



Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia Eletrônica

Desenvolvimento de Sistema de Captura de Movimento Humano Utilizando IMU

Autor: Eduardo Sousa Sales Rodrigues
Orientadora: Profa. Dra. Lourdes Mattos Brasil

Brasília, DF

2018



Eduardo Sousa Sales Rodrigues

Desenvolvimento de Sistema de Captura de Movimento Humano Utilizando IMU

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

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientadora: Profa. Dra. Lourdes Mattos Brasil

Coorientador: MSc. Roberto Aguiar Lima

Brasília, DF

2018

Eduardo Sousa Sales Rodrigues

Desenvolvimento de Sistema de Captura de Movimento Humano Utilizando IMU/
Eduardo Sousa Sales Rodrigues. – Brasília, DF, 2018-

67 p. : il.

Orientadora: Profa. Dra. Lourdes Mattos Brasil

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. IMU.2. Prevenção de lesão.I. Profa. Dra. Lourdes Mattos Brasil.II. Universidade de
Brasília.III. Faculdade UnB Gama.IV. Desenvolvimento de Sistema de Captura de Movi-
mento Humano Utilizando IMU

CDU 02:141:005.6

Eduardo Sousa Sales Rodrigues

Desenvolvimento de Sistema de Captura de Movimento Humano Utilizando IMU

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

Trabalho aprovado. Brasília, DF, 10 de dezembro de 2018.

Profa. Dra. Lourdes Mattos Brasil
Orientadora

MSc. Roberto Aguiar Lima
Coorientador

Bel. Ithallo Junior Alves Guimarães
Convidado

Bel. Renata Menezes Lopes
Convidada

Brasília, DF
2018

*"Não vos conformeis com este mundo,
mas transformai-vos pela renovação do vosso espírito,
para que possais discernir qual é a vontade de Deus:
o que é bom, o que Lhe agrada e o que é perfeito."
(Bíblia Sagrada, Romanos 12, 2)*

RESUMO

O objetivo deste trabalho é o desenvolvimento de uma Unidade de Medição Inercial para aquisição de movimentos e posições corporais em humanos, a fim de auxiliar em pesquisas futuras na área de prevenção de lesões musculoesqueléticas. Existem muitos sensores capazes de auxiliar no estudo do movimento do corpo, mas a maioria não são acessíveis ou não conseguem detalhar certos parâmetros úteis, por exemplo, velocidade e aceleração. A prevenção de lesão é muito importante para atletas de alto desempenho, pois os esportistas conseguem permanecer mais tempo treinando e atingem melhores marcas. O MPU6050 é um sensor inercial feito a partir de tecnologia *Micro Electro Mechanical System* (MEMS), que integra em um único *chip*, 3 sensores de aceleração e 3 sensores de velocidade angular. Este sensor comunica-se por protocolo I2C, o que permite a utilização da plataforma Arduino, que possui um microcontrolador programável. Este microcontrolador se comunica serialmente ao computador, por meio de um cabo *Universal Serial Bus* (USB). A linguagem de programação *Python* pode se comunicar com o Arduino por meio de um protocolo de comunicação serial e assim possibilitar que os dados adquiridos pelo sensor MPU6050 fossem armazenados no computador. Os dados podem ser manipulados para obtenção de diversas informações: aceleração linear, velocidade angular, posições e ângulos. Então, foi feito um protótipo funcional com uma *interface* simples para utilização dos pesquisadores do laboratório LIS da UnB/FGA, necessitando de futuros testes com movimentação em humanos para verificar a precisão e acurácia do IMU.

Palavras-chaves: IMU, MPU6050, Prevenção de Lesão Musculoesquelética.

ABSTRACT

The objective of this work is the development of an Inertial Measurement Unit (IMU) for the acquisition of human body movements and positions to assist research in the area of injuries musculoskeletal prevention. There are many sensors capable of assisting in the study of body movement, but most of them are not accessible or do not detail certain useful parameters(e.g. speed and acceleration). Injury prevention is very important for high-performance athletes, because of preventing injuries, athletes can stay longer in training and achieve better marks. The MPU6050 is an inertial sensor made with Micro Electro Mechanical System (MEMS) technology, integrating on a single chip, 3 acceleration sensors and 3 angular speed sensors. This sensor communicates by I2C protocol, which allows the use of the Arduino platform, which has a programmable micro-controller. This micro-controller communicates serially to the computer via a Universal Serial Bus (USB) cable. The Python programming language can communicate with the Arduino through a serial communication protocol and thus allow the data acquired by the MPU6050 sensor to be stored on the computer. The data can be manipulated to obtain various information (e.g. linear acceleration, angular velocity, positions and angles). So, a functional prototype was done with a simple interface for the use of the researchers of the laboratory LIS of the UnB/FGA, needing of future tests to verify the accuracy and precision of the IMU.

Key-words: IMU, MPU6050, Injury Musculoskeletal Prevention.

LISTA DE ILUSTRAÇÕES

Figura 1 – Ilustração esquemática de componentes MEMS	26
Figura 2 – Um motor MEMS ao lado de uma fio de cabelo humano.	26
Figura 3 – Eixos de orientação.	27
Figura 4 – Sensor inercial baseado em 2 sensores	27
Figura 5 – Princípio de funcionamento do acelerômetro.	29
Figura 6 – Imagem de microscopia eletrônica por varredura de um giroscópio diapasão.	29
Figura 7 – Arduino Nano	33
Figura 8 – WeMos D1 - R2	34
Figura 9 – Esquemático de conexão do MPU6050 com Arduíno Nano	39
Figura 10 – Transmissão de dados do MPU6050 para Arduíno	39
Figura 11 – Transferência de dados com o protocolo I2C.	40
Figura 12 – Fluxograma do <i>software</i> de leitura dos dados.	41
Figura 13 – Fluxograma do <i>software</i> que salva os dados lidos.	43
Figura 14 – Demonstração da posição do sensor	44
Figura 15 – Demonstração direção de giro do sensor	44
Figura 16 – Protótipo na <i>Protoboard</i>	45
Figura 17 – MPU6050 na <i>Protoboard</i>	46
Figura 18 – Segundo Protótipo com o Velcro	46
Figura 19 – Protótipo Fixado ao Punho e a um Cubo	47
Figura 20 – Tela Inicial	47
Figura 21 – Lista com as escalas	48
Figura 22 – Programa mostrando os resultados	48
Figura 23 – Gráficos com os 3 Eixos alinhados com a Normal.	49
Figura 24 – Gráficos para Teste do Giroscópio.	49

LISTA DE TABELAS

Tabela 1 – Tabela de comparação dos modelos de Arduíno	32
Tabela 2 – Conexões entre Arduíno e MPU6050	38
Tabela 3 – História de Usuário	42

LISTA DE ABREVIATURAS E SIGLAS

ADC	<i>Analog Digital Converter</i> - Conversor Analógico-Digital
DC	<i>Direct Current</i> - Corrente Contínua
DMP	<i>Digital Motion Processor</i> - Processador Digital de Movimento
EMG	Eletromiografia
FGA	Faculdade Gama
HARM	<i>High-Aspect-Ratio Micro-machining</i> - Micro Maquinação de Alta Proporção
I2C	<i>Inter-Integrated Circuit</i> - Circuito Inter-Integrado
IC	<i>Integrated Circuit</i> - Circuito Integrado
IDE	<i>Integrated Development Environment</i> - Ambiente de Desenvolvimento Integrado
IMU	<i>Inertial Measurement Unit</i> - Unidade de Medição Inercial
LIS	Laboratório de Informática em Saúde
MEMS	<i>Micro Electro Mechanical System</i> - Sistema Micro-eletromecânico
MOCAP	<i>Motion Capture</i> - Captura de Movimento
MST	<i>Microsystems Technology</i> - Tecnologia de Microssistemas
PDIP	<i>Plastic Dual-In-line Package</i> - Pacote de Dupla Linha de Plástico
PQFP	<i>Plastic Quad Flat Pack</i> - Pacote de Encapsulamento Quadrado de Plástico
SCL	<i>Serial Clock</i> - "Relógio" Serial
SDA	<i>Serial Data</i> - Dados Seriais
UnB	Universidade de Brasília
USB	<i>Universal Serial Bus</i> - Barramento Serial Universal

LISTA DE SÍMBOLOS

ω Velocidade angular em rad/s

\oslash Diâmetro

\forall Para todo

π Constante pi

SUMÁRIO

1	INTRODUÇÃO	21
1.1	Objetivos	22
1.1.1	Objetivo Geral	22
1.1.2	Objetivos Específicos	22
1.2	Justificativa	22
2	REFERENCIAL TEÓRICO	23
2.1	Prevenção de Lesões	23
2.2	O processo de amostragem	24
2.3	MEMS	25
2.4	IMU	26
2.4.1	Acelerômetro	28
2.4.2	Giroscópio	28
2.4.3	MPU6050	30
2.4.4	Arduíno	31
2.4.5	Esp8266	33
2.5	Python	34
2.6	Protocolo HTTP	35
3	METODOLOGIA	37
3.1	Projeto de Hardware	37
3.1.1	Materiais Utilizados	37
3.1.2	Métodos	38
3.2	Projeto de Software	39
3.2.1	Software para o Arduíno	40
3.2.2	Software para Salvar os Dados	40
3.3	Validação	42
4	RESULTADOS E DISCUSSÃO	45
4.1	Resultados	45
4.2	Discussão	48
5	CONCLUSÃO	51
	REFERÊNCIAS	53

ANEXOS	57
ANEXO A – CÓDIGO FONTE DO MICROCONTROLADOR	59
ANEXO B – CÓDIGO DO PROGRAMA EM PYTHON	63

1 INTRODUÇÃO

O interesse em analisar por meio de conceitos físicos o movimento humano é bastante antigo. Estudos clássicos como os de Aristóteles deixam claro que a relevância de analisar o movimento por meio de ensaio físico data do século III a.C. Porém, mesmo com o estudo do movimento sendo antigo, a Biomecânica se consolidou como uma ciência e disciplina acadêmica no Brasil muito recentemente. Historicamente falando, apenas na década de 60 com influência do governo da República Federal da Alemanha que apoiou algumas universidades brasileiras a introduzir a Biomecânica nos cursos de Educação Física (ACQUESTA et al., 2008).

É essencial, para associar medidas físicas ao movimento, estudos relacionados à cinética e cinemática do movimento precisos. Com os dados adquiridos é possível realizar diagnóstico de desempenho em atletas, pesquisas para prevenção de lesão musculoesquelético e adquirir melhor entendimento do movimento humano como um todo (MCGINNIS, 2013).

Dessa forma, o estudo da biomecânica do movimento se tornou importante para desenvolver atletas em diversos esportes. No caso de um velocista, a coleta e a análise de dados sobre a velocidade, aceleração, postura e qualidade de movimento são úteis para possibilitar a evolução de seu desempenho (OKAZAKI et al., 2012).

Para realizar a coleta de dados que possibilitem aos profissionais da área de esportes, procurar por diferentes estratégia e métodos de treinamento para obtenção de melhores resultados no desempenho dos atletas, são necessários equipamentos adequados, os quais estão sendo desenvolvidos (OKAZAKI et al., 2012).

As unidades de medição inercias (IMU - *Inertial Measurement Units*), são um exemplo desses equipamentos, e tornaram-se ferramentas muito úteis para aquisição de informações relacionadas ao movimento corporal. Esses sensores são baratos, pequenos e permitem mobilidade quando integrados com módulos de comunicação sem fio. Porém, sua utilização necessita de um conhecimento técnico e matemático específico e complexo (OBERLANDER, 2015).

A evolução desses sensores permitiu que a movimentação corporal humana fosse estudada em diversos ambientes e situações, sendo necessário apenas algumas unidades de sensores colocados nos pontos adequados do corpo. E, assim, os dados adquiridos são capazes de proporcionar algumas variáveis importantes como: aceleração, velocidade angular, velocidade linear, altura, direção e ângulos, todos de forma não invasiva e sem a necessidade de ambientes fechados (CHANG; GEORGY; EL-SHEIMY, 2016). A partir desses dados é possível fazer análises preditivas, diagnósticos de lesões, movimentos assíncronos de membros corporais, má postura entre outros.

1.1 OBJETIVOS

Os principais objetivos desse trabalho de conclusão de curso são os seguintes:

1.1.1 OBJETIVO GERAL

Desenvolvimento de um IMU para aquisição de movimentos e posições corporais em humanos a fim de auxiliar em pesquisas na área de prevenção de lesões musculoesqueléticas.

1.1.2 OBJETIVOS ESPECÍFICOS

- Identificar quais serão parâmetros que o IMU deve ser capaz de calcular;
- Desenvolver e realizar testes para verificar os resultados obtidos no IMU;
- Desenvolver um *case* para que o IMU possa ser colocado em diversas parte do corpo humano;
- Desenvolver Kit's para utilização em pesquisas futuras do Laboratório de Informática em Saúde (LIS) da Universidade de Brasília (UnB), Faculdade Gama (FGA).

1.2 JUSTIFICATIVA

O sensor IMU tem inúmeras aplicações na área da saúde e dos esportes como avaliação de postura, assimetria de movimentos, análise de marcha, movimentos específicos de esportes entre outras. É um sensor essencial para pesquisas nessas áreas. E são poucos os trabalhos que tentam tornar esse sensor mais amigável para utilização por pesquisadores de diversas áreas que não tem o conhecimento matemático e de linguagens de programação necessários para trabalhar com ele (OBERLANDER, 2015)(CHANG; GEORGY; EL-SHEIMY, 2016).

Além disso, este sensor, exige um custo financeiro menor do que outros equipamentos utilizados nessa área de estudo, como o *Motion Capture* (MOCAP), por exemplo, é um equipamento que exige um ambiente interno com espaço específico, muitas câmeras para melhor precisão e sistemas que normalmente tem um custo financeiro alto (CHANG; GEORGY; EL-SHEIMY, 2016).

Esse sensor deverá futuramente ser utilizado para pesquisas no laboratório LIS da UnB e auxiliar estudantes de graduação e mestrado em seus trabalhos.

2 REFERENCIAL TEÓRICO

Nesta seção do artigo serão explicados conceitos importantes para a realização do projeto, especificação de alguns materiais e a motivação para a escolha dos mesmos.

2.1 PREVENÇÃO DE LESÕES

O desempenho máximo de atletas pode ser afetado por lesões musculoesqueléticas, fazendo com que seus resultados não sejam os melhores, impedindo de participar em competições ou causando a saída do esporte precocemente. O estudo do corpo humano permite obter uma análise operacional dos atletas, ou seja, demonstrar diferentes parâmetros, tais como do sistema proprioceptivo, da estabilidade articular e da força muscular durante movimentos específicos, a mecânica dos tecidos biológicos utilizados no esporte, bem como do estudo morfológico dos atletas. Podendo assim promover a prevenção de lesões (MIZIARA, 2014).

O esforço físico feito pelo atleta durante práticas esportivas lhes proporcionam diversas consequências fisiológicas. O ramo do conhecimento dedicado ao estudo de efeitos fisiológicos agudos e crônicos dos exercícios físicos sobre os diversos sistemas corporais é a Fisiologia do Exercício. Contudo, os métodos de medição ou determinação destes estados do corpo humano em pleno emprego do seu esforço são importantes para diagnosticar e prevenir lesões, fortalecimento de musculatura e tendões, ou seja, todo o desempenho do atleta (ROCHA, 2005).

Estes métodos utilizados analisam algumas características importantes, tais como velocidade, aceleração, posicionamento no espaço e necessitam, de certa maneira, mais do que um profissional com olhar treinado para uma análise real do movimento do corpo humano, dado que a prevenção de lesões e o desempenho dos atletas estão ligados à biomecânica esportiva deles (AMADIO, 2000).

Dessa forma, as avaliações biomecânicas devem ir além do olhar clínico ao analisarem os parâmetros associados a lesões de atletas e estruturarem seus argumentos em sistemas que possam ser capazes de traduzir estes parâmetros (MIZIARA, 2014). Os métodos utilizados para estudar as diversas formas de movimento são: eletromiografia, antropometria, dinamometria, cinemática (AMADIO; SERRÃO, 2007).

A eletromiografia é a medição da atividade elétrica de músculos que podem ser relacionados durante a prática esportiva. A antropometria estabelece as propriedades físicas do corpo humano estudado como as medidas geométricas. A dinamometria mede parâmetros como força e distribuição de pressão com equipamentos específicos para cada atividade física. E, por fim, a cinemática utiliza parâmetros como posição e orientação de partes do corpo humano, que é o tipo de medição tratado neste trabalho (AMADIO; SERRÃO, 2007)(MEDEIROS, 2012).

2.2 O PROCESSO DE AMOSTRAGEM

A cinemetria pode ser realizada através da captura por imagens ou da medição de determinados pontos com sensores inerciais. E para os dois métodos é preciso saber qual a frequência de amostragem, no caso das imagens, o número de frames por segundo, necessários para captar o movimento com precisão suficiente para afirmar que ele representa o deslocamento real (AMADIO; SERRÃO, 2007).

Para definição dessa frequência existe o teorema de amostragem Nyquist-Shannon, que é a base para determinar a frequência mínima de amostragem de um sinal. Através do processo de amostragem, um sinal contínuo no tempo, é modificado para um sinal discreto no tempo. O teorema da amostragem assegura que as amostras discretas uniformemente espaçadas são uma representação completa do sinal, se sua largura de banda é menor do que a metade da taxa de amostragem (MADEIRO; LOPES, 2011).

Para formalizar o conceito, seja $x(t)$ um sinal contínuo no tempo e $X(f)$ sua transformada de Fourier (ou seja, representação do sinal em frequência):

$$X(f) = \int_{-\infty}^{\infty} x(t)e^{-i2\pi ft} dt \quad (2.1)$$

O sinal $x(t)$ é limitado em banda, B , se:

$$X(f) = 0 \quad \forall |f| > B \quad (2.2)$$

A condição suficiente para uma exata reconstrução a partir das amostras em uma taxa de amostragem uniforme f_s (em amostras por unidade de tempo) é:

$$f_s > 2B \quad (2.3)$$

$2B$ é a chamada de Taxa de Nyquist e é uma propriedade do sinal limitado em banda, e $f_s/2$ é chamado de Frequência de Nyquist e é um propriedade desse sistema de amostragem.

O tempo T entre as sucessivas amostras é o intervalo de amostragem.

$$T = \frac{1}{f_s} \quad (2.4)$$

O teorema da amostragem garante que é possível a reconstrução exata do sinal $x(t)$ original desde que sejam respeitadas as condições iniciais (OPPENHEIM; WILLSKY; NAWAB, 2010).

A velocidade do movimento humano varia muito dependendo do local analisado. Pode ser a velocidade do corpo inteiro, ou de órgãos separadamente. Esta velocidade pode ser medida em frequência e varia entre 10 e 160 Hz (SONG; GODOY, 2016).

Logo a Taxa de Nyquist para amostragem de sinais de movimento humano é de 320 Hz. E o sensor para medir os movimentos deve ter uma frequência de amostragem maior que essa (SONG; GODOY, 2016).

2.3 MEMS

MEMS (*Micro Electro Mechanical Systems* - Sistema Micro-eletromecânicos) é uma tecnologia de processamento usada para criar dispositivos integrados ou sistemas que combinam componentes mecânicos e elétricos. Eles são fabricados usando técnicas de processamento em lote de circuitos integrados e podem variar em tamanho entre alguns micrômetros para milímetros. Esses dispositivos (ou sistemas) têm a capacidade de detectar, controlar e atuar na escala micro e gerar efeitos em escala macro (PRIME, 2002).

O termo MEMS, é um acrônimo originado nos Estados Unidos. Também é conhecido como *Microsystems Technology* (MST) na Europa e *Micromachines* no Japão. Independentemente da terminologia, o fator que define um dispositivo MEMS está na maneira como é feito. Enquanto os aparelhos eletrônicos são fabricados usando tecnologia IC (*Integrated Circuit* - *chip* de computador), os componentes micromecânicos são fabricados por sofisticadas manipulações com silício e outros substratos usando processos de micro-usinagem. Processos como micro-usinagem a granel e de superfície, bem como micro maquinção de alta proporção (HARM - *High Aspect Ratio Micromachining*) remove seletivamente partes do silício ou adiciona camadas estruturais adicionais para formar os componentes mecânicos e eletromecânicos. Enquanto circuitos integrados são projetados para explorar as propriedades elétricas do silício, o MEMS aproveita as propriedades mecânicas do silício ou suas propriedades elétricas e mecânicas (PRIME, 2002).

Na forma mais geral, os MEMS consistem em microestruturas mecânicas, microssensores, microatuadores e microeletrônica, todos integrados no mesmo *chip* de silício (Figura 1). Microsensores detectam mudanças no ambiente do sistema medindo informações ou fenômenos mecânicos, térmicos, magnéticos, químicos ou eletromagnéticos. Microeletrônica processa essa informação e sinaliza aos microatuadores para reagirem e criarem alguma forma de mudanças no meio ambiente (PRIME, 2002).

Os dispositivos MEMS são muito pequenos. Seus componentes são geralmente microscópicos (Figura 2). Alavancas, engrenagens, pistões, motores e até motores a vapor foram todos fabricados por MEMS. No entanto, MEMS não se refere apenas à miniaturização de componentes mecânicos ou à fabricação de coisas a partir do silício (na verdade, o termo MEMS é enganoso, já que muitos dispositivos micro maquinados não são mecânicos em nenhum sentido). MEMS é uma tecnologia de fabricação, um paradigma para projetar e criar dispositivos e sistemas mecânicos complexos, bem como sua eletrônica integrada, utilizando técnicas de fabricação em lote (PRIME, 2002).

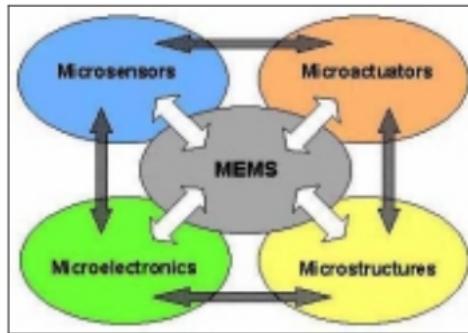


Figura 1 – Ilustração esquemática de componentes MEMS

Fonte: (PRIME, 2002)

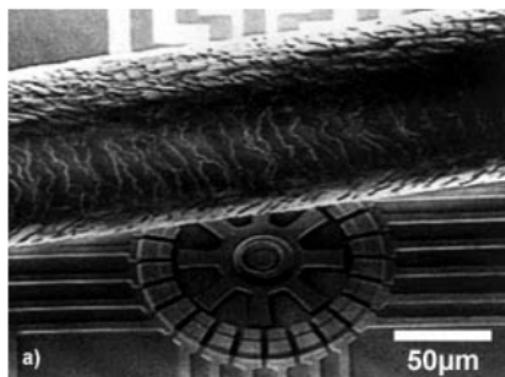


Figura 2 – Um motor MEMS ao lado de uma fio de cabelo humano.

Fonte: (PRIME, 2002)

2.4 IMU

A história do sensor IMU começou em 1930, quando foi usado para auxiliar a navegação de aeronaves e outros dispositivos de grande porte. Por causa de suas restrições, principalmente em tamanho, custo e consumo de energia, o uso do IMU naquele momento era restrito a aplicações em dispositivos grandes e, portanto, impopular para equipamentos de tamanho menor e em grande escala de consumo. Porém, recentemente, o sensor IMU feito por MEMS, foi introduzido com uma característica muito atraente de baixo custo, com poder de processamento e baixo custo. A demanda aumentou muito e as áreas de aplicação também. Atualmente, muitos fabricantes estão competindo nos melhores projetos de IMU, como *Invensense*, *Honeywell*, *STMicroelectronics*, *Microstrain* e *X-Sens* (AHMAD; GHAZILLA; KHAIRI, 2013).

Os sensores de IMU podem ser capazes de medir diversas variáveis. Os mais comuns são feitos para medir 6 graus de liberdade. São 3 medidas de aceleração linear a partir de um acelerômetro e 3 medidas de aceleração angular feitas por um giroscópio (Figura 3) (SANTOS; VIEIRA; JUNIOR, 2016).

A vantagem de usar este tipo de IMU é que não será interferido pelo campo magnético externo em torno do sensor quando é usado muito próximo de material ferromagnético. Por

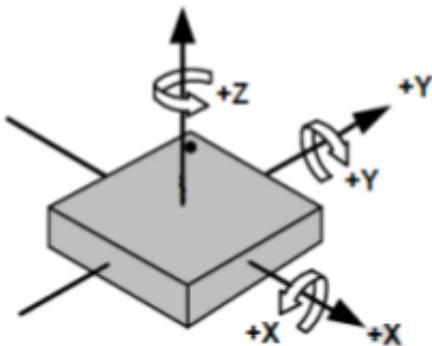


Figura 3 – Eixos de orientação.

Fonte adaptada de (INVENSENSE, 2013)

outro lado, dependendo do acelerômetro e do giroscópio, pode não ser suficiente para aumentar a precisão da medição devido ao ruído dos sensores e à questão do desvio do giroscópio. Os dados adquiridos pelo IMU são integrados como mostrado na Figura 4 (AHMAD; GHAZILLA; KHAIRI, 2013).

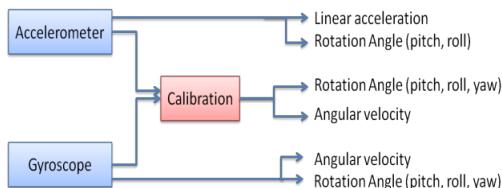


Figura 4 – Sensor inercial baseado em 2 sensores

Fonte: (AHMAD; GHAZILLA; KHAIRI, 2013)

E com essas variáveis adquiridas, um padrão de movimento pode ser traçado, bem como podem ser feitas análises sobre o movimento humano. Assim, junto com a EMG (Eletromiografia), o uso de sensores inercias tem sido mais solicitado na área de monitoramento de práticas esportivas (HOWARD, 2016).

Porém, a utilização de IMU's com equipamentos de comunicação sem fio, têm mostrado uma certa vantagem sobre o uso de sensores com EMG. Isso ocorre por conta de a integração entre acelerômetros, giroscópios e dispositivos de transmissão sem fio ser mais simplificada do que a aquisição de dados por eletromiôgrafos sem fio (HOWARD, 2016).

Os testes que foram feitos com uso de sensores iniciais, segundo Howard (2016), foram capazes de produzir dados relacionados com fadiga muscular, *performance*, postura e velocidade. O que possibilita melhores técnicas de treinamento e até mesmo prevenção de lesão pode ser realizada.

2.4.1 ACELERÔMETRO

O que sensores de movimento são feitos para aferir uma taxa de variação de posição, ou seja, o deslocamento que estiver ocorrendo. Assim, se a posição, $x(t)$, de um corpo varia ao longo tempo, então será obtida sua velocidade, $v(t)$, derivando essa mudança de posição ao longo do tempo. E ao derivar a variação de velocidade ao longo do tempo será calculada a aceleração, $a(t)$, do corpo (NUSSENZVEIG, 2013).

$$v(t) = \frac{dx(t)}{dt} \quad (2.5)$$

$$a(t) = \frac{dv(t)}{dt} = \frac{d^2x(t)}{dt^2} \quad (2.6)$$

Um acelerômetro é capaz de medir a aceleração, e a partir desse dado é possível obter também a velocidade e a posição. Basta fazer a operação inversa da derivada (NUSSENZVEIG, 2013).

$$v(t) = v(0) + \int_0^t (a(t)dt) \quad (2.7)$$

$$x(t) = x(0) + \int_0^t (v(t)dt) \quad (2.8)$$

Acelerômetros podem ser utilizados em muitas áreas, como na automotiva com *air bags*, na navegação, no monitoramento de máquinas, na saúde, nos jogos e em outras. São diversos os processos físicos utilizados para desenvolver um sensor para medir a aceleração. Em aplicações que envolvem voo, aviões e satélites, acelerômetros são baseados em propriedades de massas rotativas. Na indústria, porém, o projeto mais comuns são feitos a partir da combinação da Lei de Newton de aceleração de massa e a Lei de Hooke de ação de mola (CARNEIRO, 2003).

Existem os que funcionam a partir do efeito de cristais piezoelétricos, que é o princípio no qual alguns cristais geram uma corrente elétrica como resposta a uma pressão mecânica. Então, o sensor de acelerômetro funciona como na Figura 5 demonstra, como uma pequena caixa com uma esfera dentro, e as paredes são os cristais piezoelétricos. Sempre que a caixa é alterada de posição a esfera é forçada a se mover em direção de uma das paredes devido a força da gravidade ou de qualquer outra força de aceleração em outro sentido que não a normal. Cada par de paredes paralelos correspondem a um eixo no espaço. Assim, a partir da corrente gerada em cada parede é possível determinar a direção do movimento acelerado e sua magnitude (SANJEEV, 2018).

2.4.2 GIROSCÓPIO

Os giroscópios também são sensores de movimento, porém eles não medem aceleração linear como os acelerômetros, eles disponibilizam ao usuário a velocidade angular de um objeto

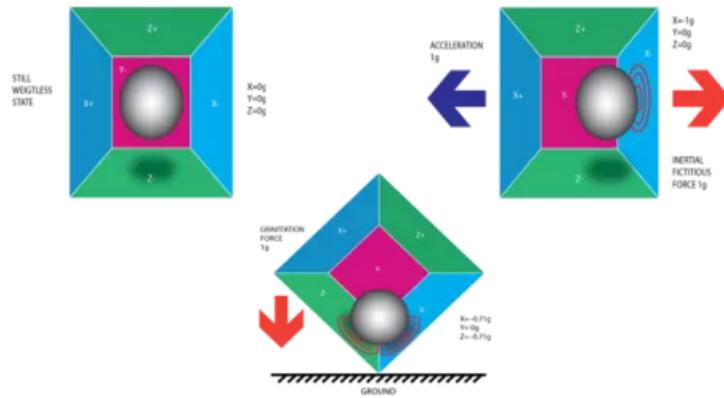


Figura 5 – Princípio de funcionamento do acelerômetro.

Fonte adaptada de (SANJEEV, 2018)

em torno de um eixo. Em geral, os giroscópios feitos com tecnologia MEMS utilizam do efeito Coriolis, no qual um objeto que se encontra em um movimento de rotação, imprime na massa um movimento ortogonal a direção de rotação. Os giroscópios têm um princípio muito mais complexo que os acelerômetros. Este é um dos motivos fizeram eles demorarem mais a parecer como dispositivos MEMS (ALMEIDA, 2014).

O giroscópio do tipo diapasão, é constituído por duas massas paralelas que oscilam com a mesma amplitude, direção igual e sentidos opostos mostrado na Figura 6.

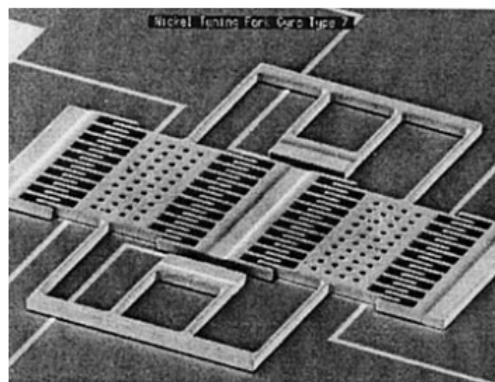


Figura 6 – Imagem de microscopia eletrônica por varredura de um giroscópio diapasão.

Fonte: (FORHAN, 2010)

Ao acontecer uma rotação, a força de Coriolis gera uma vibração ortogonal ao sentido do movimento de cada massa. A amplitude desse movimento pode ser medido de forma capacitiva. A utilização de dois giroscópios na mesma direção e em sentidos opostos aumenta a precisão da medição. Com a medida da força de Coriolis F_c , a medida da massas paralelas m , se movendo a um determinada velocidade v , com relação a um referencial fixo que possua uma velocidade angular ω é possível obter a aceleração de Coriolis a_c (FORHAN, 2010).

$$F_c = 2mv\omega \quad (2.9)$$

$$a_c = 2v\omega \quad (2.10)$$

Com aceleração de Coriolis, que é obtido pelo sensor, a unidade de processamento do sensor é capaz de calcular a velocidade angular em °/s. Com a velocidade angular em relação a cada eixo do espaço tridimensional sendo obtida, é possível encontrar posição angular do sensor e do objeto, ou corpo, em que ele estiver acoplado (FORHAN, 2010)(NUSSENZVEIG, 2013).

A unidade de medida padrão de velocidade angular é rad/s. Então, com o intuito de converter o valor obtido para o padrão internacional, tem-se que:

$$1\pi rad = 180^\circ \quad (2.11)$$

$$1\pi rad/s = 180^\circ/s \quad (2.12)$$

$$\frac{\pi}{180} rad/s = (\frac{180}{180})^\circ/s \quad (2.13)$$

$$\frac{\pi}{180} rad/s = 1^\circ/s \quad (2.14)$$

2.4.3 MPU6050

O MPU6050, da série MPU60X0, foi a primeira interface de movimento a integrar 6 eixos de leitura em um dispositivo único. Esse sensor de movimento, integra 3 eixos de um acelerômetro e 3 eixos de um giroscópio e um DMP (*Digital Motion Processor* - Processador Digital de Movimento) tudo em um pequeno componente medindo 4x4x0.9mm. As suas principais qualidades são seu tamanho reduzido, o baixo consumo de energia, a alta precisão e confiabilidade, a alta tolerância a choques mecânicos, tem seu desempenho programável para aplicações específicas e ainda um baixo custo financeiro (INVENSENSE, 2013).

O protocolo de comunicação utilizado pelo MPU6050 é o I2C (*Inter-Integrated Circuit* - Circuito Inter-Integrado). Que é um protocolo de barramento, e com os mesmos dois fios podem ser conectados vários dispositivos, um sendo o *master* e os outros como *slave*. Esta é uma característica boa para reduzir a quantidade de pinos necessários para conexão de mais dispositivos no microcontrolador (INVENSENSE, 2013).

As aplicações sugeridas pelo fabricante para o MPU6050 são diversas. Abaixo estão listadas algumas:

- Tecnologia *BlurFree^{TM1}*;
- Controles para jogos baseados em movimento;
- Sensores em roupas *TouchAnywhere* para aplicação na saúde e esportes;
- Tecnologia *MotionCommand^{TM2}*;
- Em brinquedos

Características do MPU6050

- O giroscópio MEMS de 3 eixos do MPU6050, possui as seguintes características segundo (INVENSENSE, 2013):
 - Saídas digitais com os valores de velocidade angular para os eixos X, Y e Z com escalas programáveis entre $\pm 250, \pm 500, \pm 1000$ e ± 2000 °/seg;
 - Conversores ADCs (*Analog Digital Converter* - Conversor Analógico Digital) de 16 bits integrados permitem amostragem simultânea do giroscópio
 - Corrente de operação: 3.6mA;
 - Bom desempenho com ruído de baixa frequência;
 - Filtro passa-baixa programável;
 - Fator de escala de sensibilidade calibrado de fábrica.
- O acelerômetro MEMS de 3 eixos do MPU6050, possui as seguintes características segundo (INVENSENSE, 2013):
 - Saídas digitais dos 3 eixos do acelerômetro com escalas programáveis entre $\pm 2g, \pm 4g, \pm 8g$ e $\pm 16g$. Sendo ‘g’ uma constante que equivale a aceleração da gravidade³;
 - Conversores ADCs de 16 bits integrados permitem amostragem simultânea do acelerômetro sem a necessidade de um multiplexador;
 - Orientação, detecção e sinalização.

2.4.4 ARDUÍNO

O Arduíno é uma plataforma de desenvolvimento de código aberto que contém elementos de *hardware* e *software* especificamente adaptados à simplicidade de uso, permitindo uma rápida prototipagem de projetos. Placas Arduíno têm grande potencial de desenvolvimento e uma ampla gama de recursos para uso em conjunto com sensores, transceptores de dados e atuadores, os quais são de grande interesse para este projeto (SMITH, 2016).

¹ Para estabilização de imagens e vídeos.

² Para comandos de movimento curtos.

³ aproximadamente $9,81m/s^2$.

Uma das maiores vantagens do Arduíno sobre outras plataformas de prototipagem com microcontroladores é a facilidade para utilizá-la, que permite que pessoas que não necessariamente são da área de tecnologia, possam aprender rapidamente o básico e criar seus próprios projetos. E existe um grande compartilhamento de projetos de forma livre utilizando essa plataforma. Isso é relevante para esse projeto em específico, pois ele está sendo desenvolvido para utilização de pesquisadores de diversas áreas do conhecimento, os quais poderão aprender a adaptar o projeto para suas necessidades e demandas futuras (ROBERTS, 2011).

Existem diversos modelos de placas Arduíno no mercado, sendo as mais utilizadas LilyPad, Uno, Mini Pro, Mega e Nano. Cada uma dessas placas possuem suas particularidades em relação ao número de recursos, formato físico, memória e capacidade de processamento. Na Tabela 1 estão listados alguns modelos e suas características (SMITH, 2016).

Tabela 1 – Tabela de comparação dos modelos de Arduíno

Modelos	Lilypad	Uno	Pro	Mega	Nano
Microcontrolador	ATmega32u4	ATmega328P	ATmega238	ATmega2560	ATmega328P
Tensão de Operação	3.3V	5V	3.3/5V	5V	5V
Tensão de Entrada	3.8-5V	7-12V	5-12V	7-12V	5-12V
Pinos Digitais	9	14	14	54	14
Entradas Analógicas	4	6	6	16	8
Memória Flash	32kB	32kB	32kB	256kB	32kB
Velocidade do Clock	8MHz	16MHz	8/16MHz	16MHz	16MHz
Dimensões Físicas	50mmØ	68.6 x 53.4mm	52 x 53mm	101.5x53.5mm	18,5 x 43,2mm
Custo Aproximado	R\$ 22,00	R\$ 54,00	R\$ 50,00	R\$ 80,00	R\$ 24,00

Fonte adaptada: (SMITH, 2016)

O Arduíno Nano (Figura 7) tem funcionalidades muito semelhantes às do Arduíno Duemilanove, mas com um formato diferente. O Nano possui o microcontrolador ATmega328P embutido nele, o mesmo que o Arduíno UNO. A principal diferença entre eles é que a placa UNO é apresentada em formato PDIP (*Plastic Dual-In-line Package*) com 30 pinos e Nano está disponível em PQFP(*Plastic Quad Flat Pack*) com 32 pinos. Os 2 pinos extras do Arduíno Nano servem para as funcionalidades do ADC, enquanto o UNO possui 6 portas ADC, o Nano possui 8. A placa Nano não vem com um conector para fonte DC (*Direct Current - Corrente Contínua*) como outras placas Arduíno, mas possui uma porta mini-USB. Esta porta é usada para programação, monitoramento serial e alimentação de energia, se necessário. A característica fascinante do Nano é que ele escolherá a fonte de energia mais forte com sua diferença de potencial, e o *jumper* de seleção de fonte de energia é inválido (JOHN, 2018).

Para se comunicar com qualquer microcontrolador ou computador, é necessário usar uma linguagem de programação. No caso do Arduíno, é necessário usar a linguagem C++⁴, que é uma linguagem tradicional e já bastante conhecida. Porém, essa linguagem ainda não é a que a máquina comprehende, é uma linguagem de alto nível, e para a máquina comprehendê-la é necessário o uso de um compilador, que converte a linguagem de alto nível para a linguagem

⁴ Com alguma modificações.

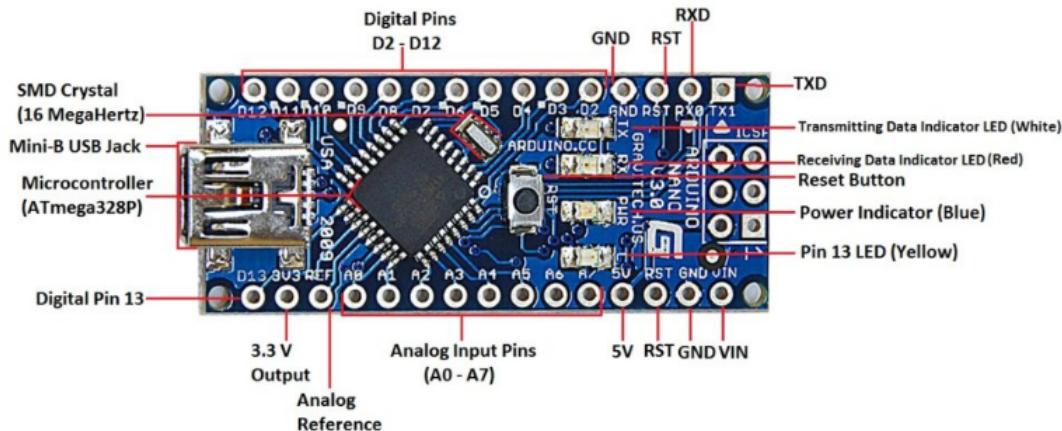


Figura 7 – Arduino Nano

Fonte: (JOHN, 2018)

de máquina. Em geral, para compilar um programa, é necessário a utilização de um IDE (*Integrated Development Environment* - Ambiente de Desenvolvimento Integrado), um aplicativo de computador que possui um compilador integrado. A plataforma do Arduíno utiliza o Arduíno IDE (CHAVIER, 2016).

2.4.5 ESP8266

O ESP8266, assim como o Arduíno, é um microcontrolador fabricado pela empresa *Espressif Systems*. Possui um sistema de comunicação WiFi próprio, podendo ser utilizado tanto para acessar a uma rede existente, como para criar o próprio ponto de acesso. Essa funcionalidade é seu grande diferencial em relação ao Arduíno, porém existem outras diferenças (OLIVEIRA, 2017).

A tensão das portas digitais é de 3.3 V, possui apenas uma porta para leitura analógica, 11 portas digitais, memória flash de 4Mb e um processador com velocidade de *clock* de 80MHz, todas essas características são diferentes nas placas Arduíno (WEMOS, 2017).

Existe uma grande variedade de placas disponíveis no Brasil. Mas em todas elas é utilizado apenas um processador ESP8266. O que distingue os modelos um do outro é o número de pinos GPIO (*General Purpose Input/Output - Entrada e Saída para Uso Geral*) expostos, a quantidade de memória flash fornecida, o estilo dos pinos do conector e várias outras considerações relacionadas à fabricação. Em relação a programação, elas são todas iguais e podem ter seu código embarcado escrito com a IDE do Arduíno, sendo apenas necessário carregar sua biblioteca de placas, através do próprio programa, e a linguagem de programação utilizada também é o C++ (KOLBAN, 2016).

Um dos modelos é a placa Wemos D1 (Figura 8), que tem o mesmo formato do Arduíno UNO, o que deixa a utilização mais confortável para usuários do Arduíno. Esse modelo possui

todos os seus pinos descritos na própria placa, também possui pinos para comunicação I2C, conector micro USB e um conector de carga que aceita tensão entre 9 e 24 V (KOLBAN, 2016)(WEMOS, 2017).

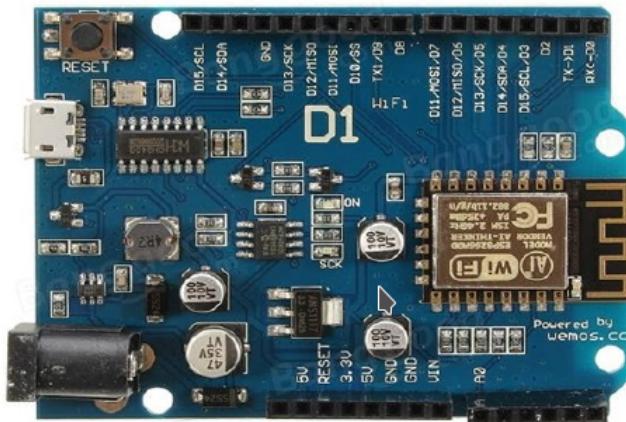


Figura 8 – WeMos D1 - R2

Fonte: (WEMOS, 2017)

2.5 PYTHON

Python é uma linguagem de programação poderosa, mas fácil de usar, desenvolvida por Guido van Rossum, em 1991. Com o *Python*, você pode escrever rapidamente um pequeno projeto e também, por se adaptar bem, pode ser usado para aplicativos comerciais. Uma prova de como é uma linguagem comercial é ela ser utilizada por empresas como *Google*, IBM, NASA, *Xerox*, *Yahoo* e outras grandes empresas (DAWSON, 2010).

O principal objetivo de qualquer linguagem de programação é fazer uma ponte entre o programador e a máquina. E a maioria das linguagens de programação tornam essa comunicação mais próxima da humana, mas o *Python* é tão direto que, segundo DAWSON (2010), é chamada de "programação na velocidade do pensamento". Essa facilidade se traduz na alta produtividade de programadores profissionais os quais com ele conseguem fazer programas mais curtos e mais rápidos de escrever do que com outras linguagens de programação.

Python é uma boa linguagem para aplicações científicas porque é fácil traduzir raciocínio em algoritmos através dela, bem como é simples ler um código em *Python* e conseguir entender o raciocínio por trás dele, por ser uma linguagem com poucos caracteres especiais, poucas palavras chaves utilizadas apenas para compilar e ter uma sintaxe muito próxima da linguagem falada (REITZ, 2018). Além disso, é uma linguagem de propósito geral. Muitas vezes, é necessário lidar com tarefas laterais: buscar dados em um banco de dados remoto, ler uma página na *internet*, exibir graficamente os resultados, criar uma planilha, etc. Linguagens de cunho especificamente científico têm um sério problema aí, mas, uma vez que *Python* é utilizada

em praticamente todos os tipos de tarefa, encontram-se módulos prontos para realizar essas tarefas que podem se tornar complicadas. Ou seja, é uma preocupação a menos para quem está desenvolvendo aplicações científicas (DOWNEY, 2012).

2.6 PROTOCOLO HTTP

Hypertext Transfer Protocol (HTTP) é o método utilizado para enviar e receber informações na *web*. A versão mais utilizada atualmente é a 1.1, definida pela especificação RFC 2616. RFCs (*Request for Comments*) são documentos técnicos desenvolvidos e mantidos pelo IETF (*Internet Engineering Task Force*), instituição que especifica os padrões que serão implementados e utilizados em toda a internet (VIEIRA, 2007).

O protocolo é baseado em requisições e respostas entre clientes e servidores. O cliente, navegador ou dispositivo que fará a requisição; também é conhecido como *user agent*, solicita um determinado *resource* (recurso), enviando um pacote de informações contendo alguns cabeçalhos (*headers*) a um URI (*Uniform Resource Identifier - Identificador Uniforme de Recurso*) ou, mais especificamente, URL (*Uniform Resource Locator - Localizador Uniforme de Recursos*). O servidor recebe estas informações e envia uma resposta, que pode ser um recurso ou um simplesmente um outro cabeçalho (VIEIRA, 2007).

O protocolo HTTP é *stateless*, ou seja, ele não é capaz por si só de reter informações entre requisições diferentes. Para manter informações é necessário utilizar *cookies*, sessões, campos de formulário ou variáveis na própria URL (VIEIRA, 2007).

Quando um requisição é realizada, é preciso especificar qual método deve ser utilizado. O HTTP possui 8, também chamados de verbos, identificam a ação a ser executada por certo recurso. Os 5 métodos mais utilizados são: *get*, *post*, *delete*, *put* e *head* (VIEIRA, 2007)(SZY-GALSKI, 2018).

Get solicita a representação de um recurso. É definido como um método seguro e não deve ser usado para disparar uma ação. *Post* é a informação enviada ao corpo da requisição e é utilizado para criar um novo recurso. *Delete* remove um recurso. *Put* atualiza um recurso na URI especificada. Caso o recurso não exista, ele pode criar um. A principal diferença entre *post* e *put* é que o primeiro pode lidar não somente com recursos, mas pode fazer processamento de informações, por exemplo. *Head* retorna informações sobre um recurso. Na prática, funciona semelhante ao método *get*, mas sem retornar o recurso no corpo da requisição. Também é considerado um método seguro (VIEIRA, 2007).

Toda requisição recebe um código de resposta conhecido como *status*. Assim é possível saber se uma operação foi realizada com sucesso (código 200), se ele foi movida e agora existe em outro lugar (código 301) ou se não existe mais (código 404), e existem muitos outros códigos (VIEIRA, 2007).

O HTTP é um protocolo extensível que é simplesl de usar. A arquitetura cliente-servidor, combinada com a habilidade de simplesmente adicionar cabeçalhos, permite que o HTTP avance suas funcionalidades juntamente com a elasticidade da *web* (SZYGALSKI, 2018).

3 METODOLOGIA

Nesta seção será apresentado quais foram os materiais e métodos utilizados para implementação do projeto, tanto da parte de *hardware* (parte física) quanto a parte de *software*. A principal metodologia utilizada foi baseada no *Scrum*, que é um método ágil iterativo incremental, onde o projeto foi dividido em ciclos de 15 dias chamados de "*sprints*". A cada *sprint*, algumas atividades são retiradas do *backlog*¹, colocadas no campo "*do*"² e tem de ser resolvidas até o início da próxima *sprint*, onde as atividades realizadas serão avaliadas, validadas ou não, se estiverem atendendo ao requisito são enviadas para o "*done*"³, se não estiverem atendendo as necessidades, elas voltam ao *backlog* ou para o *do*.

Os requisitos do projeto, que compuseram o *backlog* e definiram os objetivos desse projeto foram relacionados com o auxílio dos pesquisadores do LIS e alunos de mestrado da UnB/FGA.

O trabalho foi dividido em duas etapas, a primeira etapa foi focada em realizar a comunicação via porta serial e com o microcontrolador Arduíno, com ele foram adquiridos os parâmetros de um sensor MPU6050, em seguida foi realizada a multiplexação de 3 sensores para poder obter medidas de 3 pontos ao mesmo tempo através de um único Arduíno. A segunda etapa foi tentar comunicar o sensor com o computador por meio de transmissão sem fio e com protocolo HTTP, utilizando o ESP8266 e seu módulo de WiFi.

3.1 PROJETO DE HARDWARE

3.1.1 MATERIAIS UTILIZADOS

- Sensor MPU6050;
- Arduíno Nano V.3;
- WeMos D1 -R2;
- Fios (*jumpers*);
- Cabo USB/Micro USB;
- Cabo USB/Mini USB;
- Computador com Arduíno IDE e *Python* instalados;

¹ Conjunto de requisitos que ainda não foram solucionados.

² Campo onde estão as atividades a serem realizadas durante a *sprint*.

³ Conjunto de atividades que já foram feitas e validadas.

- *Jupyter Notebook*;
- Tiras de velcro;
- Cabo *flat* (8 fios);
- *Protoboard*;
- Placa de Cobre Perfurada - 15x10cm;
- Bateria 5V - 2A - conector - Mini USB .

3.1.2 MÉTODOS

Para realizar a comunicação do sensor MPU6050 com o Arduíno foi feita a conexão conforme o esquemático mostrado na Figura 9 e na Tabela 2. Logo abaixo, onde os pinos Vcc e Gnd do MPU6050, responsáveis por ligar do sensor, foram conectados aos pinos 5V e Gnd do Arduíno. O pino SDA (*Serial Data* - Dados Seriais)⁴ foi conectado ao pino SDA do Arduíno, que também é o pino de entrada analógica A4 e o pino SCL (*Serial Clock* - "Relógio" Serial)⁵ ao pino SCL do Arduíno, que também é o pino de entrada analógica A5.

O modelo escolhido para este projeto foi o Arduíno Nano, pois mesmo sendo apenas um protótipo, haverá a necessidade de que o tamanho do IMU seja reduzido para facilitar a fixação em diversas partes do corpo e o custo financeiro⁶ para desenvolvimento do protótipo foi levado em consideração.

Para os primeiros testes foi utilizada um *protoboard* para facilitar as conexões, pois ela permite que os componentes sejam conectados sem a necessidade de solda entre eles. Mas em um segundo momento para os teste de movimento serem realizados com maior acurácia, houve a necessidade de retirar os componentes da *protoboard* e soldá-los em uma placa furada de cobre, e isso faria o ruído devido a conexões instáveis fosse menor.

O sensor foi fixado em uma faixa de velcro, para permitir a fixação do IMU em diversas partes do corpo, e para conectá-lo ao Arduíno foi soldado ao sensor um cabo *flat* com 8 fios.

Tabela 2 – Conexões entre Arduíno e MPU6050

Arduíno	MPU6050
5V	VCC
GND	GND
A4	SDA
A5	SCL

Fonte: autoria própria

⁴ Pino de transferência de dados.

⁵ Pino de sincronização do *clock*.

⁶ Preços pesquisados em lojas de Brasília.

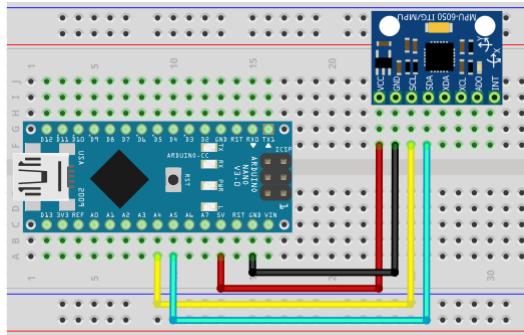


Figura 9 – Esquemático de conexão do MPU6050 com Arduíno Nano

Fonte: autoria própria.

A comunicação entre o sensor MPU6050 e o Arduíno foi realizada através do protocolo I2C, onde os dados eram enviados de forma serial através de apenas um fio e os dados organizados como mostrado na Figura 10, onde é evidenciado quais são os dados transmitidos do MPU6050 para o Arduíno e a velocidade do *clock*, e Figura 11, que mostra com mais detalhe a relação dos dados com o *clock*.

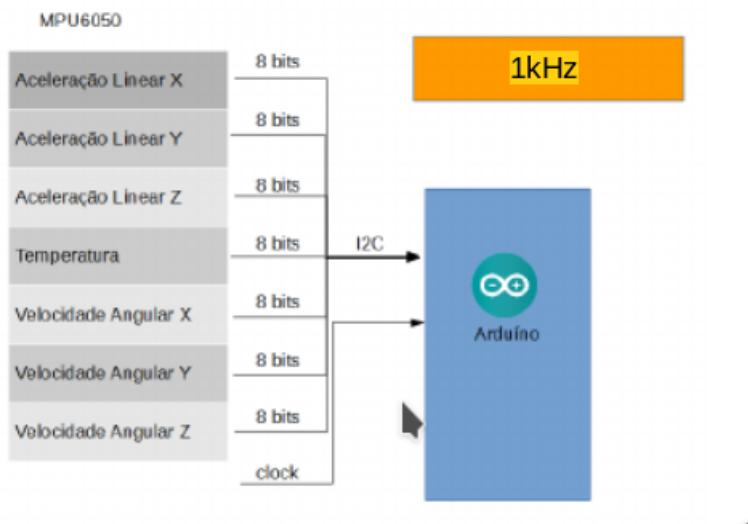


Figura 10 – Transmissão de dados do MPU6050 para Arduíno

Fonte: autoria própria.

O Arduíno Nano foi conectado ao computador por meio de um cabo USB 2.0, que além de ser responsável pela comunicação, também era por meio dele que era realizada a alimentação de energia com 5V de tensão. O protocolo de comunicação utilizado com um cabo USB também foi o serial. Então, os dados transmitidos do Arduíno para o computador são enviados em série.

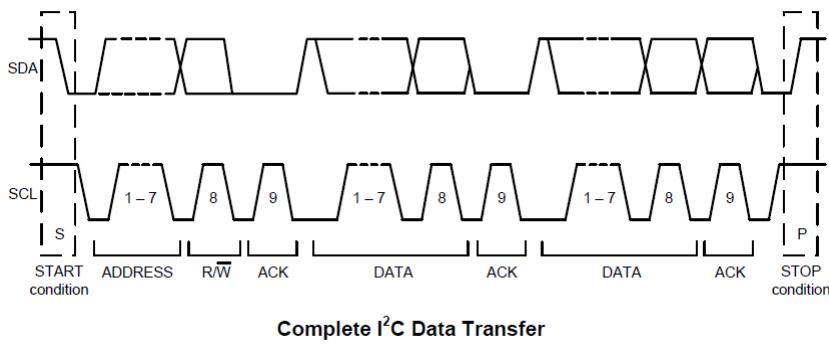


Figura 11 – Transferência de dados com o protocolo I2C.

Fonte: (INVENSENSE, 2013).

3.2 PROJETO DE SOFTWARE

Para esse trabalho foram necessários dois projetos de *softwares*, um para ficar salvo no Arduíno e outro que será o *software*, em *Python*, responsável por fazer o tratamento dos dados adquiridos do sensor e salvá-los para análises futuras.

3.2.1 SOFTWARE PARA O ARDUÍNO

Os requisitos do software salvo no microcontrolador são:

- Inicializar o MPU6050;
- Receber os dados do MPU6050;
- Enviar os dados para interface serial.

Para isso foi utilizada as bibliotecas *Wire.h*, *I2Cdev.h* e *MPU6050.h*, que são responsáveis por realizar a comunicação em I2C, enviar o comando de inicialização, a variável escolhida pelo usuário para definir os limites de leitura do acelerômetro e do giroscópio. São estas bibliotecas que realizam a leitura dos registradores que tem o valor de cada sensor armazenado neles.

O fluxograma do *software* responsável pela leitura dos dados do MPU6050 e envio para a *interface* de comunicação serial é o mostrado na Figura 12.

Para realizar a transmissão dos dados do Arduíno para o programa em *Python*, o programa salvo no Arduíno imprime no terminal serial os dados obtidos do sensor. O programa é fechado sempre que uma nova leitura é realizada.

3.2.2 SOFTWARE PARA SALVAR OS DADOS

Esse segundo programa, foi feito para realizar a leitura dos dados transmitidos pelo Arduíno, e salvá-los em um arquivo de texto, para futuras análises. Esse *software* ainda dá a

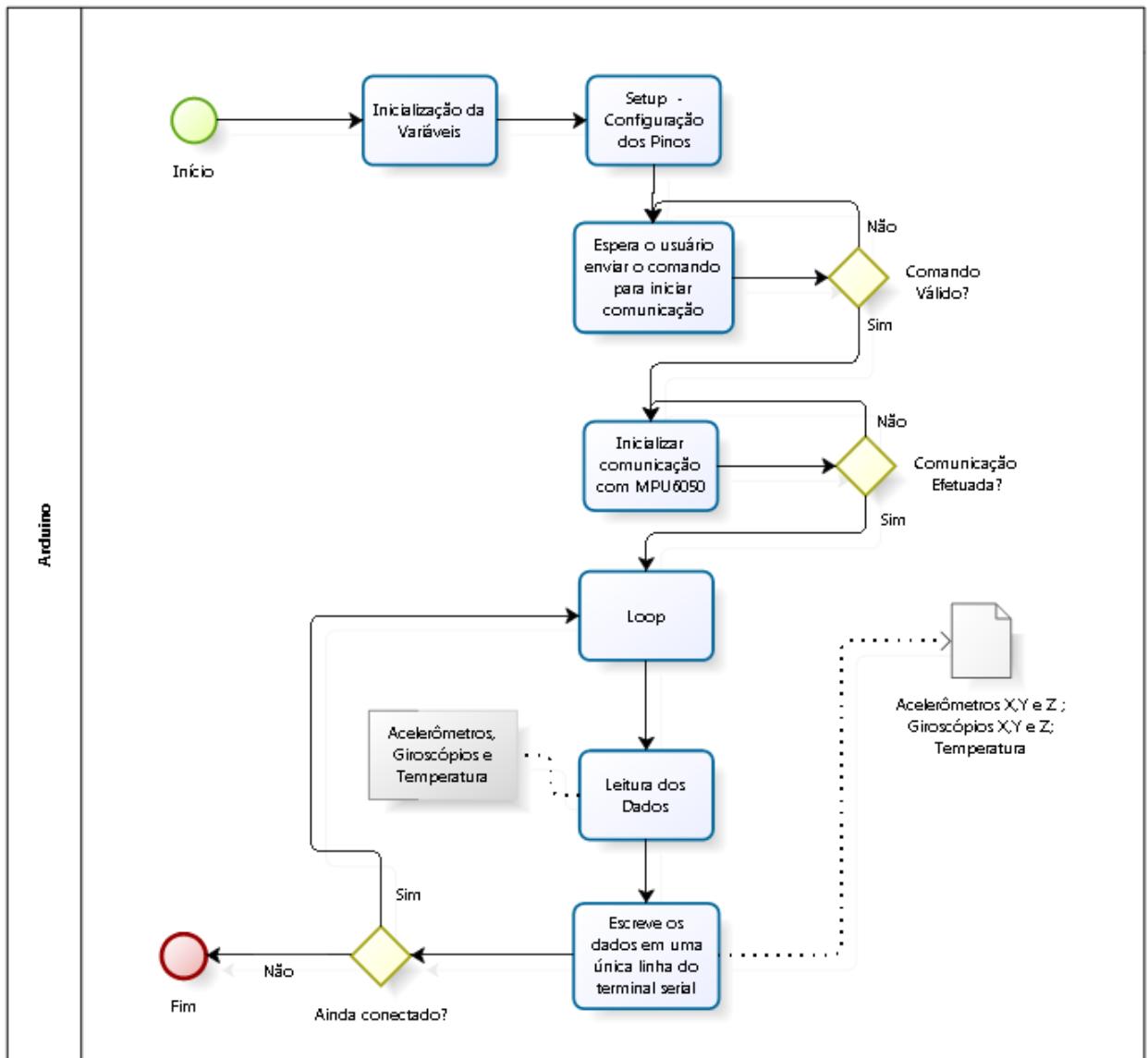


Figura 12 – Fluxograma do *software* de leitura dos dados.

Fonte: autoria própria.

possibilidade do usuário salvar os dados sem nenhum tratamento ou os valores com medidas físicas aproximadas das medições realizadas.

O *software* foi escrito em *Python 3* e compilado com o *Jupyter Notebook* e com a IDE *Spyder*, os dois compiladores são do pacote Anaconda.

O programa inicia mostrando a lista de dispositivos USB conectados ao computador. Se não houver nenhum dispositivo conectado aparece a mensagem "Conecte o Arduíno". Se houver, a mensagem que aparece pede para digitar o número correspondente ao dispositivo que o usuário pretende verificar os dados.

Após a escolha do Arduíno, o usuário deve escolher entre as 4 possibilidades de escala

dos dados obtidos. E em seguida pressionar "enter" para iniciar a leitura e "Ctrl+C"⁷ para pausar e salvar os valores. O programa então imprime na tela os valores dos sensores que estão sendo lidos e ao fim do código, quando o usuário pressionar "Ctrl+C", os valores são salvos em dois arquivos nomeados com a data de quando foi finalizada a leitura, como mostrado na Figura 13.

Para definir melhor os requisitos do *software*, foi feita a seguinte história de usuário descrita na Tabela 3, e a partir dela foi possível desenvolver o *software* inicial e com ele realizar os testes preliminares do protótipo.

Tabela 3 – História de Usuário

Usuário	O que quer fazer?	Resposta do Sistema
Pesquisador	Iniciarizar o sistema	Abre o <i>prompt</i> de comando Mostra a lista de dispositivos conectados
Pesquisador	Selecionar o sensor	Estabelecer a conexão com sensor Mostra a lista de intervalo de leitura
Pesquisador	Selecionar o intervalo	Mensagem de qual intervalo foi selecionado Aguarda comando para iniciar a coleta
Pesquisador	Iniciar coleta	Inicia a coleta Aguarda comando para pausar coleta
Pesquisador	Pausar coleta e salvar Dados	Para a coleta Fecha a conexão Salva os dados coletados Mensagem com nome dos arquivos

Fonte: autoria própria

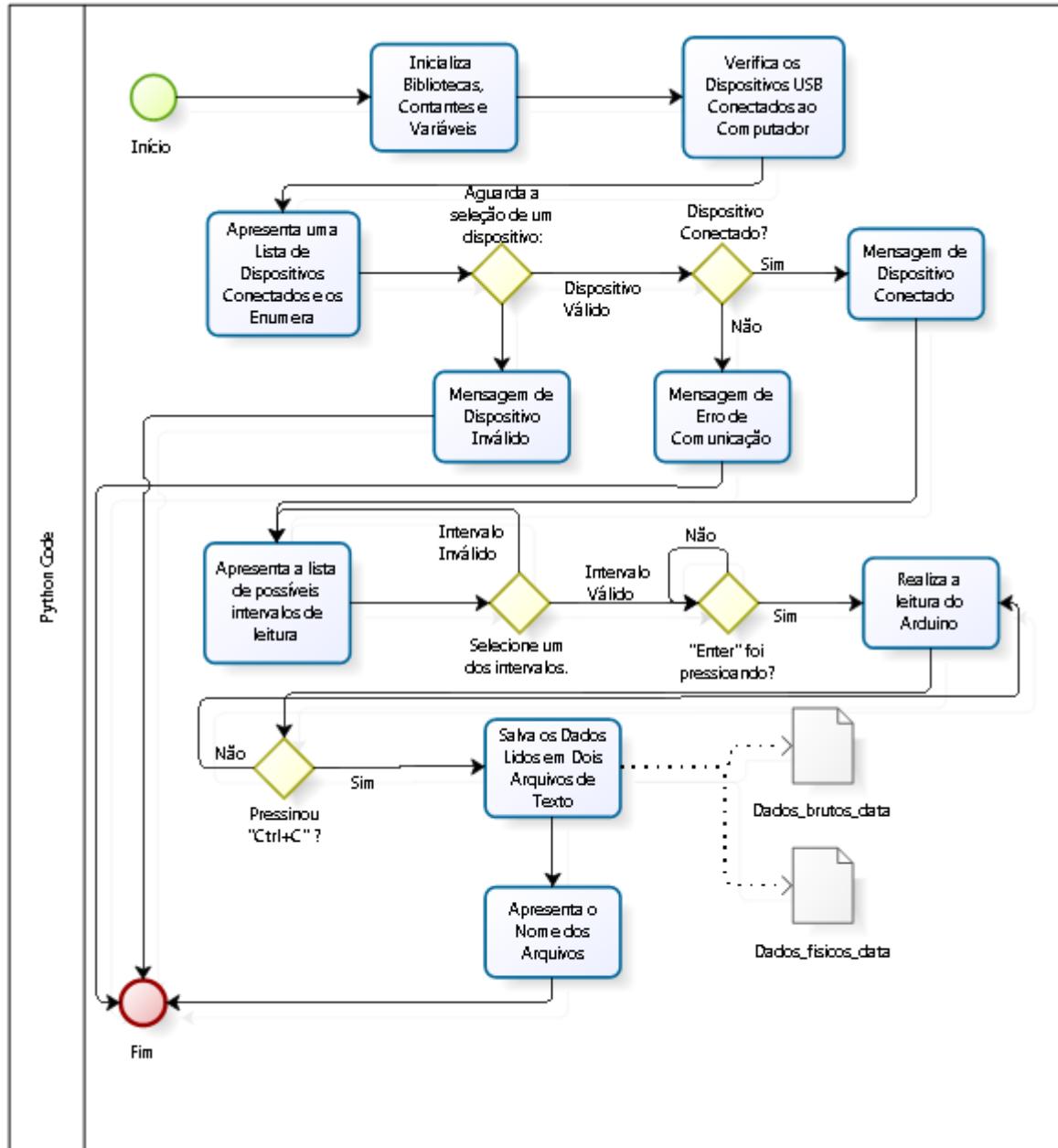
3.3 VALIDAÇÃO

Para testar o funcionamento dos acelerômetros do IMU, foi feito um teste simples. Pois sempre que o sensor estiver "parado", ele estará sobre efeito da força do peso e consequentemente da aceleração da gravidade de aproximadamente $9,8 \text{ m/s}^2$. Então para verificar, foi necessário testar o sensor com um eixo de cada vez alinhado com a Normal, como na Figura 14, onde a imagem mais à direita mostra o MPU6050 com o eixo X paralelo a Normal, a imagem à direita com o eixo Y paralelo a Normal e ao centro o eixo Z.

Após realizar as leitura, os dados foram plotados em um gráfico simples, onde o eixo horizontal representa o tempo e o vertical a aceleração medida. Foram colocados os gráficos dos 3 eixos na mesma imagem.

Para verificar se os giroscópios estavam funcionando, sem ainda se atentar a precisão deles e confiabilidade das medidas, foi realizado um movimento de rotação de 90° em relação a cada um dos eixos, como na Figura 15 e ao fim da leitura do sensor o resultado foi plotado em um gráfico para facilitar a análise.

⁷ No Jupyter Notebook o "Ctrl+C" não funcionou, precisando assim pressionar o ícone de "stop" disponível na tela.

Figura 13 – Fluxograma do *software* que salva os dados lidos.

Fonte: autoria própria.

Para verificar o funcionamento do MPU6050 e se as leituras são confiáveis, ou não, serão necessários muitos outros testes além dos realizados até o momento, como a comparação dos valores obtidos medindo movimentos corporais ao mesmo tempo que uma gravação de vídeo é realizada, e verificar se os valores obtidos com o IMU são condizentes com as imagens gravadas. E também um possível estudo de caso, ainda a ser formulado, com o auxílio dos pesquisadores do LIS. Realizando as medições em voluntários para estudo do movimento humano.

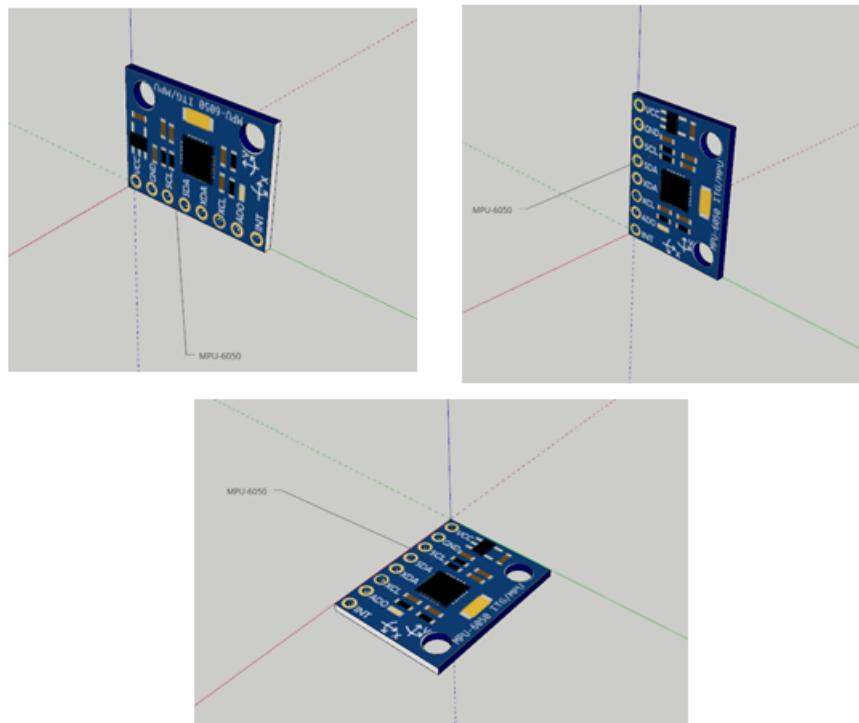


Figura 14 – Demonstração da posição do sensor

Fonte: autoria própria.

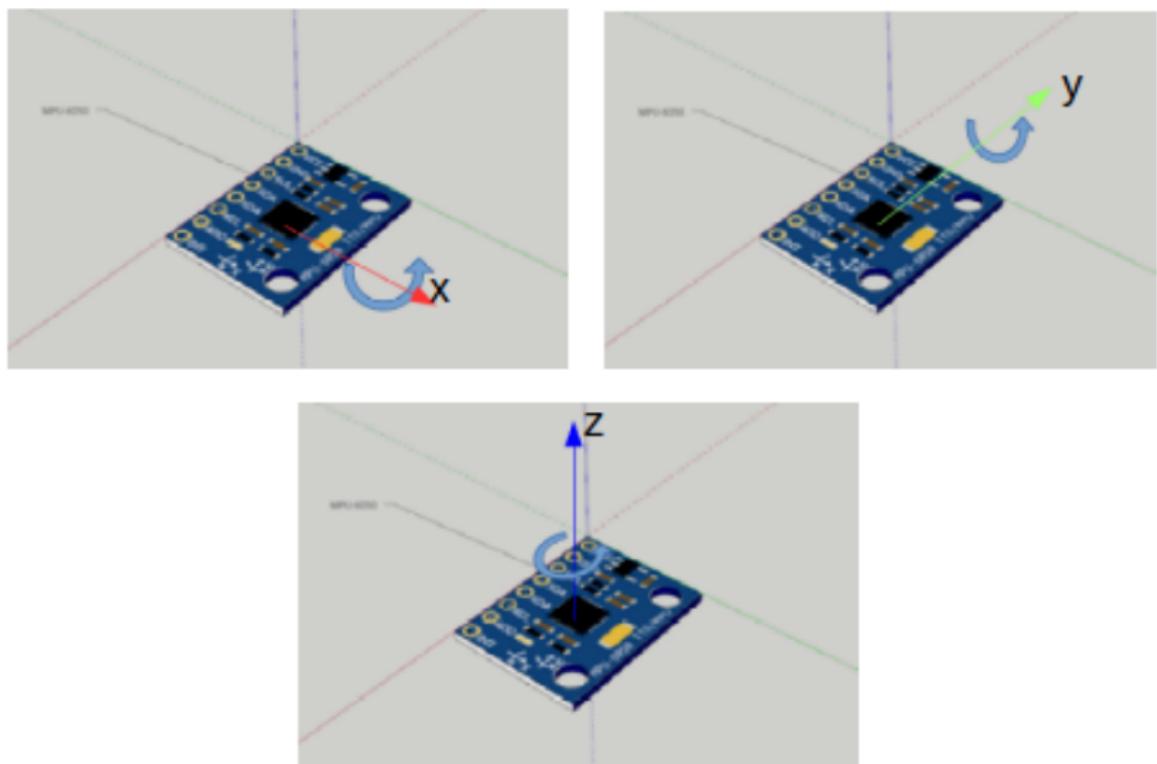


Figura 15 – Demonstração direção de giro do sensor

Fonte: autoria própria.

4 RESULTADOS E DISCUSSÃO

4.1 RESULTADOS

Os resultados obtidos são os seguintes: protótipo funcional, *case* para fixação¹, *software* para o microcontrolador, *software* para ler, tratar e salvar os dados, tabelas e gráficos parciais do funcionamento do IMU.

As Figuras 16 e 17 mostram a primeira montagem feita do IMU para programar o microcontrolador e verificar a comunicação entre o sensor e o Arduíno. As Figuras 18 e 19 mostram a segunda montagem realizada do IMU, onde ele foi soldado com um cabo *flat* e preso em uma tira de velcro.

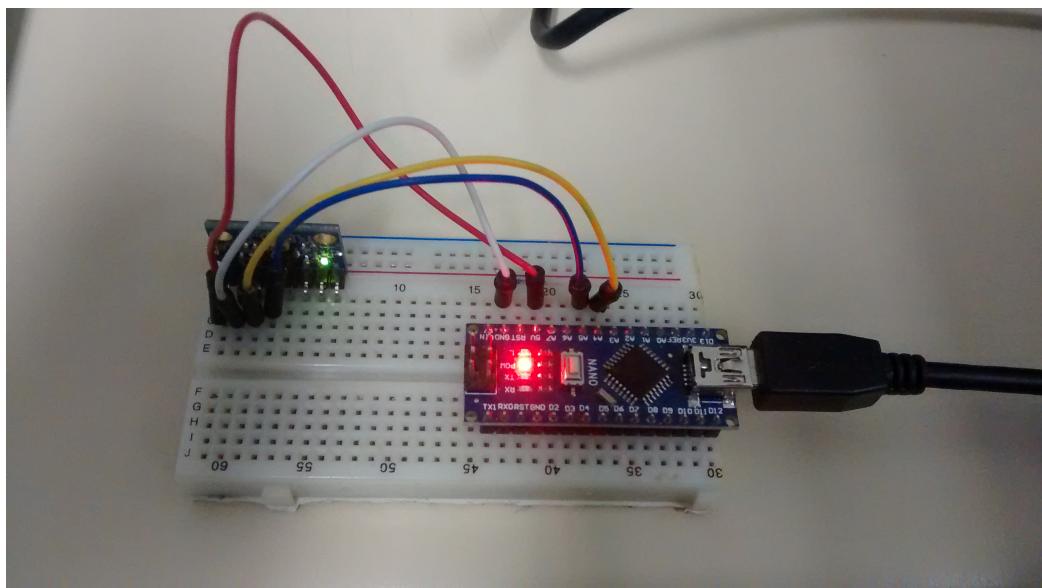


Figura 16 – Protótipo na *Protoboard*

Fonte: autoria própria.

O programa em funcionamento é mostrado nas Figuras 20, 21 e 22 que, por enquanto, apenas no *prompt* de comando. Mas futuramente precisará de uma *interface* mais amigável. A Figura 20 mostra a mensagem que aparece assim que a pessoa coloca o programa para funcionar, no caso, aparece também a lista de dispositivos conectados e pede para a pessoa escolher um deles. Se nenhum estivesse conectado, apareceria a mensagem "Conecte o Arduíno".

A Figura 21, mostra o que aparece logo depois de escolher o Arduíno. A confirmação de conexão com Arduíno, uma mensagem, que vem do próprio Arduíno, pedindo para selecionar

¹ Ainda não é a definitiva.

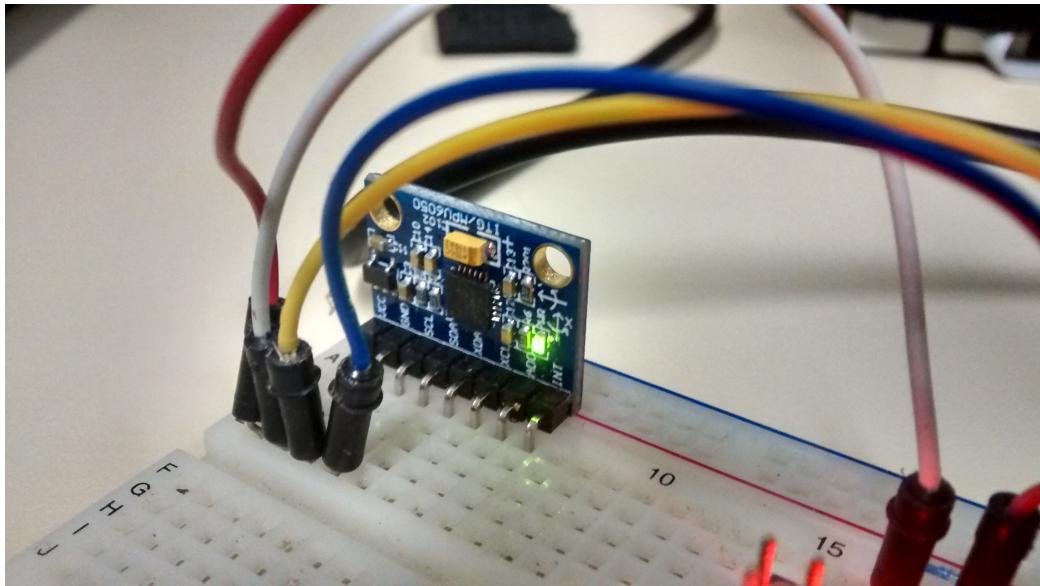


Figura 17 – MPU6050 na *Protoboard*

Fonte: autoria própria.



Figura 18 – Segundo Protótipo com o Velcro

Fonte: autoria própria.

uma das opções mostradas logo abaixo em um lista de possíveis escalas de leitura para o IMU e aguarda a escolha do usuário.

E na Figura 22, aparece a mensagem para iniciar a leitura do sensor e alguns dos dados lidos também são mostrados para o usuário, precisando apenas pressionar Ctrl+C para pausar a leitura e salvar os dados em dois arquivos. Apenas para os testes foram acrescentados também ao final do programa a geração de dois gráficos, um com os dados obtidos dos acelerômetros e o outro para o giroscópio.

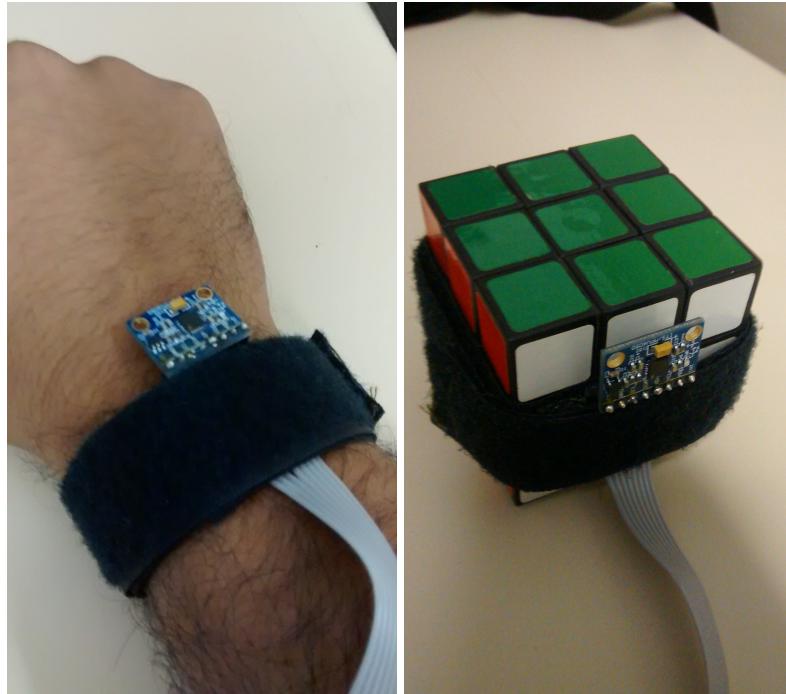


Figura 19 – Protótipo Fixado ao Punho e a um Cubo

Fonte: autoria própria.

```
===== Lista de dispositivos USB =====
0 - /dev/ttyUSB0
Escolha a porta do Arduino (e.g. 0): |
```

Figura 20 – Tela Inicial

Fonte: autoria própria.

Os gráficos apresentados na Figura 23, mostram o resultado parcial do teste para medir a aceleração da gravidade com cada um dos 3 acelerômetros. É possível a partir deles verificar que os acelerômetros estão obtendo valores realistas para as medidas feitas, pois todos os gráficos apresentaram um valor muito próximo ao da aceleração da gravidade. Os gráficos ainda não estão com aspecto ideal, dando a sensação de presença de ruído, mas isso pode ser corrigido com um *buffer* para carregar uma quantidade de valores e mostrar apenas a média deles, porque o sensor é muito sensível a pequenas variações de aceleração.

Os gráficos da Figura 24 representam à esquerda, à direita e ao centro, respectivamente, a rotação em relação aos Eixos X, Y e Z. O gráfico apresenta muitos picos e variações que parecem ruidosas, mas na realidade é porque o sensor é muito preciso. E na hora de realizar o teste precisava de uma movimentação mais suave e constante, para melhor visualização no gráfico, mas como o teste foi realizado movimentando o sensor com a mão, ele captou a vibração natural da mão e o impacto com a superfície no momento que ele é solto, o que explica a movimentação em todos os sensores e não apenas em um.

Porém, mesmo com o gráfico não apresentando o resultado ideal, é possível verificar a

```
Comunicação Serial Estabelecida.  
Digite o comando.(0,1,2 ou 3)
```

Intervalos de leitura:

```
0 ----> +- 250 deg/s e +- 2g  
1 ----> +- 500 deg/s e +- 4g  
2 ----> +- 1000 deg/s e +- 8g  
3 ----> +- 2000 deg/s e +- 16g
```

Escolha o intervalo de leitura (e.g. 0): |

Figura 21 – Lista com as escalas

Fonte: autoria própria.

```
Escolha o intervalo de leitura (e.g. 0): 0  
Pressione 'Enter' para iniciar a leitura e "Ctrl+C" para pausar:  
Comando 0 recebido.  
Erro de valor.  
-296 -4 16160 -3584 -294 -197 -68  
-324 20 16104 -3568 -287 -143 -69  
-308 56 16140 -3552 -270 -161 -90  
-368 -68 16172 -3536 -248 -175 -85  
-224 112 16116 -3568 -262 -179 -56  
-272 120 16268 -3552 -296 -168 -73  
-356 96 16156 -3520 -283 -214 -79
```

Figura 22 – Programa mostrando os resultados

Fonte: autoria própria.

variação nas velocidades de rotação em cada giroscópio separadamente, o que demonstra que os giroscópios funcionam.

4.2 DISCUSSÃO

O primeiro protótipo, na *protoboard*, foi útil por facilitar os testes de conexão, pois com a *protoboard* não era necessário soldar os pinos, mas por ser uma conexão instável está propensa a ruídos. E por isso foi feito o segundo protótipo. O cabo *flat* foi escolhido porque ele permitiu soldar o sensor e ainda mantê-lo com um tamanho confortável para fixação.

A *interface* do programa ainda precisa ser melhorada, mas atende a aplicação, porque foi feita para utilização de pesquisadores do laboratório que receberam o treinamento e saberão lidar com linhas de comando, ou receberão o treinamento necessário para a devida utilização. Com a melhoria da interface, será mais intuitivo utilizar o programa e ele se tornará ainda mais

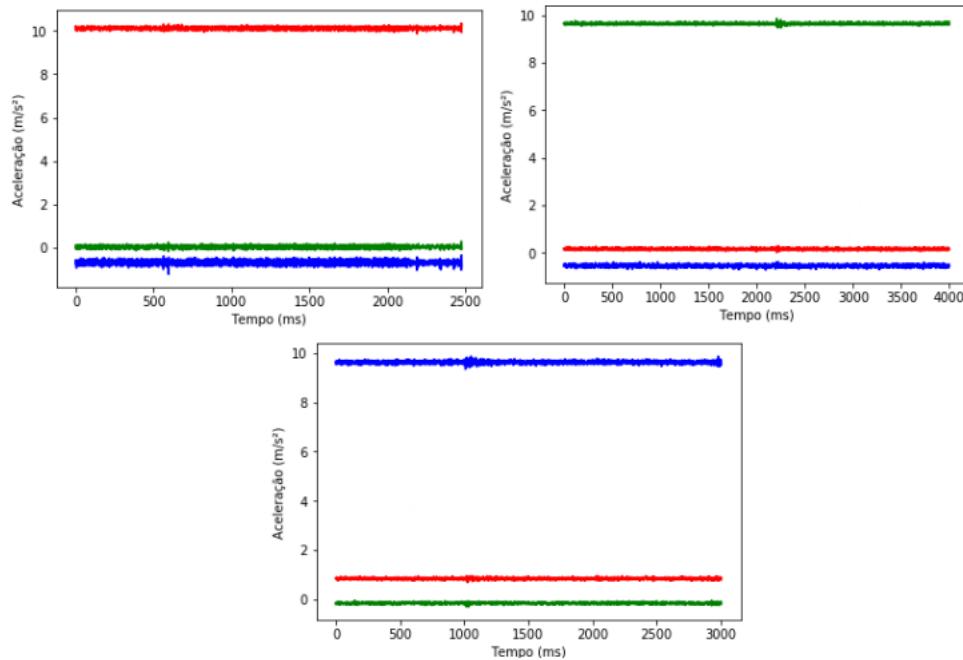


Figura 23 – Gráficos com os 3 Eixos alinhados com a Normal.

Na imagem à esquerda o Eixo X está alinhado, à direita o Eixo Y e ao centro o Eixo Z. Fonte: autoria própria.

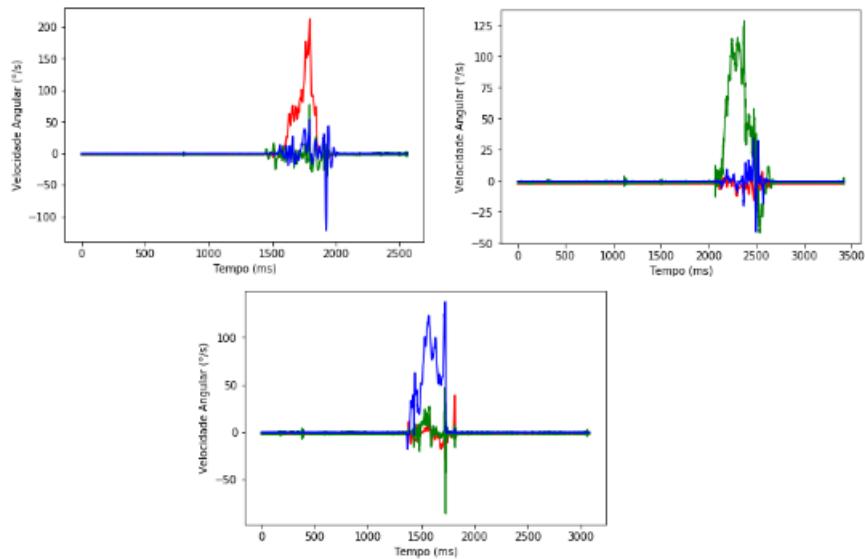


Figura 24 – Gráficos para Teste do Giroscópio.

Fonte: autoria própria.

acessível, podendo ser utilizado por outros profissionais que tiverem interesse no movimento humano. Essa melhora não foi feita ainda devido ao tempo para realização do projeto que foi curto para as demandas do mesmo.

A partir dos resultados preliminares, mostrados nos gráficos, é possível verificar que o IMU pode ser utilizado para o propósito estabelecido inicialmente, mas como este é apenas um protótipo muitos testes ainda serão necessários. Os gráficos ficariam melhores se os testes não

tivessem sido realizados com os métodos que foram utilizados. Pois, o sensor acabou medindo a movimentação da mão que o manipulou e também os impactos nele.

5 CONCLUSÃO

A medição dos movimentos corporais é muito importante para estudos sobre prevenção de lesão em atletas de alto desempenho. Para realizar essas medições são necessários equipamentos específicos. E o IMU surge como uma solução viável para facilitar o acesso dos pesquisadores a sensores de utilização menos complexas e de custo mais baixo.

Foi parcialmente concluído, neste trabalho, o protótipo de um sensor IMU capaz de medir a aceleração linear em três eixos, e a velocidade angular em três eixos. A partir dos resultados parciais obtidos, é possível afirmar que com essas seis medições os pesquisadores a utilizem com o *software* que foi desenvolvido para a leitura do sensor. E a partir dos dados coletados, possam fazer análise de movimento, posição ou postura.

A próxima etapa do trabalho é desenvolver um protocolo de comunicação sem fio para o IMU, pois traria mais conforto ao paciente, melhoraria a precisão dos dados e facilitaria o manuseio. Outra etapa é tentar miniaturizar o IMU, para facilitar a conexão dele em diversas partes do corpo e aumentar o conforto para o usuário. E, por fim, projetar uma *case*, que deixe o IMU mais protegido de impactos, mais firme e seja miniaturizada.

REFERÊNCIAS

- ACQUESTA, F. M. et al. Estudo da biomecânica do movimento humano no brasil. *Revista Brasileira de Biomecânica*, 2008. Citado na página 21.
- AHMAD, N.; GHAZILLA, R. A. R.; KHAIRI, N. M. Reviews on various inertial measurement unit (imu) sensor applications. *International Journal of Signal Processing Systems*, 2013. Citado 2 vezes nas páginas 26 e 27.
- ALMEIDA, V. M. de. Sensores inerciais. 2014. Acesso em: 06/07/2018. Disponível em: <<http://www.decom.ufop.br/imobilis/sensores-inerciais/>>. Citado na página 29.
- AMADIO, A. C. Metodologia biomecânica para o estudo das forças internas ao aparelho locomotor: importância e aplicações no movimento humano. In: *A Biomecânica do Movimento e Suas Relações Interdisciplinares*. São Paulo, Brasil: [s.n.], 2000. p. 45–70. Citado na página 23.
- AMADIO, A. C.; SERRÃO, J. C. Conyextualização da biomecânica para a investigação do movimento: fundamentos, métodos e aplicações para análise da técnica esportiva. In: *Revista Brasileira de Educação Física e Esporte*. São Paulo, Brasil: [s.n.], 2007. v. 21, p. 61–85. Citado 2 vezes nas páginas 23 e 24.
- CARNEIRO, F. M. Levantamento bibliográfico das tecnologias dos acelerômetros comerciais. 2003. Acesso em: 05/07/2018. Disponível em: <http://www.fem.unicamp.br/~lotavio/tgs/2003_BibliografiaDeAcelerometros_TG_FelipeCarneiro.pdf>. Citado na página 28.
- CHANG, H.; GEORGY, J.; EL-SHEIMY, N. Improved cycling navigation using inertial sensors measurements from portable devices with arbitrary orientation. *IEEE Transactions on Instrumentation and Measurement*, 2016. Acesso em: 29/06/2018. Disponível em: <<https://ieeexplore.ieee.org/document/7112512/citations?tabFilter=papers>>. Citado 2 vezes nas páginas 21 e 22.
- CHAVIER, L. F. Programação para arduino. 2016. Acesso em: 05/07/2018. Disponível em: <<<https://www.circuitar.com.br/tutoriais/programacao-para-arduino-primeiros-passos/>>>. Citado na página 33.
- DAWSON, M. *Python® Programming for the Absolute Beginner, Third Edition*. Massachusetts - USA: Cengage Learning, 2010. 480 p. Citado na página 34.
- DOWNEY, A. *Think Python: How to Think Like a Computer Scientist*. Massachusetts - USA: Green Tea Pess, 2012. 240 p. Citado na página 35.
- FORHAN, N. A. E. Giroscópios mems. São José dos Campos - SP, Brazil, 2010. Acesso em: 05/07/2018. Disponível em: <<<http://urlib.net/sid.inpe.br/mtc-m19@80/2010/01.25.18.42>>>. Citado 2 vezes nas páginas 29 e 30.
- HOWARD, R. Wireless sensor devices in sports performance. *IEEE Potentials*, 2016. Citado na página 27.

INVENSENSE, I. *MPU-6000 and MPU-6050 Product Specification Revision 3.4.* [S.l.], 2013. Citado 4 vezes nas páginas 27, 30, 31 e 40.

JOHN, D. Arduino nano tutorial - pinout and schematics. 2018. Acesso em: 09/07/2018. Disponível em: <<http://www.circuitstoday.com/arduino-nano-tutorial-pinout-schematics>>. Citado 2 vezes nas páginas 32 e 33.

KOLBAN, N. *Kolban's Book on ESP8266.* [S.l.: s.n.], 2016. v. 1. 436 p. Citado 2 vezes nas páginas 33 e 34.

MADEIRO, F.; LOPES, W. T. A. Introdução à compressão de sinais. *Revista de Tecnologia da Informação e Comunicação*, 2011. Citado na página 24.

MCGINNIS, R. S. *Advancing Applications of IMUs in Sports Training and Biomechanics.* Tese (Doutorado) — University of Michigan, 2013. Citado na página 21.

MEDEIROS, M. F. Identificação de assimetrias bilaterais dos membros inferiores por meio de salto vertical. Minas Gerais, Brasil, 2012. Citado na página 23.

MIZIARA, I. M. Proposta de um sistema para avaliação biomecânica de atletas de taekwondo. Uberlândia - MG, Brasil, 2014. Citado na página 23.

NUSSENZVEIG, H. M. *Curso de Física Básica: Mecânica.* [S.l.]: Blucher, 2013. v. 1. 336 p. Citado 2 vezes nas páginas 28 e 30.

OBERLANDER, K. D. *Inertial Measurement Unit (IMU) Technology: Inverse Kinematics: Joint Considerations and the Maths for Deriving Anatomical.* Tese (Doutorado) — University of Koblenz-Ladau, 2015. Citado 2 vezes nas páginas 21 e 22.

OKAZAKI, V. H. A. et al. Ciência e tecnologia aplicada à melhoria do desempenho esportivo. *Revista Mackenzie de Educação Física e Esporte*, 2012. Citado na página 21.

OLIVEIRA, R. R. de. Uso do miconcontrolador esp8266 para automação residencial. 2017. Citado na página 33.

ONORATO, P.; MASCHERETTI, P.; DEAMBROSIS, A. Investigating the magnetic interaction with geomag and tracker video analysis: static equilibrium and anharmonic dynamics. *European Journal of Physics*, 2012. Nenhuma citação no texto.

OPPENHEIM, A. V.; WILLSKY, A. S.; NAWAB, S. H. *Sinais e Sistemas.* Brasil: Pearson, 2010. 592 p. Citado na página 24.

PRIME. *An Introduction to MEMS (Micro-electromechanical Systems).* [S.l.]: Prime Faraday Technology Watch, 2002. 56 p. Citado 2 vezes nas páginas 25 e 26.

REITZ, K. *Python Guide Documentation.* [S.l.: s.n.], 2018. v. 1. 129 p. Citado na página 34.

ROBERTS, M. M. *Arduino Básico.* São Paulo, Brazil: Novatec, 2011. 456 p. Citado na página 32.

ROCHA, M. Fisiologia do exercício. In: *Atlas do esporte no Brasil.* Rio de Janeiro, Brasil: [s.n.], 2005. p. 657 – 659. Citado na página 23.

- SANJEEV, A. How to interface arduino and the mpu 6050 sensor. 2018. Acesso em: 08/07/2018. Disponível em: <<https://maker.pro/arduino/tutorial/how-to-interface-arduino-and-the-mpu-6050-sensor>>. Citado 2 vezes nas páginas 28 e 29.
- SANTOS, C. P.; VIEIRA, M. E. M.; JUNIOR, S. L. S. Sensores inercias aplicados à marcha humana no esporte. *SEA-Seminário de Eletrônica e Automação*, 2016. Citado na página 26.
- SMITH, S. *Wearable Technology and Gesture Recognition for Live Performance Augmentation*. Tese (Doutorado) — University of Southern Queensland, 2016. Citado 2 vezes nas páginas 31 e 32.
- SONG, M.; GODOY, R. I. How fast is your body motion? determining a sufficient frame rate for an optical motion tracking system using passive markers. *Plos One*, 2016. Citado 2 vezes nas páginas 24 e 25.
- SZYGALSKI, C. A. Uma visão geral do http. 2018. Acesso em: 05/12/2018. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/HTTP/Overview>>. Citado 2 vezes nas páginas 35 e 36.
- VIEIRA, N. Entendendo um pouco mais sobre o protocolo http. 2007. Acesso em: 05/12/2018. Disponível em: <<https://nandovieira.com.br/entendendo-um-pouco-mais-sobre-o-protocolo-http>>. Citado na página 35.
- WEMOS. D1: An arduino uno compatible wifi board based on esp8266ex. 2017. Acesso em: 05/12/2018. Disponível em: <<<https://wiki.wemos.cc/products:d1:d1>>>. Citado 2 vezes nas páginas 33 e 34.

Anexos

ANEXO A – CÓDIGO FONTE DO MICROCONTROLADOR

```
// Programa para leitura do Módulo MPU 6050  
// Autor: Eduardo-sr
```

```
//Carrega as bibliotecas  
#include<Wire.h>  
#include <I2Cdev.h>  
#include <MPU6050.h>
```

```
//Endereco I2C do MPU6050  
const int MPU=0x68;
```

```
//Variaveis para armazenar valores dos sensores  
double AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;  
char command;  
int test = 0;  
int c = 0;
```

```
void setup()  
{  
Serial.begin(115200);  
Serial.print("Digite o comando.(0,1,2 ou 3)&\n");  
delay(1000);  
Wire.begin();  
do{  
if (Serial.available()) {  
  
command = Serial.read();  
  
if(command == '0'){  
Serial.print("Comando 0 recebido.\n");  
c = 0;
```

```
test = 1;
delay(2);
}
else if(command == '1'){

    Serial.print("Comando 1 recebido.\n");
c = 1;
test = 1;
delay(2);
}
else if (command == '2'){
Serial.print("Comando 2 recebido.\n");
c = 2;
test = 1;
delay(2);
}
else if (command == '3'){
Serial.print("Comando 3 recebido.\n");
c = 3;
test = 1;
delay(2);
}
else{
Serial.print("Comando invalido. Tente outro.\n");
delay(1000);
test = 0;
}
}

}while(test == 0);
Wire.beginTransmission(MPU);
Wire.write(0x6B);

//Inicializa o MPU-6050
Wire.write(c);
Wire.endTransmission(true);
}
void loop()
{
```

```
Wire.beginTransmission(MPU);
Wire.write(0x3B); // starting with register
0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false);

//Solicita os dados do sensor
Wire.requestFrom(MPU,14,true);

//Armazena o valor dos sensores nas variaveis correspondentes
AcX=Wire.read()<<8|Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
AcY=Wire.read()<<8|Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
AcZ=Wire.read()<<8|Wire.read(); //0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
Tmp=Wire.read()<<8|Wire.read(); //0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
GyX=Wire.read()<<8|Wire.read(); //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
GyY=Wire.read()<<8|Wire.read(); //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
GyZ=Wire.read()<<8|Wire.read(); //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)

//Envia valor X do acelerometro para a serial
Serial.print(AcX);
Serial.print("\t");

//Envia valor Y do acelerometro para a serial
Serial.print(AcY);
Serial.print("\t");

//Envia valor Z do acelerometro para a serial
Serial.print(AcZ);
Serial.print("\t");

//Envia valor da temperatura para a serial
//Calcula a temperatura em graus Celsius
Serial.print(Tmp);//(Tmp/340.00+36.53)
Serial.print("\t");

//Envia valor X do giroscopio para a serial
Serial.print(GyX);
Serial.print("\t");
```

```
//Envia valor Y do giroscópio para a serial  
Serial.print(GyY);  
Serial.print("\t");
```

```
//Envia valor Z do giroscópio para a serial  
Serial.println(GyZ);  
//Serial.print("\t");  
//Serial.println();  
// delay(100);
```

```
}
```

ANEXO B – CÓDIGO DO PROGRAMA EM PYTHON

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""

Created on Fri Jun 29 09:31:48 2018

@author: eduardo-ssr
"""

import sys
import serial
import glob
import os
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from drawnow import drawnow
from time import sleep
from datetime import datetime

read1 = []
read2 = []
read3 = []
read4 = []
read5 = []
read6 = []
read7 = []

acelX = []
acelY = []
acelZ = []
gyroX = []
gyroY = []
gyroZ = []

constant_Calib_Acel = (16384/9.81)
constant_Calib_Gyro = 131

t=0
```

```
os.system('clear')
print ("=====Lista_de_dispositivos_USB=====")
serial_ports = glob.glob('/dev/ttyUSB*')
while(len(serial_ports)==0):
    print ("Conecte_o_Arduino...\n")
    sleep(5);
    serial_ports = glob.glob('/dev/ttyUSB*')

for i in range(len(serial_ports)):
    print (i, " - ", serial_ports[i])
    port = input("Escolha_a_porta_do_Arduino_(e.g._0):_")

ser = serial.Serial(serial_ports[int(port)], 115200,timeout=1)

def main():
    global contador
    global t
    global constant_Calib_Acel
    global constant_Calib_Gyro

    if ser.is_open:
        print("Comunicacao_Serial_Estabelecida.\n")
    else :
        print("Erro_na_Comunicacao_Serial.\n")

    while(t==0):
        sleep(3)
        print(ser.readline().decode("utf-8"))
        print("\n")
        print("Intervalos_de_leitura : \n")
        print("0---->+-250_deg/s_e+-2g\n")
        print("1---->+-500_deg/s_e+-4g\n")
        print("2---->+-1000_deg/s_e+-8g\n")
        print("3---->+-2000_deg/s_e+-16g\n")

        command = (input("Escolha_o_intervalo_de
                            leitura_(e.g._0):_"))

        sleep(1.8)

        if command == '0':
            ser.write(b'0')
            constant_Calib_Acel = (16384/9.81)
            constant_Calib_Gyro = 131
            t = 1
        elif command =='1':
            ser.write(b'1')
```

```

constant_Calib_Acel = (8192/9.81)
constant_Calib_Gyro = 65.5
#sleep(2)
t = 1
elif command == ('2'):
    ser.write(b'2')
    constant_Calib_Acel = (4096/9.81)
    constant_Calib_Gyro = 32.8
    #sleep(2)
    t = 1
elif command == ("3") :
    ser.write(b'3')
    constant_Calib_Acel = (2048/9.81)
    constant_Calib_Gyro = 16.4

    t = 1
else:
    print("Comando invalido , tente outro .\n")
    t = 0

input("Pressione ' Enter ' para iniciar a leitura e\nCtrl+C para pausar : \n")

while(True):
    try:
        line = ser.readline().decode("utf-8")

        print(line)
        try:
            entry = line.split("\t")
            AcX = np.float(entry[0])
            AcY = np.float(entry[1])
            AcZ = np.float(entry[2])
            temp = np.float(entry[3])
            Gx = np.float(entry[4])
            Gy = np.float(entry[5])
            Gz = np.float(entry[6])

            #entre --20 m/s^2 e + 20 m/s^2
            ACX = AcX/constant_Calib_Acel

            #entre --20 m/s^2 e + 20 m/s^2
            ACY = AcY/constant_Calib_Acel

            #entre --20 m/s^2 e + 20 m/s^2

```

```

ACZ = AcZ/constant_Calib_Acel

Temp = temp/340.00 + 36.53

# entre +250deg/s e -250deg/s
GX = Gx/constant_Calib_Gyro
# entre +250deg/s e -250deg/s
GY = Gy/constant_Calib_Gyro
# entre +250deg/s e -250deg/s
GZ = Gz/constant_Calib_Gyro

read1.append(AcX)
read2.append(AcY)
read3.append(AcZ)
read4.append(Gx)
read5.append(Gy)
read6.append(Gz)
read7.append(temp)

acelX.append(ACX)
acelY.append(ACY)
acelZ.append(ACZ)
gyroX.append(GX)
gyroY.append(GY)
gyroZ.append(GZ)

except (ValueError):
    print("Erro_de_valor.")
    pass

except (KeyboardInterrupt):
    now = datetime.now()
    print ("Voce_pressionou_Ctrl+C_para_interromper
este_programa!_Seus_dados_foram_salvos
em '_Dados_%s.csv'%"%(str(now)[: - 7]))

ser.close()

plt.plot(acelX, "-r")
plt.plot(acelY, "-g")
plt.plot(acelZ, "-b")
plt.xlabel('Tempo_(ms)');
plt.ylabel('Aceleracao_(m/s^2)');
plt.show()

plt.plot(gyroX, "-r")
plt.plot(gyroY, "-g")

```

```
plt.plot(gyroZ, "-b")
plt.xlabel('Tempo_(ms)');
plt.ylabel('Velocidade_Angular_(deg/s)');
plt.show()

x = np.vstack((read1, read2, read3, read4, read5, read6,
               read7))
y = np.vstack((acelX, acelY, acelZ, gyroX, gyroY, gyroZ))

np.savetxt('Dados_Brutos_%s.csv'%(str(now)[:-7]),
           np.transpose(x), delimiter=';')
np.savetxt('Dados_Fisicos_%s.csv'%(str(now)[:-7]),
           np.transpose(y), delimiter=';')

break
if __name__ == "__main__":
    main()
```