UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

PROJETO FINAL DE CURSO EM ENGENHARIA FÍSICA II

CONSOLE SEM FIO PARA MOVIMENTO DE OBJETOS 3D VIRTUAIS

BRUNO THOMAZI ZANETTE

Orientador: Milton Andre Tumelero

29 de ago. de 2023

1. INTRODUÇÃO

Modelos 3D em ambientes virtuais têm a capacidade de expandir nosso entendimento sobre certos objetos e os fenômenos que os cercam, seja no desenvolvimento de peças e instrumentação industrial, efeitos visuais ou ainda para fins de educação. Para corroborar esta afirmação temos o artigo “Interactive 3D Visualization of Chemical Structure Diagrams Embedded in Text to Aid Spatial Learning Process of Students”[1] que conclui que “Em termos de habilidade espacial, o aplicativo, que proporciona modelos 3D de estruturas químicas, ajuda os alunos significativamente a imaginar como um composto ficaria de uma perspectiva diferente (orientação espacial)”. Todavia, ferramentas que permitem o controle de modelos 3D, com Blender, Maya, Houdini, são de difícil uso para visualização final, já que estão limitadas ao uso de mouse e teclado exigindo conhecimentos de diferentes atalhos para rotacionar e transladar o objeto, afinal estamos tentando mover um objeto 3D com uma ferramenta 2D (mouse). Logo, um console 3D seria mais intuitivo, mais imersivo e poderia ser melhor aplicado nas áreas de uso previamente citadas.

Soluções diversas já foram pensadas para esse problema, um exemplo emblemático foi com controle do videogame Nintendo Wii, o Wii Remote, ele consegue calcular a rotação e posição do controle com acelerômetro, giroscópio e sensor infravermelho, enquanto os dados são enviados por bluetooth. O Wii Remote funciona bem, porém alguns erros ocorrem principalmente em situações que o emissor de infravermelho não aponta para o sensor. A necessidade desse sensor se dá principalmente por causa de jogos que envolvem mirar, onde a posição exata de onde se aponta para tela é necessária.

Outro projeto que tem uma abordagem similar mas que busca resolver um problema diferente é o “*A smart foam football that tracks your throws*” [2]. Com o objetivo de acompanhar a velocidade do movimento de uma bola de football ele utiliza um sensor de acelerômetro e giroscópio MPU-6050 e envia os dados por Wi-Fi por uma placa NodeMCU. Uma diferença fundamental é que o projeto possui dois MPUs, um focado em medir a força centrípeta e outro que instalado no centro de rotação da *foam ball* buscando medir a velocidade da bola. O modo de comunicação desse projeto é FTP (File Transfer Protocol) através de Wi-Fi, que comparado ao bluetooth consegue enviar muito mais dados em um mesmo intervalo de tempo.

Embora este trabalho tenha foco na discussão e desenvolvimento de um console para controle de modelos 3D em ambientes virtuais, é importante entender que tais tecnologias podem ser facilmente generalizadas para outras aplicações importantes, como na robótica. Um exemplo interessante são as cirurgias robotizadas, onde o médico realiza o procedimento em um ambiente virtual controlando braços mecânicos via consoles como estes discutidos acima. Conforme aprofundamos nossa pesquisa sobre a extração de movimento a partir dos sensores acelerômetro e giroscópio, ficou evidente que reconstruir a translação com base nesses sensores é uma tarefa extremamente desafiadora. Esta dificuldade advém principalmente da complicação em eliminar de forma precisa os efeitos da gravidade, o que resulta no fenômeno conhecido como deriva. No entanto, no que diz respeito à rotação, constatamos que é possível obter informações precisas, com cada sensor contribuindo de forma complementar aos outros, resultando na determinação de uma posição angular bastante exata e com taxas de comunicação suficientemente rápidas para uma experiência imersiva, mesmo quando o controle é movimentado de maneira complexa e com acelerações consideravelmente altas. A abordagem deste projeto, utilizando a interface Python-Blender de comunicação, permitiu obter resultados consistentes a partir dos dados dos sensores giroscópio e acelerômetro.

2. OBJETIVO

Desenvolver um console que utiliza acelerações lineares, medidas com um acelerômetro e velocidades angulares, medidas com um giroscópio, para determinar as posições espacial e angular de um objeto 3D, imerso em um ambiente virtual de computador. Entre os objetivos mais específicos deste trabalho estão:

- Preparar um ambiente composto por hardware e software que sejam de fácil e amigável operação dos usuários que não têm perfil técnico para interação com modelos 3D;
- Utilizar tecnologias livres, para que o projeto possa ser facilmente reproduzido em outros lugares, permitindo um avanço mais rápido deste tipo de tecnologia;
- Determinar as velocidades e posições angulares utilizando os sensores presentes no giroscópio;
- Determinar as velocidades lineares utilizando as acelerações obtidas com o acelerômetro, removendo, na medida do possível, influências da gravidade;
- Preparar um hardware que forneça as medidas necessárias;

- Preparar uma interface em python para aquisição, tratamento dos dados e submissão para outros softwares de apresentação gráfica.

3. METODOLOGIA

3.1 DINÂMICA DE MOVIMENTO

Podemos descrever a posição de um objeto através da medida das acelerações do console utilizando as equações que regem a mecânica Newtoniana, demonstrada na Eq. 1. Considerando que o objeto está inicialmente em estado de repouso ($v_0 = 0$), o deslocamento vai depender unicamente da aceleração, como indicado na Eq 2.

$$d(x,y,z) = \frac{a(x,y,z)t^2}{2} + v_0 t + d_0$$

Eq. 1: equação do deslocamento da mecânica newtoniana.

$$d_1 = \frac{a_0 t^2}{2} + d_0$$

Eq. 2: A posição logo após o repouso é definida pela aceleração.

$$d_{n+1} = d_n + (v_{n-1} + a_n t)t + \frac{a_n t^2}{2}$$

Eq 3: generalização da Equação 2 para qualquer posição.

Transformar esta teoria em uma realidade prática é mais difícil que parece, por alguns motivos: (i) Problemas teóricos, como deslocamento calculado a partir da aceleração vai naturalmente se afastar do valor real, decorrente da incapacidade de descontar a contribuição da gravidade e terá que ser recalibrado no ponto (0,0,0) em repouso (ii) Problemas instrumentais, como a existência de um ruído intrínseco na medição dos parâmetros. Outro ponto de atenção é a alta precisão do sensor de aceleração, 16 bits de 0 a 2g, é esperado que essa alta precisão irá produzir sinais irreais de variação da aceleração mesmo em repouso, que podem ser reduzidos através de filtros.

Com o advento dos smartphones, muitos estudos já foram realizados sobre como utilizar sensores embarcados para colher informações sobre a posição e movimento dos telefones. Em uma publicização dos achados do grupo que desenvolveu o *Sensor Fusion*

Motion, usado em muitos celulares android, é discutido as dificuldades de se calcular a posição utilizando somente o acelerômetro e o giroscópio. O gráfico à esquerda na Figura 1 mostra um *drift* de 20 cm em apenas um segundo devido a dupla integração de ruído da aceleração, primeira integração para a velocidade e a segunda integração para a posição. A direita apresenta o problema da remoção da aceleração da gravidade para determinar a aceleração linear do corpo. Assumindo um erro de 1 grau na determinação da posição angular (com o giroscópio), que é utilizado para remover a contribuição da gravidade da aceleração linear, é esperado um erro propagado na posição linear de cerca de 8,5 metros (deriva) em apenas 1 segundo. Portanto, o autor do trabalho conclui que qualquer tipo de deslocamento linear deve ser estimado com muito cuidado.

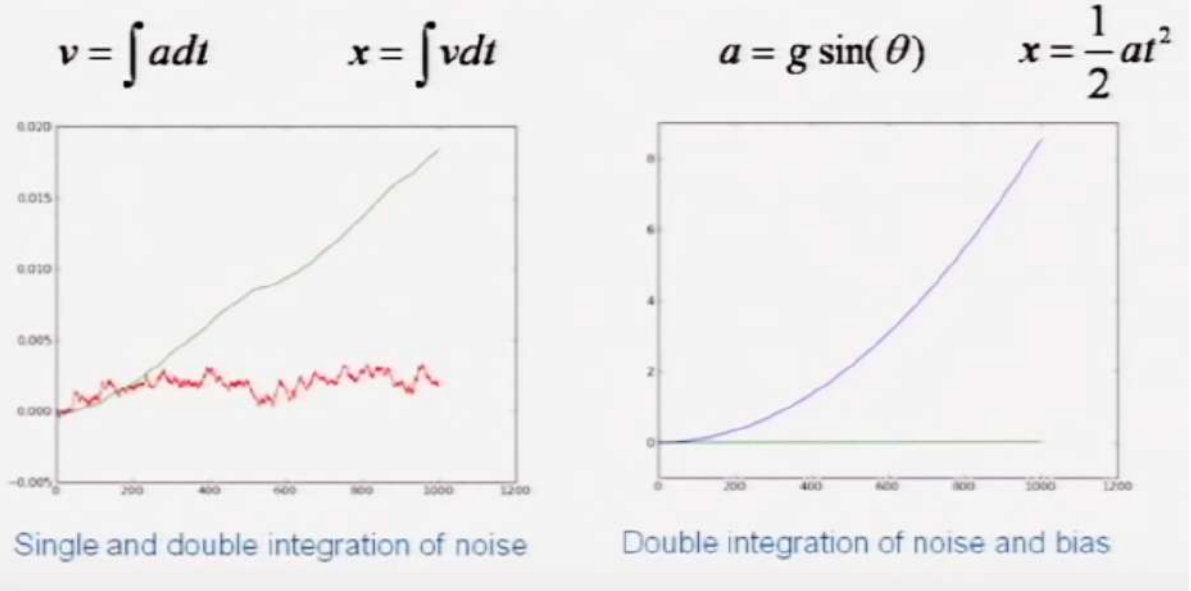

Figura 1: a esquerda integração de ruído do acelerômetro causa drift, a direita erro ao calcular translação quando se estipula 1 grau de inclinação incorreto no desconto da gravidade. Capturada através de captura de tela da palestra disponível em: https://www.youtube.com/watch?v=C7JQ7Rpwn2k&ab_channel=GoogleTechTalks .

A solução para este problema foi adicionar mais sensores que dependam de propriedades físicas diferentes, e que possam produzir um efeito de "âncora" definindo uma direção absoluta no espaço. Em um outro trabalho, os autores conseguiram desenvolver um produto interessante e similar ao que está sendo discutido aqui, é o chamado *Navigation Soft- and Hardware Solution* de Jan Zwiener, que conta com barômetro, GPS, acelerômetro, giroscópio e magnetômetro, com os dados sendo processados com um filtro tipo Kalman. Como o projeto é fechado não foi possível obter informações sobre o funcionamento em nível

de software, sabemos somente que as rotinas dos filtros Kalman foram escritas em C e C++ em um microcontrolador com arquitetura ARM/ Cortex-M4. No caso do hardware há mais informações disponíveis, como por exemplo uma antena de GPS é do tipo Helix, como pode ser visto na Figura 2 [3].

Assumindo que o eixo Z é normal à superfície da terra e o plano XY é perpendicular a este eixo. Pode-se intuir que um Barômetro poderia indicar variações no eixo Z sempre em relação à posição inicial (nível do mar), enquanto o GPS possibilita dados de velocidade e posição com baixa sensibilidade no eixo XY. Um magnetômetro ainda pode indicar um vetor no plano XY que é fixo e decorrente do campo magnético do planeta terra. A informação destes sensores é, em geral, processada em nível de hardware, em softwares embarcados como é o caso do Digital Motion Processor™ - DMP da MotionTracking que acompanha o MPU6050.

Figura 2: controle do projeto de Jan Zwiener que inclui acelerômetro, giroscópio, magnetômetro, barômetro dentro da caixa metálica e antena GPS externa, disponível no site http://zwiener.org/projects.html.

3.2 ROTAÇÃO

Para o processamento das posições angulares foi utilizado o sistema numérico de quatérnions. Este sistema é baseado em números complexos e foi primeiramente descrito em 1843 pelo matemático irlandês Sir William Rowan Hamilton. A notação usual para descrição de rotações é baseado no teorema de rotação de Euler, que mostra que é possível descrever qualquer rotação 3D como uma rotação 2D de deslocamento angular θ em torno de uma direção no espaço 3D, descrita pelo vetor **r**. O quatérnio é uma forma mais simples e

consolidada para representar rotações, também com 4 variáveis, neste caso chamadas de w, x, y, z. Em uma conexão direta entre a notação de Euler e de quatérnio, podemos atribuir os números x, y, z a determinação do vetor r, enquanto que w está diretamente relacionado com o deslocamento angular θ. Este tipo de notação baseada em quatérnio é muito utilizadas para representar rotações no espaço tridimensional, pois evitam o problema de *gimbal lock* (onde uma dimensão de rotação é perdida ao alinhar os eixos de rotação) e diminuem o número de cálculos necessário para interpolar entre posições angulares.

A escolha dos quatérnions foi viabilizada pelo fato de o software Blender ser capaz de processar este tipo de coordenada. O DMP processa automaticamente as rotações e as descreve em forma de quatérnions, tornando-se assim a melhor opção para a descrição precisa das rotações 3D no contexto deste projeto.

3.3 HARDWARE

Partes necessárias para um primeiro protótipo:

- Módulo acelerômetro-giroscópio MPU6050;
- Placa microcontroladora modelo NodeMCU, com módulo Wi-Fi e precessador esp32;
- Fios e protoboard para conexão;
- Conector com sua respectivas 9V.

O NodeMCU e o módulo MPU-6050 juntos da protoboard, Figura 3, têm dimensões da ordem de 7,9 x 5,4 x 1,7 cm e podem ser visualizados na Figura 3. Para alimentação foi utilizada uma bateria recarregável 9 V.

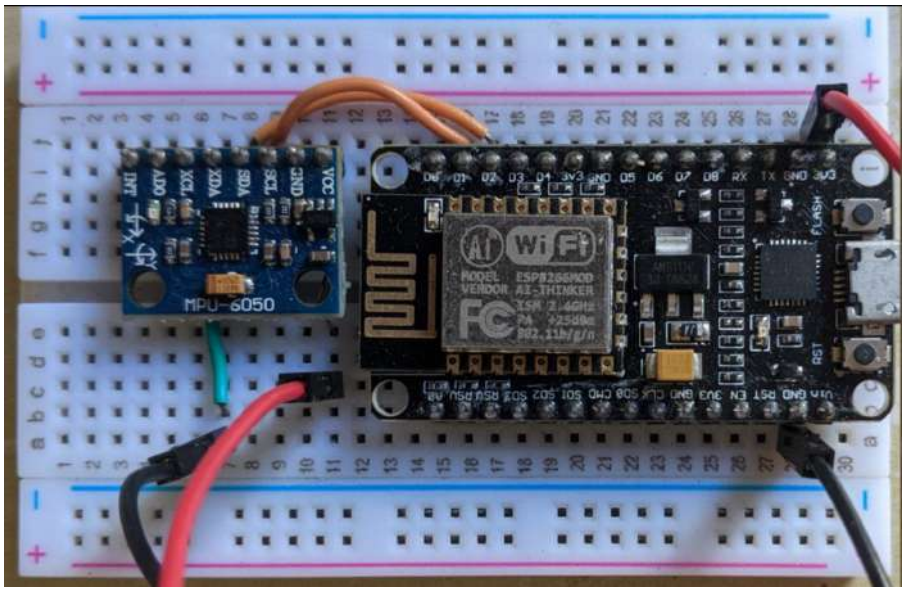

Figura 3: protótipo do circuito com portas D1 e D2 controlando o MPU-6050 por comunicação I2C, acervo pessoal.

3.3.1 Módulo MPU-6050

O MPU-6050 é um integrado que inclui módulo de acelerômetro, giroscópio e sensor de temperatura em um único chip, possui ainda função de calibração. Por ser uma solução compacta, de baixo custo, acessível e bem documentada, este chip tem sido amplamente utilizado em aplicações de robótica, drones, controles de realidade virtual e outras aplicações que necessitem medir aceleração, inclinação e rotação. A disposição dos sensores neste módulo permite adquirir dados nos 3 eixos (x, y, z) que podem ser usados para determinar a orientação e movimento de um objeto em três dimensões.

A calibração do módulo se dá através do sensor de temperatura, já que sensores do tipo MEMS (Sistemas Microeletromecânicos) de silício tem como fonte mais comum de deriva a sensibilidade à variações de temperatura. A calibração pode ser feita por meio de algoritmos de compensação que levam em conta as características específicas do giroscópio e acelerômetro e as variações ambientais relevantes [4].

Em termos gerais, o sensor acelerômetro é composto por uma massa de prova que fica suspensa dentro do integrado. Quando ocorre aceleração ao longo de um determinado eixo, a massa de prova é submetida a uma força inercial na direção oposta, fenômeno que fica explícito na Figura 4.

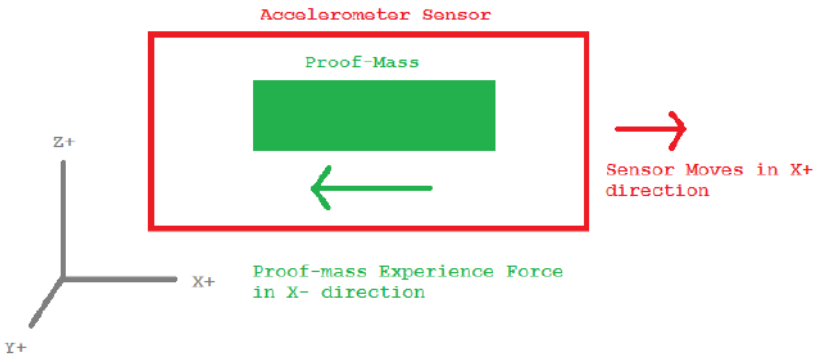

Figura 4: diagrama conceitual do acelerômetro, reproduzida de artigo do engineersgarage.com.

No caso do acelerômetro MEMS, uma massa de prova é suspensa por *cantilever* de polisilício, onde fica livre para se deslocar em uma, duas ou três dimensões quando

submetida a uma força externa. A massa de prova se move entre placas fixas que atuam como eletrodos de um capacitor. A mudança na distância entre a massa de prova e as placas fixas é proporcional à aceleração ao longo desse eixo. Isso produz uma mudança proporcional na capacitância do sistema, que é detectada por um sensor de alta precisão.

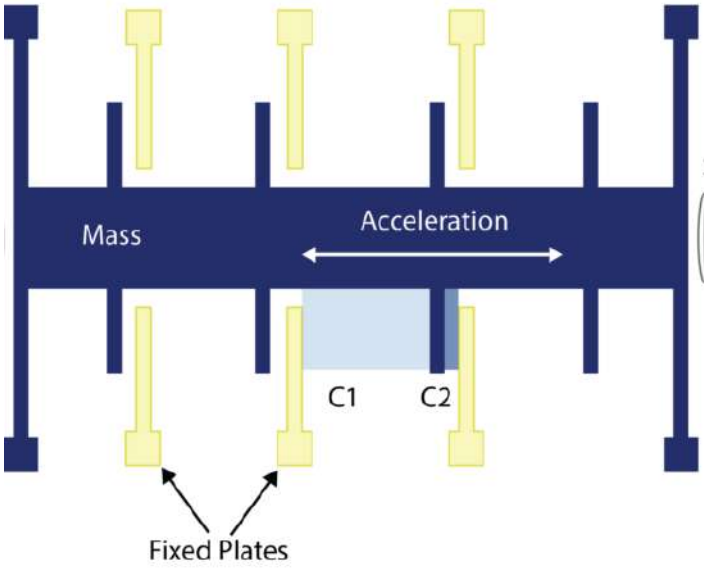

Figura 5: diagrama de funcionamento do acelerômetro MEMS, reproduzida de artigo do engineersgarage.com.

Já os giroscópios são utilizados para detectar momento angular e são construídos utilizando o efeito Coriolis. Giroscópios do tipo MEMS, possuem uma massa de prova dividida em quatro partes, suspensa por massas fixas em seu centro e que oscilam continuamente em um plano horizontal. Quando o giroscópio é rotacionado, a força de Coriolis atua sobre as massas de prova, fazendo com que elas se desviem em direções opostas ao longo do eixo da força de Coriolis, o que causa uma mudança na capacitância entre elas. Essa mudança é detectada por um sensor de alta precisão e convertida em um sinal de tensão proporcional, como exemplo temos a Figura 6 onde as placas M1 e M3 se movem para cima e para baixo, fora do plano horizontal.

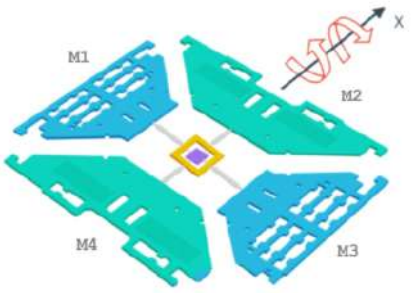

Figura 6: diagrama de funcionamento do giroscópio MEMS, reproduzida de artigo do engineersgarage.com.

O MPU-6050 pode ser facilmente conectado a uma placa de controle, como Arduino ou NodeMCU, e ser acessado por meio de bibliotecas próprias para interpretar os dados dos sensores. Além disso, o MPU-6050 também inclui um processador de movimento embutido que pode realizar cálculos que combinam as leituras do acelerômetro e giroscópio para obter uma representação mais precisa da orientação e aceleração do objeto. Na Figura 7 podemos ver o *PINOUT* do módulo, que além da alimentação e dos pinos I2C que estão sendo utilizados observa-se os pinos XDA e XCL que fazem interface I2C com outros sensores e o AD0 que é utilizado para trocar o endereço do I2C. O pino INT pode ser utilizado para indicar que uma nova medida está disponível.

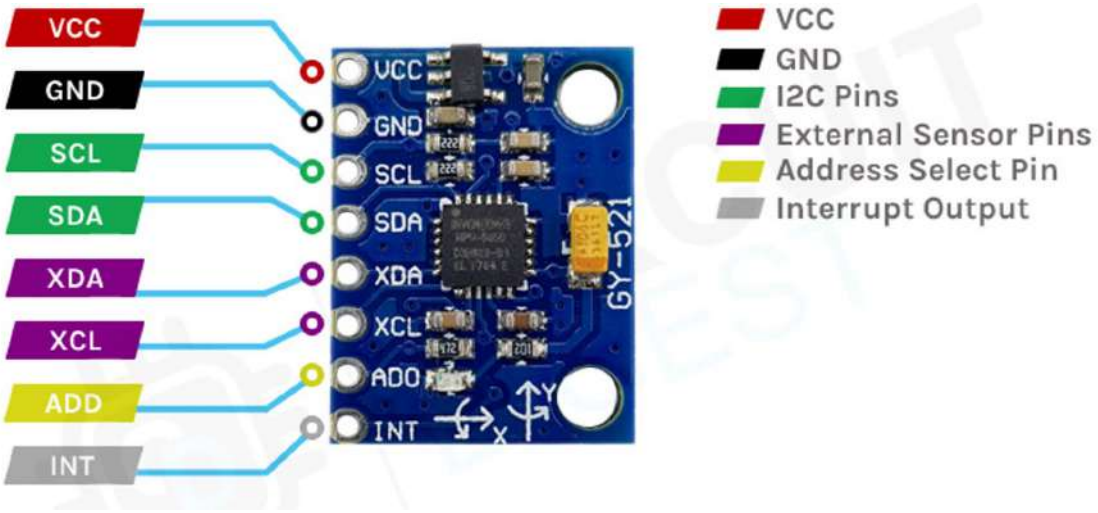

Figura 7: PINOUT do MPU-6050 disponível no site do Circuit Digest

Grandeza	Unidade	Valor
Tensão de operação	Volts (V)	3 a 5
Conversor AD	Bits	16

Faixa do acelerômetro	g (9,8 m/s²)	±2, 4, 8, 16
Faixa do giroscópio	°/s	±250, 500, 1000, 2000
Faixa temperatura	°C	-40 a 80
Dimensões	mm	20 x 16 x 1

Tabela 1: dados do módulo MPU-6050, extraídos do datasheet.

Ainda que o MPU disponha de um Processador de Movimento Digital (Digital Motion Processor™ - DMP) da MotionTracking, é importante destacar que esse processador integrado e sua utilização é consideravelmente complexa devido à falta de informações detalhadas e documentação do produto, conforme evidenciado com pesquisa [5]. Essa falta de documentação aprofundada tem sido um desafio significativo para os desenvolvedores interessados em aproveitar ao máximo o potencial do DMP, ainda assim existem algumas bibliotecas disponíveis que viabilizam algumas operações do DMP. Entretanto, ainda é importante mencionar que determinadas funcionalidades, especialmente relacionadas à detecção de gestos, parecem ter sido reservadas apenas para clientes específicos e uso interno pela empresa detentora dos direitos do DMP.

Neste trabalho, foi utilizada a biblioteca "I2C Device Library", licenciada sob a licença livre do MIT. Com essa biblioteca é possível receber os valores de saída provenientes do DMP com facilidade. As saídas disponíveis são:

- OUTPUT_READABLE_QUATERNION: Essa opção permite visualizar os componentes reais do quaternion em um formato [w, x, y, z], que foi o formato utilizado para descrever rotação;
- OUTPUT_READABLE_WORLDACCEL: Ao habilitar essa saída, é possível visualizar os componentes da aceleração com a gravidade removida e ajustada para o referencial do mundo. Essa opção é útil quando utiliza-se comandos no referencial do mundo, como é o caso do projeto;
- OUTPUT_READABLE_EULER: A ativação desta saída proporciona a exibição dos ângulos de Euler (em graus) calculados a partir dos quatérnions. É importante observar que os ângulos de Euler podem ser afetados pelo *gimbal lock*;
- OUTPUT_READABLE_YAWPITCHROLL: Essa saída permite visualizar os ângulos de rotação (yaw/pitch/roll em graus) calculados a partir dos quatérnions. Essa escolha também está sujeita ao efeito do *gimbal lock*;

- OUTPUT_READABLE_REALACCEL: Ao habilitar essa opção, é possível visualizar os componentes da aceleração com a gravidade removida. O referencial de aceleração fornecido por essa saída não é compensado em relação à orientação, o que significa que o eixo +X sempre corresponderá ao +X conforme detectado pelo sensor, porém, tentando retirar os efeitos da gravidade.

Também é possível receber as medidas sem processamento (*raw)* do acelerômetro e giroscópio. Foi percebido um atraso na frequência de produção de dados quando se usa o DMP. A seleção das saídas foi feita com base nos requisitos e objetivos específicos do projeto, buscando a melhor adequação para análise e processamento dos dados gerados pelo MPU, permitindo assim a obtenção de informações relevantes para a aplicação em questão.

3.3.2 Placa de controle NodeMCU

A placa de controle NodeMCU é uma placa de desenvolvimento que combina um microcontrolador esp8266 com módulo Wi-Fi integrado, isso a torna uma opção ideal para projetos de controle sem fio de IOT. Por possuir uma comunidade de desenvolvedores muito ativa, existe uma extensa documentação sobre a placa.

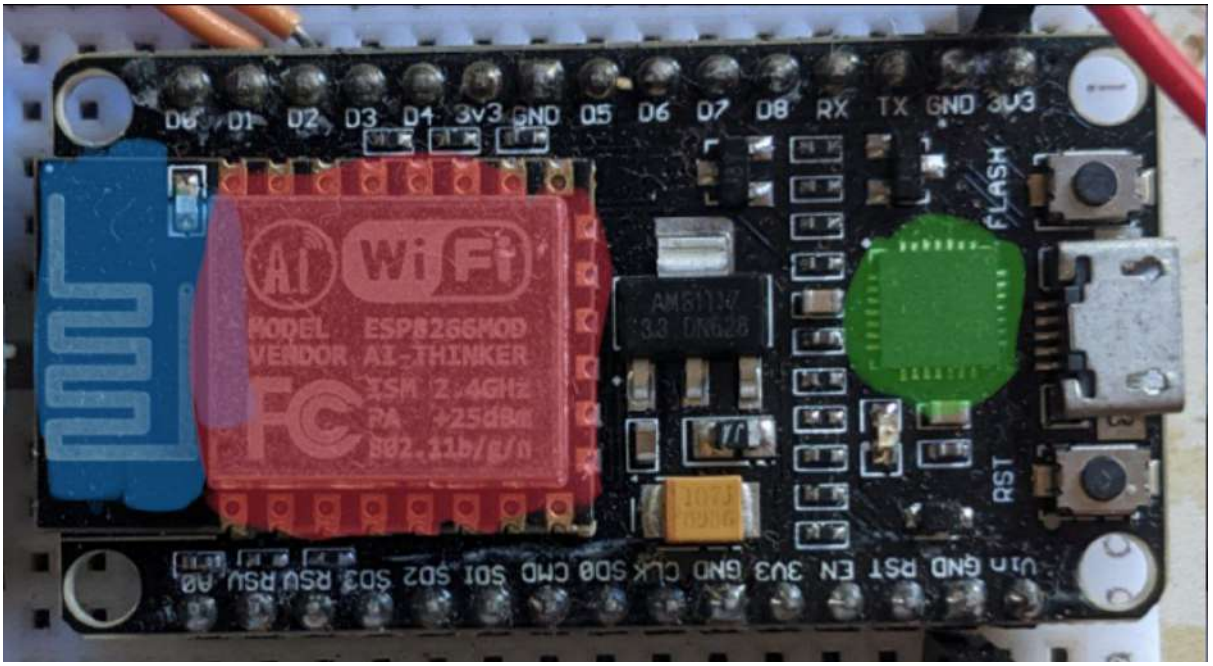

Figura 8: fotografia NodeMCU, acervo pessoal

Devido ao chip integrado a placa está limitada a outputs de 3,3V. Alguns componentes importantes estão coloridos na Figura 8:

- Azul: Antena de 2,4 GHz;
- Vermelho: chip ESP8266 projetado pela empresa chinesa Espressif Systems, possui processador de 32 bits com clock de 80 MHz, memória flash integrada de 32 KB para instruções e 80 KB para *user data* (a placa possui 4MB de capacidade de armazenamento externo ao chip), suporta Wi-Fi b/g/n e interfaces de comunicação SPI, I2C e UART;
- Verde: um circuito integrado de conversão de USB para TTL (Transistor–Transistor Logic).

3.4 COMUNICAÇÃO

Dos diferentes protocolos de comunicação de internet possíveis com NodeMCU duas se destacam: Server-Sent Events (SSE) e User Datagram Protocol (UDP). SSE é um protocolo baseado em HTTP que permite a transferência unidirecional do servidor para o cliente, ele opera com taxas de comunicação superiores quando comparados ao padrão FTP (file transfer protocol) [6]. Isso significa que o servidor pode enviar dados ao cliente a qualquer momento, sem que o cliente precise fazer uma requisição explícita, o diagrama de funcionamento pode ser visualizado na Figura 9. O NodeMCU pode ser usado como um servidor web para implementar essa técnica. É importante levar em conta que SSE garante a chegada de todos dados enviados através de respostas dos clientes quando o dado chegou e também garante a ordem de chegada dos dados.

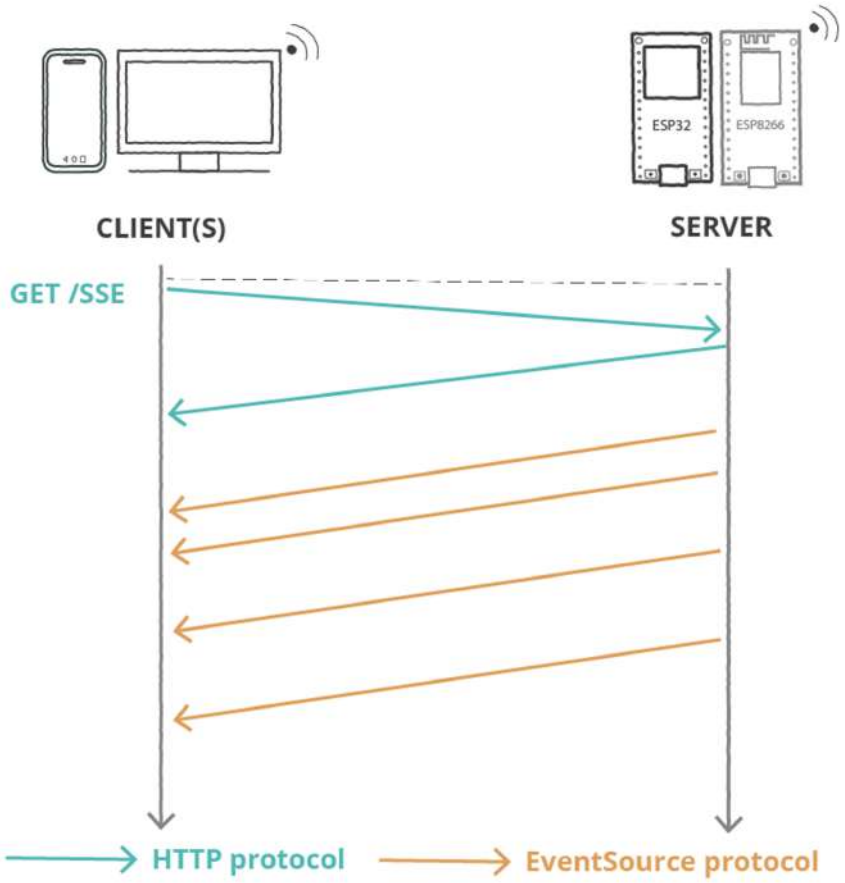

Figura 9: diagrama de funcionamento do SSE, disponível em randomnerdtutorials.com

Já o UDP é um protocolo mais simples que não está preocupado com garantias de chegada ou ordem de chegada das respostas, isso permite um aumento da frequência de dados enviados e simplificação do código, como visto na Figura 10.

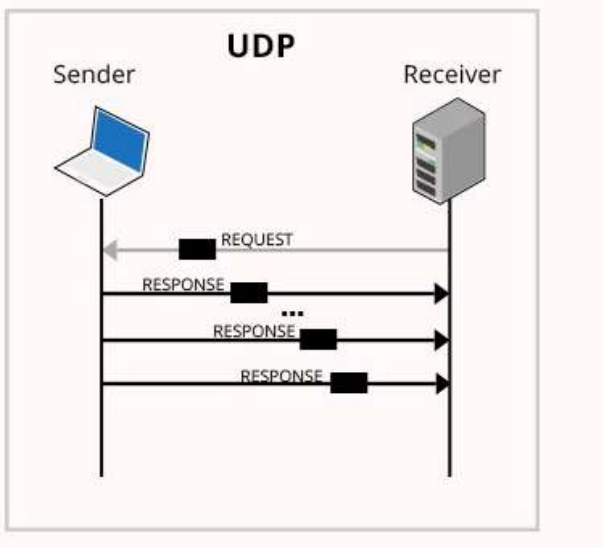

Figura 10: diagrama de funcionamento do UDP, disponível em bizcloud.vn

É relevante enfatizar que, embora o protocolo SSE (Server-Sent Events) tenha sido inicialmente adotado, possibilitando um aumento na taxa de transmissão de dados, sua capacidade se mostrou inadequada para a aplicação final. Em busca de uma alternativa que oferecesse uma comunicação mais eficiente sem aumentar a complexidade, optou-se pela adoção do protocolo User Datagram Protocol (UDP). Com essa mudança, tornou-se dispensável o envio de mensagens "Keep Alive" para manter a comunicação ou a troca de um primeiro pedido (*request*) como requisito inicial.

Por meio do UDP, os pacotes de dados são enviados para um endereço IP específico e uma porta designada. É importante destacar que o protocolo UDP não assegura a ordem das mensagens nem garante a entrega completa, porém, como contrapartida, oferece um aumento notável na taxa de chegada de mensagens. Ao passar de uma frequência média de 270 Hz utilizando o protocolo SSE para aproximadamente 370 Hz com o UDP, observa-se uma melhoria significativa na velocidade de transmissão de dados. Essa escolha proporcionou um aumento substancial na eficiência da comunicação, contribuindo para uma experiência mais fluida e responsiva na interação com o dispositivo.

3.5 SOFTWARE

O software escolhido para realizar o processamento das posições do modelo 3D foi o Blender, pois possui uma aba específica para scripting, Figura 11, essa interface permite que quase qualquer função executada na UI (*User Interface*) possa ser realizada com programação em Python. No canto inferior esquerdo da Figura 11 algumas funções são mostradas em python quando um comando na UI é executado, facilitando o processo de criação de códigos que reproduzem o que o usuário pode fazer.

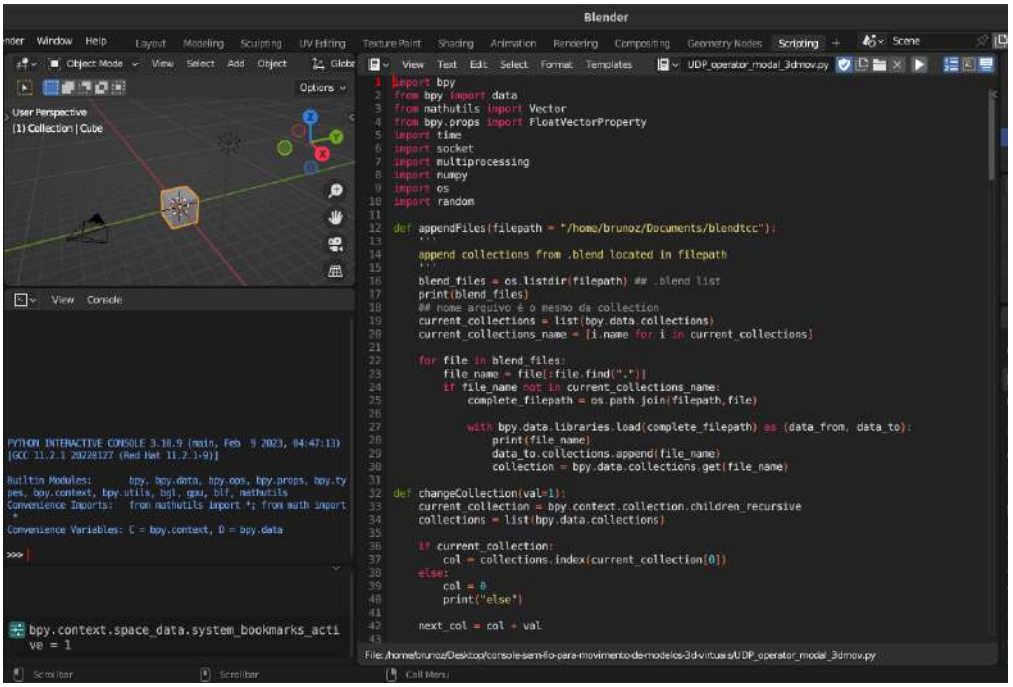

Figura 11: interface de programação do programa Blender.

Quando se executa um programa no Blender, a interface fica parada, aguardando a finalização do programa, assim a forma de fazer um programa operar em tempo real é definindo um operador modal. Segue um exemplo de um operador modal simples no Código 1, dentro da classe `Operator` cada método (`def`) possui sua funcionalidade, o método `execute` e `invoke` rodam somente uma vez quando o programa é executado, a diferença é que `invoke` usualmente é utilizado para preparar o programa para rodar o método `execute`, já que `invoke` também recebe `event`.

O método mais importante é o `modal` que roda em loop continuamente até devolver `return {'CANCELLED'}` ou `return {'FINISHED'}`, onde o primeiro retorna ao estado inicial e o segundo encerra o programa. No caso do seguinte código o programa o `execute` define um timer de 0,01s atrelado a uma janela que salva em `event.type` a *string* `'TIMER'`. Caso a *flag* do timer não esteja pronta o programa identifica se o mouse está sendo clicado com `'LEFTMOUSE'` e `'RIGHTMOUSE'`

```
class Operator(bpy.types.Operator):
    bl_idname = "view3d.modal_operator"
    bl_label = "UDP modal operator"
    def execute(self, context):
        wm = context.window_manager
        self._timer = wm.event_timer_add(0.01, window=context.window)
    def modal(self, context, event):
```

```
        if event.type == 'TIMER':
            print("time!")
        elif event.type == 'LEFTMOUSE':
            return {'FINISHED'}
        elif event.type in {'RIGHTMOUSE', 'ESC'}:
            return {'CANCELLED'}
        else:
            return {'PASS_THROUGH'}
        return {'RUNNING_MODAL'}
    def invoke(self, context, event):
        print("invoke")
```

Código 1: exemplo de operador modal no Blender.

Como o software do console necessita de comunicação por WiFi com UDP a melhor forma de implementar essa execução é com algum tipo de processamento paralelo com python, para isso existem duas formas: *multithreads* ou *multiprocess*.

Em relação à utilização de *threads* e processos em Python, é importante destacar a diferença entre eles em termos de execução paralela. Ao criar uma nova thread, ela não é verdadeiramente paralela, mas sim concorrente ao programa principal, como pode ser visto na Figura 12. Isso significa que ela compartilha a mesma memória e processa em intervalos dentro do programa principal, proporcionando um melhor aproveitamento do processamento em comparação com o formato sequencial.

Por outro lado, o multiprocessamento é verdadeiramente paralelo, o que significa que cada processo é executado em núcleos diferentes do processador, garantindo uma execução simultânea e independente. No entanto, uma desvantagem do multiprocessamento em relação ao multithreading é que não há compartilhamento direto de memória entre os processos. Para contornar essa limitação, é necessário estabelecer um canal de comunicação entre o programa principal e os processos, utilizando um *pipe* para enviar os dados do programa paralelo para o programa principal.

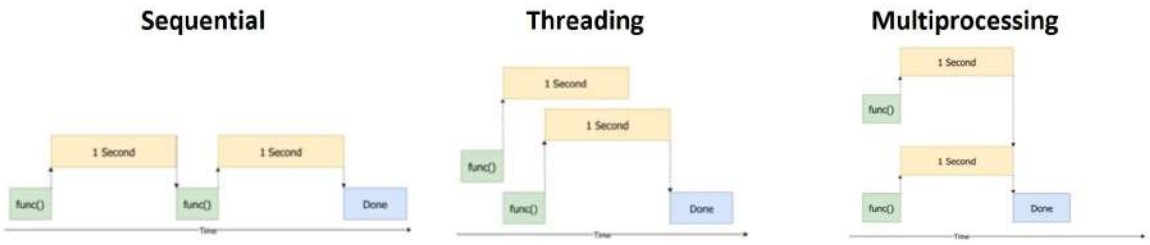

Figura 12: diagrama de diferença entre processamento sequencial, Threading e Multiprocessing. Disponível em site lightbringercap.com

Finalmente, o projeto mistura os conhecimentos desenvolvidos ao longo do curso de engenharia física como: instrumentação, eletrônica, mecânica, modelagem 3D e processamento de dados, em um produto com possibilidade de aplicação tanto em sala de aula como na indústria, quando for necessário controlar analisar com facilidade um modelos 3D em ambientes virtuais.

4. EXECUÇÃO

4.1 SOFTWARE

4.1.1 Instalação das Bibliotecas:

A instalação das bibliotecas necessárias para o código do firmware não é possível do modo usual que se instala bibliotecas .zip na IDE arduino, como é explicado na página github da biblioteca I2Cdev, para este caso é necessário:

1. Fazer o download do arquivo .zip disponível no repositório github i2cdevlib de jrowberg;
2. Mover o arquivo *MPU6050* e *I2Cdev* dentro da pasta que ocorreu o download para o *path* que a IDE arduino busca bibliotecas;
3. Re-escanear as bibliotecas na IDE-Arduino.

4.1.2 Recepção dos Dados

Foram desenvolvidos 3 códigos para realizar a aquisição dos dados do MPU6050 e enviar para o computador via NodeMCU. O primeiro, baseado em comunicação SSE, transmitindo dados sem pré-processamento, o segundo código transmite os dados com o protocolo UDP e dados sem pré-tratamento e o terceiro código e final transmite dados processados na unidade DMP e envio para o computador via UDP. A mudança de SSE para UDP do primeiro para o segundo foi bem justificada pois houve aumento significativo da frequência de chegada de dados de 270 Hz em média com SSE para 370 Hz com UDP. No terceiro código a taxa de transferência média é 100 Hz, a utilização do DMP gera uma maior latência no MPU6050. Uma futura análise pode ser feita para implementar a transferência de dados do DMP utilizando SSE, o que garante a ordem de chegada, visto que no caso de transferências de dados com o DMP, a taxa de comunicação é limitada por esta própria unidade de processamento, e não o protocolo de envio de dados.

Dado que o recebimento de dados é uma tarefa completamente independente do restante do programa e a quantidade de dados influencia diretamente a qualidade da visualização, optar por um processo paralelo para executar a comunicação UDP é uma escolha coerente. Para implementar essa abordagem, o processo paralelo é definido utilizando o código abaixo, no qual a função `multiprocessing.Pipe()` é utilizada para estabelecer as duas extremidades do *pipe*, permitindo a escrita e leitura de dados. O processo paralelo, então, executa a função **`udp`**, utilizando o argumento apropriado para enviar os dados ao programa principal:

```
# multiprocess
w , self.r = multiprocessing.Pipe()
p1 = multiprocessing.Process(target=udp,args=[w])
p1.start()
```

Código 2: definição do pipe para comunicação entre multiprocessos, o segundo processo é definido em p1.

A função **`udp`** foi implementada com um loop infinito para garantir um recebimento contínuo de dados vindos do NodeMCU. É importante notar que, ao rodar o programa mais de uma vez, pode ocorrer um erro relacionado ao uso da porta de comunicação. Para evitar esse problema, foi utilizado o comando `sck.close()` para interromper a conexão UDP após o recebimento dos dados, como visto no Código 3, essa abordagem permite liberar a porta e evitar conflitos ao executar o programa novamente. No entanto, é válido mencionar que essa solução acarreta uma pequena redução na quantidade de dados recebidos, já que a porta precisa ser reaberta após cada recebimento. Outra alternativa encontrada foi simplesmente desligar o computador, o que também permitiria liberar a porta e reiniciar o processo de comunicação sem problemas.

Após a recepção dos dados em formato binário, a função realiza um processamento para transformá-los em uma lista de valores de ponto flutuante (*floats*), armazenados na variável `lista_data`. Essa conversão é realizada usando a função `decode("utf-8")` para transformar os dados em formato de texto (*string*) e, em seguida, separando-os em uma lista com a função `split(",")`, considerando que os valores são separados por vírgula. Uma vez processados, os dados são enviados para o programa principal por meio da extremidade do *pipe* chamada `write`.

```
def udp(write):
```

```
    '''
    process that handles receiving data from nodemcu
    '''
    l=[]
    tim_udp = time.time()

    while 1:
        sck = conectUDP() # UDP_IP = "192.168.4.2",UDP_PORT = 1234
        data,addr = sck.recvfrom(1024)  # Buffer size is 1024 bytes
        sck.close()

        lista_data = data.decode("utf-8").split(",")
        lista_data = [float(f) for f in lista_data]

        write.send(lista_data)
```

Código 3: seção de código que recebe dados do NodeMCU por UDP.

4.1.3. FIRMWARE

O firmware implementado no NodeMCU é responsável por controlar o MPU6050 usando a biblioteca I2Cdev e a classe MPU6050_6Axis_MotionApps20, que utiliza o DMP (*Digital Motion Processor*) da MotionApps v2.0. O objetivo é ler os dados do sensor já processado em quatérnios e aceleração sem gravidade, e enviá-los para um servidor remoto através de uma conexão WiFi com protocolo UDP.

O código se inicia configurando o WiFi do NodeMCU em modo de ponto de acesso (*Access Point*) com as credenciais pré-definidas para permitir a comunicação com o servidor remoto. O endereço IP do NodeMCU é definido como 192.168.4.2, e o *gateway* e máscara de sub-rede são configurados para uma rede local simples.

```
void initWiFi() {
  IPAddress staticIP(192, 168, 4, 2); // IP set to Static
  IPAddress gateway(192, 168, 4, 1);// gateway set to Static
  IPAddress subnet(255, 255, 255, 0);// subnet set to Static

  WiFi.mode(WIFI_AP);// Working mode only as Acess Point

  WiFi.softAP(ssid, password, 2, 0);
  WiFi.config(staticIP, gateway, subnet);
  }
```

Código 4: configuração do WiFi do NodeMCU como AcessPoint.

O sensor MPU6050 junto da porta UDP e o WiFi são inicializados e testados para verificar a conexão, para então o DMP ser inicializado.

```
 setup() {
    initWiFi();
    Udp.begin(port);
    while (!mpu.testConnection()) {mpu.initialize();}

    // load and configure the DMP
    mpu.dmpInitialize();

    mpu.CalibrateAccel(6);
    mpu.CalibrateGyro(6);

    mpu.setDMPEnabled(true);
    }
}
```

Código 5: função setup que roda somente uma vez no NodeMCU.

Através do DMP, os dados de aceleração linear são obtidos, compensados pela gravidade e transformados no referencial do mundo, utilizando os valores de orientação angular em quaternion. Os dados ficam armazenados no `fifoBuffer`, nesse contexto, o FIFO "First-In-First-Out" é utilizado para armazenar um *buffer* de dados do sensor, para que o NodeMCU possa recuperar os dados no seu próprio ritmo, garantindo a eficiência e evitando perdas. Os dados de aceleração no referencial do mundo e os valores do quaternion são convertidos para strings e enviados usando o protocolo UDP com `Udp.write(datastr);`

```
void loop() {
   if (mpu.dmpGetCurrentFIFOPacket(fifoBuffer)) { // Get the Latest

       mpu.dmpGetQuaternion(&q, fifoBuffer);
       mpu.dmpGetAccel(&aa, fifoBuffer);
       mpu.dmpGetGravity(&gravity, &q);
       mpu.dmpGetLinearAccel(&aaReal, &aa, &gravity);
       mpu.dmpGetLinearAccelInWorld(&aaWorld, &aaReal, &q);

       char datastr[300] = "";
       char ax[7], ay[7], az[7];
       strcpy(datastr, dtostrf(aaWorld.x,6,4, ax));strcat(datastr, ",");
```

```
        strcat(datastr, dtostrf(aaWorld.y,6,4, ay));strcat(datastr, ",");
        strcat(datastr, dtostrf(aaWorld.z,6,4, az));strcat(datastr, ",");
        char qw[7], qx[7], qy[7], qz[7];
        strcat(datastr, dtostrf(q.w,6, 4, qw));strcat(datastr, ",");
        strcat(datastr, dtostrf(q.x,6, 4, qx));strcat(datastr, ",");
        strcat(datastr, dtostrf(q.y,6, 4, qy));strcat(datastr, ",");
        strcat(datastr, dtostrf(q.z,6, 4, qz));

        Udp.beginPacket(rip, rport);
        Udp.write(datastr);
        Udp.endPacket();
    }
```

Código 6: Código que roda em ciclo, montando a string que é enviada e enviada por UDP.

4.1.4. FILTROS EMBARCADOS E PROCESSADOS

MPU-6050 utiliza um ADC para transformar dados analógicos em digitais, portanto já existe um filtro passa baixa intrínseco do módulo, já que em um ADC a frequência máxima a ser convertida é a frequência de Nyquist, metade da frequência de *sampling*. O próprio MPU já faz o manejo da frequência de sampling como também do *Digital Low Pass Filter* (DLPF). O DLPF é configurado pelo registrador DLPF_CFG que afeta tanto o acelerômetro quanto o giroscópio, como pode ser observado na Tabela 1 que foi extraída do datasheet do módulo.

DLPF_CFG	Accelerometer (Fs =1kHz)		Gyroscope		
	Bandwidth (Hz)	Delay (ms)	Bandwidth (Hz)	Delay (ms)	Fs (kHz)
0	260	0	256	0.98	8
1	184	2.0	188	1.9	1
2	94	3.0	98	2.8	1
3	44	4.9	42	4.8	1
4	21	8.5	20	8.3	1
5	10	13.8	10	13.4	1
6	5	19.0	5	18.6	1
7	RESERVED		RESERVED		8

Tabela 2: Configuração do filtro passa baixa embarcado através do registrador DLPG_CFG .

Os dados que chegam por WiFi devem ser processados de maneira rápida garantindo o movimento suave com atualização de no mínimo 24 frames por segundo, ou 41,5 milissegundos para cada envio, limitando temporalmente o processamento de dados.

O python possui bibliotecas especializadas para processamento de sinal em tempo real, são essas: Numpy e Scypi. Com elas podemos implementar filtros digitais do tipo *Finite Impulse Response* (FIR), principalmente com o método *scipy.signal.firwin()*, método de filtragem digital que se comparado com um filtro de *Infinite Impulse Response* (IIR) é mais condizente com a aplicação proposta, pois trabalhar com seções de dados permite mais controle do resultado final. Os filtros podem variar de média temporal integrando os sinais como também podemos eliminar ruídos de movimento da mão com filtro passa baixa e evitar influência da rede elétrica com um passa alta.

No último código não houve modificações no DLPF, pois o DMP parece manejar automaticamente esse filtro, todavia, se não for o caso, a configuração padrão é adequada ao projeto. Em relação aos filtros possíveis de serem implementados com Python foi utilizado um filtro de média da biblioteca *numpy* nos últimos cinco dados chegados para detecção de gestos, o processo de detecção de gestos teve resultados incoerentes que necessitam de uma melhor exploração matemática do problema.

4.2 DESIGN ESTRUTURAL

Para o desenvolvimento do design estrutural, foram estabelecidos dois objetivos fundamentais: a necessidade de ser um modelo generalista, capaz de se ajustar a diferentes geometrias, e garantir segurança durante o manuseio para evitar quedas. A escolha de uma geometria generalista naturalmente recai sobre formas primitivas, como esferas, cubos, pirâmides, entre outras. Uma esfera foi adotada como a forma ideal para o controle, pois o objeto virtual pode estar em qualquer ângulo de rotação, portanto, é importante que o controle possua um padrão consistente que se mantenha igual em todas as direções.

Para aprimorar a “pegada” (*grip*) do controle, optou-se por utilizar uma esfera com buracos. Esses buracos foram projetados com formas não circulares, a fim de proporcionar uma melhor adaptação à ponta do dedo, aumentando assim a pegada durante o manuseio.

Essas decisões de design foram tomadas considerando as necessidades de usabilidade, versatilidade e segurança do dispositivo, contribuindo para a efetividade e eficiência de sua operação em diversas situações. Em apresentações informais do controle foi relatado clara

semelhança com estruturas naturais, sendo comum a descrição como *Biophilic design* [7] que parece se adaptar bem a estrutura.

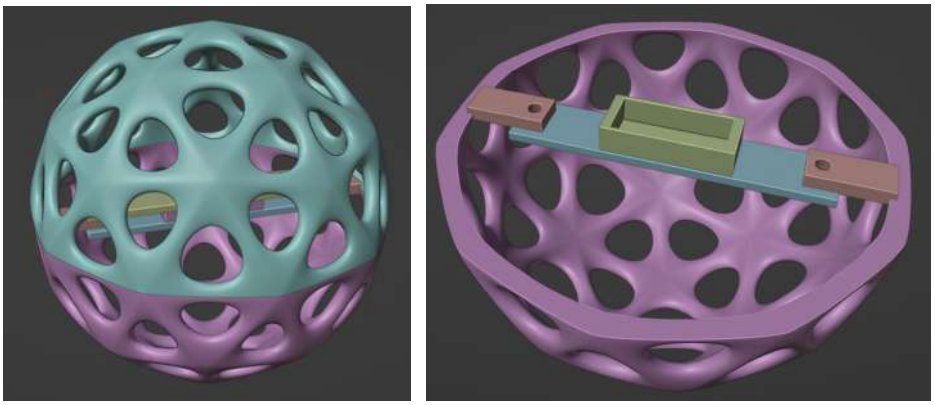

Figura 13: primeira versão do modelo 3D do design estrutural, feito em Blender.

A caixa verde do modelo, à direita da Figura 13, mostra como será pensado o armazenamento da eletrônica do projeto. Para que a ligação seja sólida foram utilizados dois parafusos para conectar as duas semi esferas (azul e roxa) que são impressas separadamente.

A montagem final da esfera com 145 mm de diâmetro é segurada por parafusos e porcas e pode ser visualizados na Figura 14, em questão de estabilidade o formato se mantém porém não é firme, as partes não agem como um objeto maciço, em versões futuras pode-se pensar em encaixes no modelo 3D impresso que fixe melhor a esfera.

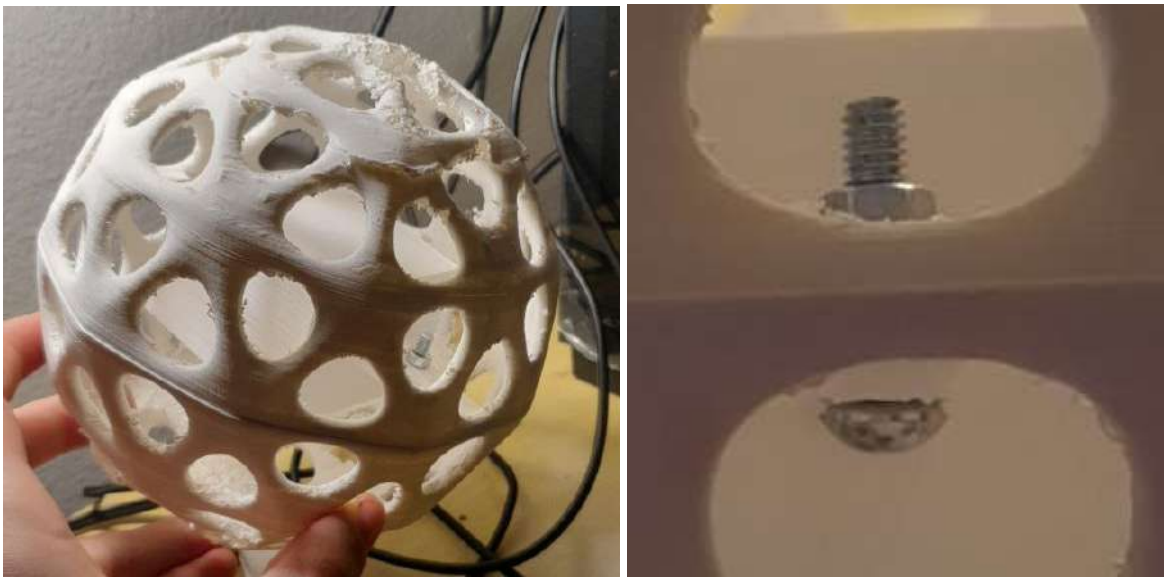

Figura 14: a esquerda montagem final da esfera e a direita demonstração da fixação da esfera com parafuso e porca.

Durante a montagem do projeto, percebemos que o uso de parafusos com porcas não era uma abordagem intuitiva ou fácil, para torná-lo mais conveniente seria melhor substituir as porcas por borboletas, facilitando a aplicação de pressão aos parafusos. Como pode-se observar na Figura 15 a impressão 3D encontrou desafios nos pólos da esfera, principalmente devido a erros de configuração dos suportes. Diversos padrões de suporte foram testados, totalizando quatro tentativas distintas, mas nenhum deles conseguiu concluir a parte mais alta da impressão da meia esfera com sucesso. Essa questão foi identificada como um ponto crucial para futuras melhorias no processo de impressão e montagem da estrutura do projeto.

Figura 15: erro na impressão dos pólos da esfera.

Foi realizado um teste com a esfera envolvendo uma criança de 4 anos sem a presença dos componentes eletrônicos, objetivando avaliar sua capacidade de segurá-la adequadamente. O teste foi concluído com êxito, uma vez que a criança conseguiu segurar a esfera com facilidade usando apenas uma das mãos, não sendo necessário qualquer intervenção para auxiliá-la. Durante o teste, a criança apontou a semelhança da esfera com uma bola de futebol, o que ressalta a sua familiaridade e atração pela forma do objeto. Essa observação é valiosa, uma vez que demonstra que a esfera possui uma usabilidade adequada para crianças dessa faixa etária, o que é relevante para a proposta e aplicação do projeto.

5. CONSIDERAÇÕES FINAIS

Após a conclusão do projeto, uma série de testes foram conduzidos para avaliar a capacidade de translação e rotação de um objeto virtual 3D no ambiente Blender com o

controle. O dispositivo desenvolvido demonstrou ser altamente responsivo a variações de movimento, proporcionando uma representação em tempo real dos movimentos do controle na tela. No entanto, é importante ressaltar que, devido à natureza do Blender como um software de modelagem 3D que não é estritamente orientado a medidas exatas, a avaliação quantitativa da precisão da rotação não foi viável. Uma gravação de demonstração está disponível no YouTube no seguinte link: https://youtu.be/gtrRwuMZdf4, no vídeo é visível um dos problemas comuns do projeto que é a inconsistência da alimentação da bateria, o que acarreta no encerramento da conexão.

No que diz respeito à translação, a situação é mais complexa. Embora testes mais tangíveis pudessem ser executados, como mover um objeto ao longo de um eixo juntamente com uma régua física e comparar com o movimento no Blender, a precisão é rapidamente comprometida por erros que se acumulam. A medição da translação através da aceleração enfrenta duas limitações principais:

- A dificuldade em obter uma medida precisa de aceleração ao desconsiderar totalmente os efeitos da gravidade.
- Os desafios inerentes à integração dupla, que resulta na acumulação de ruídos provenientes do acelerômetro ao longo do tempo.

Dois pontos dificultam a eliminação da gravidade da medida de aceleração, o primeiro, e principal, é que a dificuldade de obter um valor exato e preciso para posição angular do controle com os sensores disponíveis no MPU-6050. Isso é um problema conhecido na teoria e que é apresentado muito bem no *google tech talk* [8] do criador do *Sensor Fusion Motion* David Sachs, onde a dificuldade de estabelecer a posição angular de forma precisa acaba fazendo com que pequenas quantidades de gravidade restem nos valores medidos de aceleração. É importante notar que os movimentos cotidianos ocorrem nas escalas de centímetros e segundos, gerando acelerações na faixa de 0,1 a 10 cm/s^2. Note que a remanescência de 1% do valor da gravidade, que reste após a análise do DMP implica em acelerações parasitas de 9,8 cm/s^2. O que suprime as acelerações implicadas no console pelo usuário.

Uma forma de amenizar esse problema é adicionando mais tipos de sensores, podendo contar com uma variedade maior de dados para caracterizar o movimento. O MPU 9250, por exemplo, é um chip integrado que possui três tipos de sensores com acelerador de 3 eixos, giroscópio de 3 eixos e magnetômetro de 3 eixos.

O segundo ponto que dificulta a eliminação da gravidade é que cada eixo do sensor de aceleração é único e não apresenta exatamente a mesma calibração de 9,8m/s². Esse valor varia também com a temperatura, portanto o correto seria calibrar cada eixo do sensor a cada uso, essa é a solução que muitos drones utilizam, mas que coloca uma dificuldade adicional aos usuários no início do uso.

O MPU 6050 é um acelerômetro de baixo custo e fácil acesso, no entanto, é o mais ruidoso disponível no mercado segundo o compilado disponível no site da empresa TDK [9], demonstrado na Tabela 3.

Product	ICM-20602	ICM-20600	ICM-20601	ICM-20603	ICM-20609	ICM-20690	ICM-20649	ICM-20689	**MPU-6050**	MPU-6500
Accel Sensitivity Error (%)	±1%	±1%	±2%	±1%	±2%	±0.5%	±0.5%	±2%	± 3%	±3%
Accel Nonlinearity (%)	± 0.3%	± 0.3%	± 0.5%	± 0.3%	± 0.25%	± 0.3%	± 0.5%	± 0.5%	± 0.5%	± 0.5%
Accel Noise (µg/√Hz)	100	100	390	100	210	100	285	150	400	300

Tabela 3: comparação entre diferentes sensores de aceleração com base em sensibilidade de erro, linearidade do sensor e ruído.

Segundo a Tabela 3, o valor de densidade ruído do acelerômetro do MPU-6050 medido em $\frac{\mu g}{\sqrt{Hz}}$ (microgramas pela raiz quadrada da frequência em Hertz) é maior em até 4 vezes se comparado com o sensor de menor ruído. Também possui medidas de erro de sensitividade e não linearidade altas, o que contribui para um resultado menos representativo.

5.1 APLICAÇÕES

Primeiramente, está sendo estudado a possibilidade de comercializar este projeto como um serviço ou produto para exposições de museus para possibilitar uma forma mais mais completa de interagir com patrimônio material que não pode ser tocado, seja ele arqueológico ou paleontológico.

O controle também tem a possibilidade de beneficiar outras áreas em que a visualização 3D é importante, como as seguintes:

- Design de produto;
- Arquitetura e engenharia civil;

- Animação e jogos;
- Visualização de dados e análise de negócios;
- Medicina e ciências biológicas;
- Educação e treinamento profissional;

Devido ao baixo custo do projeto e a ele ser desenvolvido de maneira livre e aberta, a tecnologia se torna mais acessível para estudantes, pesquisadores e profissionais que podem reproduzir e agregar no trabalho.

6. REFERÊNCIAS

[1] Interactive 3D Visualization of Chemical Structure Diagrams Embedded in Text to Aid Spatial Learning Process of Students

J. Chem. Educ. 2020, 97, 4, 992–1000

https://doi.org/10.1021/acs.jchemed.9b00690

Publication Date: March 4, 2020

[2] *A smart foam football that tracks your throws*

https://hackaday.io/project/170654-nerfornothing/details

[3] Navigation Kalman Filter with Accelerometer, Gyroscope and GPS, Jan Zwiener.

https://www.youtube.com/watch?v=ymuhJ6pt52o

[4] Thermal Calibration of Silicon MEMS Gyroscopes

https://cpb-us-e2.wpmucdn.com/faculty.sites.uci.edu/dist/e/700/files/2013/09/prikhodko-zotov-trusov-shkel-imaps-dpc2012.pdf

[5] MPU6050 DMP values read

https://electronics.stackexchange.com/questions/161291/mpu6050-dmp-values-read

[6] FTP server

https://en.wikipedia.org/wiki/FTP_server

[7] Biophilic design

https://en.wikipedia.org/wiki/Biophilic_design

[8] Sensor Fusion on Android Devices: A Revolution in Motion Processing

https://www.youtube.com/watch?v=C7JQ7Rpwn2k&ab_channel=GoogleTechTalks

[9] 6-Axis MEMS Motion Sensors

https://invensense.tdk.com/products/motion-tracking/6-axis/