possivel ajustar a escala horizontal dos gráficos com o número de pontos que se deseja apresentar em cada gráfico, de forma que a janela de verificação imediata seja mais amigável ao usuário.

Por fim, temos a Janela de verificação imediata, que possui os gráficos de cada um dos sensores. Nesta janela são mostrados os últimos N dados definidos na Escala Horizontal. Para os sensores da IMU, são apresentados os dados para os 3 eixos e, para facilitar a visualização do usuário, há um ajuste automático da escala vertical baseado nos pontos que estão apresentados na janela do gráfico. Para o EMG, são apresentados somente os canais que são selecionados pelo usuário no painel de operação lateral, porém, para este gráfico, a escala vertical é fixada no intervalo de -1000 a 1000 μV, pois os sinais de interesse encontram-se nesta faixa de valores e a escala fixada facilita a visualização pelo usuário.

4 RESULTADOS

O presente capítulo discorre sobre os resultados obtidos durante a execução do trabalho. Na seção 4.1 são apresentados os dados adquiridos em um ensaio com todos os equipamentos sincronizados, comparando os resultados com o sistema de aquisição proprietário do EMG, destacando a influência que o ruído tem na avaliação do sinal adquirido pelo dispositivo de EMG. Finalmente, na seção 4.3, são relacionados os testes que haviam sido planejados para a avaliação da ferramenta, mas que não puderam ser realizados em decorrência da Pandemia.

4.1 Testes de execução

Nesta seção serão apresentados dados obtidos através do sistema de aquisição em uma sessão de testes com o autor deste trabalho como paciente, de forma a demonstrar o funcionamento geral do sistema de aquisição.

O teste consistiu na aplicação de eletrodos sobre músculo Bíceps Braquial esquerdo, com o eletrodo de referência posicionado sobre osso Úmero esquerdo.

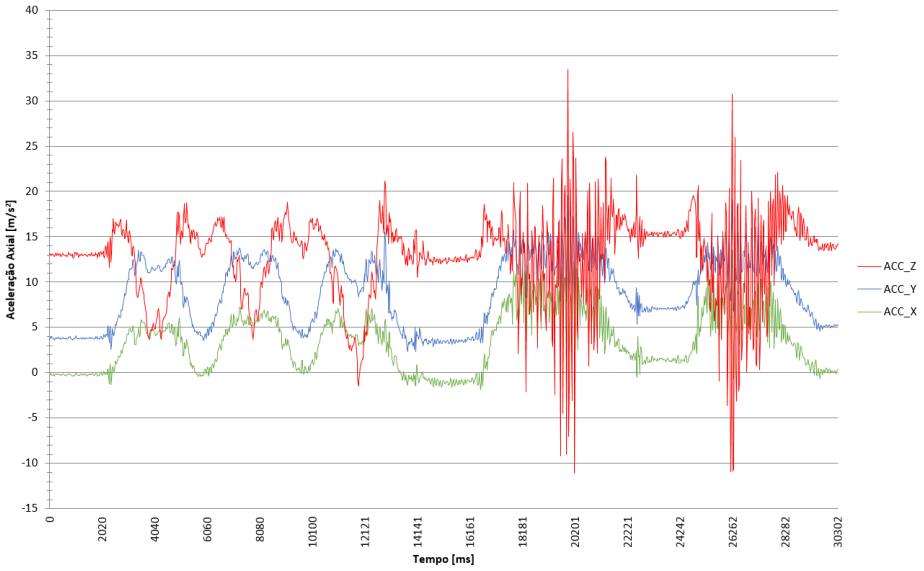

Figura 13: *Dados obtidos do ensaio com o acelerômetro Fonte: Autor*

Os procedimentos utilizados para os testes foram os seguintes:

1. Partindo o repouso, a realização de 3 movimentos de contração seguidos do relaxamento em movimentos de vai e vem do antebraço.

2. Em seguida, duas contrações intensas do músculo, aplicando uma força contrária com o braço direito de forma a aumentar a necessidade de tônus muscular, seguidas do repouso.

Nas Figuras 13, 14 e 15 são apresentados os resultados para o ensaio de testes com acelerômetro, giroscópio e EMG respectivamente.

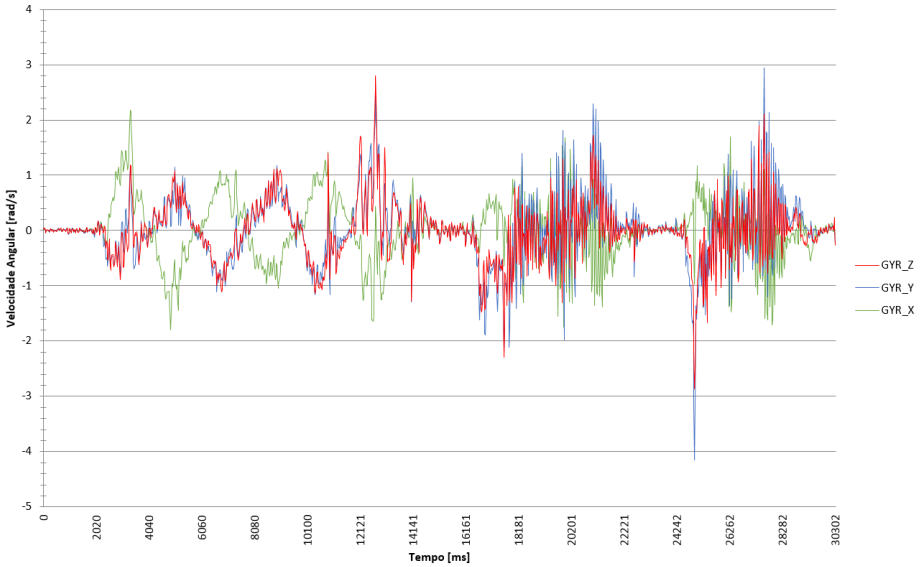

Figura 14: *Dados obtidos do ensaio com o giroscópio Fonte: Autor*

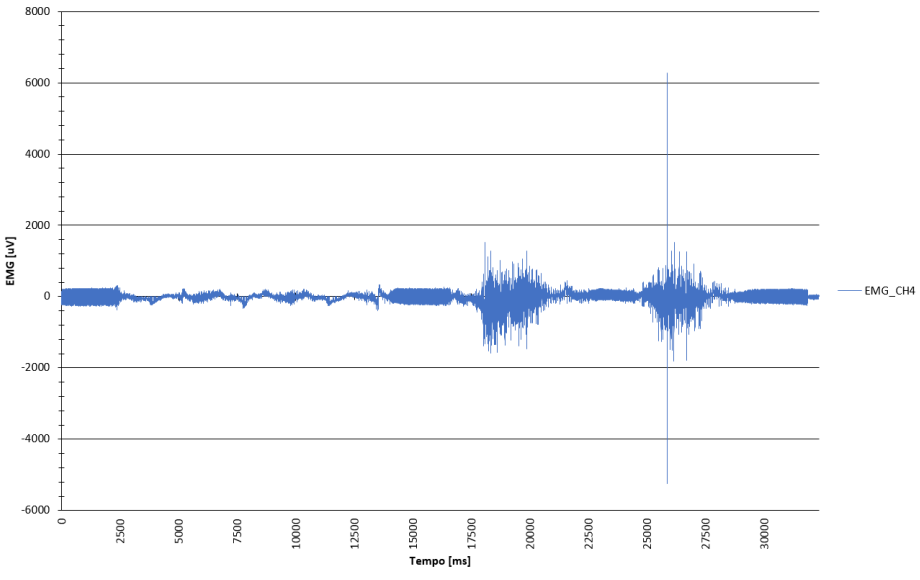

Figura 15: *Dados obtidos do ensaio com o EMG Fonte: Autor*

4.2 Ruído de medição

Na seção 2.2.1.2 é informado que um dispositivo de EMG, devido ao seu princípio de funcionamento amplifica também o ruído do ambiente.

Na Figura 15 podemos analisar um forte ruído de medição. Com o objetivo de comparar os resultados da aquisição com o teste realizado em 4.1, o mesmo procedimento foi executado adquirindo os dados com o software Miotec Suite, programa proprietário da empresa MIOTEC, fabricante do MIOTOOL 400.

Na Figura 16 podemos verificar os dados obtidos através do ensaio utilizando o software do fabricante.

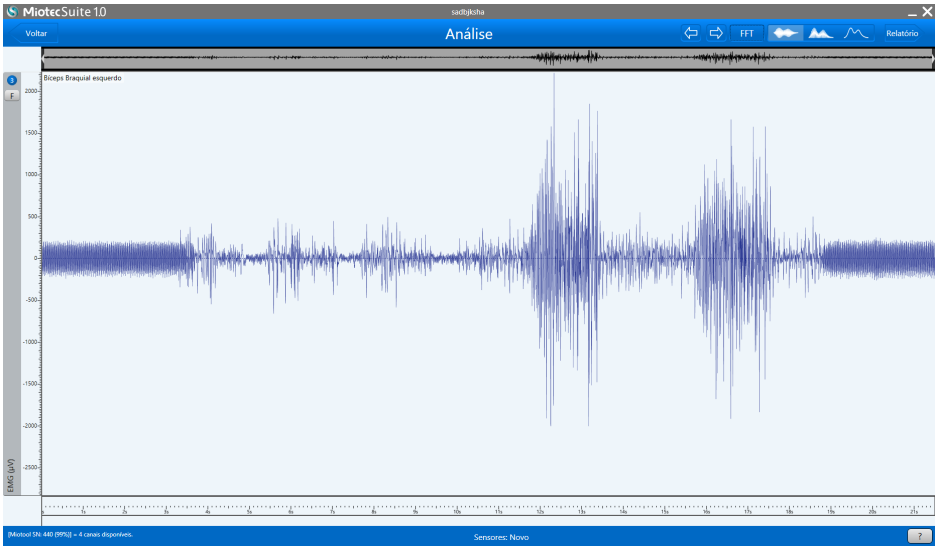

Figura 16: *Dados obtidos do ensaio com o EMG utilizando o software Miotec Suite Fonte: Autor*

Através de uma análise qualitativa, uma vez que não é possível adquirir dados do mesmo ensaio com os dois *softwares* simultaneamente devido à comunicação serial, podemos perceber que os sinais adquiridos pelo sistema de aquisição e pelo software do fabricante em muito se assemelham, e em ambos os casos um ruído considerável é percebido nas amostras.

A ferramenta Miotec Suite possui alguns recursos de análise do sinal que permitem um melhor entendimento dos dados amostrados. Na Figura 17 podemos verificar a FFT do sinal amostrado, onde podemos ver o pico de amplitude na frequência de 60 Hz e picos menores nas harmônicas desta frequência fundamental. A incidência deste ruído se dá provavelmente à fonte de alimentação do computador utilizado para a aquisição dos dados, que é conectada à rede elétrica.

A aplicação desenvolvida, assim como o Miotec Suite apresenta o dado adquirido em tela sem realizar qualquer filtragem no sinal, permitindo que o usuário tenha o sinal exato, podendo posteriormente tratar este sinal da forma que acreditar ser adequado. O Miotec Suite, no entanto, possibilita após a captura dos dados brutos, a aplicação de alguns filtros pré-definidos de forma a tratar este sinal.

Não fez parte do escopo do desenvolvimento deste trabalho a implementação de rotinas de filtragem, primeiramente por não se tratar de uma necessidade para o estudo que este dado fosse filtrado, pois para os testes propostos, somente é relevante saber o instante em que ocorre a ativação do músculo, e não a intensidade. Além disso, existem algumas informações relevantes que podem ser perdidas caso a filtragem fosse fixada, como os tempos de potencial de ação na unidade motora, conforme (CHRISTOU; NETO, 2010).

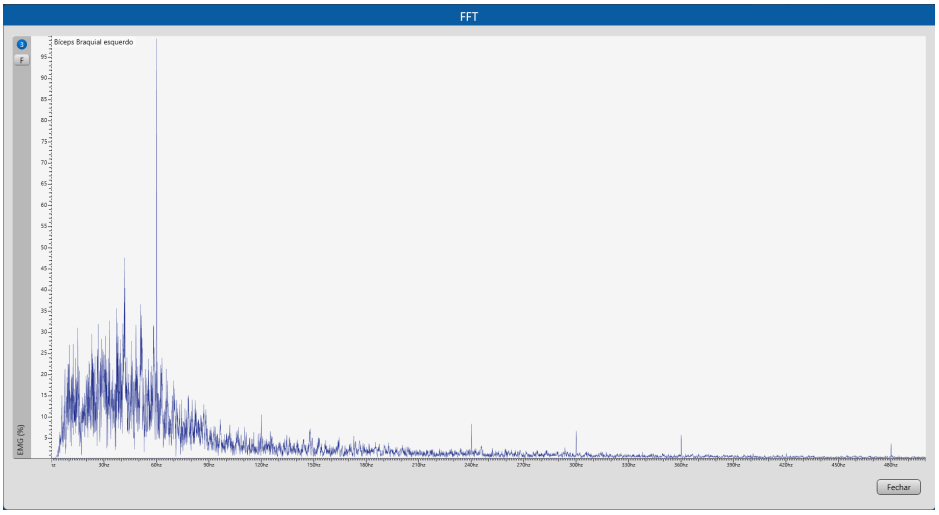

Figura 17: *FFT do sinal amostrado utilizando o software Miotec Suite*
Fonte: Autor

4.3 Testes planejados

A proposta original do projeto consistia na execução de alguns métodos de avaliação do equilíbrio que fazem parte dos protocolos de testes do laboratório de Fisioterapia da UFCSPA.

Em decorrência da pandemia de Sars-CoV-2 e os protocolos sanitários impostos não foi possível a execução destes estudos, com o objetivo de evitar a exposição de pacientes à um risco de contaminação, e também pela situação de que o laboratório em que os estudos seriam realizados ficou interditado desde o início de março até a entrega deste trabalho. Além disso, todos os trabalhos que são realizados com humanos devem passar pela aprovação do comitê de ética da UFCSPA, que devido à situação mencionada, não está recebendo novas solicitações de procedimentos.

São alguns dos testes que haviam sido planejados para a avaliação do desempenho da ferramenta:

- Teste de alcance: O paciente sentado deve estender o braço para atingir com a mão um alvo pré-definido, retornando o braço para a posição original. São avaliados neste teste, o tempo de execução da tarefa, a precisão do alvo, o percurso realizado e a suavidade do movimento durante o percurso.

- Teste postural estático: O paciente em pé deve postar-se em postura ereta e manter o corpo nesta posição. São avaliados a capacidade de manter-se nesta posição, bem como as atividades musculares e os movimentos realizados pelo paciente para permanecer em equilíbrio.

- *Sit and Stand*: O paciente deve alternar entre as posições sentado e em pé, com o objetivo de avaliar o condicionamento físico. Também podem ser analisados com maior detalhe os movimentos transitórios entre as posições, pois normalmente a DP acarreta em anomalias posturais estáticas e dinâmicas.

- Análise da Marcha: O paciente deve marchar por um determinado período de avaliação. Neste estudo são mensurados o comprimento do passo, os tempos de duplo apoio e da passada, além da ativação dos grupos musculares durante o movimento.

- Avaliação da intensidade dos tremores: O paciente realiza um conjunto de tarefas que reproduzem atividades cotidianas e situações de esforço, podendo ser avaliada a evolução da doença através da intensidade dos tremores.
- Testes de mobilidade: O paciente deve equilibrar-se com apenas um dos pés e alternar a distribuição da pressão nas regiões do pé de apoio. Nestes testes é possível avaliar a capacidade de equilíbrio, a ativação muscular para manter a posição e os comportamentos transitórios durante as instabilidades no posicionamento.

Após a execução destes métodos avaliativos, os dados brutos adquiridos passaríam por um processamento onde seria possível a análise correlacional do equilíbrio com a DP.

5 CONCLUSÃO

No decorrer deste trabalho foi possível desenvolver uma ferramenta simples, mas que traz um benefício considerável aos estudos da área da saúde. A aplicação consegue fornecer ao usuário o mesmo nível de precisão que o software proprietário do dispositivo de EMG fornece, agregando ainda sinais de sensores inerciais, podendo contribuir consideravelmente para a pesquisa na análise do equilíbrio sua correlação com a Doença de Parkinson.

O *feedback* sobre a ferramenta foi positivo e espera-se, após a normalização da Pandemia de Sars-CoV-2, os estudos possam ser retomados e que a ferramenta auxilie os profissionais envolvidos.

Uma vez que não foi possível, durante a execução deste trabalho, a realização de testes com pacientes acomedidos por DP, planeja-se após a normalização da Pandemia de Sars-CoV-2, retomar as atividades do laboratório de fisioterapia da UFCSPA, onde será possível realizar testes mais aprofundados da ferramenta, podendo ser realizados ajustes nas rotinas, de forma a adequar a aplicação às necessidades do estudo.

Propõe-se uma oportunidade de melhoria para o sistema na aplicação de filtros digitais para eliminar frequências indesejadas como apresentado na seção 4.2. É válido salientar que esta melhoria não se trata de uma solução simples, pois existem estudos na área da saúde que identificaram sinais relevantes do corpo humano na faixa de frequência de 60 Hz. Desta forma, é necessário tomar-se o devido cuidado na execução deste filtro, para não desprezar sinais importantes para os estudos vindouros.

APÊNDICE A - CÓDIGO-FONTE DO SISTEMA DE AQUISIÇÃO

```
 1  using System;
 2  using System.Collections.Generic;
 3  using System.ComponentModel;
 4  using System.Data;
 5  using System.Drawing;
 6  using System.IO;
 7  using System.Linq;
 8  using System.Net;
 9  using System.Net.Sockets;
10  using System.Text;
11  using System.Threading.Tasks;
12  using System.Windows.Forms;
13  using System.Windows.Forms.DataVisualization.Charting;
14  using System.IO.Ports;
15  using System.Threading;
16  using System.Runtime.Serialization;
17  using System.Json;
18  using OfficeOpenXml;
19  using OfficeOpenXml.Table;
20  using OfficeOpenXml.Drawing.Chart;
21  using System.Globalization;
22  using System.Diagnostics;
23
24  namespace WindowsFormsApp1
25  {
26      public partial class Form1 : Form
27      {
28          //Parametros de Comunicação EMG
29          String portName = "";
30          SerialPort _port;
31
32          //Parametros de Comunicação IMU
33          IPAddress IMUConnection = Dns.GetHostByName(Dns.GetHostName
              ()).AddressList[0];
34
35          //Carregamento do Horário de Inicialização
36          DateTime initializationDateTime = DateTime.Now;
37
38          //Definição do diretório de arquivamento dos dados padrão
39          String directoryName = @"C:\Temp\Dados " +
              DateTime.Now.ToShortDateString().Replace("/", ".");
40
41          //DataTables de todos os Dados
42          DataTable PhoneAccelerometer = new DataTable();
43          DataTable PhoneGyroscope = new DataTable();
44          DataTable EMG = new DataTable();
45
46          //DataTables de Saída
47          DataTable AccToSave = new DataTable();
48          DataTable GyrToSave = new DataTable();
49          DataTable EMGToSave = new DataTable();
50
51          //Ativa aquisição nas tabelas de arquivamento
52          Boolean SalvarDados = false;
53          DateTime AquisitionStartTime = DateTime.Now;
54
```

```
55          //Marcar Amostra
56          Boolean MarcarIMU = false;
57          Boolean MarcarEMG = false;
58
59          //Tratamento dos dados de coleta do EMG
60          String EMGGain = "";
61          int nPoints = 1000;
62
63          public Form1()
64          {
65              //Necessário para trabalhar em computadores com região no Brasil
66              System.Globalization.CultureInfo customCulture =
                  (System.Globalization.CultureInfo)
                  System.Threading.Thread.CurrentThread.CurrentCulture.Clone();
67              customCulture.NumberFormat.NumberDecimalSeparator = ".";
68              System.Threading.Thread.CurrentThread.CurrentCulture =
                  customCulture;
69
70              //Inicialização da Interface Gráfica
71              InitializeComponent();
72          }
73
74          private void Form1_Load(object sender, EventArgs e)
75          {
76              //Início do timer do relógio
77              timer1.Start();
78
79              //Início do Timer de atualização dos Gráficos
80              timer2.Start();
81
82              comboBox3.SelectedItem = "Manual";
83
84              //Carregamento das Portas COM disponíveis para o EMG
85              string[] ports = SerialPort.GetPortNames();
86              comboBox1.Items.Clear();
87              foreach (string port in ports)
88              {
89                  comboBox1.Items.Add(port);
90              }
91
92              //Inicialização da pasta de destino na interface
93              label3.Text = "Pasta: " + Environment.NewLine + directoryName;
94
95              //Texto de ajuda para visualização de caminhos de arquivo
                  personalizados com mais caracteres
96              toolTip1.SetToolTip(label3, directoryName);
97
98              //Inicialização de pontos na escala horizontal dos gráficos
99              nPoints = Convert.ToInt32(numericUpDown1.Value);
100
101             //Carregamento dos dados de conexão do computador com a IMU
102             label1.Text = "Dados de Conexão: " + Environment.NewLine +
                  IMUConnection.ToString() + ":80";
103
104             //Inicia o servidor da IMU e aguarda o recebimento de informações
105             backgroundWorker1.RunWorkerAsync();
```

```
106         }
107
108         private void Form1_FormClosing(object sender, FormClosingEventArgs e)
109         {
110             //Interrompe a atualização dos gráficos durante o encerramento do
                programa
111             timer2.Stop();
112
113             //Interrompe a conexão com o EMG
114             if((comboBox1.SelectedIndex != -1) && (button1.Enabled = false))
115             {
116                 Stop();
117                 _port.Close();
118             }
119
120             //Caso o diretório de arquivamento dos dados não exista, cria
121             if (!Directory.Exists(directoryName))
122             {
123                 Directory.CreateDirectory(directoryName);
124             }
125
126             //Arquivamento dos ensaios com o acelerômetro
127             if(AccToSave.Rows.Count > 0)
128             {
129                 lock (AccToSave)
130                 {
131                     ExportaDataTable(AccToSave, "Acelerômetro");
132                 }
133             }
134
135             //Arquivamento dos ensaios com o giroscópio
136             if (GyrToSave.Rows.Count > 0)
137             {
138                 lock (GyrToSave)
139                 {
140                     ExportaDataTable(GyrToSave, "Giroscópio");
141                 }
142             }
143
144             //Arquivamento dos ensaios com o EMG
145             if (EMGToSave.Rows.Count > 0)
146             {
147                 lock (EMGToSave)
148                 {
149                     ExportaDataTable(EMGToSave, "Eletromiógrafo");
150                 }
151             }
152
153             //Solicitação para abrir a pasta de resultados
154             if(MessageBox.Show("Resultados Exportados. Deseja mostrar a
                pasta?","Abrir resultados?",MessageBoxButtons.YesNo) ==
                DialogResult.Yes)
155             {
156                 System.Diagnostics.Process.Start(directoryName);
157             }
158         }
```

```
159
160         private void backgroundWorker1_DoWork(object sender, DoWorkEventArgs
              e)
161         {
162             TcpListener server = new TcpListener(IMUConnection, 80);
163
164             server.Start();
165
166             TcpClient client = server.AcceptTcpClient();
167
168             NetworkStream stream = client.GetStream();
169
170             //Inicialização das Colunas das DataTables
171             PhoneAccelerometer.Columns.Add("ACC_DT");
172             PhoneAccelerometer.Columns.Add("ACC_ST");
173             PhoneAccelerometer.Columns.Add("ACC_X");
174             PhoneAccelerometer.Columns.Add("ACC_Y");
175             PhoneAccelerometer.Columns.Add("ACC_Z");
176             PhoneAccelerometer.Columns.Add("FLAG");
177
178             AccToSave.Columns.Add("ACC_DT");
179             AccToSave.Columns.Add("ACC_ST");
180             AccToSave.Columns.Add("ACC_X");
181             AccToSave.Columns.Add("ACC_Y");
182             AccToSave.Columns.Add("ACC_Z");
183             AccToSave.Columns.Add("FLAG");
184
185             PhoneGyroscope.Columns.Add("GYR_DT");
186             PhoneGyroscope.Columns.Add("GYR_ST");
187             PhoneGyroscope.Columns.Add("GYR_X");
188             PhoneGyroscope.Columns.Add("GYR_Y");
189             PhoneGyroscope.Columns.Add("GYR_Z");
190             PhoneGyroscope.Columns.Add("FLAG");
191
192             GyrToSave.Columns.Add("GYR_DT");
193             GyrToSave.Columns.Add("GYR_ST");
194             GyrToSave.Columns.Add("GYR_X");
195             GyrToSave.Columns.Add("GYR_Y");
196             GyrToSave.Columns.Add("GYR_Z");
197             GyrToSave.Columns.Add("FLAG");
198
199             //TimeStamp Inicial
200             long AccTimeStamp = -1, GyrTimeStamp = -1;
201
202             //Ciclo infinito de aquisição de dados da IMU
203             while (true)
204             {
205                 try
206                 {
207                     //Trata os dados adquiridos e descarta linhas incompletas
208                     while (!stream.DataAvailable) ;
209                     byte[] bytes = new byte[client.Available];
210                     stream.Read(bytes, 0, client.Available);
211                     string s = Encoding.UTF8.GetString(bytes);
212                     List<String> lines = s.Split
                        (Environment.NewLine.ToCharArray()).ToList();
```

```
213                     List<String> CorrectLines = new List<String>();
214
215                     foreach (String line in lines)
216                     {
217                         int open = 0, close = 0;
218                         foreach (char c in line.ToCharArray())
219                         {
220                             if (c == '{')
221                             {
222                                 open++;
223                             }
224                             else if (c == '}')
225                             {
226                                 close++;
227                             }
228                         }
229                         if ((open == close) || (line.Length > 160))
230                         {
231                             CorrectLines.Add(line);
232                         }
233                     }
234
235                     //Decodifica o pacote JSON e armazena os dados nas tabelas
236                     foreach (String line in CorrectLines)
237                     {
238                         if (line != "")
239                         {
240                             JsonValue Package = JsonObject.Parse(line);
241                             foreach (KeyValuePair<string, JsonValue> Sensor in
                     Package)
242                             {
243                                 DateTime dateTime = DateTime.Now;
244                                 if (Sensor.Key == "accelerometer")
245                                 {
246                                     JsonValue accelerometer = Sensor.Value;
247
248                                     if (AccTimeStamp == -1 && SalvarDados)
249                                     {
250                                         AccTimeStamp = Convert.ToInt64
                    (accelerometer["timestamp"].ToString());
251                                     }
252
253                                     PhoneAccelerometer.Rows.Add(dateTime, 0,
                    Double.Parse(accelerometer["value"][0].ToString()),
                    Double.Parse(accelerometer["value"][1].ToString()),
                    Double.Parse(accelerometer["value"][2].ToString()),
                    MarcarIMU);
254                                     if (SalvarDados)
255                                     {
256                                         AccToSave.Rows.Add(dateTime,
                    (Convert.ToInt64(accelerometer["timestamp"].ToString()) -
                    AccTimeStamp)/1000000, Double.Parse(accelerometer["value"]
                    [0].ToString()), Double.Parse(accelerometer["value"]
                    [1].ToString()), Double.Parse(accelerometer["value"]
                    [2].ToString()), MarcarIMU);
257                                     }
```

```
258                                 }
259                                 else if (Sensor.Key == "gyroscope")
260                                 {
261                                     JsonValue gyroscope = Sensor.Value;
262 
263                                     if (GyrTimeStamp == -1 && SalvarDados)
264                                     {
265                                         GyrTimeStamp = Convert.ToInt64
                        (gyroscope["timestamp"].ToString());
266                                     }
267 
268                                     PhoneGyroscope.Rows.Add(dateTime, 0,
                        Double.Parse(gyroscope["value"][0].ToString()),
                        Double.Parse(gyroscope["value"][1].ToString()),
                        Double.Parse(gyroscope["value"][2].ToString()),MarcarIMU);
269                                     if (SalvarDados)
270                                     {
271                                         GyrToSave.Rows.Add(dateTime,
                        (Convert.ToInt64(gyroscope["timestamp"].ToString()) -
                        GyrTimeStamp)/1000000, Double.Parse(gyroscope["value"]
                        [0].ToString()), Double.Parse(gyroscope["value"]
                        [1].ToString()), Double.Parse(gyroscope["value"]
                        [2].ToString()),MarcarIMU);
272                                     }
273                                 }
274                             }
275                         }
276                     }
277                 }
278                 catch(Exception excp)
279                 {
280 
281                 }
282             }
283         }
284 
285         private void backgroundWorker2_DoWork(object sender, DoWorkEventArgs
              e)
286         {
287             EMG.Columns.Add("EMG_DT");
288             EMG.Columns.Add("EMG_ST");
289             EMG.Columns.Add("EMG_CH1");
290             EMG.Columns.Add("EMG_CH2");
291             EMG.Columns.Add("EMG_CH3");
292             EMG.Columns.Add("EMG_CH4");
293             EMG.Columns.Add("FLAG");
294 
295             EMGToSave.Columns.Add("EMG_DT");
296             EMGToSave.Columns.Add("EMG_ST");
297             EMGToSave.Columns.Add("EMG_CH1");
298             EMGToSave.Columns.Add("EMG_CH2");
299             EMGToSave.Columns.Add("EMG_CH3");
300             EMGToSave.Columns.Add("EMG_CH4");
301             EMGToSave.Columns.Add("FLAG");
302 
303             _port = new SerialPort(portName, 230400, Parity.None, 8,
```

```
                StopBits.One);
304
305             if (_port.IsOpen)
306             {
307                 var response = EnviarReceber
                      (MiotoolCodes.ConfirmStopAcquisition);
308                 MiotoolApi.IsValidFrame(response);
309
310                 Thread.Sleep(150);
311
312                 while (_port.BytesToRead > 0)
313                 {
314                     byte[] buffer = new byte[_port.BytesToRead];
315                     _port.Read(buffer, 0, _port.BytesToRead);
316                     Thread.Sleep(150);
317                 }
318
319                 _port.Close();
320             }
321
322             Open();
323
324             var gains = MiotoolApi.ArrayGanhos(MiotoolGains.G100);
325             switch (EMGGain)
326             {
327                 case "G100":
328                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G100);
329                     break;
330                 case "G200":
331                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G200);
332                     break;
333                 case "G400":
334                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G400);
335                     break;
336                 case "G500":
337                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G500);
338                     break;
339                 case "G800":
340                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G800);
341                     break;
342                 case "G1000":
343                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G1000);
344                     break;
345                 case "G1600":
346                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G1600);
347                     break;
348                 case "G2000":
349                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G2000);
350                     break;
351                 default:
352                     gains = MiotoolApi.ArrayGanhos(MiotoolGains.G100);
353                     break;
354             }
355
356             EnviarReceber(MiotoolCodes.StartAcquisition, gains);
357
```

```
358             var extraBytes = new byte[2];
359             _port.Read(extraBytes, 0, extraBytes.Length);
360
361             DateTime EMGAquisitionStartTime = DateTime.MinValue;
362             List<int> milisecs = new List<int> { 0, 0 };
363
364             while (true)
365             {
366                 if (EMGAquisitionStartTime == DateTime.MinValue &&
                      SalvarDados)
367                 {
368                     EMGAquisitionStartTime = DateTime.Now;
369                 }
370
371                 var response = new byte[8];
372                 _port.Read(response, 0, response.Length);
373
374                 var intVals = Enumerable.Range(0, 4)
375                     .Select(i => new[] { BitConverter.ToInt16(response, 2 * i)
                         >> 2 })
376                     .ToArray();
377
378                 int ganho = int.Parse(EMGGain.Substring(1));
379
380                 lock (EMG)
381                 {
382                     if (true)
383                     {
384                         EMG.Rows.Add(DateTime.Now, 0, intVals[1][0] *
                        ganho/200, intVals[2][0] * ganho / 200, intVals[3][0] *
                        ganho / 200, intVals[0][0] * ganho / 200, MarcarEMG);
385                         if (SalvarDados)
386                         {
387                             if (EMGAquisitionStartTime != DateTime.MinValue)
388                             {
389                                 DateTime now = DateTime.Now;
390
391                                 if (milisecs.Count > 1)
392                                 {
393                                     milisecs.RemoveAt(0);
394                                 }
395                                 milisecs.Add((now -
                            EMGAquisitionStartTime).Milliseconds);
396
397                                 if (milisecs[1] < milisecs[0])
398                                 {
399                                     milisecs[1] += 1000;
400                                 }
401
402                                 EMGToSave.Rows.Add(now, milisecs[1], intVals
                            [1][0] * ganho / 200, intVals[2][0] * ganho / 200, intVals
                            [3][0] * ganho / 200, intVals[0][0] * ganho / 200,
                            MarcarEMG);
403                             }
404                         }
405                     }
```

```
406                 }
407             }
408         }
409
410         #region Miotec API
411         public string Id
412         {
413             get
414             {
415                 var response = EnviarReceber(MiotoolCodes.ReadSerialNumber);
416                 var destArray = new byte[4];
417                 Buffer.BlockCopy(response, 2, destArray, 0, 4);
418                 var result = MiotoolApi.DecodeSerialNumber(destArray).ToString
                      ();
419
420                 return result;
421             }
422         }
423
424         public double BatteryLevel
425         {
426             get
427             {
428                 var leituraBateria = EnviarReceber(MiotoolCodes.ReadBattery);
429
430                 var val = BitConverter.ToUInt16(leituraBateria.Skip(2).Take
                      (2).Reverse().ToArray(), 0);
431                 var max = BitConverter.ToUInt16(leituraBateria.Skip(4).Take
                      (2).Reverse().ToArray(), 0);
432                 var min = BitConverter.ToUInt16(leituraBateria.Skip(6).Take
                      (2).Reverse().ToArray(), 0);
433
434                 double range = max - min;
435                 double normalized = val - min;
436
437                 if (range <= 0) return 0;
438
439                 var valor = 100 * normalized / range;
440
441                 return valor;
442             }
443         }
444
445         public void Stop()
446         {
447             var response = EnviarReceber(MiotoolCodes.ConfirmStopAcquisition);
448             MiotoolApi.IsValidFrame(response);
449         }
450
451         public void Open()
452         {
453             _port.Open();
454
455             var response = EnviarReceber(MiotoolCodes.Connect);
456         }
457
```

```
458         public byte[] EnviarReceber(byte code, byte[] payload = null)
459         {
460             Enviar(code, payload);
461
462             Thread.Sleep(150);
463
464             var response = new byte[10];
465             var status = _port.Read(response, 0, response.Length);
466
467             return response;
468         }
469
470         void Enviar(byte code, byte[] payload = null)
471         {
472             var message = MiotoolApi.CreateMessage(code, payload);
473             _port.Write(message, 0, message.Length);
474         }
475         #endregion
476
477         private void comboBox1_SelectedIndexChanged(object sender, EventArgs
              e)
478         {
479             if(comboBox1.SelectedIndex != -1)
480             {
481                 portName = comboBox1.Text;
482                 button1.Enabled = true;
483             }
484         }
485
486         private void button1_Click_1(object sender, EventArgs e)
487         {
488             button1.Enabled = false;
489             comboBox1.Enabled = false;
490             comboBox2.Enabled = false;
491             EMGGain = comboBox2.Text;
492             backgroundWorker2.RunWorkerAsync();
493         }
494
495         private void button2_Click(object sender, EventArgs e)
496         {
497             if(folderBrowserDialog1.ShowDialog() == DialogResult.OK)
498             {
499                 directoryName = folderBrowserDialog1.SelectedPath + @"\Dados "
                      + initializationDateTime.ToShortDateString().Replace("/",
                    ".");
500                 if (directoryName.Length > 27)
501                 {
502                     label3.Text = "Pasta: " + Environment.NewLine +
                        directoryName.Substring(0,24) + "(...)";
503                 }
504                 else
505                 {
506                     label3.Text = "Pasta: " + Environment.NewLine +
                        directoryName;
507                 }
508                 toolTip1.SetToolTip(label3, directoryName);
```

```
509             }
510         }
511
512         private void numericUpDown1_ValueChanged(object sender, EventArgs e)
513         {
514             nPoints = Convert.ToInt32(numericUpDown1.Value);
515         }
516
517         private void timer1_Tick(object sender, EventArgs e)
518         {
519             this.Text = "SISTEMA SUPERVISÓRIO PARA ESTUDO DE EQUILÍBRIO   [" +
                DateTime.Now.ToString("dd/MM/yyyy - HH:mm:ss.f") + "]";
520         }
521
522         private void ExportaDataTable(DataTable Dados, String Sensor)
523         {
524             //IMPORTAÇÃO DA DATATABLE
525             FileInfo fileInfo = new FileInfo(directoryName + "\\" + Sensor + "
                " + initializationDateTime.ToLongTimeString().Replace(":", "_")
                + ".xlsx");
526             ExcelPackage package = new ExcelPackage(fileInfo);
527             ExcelWorkbook workbook = package.Workbook;
528             ExcelWorksheet sheet = workbook.Worksheets.Add("Dados");
529             sheet.Cells["A1"].LoadFromDataTable(Dados, true);
530
531             //ADEQUAÇÃO DOS DADOS PARA O SEPARADOR DECIMAL
532             for (int i = 2; i <= sheet.Dimension.Rows; i++)
533             {
534                 sheet.Cells[i, 1].Value = DateTime.Parse(sheet.Cells[i,
                    1].Value.ToString()).ToString("dd/MM/yyyy - HH:mm:ss.f");
535                 for (int j = 3; j <= sheet.Dimension.Columns-1; j++)
536                 {
537                     sheet.Cells[i, j].Value = Convert.ToDouble(sheet.Cells[i,
                        j].Value);
538                 }
539             }
540
541             //CRIAÇÃO DO GRÁFICO
542             ExcelChartsheet chartsheet = workbook.Worksheets.AddChart
                  ("Gráfico", eChartType.LineStacked);
543             ExcelChart chart = chartsheet.Chart;
544             for (int i = 3; i <= sheet.Dimension.Columns-1; i++)
545             {
546                 char currentLetter = (char)(i + 64);
547                 String XRANGE = "Dados!A2:A" + sheet.Dimension.Rows.ToString
                    ();
548                 String YRANGE = "Dados!" + currentLetter + "2:" +
                    currentLetter + sheet.Dimension.Rows.ToString();
549                 ExcelChartSerie serie = chart.Series.Add(YRANGE, XRANGE);
550                 serie.Header = sheet.Cells["Dados!" + currentLetter +
                    "1"].Value.ToString();
551             }
552
553             //SALVAR OS DADOS
554             package.SaveAs(fileInfo);
555             package.Dispose();
```

```
556         }
557
558         private void timer2_Tick(object sender, EventArgs e)
559         {
560             chart1.ChartAreas[0].AxisX.Minimum = 0;
561             chart1.ChartAreas[0].AxisX.Maximum = nPoints;
562             chart1.ChartAreas[0].AxisY.Title = "Aceleração [m/s²]";
563             chart1.Series.Clear();
564             chart1.Series.Add("X Axis");
565             chart1.Series["X Axis"].ChartType = SeriesChartType.Spline;
566             chart1.Series["X Axis"].XValueMember = "ACC_DT";
567             chart1.Series["X Axis"].YValueMembers = "ACC_X";
568             chart1.Series["X Axis"].YValueType = ChartValueType.Double;
569             chart1.Series.Add("Y Axis");
570             chart1.Series["Y Axis"].ChartType = SeriesChartType.Spline;
571             chart1.Series["Y Axis"].XValueMember = "ACC_DT";
572             chart1.Series["Y Axis"].YValueMembers = "ACC_Y";
573             chart1.Series["Y Axis"].YValueType = ChartValueType.Double;
574             chart1.Series.Add("Z Axis");
575             chart1.Series["Z Axis"].ChartType = SeriesChartType.Spline;
576             chart1.Series["Z Axis"].XValueMember = "ACC_DT";
577             chart1.Series["Z Axis"].YValueMembers = "ACC_Z";
578             chart1.Series["Z Axis"].YValueType = ChartValueType.Double;
579             if (PhoneAccelerometer.Rows.Count > 0)
580             {
581                 lock (PhoneAccelerometer)
582                 {
583                     try
584                     {
585                         chart1.DataSource = PhoneAccelerometer.AsEnumerable
                    ().Reverse().Take(nPoints).Reverse().CopyToDataTable();
586                     }
587                     catch
588                     {
589                     }
590                 }
591             }
592             chart1.ChartAreas[0].RecalculateAxesScale();
593
594             chart2.ChartAreas[0].AxisX.Minimum = 0;
595             chart2.ChartAreas[0].AxisX.Maximum = nPoints;
596             chart2.ChartAreas[0].AxisY.Title = "Velocidade Angular [rad/s]";
597             chart2.Series.Clear();
598             chart2.Series.Add("X Axis");
599             chart2.Series["X Axis"].ChartType = SeriesChartType.Spline;
600             chart2.Series["X Axis"].XValueMember = "GYR_DT";
601             chart2.Series["X Axis"].YValueMembers = "GYR_X";
602             chart2.Series["X Axis"].YValueType = ChartValueType.Double;
603             chart2.Series.Add("Y Axis");
604             chart2.Series["Y Axis"].ChartType = SeriesChartType.Spline;
605             chart2.Series["Y Axis"].XValueMember = "GYR_DT";
606             chart2.Series["Y Axis"].YValueMembers = "GYR_Y";
607             chart2.Series["Y Axis"].YValueType = ChartValueType.Double;
608             chart2.Series.Add("Z Axis");
609             chart2.Series["Z Axis"].ChartType = SeriesChartType.Spline;
610             chart2.Series["Z Axis"].XValueMember = "GYR_DT";
```

```
611             chart2.Series["Z Axis"].YValueMembers = "GYR_Z";
612             chart2.Series["Z Axis"].YValueType = ChartValueType.Double;
613             if (PhoneGyroscope.Rows.Count > 0)
614             {
615                 lock (PhoneGyroscope)
616                 {
617                     try
618                     {
619                         chart2.DataSource = PhoneGyroscope.AsEnumerable
                    ().Reverse().Take(nPoints).Reverse().CopyToDataTable();
620                     }
621                     catch
622                     {
623                     }
624                 }
625             }
626             chart2.ChartAreas[0].RecalculateAxesScale();
627 
628             DataTable tempTable = null;
629             lock (EMG)
630             {
631                 tempTable = EMG;
632             }
633 
634             if (tempTable != null)
635             {
636                 chart3.ChartAreas[0].AxisX.Minimum = 0;
637                 chart3.ChartAreas[0].AxisX.Maximum = nPoints;
638                 chart3.ChartAreas[0].AxisY.Minimum = -1000;
639                 chart3.ChartAreas[0].AxisY.Maximum = 1000;
640                 chart3.ChartAreas[0].AxisY.Title = "EMG [µV]";
641                 chart3.Series.Clear();
642                 if (checkBox1.Checked)
643                 {
644                     chart3.Series.Add("Channel 1");
645                     chart3.Series["Channel 1"].ChartType =
                    SeriesChartType.Spline;
646                     chart3.Series["Channel 1"].XValueMember = "EMG_DT";
647                     chart3.Series["Channel 1"].YValueMembers = "EMG_CH1";
648                     chart3.Series["Channel 1"].YValueType =
                    ChartValueType.Double;
649                 }
650                 if (checkBox2.Checked)
651                 {
652                     chart3.Series.Add("Channel 2");
653                     chart3.Series["Channel 2"].ChartType =
                    SeriesChartType.Spline;
654                     chart3.Series["Channel 2"].XValueMember = "EMG_DT";
655                     chart3.Series["Channel 2"].YValueMembers = "EMG_CH2";
656                     chart3.Series["Channel 2"].YValueType =
                    ChartValueType.Double;
657                 }
658                 if (checkBox3.Checked)
659                 {
660                     chart3.Series.Add("Channel 3");
661                     chart3.Series["Channel 3"].ChartType =
```

```
                    SeriesChartType.Spline;
662                 chart3.Series["Channel 3"].XValueMember = "EMG_DT";
663                 chart3.Series["Channel 3"].YValueMembers = "EMG_CH3";
664                 chart3.Series["Channel 3"].YValueType =
                    ChartValueType.Double;
665             }
666             if (checkBox4.Checked)
667             {
668                 chart3.Series.Add("Channel 4");
669                 chart3.Series["Channel 4"].ChartType =
                    SeriesChartType.Spline;
670                 chart3.Series["Channel 4"].XValueMember = "EMG_DT";
671                 chart3.Series["Channel 4"].YValueMembers = "EMG_CH4";
672                 chart3.Series["Channel 4"].YValueType =
                    ChartValueType.Double;
673             }
674             if (checkBox1.Checked || checkBox2.Checked ||
                  checkBox3.Checked || checkBox4.Checked)
675             {
676                 lock (tempTable)
677                 {
678                     try
679                     {
680                         chart3.DataSource = tempTable.AsEnumerable
                    ().Reverse().Take(nPoints).Reverse().CopyToDataTable();
681                     }
682                     catch { }
683                 }
684             }
685         }
686     }
687
688     private void button3_Click(object sender, EventArgs e)
689     {
690         if (comboBox3.SelectedIndex != -1)
691         {
692             comboBox3.Enabled = false;
693             if (!SalvarDados)
694             {
695                 AquisitionStartTime = DateTime.Now;
696                 SalvarDados = true;
697                 button3.BackColor = Color.Red;
698                 if (comboBox3.Text != "Manual")
699                 {
700                     button3.Enabled = false;
701                     button3.Text = "Wait...";
702                     switch (comboBox3.Text)
703                     {
704                         case "15s":
705                             timer3.Interval = 15000;
706                             break;
707                         case "30s":
708                             timer3.Interval = 30000;
709                             break;
710                         case "60s":
711                             timer3.Interval = 60000;
```

```
712                                 break;
713                             }
714                             timer3.Start();
715                         }
716                         else
717                         {
718                             button3.Text = "■ Stop";
719                         }
720                     }
721                     else
722                     {
723                         SalvarDados = false;
724                         comboBox3.Enabled = true;
725                         button3.BackColor = Color.Green;
726                         button3.Text = "▷ Play";
727                     }
728                 }
729                 else
730                 {
731                     MessageBox.Show("É necessário informar um intervalo de aquisição");
732                 }
733             }
734 
735             private void timer3_Tick(object sender, EventArgs e)
736             {
737                 SalvarDados = false;
738                 button3.BackColor = Color.Green;
739                 comboBox3.Enabled = true;
740                 button3.Enabled = true;
741                 button3.Text = "▷ Play";
742                 timer3.Stop();
743             }
744 
745             private void button4_Click(object sender, EventArgs e)
746             {
747                 if(button4.BackColor == Color.Transparent)
748                 {
749                     button4.BackColor = Color.Orange;
750                     button4.Text = "ON";
751                     MarcarEMG = true;
752                     MarcarIMU = true;
753                 }
754                 else
755                 {
756                     button4.BackColor = Color.Transparent;
757                     button4.Text = "OFF";
758                     MarcarEMG = false;
759                     MarcarIMU = false;
760                 }
761             }
762         }
763         #region Miotec API
764         public static class MiotoolApi
765         {
766             public static IEnumerable<SerialPort> GetPorts()
```

```
767         {
768             var result = new List<SerialPort>();
769             foreach (var portName in SerialPort.GetPortNames())
770             {
771                 try
772                 {
773                     var port = new SerialPort(portName)
774                     {
775                         BaudRate = 230400,
776                         DataBits = 8,
777                         StopBits = StopBits.One,
778                         Parity = Parity.None,
779                         DtrEnable = false,
780                         RtsEnable = false,
781                         WriteTimeout = 1000,
782                         ReadTimeout = 1000
783                     };
784 
785                     if (port != null)
786                     {
787                         result.Add(port);
788                     }
789 
790                 }
791                 catch (Exception)
792                 {
793                     throw;
794                 }
795             }
796             return result;
797         }
798 
799         public static bool IsValidFrame(byte[] data)
800         {
801 
802             if (data == null ||
803                 data.Length != 10 ||
804                 data[0] != MiotoolCodes.MiotoolReceivedFrameStart ||
805                 data[8] != MiotoolCodes.MiotoolReceivedFrameEnd)
806 
807                 return false;
808 
809 
810             var result = Checksum(data) == data[9];
811 
812             return result;
813         }
814 
815         public static int DecodeSerialNumber(byte[] data)
816         {
817             Array.Reverse(data);
818             var numeroInflado = BitConverter.ToInt32(data, 0);
819             if (numeroInflado < 0)
820                 return numeroInflado;
821             var hexstring = numeroInflado.ToString("X"); // "X" é hexadecimal
822             var numeroVerdadeiro = int.Parse(hexstring);
```

```
823             return numeroVerdadeiro;
824         }
825
826         public static byte[] CreateMessage(byte code, byte[] payload = null)
827         {
828             var result = new byte[10];
829
830             result[0] = MiotoolCodes.MiotoolFrameStart;
831
832             result[1] = code;
833
834             if (payload != null)
835                 Buffer.BlockCopy(payload, 0, result, 2, payload.Length);
836
837             result[8] = MiotoolCodes.MiotoolFrameEnd;
838
839             result[9] = Checksum(result);
840
841             return result;
842         }
843
844         public static byte Checksum(byte[] result)
845         {
846             // soma todos os valores exceto o último
847             byte checksum = 0;
848             for (var i = 0; i < result.Length - 1; i++)
849                 checksum += result[i];
850             return checksum;
851         }
852
853         public static byte[] ArrayGanhos(MiotoolGains ganho)
854         {
855             var result = Enumerable
856                 .Repeat(ganho, 4)
857                 .Select((g, i) => Convert.ToByte((byte)g + i))
858                 .ToArray();
859
860             return result;
861         }
862     }
863
864     public enum MiotoolGains : byte
865     {
866         // para mais informações, ler o datasheet do chip ADS7871
867         G100 = 8 + 128 + (0 << 4),
868         G200 = 8 + 128 + (1 << 4),
869         G400 = 8 + 128 + (2 << 4),
870         G500 = 8 + 128 + (3 << 4),
871         G800 = 8 + 128 + (4 << 4),
872         G1000 = 8 + 128 + (5 << 4),
873         G1600 = 8 + 128 + (6 << 4),
874         G2000 = 8 + 128 + (7 << 4)
875     }
876
877     public static class MiotoolCodes
878     {
```

```
879         public static byte Connect => 10;
880         public static byte Disconnect => 11;
881         public static byte ConnectionStatus => 12;
882         public static byte ReadSerialNumber => 20;
883         public static byte WriteSerialNumber => 21;
884         public static byte ReadVersion => 30;
885         public static byte ReadModel => 31;
886         public static byte ReadMemory => 40;
887         public static byte WriteMemory => 41;
888         public static byte ReadBattery => 50;
889         public static byte StartAcquisition => 60;
890         public static byte StopAcquisition => 61;
891         public static byte ConfirmStopAcquisition => 62;
892         public static byte SyncDataAquisition => 63;
893         public static byte WriteCalibration => 71;
894         public static byte ReadCalibration => 72;
895         public static byte MiotoolFrameStart => 200;
896         public static byte MiotoolFrameEnd => 206;
897         public static byte MiotoolReceivedFrameStart => 205;
898         public static byte MiotoolReceivedFrameEnd => 207;
899     }
900
901     #endregion
902 }
```

APÊNDICE B - CÓDIGO-FONTE DE INTERFACE GRÁFICA DO SISTEMA DE AQUISIÇÃO

```
1  namespace WindowsFormsApp1
2  {
3      partial class Form1
4      {
5          /// <summary>
6          /// Variável de designer necessária.
7          /// </summary>
8          private System.ComponentModel.IContainer components = null;
9
10         /// <summary>
11         /// Limpar os recursos que estão sendo usados.
12         /// </summary>
13         /// <param name="disposing">true se for necessário descartar os
             recursos gerenciados; caso contrário, false.</param>
14         protected override void Dispose(bool disposing)
15         {
16             if (disposing && (components != null))
17             {
18                 components.Dispose();
19             }
20             base.Dispose(disposing);
21         }
22
23         #region Código gerado pelo Windows Form Designer
24
25         /// <summary>
26         /// Método necessário para suporte ao Designer - não modifique
27         /// o conteúdo deste método com o editor de código.
28         /// </summary>
29         private void InitializeComponent()
30         {
31             this.components = new System.ComponentModel.Container();
32             System.Windows.Forms.DataVisualization.Charting.ChartArea
                 chartArea1 = new
                 System.Windows.Forms.DataVisualization.Charting.ChartArea();
33             System.Windows.Forms.DataVisualization.Charting.Legend legend1 =
                 new System.Windows.Forms.DataVisualization.Charting.Legend();
34             System.Windows.Forms.DataVisualization.Charting.Series series1 =
                 new System.Windows.Forms.DataVisualization.Charting.Series();
35             System.Windows.Forms.DataVisualization.Charting.Series series2 =
                 new System.Windows.Forms.DataVisualization.Charting.Series();
36             System.Windows.Forms.DataVisualization.Charting.Series series3 =
                 new System.Windows.Forms.DataVisualization.Charting.Series();
37             System.Windows.Forms.DataVisualization.Charting.Title title1 = new
                  System.Windows.Forms.DataVisualization.Charting.Title();
38             System.Windows.Forms.DataVisualization.Charting.ChartArea
                 chartArea2 = new
                 System.Windows.Forms.DataVisualization.Charting.ChartArea();
39             System.Windows.Forms.DataVisualization.Charting.Legend legend2 =
                 new System.Windows.Forms.DataVisualization.Charting.Legend();
40             System.Windows.Forms.DataVisualization.Charting.Series series4 =
                 new System.Windows.Forms.DataVisualization.Charting.Series();
41             System.Windows.Forms.DataVisualization.Charting.Series series5 =
                 new System.Windows.Forms.DataVisualization.Charting.Series();
42             System.Windows.Forms.DataVisualization.Charting.Series series6 =
                 new System.Windows.Forms.DataVisualization.Charting.Series();
```

```
43              System.Windows.Forms.DataVisualization.Charting.Title title2 = new
                    System.Windows.Forms.DataVisualization.Charting.Title();
44              System.Windows.Forms.DataVisualization.Charting.ChartArea
                  chartArea3 = new
                  System.Windows.Forms.DataVisualization.Charting.ChartArea();
45              System.Windows.Forms.DataVisualization.Charting.Legend legend3 =
                  new System.Windows.Forms.DataVisualization.Charting.Legend();
46              System.Windows.Forms.DataVisualization.Charting.Series series7 =
                  new System.Windows.Forms.DataVisualization.Charting.Series();
47              System.Windows.Forms.DataVisualization.Charting.Series series8 =
                  new System.Windows.Forms.DataVisualization.Charting.Series();
48              System.Windows.Forms.DataVisualization.Charting.Series series9 =
                  new System.Windows.Forms.DataVisualization.Charting.Series();
49              System.Windows.Forms.DataVisualization.Charting.Series series10 =
                  new System.Windows.Forms.DataVisualization.Charting.Series();
50              System.Windows.Forms.DataVisualization.Charting.Title title3 = new
                    System.Windows.Forms.DataVisualization.Charting.Title();
51              System.ComponentModel.ComponentResourceManager resources = new
                  System.ComponentModel.ComponentResourceManager(typeof(Form1));
52              this.chart1 = new
                  System.Windows.Forms.DataVisualization.Charting.Chart();
53              this.backgroundWorker1 = new
                  System.ComponentModel.BackgroundWorker();
54              this.chart2 = new
                  System.Windows.Forms.DataVisualization.Charting.Chart();
55              this.label1 = new System.Windows.Forms.Label();
56              this.groupBox1 = new System.Windows.Forms.GroupBox();
57              this.backgroundWorker2 = new
                  System.ComponentModel.BackgroundWorker();
58              this.chart3 = new
                  System.Windows.Forms.DataVisualization.Charting.Chart();
59              this.groupBox2 = new System.Windows.Forms.GroupBox();
60              this.comboBox2 = new System.Windows.Forms.ComboBox();
61              this.label5 = new System.Windows.Forms.Label();
62              this.checkBox3 = new System.Windows.Forms.CheckBox();
63              this.checkBox4 = new System.Windows.Forms.CheckBox();
64              this.checkBox2 = new System.Windows.Forms.CheckBox();
65              this.checkBox1 = new System.Windows.Forms.CheckBox();
66              this.comboBox1 = new System.Windows.Forms.ComboBox();
67              this.label2 = new System.Windows.Forms.Label();
68              this.button1 = new System.Windows.Forms.Button();
69              this.pictureBox1 = new System.Windows.Forms.PictureBox();
70              this.pictureBox2 = new System.Windows.Forms.PictureBox();
71              this.splitContainer1 = new System.Windows.Forms.SplitContainer();
72              this.groupBox6 = new System.Windows.Forms.GroupBox();
73              this.button4 = new System.Windows.Forms.Button();
74              this.groupBox5 = new System.Windows.Forms.GroupBox();
75              this.button3 = new System.Windows.Forms.Button();
76              this.comboBox3 = new System.Windows.Forms.ComboBox();
77              this.label7 = new System.Windows.Forms.Label();
78              this.groupBox4 = new System.Windows.Forms.GroupBox();
79              this.label4 = new System.Windows.Forms.Label();
80              this.numericUpDown1 = new System.Windows.Forms.NumericUpDown();
81              this.groupBox3 = new System.Windows.Forms.GroupBox();
82              this.button2 = new System.Windows.Forms.Button();
83              this.label3 = new System.Windows.Forms.Label();
```

```
84              this.folderBrowserDialog1 = new
                  System.Windows.Forms.FolderBrowserDialog();
85              this.toolTip1 = new System.Windows.Forms.ToolTip(this.components);
86              this.timer1 = new System.Windows.Forms.Timer(this.components);
87              this.timer2 = new System.Windows.Forms.Timer(this.components);
88              this.timer3 = new System.Windows.Forms.Timer(this.components);
89              ((System.ComponentModel.ISupportInitialize)
                  (this.chart1)).BeginInit();
90              ((System.ComponentModel.ISupportInitialize)
                  (this.chart2)).BeginInit();
91              this.groupBox1.SuspendLayout();
92              ((System.ComponentModel.ISupportInitialize)
                  (this.chart3)).BeginInit();
93              this.groupBox2.SuspendLayout();
94              ((System.ComponentModel.ISupportInitialize)
                  (this.pictureBox1)).BeginInit();
95              ((System.ComponentModel.ISupportInitialize)
                  (this.pictureBox2)).BeginInit();
96              ((System.ComponentModel.ISupportInitialize)
                  (this.splitContainer1)).BeginInit();
97              this.splitContainer1.Panel1.SuspendLayout();
98              this.splitContainer1.Panel2.SuspendLayout();
99              this.splitContainer1.SuspendLayout();
100             this.groupBox6.SuspendLayout();
101             this.groupBox5.SuspendLayout();
102             this.groupBox4.SuspendLayout();
103             ((System.ComponentModel.ISupportInitialize)
                  (this.numericUpDown1)).BeginInit();
104             this.groupBox3.SuspendLayout();
105             this.SuspendLayout();
106             //
107             // chart1
108             //
109             this.chart1.Anchor = ((System.Windows.Forms.AnchorStyles)
                  ((System.Windows.Forms.AnchorStyles.Left |
                  System.Windows.Forms.AnchorStyles.Right)));
110             this.chart1.BorderlineColor = System.Drawing.Color.Black;
111             this.chart1.BorderlineDashStyle =
                  System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.S
                  olid;
112             chartArea1.Name = "ChartArea1";
113             this.chart1.ChartAreas.Add(chartArea1);
114             legend1.Font = new System.Drawing.Font("Microsoft Sans Serif",
                  12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
115             legend1.IsTextAutoFit = false;
116             legend1.Name = "Legend1";
117             this.chart1.Legends.Add(legend1);
118             this.chart1.Location = new System.Drawing.Point(12, 12);
119             this.chart1.Name = "chart1";
120             series1.ChartArea = "ChartArea1";
121             series1.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
122             series1.Legend = "Legend1";
123             series1.Name = "X Axis";
```

```
124             series2.ChartArea = "ChartArea1";
125             series2.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
126             series2.Legend = "Legend1";
127             series2.Name = "Y Axis";
128             series3.ChartArea = "ChartArea1";
129             series3.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
130             series3.Legend = "Legend1";
131             series3.Name = "Z Axis";
132             this.chart1.Series.Add(series1);
133             this.chart1.Series.Add(series2);
134             this.chart1.Series.Add(series3);
135             this.chart1.Size = new System.Drawing.Size(745, 275);
136             this.chart1.TabIndex = 1;
137             title1.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
                   System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
138             title1.Name = "Title1";
139             title1.Text = "Acelerômetro";
140             this.chart1.Titles.Add(title1);
141             //
142             // backgroundWorker1
143             //
144             this.backgroundWorker1.WorkerReportsProgress = true;
145             this.backgroundWorker1.WorkerSupportsCancellation = true;
146             this.backgroundWorker1.DoWork += new
                  System.ComponentModel.DoWorkEventHandler
                  (this.backgroundWorker1_DoWork);
147             //
148             // chart2
149             //
150             this.chart2.Anchor = ((System.Windows.Forms.AnchorStyles)
                  ((System.Windows.Forms.AnchorStyles.Left |
                  System.Windows.Forms.AnchorStyles.Right)));
151             this.chart2.BorderlineColor = System.Drawing.Color.Black;
152             this.chart2.BorderlineDashStyle =
                  System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.S
                  olid;
153             chartArea2.Name = "ChartArea1";
154             this.chart2.ChartAreas.Add(chartArea2);
155             legend2.Font = new System.Drawing.Font("Microsoft Sans Serif",
                  12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
156             legend2.IsTextAutoFit = false;
157             legend2.Name = "Legend1";
158             this.chart2.Legends.Add(legend2);
159             this.chart2.Location = new System.Drawing.Point(12, 293);
160             this.chart2.Name = "chart2";
161             series4.ChartArea = "ChartArea1";
162             series4.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
163             series4.Legend = "Legend1";
```

```
164             series4.Name = "X Axis";
165             series5.ChartArea = "ChartArea1";
166             series5.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
167             series5.Legend = "Legend1";
168             series5.Name = "Y Axis";
169             series6.ChartArea = "ChartArea1";
170             series6.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
171             series6.Legend = "Legend1";
172             series6.Name = "Z Axis";
173             this.chart2.Series.Add(series4);
174             this.chart2.Series.Add(series5);
175             this.chart2.Series.Add(series6);
176             this.chart2.Size = new System.Drawing.Size(745, 275);
177             this.chart2.TabIndex = 1;
178             title2.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
                   System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
179             title2.Name = "Title1";
180             title2.Text = "Giroscópio";
181             this.chart2.Titles.Add(title2);
182             //
183             // label1
184             //
185             this.label1.AutoSize = true;
186             this.label1.Location = new System.Drawing.Point(6, 22);
187             this.label1.Name = "label1";
188             this.label1.Size = new System.Drawing.Size(149, 20);
189             this.label1.TabIndex = 6;
190             this.label1.Text = "Dados de Conexão:";
191             //
192             // groupBox1
193             //
194             this.groupBox1.Controls.Add(this.label1);
195             this.groupBox1.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
196             this.groupBox1.Location = new System.Drawing.Point(10, 3);
197             this.groupBox1.Name = "groupBox1";
198             this.groupBox1.Size = new System.Drawing.Size(202, 70);
199             this.groupBox1.TabIndex = 7;
200             this.groupBox1.TabStop = false;
201             this.groupBox1.Text = "Conexão IMU";
202             //
203             // backgroundWorker2
204             //
205             this.backgroundWorker2.WorkerReportsProgress = true;
206             this.backgroundWorker2.WorkerSupportsCancellation = true;
207             this.backgroundWorker2.DoWork += new
                  System.ComponentModel.DoWorkEventHandler
                  (this.backgroundWorker2_DoWork);
208             //
209             // chart3
```

```
210             //
211             this.chart3.Anchor = ((System.Windows.Forms.AnchorStyles)
                  ((System.Windows.Forms.AnchorStyles.Left |
                  System.Windows.Forms.AnchorStyles.Right)));
212             this.chart3.BorderlineColor = System.Drawing.Color.Black;
213             this.chart3.BorderlineDashStyle =
                  System.Windows.Forms.DataVisualization.Charting.ChartDashStyle.S
                  olid;
214             chartArea3.Name = "ChartArea1";
215             this.chart3.ChartAreas.Add(chartArea3);
216             legend3.Font = new System.Drawing.Font("Microsoft Sans Serif",
                  12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
217             legend3.IsTextAutoFit = false;
218             legend3.Name = "Legend1";
219             this.chart3.Legends.Add(legend3);
220             this.chart3.Location = new System.Drawing.Point(12, 574);
221             this.chart3.Name = "chart3";
222             series7.ChartArea = "ChartArea1";
223             series7.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
224             series7.Legend = "Legend1";
225             series7.Name = "Channel 1";
226             series8.ChartArea = "ChartArea1";
227             series8.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
228             series8.Legend = "Legend1";
229             series8.Name = "Channel 2";
230             series9.ChartArea = "ChartArea1";
231             series9.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
232             series9.Legend = "Legend1";
233             series9.Name = "Channel 3";
234             series10.ChartArea = "ChartArea1";
235             series10.ChartType =
                  System.Windows.Forms.DataVisualization.Charting.SeriesChartType.
                  Spline;
236             series10.Legend = "Legend1";
237             series10.Name = "Channel 4";
238             this.chart3.Series.Add(series7);
239             this.chart3.Series.Add(series8);
240             this.chart3.Series.Add(series9);
241             this.chart3.Series.Add(series10);
242             this.chart3.Size = new System.Drawing.Size(745, 275);
243             this.chart3.TabIndex = 2;
244             title3.Font = new System.Drawing.Font("Microsoft Sans Serif", 12F,
                   System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
245             title3.Name = "Title1";
246             title3.Text = "Eletromiógrafo";
247             this.chart3.Titles.Add(title3);
248             //
249             // groupBox2
```

```
250             //
251             this.groupBox2.Controls.Add(this.comboBox2);
252             this.groupBox2.Controls.Add(this.label5);
253             this.groupBox2.Controls.Add(this.checkBox3);
254             this.groupBox2.Controls.Add(this.checkBox4);
255             this.groupBox2.Controls.Add(this.checkBox2);
256             this.groupBox2.Controls.Add(this.checkBox1);
257             this.groupBox2.Controls.Add(this.comboBox1);
258             this.groupBox2.Controls.Add(this.label2);
259             this.groupBox2.Controls.Add(this.button1);
260             this.groupBox2.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
261             this.groupBox2.Location = new System.Drawing.Point(11, 79);
262             this.groupBox2.Name = "groupBox2";
263             this.groupBox2.Size = new System.Drawing.Size(202, 181);
264             this.groupBox2.TabIndex = 0;
265             this.groupBox2.TabStop = false;
266             this.groupBox2.Text = "Conexão EMG";
267             //
268             // comboBox2
269             //
270             this.comboBox2.DropDownStyle =
                  System.Windows.Forms.ComboBoxStyle.DropDownList;
271             this.comboBox2.FormattingEnabled = true;
272             this.comboBox2.Items.AddRange(new object[] {
273             "G100",
274             "G200",
275             "G400",
276             "G500",
277             "G800",
278             "G1000",
279             "G1600",
280             "G2000"});
281             this.comboBox2.Location = new System.Drawing.Point(103, 59);
282             this.comboBox2.Name = "comboBox2";
283             this.comboBox2.Size = new System.Drawing.Size(93, 28);
284             this.comboBox2.TabIndex = 13;
285             //
286             // label5
287             //
288             this.label5.AutoSize = true;
289             this.label5.Location = new System.Drawing.Point(6, 62);
290             this.label5.Name = "label5";
291             this.label5.Size = new System.Drawing.Size(62, 20);
292             this.label5.TabIndex = 14;
293             this.label5.Text = "Ganho:";
294             //
295             // checkBox3
296             //
297             this.checkBox3.AutoSize = true;
298             this.checkBox3.Location = new System.Drawing.Point(103, 93);
299             this.checkBox3.Name = "checkBox3";
300             this.checkBox3.Size = new System.Drawing.Size(82, 24);
301             this.checkBox3.TabIndex = 3;
302             this.checkBox3.Text = "Canal 3";
```

```
303             this.checkBox3.UseVisualStyleBackColor = true;
304             //
305             // checkBox4
306             //
307             this.checkBox4.AutoSize = true;
308             this.checkBox4.Location = new System.Drawing.Point(103, 123);
309             this.checkBox4.Name = "checkBox4";
310             this.checkBox4.Size = new System.Drawing.Size(82, 24);
311             this.checkBox4.TabIndex = 4;
312             this.checkBox4.Text = "Canal 4";
313             this.checkBox4.UseVisualStyleBackColor = true;
314             //
315             // checkBox2
316             //
317             this.checkBox2.AutoSize = true;
318             this.checkBox2.Location = new System.Drawing.Point(9, 123);
319             this.checkBox2.Name = "checkBox2";
320             this.checkBox2.Size = new System.Drawing.Size(82, 24);
321             this.checkBox2.TabIndex = 2;
322             this.checkBox2.Text = "Canal 2";
323             this.checkBox2.UseVisualStyleBackColor = true;
324             //
325             // checkBox1
326             //
327             this.checkBox1.AutoSize = true;
328             this.checkBox1.Location = new System.Drawing.Point(9, 93);
329             this.checkBox1.Name = "checkBox1";
330             this.checkBox1.Size = new System.Drawing.Size(82, 24);
331             this.checkBox1.TabIndex = 1;
332             this.checkBox1.Text = "Canal 1";
333             this.checkBox1.UseVisualStyleBackColor = true;
334             //
335             // comboBox1
336             //
337             this.comboBox1.DropDownStyle =
                  System.Windows.Forms.ComboBoxStyle.DropDownList;
338             this.comboBox1.FormattingEnabled = true;
339             this.comboBox1.Location = new System.Drawing.Point(103, 25);
340             this.comboBox1.Name = "comboBox1";
341             this.comboBox1.Size = new System.Drawing.Size(93, 28);
342             this.comboBox1.TabIndex = 0;
343             this.comboBox1.SelectedIndexChanged += new System.EventHandler
                  (this.comboBox1_SelectedIndexChanged);
344             //
345             // label2
346             //
347             this.label2.AutoSize = true;
348             this.label2.Location = new System.Drawing.Point(6, 28);
349             this.label2.Name = "label2";
350             this.label2.Size = new System.Drawing.Size(91, 20);
351             this.label2.TabIndex = 12;
352             this.label2.Text = "Porta COM:";
353             //
354             // button1
355             //
356             this.button1.Dock = System.Windows.Forms.DockStyle.Bottom;
```

```
357             this.button1.Enabled = false;
358             this.button1.Location = new System.Drawing.Point(3, 150);
359             this.button1.Name = "button1";
360             this.button1.Size = new System.Drawing.Size(196, 28);
361             this.button1.TabIndex = 1;
362             this.button1.Text = "Conectar MioTool";
363             this.button1.UseVisualStyleBackColor = true;
364             this.button1.Click += new System.EventHandler
                  (this.button1_Click_1);
365             //
366             // pictureBox1
367             //
368             this.pictureBox1.Image = ((System.Drawing.Image)
                  (resources.GetObject("pictureBox1.Image")));
369             this.pictureBox1.Location = new System.Drawing.Point(8, 761);
370             this.pictureBox1.Name = "pictureBox1";
371             this.pictureBox1.Size = new System.Drawing.Size(100, 100);
372             this.pictureBox1.TabIndex = 11;
373             this.pictureBox1.TabStop = false;
374             //
375             // pictureBox2
376             //
377             this.pictureBox2.Image = ((System.Drawing.Image)
                  (resources.GetObject("pictureBox2.Image")));
378             this.pictureBox2.Location = new System.Drawing.Point(114, 761);
379             this.pictureBox2.Name = "pictureBox2";
380             this.pictureBox2.Size = new System.Drawing.Size(100, 100);
381             this.pictureBox2.TabIndex = 12;
382             this.pictureBox2.TabStop = false;
383             //
384             // splitContainer1
385             //
386             this.splitContainer1.Dock = System.Windows.Forms.DockStyle.Fill;
387             this.splitContainer1.FixedPanel =
                  System.Windows.Forms.FixedPanel.Panel2;
388             this.splitContainer1.IsSplitterFixed = true;
389             this.splitContainer1.Location = new System.Drawing.Point(0, 0);
390             this.splitContainer1.Name = "splitContainer1";
391             //
392             // splitContainer1.Panel1
393             //
394             this.splitContainer1.Panel1.Controls.Add(this.chart1);
395             this.splitContainer1.Panel1.Controls.Add(this.chart2);
396             this.splitContainer1.Panel1.Controls.Add(this.chart3);
397             //
398             // splitContainer1.Panel2
399             //
400             this.splitContainer1.Panel2.Controls.Add(this.groupBox6);
401             this.splitContainer1.Panel2.Controls.Add(this.groupBox5);
402             this.splitContainer1.Panel2.Controls.Add(this.groupBox4);
403             this.splitContainer1.Panel2.Controls.Add(this.groupBox3);
404             this.splitContainer1.Panel2.Controls.Add(this.groupBox1);
405             this.splitContainer1.Panel2.Controls.Add(this.pictureBox2);
406             this.splitContainer1.Panel2.Controls.Add(this.groupBox2);
407             this.splitContainer1.Panel2.Controls.Add(this.pictureBox1);
408             this.splitContainer1.Size = new System.Drawing.Size(984, 861);
```

```
409             this.splitContainer1.SplitterDistance = 760;
410             this.splitContainer1.TabIndex = 0;
411             //
412             // groupBox6
413             //
414             this.groupBox6.Controls.Add(this.button4);
415             this.groupBox6.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
416             this.groupBox6.Location = new System.Drawing.Point(10, 453);
417             this.groupBox6.Name = "groupBox6";
418             this.groupBox6.Size = new System.Drawing.Size(202, 70);
419             this.groupBox6.TabIndex = 16;
420             this.groupBox6.TabStop = false;
421             this.groupBox6.Text = "Destacar Dados";
422             //
423             // button4
424             //
425             this.button4.BackColor = System.Drawing.Color.Transparent;
426             this.button4.Dock = System.Windows.Forms.DockStyle.Fill;
427             this.button4.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 21.75F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
428             this.button4.Location = new System.Drawing.Point(3, 22);
429             this.button4.Name = "button4";
430             this.button4.Size = new System.Drawing.Size(196, 45);
431             this.button4.TabIndex = 3;
432             this.button4.Text = "OFF";
433             this.button4.UseVisualStyleBackColor = false;
434             this.button4.Click += new System.EventHandler(this.button4_Click);
435             //
436             // groupBox5
437             //
438             this.groupBox5.Controls.Add(this.button3);
439             this.groupBox5.Controls.Add(this.comboBox3);
440             this.groupBox5.Controls.Add(this.label7);
441             this.groupBox5.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
442             this.groupBox5.Location = new System.Drawing.Point(11, 316);
443             this.groupBox5.Name = "groupBox5";
444             this.groupBox5.Size = new System.Drawing.Size(202, 131);
445             this.groupBox5.TabIndex = 15;
446             this.groupBox5.TabStop = false;
447             this.groupBox5.Text = "Intervalo de Aquisição";
448             //
449             // button3
450             //
451             this.button3.BackColor = System.Drawing.Color.Green;
452             this.button3.Dock = System.Windows.Forms.DockStyle.Bottom;
453             this.button3.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 21.75F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
454             this.button3.Location = new System.Drawing.Point(3, 64);
455             this.button3.Name = "button3";
456             this.button3.Size = new System.Drawing.Size(196, 64);
```

```
457             this.button3.TabIndex = 3;
458             this.button3.Text = "▷ Play";
459             this.button3.UseVisualStyleBackColor = false;
460             this.button3.Click += new System.EventHandler(this.button3_Click);
461             //
462             // comboBox3
463             //
464             this.comboBox3.DropDownStyle =
                  System.Windows.Forms.ComboBoxStyle.DropDownList;
465             this.comboBox3.FormattingEnabled = true;
466             this.comboBox3.Items.AddRange(new object[] {
467             "15s",
468             "30s",
469             "60s",
470             "Manual"});
471             this.comboBox3.Location = new System.Drawing.Point(74, 30);
472             this.comboBox3.Name = "comboBox3";
473             this.comboBox3.Size = new System.Drawing.Size(122, 28);
474             this.comboBox3.TabIndex = 2;
475             //
476             // label7
477             //
478             this.label7.AutoSize = true;
479             this.label7.Location = new System.Drawing.Point(5, 33);
480             this.label7.Name = "label7";
481             this.label7.Size = new System.Drawing.Size(62, 20);
482             this.label7.TabIndex = 1;
483             this.label7.Text = "Tempo:";
484             //
485             // groupBox4
486             //
487             this.groupBox4.Controls.Add(this.label4);
488             this.groupBox4.Controls.Add(this.numericUpDown1);
489             this.groupBox4.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
490             this.groupBox4.Location = new System.Drawing.Point(8, 692);
491             this.groupBox4.Name = "groupBox4";
492             this.groupBox4.Size = new System.Drawing.Size(206, 63);
493             this.groupBox4.TabIndex = 14;
494             this.groupBox4.TabStop = false;
495             this.groupBox4.Text = "Escala Horizontal";
496             //
497             // label4
498             //
499             this.label4.AutoSize = true;
500             this.label4.Location = new System.Drawing.Point(5, 33);
501             this.label4.Name = "label4";
502             this.label4.Size = new System.Drawing.Size(63, 20);
503             this.label4.TabIndex = 1;
504             this.label4.Text = "Pontos:";
505             //
506             // numericUpDown1
507             //
508             this.numericUpDown1.Location = new System.Drawing.Point(74, 31);
509             this.numericUpDown1.Maximum = new decimal(new int[] {
```

```
510             10000,
511             0,
512             0,
513             0});
514             this.numericUpDown1.Minimum = new decimal(new int[] {
515             100,
516             0,
517             0,
518             0});
519             this.numericUpDown1.Name = "numericUpDown1";
520             this.numericUpDown1.Size = new System.Drawing.Size(122, 26);
521             this.numericUpDown1.TabIndex = 0;
522             this.numericUpDown1.Value = new decimal(new int[] {
523             300,
524             0,
525             0,
526             0});
527             this.numericUpDown1.ValueChanged += new System.EventHandler
                  (this.numericUpDown1_ValueChanged);
528             //
529             // groupBox3
530             //
531             this.groupBox3.Controls.Add(this.button2);
532             this.groupBox3.Controls.Add(this.label3);
533             this.groupBox3.Font = new System.Drawing.Font("Microsoft Sans
                  Serif", 12F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
534             this.groupBox3.Location = new System.Drawing.Point(8, 591);
535             this.groupBox3.Name = "groupBox3";
536             this.groupBox3.Size = new System.Drawing.Size(205, 95);
537             this.groupBox3.TabIndex = 1;
538             this.groupBox3.TabStop = false;
539             this.groupBox3.Text = "Exportação dos Dados";
540             //
541             // button2
542             //
543             this.button2.Dock = System.Windows.Forms.DockStyle.Bottom;
544             this.button2.Location = new System.Drawing.Point(3, 64);
545             this.button2.Name = "button2";
546             this.button2.Size = new System.Drawing.Size(199, 28);
547             this.button2.TabIndex = 0;
548             this.button2.Text = "Alterar Diretório";
549             this.button2.UseVisualStyleBackColor = true;
550             this.button2.Click += new System.EventHandler(this.button2_Click);
551             //
552             // label3
553             //
554             this.label3.AutoSize = true;
555             this.label3.Font = new System.Drawing.Font("Microsoft Sans Serif",
                   9.75F, System.Drawing.FontStyle.Regular,
                  System.Drawing.GraphicsUnit.Point, ((byte)(0)));
556             this.label3.Location = new System.Drawing.Point(6, 22);
557             this.label3.Name = "label3";
558             this.label3.Size = new System.Drawing.Size(46, 16);
559             this.label3.TabIndex = 6;
560             this.label3.Text = "Pasta:";
```

```
561             //
562             // toolTip1
563             //
564             this.toolTip1.ToolTipIcon = System.Windows.Forms.ToolTipIcon.Info;
565             this.toolTip1.ToolTipTitle = "Pasta:";
566             //
567             // timer1
568             //
569             this.timer1.Interval = 10;
570             this.timer1.Tick += new System.EventHandler(this.timer1_Tick);
571             //
572             // timer2
573             //
574             this.timer2.Tick += new System.EventHandler(this.timer2_Tick);
575             //
576             // timer3
577             //
578             this.timer3.Tick += new System.EventHandler(this.timer3_Tick);
579             //
580             // Form1
581             //
582             this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
583             this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
584             this.BackColor = System.Drawing.SystemColors.ControlLightLight;
585             this.ClientSize = new System.Drawing.Size(984, 861);
586             this.Controls.Add(this.splitContainer1);
587             this.Icon = ((System.Drawing.Icon)(resources.GetObject
                  ("$this.Icon")));
588             this.MaximizeBox = false;
589             this.MaximumSize = new System.Drawing.Size(3000, 900);
590             this.MinimumSize = new System.Drawing.Size(1000, 900);
591             this.Name = "Form1";
592             this.StartPosition =
                  System.Windows.Forms.FormStartPosition.CenterScreen;
593             this.Text = "SISTEMA SUPERVISÓRIO PARA ESTUDO DE EQUILÍBRIO";
594             this.FormClosing += new
                  System.Windows.Forms.FormClosingEventHandler
                  (this.Form1_FormClosing);
595             this.Load += new System.EventHandler(this.Form1_Load);
596             ((System.ComponentModel.ISupportInitialize)(this.chart1)).EndInit
                  ();
597             ((System.ComponentModel.ISupportInitialize)(this.chart2)).EndInit
                  ();
598             this.groupBox1.ResumeLayout(false);
599             this.groupBox1.PerformLayout();
600             ((System.ComponentModel.ISupportInitialize)(this.chart3)).EndInit
                  ();
601             this.groupBox2.ResumeLayout(false);
602             this.groupBox2.PerformLayout();
603             ((System.ComponentModel.ISupportInitialize)
                  (this.pictureBox1)).EndInit();
604             ((System.ComponentModel.ISupportInitialize)
                  (this.pictureBox2)).EndInit();
605             this.splitContainer1.Panel1.ResumeLayout(false);
606             this.splitContainer1.Panel2.ResumeLayout(false);
607             ((System.ComponentModel.ISupportInitialize)
```

```
                (this.splitContainer1)).EndInit();
            this.splitContainer1.ResumeLayout(false);
            this.groupBox6.ResumeLayout(false);
            this.groupBox5.ResumeLayout(false);
            this.groupBox5.PerformLayout();
            this.groupBox4.ResumeLayout(false);
            this.groupBox4.PerformLayout();
            ((System.ComponentModel.ISupportInitialize)
              (this.numericUpDown1)).EndInit();
            this.groupBox3.ResumeLayout(false);
            this.groupBox3.PerformLayout();
            this.ResumeLayout(false);

        }

        #endregion
        private System.Windows.Forms.DataVisualization.Charting.Chart chart1;
        private System.ComponentModel.BackgroundWorker backgroundWorker1;
        private System.Windows.Forms.DataVisualization.Charting.Chart chart2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.ComponentModel.BackgroundWorker backgroundWorker2;
        private System.Windows.Forms.DataVisualization.Charting.Chart chart3;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.ComboBox comboBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.CheckBox checkBox3;
        private System.Windows.Forms.CheckBox checkBox4;
        private System.Windows.Forms.CheckBox checkBox2;
        private System.Windows.Forms.CheckBox checkBox1;
        private System.Windows.Forms.PictureBox pictureBox1;
        private System.Windows.Forms.PictureBox pictureBox2;
        private System.Windows.Forms.SplitContainer splitContainer1;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.FolderBrowserDialog folderBrowserDialog1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.ToolTip toolTip1;
        private System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.NumericUpDown numericUpDown1;
        private System.Windows.Forms.Timer timer1;
        private System.Windows.Forms.ComboBox comboBox2;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Timer timer2;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.GroupBox groupBox5;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.ComboBox comboBox3;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Timer timer3;
        private System.Windows.Forms.GroupBox groupBox6;
        private System.Windows.Forms.Button button4;
    }
}

```

ANEXO A - PROTOCOLO DE MENSAGENS DO MIOTOOL 400

Instruções para Comunicação com MioTool

Comunicação Serial

Parâmetro	**Valor**
BaudRate	230400
DataBits	8
Parity	None
StopBits	One
Dtr	Disabled
Rts	Disabled

Protocolo de Mensagens

O MioTool troca dados de duas formas:

- A **leitura de propriedades e configuração de parâmetros** é feita através do envio e recebimento de *mensagens*, compostas de pacotes com uma estrutura definida;
- A **transmissão dos dados de captura** é feita através de blocos contíguos de 8 bytes, cada bloco representando um array `UInt16[4]` com as amostras de cada canal.

As *mensagens* possuem a seguinte estrutura:

Start	Command	Payload	Stop	Verificador
1 byte	1 byte	6 bytes	1 byte	1 byte

- No sentido PC -> MioTool, start byte = **200 (0xC8)** e stop byte = **206 (0xCE)**;
- No sentido MioTool -> PC, start byte = **205 (0xCD)** e stop byte = **207 (0xCF)**;
- Somatório dos bytes anteriores (incluindo start e stop), desconsiderando o overflow:

```
byte checksum(byte[] inputBytes)
{
    byte result = 0;
    foreach (var b in inputBytes)
        result += b;
    return result;
}
```

Os principais comandos para operação do Miotool são os seguintes:

Comando	código (decimal)	código (hexadecimal)
Connect	10	0x0A
ReadSerialNumber	20	0x14
ReadMemory	40	0x28
WriteMemory	41	0x29
StartAcquisition	60	0x3C
ConfirmStopAcquisition	62	0x3E
WriteCalibration	71	0x47
ReadCalibration	72	0x48

Funções de Configuração

Connect (10 | 0x0A)

Deve ser enviado pelo PC para habilitar a interpretação de todos os próximos comandos.
Solicitação não possui payload.
Resposta não possui payload.

ReadSerialNumber (20 | 0x14)

Solicitação não possui payload.
Resposta contém quatro bytes de payload, cujo valor está codificado como Int32 *big endian*.

ReadBattery (40 | 0x28)

Solicitação não possui payload.
Resposta contém seis bytes de payload, sendo três valores UInt16 *big endian*, respectivamente “leitura”, “máximo” e “mínimo”, sendo o valor percentual estabelecido pela fórmula `100 * (leitura - mínimo) / (double)(máximo - mínimo)`

Funções de Captura de Dados

StartAcquisition (60 | 0x3C)

Solicitação possui quatro bytes de payload, cada um deles indicando índice do canal e ganho do amplificador, sendo repassado diretamente para o chip ADS7871 do Miotool.
Resposta não possui payload;

Lista de ganhos possíveis: [1, 2, 4, 5, 8, 10, 16, 20]. Para cada byte do payload, preencher da seguinte forma:

bit 7	bits 6, 5, 4	bit 3	bits 2, 1, 0
1*	índice do ganho	1**	índice do canal

O uso de três bits possibilita representar valores (índices) de 0 a 7:
000 = 0; 001 = 1; 010 = 2; 011 = 3; 100 = 4; 101 = 5; 110 = 6; 111 = 7;

* O valor do bit 7 deve ser “1”, indicando captura em modo direto do ADS7871;
** O valor do bit 3 deve ser “1”, indicando modo “single-ended”, conforme montagem do ADS7871 no circuito;

O índice do canal, nos quatro bytes consecutivos do payload, deve ser respectivamente 0, 1, 2 e 3.

ConfirmStopAcquisition (62 | 0x3E)

É enviado para interromper a transmissão de amostras, e obter resposta confirmando sucesso da chamada de função.
Solicitação não tem payload;
Resposta tem payload com um byte booleano indicando sucesso do comando.

Função de Leitura de frames

Imediatamente após o recebimento da resposta do comando *StartAcquisition*, devem ser lidos dois bytes (com valor de zero). Esses bytes são enviados pelo Miotool, e a razão para isso não é conhecida no momento.
Após isso, os valores das amostras são lidos em blocos contíguos de 8 bytes, cada bloco representando um array `UInt16[4]` com as amostras de cada canal. É importante considerar que o chip ADS7871, que faz a conversão AD, tem somente 14 bits de resolução.

Funções de Calibração / Memória

O Miotool possui uma EEPROM de 512 bytes que pode ser lida e gravada livremente (mensagens *ReadMemory* e *WriteMemory*), ou ser usada como uma “tabela de calibrações” (mensagens *WriteCalibration* e *ReadCalibration*).

ReadMemory (40 | 0x28)

Solicitação possui dois bytes de payload, indicando a posição do byte a ser lido, de acordo com a fórmula:

```
posição = payload[0] * 256 + payload[1];
```

Resposta possui dois bytes, sendo o primeiro zero e o segundo o valor do byte lido;

WriteMemory (41 | 0x29)

Solicitação possui quatro bytes de payload, sendo os dois primeiros a posição do byte a ser gravado conforme mesma fórmula da função *ReadMemory*, e o QUARTO o valor do byte a ser gravado.
Resposta possui um byte de payload, correspondendo ao valor recém gravado, lido da memória.

ReadCalibration (72 | 0x48)

Solicitação possui dois bytes de payload, indicando a posição do valor a ser lido, conforme a fórmula:

```
posição = payload[0] * 32 + payload[1] * 4;
```

Resposta possui quatro bytes de payload, correspondendo ao valor Float32 lido.

WriteCalibration (71 | 0x47)

Solicitação possui seis bytes de payload, sendo os dois primeiros a posição do valor a ser gravado conforme mesma fórmula da função *ReadCalibration*, e os outros quatro o valor em Float32 da calibração a ser gravada.
Resposta possui um byte de payload, representando o valor booleano de sucesso da gravação.

REFERÊNCIAS

ARKIPELAGO. *Baropodômetros Eletrônicos*. 2020. Disponível em: <http://www.arkipelago.com.br/produtos/baropodometros.html>.

ARTIGAS, N. R. et al. Evaluation of Knee Proprioception and Factors Related to Parkinson's Disease. *Neuroscience Journal*, Porto Alegre, Online, set. 2016.

ASSOCIATION, A. P. D. *What role does DaTscan have in the diagnosis of Parkinson's disease?* 2019. Disponível em: <https://www.apdaparkinson.org/article/what-is-a-datscan-and-should-i-get-one/>. Acesso em: 29 set. 2020.

BELL, D. D. J. et al. *Parkinson disease*. 2020. Disponível em: <https://radiopaedia.org/articles/parkinson-disease-1>. Acesso em: 21 set. 2020.

BERG, W. P.; STRANG, A. J. *The Role of Electromyography (EMG) in the Study of Anticipatory Postural Adjustments*. Jan. 2012. Disponível em: <https://www.intechopen.com/books/applications-of-emg-in-clinical-and-sports-medicine/the-role-of-electromyography-emg-in-the-study-of-anticipatory-postural-adjustments>. Acesso em: 14 nov. 2020.

CBIE. *What are hydroelectric reservoirs for?* 2019. Disponível em: <https://cbie.com.br/artigos/para-que-servem-os-reservatorios-das-hidreletricas/>. Acesso em: 27 out. 2020.

CHRISTOU, E.; NETO, O. Identification of Oscillations in Muscle Activity From Surface EMG: Reply to Halliday and Farmer. *American Physiological Society*, 2010.

FOUNDATION, P. *Parkinson's Disease Statistics*. 2020. Disponível em: <https://www.parkinson.org/Understanding-Parkinsons/Statistics>. Acesso em: 20 set. 2020.

GRIMES, D. et al. *Canadian guideline for Parkinson disease*. 2019. Disponível em: <https://www.cmaj.ca/content/191/36/E989>. Acesso em: 14 nov. 2020.

GUJARATI, P. *What is Accelerometer and how does it work on smartphones*. 2020. Disponível em: <https://www.techulator.com/resources/8930-How-does-smartphone-accelerometer-work.aspx>. Acesso em: 16 nov. 2020.

INTERNATIONAL, E. *The JSON Data Interchange Syntax*. 2017. Disponível em: <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Acesso em: 15 nov. 2020.

JAN MRÁZEK, M. M. *SensorStreamer*. [S.l.: s.n.], 22 out. 2020. Disponível em: <https://github.com/yaqwsx/SensorStreamer>.

LUCA, C. J. D. *SURFACE ELECTROMYOGRAPHY: DETECTION AND RECORDING*. [S.l.], 2002. p. 10.

MARRAS, C. et al. *Prevalence of Parkinson's disease across North America*. 2018. Disponível em: <https://www.nature.com/articles/s41531-018-0058-0>. Acesso em: 14 nov. 2020.

MENEZES ARRIAL, T. DE. *Manuais do Usuário - Miotool 200 e Miotool 400*. [S.l.]. p. 23.

MORASSO, P. et al. Quiet standing: The Single Inverted Pendulum model is not so bad after all. *PLoS One*, mar. 2019.

NOUREDANESH, M.; TUNG, J. IMU, sEMG, or their cross-correlation and temporal similarities: Which signal features detect lateral compensatory balance reactions more accurately? *Elsevier*, dez. 2019.

PAULA, F. O. DE. *Sensores IMU – uma abordagem completa*. 2015. Disponível em: <http://www2.decom.ufop.br/imobilis/sensores-imu-uma-abordagem-completa-parte-1/>. Acesso em: 27 out. 2020.

ROSARIO, M. B. DEL et al. *Tracking the Evolution of Smartphone Sensing for Monitoring Human Movement*. Jul. 2015. Disponível em: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4570352/>. Acesso em: 14 nov. 2020.

ST. LOUIS, W. U. IN. *Inertial Measurement Unit (IMU) Use*. 2018. Disponível em: <https://classes.engineering.wustl.edu/ese205/core/index.php?title=Inertial_Measurement_Unit_(IMU)_Use>. Acesso em: 16 nov. 2020.

UK, P. F. *Does Parkinson's run in families?* 2020a. Disponível em: <https://www.parkinsons.org.uk/information-and-support/does-parkinsons-run-families>. Acesso em: 22 set. 2020.

UK, P. F. *Types of Parkinsonism*. 2019. Disponível em: <https://www.parkinsons.org.uk/information-and-support/types-parkinsonism>. Acesso em: 22 set. 2020.

UK, P. F. *What causes Parkinson's?* 2020b. Disponível em: <https://www.parkinsons.org.uk/information-and-support/what-causes-parkinsons>. Acesso em: 22 set. 2020.