PUC-SP

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE SÃO PAULO

Programa de Pós-Graduação em Desenvolvimento de Jogos Digitais

Ezequiel França dos Santos

GESTOS E JOGOS: REFLEXÕES E DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO DE GESTOS BASEADO EM WEARABLES PARA CONTROLE DE JOGOS

São Paulo

2022

Ezequiel França dos Santos

GESTOS E JOGOS: REFLEXÕES E DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO DE GESTOS BASEADO EM WEARABLES PARA CONTROLE DE JOGOS

Trabalho Final apresentado ao Mestrado Profissional em Desenvolvimento de Jogos Digitais, area de concentação Engenharia e Design de Jogos Digitais - Software de Jogos Digitais, como requisito parcial para qualificação ao título de Mestre Profissional em Desenvolvimento de Jogos Digitais pela Pontifícia Universidade Católica de São Paulo

Orientador: Dr. David de Oliveira Lemes

São Paulo

2022

Sistemas de Bibliotecas da Pontifícia Universidade Católica de São Paulo -
Ficha Catalográfica com dados fornecidos pelo autor

dos Santos, Ezequiel França
E99 GESTOS E JOGOS: REFLEXÕES E DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO DE GESTOS BASEADO EM WEARABLES PARA CONTROLE DE JOGOS. / Ezequiel França dos Santos. -- São Paulo: [s.n.], 2022.
103p. il. ; cm.

Orientador: David de Oliveira Lemes.
Trabalho Final (Mestrado Profissional) -- Pontifícia Universidade Católica de São Paulo, Programa de Estudos Pós-Graduados em Desenvolvimento de Jogos Digitais, 2022.

1. dispositivos não convencionais no controle de jogos. 2. controle baseado em gestos. 3. redes neurais artificiais. 4. acelerômetro. I. Lemes, David de Oliveira. II. Pontifícia Universidade Católica de São Paulo, Mestrado Profissional em Desenvolvimento de Jogos Digitais. III. Título.

CDD

TERMO DE APROVAÇÃO

Ezequiel França dos Santos

GESTOS E JOGOS: REFLEXÕES E DESENVOLVIMENTO DE UM SISTEMA DE DETECÇÃO DE GESTOS BASEADO EM WEARABLES PARA CONTROLE DE JOGOS

Trabalho Final apresentado ao Mestrado Profissional em Desenvolvimento de Jogos Digitais, area de concentação Engenharia e Design de Jogos Digitais - Software de Jogos Digitais, como requisito parcial para qualificação ao título de Mestre Profissional em Desenvolvimento de Jogos Digitais pela Pontifícia Universidade Católica de São Paulo.

Prof. Dr. David de Oliveira Lemes Orientador – PUC São Paulo

Prof. Dr. Reinaldo A. de O. Ramos Membro interno – PUC São Paulo

---

Prof. Dr. Daniel Paz de Araujo – PUC Campinas

São Paulo, 26 de Janeiro de 2022

Este trabalho aconteceu durante a pandemia de COVID-19 em 2020/21. Dedico a todos os que afetados por ela de alguma forma.

AGRADECIMENTOS

Algumas vezes já pensei que textos de agradecimento eram apenas *rasgação de seda*, porém vale lembrar o provérbio africano, ... *Se quer ir rápido, vá sozinho. Se quer ir longe, vá em grupo.* E a dissertação é o encerramento de um trabalho em grupo, por mais que esteja apenas no nome do autor.

O primeiro grupo são aqueles que estudam com você, e tive a honra que estudar com excelentes profissionais com quem tive grandes conversas mais técnicas.

O segundo grupo são aqueles que também estudam com você, mas do outro lado da mesa, e aqui começo pelo Prof. David de Oliveira Lemes, professor e orientador, venceu a COVID-19 (provavelmente um dos games mais difíceis da vida), o coordenador e todos professores ao quais as disciplinas ajudaram a formar a ideia deste projeto.

O terceiro grupo são aqueles fora da sala de aula, mas que sem eles as coisas não andariam, o Assistente de Coordenação de Jogos sempre solícito. Além disso, o professor Daniel Couto Gatti por todo suporte quando necessário, além da Assistente de Coordenação do TIDD pelo suporte quando fui fazer disciplinas lá.

O quarto grupo são aqueles que te ajudaram antes, durante e depois das aulas. Agradeço ao meu pai, Joaquim França e minha mãe Maria Denise pelo incentivo a sempre estudar e acreditar em mim mesmo. Assim como também meus sogros, Dina e Luis, que me ajudaram em momentos onde pensei que não daria certo. E *last but not least*, minha querida Van. Eu teria que fazer uma página só para enumerar o quanto você me ajudou. Seja compartilhando sua experiência com a PUC, mostrando o que eu conseguia fazer, todo apoio moral além da paciência nos momentos difíceis.

O quinto grupo são aqueles que fazem tudo ser possível, desenvolvedores que ajudaram com dicas, desenvolvedores de projetos *open-source*, os pesquisadores em pude me fundamentar.

Sapientia et Avgebitur Scientia

"People who know how to make games need to start focusing on the task of making real life better for as many people as possible."

Jane McGonigal

RESUMO

Esta pesquisa pretende geral desenvolver um estudo sobre o uso de *wearables* em jogos, em particular detectar gestos e utilizar como dispositivos de entrada em jogos, contemplando os fundamentos envolvidos, algoritmos utilizados além da análise de game design, ergonomia e aspectos sociais na abordagem. Apresentamos a modelagem do algoritmo de detecção dos gestos utilizando redes neurais, além de relatar o processo de desenvolvimento e experimentação de hardware e software, sua arquitetura e integração. A justificava dessa investigação é a possibilidade de exploração de uma interface não-convencional para o controle de jogos e escassez de debate sobre especificamente *wearables* sob a mesma ótica. Além disso, com a ascensão mercado de *wearables* e da indústria de desenvolvimento de jogos digitais, existe espaço para se explorar o tema. Uma das motivações para a pesquisa teve início no desenvolvimento de um projeto *open-source*, uma biblioteca para aquisição do movimento de *shake* (chacoalhar) do Apple Watch, visto que nas bibliotecas padrões do sistema operacional *watchOS* esta categoria de recurso não existe. Com este projeto verificou-se que as interfaces digitais de *wearables* não estão limitadas somente as micro-interações, mas também a toda possibilidade de aquisição de dados e detecção de padrões. Deste modo, a escolha do desenvolvimento de um sistema de controle baseado em *wearables* para este projeto tem o intuito também de fomentar a aplicabilidade dessas tecnologias em jogos digitais.

Palavras-chave: wearables, dispositivos não convencionais no controle de jogos, redes neurais artificiais, controle baseado em gestos, acelerômetro.

ABSTRACT

This research has as general objective to develop a study on the use of wearables in games, particularly the detection of gestures to use as input devices in games, contemplating the fundamentals involved, algorithms used, and game design analysis, ergonomics, and social aspects in the approach. We present the modelling of the gesture detection algorithm using neural networks and reporting the process of development and experimentation of hardware and software, their architecture, and integration. The justification for this investigation is the possibility of exploiting an unconventional interface for the control of games and the scarcity of debate about specific wearables from the same perspective. In addition, with the rise in wearables and the digital game development industry, there is space to explore the theme. One motivation for the research started with developing an open-source project, a library for the acquisition of the shaking movement of the Apple Watch since within the standard libraries of the operating system watchOS, this resource does not exist. Furthermore, we verified that the wearables interfaces are not limited to micro-interactions, opening others possibilities like data acquisition and pattern detection with this project. Thus, the choice of developing a wearable control system for this project also aims to promote the applicability of these technologies in digital games.

Keywords: wearables, non-conventional devices in game control, artificial neural networks, gestures based controls, accelerometer

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE GRAFICOS

LISTA DE ABREVIATURAS E SIGLAS

ACM – *Association for Computing Machinery*

PUCSP – Pontifícia Universidade Católica de São Paulo

SBC – Sociedade Brasileira de Computação

NES – *Nintendo Entertainment System*

GPS – *Global Positioning System*

HCI – *Human-Computer Interaction*

IBM – International Business Machines Corporation

INPI – Instituto Nacional da Propriedade Industrial

LSTM – *Long short-term memory*

ML – *Machine Learning*

MVC – *Model-View-Controller*

RNAs – Redes Neurais Artificiais

IEEE – *Institute of Electrical and Electronics Engineers*

SUMÁRIO

INTRODUÇÃO

As interfaces computacionais estão em constante evolução e mudanças, com o lançamento, no mercado de tecnologia, de novos modos de interação, como pelos *wearables*, conhecidos como dispositivos vestíveis.

Para Poslad (POSLAD, 2011) *wearables* são como computadores embutidos em qualquer coisa que as pessoas geralmente usam para cobrir ou acessórios para seu corpo, para (JEONG et al., 2017) os *wearables* podem ser definidos como produtos eletrônicos projetados para fornecer serviços que podem ser usados pelos consumidores.

Ashbrook (ASHBROOK, 2010) definiu micro-interações como interrupções de curta duração de tarefas primárias, podendo ter grandes benefícios ao permitir o controle de outros aplicativos móveis em paralelo às tarefas primárias em andamento, além de expandir significativamente o conjunto de tarefas que poderíamos realizar em trânsito. As micro-interações no contexto dos *wearables* se enquadram na definição de Ashbrook, sendo estas micro-interações rápidas ações direcionadas a tarefas e orientadas a objetivos e que geralmente incluem um *feedback* do sistema.

Um exemplo seria o recebimento de uma notificação de um aplicativo móvel. As ações principais, ou interações principais, são feitas neste aplicativo, porém podemos ter micro-interações em um aplicativo para um *smartwatch*, como respostas padrões prontas.

Com isto, como estes dispositivos *wearables*, dispositivos vestíveis, que já utilizados no nosso dia-a-dia podem ser utilizados com interfaces para jogos através de gestos? É possível desenvolver um jogo que seja compatível com mais de uma plataforma *wearable* e quais seriam suas aplicações? Qual o seu impacto nos processos de criatividade e game design? Como é feita a detecção de gestos, utilizando aprendizado de máquina?

Com este projeto verificou-se que as interfaces digitais de *wearables* não estão limitadas somente as micro-interações, mas também a toda possibilidade de aquisição de dados e detecção de padrões.

Deste modo, a escolha do desenvolvimento de um sistema de controle baseado em *wearables* para este projeto tem o intuito também de fomentar a aplicabilidade dessas tecnologias em jogos digitais.

Revisão Bibliográfica

Abordando o uso de acelerômetros, ao qual veremos em mais detalhes e são dispositivos usados para medir a aceleração própria de um sistema, na detecção de gestos, temos diversos autores citados, dois deles com maior relevância. A professora Dra. Xiang Chen e o Professor Dr. Xu Zhang, ambos professores da Universidade de Ciência e Tecnologia da China. Todos os seus artigos acerca de detecção de gestos através de acelerômetros contribuíram para construção teórica deste trabalho, desde o *'Hand Gesture Recognition Research Based on Surface EMG Sensors and 2D-accelerometers'* de 2007 até o *'Comparative Study of Gesture Recognition Based on Accelerometer and Photoplethysmography Sensor for Gesture Interactions in Wearable Devices'* de 2021.

Outra grande referência neste trabalho é a pesquisadora brasileira Lucia Santaella. Ela é considerada uma das pioneiras na área da semiótica e da metodologia da ciência, e suas vertentes vão de encontro com fundamentos bio-cognitivos da comunicação, computacional e estéticas tecnológicas, novas tecnologias, jogos eletrônicos, entre outras (LUCIA. . ., 2021), sendo uma autora essencial para diversas definições desta pesquisa, especialmente em tópicos de análise semiótica e filosófica da tecnologia. Muitas de suas obras levantam questões a respeito dos dispositivos, máquinas, tecnologia e sociedade, as diversas formas de socialização na cultura digital, suas complexidades semióticas, os novos ambientes comunicacionais, as comunidades virtuais, linguagem e constituição do sujeito cultural e as formações psicossociais na era digital.

Ainda como referências teóricas, o banco de dissertações da Pontifícia Universidade Católica de São Paulo (PUCSP), os repositórios e periódicos da Sociedade Brasileira de Computação (SBC), do *Institute of Electrical and Electronics Engineers* (IEEE), da *Association for Computing Machinery* (ACM) além do repositório da *International Journal of Computer Games Technology* foram utilizados.

E por fim, as documentações oficiais de todas as tecnologias utilizadas também foram exploradas, não apenas como referência técnica na construção do projeto, mas também seus referenciais e nuances teóricas.

Objetivos Gerais

A pesquisa pretende geral desenvolver um estudo sobre o uso de *wearables* em jogos, em particular detectar gestos e utilizar como dispositivos de entrada em jogos, contemplando os fundamentos envolvidos, algoritmos utilizados além da análise de game design, ergonomia e aspectos sociais na abordagem. Apresentamos a modelagem do algoritmo de detecção dos gestos utilizando redes neurais, além de relatar o processo de

desenvolvimento e experimentação de hardware e software, sua arquitetura e integração.

Organização da dissertação

No primeiro capítulo, introduzimos uma breve reflexão sobre gestos, *wearables* e sociedade e a aplicabilidade e usabilidade de um sistema baseado em gestos.

No segundo capítulo, temos aspectos mais técnicos, iniciando com um histórico dos *wearables* de pulso, avançando para os conceitos e fundamentos dos sensores, sensores no controle de jogos, princípios de eletrônica e um pouco do mercado de jogos controlados por gestos.

No terceiro capítulo apresentamos o resultado obtido no sistema de detecção de gesto desenvolvido, ele em funcionamento e discutimos suas características, os algoritmos de aquisição dos dados e tratamento dos dados, a devida modelagem dos dados e a construção da rede neural para detecção dos gestos, assim como sua integração.

No quarto e último capítulo apresentamos as conclusões e possibilidades de trabalhos futuros.

Motivações

Como dito anteriormente a ascensão mercado de *wearables*, sua evolução em *hardware* e o crescimento da indústria de desenvolvimento de jogos digitais.

Além das perspectivas de mercado do tema, especialmente em um mestrado profissional, outra justificava dessa investigação é a possibilidade de exploração de uma interface não-convencional para o controle de jogos e escassez de debate sobre especificamente *wearables* sob a mesma ótica.

Outra das motivações para a pesquisa teve início no desenvolvimento de um projeto *open-source*, uma biblioteca para aquisição do movimento de *shake* (chacoalhar) do Apple Watch (F. SANTOS, 2021), visto que nas bibliotecas padrões do sistema operacional *watchOS* esta categoria de recurso não existe.

Metodologia

Neste trabalho foi utilizada a metodologia *Design Science Research* (DSR) (VAISHNAVI; KUECHLER, 2015). Esta metodologia foi adotada, pois, auxilia a construção de um artefato e melhorá-lo por um processo contínuo de refinamento e avaliação. Neste

[0]https://github.com/ezefranca/WatchShaker

trabalho foi utilizado o ciclo de DSR baseado em Vaishnavi et al. (VAISHNAVI; KUECHLER, 2015), composto pelas seguintes fases: compreensão do problema, sugestão, desenvolvimento, avaliação e conclusão.

1. **Compreensão do problema:** Nesta fase ocorre a busca de informações sobre o problema a ser investigado, sem solucioná-lo todavia. Busca-se o entendimento e a descrição do problema, identificando os principais conceitos e afetados pelo problema, além dos objetivos, causa do problema, efeitos e contribuições quando atingidos os objetivos. Portanto, nesta etapa foi conduzida uma exploração da literatura, uma exploração semiótica, a observação e levantamento do mercado de jogos baseado em gestos;

2. **Sugestão:** A sugestão é primordialmente uma etapa criativa onde novas configurações são concebidas e assentadas em uma nova configuração de novos elementos ou de elementos previamente existentes. Para esta fase, foi concebido um artefato. O artefato proposto foi um protótipo baseado em sensores e um sistema microcontrolado – em forma de prova de conceito e código-fonte – para apoiar a validação da ideia de jogo baseado em gestos através de *wearables*. Esta fase tem como saída um projeto-piloto.

3. **Desenvolvimento:** O desenvolvimento do protótipo de um jogo controlado através de um *wearable*. Inclui todas as etapas do desenvolvimento, da aquisição dos dados do acelerômetro, o treinamento do modelo para reconhecimento de gestos, sua integração e a integração deste sistema com a *game engine*.

4. **Avaliação:** Por estarmos vivenciando uma pandemia, a avaliação de *gameplay* foi postergada, todavia, utilizando referências no estado da arte a qualidade do modelo treinado foi avaliada em comparação a outros modelos.

5. **Conclusão:** Por último, apresentamos as considerações finais, apontando possíveis trabalhos futuros.

1 SOCIEDADE, TECNOLOGIA, GESTOS E JOGOS

1.1 Sociedade, Tecnologia e Gestos

A tecnologia e a sociedade são elementos inseparáveis, portanto não compreendemos a tecnologia sem entender a sociedade e cultura do tempo em questão, assim como para compreender a sociedade precisa-se compreender suas tecnologias. Já em 1861, Charles Darwin escreveu *"Não posso duvidar que a linguagem deva sua origem à imitação e modificação, com apoio de sinais e gestos, de vários sons naturais, e gritos instintivos do próprio homem"* (DARWIN; BONNER; MAY, 1981).

Marcel Jousse (1886 – 1961), foi um antropólogo, ele trabalhou na intersecção do empírico com o fenomenológico na compreensão do ser humano introduzindo uma concepção do gesto, em obras como Antropologia do gesto (JOUSSE, 1969), que não pode ser totalmente explicada pelos fatores biológicos, nem pelos eventos fenomenológicos. Para ele, o gesto começa nas possibilidades que se apresentam desde o princípio da existência até o sentido que dado na consciência reflexiva (linguagem) que passa pelo estágio de formação pré-reflexiva e pré-linguística. Para ele, o gesto antecede e está na origem da linguagem, partindo de uma constatação empírica: organismos vivos são transformam as formas de energia e formam diferentes novas formas de absorver e gastar energia. O corpo humano possui uma qualidade própria de comutar o dado em sentido e transformar movimentos em gestos ou o corpo encarnado em formas de expressão: o corpo humano é um gerador de gestos sendo o ele mesmo visto no macro como um gesto no mundo de sua intersubjetividade. Desde as ações mais simples do ser humano em relação à formação de si o corpo é um gerador de variações possíveis (como no imaginário) similar a um jogo de gestos e ritmos. (JOSGRILBERG, 2016)

Já em um sentido linguístico, para (ROMERO, 2009) o gesto nasce de uma necessidade interior de expressão, de comunicação — é a primeira e a mais rica linguagem do corpo. Segundo (ROSÁRIO SILVA LIMA; CRUZ-SANTOS, 2012) temos os gestos naturais como a primeira ferramenta de comunicação simbólica. As funções foram sendo transformadas ao longo dos períodos de evolução da comunicação, mas que suporta a nossa eficácia comunicativa ao longo da vida, e que embora não sejam convencionados ou símbolos arbitrários como as palavras são, eles ocorrem repetidamente na mesma forma física.

Mark L. Knapp, (HALL; KNAPP, 1999) classifica de três formas distintas não verbais de codificação, sendo:

- Linguagem dos sinais: inclui todas as formas de codificação em que palavras foram substituídas por gestos, variando desde mais simplificado ao mais complexo (Libras);
- Linguagem das ações: abrange os movimentos que não são usados como signos, por exemplo: a maneira de caminhar pode possuir propósito próprio e, em simultâneo, declarar algo para quem consegue interpretar;
- Linguagem dos objetos: compreende a exibição intencional e não intencional de coisas materiais como objetos de arte, máquinas e também do corpo humano.

Com isso, é muito improvável que se entre em um debate sobre jogos digitais e reconhecimento de gestos e não debater o corpo, a ludicidade e o movimento. Neste sentido há muitas conceituações para a ludicidade, algumas contradições e até mesmo confusões. Neste trabalho nos restringimos a trabalhar a ludicidade sob a ótica de Vygotsky, especialmente por estarmos em um ambiente relacionado a jogos e criatividade.

Os estudos de (VYGOTSKY, 1994) a respeito do fenômeno criativo e suas características trouxeram diversas contribuições para esse campo do conhecimento, especialmente se considerarmos a maneira clara e objetiva como conceituou criatividade. Visto estar lidando com a detecção de gestos, utilizando sensores eletrônicos, a sugestão de Vygotsky de elaborar uma analogia entre os fenômenos da criatividade e eletricidade, se faz muito familiar.

A eletricidade está presente em eventos de diferentes magnitudes. Em enorme quantidade nas tempestades, com os raios e trovões, mas ocorre também na pequena lâmpada, quando se liga o interruptor. A eletricidade é a mesma, o fenômeno é o mesmo, contudo expresso com intensidades diferentes.

Para Vygotsky a criatividade se processa da mesma forma. Todos somos portadores dessa energia criativa. Alguns vãos apresentarem essa energia em maior quantidade, outros menos, mas ela é a mesma, a capacidade de imaginar também, apenas distribuídas diferenciadamente.

Quando se pensa na ideia do imaginar, a criatividade de brincar, a criatividade de jogar, o que para muitos pode ser apenas um gesto comum, como o ato de desenhar um círculo no ar, o gesto de levar o braço de um lado para outro, ou o simples chacoalhar de uma mão, podem ter um significado lúdico muito mais profundo. Podem ser o lançamento de um feitiço, ou bloqueio de um ataque, a utilização de um item ou qualquer outra possibilidade associada a jogabilidade e criatividade da situação.

Segundo (VYGOTSKY, 1994) os gestos também ligam-se à escrita por meio dos jogos das crianças. Os gestos das crianças dão significado aos objetos. As crianças brincam com quaisquer objetos, aos quais são dados o nome de símbolos ou signos, e dar-lhes um significado para o que querem representar. Com base nestas observações, podemos considerar que o reconhecimento de gestos, com fins de uso em um jogo, apresentam uma ligação com os pensamentos Vygotsky e a criação de signos e símbolos.

Assim, podemos observar as diversas evoluções do mercado de jogos digitais, alinhadas às mudanças comportamentais da sociedade e a ressignificação de objetos como símbolos e sinais.

Jogos são sistemas computacionais multidisciplinares, para (SANTAELLA, 2004), os games, fazem parte de um campo híbrido, pois envolvem diversas áreas do conhecimento, como programação, navegabilidade, design de interface, usabilidade, entre outras, se tratando de uma mídia interdisciplinar.

A interface de gestos em jogos, é uma abordagem de interface homem-máquina que visa aumentar a sensação de realidade para um indivíduo através do uso de sensores e softwares onde se possa interpretar movimentos e sensações. As interações baseadas em gestos devem ocorrer em tempo real, assim como as respostas das ações efetuadas pelo usuário. Para isto, estas ações devem ser capturadas e tratadas em tempo real. (KORTUM, 2008).

Essas interfaces também podem traduzir os eventos captados no mundo real e adaptá-los a universos não propriamente reais, seja por ícones, símbolos ou regras diferentes das daquele contexto, criando um mundo ficcional, onde se podem experimentar sensações não percebidas no mundo real, como, por exemplo, voar (KORTUM, 2008).

1.2 Wearables e Interfaces Computacionais

As interfaces computacionais estão em constante evoluc a o e mudanças, com o lançamento, no mercado de tecnologia, de novos modos de interação, como por os *wearables*, conhecidos como dispositivos vestíveis.

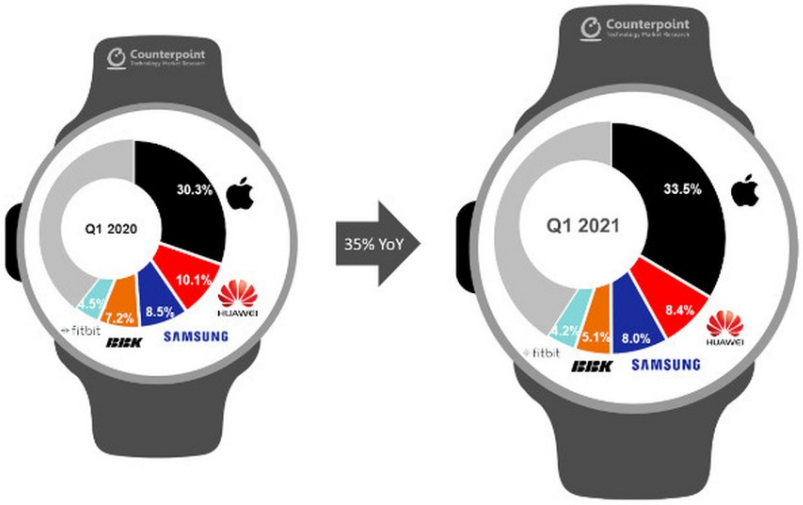
A relação entre os corpos e os objetos foi amplamente apresentada na literatura do século XIX. Do Romantismo ao Decadentismo, apresentou-se uma semântica dos objetos a partir da qual as trocas simbólicas mediavam os significados orientando novas práticas sociais. Neste ponto, revelaram-se como os corpos e os objetos exprimiam o ethos da cultura material em nossa existência (DORNAS; ADVERSE; GOUVEIA, 2020).

Poslad (POSLAD, 2011) definiu a *wearables* como computadores embutidos em qualquer coisa que as pessoas geralmente usam para cobrir ou acessórios para seu corpo.

De acordo com (JEONG et al., 2017), *wearables* podem ser definidos como produtos eletrônicos projetados para fornecer serviços que podem ser usados pelos consumidores.

Segundo o relatório da *Counterpoint Research* (GLOBAL..., 2021) as vendas de *smartwatches* cresceram 35% no primeiro trimestre de 2021, e a Apple lidera o segmento, com aumento de 3% em relação ao primeiro trimestre de 2020 na participação de mercado, conforme infográfico da Fig. 1.

Figura 1: Gráfico comparativo da Counterpoint Research, 2020-2021



Quando se fala de interação entre humano/não-humano, se encontram as interfaces viabilizadas por paisagens híbridas, onde "espaços e ambientes biológicos misturam-se com imagens, espaços e ambientes sintetizados" (SANTAELLA, 1997), processo atualmente difundido pela designação de ciberespaço. Com os jogos como sistemas computacionais e os *wearables* como sistemas ciber-físicos de controle baseados em gestos, a ergonomia destas ações de movimentos e gestos devem ser observadas também na análise desta interação. Para (BRAGA; LEMES, 2005) um processo de interação entre usuário e computador a ergonomia pode se fundamentar no sistema homem tarefa máquina, onde o objetivo único do sistema passa a ser a tarefa designada ao homem e não à máquina, obrigando o sistema a obedecer um processo que dá primazia ao homem, para que ele realize a tarefa da qual foi incumbido.

Quando considera-se os dispositivos *wearables* como máquinas de controle ou interfaces de controle, por alguma categoria de sensor acoplado, podemos o *wearable* como máquinas sensórias, definidas por (SANTAELLA, 1997) como máquinas que funcionam como extensões dos sentidos humanos especializados, extensões do olho e do ouvido de que a câmera fotográfica foi inaugural. O funcionamento de tais máquinas está ligado de maneira tão visceral à especialização dos sentidos ou aparelhamentos da visão e da escuta humanas que a denominação de aparelhos lhes cabe muito mais ajustadamente do que a

de máquinas.

Atualmente os wearables estão mais acessíveis e com maior capacidade computacional, criando uma excelente oportunidade de exploração destes dispositivos, como neste trabalho, no controle de jogos. Dada as modificações necessárias, a transformação de um *wearable* comercial em um dispositivo de controle, ou seja, alterar sua função original, devemos trabalhar em campo multidisciplinar.

1.3 Gestos e Jogos

Segundo (HEIMONEN et al., 2013), em um contexto industrial, a interação baseada em gestos tem o potencial de libertar os usuários dos tediosos controles físicos, mas também deve considerar as considerações de segurança e as percepções dos usuários.

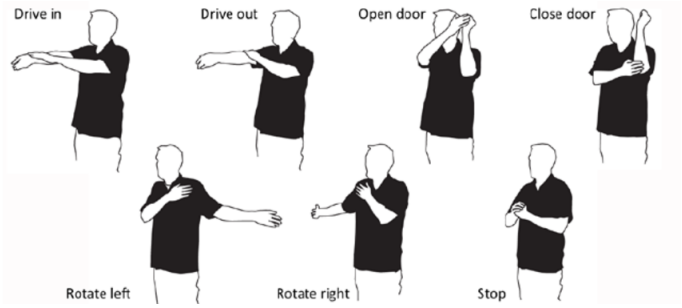


Figura 2: Sistema para controle industrial (HEIMONEN et al., 2013)

Gestos como interfaces de interação não são novidades na área de desenvolvimento de jogos digitais. Como veremos nas próximas seções, existem diversos estudos que abordam e utilizam as interfaces com base em gestos.

Ao falar em interfaces, lida-se com alguns conceitos que assumem um caráter abstrato para os desenvolvedores, um deles o de interface intuitiva.(TURNER, 2008) aborda a ideia de uma interface intuitiva, na direção do que se trata de significados diferentes quando utilizados nos estudos de interface homem computador *Human-Computer Interaction* (HCI) e nas guias do design de interface ou no discurso do marketing dos produtos tecnológicos.

Turner afirma que este conceito está relacionado à ação e percepção, vistas conjuntamente, assim como (GIBSON, 1977) ao trabalhar o conceito de *affordance*, que diz respeito ao fato de que uma coisa permite observar o que podemos fazer com ela. O chão, por exemplo, permite a ação de caminhar, mas não de mergulhar, uma pedra, dependendo do tamanho, pode permitir a ação de escalar ou de sentar. Percebemos algo quando nos movimentamos, mas também percebemos o categoria de movimentos que podemos fazer em função das *affordances* que identificamos. Em jogos baseados em gestos podemos identificar

o *affordance* no controle e interfaces dos jogos, sendo de maior interesse neste trabalho a contextualização feita nos jogos quando da utilização de um gesto ou movimento.

1.4 Jogos baseados em gestos

Os jogos baseados em gestos, são jogos que utilizam interfaces gestuais e já existem há alguns anos. O Nintendo Wii é o exemplo mais familiar dos consoles modernos, mas essas interfaces podem ser encontradas há algumas décadas. Na impossibilidade de enumerar e elencar todos os jogos existentes dados os fabricantes e particularidades de cada país, elenca-se a seguir uma linha do tempo de jogos que tinham os gestos como elemento (seja primário ou secundário) de interface com o usuário.

1.4.1 Fliperamas 1980-1990

Pole Position (1982)

Pole Position foi um dos primeiros games de corrida dos arcades, lançado pela Atari, foi produzido e licenciado da Namco no ocidente, a Atari desenvolveu uma cabine grande, apresentada na Fig.3, muito popular no Brasil. Para popularizar o jogo, a Namco encomendou também um desenho animado, que não tinha nada relação com o jogo, mas ficou bastante conhecido no Brasil.

Figura 3: Máquina de Pole Position

Star Wars (1983)

Em 1983, a saga *Star Wars* já era uma febre mundial e o filme recebia uma grande variedade de brinquedos incluindo videogames. Dentre eles, havia a máquina arcade de *Star Wars*. Com dois modelos diferentes, a que se popularização mais parecia um cockpit de um X-Wing, apresentada na Fig.4. Era estilizada com personagens do filme como o Darth Vader.

Figura 4: Arcade de Star Wars

Hang-On (1985)

Hang-On foi outro jogo de corrida ganhou popularidade graças a sua cabine, vista na Fig.5. Em formato de uma moto, a cabine do jogo era compacta e toda a eletrônica ficava oculta dentro da carcaça da moto. A tela ficava posicionada no painel da moto e os controles da cabine eram um dos destaques, pois era possível, por exemplo, fazer curvas inclinando o corpo de um lado para outro, igual aos pilotos de moto.

Figura 5: Arcade moto do Hang-On

Out Run (1986)

Outro famoso jogo de corrida foi o Out Run. Lançado pela SEGA em 1986, a cabine padrão do jogo se assemelhava a um simulador de corrida Fig.6. O jogo, no entanto, era um *arcade racing*. Outro destaque da máquina era o som de alta qualidade que vinha dos dois alto-falantes, um dos principais elementos do jogo era a sua trilha sonora.

Figura 6: Arcade Cabine de Out Run

Arm Champs (1988)

Arm Champs era um jogo diferente para a época, no estilo *Punch Out*, o jogo possuía uma sequência de combates que eram decididos por quem tem a melhor técnica. O jogador disputava "braço de ferro"com um literalmente um braço de ferro motor revestido por plástico. Uma interface de controle nada convencional para época. A Fig.7 apresenta a segunda versão da máquina lançada em 1992 na segunda versão do jogo.

Figura 7: Arcade Arm Champs II (1992)

1.4.2 Joyboard - Atari (1982)

O Joyboard foi uma placa de equilíbrio para o console de videogame Atari 2600, lançada em 1982. Nela, o jogador ficava sob a placa e inclinado-se em uma determinada direção. Funcionalmente, o Joyboard levou os quatro botões direcionais de um joystick para a parte inferior da placa, possibilitando o controle do jogo. Possui grande semelhança em design e função com a Wii Balance Board, vista na subseção 1.4.7.

Figura 8: Joyboard Atari (peça publicitária da época)

1.4.3 PowerPad - Bandai (1986)

O Power Pad foi tapete para controle de jogos, Fig.9, para o console *Nintendo Entertainment System* (NES). Possuía doze sensores de pressão embutidos entre duas camadas de plástico flexível. A Bandai lançou o acessório pela primeira vez em 1986 como o pacote Family Trainer para a NES no Japão, posteriormente a Nintendo o lançou em 1988 como Power Pad, com o jogo *World Class Track Meet*, que foi uma reformulação do jogo anterior.

Figura 9: Tapete controlador PowePad

1.4.4 Power Glove - Mattel (1989)

A Power Glove foi um acessório no formato de luva virtual para o NES lançado em 1989, contudo não obteve o sucesso esperado devido a imprecisão dos sensores e a dificuldade para usar os controles, tendo apenas 2 jogos feitos especificamente para o acessório, Super Glove Ball e Bad Street Brawler. Desenvolvida a partir da VPL Dataglove[1] com modificações para reduzir o custo, a original utilizava sensores de fibra ótica e conseguia detectar os 3 eixos de rotações com 256 posições diferentes, já a Power Glove utilizava um sensor de tinta condutiva e detectava apenas um eixo de rotação e tinha apenas 4 posições.

Figura 10: Power Glove da Mattel

1.4.5 Kirby Tilt 'n' Tumble (2001)

O jogo *Kirby Tilt 'n' Tumble* foi desenvolvido pela HAL Laboratory e publicado pela Nintendo para o console portátil Game Boy Color. Foi lançado no Japão em 23 de agosto de 2000 e na América do Norte em 11 de abril de 2001. O cartucho do jogo possui acelerômetro embutido, usados para controlar o Kirby inclinando o Game Boy na direção em que o jogador deseja movê-lo. além da ação de "pop"quando o jogador rapidamente chacoalha o Game Boy.

Figura 11: Cartucho do *Kirby Tilt 'n' Tumble*

[1]VPL Dataglove foi uma luva tátil criada pela VPL, empresa pioneira em realidade virtual

1.4.6 EyeToy - Sony (2003)

O EyeToy foi uma câmera digital colorida, como uma *webcam* comum, para o PlayStation 2. A tecnologia usa a visão computacional para processar imagens permitindo a interação com jogos através do movimento do corpo, além de detecção de cores e o som através do microfone incorporado. A câmera era fabricada pela Logitech, e posteriormente pela empresa Nam Tai.

Figura 12: EyeToy - a câmera para o Playstation 2

1.4.7 Wii Balance Board (2008)

A Wii Balance Board, acessório usado nos consoles Wii e Wii U, foi exibida pela primeira vez em 11 de julho de 2007 na *Eletronic Entertainment Expo* e lançada em maio de 2008. Ao contrário de uma prancha de equilíbrio comum, a balança rastreia o centro de equilíbrio do próprio jogador sendo utilizada em jogos de exercícios. Acompanhava os jogos Wii Fit, Wii Fit Plus e Wii Fit U. O acessório utiliza a tecnologia Bluetooth e contém quatro sensores de pressão usados para medir o centro de equilíbrio do jogador, suportando pessoas com até 150 kg de massa corporal.

Figura 13: Wii Balance Board

1.4.8 Projeto Natal - Microsoft (2010)

O nome “Projeto Natal” foi em homenagem à cidade brasileira de Natal, porque o desenvolvedor da tecnologia Alex Kipman é brasileiro e decidiu fazer esta homenagem a sua cidade no nome inicial do projeto.

Foi anunciado na *Eletronic Entertainment Expo* 2010, lançado posteriormente como *Kinect*. A tecnologia no primeiro Kinect conseguia detectar de movimentos, identificando gestos sutis como mover os dedos, girar o pulso ou mesmo suas expressões faciais e até identificar seu batimento cardíaco ou até identificar a força empregada em um movimento, como um soco.

Em outubro de 2017 a Microsoft encerrou a produção do Kinect. O baixo número nas vendas e quantidade de conteúdos para o dispositivo são as razões apontadas para o desfecho.

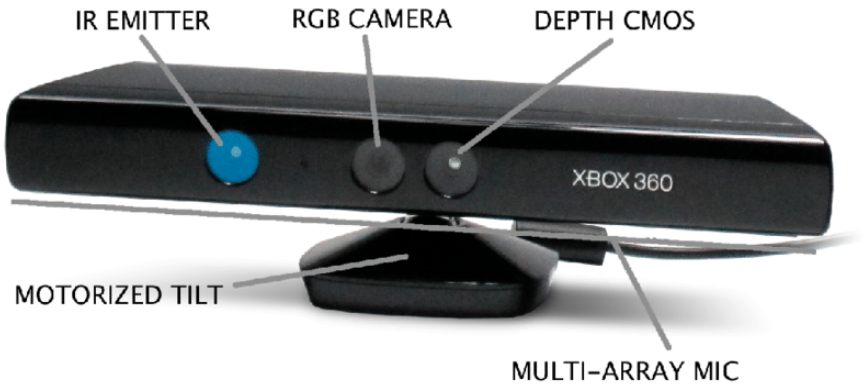


Figura 14: Kinect e seus Sensores

Apesar desta seção mostrar alguns jogos e projetos com interfaces baseadas em gestos que já não existem, na seção 1.6 observaremos os dispositivos de gerações mais recentes, com maior destaque em acelerômetros, foco deste trabalho.

1.5 Nintendo e o controle de gestos baseado em acelerômetros

É notável a contribuição da Nintendo no controle de jogos baseado em gestos. (KAO et al., 2020) menciona que além da criação de Mario por Miyamoto[1], a Nintendo também contribuiu com jogos portáteis (Nintendo DS 2004), e jogos baseados em movimento (Nintendo Wii de 2006), além dos jogos híbridos (Nintendo Switch 2017).

Apesar destes consoles não estarem necessariamente usando as tecnologias de ponta na época; o Nintendo Wii usou acelerômetro para incorporar criativamente a detecção de movimento para jogos, compensando as especificações de *hardware* ficaram atrás de seus concorrentes.

De fato, os estudos de (SHIRATORI; HODGINS, 2008) mostram o potencial do *Wiimote* para o desenvolvimento de jogos mais imersivos através de gestos. A figura 15 apresenta os eixos dos sensores do controle do Nintendo Switch.

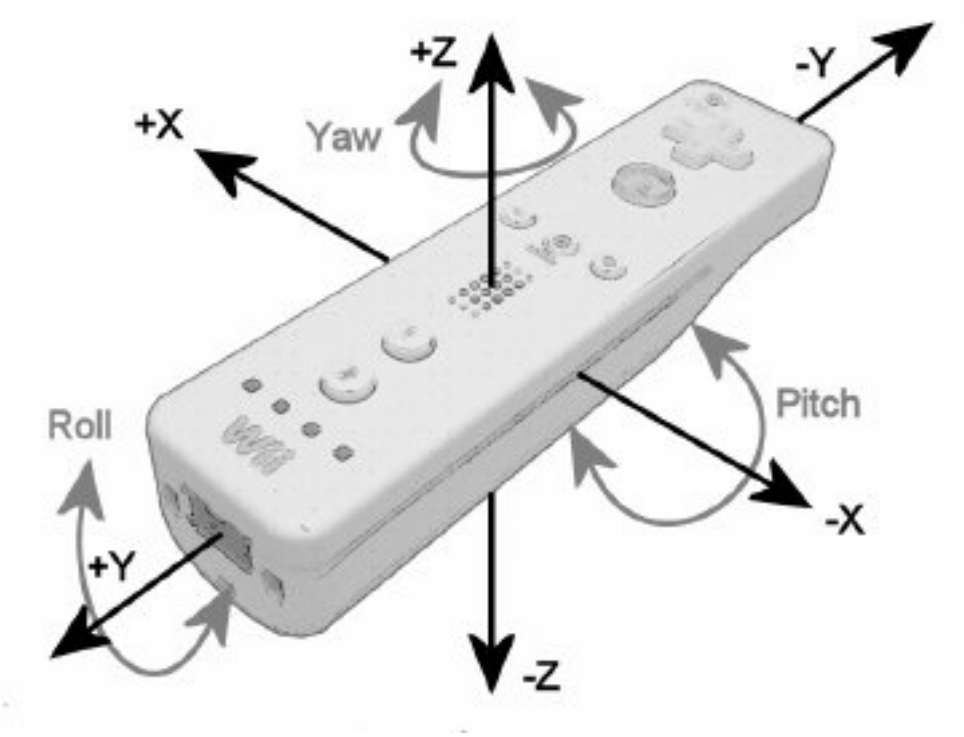


Figura 15: Eixos do acelerômetro e giroscópio (SHIRATORI; HODGINS, 2008)

No caso do Wiimote também temos botões, temos force feedback, e temos um direcionador baseado em joystick, mas o que chama atenção é o direcionamento através de movimentos, gestos. (PFUTZENREUTER, 2009)

O Wiimote tem uma forma neutra, que permite-lhe representar outros objetos, em

[1] Shigeru Miyamoto, Sonobe, 16 de novembro de 1952) é um designer e produtor de jogos eletrônicos japonês, conhecido por ser o criador de algumas das mais bem-sucedidas séries de jogos eletrônicos de todos os tempos. Miyamoto entrou na Nintendo em 1977, quando a companhia entrava no mercado de jogos. Seus primeiros trabalhos foram em arcades no final da década, e desde então, seus jogos estão em todos os consoles da Nintendo. https://pt.wikipedia.org/wiki/Shigeru_Miyamoto

função do gesto efetuado pelo jogador, sendo que, no que lhe concerne, esse depende do jogo que está sendo jogado. O diferencial desse controle seria, então, o aproveitamento dos gestos e a forma simples. (PFUTZENREUTER, 2009)

Os jogadores usam, por exemplo, o Wiimote para imitar ações realizadas em esportes da realidade, como balançar uma raquete de tênis.

Um dos grandes destaques foi o jogo Wii Sports, apresentado na figura 16. O jogo é uma coleção de cinco simulações esportivas, projetadas para demonstrar os recursos de detecção de movimento do Wiimote. Os cinco esportes incluídos são tênis, beisebol, boliche, golfe e boxe.

Figura 16: Exemplo de funcionalidade do Wii Remote no Wii Sports

1.6 Jogos e seus controles no contexto dos gestos

Temos como definição em (FLEURY; NAKANO; CORDEIRO, 2014) o fenômeno de se utilizar imagens, acelerômetros e giroscópios no controle de jogos como cinestesia. Segundo (FLEURY; NAKANO; CORDEIRO, 2014), captar movimentos humanos através de sensores corporais e/ou por imagens já é uma tecnologia desenvolvida. Apesar de já estarem no mercado há algum tempo, ainda apresentam grande potencial de exploração.

Segundo (NAKATSU, 2016) a maioria dos especialistas em tecnologia e engenharia deseja ansiosamente alcançar uma ambição, a de desenvolver uma tecnologia prática que possibilite as pessoas utilizarem quaisquer sistemas artificiais de comunicação e tecnologia, apenas por gestos, sem a necessidade de sensores equipados e para a aprendizagem prévia de manipulação.

Esta mesma abordagem pode ser encontrada em (CHAN; LEONG; KONG, 2009), onde um sistema de gestos para controle de jogos foi desenvolvido sem nenhum sensor acoplado ao corpo, utilizando-se de sistemas de visão computacional.

Em contraponto, vemos em (KHAN, 2011) que técnicas de reconhecimento de gestos baseadas em visão computacional possuem as desvantagens da invasão de privacidade do indivíduo e as dificuldades relacionadas ao próprio processamento de imagens.

Entretanto, o intuito deste trabalho não é contrapor técnicas de reconhecimento, e sim revisitar as possibilidades dos *wearables* no controle de jogos. Nesta abordagem de transformação, deseja-se adicionar novas inteligências e capacidades aos *wearables*.

Antes dos atuais dispositivos equipados com acelerômetros, um dos primeiros usos dos acelerômetros em jogos não foi por um *smartphone*. Apesar destes sensores parecerem uma tecnologia nova, esta já existe há certo tempo. O Amida Simputer na Figura 17, por exemplo, foi um dispositivo portátil baseado em Linux lançado em 2004, sendo o primeiro portátil comercial a ter um acelerômetro embutido. Ele incorporou muitas interações baseadas em gestos usando este acelerômetro, incluindo virada de página, zoom-in e zoom-out de imagens, mudança de retrato para modo paisagem e muitos jogos simples baseados em gestos (DEEP et al., 2010).

Figura 17: Amida Simputer, jogos: a direita caroms e a esquerda bowling

Diversos fabricantes incorporaram detecção de gestos e movimentos através de

acelerômetros em seus produtos para interação homem-máquina.

Ainda que os acelerômetros em dispositivos móveis tenham certas limitações, a pesquisa sobre o uso da tecnologia de gestos baseada em acelerômetros na interação humano-computador (HCI) um longo tempo. A mudança para uma interface baseada em gestos interfere diretamente no processo de interação homem-máquina, a percepção na entrada e saída de dados se difere de uma interface convencional. (BONGERS, 2000) propôs o chamado ciclo clássico de interação homem-máquina, apresentado na figura 18, onde o usuário fornece entrada de controle; o sistema processa a entrada e produz uma saída. Essa saída é passada de volta ao usuário na interface. O usuário percebe a saída do sistema, processa-a e fornece outras entradas. Este ciclo pode ser benéfico no projeto de um sistema baseado em gestos, visto que considera o *Sensory Stimulate* (Estimulo de um sensor) como mecanismo de entrada, ao qual podemos considerar os dados de um acelerômetro.

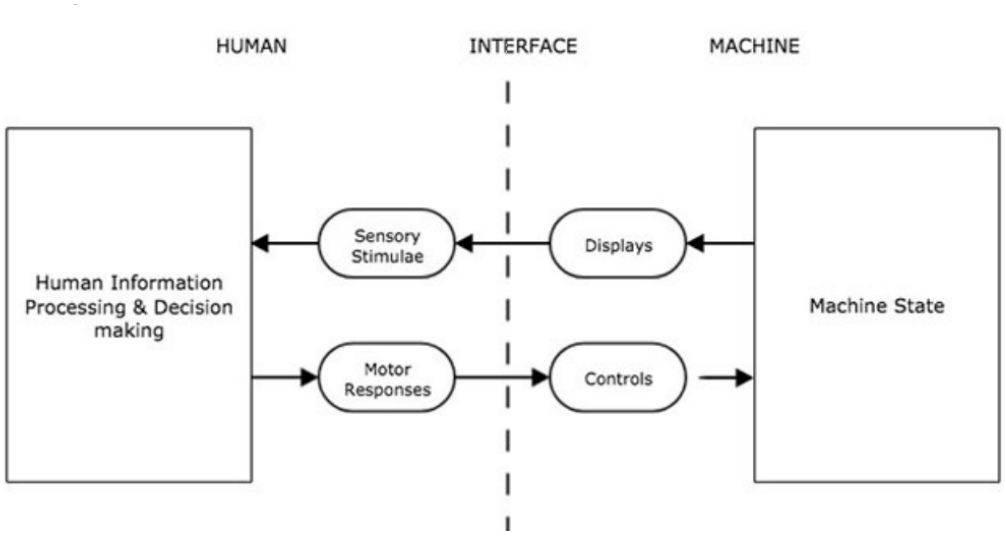


Figura 18: O ciclo de interação homem-máquina (BONGERS, 2000)

As aplicações do reconhecimento de gestos em um dispositivo móvel hoje adicionam diversas capacidades, incluem, por exemplo, a rolagem de um documento, navegação no menu, alteração da orientação da tela, zoom, panorâmica e assim por diante (MACKENZIE; TEATHER, 2012).

E de fato, mesmo com as limitações, com a evolução dos dispositivos e desenvolvimento novos de jogos, interfaces baseadas em movimentos de acelerômetros foram ganhando mercado. (MEDRYK; MACKENZIE, 2013) conduziu experimentos com jogadores utilizando interfaces de toque e interfaces de controle baseadas no acelerômetro, onde 72% dos jogadores acharam o controle por inclinação (acelerômetro) mais envolvente.

2 TECNOLOGIAS EM WEARABLES

2.1 A evolução dos *wearables* de pulso

A jornada dos vestíveis começou com a invenção dos óculos por volta do século XIII pelo frade inglês Roger Bacon, que morou em Paris e delineou os princípios científicos por trás do uso de lentes corretivas em seu Opus Majus (c.1266) (OMETOV et al., 2021).

O primeiro relógio mecânico de bolso, passível de ser transportado, remonta ao início do século XVI, e acredita-se ser o relógio Pomander (Bisamapfeluhr em alemão) (OMETOV et al., 2021). PeterHenlein o tornou em 1505 como um relógio portátil, mas não muito preciso. Mais tarde, relógios de bolso também foram desenvolvidos significativamente com a evolução da miniaturização, que levou à ideia de prender o dispositivo ao pulso no século XIX (OMETOV et al., 2021).

Já o aviador Alberto Santos-Dumont, *Pai da Aviação*, em 1907 encomendou a criação do primeiro relógio de pulso ao famoso joalheiro Louis Cartier de acordo com suas especificações. O relógio de pulso permitiu que ele mantivesse as mãos livres para a pilotagem.

Figura 19: Alberto Santos-Dumont, *Pai da Aviação*

É difícil enumerar e elencar todos os *wearables* de pulso, foco deste trabalho, existentes dado os fabricantes e particularidades de cada país, todavia, por dados dos portais (WAREABLE, 2021), (WIKIPEDIA, 2021) e (SMARTWATCHES, 2021), é apresentada uma linha do tempo dos *wearables*, que tiveram destaque comercialmente e/ou culturalmente, mencionando suas características, além de seus aspectos de envolvimento com jogos, quando ocorrem.

- 1927 - Considerado por muitos como o "primeiro *Global Positioning System* (GPS) de pulso", o *Plus Four Wristlet Route Indicator* (figura 20) era um mapa de pulso. Bastava inserir o cartucho de mapa de rolagem para sua rota definida.

Figura 20: 1927: *Plus Four Wristlet Route Indicator*

- 1930 - Chester Gould [4] criou e escreveu uma história em quadrinhos sobre um detetive policial engenhoso chamado Dick Tracy. Tracy fazia uso frequente de um dispositivo parecido com *smartwatch* figura 21 que funcionava como um rádio bidirecional ou telefone celular com recursos avançados.

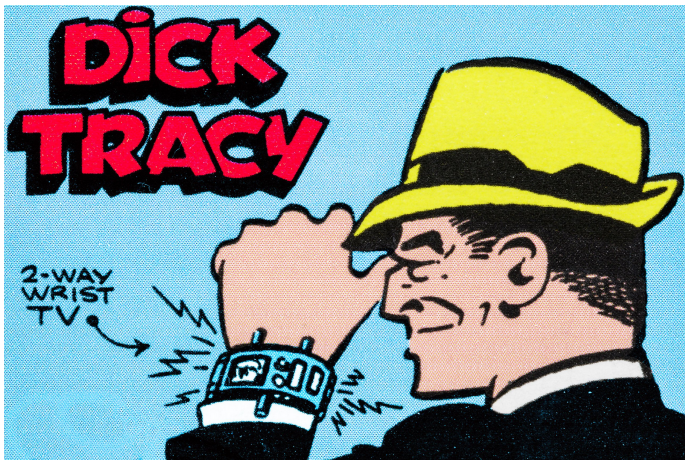


Figura 21: 1930: Dick Tracy e seu relógio

[4]Chester Gould (Pawnee, 20 de novembro de 1900 - Woodstock, 11 de maio de 1985) foi um cartunista norte-americano. Gould foi o criador dos quadrinhos Dick Tracy

- **1960** Os *smartwatches* invadem a cultura pop primeiro com Dick Tracy, mas se tornou ainda mais prevalente nos anos 60 com o capitão James T. Kirk falando em seu relógio de pulso futurista (figura 22) e mais tarde nos anos 80 com Michael Knight chamando KITT em seu famoso relógio preto.

Figura 22: 1960: Star Trek

- **1972** - Em 1972, temos o primeiro relógio digital totalmente elétrico, da *Hamilton Watch Company*, banhado a ouro de 18 quilates. Ele possuía LEDs e você precisava apertar um botão para ver a hora. c

Figura 23: 1972: Pulsar

- **1978 - 1980** Seiko adquire a marca “Pulsar” e produz vários modelos de relógios digitais com recursos avançados. A marca Pulsar posteriormente evoluiu para a série RC, que se tornou mais parecida com os *smartwatches* que conhecemos hoje. O "computador de pulso RC-20"(figura 24) inclui um microprocessador Z-80 de 8 bits, 8 KB de ROM (armazenamento) e 2 KB de RAM. Ele também incluiu um display LCD de matriz de pontos que era sensível ao toque. Os futuros modelos da série RC tornaram-se cada vez mais avançados, como o RC-4000 PC, que foi rotulado como o "menor terminal de computador do mundo".

Figura 24: 1978: Seiko

- **1982** - Como o usado por *James Bond*[1] em *Octopussy*[2], este relógio ’inteligente’ precisava de um adaptador e um receptor colossal para mostrar imagens de TV granuladas abaixo do mostrador de hora digital (figura 25).

Figura 25: 1982: Seiko TV Watch

[1]James Bond, também conhecido pelo código 007, é um agente secreto fictício do serviço de espionagem britânico MI-6, criado pelo escritor Ian Fleming em 1953.

[2]Octopussy é um filme estadunidense-britânico de 1983, dos gêneros ação e espionagem, dirigido por John Glen.

- **1983** - O Data-2000 (figura 26) podia armazenar dois memorandos e eventos de calendário, também funcionava como uma calculadora. A Seiko foi bastante proeminente na área de *computerwatches* nos anos 80. Também lançou o UC-2000, o RC-1000, o Memo Diary e o UC-3000 um ano após o Data-2000.

Figura 26: 1983: Seiko Data-2000 (1983)

- **1985** - O Sinclair FM Weistwatch Radio (figura 27) nunca passou do estágio de protótipo, criado pela mesma empresa por trás do ZX Spectrum [3]. Esta talvez seja a primeira intersecção de um smartwatch e a industria de jogos, pois o ZX Spectrum além de sua popularidade como um microcomputador pessoal era usando como meio de entretenimento. Segundo (VASCONCELLOS, 2020) os consumidores acabaram por preferir estes sistemas como principais fontes de entretenimento, visto que era muito mais interessante ter um sistema eletrônico multi-uso do que um sistema que seria utilizado exclusivamente para jogos.

Figura 27: 1985: Sinclair FM Wristwatch Radio

[3] O Sinclair ZX Spectrum foi um dos mais influentes microcomputadores europeus de 8 bits durante a década de 1980. Foi lançado na Inglaterra em 1982 pela companhia Sinclair Research.

- **1985** - O Seiko MessageWatch (figura 28) podia exibir IDs de chamadas (usando frequências de banda lateral FM) e também pode exibir atualizações sobre uma variedade de assuntos, como placares esportivos, preços de ações e previsões do tempo. Podemos extrapolar e dizer que este *smartwatch* está para o rádio, assim como os *smartwatches* atuais estão para internet.

Figura 28: 1995: Seiko MessageWatch

- **1995** - Com capacidade de enviar um sinal de socorro que poderia ser captado em qualquer lugar dentro de 90 milhas náuticas, aproximadamente 166,68 kilometros, quase a distância de ida e volta de São Paulo/Campinas. O Breitling Emergency Watch (figura 29) foi creditado por ajudar no resgate de dois pilotos britânicos depois que seu helicóptero caiu na Antártica em 2003. Em 2013, uma versão chamada Emergency II foi lançada.

Figura 29: 1995: Breitling Emergency Watch

- **1998** - Apelidado de 'Pai da computação vestível', Steve Mann[4] iniciou o projeto do primeiro relógio com Linux (figura 30) em 1998. "Projetado para se comunicar sem fio com PCs, telefones celulares e outros dispositivos habilitados para sem fio, o 'relógio inteligente' tinha a capacidade de visualizar mensagens de e-mail e receber mensagens semelhantes a um pager[5]. O projeto tinha ambição de possuir uma tela de alta resolução e já se pensava em aplicativos que permitiriam que o relógio fosse usado como um dispositivo de acesso para vários serviços através da internet.

Figura 30: 1998: Linux Wristwatch

- **2000** - A International Business Machines Corporation (IBM) mostra a evolução do protótipo de relógio rodando o sistema operacional Linux (versão 2.2) figura 31. Este protótipo já incluía 8MB de memória, um acelerômetro, um motor de vibração e um sensor de impressão digital. Mais tarde, a IBM começou a colaborar com a Citizen Watch para criar o mesmo dispositivo com o nome "WatchPad". O projeto foi descontinuado por volta de 2001-2002. O WatchPad teria incluído uma tela 320x240 QVGA, Linux 2.4, Bluetooth, 8 MB de RAM e 16 MB de Armazenamento.

Figura 31: 2000: IBM Linux Wristwatch

[4]William Stephen George Mann (1962) é um engenheiro, professor e inventor canadense conhecido por seu trabalho em realidade aumentada, fotografia computacional, particularmente computação vestível e imagens de alta faixa dinâmica.(HDR)

[5]Pager ou bipe, como o aparelho também era chamado no Brasil, foi o meio de comunicação móvel que precedeu os celulares.

- **2002** - Premiado como o 'melhor da Comdex 2002' [6], o Fossil Palm Pilot figura 32 apresentava um visor de 160x160, 2MB de memória interna e aplicativos Palm como a agenda de contatos, bloco de notas, lista de tarefas e uma calculadora. Ele tinha uma caneta integrada na pulseira para uso na tela.

Figura 32: 2002: Fossil Palm Pilot

- **2003** - Lançado em 2003, descontinuado em 2008, o Microsoft SPOT figura 33 foi um relógio da Microsoft em parceria com outras empresas

Figura 33: 2003: Microsoft SPOT

[6]COMDEX (uma abreviação de *COMputer Dealers' EXhibition*) foi uma feira de tecnologia que acontecia em Las Vegas em novembro de 1979 até 2003.

- **2003** - A Garmin tem uma posição forte na arena de relógios esportivos com GPS; é uma área na qual está envolvida há mais de 10 anos. O Garmin Forerunner figura 34 abriu um mercado de nicho em que a Garmin atua até hoje.

Figura 34: 2003: Garmin Forerunner

- **2012** - Temos um período de repetições de dispositivos similares, bolha da internet, evolução e foco nos *smartphones* e quase dez anos depois o enorme sucesso Nike+ Fuelband figura 35. Ela rastreva seus passos, oferecia sincronização automática usando Bluetooth e a segunda edição, lançada em 2013, melhorou as configurações de luz ambiente para que o display fosse melhor em ambientes mais escuros.

Figura 35: 2012: Nike+ Fuelband

- **2013** - O Sony SmartWatch figura 36 original era um dispositivo complementar para a linha de smartphones Xperia, rodando uma versão modificada do Android, foi sucedido pelo SmartWatch 2 em 2013.

Figura 36: 2012: Sony SmartWatch

- **2013** - Entre 2012 e 2013 o mercado de smartwatches ganha força e muitas empresas começam a trabalhar com tecnologia *wearable*. O aumento da exposição e da imprensa relacionada ao Google Glass impulsionou outras áreas do mercado. Em julho de 2013, diversos rumores e anuncios resultaram em uma enorme lista de empresas que estavam trabalhando em seus smartwatches, incluindo: Acer, Apple, BlackBerry, Foxconn / Hon Hai, Google, LG, Microsoft, Qualcomm, Samsung, Sony, Toshiba, HP, HTC, Huawei, Motorola, Lenovo, Nokia entre outras.

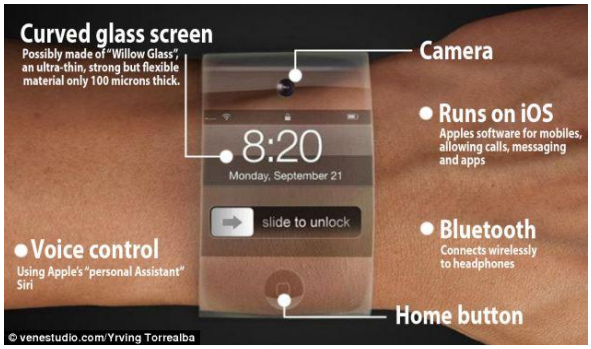


Figura 37: A Corrida das pulseiras *The Wrist Rush* (2012-2013)

- **2013** - Pebble figura 38, foi um *smartwatch* lançado via financiamento coletivo, arrecadou $ 10,2 milhões no Kickstarter[7], tornando-o um dos produtos de financiamento coletivo de maior sucesso já lançado. O Pebble, lançado em julho de 2013, foi vendido pela Best Buy[8] esgotado-se em cinco dias.

Figura 38: 2013: Pebble

- **2014** - O Gear Fit figura 39 foi o primeiro *smartwatch* da coreana Samsung. Não obteve sucesso, e posteriormente foi relançado em novas versões, como Gear S.

Figura 39: 2014: Samsung Gear Fit

[7]Kickstarter é o maior site de financiamento coletivo do mundo e que busca apoiar projetos inovadores. O site foi fundado em 2008 por Perry Chen, Yancey Strickler, e Charles Adler

[8]Best Buy é uma empresa multinacional de eletrônicos dos Estados Unidos

- **2014** - O sistema operacional Android Wear (posteriormente renomeado para Wear OS) foi anunciado em março de 2014 na Google I/O, conferência anual de desenvolvedores da Google, e a escolha do trio de dispositivos de lançamento o que mais se destacou foi o da Motorola. O Moto 360 figura 40 foi muito popular, assumiu um status quase mítico graças ao seu design redondo e ao fato de parecer muito mais elegante do que os outros dois estreantes do Android Wear, o LG G Watch, e o ASUS ZenWatch.

Figura 40: 2014: Moto 360

- **2014** - A grande novidade foi o Samsung Gear S figura 41 possuir conectividade 3G, o que significava ele pode operar sem um smartphone - uma novidade para os smartwatches Samsung, que antes exigiam um aparelho Galaxy.

Figura 41: Samsung Gear S

- **2015** - O Apple Watch figura42, foi lançado em 24 abril de 2015 no Estados Unidos e rapidamente tornou-se o dispositivo vestível mais vendido do mundo: 4,2 milhões de unidades foram vendidas no segundo trimestre do ano fiscal de 2015. A Apple, posteriormente, introduziu novas gerações do Apple Watch com componentes internos melhorados, adotando o título 'series' a cada novo modelo.

Figura 42: 2015: Apple Watch

- **2020** - Bangle.js - Após uma campanha de financiamento coletivo bem-sucedida, o *smartwatch* programável em Javascript, Bangle.js figura43, foi lançado. Com foco no público de tecnologia, este dispositivo possibilita instalar novos aplicativos da web ou desenvolver o seu próprio usando JavaScript ou uma linguagem de programação gráfica. O Bangle.js possui suporte habilitado para inteligência artificial e possui bluetooth de baixa energia, GPS, monitor de frequência cardíaca e um acelerômetro com giroscópio.

Figura 43: 2020: Bangle.js

2.1.1 Sistemas operacionais de smartwatches atuais

Com a capacidade computacional os *smartwatches* podem ser considerados mini-computadores de pulso. O que nos leva a um dispositivo dotado de sistema operacional.

A tabela 1 apresenta uma lista com os principais sistemas comerciais e open-source para wearables na atualidade.

Tabela 1: Sistemas operacionais de *smartwatches* atuais.

	Nome	Empresa
1	Wear OS	Google
2	watchOS	Apple
3	Tizen	Samsung e Intel
4	AsteroidOS	Open-source
5	Sailfish OS	Open-source
6	Ubuntu Touch	Canonical UK Ltd
7	Bangle.js	Open-source

- 1 - O Wear OS, anteriormente conhecido como Android Wear, é um sistema operacional para smartwatch desenvolvido pela Google Inc.
- 2 - watchOS é um sistema operacional móvel proprietário desenvolvido pela Apple Inc. para rodar no Apple Watch.
- 3 - Tizen foi um sistema operacional baseado em Linux para várias plataformas, incluindo smartwatches. O Tizen foi um projeto dentro da Linux Foundation e é facilitado por um Grupo de Orientação Técnica (TSG) composto por Samsung e Intel, entre outros. O projeto foi descontinuado.
- 4 - AsteroidOS é uma substituição de firmware de código aberto para alguns dispositivos que rodam Android Wear.
- 5 - Sailfish OS é um sistema operacional baseado em Linux para várias plataformas, incluindo smartwatches Sailfish.
- 6 - Ubuntu Touch é um sistema operacional desenvolvido pela Canonical UK Ltd e Ubuntu Community para várias plataformas móveis, incluindo smartwatches.
- 7 - Bangle.js é um sistema operacional baseado em Javascript para o dispositivo de mesmo nome. Tem foco na comunidade desenvolvedora por ser de código aberto e customizável.

2.2 Sensores no Controle de Jogos

> *Objetos estáticos e mudos tornar-se-ão seres dinâmicos e comunicantes, incrustando inteligência nos ambientes. No momento em que os objetos se tornarem inteligentes, o mundo das coisas e o mundo humano estarão comunicando-se sob condições inéditas ((SANTAELLA, 2003)*

Diversos jogos utilizam acelerômetros, dentro dos controles, para mensurar os movimentos da mão do usuário em três dimensões, ampliando os aspectos de interação humana e imersão em um game (FITZ-WALTER; JONES; TJONDRONEGORO, 2008).

Estes mesmos sensores estão disponíveis nos dispositivos *wearables*, por este motivo na próxima sub-seção (2.2.1) e na seção (2.3) revisa-se estes sensores conceitualmente e sua aplicabilidade.

2.3 Acelerômetro

O acelerômetro é um dispositivo projetado para medir a aceleração de um corpo quando submetido a uma força externa, capaz de medir as acelerações dinâmica e estática. São consideradas acelerações dinâmicas as decorrentes da variação de velocidade em um deslocamento, vibrações e choques; e a aceleração estática é aquela decorrente da aceleração gravitacional. No mecanismo interno do acelerômetro ocorre a conversão de um vetor de aceleração em um sinal elétrico, que pode ser coletado e processado por sistemas eletrônicos (SILVA, 2015).

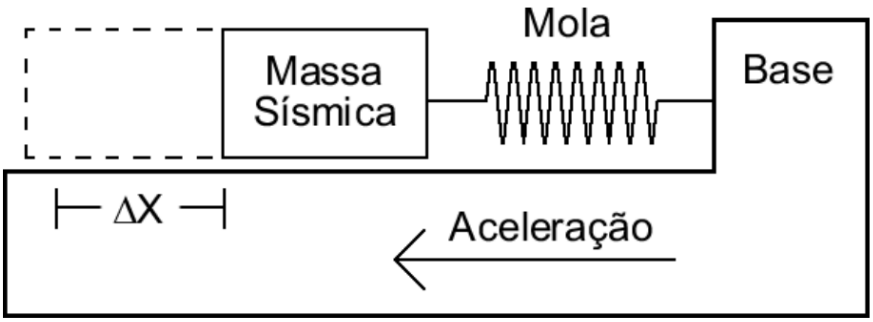


Figura 44: Princípio básico de um acelerômetro (BRUXEL, 2010)

2.4 Giroscópio

Enquanto os acelerômetros medem a aceleração linear, os giroscópios medem a rotação angular. Para fazer isso, eles medem a força gerada pelo que é conhecido como Efeito Coriolis.

2.4.1 Efeito Coriolis

O Efeito Coriolis nos diz que quando uma massa (m) se move em uma direção particular com uma velocidade (v) e uma taxa angular externa (Ω) é aplicada, (seta vermelha) em figura45. O Efeito Coriolis gera uma força (seta amarela) em figura 45 que causa um deslocamento perpendicular da massa. O valor deste deslocamento está diretamente relacionado à taxa angular aplicada, conforme a equação 2.1.

Uma das maneiras que os giroscópios eletrônicos funcionam é através de um sensores capacitivos medindo o deslocamento da massa ressonante e sua estrutura devido ao efeito Coriolis.

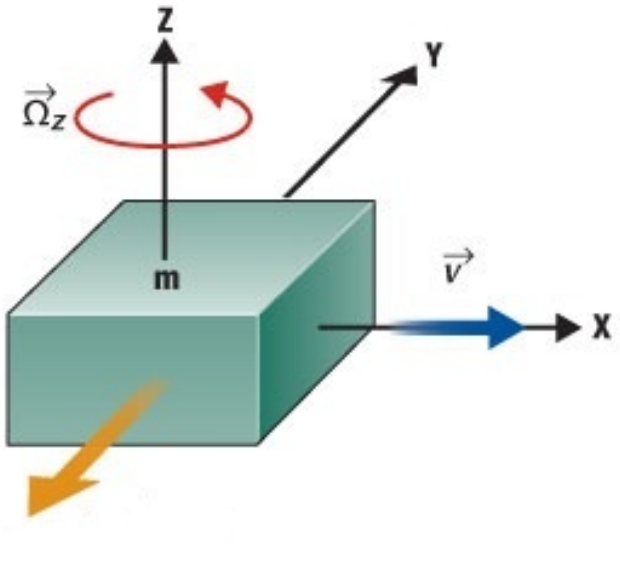


Figura 45: Efeito Coriolis

$$\vec{F_{\text{cor}}} = 2m\vec{\Omega} \times \vec{v} \tag{2.1}$$

2.4.2 Ângulo de Tait–Bryan

Os ângulos de Tait–Bryan (também chamados de ângulos náuticos ou *yaw, pitch and roll*), (Figura 46) são uma variante dos ângulos de Euler, normalmente usados para

aplicações aeroespaciais (onde são frequentemente chamados de ângulos de Euler). Os ângulos de Euler são uma representação mínima (um conjunto de três números) da orientação relativa. Este conjunto de três ângulos descreve uma sequência de rotações em torno dos eixos de um referencial. Existem, entretanto, conjuntos que descrevem a mesma orientação: diferentes combinações dos eixos x, y e z levam a diferentes ângulos de Euler (ROBOTICS. . ., 2018).

A única diferença é que os ângulos de Tait-Bryan representam rotações sobre três eixos distintos (por exemplo, x - y - z), enquanto ângulos de Euler adequados usam o mesmo eixo para a primeira e terceira rotações elementares (por exemplo, z - x - z) (STANLEY; LEE, J., 2018).

Uma vez fixado um quadro de referência, *yaw, pitch and roll* representa o ângulo de rotação em torno dos eixos z, y e x, respectivamente. Se a rotação ocorrer sobre os eixos do sistema de coordenadas original, que permanece imóvel, a rotação é chamada *extrínseca*, e quando ocorre sobre os eixos do sistema de coordenadas em rotação, que muda sua orientação após cada rotação elemental, o a rotação é chamada *intrínseca*. É interessante saber que z - y - x rotações intrínsecas correspondem a x - y - z rotações extrínsecas, e o mesmo se aplica a todas as rotações de ordem reversa.

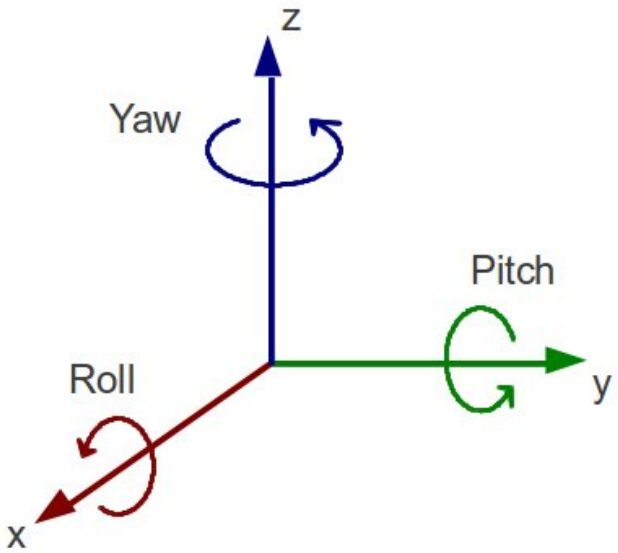


Figura 46: Ângulos Yaw (ψ), pitch (θ) e roll (φ)

2.4.3 Quatérnios

Quatérnios [9] São entidades quadridimensionais que estendem os números complexos, e representam os elementos de $\mathbb{R}^4$ (HAMILTON, 1847).

Como representa um elemento de $\mathbb{R}^4$, um quatérnio pode ser escrito como:

$$q = (q_0, q_1, q_2, q_3)$$

onde q_0, q_1, q_2 e q_3 são números reais, mas são geralmente representados desta forma:

$$\mathbf{q} = q_0 + \mathbf{i}q_1 + \mathbf{j}q_2 + \mathbf{k}q_3$$

onde **i**, **j** e **k** são a base ortonormal padrão em $\mathbb{R}^3$ (KUIPERS, 1999); sempre segue a fórmula fundamental da álgebra de quatérnio:

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1$$

Os quatérnios têm um papel fundamental no projeto, já que o giroscópio do Apple Watch trabalha com quatérnios, todavia, a álgebra por trás das rotações dos quatérnios não será tratada, pois, sua complexidade não permite simplificar os conceitos principais em poucas linhas; entretanto, ele pode ser encontrado em (KUIPERS, 1999, p. 127–134).

Além disso, os quatérnions podem ser utilizados como uma alternativa aos métodos de representação do espaço tridimensional, são mais intuitivos e econômicos em relação ao uso de memória e tempo de processamento se comparados a outros métodos, como, por exemplo, os ângulos de Euler. (FRANQUEIRA, 1993)

[9] Inventados em 1843 pelo famoso físico matemático William Rowan Hamilton, já haviam sido descritos por Gauss em 1819, mas seu trabalho só foi publicado em 1900 (PUJOL, 2012).

3 SISTEMA PARA CONTROLE DE JOGOS BASEADO EM GESTOS

3.1 Desenvolvimento

Neste capítulo apresenta-se o projeto desenvolvido. O experimento foco deste trabalho foi feito utilizando um Apple Watch Series 4, acompanhado de um iPhone XR.

Neste capítulo aborda-se a arquitetura para coletar e processar os dados, inclusive a rede neural embarcada e o processo de avaliação de um gesto.

3.2 Reconhecimento de Atividades Humanas

O reconhecimento de atividade humana, ou do inglês, *Human Activity Recognition*, baseado em sensores tem sido usado em muitas aplicações reais, com aplicações em muitas áreas, como interação homem-máquina, medicina, militar e segurança.

A popularização de dispositivos inteligentes portáteis e vestíveis, como *smartphones* e *smartwatches*, permitiu a fácil coleta de dados de atividades humanas usando vários sensores integrados aos dispositivos. Antes da popularização destes dispositivos inteligentes, o reconhecimento da atividade humana era realizado acoplando sensores ao corpo do usuário, o que era complicado (KHAN et al., 2010).

Neste trabalho tem-se foco na utilização de um *smartwatch* para reconhecimento de gestos para controle de jogos. Durante os experimentos, alguns testes também foram feitos utilizando um sensor acoplado ao braço através de um sistema micro-controlado, até por fim, a utilização de um *smartwatch* padrão de mercado para o mesmo fim.

De modo a abordar a utilização dos dados brutos dos sensores no reconhecimento da atividade humana por wearables, apresenta-se neste trabalho um método baseado em Redes Neurais Artificiais (RNA) que utiliza dados do acelerômetro e giroscópio para o reconhecimento de padrões.

3.3 Classificação de gestos utilizando redes neurais

A classificação de atividades e gestos humanos usando acelerômetros usou redes neurais (seção 3.3), já foi explorada muitos casos. Em (ZENG, M. et al., 2014) (ZHENG et al., 2014) (WAGNER et al., 2017) (IGNATOV, 2018) (LEE, S.-M.; YOON; CHO, 2017) (ZENG, Z.; GONG; ZHANG, 2019) sistemas usando baseados em Redes Neurais com

melhores resultados em comparação com outros sistemas, com grande destaque para o uso de redes com camadas *fully connected* (DALMAZZO; WADDELL; RAMÍREZ, 2021).

Neste projeto utilizou-se do *smartphone* como *gateway* para a avaliação e predição dos gestos.

Esta integração será feita através dos *frameworks* disponíveis nos próprios sistemas operacionais, neste caso, watchOS (relógio), iOS (smartphone) e macOS (computador). Nas Seções 3.6.1, 3.6 e 3.6.2 adentra-se em mais detalhes de cada sistema operacional.

3.4 Redes Neurais Artificiais

A computação neural surgiu inspirada no funcionamento do cérebro. O termo neural está relacionado com as redes estudadas na neurociência, as quais serviram como motivação para os modelos atualmente utilizados (HERTZ; KROGH; PALMER, s.d.).

Um dos primeiros estudos que deram origem as Redes Neurais Artificiais (RNAs) foi o trabalho de Warren McCulloch e Walter Pitts em 1943, onde foi proposto um modelo artificial de um neurônio biológico (ao qual chamaram neurônio MCP em homenagem às iniciais de seus nomes) descrevendo suas capacidades computacionais.

Em 1949, Donald Hebb, foi o pioneiro ao demonstrar uma técnica de aprendizado através da variação dos pesos de entrada dos neurônios, sendo uma teoria baseada no reforço das ligações sinápticas entre neurônios. Mais tarde, Widrow e Hoff sugeriram uma técnica de aprendizado conhecida como Widrow-Hoff, ou regra delta, baseada em um método do gradiente descendente para minimização do erro na saída de um neurônio com resposta linear. Essas técnicas de aprendizado são conhecidas e bastante utilizadas até hoje em diversos algoritmos (BRAGA, 2000).

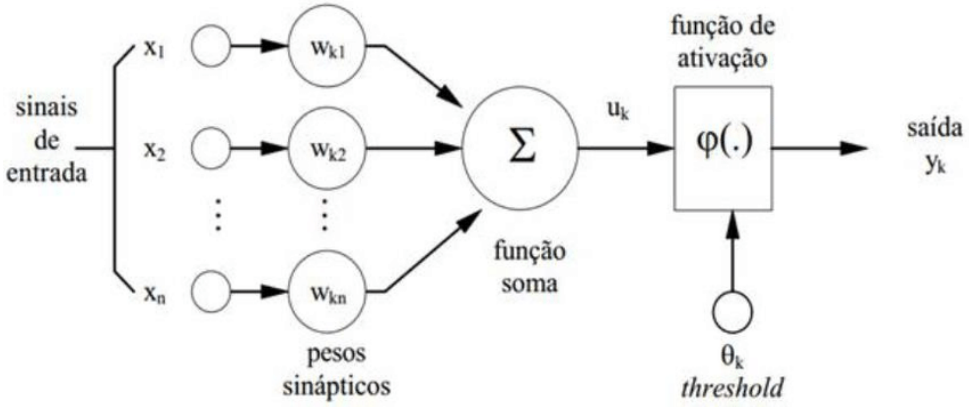
Em 1958, Frank Rosenblatt propôs um novo modelo, onde as RNAs poderiam ser treinadas para classificar certas categorias de padrões, denominado perceptron, este consistia em sinapses ajustáveis as RNAs com neurônios MCP, possuindo três camadas: a primeira com entradas do exterior com conexões fixas (retina), a segunda que recebe impulsos da primeira por conexões (pesos) ajustáveis, enviando saídas a terceira camada (respostas).

No início a RNA fornecia uma saída qualquer em função da condição inicial, porém devido ao ajuste gradual dos pesos o perceptron poderia ser treinado para fornecer saídas mais adequadas conforme o conjunto de treinamento. A ideia de Rosenblatt, era fazer com que as RNAs conseguissem gerar novas descobertas sem a necessidade de regras explícitas fornecidas pelo ser humano (BRAGA, 2000).

RNAs são formadas por conjuntos de diversos neurônios artificiais. Este modelo de

neurônio consiste em receber sinais de entrada x_n nos dendritos do neurônio e retorna um único sinal de saída y no axônio do mesmo. Este modelo é apresentado na Fig. 47.

Figura 47: Representação de um neurônio artificial não linear.



Fonte: Adaptado de (HAYKIN et al., 2009)

A expressão que representa este neurônio, com função de ativação unitária, é apresentado pela Equação 3.1,

$$u_k = y_k = \sum_{i=1}^{m} w_i x_i + b \tag{3.1}$$

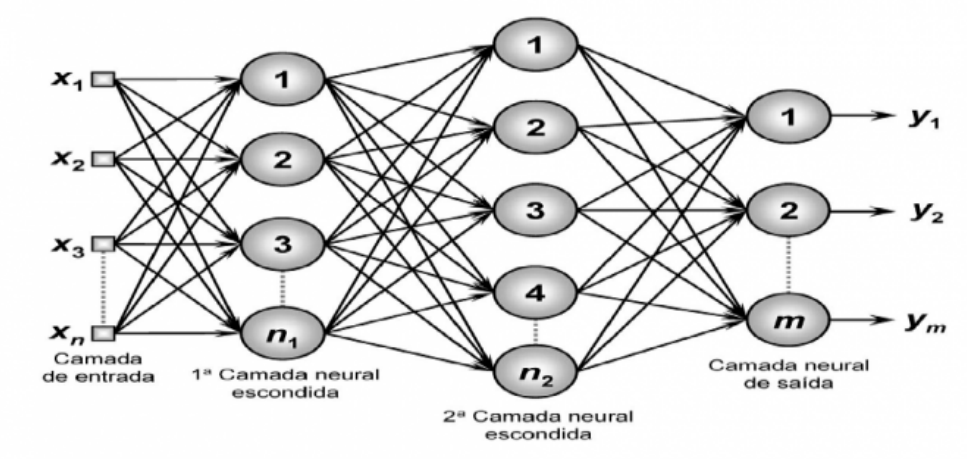
onde x_i são as entradas dos neurônios, w_i são os pesos das entradas, b é o *bias* e y é a saída para uma função de ativação unitária. A função de ativação é o elemento que determina a saída do neurônio em termos do potencial de ativação, muitas das vezes limitando a saída entre $\{0, 1\}$ e $\{-1, 1\}$. Dentre as funções mais utilizadas para esta operação estão as funções sigmoide, linear, tangente hiperbólica, logarítmica e senoidal.

As combinações destes neurônios formam uma Rede Neural Artificial. Dentre as muitas arquiteturas utilizadas, a mais comum é a rede com múltiplas camadas.

Existem diversas estratégias de treinamento supervisionado para RNAs, o mais difundido o algoritmo de retro-propagação de erro (*error backpropagation*). Treinamento supervisionado é definido como o estímulo de uma entrada x e a saída da rede y é comparada com o valor desejado (valor real). A partir desta diferença os pesos são ajustados até que o resultado desejado seja obtido.

(HAYKIN et al., 2009) definiu estas redes como sendo um conjunto de unidades sensoriais que constituem a camada de entrada, uma ou mais camadas ocultas e uma camada de saída. Estas redes são conhecidas como Perceptron Multicamadas ou *Multilayer Perceptron — MLP* (ROSENBLATT, 1962). Na Fig.48, é apresentado um modelo de uma MLP com duas camadas escondidas (*hidden layers*).

Figura 48: Representação de uma MLP com duas camadas escondidas.



Fonte: Adaptado de (HAYKIN et al., 2009).

3.4.1 Redes Neurais Convolucionais

Redes neurais convolucionais consistem em uma variação da arquitetura de redes neurais tradicionais em que parâmetros treináveis podem ser reutilizados em diferentes operações durante o cálculo de *feedforward* de uma rede neural. Isso permite o reconhecimento de um mesmo padrão em distintos subconjuntos dos dados de entrada, tornando essa categoria de arquitetura ideal para reconhecimento de objetos em imagens conforme originalmente descrito por (LECUN et al., 1999). Contudo, uma variação da rede neural convolucional também pode ser útil no reconhecimento de padrões de gestos, conforme visto em 3.3.

3.4.2 Redes Neurais Recorrentes

Segundo (OLAH, 2015), redes neurais recursivas são uma extensão das redes neurais tradicionais que visam permitir a representação de um estado de memória no processo de *feedforward* da rede. Isso é especialmente útil quando os dados de entrada dependem de uma sequência para possuírem coerência, como ocorre no caso de um gesto, que possui um estado inicial e um estado final.

A representação de estado de memória em uma rede neural é viabilizada através de uma célula, i.e., um conjunto de neurônios com operações aplicadas uma vez para cada elemento da sequência de um dado de entrada. Dessa maneira, além de haver um compartilhamento de parâmetros treináveis como nas redes neurais convolucionais, também há um estado armazenado nos neurônios da célula realimentados a essa mesma célula ao executar as operações envolvendo o elemento seguinte da sequência, o que dá origem ao nome recorrente.

3.4.3 Redes (LSTM)

Entre as arquiteturas de redes neurais propostas uma das mais utilizadas na classificação de atividades e gestos são as redes neurais recorrentes (RNN, do inglês *Recurrent Neural Networks*).

As redes LSTM conseguem aprender essa categoria de dependência temporal devido à sua capacidade de armazenar informações por um tempo arbitrariamente longo aprendido pela própria rede. Uma de suas primeiras aplicações veio como a rede de Hopfield, proposta em (HOPFIELD, 1982), esta rede foi aplicada a problemas envolvendo séries temporais, mais especificamente a rede recorrente de memória de longo prazo (LSTM, do inglês *Long Short-Term Memory*), proposta por Sepp Hochreiter e Jürgen Schmidhuber (HOCHREITER; SCHMIDHUBER, 1997), aplicada a mesma classe de problemas.

Uma unidade LSTM comum é composta de uma célula com, *input gate* **i**, um *output gate* **o**, um *forget gate* **f** e um *gate* **g**. A célula lembra valores em intervalos de tempo arbitrários e os três *gates* regulam o fluxo de informações para dentro e para fora da célula. Uma comparação entre uma célula RNN normal e uma célula LSTM pode ser observada na Figura 49.

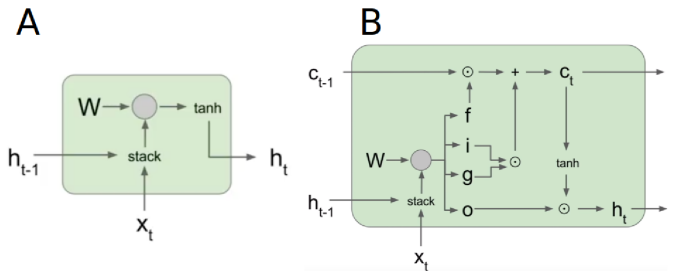


Figura 49: Comparação entre uma célula RNN normal **A** e uma célula LSTM **B**.

O *gate vector* pode ser escrito como

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}, \tag{3}$$

onde $W = \begin{pmatrix} W_i & W_f & W_o & W_g \end{pmatrix}$., o estado da célula é definido como o seguinte:

$$\begin{aligned} c_t =& f \odot c_{t-1} + i \odot g \\ =& \sigma(W_{hf} h_{t-1} + W_{xf} x_t) \odot c_{t-1} \\ &+ \sigma(W_{hi} h_{t-1} + W_{xi} x_t) \odot \tanh(W_{hg} h_{t-1} + W_{xg} x_t) \end{aligned} \tag{4}$$

e o estado oculto é então dado por

$$\begin{aligned} h_t &= o \odot \tanh(c_t) \\ &= \sigma(W_{hho} h_{t-1} + W_{xho} x_t) \odot \tanh(c_t) \end{aligned} \tag{5}$$

A praticidade de ter esta estrutura celular particular é evidente se olharmos para várias células em simultâneo, ou seja, o processamento em várias sequências, como é efetuado na Figura. 50. O treinamento funciona novamente com *Back-propagation-Through-Time*. O gradiente agora pode ser passado sem ser interrompido, ou seja, os problemas de atualizações de peso dispendiosas, o desaparecimento e explosão de gradientes não devem ocorrer mais.

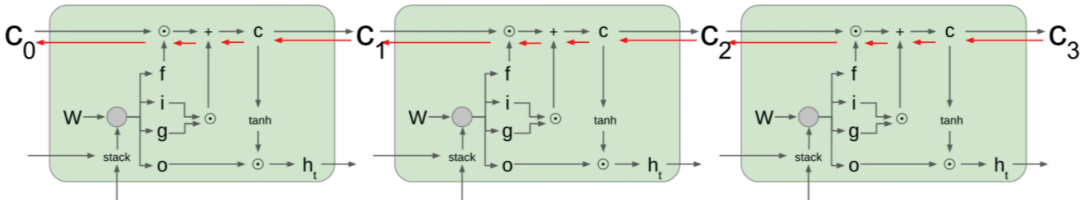


Figura 50: LSTM em várias sequências. A seta vermelha indica o gradiente, que pode fluir ininterruptamente..

3.5 Funções de Ativação e camadas intermediarias

Além da entrada de dados, o usuário deve inserir as camadas intermediárias e finais da rede, que também pode ser conectado entre duas redes neurais diferentes. Essas camadas podem ser de diversos tipos, conforme será visto nas próximas subseções.

3.5.1 Concat: concatenação ou combinação

A concatenação ou combinação é uma abordagem no *deep learning* (aprendizado profundo). Uma camada de concatenação pega entradas e as concatena ao longo de uma dimensão especificada. As entradas devem ter o mesmo tamanho em todas as dimensões, exceto a dimensão de concatenação.

3.5.2 Reshape

Dependendo da arquitetura da rede neural após receber ou concatenar os dados, pode ser necessário reformulá-los (reshap). Por exemplo, algumas arquiteturas ou algoritmos, podem exigir que uma matriz unidimensional de variáveis de saída (y) seja moldada como uma matriz bidimensional com uma coluna com os resultados para cada linha. A rede neural recorrente Long Short-Term Memory (LSTM) em algumas arquiteturas, exige que

a entrada seja especificada como uma matriz tridimensional composta de amostras, passos de tempo e recursos, para obtermos o formato necessário utiliza das camadas de *Reshape*.

3.5.3 Função de Ativação Softmax

A função de ativação *Softmax* é usada em redes neurais de classificação. Ela força a saída de uma rede neural a representar a probabilidade dos dados serem de uma das classes definidas. Sem ela as saídas dos neurônios são simplesmente valores numéricos onde o maior indica a classe vencedora, dados pela Equação 3.2.

$$softmax(x) = \frac{e^{x_j}}{\sum_{k=1}^{n} e^{x_k}}, \quad \text{para } j = 1, ..., n \tag{3.2}$$

Nessa equação, i representa o índice do neurônio de saída (o) sendo calculado e j representa os índices de todos os neurônios de um nível. A variável z designa o vetor de neurônios de saída. Vale notar que a função de ativação *Softmax* é calculada diferentemente das demais apresentadas, visto que a saída de um neurônio depende dos outros neurônios de saída.

3.5.4 Função de Ativação RELU

A *Rectified Linear Unit* (ReLU) é uma função de ativação relativamente simples, no entanto, uma das mais utilizadas ainda. Quando a função ReLU é chamada com alguma entrada, ela retorna 0 para todos os valores negativos, e os números positivos simplesmente mantêm seus valores. A (ReLU) pode ser representada respectivamente pelas Equações 3.3 e 3.4.

$$ReLU(x) = max(0, x) \tag{3.3}$$

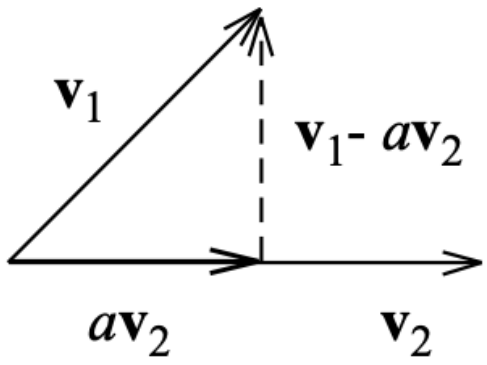
ou

$$ReLU'(x) = \begin{cases} x, & \text{if } x \geq 0 \\ \\ 0, & \text{if } x < 0 \end{cases} \tag{3.4}$$

3.5.5 Produto interno e projeção

Sejam v_1,$v_2 \in \Re^2$ elementos não-nulos, conforme a Fig.51 .

Figura 51: Projeção realizada pelo produto interno no $\Re^2$.



Considere um escalar α tal que αv_2 corresponda à projeção de v_1 na direção de v_2. Então, tem-se a Equação 3.5

$$\alpha v_2 \perp v_1 - \alpha v_2, \tag{3.5}$$

conduzindo a Equação 3.6

$$\langle\, \alpha v_2, v_1 \,-\, \alpha v_2 \,\rangle = 0. \tag{3.6}$$

Logo, tem-se a Equação 3.7

$$\alpha \,\langle v_2, v_1 \,\rangle \,-\, \alpha^2 \,\langle v_2, v_2 \,\rangle \,=\, 0, \tag{3.7}$$

permitindo obter α (alpha) conforme a Equação 3.8

$$\alpha \,=\, \frac{\langle\, v_2, v_1 \,\rangle}{\langle\, v_2, v_2 \,\rangle.} \tag{3.8}$$

Isto significa que a projeção de v_1 na direção de $v_2\,(\,v_2 \,\neq\, 0)$ assume a Equação 3.9

$$proj_{v2}\,(v_1) \,=\, \frac{\langle\, v_2, v_1 \,\rangle}{\langle\, v_2, v_2 \,\rangle}\, v_2 \tag{3.9}$$

Mantendo constante o módulo de v_1, a sua projeção na direção de v_2 aumenta quanto mais colineares forem esses dois vetores.

O produto interno, portanto, permite o estabelecimento de uma associação bem definida entre o vetor de estímulos de entrada de um neurônio e o seu vetor de pesos sinápticos, de modo que o neurônio é mais fortemente ativado quanto mais colineares forem esses dois vetores, supondo norma constante para esses vetores.

3.6 Ambiente de Desenvolvimento

Para o desenvolvimento do projeto, foi utilizada a linguagem de programação Swift, em sua versão 5.0. A arquitetura utilizada para organização do código foi a *Model-View-Controller* (MVC), com algumas adaptações. E para criação da interface gráfica foi utilizado o *framework* SwiftUI.

Tabela 2: Definições e Terminologias

	Definição
Xcode	Ambiente de desenvolvimento integrado (IDE) desenvolvido pela Apple.
Swift	Linguagem de programação utilizada para o desenvolvimento de aplicativos.
SwiftUI	*Framework* da Apple projetado para desenvolvimento declarativo de interfaces.
Core ML	*Framework* da Apple para desenvolvimento de recursos de *Machine Learning*
Create ML	*Ferramenta* da Apple para treinamento de modelos de *Machine Learning*
Framework	Coleção de recursos que auxiliam no desenvolvimento de software

3.6.1 watchOS

O watchOS é o sistema operacional do Apple Watch, desenvolvido pela Apple Inc. É baseado no iOS, o sistema operacional utilizado pelo iPhone e iPod Touch, com funcionalidades semelhantes. Foi lançado em 24 de abril de 2015, com o Apple Watch, o único dispositivo que roda watchOS. O watchOS disponibiliza API chamada *WatchKit* para uso de desenvolvedores, a qual utilizou-se neste projeto.

A oitava e última versão, watchOS 8, foi lançada em 20 setembro 2021, com atualizações para monitorização em saúde, *visuals* e aplicativos.

3.6.2 iOS

O iOS (antes chamado iPhone OS) é um sistema operacional móvel da Apple Inc. desenvolvido originalmente para o iPhone, iPod Touch e o iPad até a introdução do iPadOS em 2019, um sistema derivado do iOS. A primeira versão foi lançada em 2007, e a última, até a publicação deste trabalho foi o iOS 15.1[1].

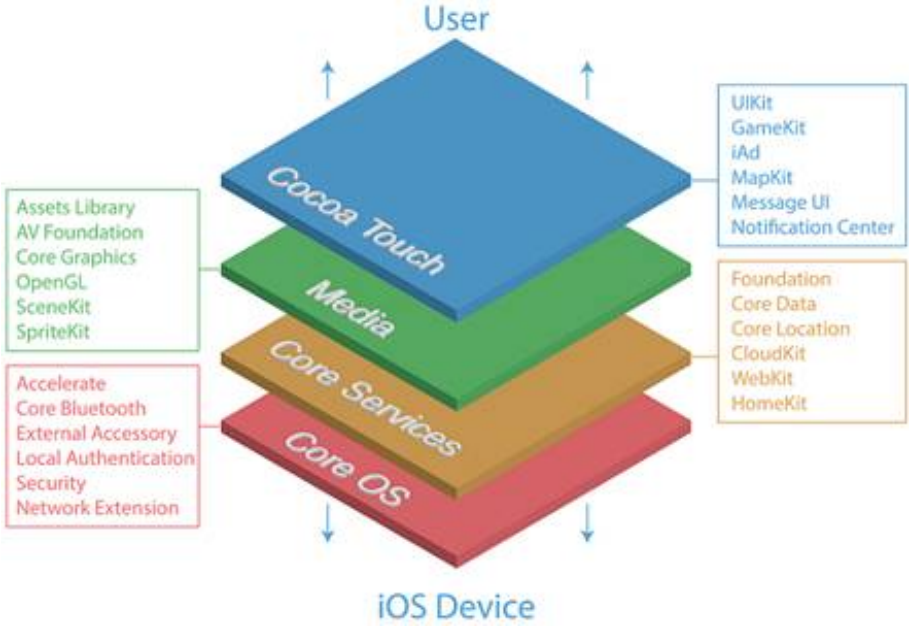
A interface do usuário do iOS é baseado no conceito de manipulação direta, utilizando gestos em multi-toque. A interação com o sistema operacional inclui gestos como apenas tocar na tela, deslizar o dedo, e o movimento de p̈inçaütilizado para se ampliar ou reduzir a imagem. Acelerômetros internos são usados por alguns aplicativos para responder

[1]https://www.apple.com/br/ios/ios-15/

à agitação do aparelho (resultando comumente no comando desfazer) ou rotação do mesmo (resultando comumente na mudança do modo retrato para modo paisagem).

O iOS consiste em sistema com quatro camadas de abstração: a camada Core OS, a camada Core Services, a camada mídia, e a camada Cocoa Touch, conforme visto na Fig.52. Cada camada possui seus respectivos *frameworks* para acesso a recursos do dispositivo e/ou do sistema. Alguns destes *frameworks* podem ser observados na figura 52.

Figura 52: Arquitetura do sistema operacional iOS. (GRUMMITT, 2018)



3.6.3 macOS

O macOS, anteriormente Mac OS X e posteriormente OS X, na fase de desenvolvimento inicialmente chamado Rhapsody Project) é um sistema operacional proprietário desenvolvido e distribuído pela empresa Apple Inc. desde 2001 e destinado exclusivamente aos computadores Mac.

Os lançamentos do Mac OS X de 1999 até 2005 foram lançados nos Macs baseados em processadores PowerPC. A Apple anunciou em 2006 que a partir daquele ano a empresa adotaria a arquitetura Intel, as versões posteriores do sistema foram lançadas para os Macs baseados em Intel com processadores de 32 bits e atualmente de 64 bits. Em 2020, ocorreu uma nova transição de arquitetura, o macOS 11 Big Sur adotou uma arquitetura chamada Apple Silicon baseada em ARM de 64 bits (M1). A última versão é o macOS Monterey.

3.6.4 SwiftUI

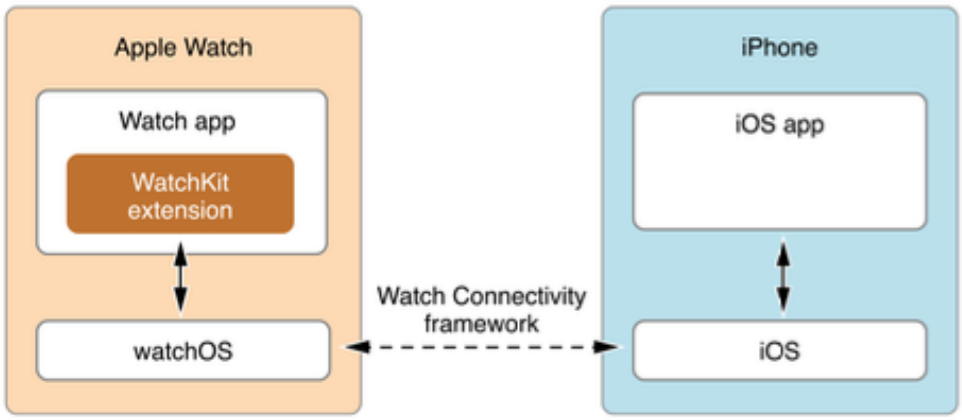
Em 2019 durante a WWDC *WorldWide Developers Conference*, a Apple lançou o *SwiftUI*, um *framework* para o desenvolvimento de interfaces de maneira declarativa. Em linguagens de programação tradicionais, geralmente se cria um elemento de interface do usuário; em seguida, defini-se seu estilo visual; definem-se as cores, plano de fundo, o primeiro plano e outros atributos; e, somente em seguida, configure-se sua hierarquia visual. Com o desenvolvimento de interfaces declarativas, só se precisa especificar um elemento, e após eles podem ser modificados usando operadores de modificação. (VARMA, 2019)

Na figura 72, *SwiftUI* refere-se a camada *View* da aplicação, desenvolvida em *SwiftUI* onde o usuário poderá interagir com o aplicativo.

3.6.5 Watch Connectivity

O *framework Watch Connectivity* é um framework para estabelecer uma comunicação bidirecional entre os aplicativos iOS e WatchOS. A comunicação é usada para compartilhamento de dados. Os dados compartilhados podem ser apenas pequenos pedaços de dados, como *dictionaries*, ou arquivos inteiros. Os dados podem ser enviados instantaneamente ou como transferências em segundo plano(GUSGÅRD, 2018). As figuras 57 e 72 apresentam como o *framework* foi utilizado. A figura 53 a arquitetura interna de um aplicativo iOS e watchOS.

Figura 53: Arquitetura de uma aplicação watchOS com iOS.



Fonte: Adaptado de (GUSGÅRD, 2018)

3.6.6 Multipeer Connectivity

O *framework* do iOS e macOS **Multipeer Connectivity** fornece suporte para descobrir dispositivos próximos e conectar-se a eles. A conexão é realizada em duas fases, primeiro descobrindo pares próximos e, em seguida, iniciando uma sessão entre os pares, a sessão também lida com o envio de dados entre os pares (MEFTAH; ROUVOY; CHRISMENT, 2019).

Neste projeto utilizou-se do **Multipeer Connectivity** como pode ser observado na figura 72 por uma biblioteca *open source*, MultipeerKit, uma abstração de alto nível construída sobre a estrutura **Multipeer Connectivity**, que permite que dispositivos iOS, macOS e tvOS troquem dados entre eles através de redes Wi-Fi, Wi-Fi ponto a ponto e Bluetooth (RAMBO, 2021).

3.6.7 Core ML

O Core ML que foi introduzido pela Apple Inc. em 2017 para permitir que os desenvolvedores de aplicativos pudessem utilizar *machine learning* em seus aplicativos iOS. Ele é uma *framework* de desenvolvimento de aplicativos que permite que você use modelos de *machine learning* pré-treinados em seus aplicativos. (THAKKAR, 2019)

A Fig.54 apresenta o fluxo simplificado de utilização de modelos do Core ML em um aplicativo.

Figura 54: Fluxo de utilização do Core ML.

Fonte: Adaptado de (THAKKAR, 2019)

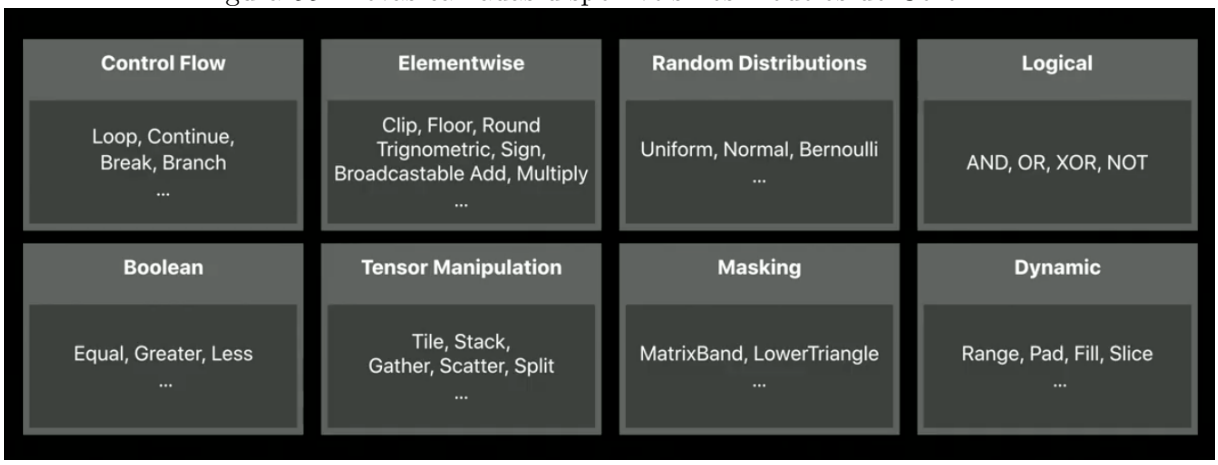
Tradicionalmente os aplicativos com recursos de *machine learning* dependiam de serviços em nuvem para executar os algoritmos. Isso fazia com que sua performance ficasse comprometida dadas as condições de rede e internet. Com o lançamento do Core ML, os aplicativos agora podem executar localmente os algoritmos de *machine learning* otimizados e treinados no dispositivo, levando a uma velocidade de processamento melhor. (THAKKAR, 2019)

Em 2018 a Apple Inc. anunciou a segunda versão do Core ML, o Core ML 2 em 2019 a versão atual e utilizada neste trabalho o Core ML 3.

O Core ML 3, que nas próximas seções será referido apenas como Core ML, possui vasta evolução perante seus antecessores, se destacando o treinamento no dispositivo, até então tinha-se apenas o suporte de "inferência no dispositivo", isso basicamente significa que se treina o modelo em outra máquina e, em seguida, utiliza-se deste modelo treinado para fazer previsões em tempo real no próprio dispositivo. O Core ML 3 agora também oferece suporte ao treinamento no dispositivo. Você pode utilizar a CPU ou GPU e chip de processamento neural do iPhone para treinar seus modelos de *machine learning* e *deep learning*.

Nesta atualização novos categorias de camadas de rede neural foram adicionados no Core ML 3. Além de possuir camadas para diferentes categorias de modelos, o Core ML 3 também possui mais de 100 camadas para operações intermediárias como Masking, Tensor Manipulation, Boolean logic, Control Flow, entre outras conforme Fig. 55. (SAHIN, 2021)

Figura 55: Novas camadas disponíveis nos modelos do Core ML.



Fonte: Adaptado de (APPLE, 2021a)

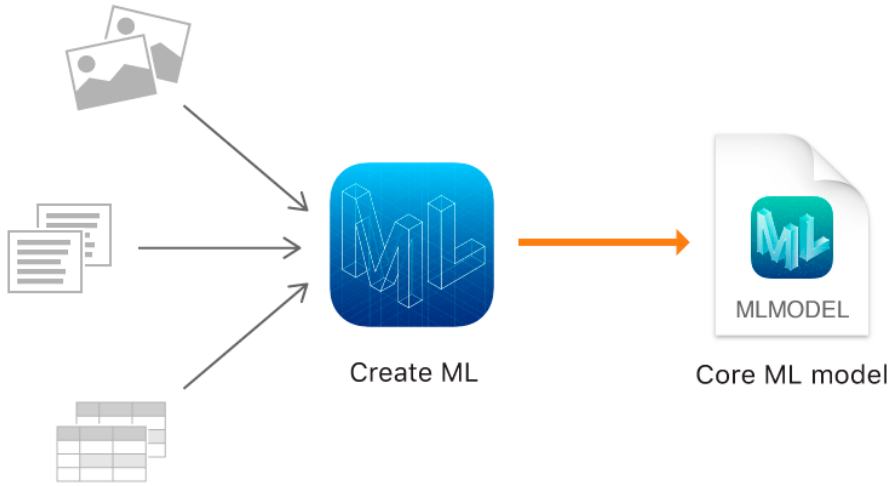
3.6.8 Create ML

Durante a Conferência Mundial de Desenvolvedores de 2018 (WWDC), a Apple Inc. apresentou o *Create ML*. Uma nova ferramenta de aprendizado de máquina que permite aos desenvolvedores importarem seus dados, criarem e treinarem modelos de aprendizado de máquina no macOS. (APPLE, 2021a).

O *Create ML* é baseado em Swift, a linguagem de programação da Apple. Ele permite a criação de modelos de *machine learning* usando uma interface gráfica e simplesmente arrastando as pastas de treinamento e de teste do modelo. Os modelos são criados no Create ML e executados nos aplicativos com o *Core ML*. Com o *Create ML*

pode-se treinar modelos para realizar tarefas como reconhecimento de imagens, extração de significado de texto ou localização de relações entre valores numéricos. A Figura 56 apresenta um diagrama de funcionamento do *Create ML*.

Figura 56: Diagrama de funcionamento do *Create ML*.



Fonte: Adaptado de (APPLE, 2021a)

Na Seção 3.6.8.1 observam-se as categorias de dados suportados pelo Create *Machine Learning* (ML) e na Seção 3.9 tem-se sua utilização mais a fundo.

3.6.8.1 Categorias dede dados suportados pelo Create ML

Antes de iniciar o treinamento do modelo de rede neural, precisa-se entender os dados que esta a se utilizar. A Tabela. 3 a seguir estão as categorias de dados que podem ser treinados em um modelo usando o Create ML. Basta selecionar uma categoria de modelo no aplicativo e adicionar seus dados e parâmetros para iniciar o treinamento.

Tabela 3: Categorias de dados suportados pelo Create ML.

	Categoria de dado
1	Imagens
2	Vídeo
3	Movimento
4	Som
5	Texto
6	Dados Tabulares

Para coletar dos dados de sensores de interesse no Apple Watch, foi desenvolvido

para este projeto um aplicativo para o Apple Watch pareado com o iPhone para coleta dos dados, ao qual abordada em detalhe na Seção 3.8.

3.7 Arquitetura do sistema para aquisição dos dados para treinamento

Um aplicativo watchOS e iOS foi desenvolvido para a aquisição dos dados para serem posteriormente utilizados no treinamento. A arquitetura simplificada é mostrada na Fig. 57.

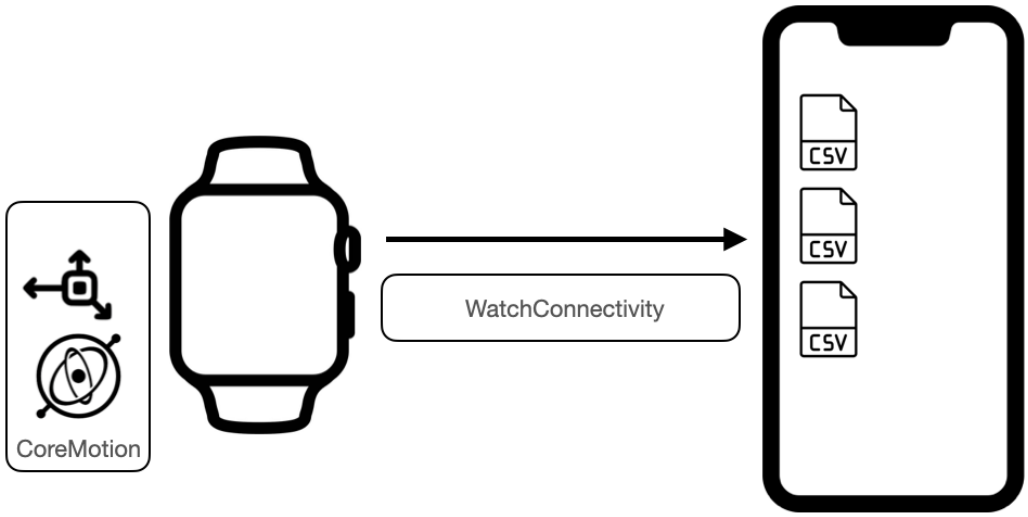


Figura 57: Arquitetura simplificada do processo de aquisição de dados

1. Utilizando o *framework* `CoreMotion` foi captado os dados dos sensores durante o movimento.
2. Os dados são enviados ao iPhone pareado ao relógio.
3. Os dados são armazenados em formato *csv* e agrupados por gesto para posterior utilização no processo de treinamento do modelo.

Na Seção 3.8 têm-se em maiores detalhes as funcionalidades e operação dos aplicativos.

3.8 Aquisição dos dados do acelerômetro e giroscópio

O aplicativo para coleta dos dados do Apple Watch foi desenvolvido para responder de forma simples e rápida e em poucos passos. A cada passo um *feedback* táctil (diferentes categorias de vibrações) é feito, para se saber sobre avanço ou regresso de um passo.

A Fig. 58 exemplifica os passos necessários para iniciar a coleta dos dados.

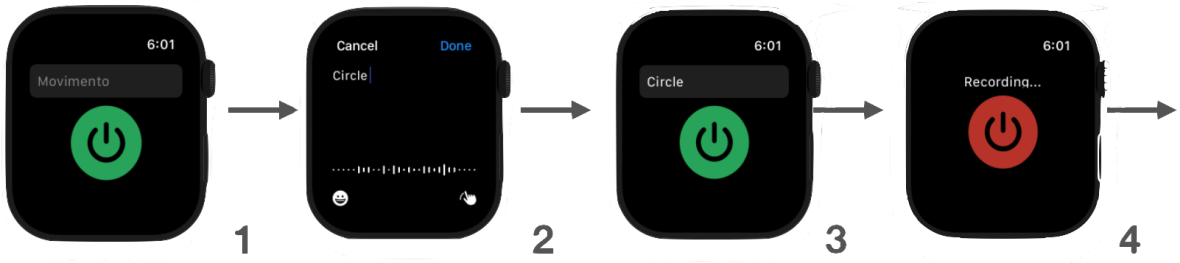


Figura 58: Passos para aquisição de um gesto utilizando o Apple Watch

1. Tela de Início da aplicação, usuário vai para seleção do nome do gesto (exemplo: soco, circulo, etc).

2. Tela de Seleção do nome do gesto. O nome é digitado pelo usuário

3. Usuário volta a tela inicial e inicializa a aquisição dos dados.

4. Tela de aquisição dos dados. Possui um botão para parar a aquisição. Esta tela dá feedbacks hápticos a cada gesto gravado. Ao fim de um ciclo, o qual observa-se a seguir, os dados são transmitidos para o iPhone onde são tabulados e salvos.

Para aquisição dos dados foi utilizado o `CMMotionManager`, o objeto do `WatchKit` responsável por iniciar os serviços que relatam movimentos detectados pelos sensores do dispositivo. Conforme a documentação da Apple (APPLE, 2021b) com este objeto, pode-se receber quatro categorias de dados de movimento: magnetômetro, dispositivo de movimento, giroscópio e acelerômetro (MARK et al., 2014). Neste trabalho, concentrou-se no uso de dados de acelerômetro e giroscópio para nosso reconhecimento de gestos.

— Dados do acelerômetro, indicando a aceleração instantânea do dispositivo no espaço tridimensional.

— Dados do giroscópio, indicando a rotação instantânea em torno dos eixos primários do dispositivo.

A Fig. 59 demonstra os eixos (x, y, z) do acelerômetro e giroscópio do Apple Watch.

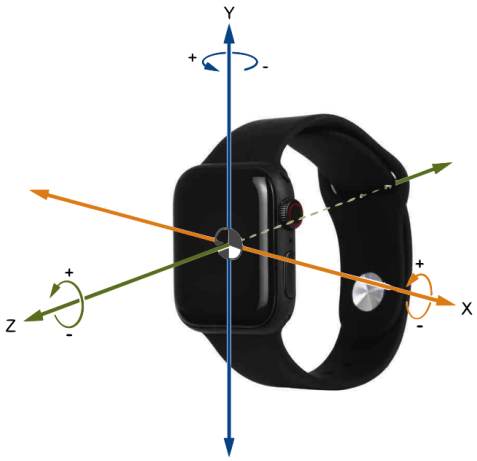


Figura 59: Eixos de referência dos sensores do Apple Watch em (x, y, z).

A aquisição de dados do acelerômetro em (x, y, z) coordenadas e ângulos do giroscópio normalmente chamados de (ϕ, ψ, θ), são medidos com intervalo de 0,01 segundos (100 Hz). Um gesto típico tem entre 0,5 e 2,5 segundos de duração, o que significa 50 a 250 amostras. Neste projeto utilizou-se um limiar de 150 amostras por gesto. Quando o gesto é realizado com menos capturas, os valores faltantes são completados com zeros. Como mencionado, utilizou-se 100 Hz. Deve-se ter cuidado, pois, se solicitarmos uma frequência maior do que a que o hardware suporta, a biblioteca usará o máximo suportado.

O aplicativo através do objeto `CMMotionManager` armazena os valores em um `array`, após a conclusão do ciclo (50 valores ou 1.5 segundos) o gesto é salvo e os valores reiniciados. Assim, cada observação de gesto se estende a 50 amostras. O Gráfico.1 apresenta uma amostra dos dados do acelerômetro capturados por um gesto (x, y, z) por tempo.

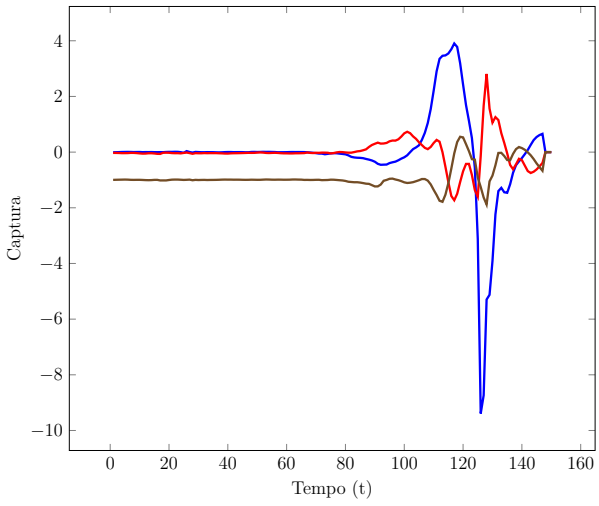


Gráfico 1: Captura de movimento do acelerômetro x (—), y (—), e z (—) no tempo

No giroscópio, têm-se uma forma semelhante com o `CMMotionManager`, porém este método envia cada novo conjunto de dados de movimento do dispositivo processado. Ele recupera os dados que já foram processados para remover as influências ambientais, como, por exemplo, os efeitos da gravidade.

Os dados são obtidos através de um objeto chamado `CMAttitude`, representado no código 1. O objeto `CMAttitude` possui uma propriedade *quartenion* que pode ser vista no código 3.

Listing 1: Objeto CMMotionManager

```
open class CMAttitude : NSObject, NSCopying, NSSecureCoding {

    open var roll: Double { get }

    open var pitch: Double { get }

    open var yaw: Double { get }

    open var rotationMatrix: CMRotationMatrix { get }

    open var quaternion: CMQuaternion { get }

    open func multiply(byInverseOf attitude: CMAttitude)
}
```

Listing 2: Objeto CMQuaternion

```
public struct CMQuaternion {

    public var x: Double

    public var y: Double

    public var z: Double

    public var w: Double

    public init()

    public init(x: Double, y: Double, z: Double, w: Double)
}
```

A coleta de dados também captura 50 valores e o gesto é salvo, assim cada observação de gesto através do giroscópio também se estende a 50 amostras durante o estágio de treinamento. Diferente do acelerômetro onde se utilizam os valores de (x, y, z), no giroscópio utilizam-se os (*yaw, pitch and roll*) e os valores do *quaternion* (x, y, z, w).

O Gráfico 2 apresenta uma amostra dos dados do giroscópio capturados por um gesto (*yaw, pitch and roll*) no tempo. Já o Gráfico 3 apresenta uma amostra dos dados do giroscópio, do mesmo gesto para os valores do quatérnion (x, y, z, w).

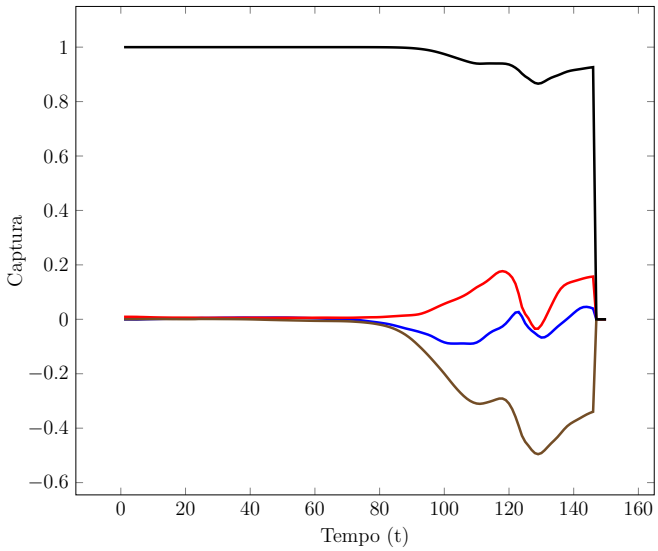


Gráfico 2: Captura de movimento do giroscópio q_x (—), q_y (—), e q_z (—) q_w (—) no tempo

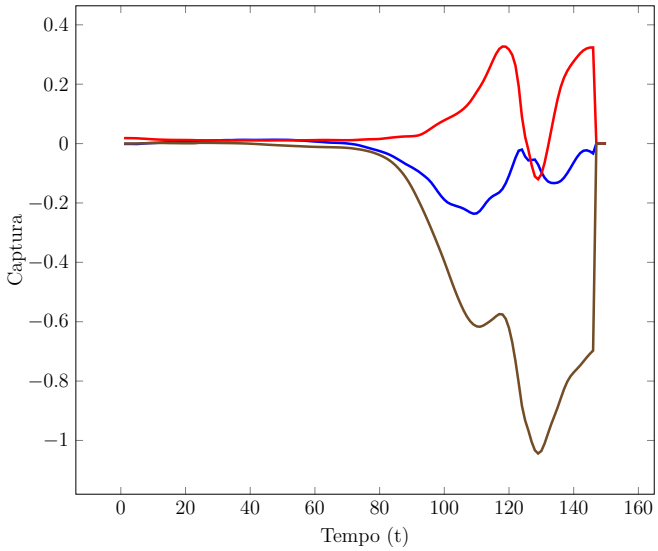


Gráfico 3: Captura de movimento do giroscópio *yaw* (—), *pitch* (—), e *roll* (—) no tempo

O resultado de cada movimento são 150 coletas de 10 variáveis. A Figura 60 mostra um *sample* de uma coleta.

x_accel	y_accel	z_accel	w_motion	x_motion	y_motion	z_motion	pitch_motion	roll_motion	yaw_motion
-0.2932586666921875	-0.117156982421875	-0.985870361328125	0.999098538995717	0.0340169273699982 36	0.0253991030302781 4	0.0	0.0680249758346089 1	0.0508920315941863 16	-0.0017320055432764 69
0.001251220703125	-0.060760498046875	-1.0070953369140625	0.9990674205507543	0.0344471354368769 7	0.0260126407794842 48	0.0010858458264852988	0.0689411113860781	0.0520490396475006 7	0.0003784434582589469
0.02459716796875	-0.1164093017578125	-0.9948577880859375	0.9990256418647336	0.0349167429892102 35	0.0269186832341866 24	0.0020253893824482283	0.0699314697798287 8	0.0538008576328947 6	0.0021723176692678 62
-0.02467346191406 25	-0.17071533203125	-0.9243316650390625	0.9989966832719273	0.0346988615260754 3	0.0283097873878243 76	0.0004469524559899 6847	0.0694091191848543 2	0.0566985717468593 2	-0.0010742134892609 808
0.1259765625	-0.275360107421875	-0.8223724365234375	0.9989756983792766	0.0347974080677795	0.0289237317072905 26	0.0003679008543783 7794	0.0696009932359271 7	0.0579352031329345 8	-0.0012809956237490 247
0.2708282470703125	-0.3026275634765625	-0.8591156005859375	0.9989932809029878	0.0346174006571447 9	0.0285165580414233 6	0.0010080495225354147	0.0692779961659864 8	0.0570737198642587 7	3.9826704226521 84e-05
0.17138671875	-0.250518798828125	-0.8881378173828125	0.9989903269863003	0.0348855045397051 95	0.0282850464216513 9	0.0011241695256041 24	0.0698208726496296 6	0.0566025985502928 7	0.0002732585275222921

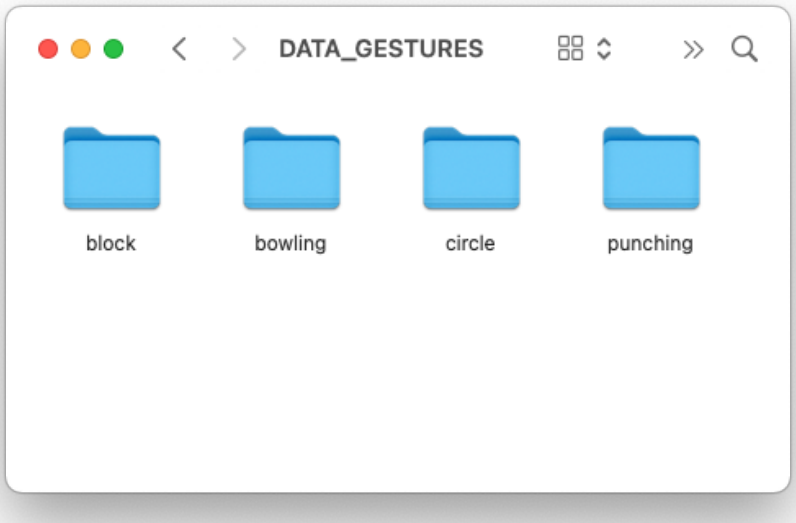
Figura 60: *Sample* de uma coleta do acelerômetro e giroscópio

3.9 Treinamento da Rede Neural através do Create ML

O *Create ML* permite a criação de modelos de *machine learning* usando uma interface gráfica e simplesmente arrastando as pastas de treinamento e de teste do modelo. Os modelos são criados no Create ML e executados nos aplicativos com o *Core ML*.

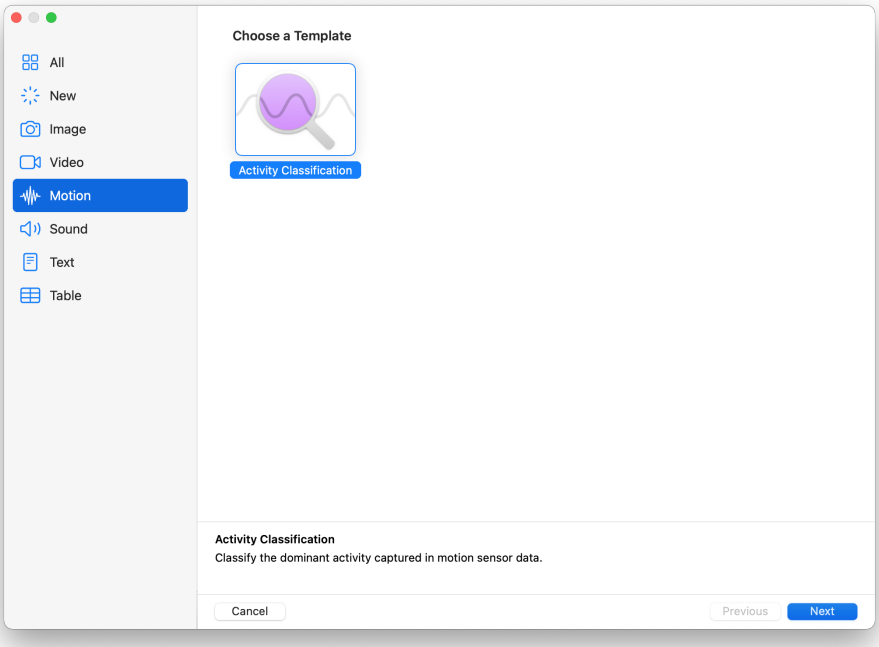
Para este projeto utilizou-se de quatro gestos: *punching*, *block*, *bowling* e *circle*. Cada conjunto de dados foi salvo em uma pasta, onde o nome da pasta já funciona como *label* daquele conjunto. A Figura 61 apresenta a organização dos dados para o treinamento.

Figura 61: Organização dos dados para o treinamento



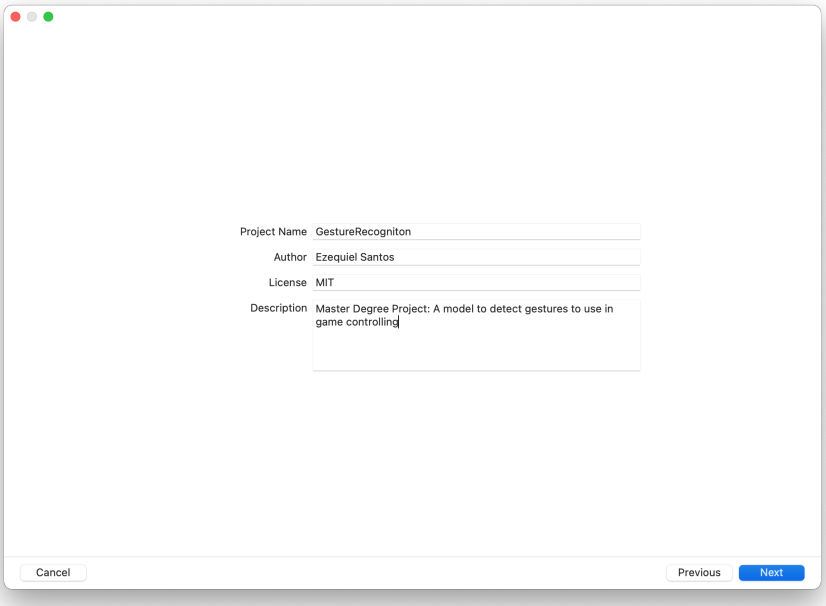
Após a preparação dos dados, inicia-se o treinamento do modelo. O Create ML é instalado com o Xcode. Na tela inicial, figura 62, deve-se escolher um *Activity Classification*.

Figura 62: Tela inicial do Create ML para escolha do modelo.



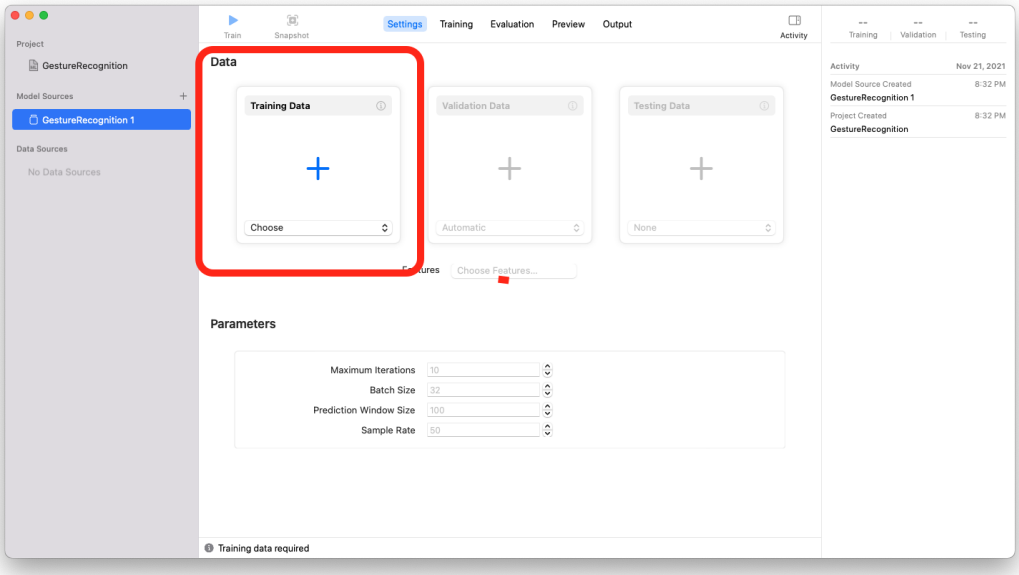
Após isso adicionam-se algumas informações básicas do projeto, conforme a Figura 63 e seguimos no *Next*.

Figura 63: Adicionada informações do projeto no Create ML.



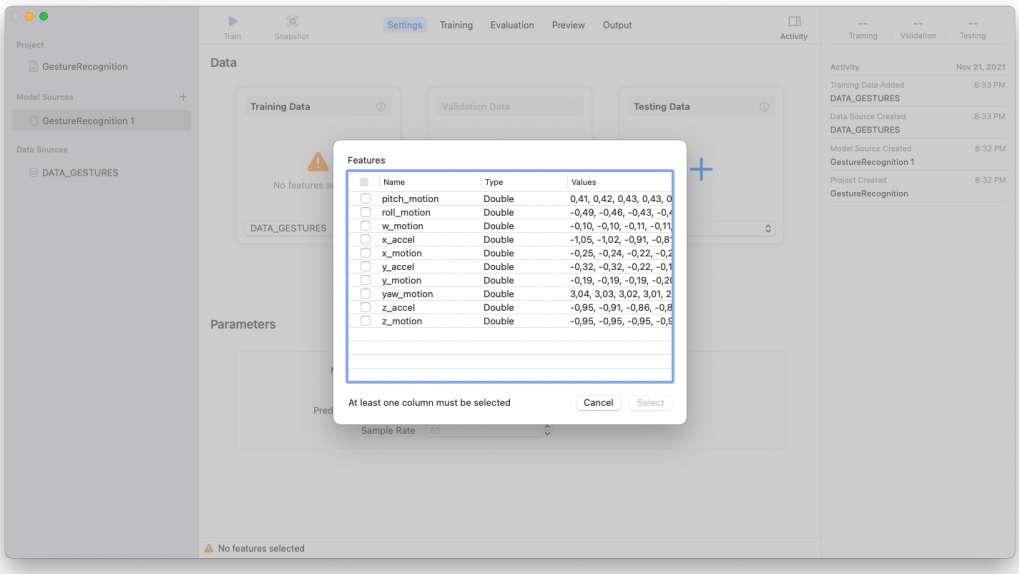
Já na tela inicial, escolhem-se os dados que serão utilizados no projeto, conforme a Figura 64.

Figura 64: Campo para seleção dos dados Create ML.



Após a seleção dos dados, ao se tratar de um treinamento de modelo através de aprendizado supervisionado, deve-se escolher as *features*, a figura 65 as *features* selecionadas.

Figura 65: Features detectadas nos dados pelo Create ML.



Após a escolha das *features* um sumário do projeto é apresentado, com a quantidade de classes, neste caso os gestos, a quantidade de itens e as *features* escolhidas, conforme a Figura 66. Ainda precisa-se escolher como será a divisão dos dados para treino e dados para testes, figura 67, ou pode-se atribuir *Automatic* para que o próprio programa escolha estes parâmetros.

Figura 66: Tela do Create ML após seleção das Features.

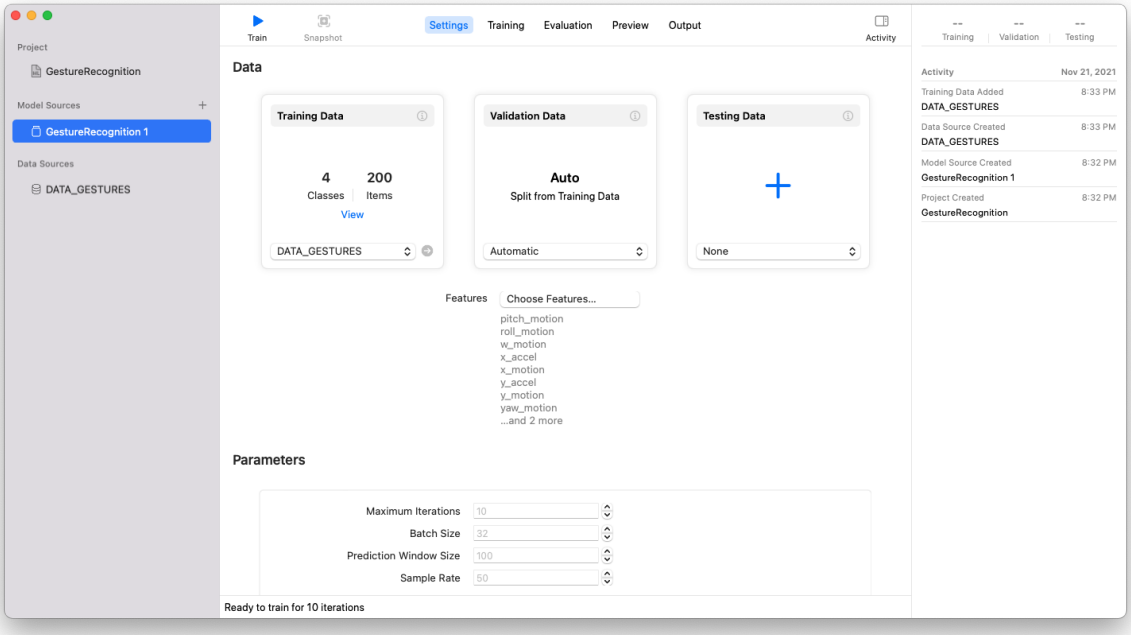
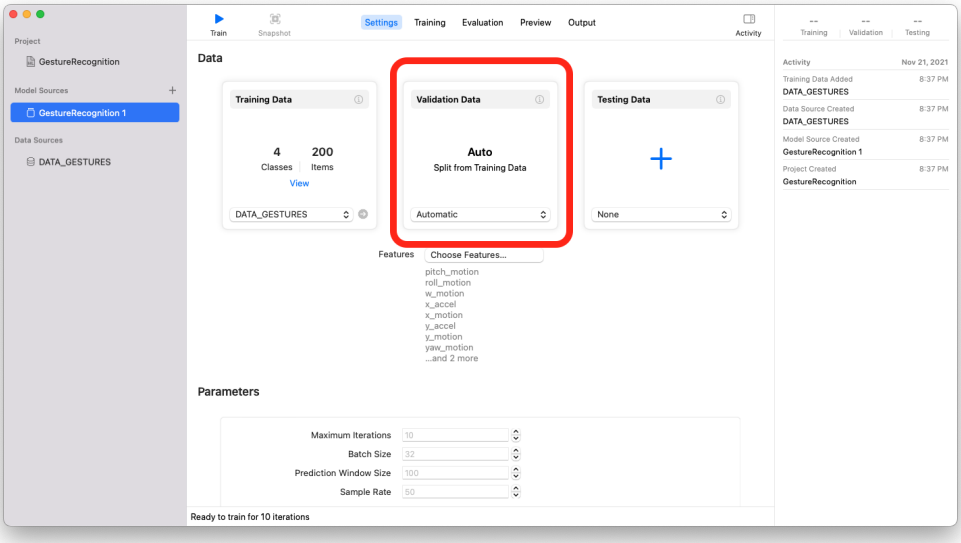


Figura 67: Escolha da divisão entre dados de treino e dados de validação.



Para o início do treino, deve-se selecionar a opção *train* e inicia-se. Neste caso, o primeiro treino foi realizado com 10 iterações, resultando em acurácia de 78,2% no treino 74,1% nas validações, conforme a Figura 68.

Para melhorar este resultado pode-se utilizar de mais uma etapa de treinamento através da opção *Train More*, e nesta segunda etapa utilizou-se 30 iterações (escolhido

empiricamente, com acréscimo aos 10 do primeiro treino e mantendo abaixo dos 50 que é o total de objetos (dados) por *feature*. Na Figura 69 pode-se ver a seleção da quantidade de iterações após rodar o treino novamente através do *Train More*. para que o próprio programa escolha estes parâmetros.

Figura 68: Resultados do primeiro treino com 40 iterações.

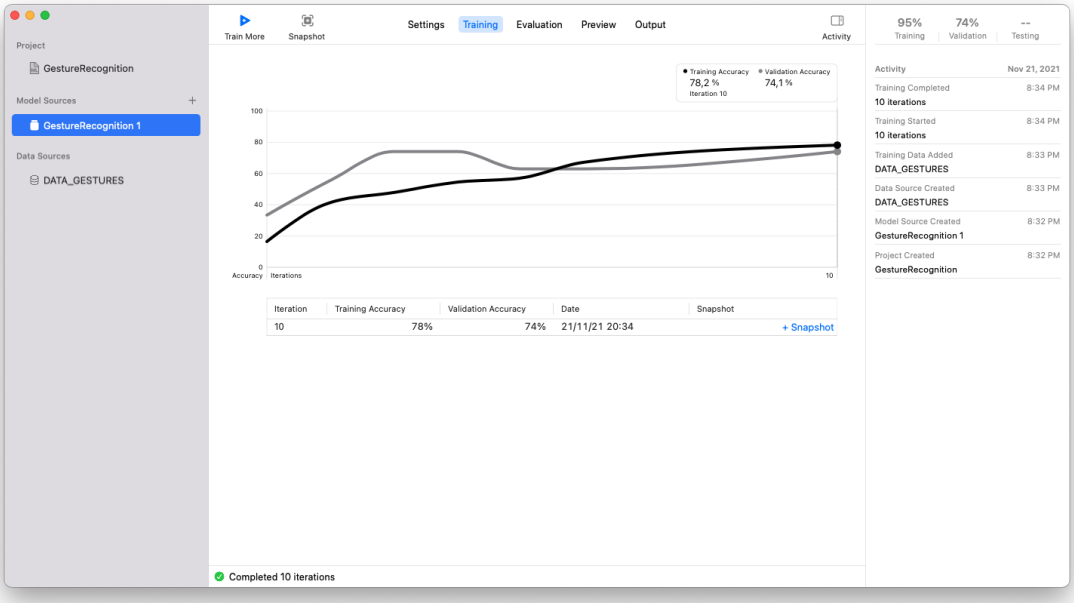
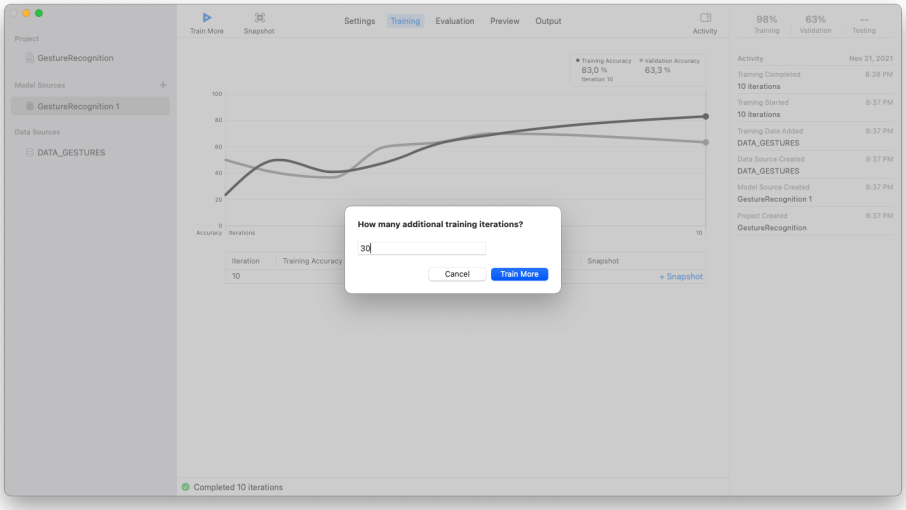
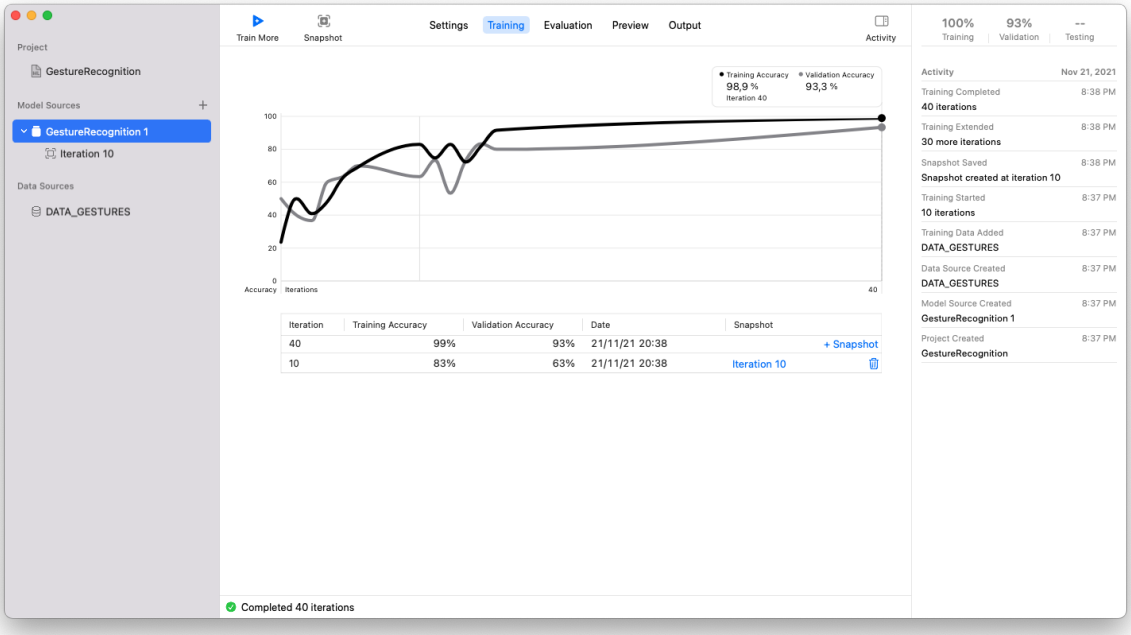


Figura 69: Preparação para segundo treino com 30 iterações.



Após o segundo treino foi realizado com mais 30 iterações, resultado em 40 iterações obteve-se uma acurácia de 98,9% nos treinos e 93,3% nas validações, conforme a Figura 70.

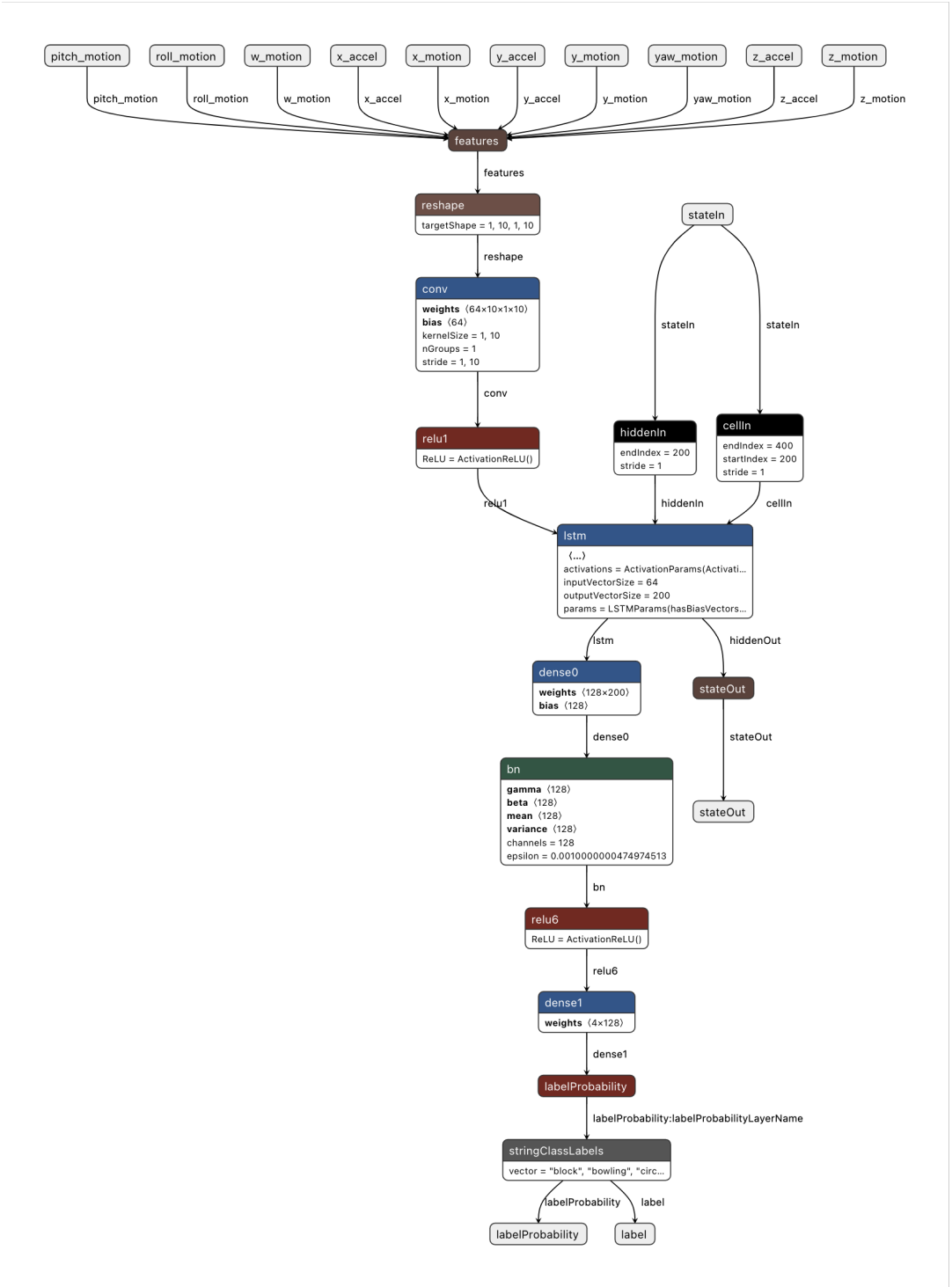
Figura 70: Resultado do segundo treino com 30 iterações.



Este valor se mostrou suficiente para nosso caso de estudo, mas seria possível efetuar mais etapas de treino alterando-se os parâmetros desejados. Com a utilização do *software* Netron[2] pode-se visualizar a arquitetura da rede neural gerada pelo Create ML na Figura 71.

[2]https://github.com/lutzroeder/netron

Figura 71: Arquitetura da Rede Neural gerada pelo Create ML.



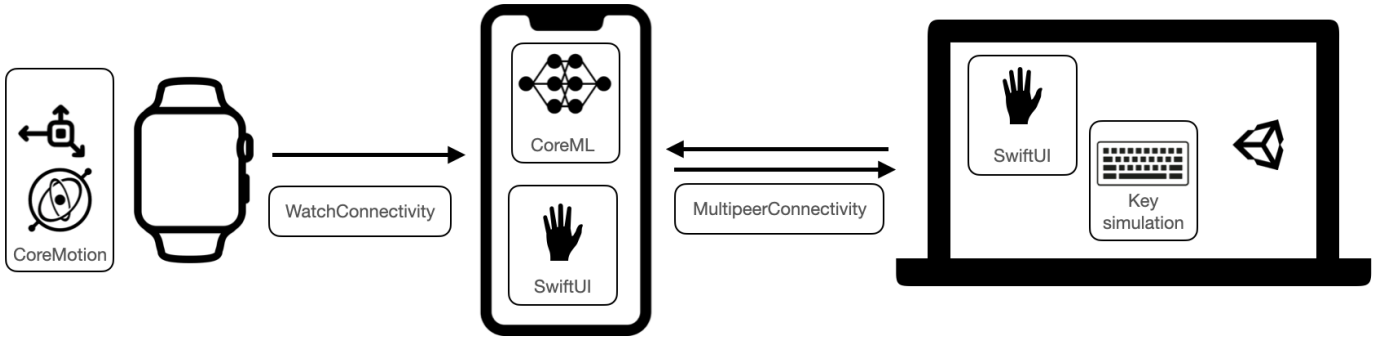
3.10 Detecção dos gestos

Após a obtenção dos dados e treinamento do modelo (rede neural), embarcou-se o modelo no nosso sistema através do aplicativo iOS. O processo de utilização de um modelo treinado é através do *Core ML* mencionado na seção 3.6.7.

Diferente do sistema para aquisição de dados, o sistema para reconhecimento de gestos possui além do aplicativo *watchOS* e *iOS* também possui um aplicativo *macOS*. O aplicativo *macOS* é responsável por controlar o jogo, simulando teclas apertadas.

A arquitetura simplificada do sistema pode ser observada na figura 72.

Figura 72: Arquitetura do Projeto de Reconhecimento de Gestos.



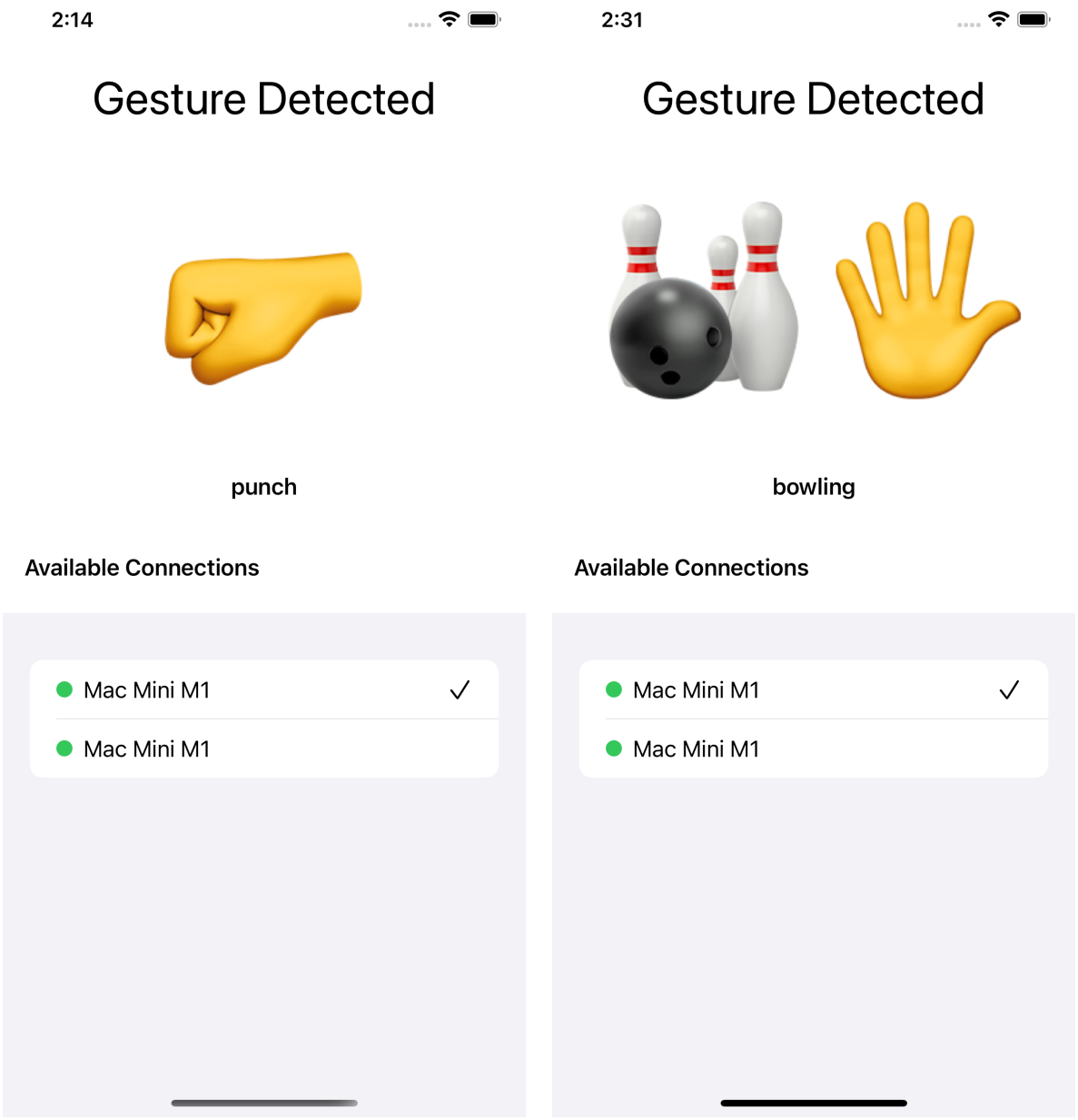
Listing 3: Objeto CMQuaternion

```
//var sensorsData = [[Double]] *valor obtido do Apple Watch*
var initialMemory: [Double] {
    return Array<Double>.init(repeating: 0, count: 400)
}

let predictor = try? GestureMestrado(configuration: .init())

let prediction = try? predictor?.prediction(sensorsData)

print("Label: \(prediction.label)")
```

3.11 Resultados Obtidos

O aplicativo para iOS denominado *Gateway*, recebe os dados do relógio, processa e faz a avaliação (predição) utilizando o modelo treinado. Em seguida, envia estes dados

instantaneamente para o computador conectado, além de apresentar um feedback visual conforme visto na figura 74.



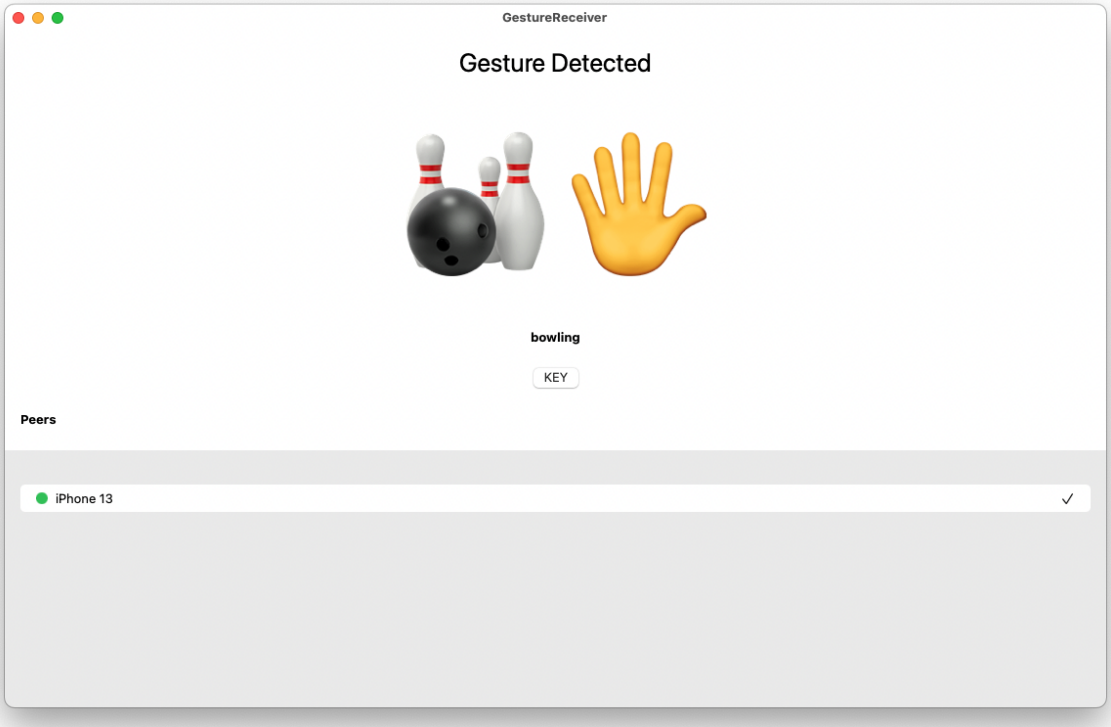
(a) Detecção do gesto (*punch*) **(b) Detecção do gesto (*bowling*)**

Figura 73: Captura de tela do aplicativo iOS do sistema

O aplicativo macOS ao receber efetua a simulação de uma tecla (ou múltiplas) utilizando o objeto `CGEvent`[3]. As teclas são definidas através de seu código. O Apêndice B traz todos os códigos disponíveis extraídos de uma biblioteca do sistema.

[3]https://developer.apple.com/documentation/coregraphics/cgevent

Figura 74: Captura de tela do aplicativo macOS (desktop) do sistema.



As teclas virtualmente apertadas realizam as ações nos jogos ou game engines desejados.

3.12 Discussão

Deste trabalho resultou um aplicativo capaz de reconhecer gestos através dos sensores de um *wearable*, especificamente o Apple Watch. O sistema é composto por dois sistemas menores, o de aquisição de dados e o de processamento e reconhecimento de padrões. Ambos envolvem aplicativos em múltiplas plataformas, *wearable*, *mobile* e *desktop*.

Observou-se que o aplicativo é viável para ser executado mesmo em dispositivos que não estejam em sua versão de *hardware*, porém de *software*.

O aplicativo apresenta algumas vantagens em relação a outros relacionados na literatura ou em soluções de mercado no quesito de ser aberta e totalmente customizável. No âmbito deste trabalho tem-se os gestos como meios recreativos de expressão e criatividade, todavia não deixa-se de mencionar os gestos como extensão da linguagem, ou mesmo como primeira linguagem, com isto o mesmo aplicativo poderia ser aplicado, com as devidas

adaptações, em qualquer área do conhecimento que se beneficie de gestos e linguagem.

Embora o aplicativo em si tenha sido concluído, não foi possível responder à extensão desejada sobre a usabilidade do mesmo. A pandemia de COVID-19 fechou todos os ambientes e restringiu o compartilhamento de objetos, em pró de um bem maior - a vida, não acredito no pesar desta situação e sim na ideia de uma possível aplicação futura da metodologia aqui proposta poderia preencher esta lacuna.

A observação dos resultados aqui obtidos permitem indicar que este trabalho fez surgir a necessidade de realização de novos testes que permitam, entre outros objetivos, implementar o método de avaliação aqui descrito. O desenvolvimento de novas versões deste aplicativo pode trazer-lhe mais funcionalidades, em especial as que possam estender suas capacidades e melhorar sua acurácia.

Para fins de projetos futuros, o aplicativo desenvolvido foi registrado no INPI com os seguintes dados.

> Patente: Programa de Computador. Número do registro: BR5120210027-1, data de registro: 17/11/2021, título: "Aplicativo para reconhecimento de gestos através dos sensores acelerômetro e giroscópio do Apple Watch", Instituição de registro: Instituto Nacional da Propriedade Industrial (INPI) - Instituto Nacional da Propriedade Industrial.

O certificado de registro encontra-se no Apêndice A.

4 CONCLUSÕES

4.1 Conclusão

Esta pesquisa tinha como objetivo desenvolver um estudo sobre o uso de *wearables* em jogos, em particular detectar gestos e utilizar como dispositivos de entrada em jogos. No decorrer do trabalho foi contextualizado os gestos e as relações com a tecnologia, a evolução dos dispositivos *wearables*.

Foi explorada a possibilidade de utilização de uma interface não-convencional no controle de jogos. Com este projeto verificou-se, com necessidade de mais testes, que as interfaces digitais de *wearables* podem sim, ser utilizados como ferramenta complementar ou total de jogos, contextualizado pela indústria de jogos que já trabalha à muitos anos explora o reconhecimento de gestos, voz e padrões.

E por fim, o sistema de controle baseado em *wearables* deste projeto, mostrou a aplicabilidade dessas tecnologias em jogos digitais.

4.1.1 Trabalhos futuros

Como mencionado na discussão, os trabalhos futuros em um primeiro momento envolvem a aplicação do sistema atual com um público diverso de modo a enriquecer os dados e obter uma melhor acurácia. Após este primeiro momento expandir suas capacidades para incluir a opção de treinamento da rede neural localmente (sem a necessidade do treinamento anterior), pois o treinamento *no dispositivo* é um dos novos recursos do *iOS*.

Dentro deste contexto, o trabalho constante com as ferramentas me gerou experiência e confiança para futuramente lançar algum tipo de publicação técnico-científica de cunho prático nestas ferramentas (*ebook*, artigo, etc).

Em um futuro um pouco mais distante seria analisar a viabilidade de transpor ou integrar o sistema para uma plataforma expandida, (abrangendo outros sistemas operacionais *mobile*, outros modelos de *wearables* e sistemas operacionais *desktop*.

REFERÊNCIAS

APPLE. **Apple Developer Documentation — developer.apple.com**. [S.l.: s.n.], 2021. `https://developer.apple.com/documentation/createml`. [Accessed 10-Set-2021].

_____. **CMMotionManager - Apple Developer Documentation**. [S.l.: s.n.], 2021. `https://developer.apple.com/documentation/coremotion/cmmotionmanager`. [Accessed 10-Set-2021].

ASHBROOK, D. L. **Enabling mobile microinteractions**. [S.l.]: Georgia Institute of Technology, 2010.

BONGERS, B. Physical interfaces in the electronic arts. **Trends in gestural control of music**, IRCAM-Centre Pompidou, p. 41–70, 2000.

BRAGA. **Redes neurais artificiais teoria e aplicações**. Rio de Janeiro: Livros Tećnicos e Cientif icos, 2000. v. 2. https://books.google.com.br/books?id=cUgEaAEACAAJ. ISBN 9788521612186.

BRAGA, A.; LEMES, D. d. O. L. Ergonomia e usabilidade em advergames. In: SBGAMES - Simpósio Brasileiro de Jogos para Computador e Entretenimento Digital. São Paulo, Brasil: [s.n.], nov. 2005.

BRUXEL, Y. Sistema para análise de impacto na marcha humana, 2010.

CHAN, A. T.; LEONG, H. V.; KONG, S. H. Real-time tracking of hand gestures for interactive game design. In: 2009 IEEE International Symposium on Industrial Electronics. [S.l.]: IEEE, jul. 2009. p. 98–103.

DALMAZZO, D.; WADDELL, G.; RAMÍREZ, R. Applying Deep Learning Techniques to Estimate Patterns of Musical Gesture. **Frontiers in Psychology**, Frontiers Media SA, v. 11, p. 3546, jan. 2021. ISSN 1664-1078.

DARWIN, C.; BONNER, J. T.; MAY, R. M. **The Descent of Man, and Selection in Relation to Sex**. REV - Revised. [S.l.]: Princeton University Press, 1981. ISBN 9780691082783.

DEEP, G. et al. Future Trends in Wireless World. **Rozy Computech Services, Kurukshetra, Haryana, Department of Computer Science, University College, Kurukshetra University, Lingaya's Institute of Management & Technology, Faridabad, Haryana**, 2010.

DORNAS, A.; ADVERSE, A.; GOUVEIA, M. A Prática das Correspondências no Design: Objetos do Corpo, Corpobjeto. **Estudos em Design**, v. 28, n. 2, 2020.

F. SANTOS, E. **WatchShaker**. [S.l.: s.n.], ago. 2021. DOI: 10.5281/zenodo.5224580. Disponível em: <https://github.com/ezefranca/WatchShaker>.

FITZ-WALTER, Z.; JONES, S.; TJONDRONEGORO, D. Detecting gesture force peaks for intuitive interaction. In: PROCEEDINGS of the 5th Australasian Conference on Interactive Entertainment. [S.l.: s.n.], 2008. p. 1–8.

FLEURY, A.; NAKANO, D. N.; CORDEIRO, J. H. D. **Mapeamento da indústria brasileira e global de jogos digitais**. [S.l.]: GEDIGames: NPGT, 2014.

FRANQUEIRA, T. C. **UM ESTUDO SOBRE QUATÉRNIOS E SUA APLICAÇÃO EM ROBÓTICA**. 1993. Diss. (Mestrado) – Curso de Pós- graduação em Engenharia Elétrica, Universidade Federal de Minas Gerais, Belo Horizonte. Disponível em: <http://www.ppgee.ufmg.br/defesas/742M.PD1>.

GIBSON, J. J. The theory of affordances. **Hilldale, USA**, v. 1, n. 2, p. 67–82, 1977.

GLOBAL Smartwatch Shipments Jump 35% YoY in Q1 2021. [S.l.: s.n.], jun. 2021. https://www.counterpointresearch.com/global-smartwatch-shipments-q1-2021/. (Acessado em 01/05/2021).

GRUMMITT, C. **IOS development with Swift**. Shelter Island, NY: Manning Publications, 2018. ISBN 9781617294075.

GUSGÅRD, O. Application development for the Apple Watch. Metropolia Ammattikorkeakoulu, 2018.

HALL, J. A.; KNAPP, M. Comunicação não-verbal na interação humana. **São Paulo: SJN Editora**, 1999.

HAMILTON, W. Quaternions. **Proceedings of the Royal Irish Academy**, v. 50, p. 89–92, jan. 1847.

HAYKIN, S. S. et al. **Neural networks and learning machines**. [S.l.]: Pearson Upper Saddle River, NJ, USA: 2009. v. 3.

HEIMONEN, T. et al. Designing Gesture-Based Control for Factory Automation. In: p. 202–209. ISBN 9783642404795. DOI: 10.1007/978-3-642-40480-1_13.

HERTZ, J. A.; KROGH, A. S.; PALMER, R. G. **Introduction to the Theory of Neural Computation**. [S.l.]: Citeseer.

HOCHREITER, S.; SCHMIDHUBER, J. Long Short-Term Memory. **Neural Computation**, v. 9, n. 8, p. 1735–1780, nov. 1997. ISSN 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://direct.mit.edu/neco/article-pdf/9/8/1735/813796/neco.1997.9.8.1735.pdf. Disponível em: <https://doi.org/10.1162/neco.1997.9.8.1735>.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. **Proceedings of the National Academy of Sciences of the United States of America**, v. 79, n. 8, p. 2554–2558, abr. 1982. ISSN 0027-8424. Disponível em: <http://view.ncbi.nlm.nih.gov/pubmed/6953413]>.

IGNATOV, A. Real-time human activity recognition from accelerometer data using Convolutional Neural Networks. **Applied Soft Computing**, Elsevier BV, v. 62, p. 915–922, jan. 2018. ISSN 1568-4946.

JEONG, S. C. et al. Domain-specific innovativeness and new product adoption: A case of wearable devices. **Telematics and Informatics**, Elsevier BV, USA, v. 34, n. 5, p. 399–412, ago. 2017. ISSN 0736-5853.

JOSGRILBERG, R. O corpo e seus desdobramentos interativos: os jogos de si mesmo como rejogo com os outros. **Revista Internartional Studies on law and education**, v. 23, p. 13–24, 2016.

JOUSSE, M. L'anthropologie du geste. Paris. **France: Les Éditions Resma**, 1969.

KAO, G. et al. Innovating an Industry. In: TURNING Silicon into Gold. [S.l.]: Apress, 2020. p. 55–59.

KHAN, A. M. et al. A Triaxial Accelerometer-Based Physical-Activity Recognition via Augmented-Signal Features and a Hierarchical Recognizer. **IEEE Transactions on Information Technology in Biomedicine**, Institute of Electrical e Electronics Engineers (IEEE), v. 14, n. 5, p. 1166–1172, set. 2010.

KHAN, A. M. **Human Activity Recognition Using A Single Tri-axial Accelerometer**. 2011. Tese (Doutorado) – Kyung Hee University, Department of Computer Engineering. Seoul. An optional note.

KORTUM, P. **HCI beyond the GUI: Design for haptic, speech, olfactory, and other nontraditional interfaces**. [S.l.]: Elsevier, 2008.

KUIPERS, J. B. **Quaternions and Rotation Sequences: A Primer With Applications to Orbits, Aerospace and Virtual Reality**. [S.l.]: Princeton University Press, 1999. 5.2 Quaternions Defined, p. 104–105.

LECUN, Y. et al. Object Recognition with Gradient-Based Learning. In: SHAPE, Contour and Grouping in Computer Vision. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999. p. 319–345. 12 de maio de 2018. ISBN 978-3-540-46805-9. DOI: 10.1007/3-540-46805-6_19. Disponível em: <https://doi.org/10.1007/3-540-46805-6_19>.

LEE, S.-M.; YOON, S. M.; CHO, H. Human activity recognition from accelerometer data using Convolutional Neural Network. In: 2017 IEEE International Conference on Big Data and Smart Computing (BigComp). [S.l.]: IEEE, fev. 2017. p. 131–134.

LUCIA Santaella - Enciclopédia Itaú Cultural. [S.l.: s.n.], mai. 2021. https://enciclopedia.itaucultural.org.br/pessoa3192/lucia-santaella. (Accessed on 01/05/2021).

MACKENZIE, I. S.; TEATHER, R. J. FittsTilt: The Application of Fitts' Law to Tilt-Based Interaction. In: PROCEEDINGS of the 7th Nordic Conference on Human-Computer Interaction: Making Sense Through Design. Copenhagen, Denmark: Association for Computing Machinery, 2012. (NordiCHI '12), p. 568–577. ISBN 9781450314824. DOI: 10.1145/2399016.2399103. Disponível em: <https://doi.org/10.1145/2399016.2399103>.

MARK, D. et al. Whee! Gyro and Accelerometer! In: BEGINNING iPhone Development. [S.l.]: Springer, 2014. p. 699–726.

MEDRYK, S.; MACKENZIE, I. S. A comparison of accelerometer and touch-based input for mobile gaming. In: INTERNATIONAL Conference on Multimedia and Human-Computer Interaction-MHCI 2013. [S.l.: s.n.], 2013. p. 117–1.

MEFTAH, L.; ROUVOY, R.; CHRISMENT, I. Testing Nearby Peer-to-Peer Mobile Apps at Large. In: 2019 IEEE/ACM 6th International Conference on Mobile Software Engineering and Systems (MOBILESoft). [S.l.: s.n.], 2019. p. 1–11. DOI: 10.1109/MOBILESoft.2019.00009.

NAKATSU, R. Handbook of digital games and entertainment technologies. In: **Handbook of Digital Games and Entertainment Technologies**. Edição: Ryohei Nakatsu, Matthias Rauterberg e Paolo Ciancarini. Singapore: Springer, 2016. p. 293–312. ISBN 9789814560504.

OLAH, C. **Understanding LSTM Networks**. [S.l.: s.n.], 2015. 12 de maio de 2018. Disponível em: <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>.

OMETOV, A. et al. A Survey on Wearable Technology: History, State-of-the-Art and Current Challenges. **Computer Networks**, v. 193, p. 108074, 2021. ISSN 1389-1286.

PFUTZENREUTER, E. P. Magia e imaginário nos controles baseadaos em gestos, 2009.

POSLAD, S. **Ubiquitous computing: smart devices, environments and interactions**. [S.l.]: John Wiley & Sons, 2011.

PUJOL, J. Hamilton, Rodrigues, Gauss, Quaternions, and Rotations: a Historical Reassessment. **Commun. Math. Anal.**, Mathematical Research Publishers, v. 13, n. 2, p. 1–14, 2012. Disponível em: <https://projecteuclid.org:443/euclid.cma/1349803591>.

RAMBO, G. **MultipeerKit**. [S.l.: s.n.], out. 2021. DOI: 10.5281/zenodo.1234. Disponível em: <https://github.com/insidegui/MultipeerKit>.

ROBOTICS Kinematics and Dynamics. [S.l.: s.n.], 2018. Description of Position and Orientation. Disponível em: <https://en.wikibooks.org/wiki/Robotics_Kinematics_and_Dynamics/Description_of_Position_and_Orientation>. Acesso em: 31 jan. 2020.

ROMERO, E. L. G. A gestualidade das mãos: o gesto técnico e o gesto poético. **dObra[s] – revista da Associação Brasileira de Estudos de Pesquisas em Moda**, Dobras, v. 3, n. 7, p. 89, fev. 2009.

ROSÁRIO SILVA LIMA, E. do; CRUZ-SANTOS, A. Aquisição dos gestos na comunicação pré-linguística: uma abordagem teórica. **Revista da Sociedade Brasileira de Fonoaudiologia**, FapUNIFESP (SciELO), v. 17, n. 4, p. 495–501, dez. 2012.

ROSENBLATT, F. Principles of neurodynamics. Spartan Book, 1962.

SAHIN, Ö. Introduction to Apple ML Tools. In: DEVELOP Intelligent iOS Apps with Swift. [S.l.]: Springer, 2021. p. 17–39.

SANTAELLA, L. **Culturas e artes do pós-humano: da cultura das mídias à cibercultura**. [S.l.]: Paulus, 2003. (Comunicação (Paulus (Firm : Brazil)))). ISBN 9788534921015.

SANTAELLA, L. O homem e as máquinas. **A arte no século XXI: A humanização das tecnologias**, p. 37–59, 1997.

SANTAELLA, L. Games e comunidades virtuais, 2004.

SHIRATORI, T.; HODGINS, J. K. Accelerometer-based user interfaces for the control of a physically simulated character. **ACM Transactions on Graphics**, Association for Computing Machinery (ACM), New York, NY, USA, v. 27, n. 5, p. 1–9, dez. 2008. ISSN 0730-0301.

SILVA, F. G. d. **Reconhecimento de movimentos humanos utilizando um acelerômetro e inteligência computacional**. 2015. Tese (Doutorado).

SMARTWATCHES. **SmartWatches.org - Resources and Reviews on Smartwatches**. [S.l.: s.n.], abr. 2021. https://smartwatches.org/. (Accessed on 04/10/2021).

STANLEY, M.; LEE, J. **Sensor Analysis for the Internet of Things**: Synthesis Lectures of Algorithm and Software in Engineering. [S.l.]: Morgan & Claypool Publishers, 2018. 3.3 Orientation Representations, p. 43.

THAKKAR, M. **Beginning Machine Learning in iOS**. [S.l.]: Apress, 2019. DOI: 10.1007/978-1-4842-4297-1. Disponível em: <https://doi.org/10.1007/978-1-4842-4297-1>.

TURNER, P. Towards an account of intuitiveness. **Behaviour & Information Technology**, Taylor & Francis, v. 27, n. 6, p. 475–482, 2008.

VAISHNAVI, V. K.; KUECHLER, W. **Design science research methods and patterns: innovating information and communication technology**. [S.l.]: Crc Press, 2015.

VARMA, J. What Is SwiftUI. In: SWIFTUI for Absolute Beginners. [S.l.]: Springer, 2019. p. 1–8.

VASCONCELLOS, N. S. d. F. e. **O Point-And-Click adventure ressurge: uma discussão analítica sobre um gênero adormecido**. 2020. Diss. (Mestrado) – Programa de Estudos Pós-Graduados em Desenvolvimento de Jogos Digitais, Pontifícia Universidade Católica de São Paulo, São Paulo.

VYGOTSKY, L. S. **Formação social da mente**. [S.l.: s.n.], mai. 1994. ISBN 9788533603349.

WAGNER, D. et al. Activity recognition using inertial sensors and a 2-D convolutional neural network. In: 2017 10th International Workshop on Multidimensional (nD) Systems (nDS). [S.l.]: IEEE, set. 2017. p. 1–6.

WAREABLE. **Wareable - Wearable technology reviews, news and features**. [S.l.: s.n.], abr. 2021. https://www.wareable.com/. (Accessed on 04/10/2021).

WIKIPEDIA. **Wearable technology - Wikipedia**. [S.l.: s.n.], abr. 2021. https://en.wikipedia.org/wiki/Wearable\%5Ftechnology. (Accessed on 06/10/2021).

ZENG, M. et al. Convolutional Neural Networks for Human Activity Recognition using Mobile Sensors. In: PROCEEDINGS of the 6th International Conference on Mobile Computing, Applications and Services. [S.l.]: ICST, 2014. p. 197–205.

ZENG, Z.; GONG, Q.; ZHANG, J. CNN Model Design of Gesture Recognition Based on Tensorflow Framework. In: 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). [S.l.]: IEEE, mar. 2019. p. 1062–1067.

ZHENG, Y. et al. Time Series Classification Using Multi-Channels Deep Convolutional Neural Networks. In: LI, F. et al. (Ed.). **Web-Age Information Management**. Cham: Springer International Publishing, 2014. p. 298–310. ISBN 978-3-319-08010-9.

APÊNDICE A – CERTIFICADO DE REGISTRO DE SOFTWARE NO INPI

REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512021002711-6**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 16/11/2021, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Aplicativo para reconhecimento de gestos através dos sensores acelerômetro e giroscópio do Apple Watch

Data de publicação: 16/11/2021

Data de criação: 01/10/2021

Titular(es): EZEQUIEL FRANÇA DOS SANTOS

Autor(es): EZEQUIEL FRANÇA DOS SANTOS

Linguagem: SWIFT

Campo de aplicação: IF-10

Tipo de programa: AP-01; SO-04; TC-03

Algoritmo hash: SHA-512

Resumo digital hash:
2fda595320f6735903e857d3d10de1c6c8969d6eddff11fdf580494f2b7245fa70dc39726abc81a39afae9e175a298a3bb1a1f6dbacde6e36b7edf927b1c2aa7

Expedido em: 23/11/2021

Aprovado por:
Carlos Alexandre Fernandes Silva
Chefe da DIPTO

APÊNDICE B – CÓDIGO PARCIAL DO HITOOLBOX/EVENTS.H COM O CÓDIGO DAS TECLAS VIRTUAIS

```
1 /Library/Developer/CommandLineTools/SDKs/MacOSX.sdk/System/Library/
    Frameworks/Carbon.framework/Versions/A/Frameworks/HIToolbox.framework/
    Versions/A/Headers/Events.h
```

```
1 /*
2      File:       HIToolbox/Events.h
3
4      Contains:   Event Manager Interfaces.
5
6      Copyright:  1985-2008 by Apple Computer, Inc., all rights
           reserved
7
8      Bugs?:      For bug reports, consult the following page on
9                  the World Wide Web:
10
11                     http://developer.apple.com/bugreporter/
12
13 */
14
15 /*
16  - Summary:
17  - Virtual keycodes
18
19  - Discussion:
20  These constants are the virtual keycodes defined originally in
        Inside Mac Volume V, pg. V-191. They identify physical
       keys on a keyboard. Those constants with "ANSI" in the name
        are labeled according to the key position on an ANSI-
       standard US keyboard. For example, kVK_ANSI_A indicates the
        virtual keycode for the key with the letter 'A' in the US
       keyboard layout. Other keyboard layouts may have the 'A'
```

```
      key label on a different physical key; in this case,
      pressing 'A' will generate a different virtual keycode.
21  */
22 enum {
23   kVK_ANSI_A                    = 0x00,
24   kVK_ANSI_S                    = 0x01,
25   kVK_ANSI_D                    = 0x02,
26   kVK_ANSI_F                    = 0x03,
27   kVK_ANSI_H                    = 0x04,
28   kVK_ANSI_G                    = 0x05,
29   kVK_ANSI_Z                    = 0x06,
30   kVK_ANSI_X                    = 0x07,
31   kVK_ANSI_C                    = 0x08,
32   kVK_ANSI_V                    = 0x09,
33   kVK_ANSI_B                    = 0x0B,
34   kVK_ANSI_Q                    = 0x0C,
35   kVK_ANSI_W                    = 0x0D,
36   kVK_ANSI_E                    = 0x0E,
37   kVK_ANSI_R                    = 0x0F,
38   kVK_ANSI_Y                    = 0x10,
39   kVK_ANSI_T                    = 0x11,
40   kVK_ANSI_1                    = 0x12,
41   kVK_ANSI_2                    = 0x13,
42   kVK_ANSI_3                    = 0x14,
43   kVK_ANSI_4                    = 0x15,
44   kVK_ANSI_6                    = 0x16,
45   kVK_ANSI_5                    = 0x17,
46   kVK_ANSI_Equal                = 0x18,
47   kVK_ANSI_9                    = 0x19,
48   kVK_ANSI_7                    = 0x1A,
49   kVK_ANSI_Minus                = 0x1B,
50   kVK_ANSI_8                    = 0x1C,
51   kVK_ANSI_0                    = 0x1D,
52   kVK_ANSI_RightBracket         = 0x1E,
53   kVK_ANSI_O                    = 0x1F,
54   kVK_ANSI_U                    = 0x20,
55   kVK_ANSI_LeftBracket          = 0x21,
56   kVK_ANSI_I                    = 0x22,
57   kVK_ANSI_P                    = 0x23,
```

```
58    kVK_ANSI_L                    = 0x25,
59    kVK_ANSI_J                    = 0x26,
60    kVK_ANSI_Quote                = 0x27,
61    kVK_ANSI_K                    = 0x28,
62    kVK_ANSI_Semicolon            = 0x29,
63    kVK_ANSI_Backslash            = 0x2A,
64    kVK_ANSI_Comma                = 0x2B,
65    kVK_ANSI_Slash                = 0x2C,
66    kVK_ANSI_N                    = 0x2D,
67    kVK_ANSI_M                    = 0x2E,
68    kVK_ANSI_Period               = 0x2F,
69    kVK_ANSI_Grave                = 0x32,
70    kVK_ANSI_KeypadDecimal        = 0x41,
71    kVK_ANSI_KeypadMultiply       = 0x43,
72    kVK_ANSI_KeypadPlus           = 0x45,
73    kVK_ANSI_KeypadClear          = 0x47,
74    kVK_ANSI_KeypadDivide         = 0x4B,
75    kVK_ANSI_KeypadEnter          = 0x4C,
76    kVK_ANSI_KeypadMinus          = 0x4E,
77    kVK_ANSI_KeypadEquals         = 0x51,
78    kVK_ANSI_Keypad0              = 0x52,
79    kVK_ANSI_Keypad1              = 0x53,
80    kVK_ANSI_Keypad2              = 0x54,
81    kVK_ANSI_Keypad3              = 0x55,
82    kVK_ANSI_Keypad4              = 0x56,
83    kVK_ANSI_Keypad5              = 0x57,
84    kVK_ANSI_Keypad6              = 0x58,
85    kVK_ANSI_Keypad7              = 0x59,
86    kVK_ANSI_Keypad8              = 0x5B,
87    kVK_ANSI_Keypad9              = 0x5C
88 };
89
90 /* keycodes for keys that are independent of keyboard layout*/
91 enum {
92    kVK_Return                    = 0x24,
93    kVK_Tab                       = 0x30,
94    kVK_Space                     = 0x31,
95    kVK_Delete                    = 0x33,
96    kVK_Escape                    = 0x35,
```

```
97   kVK_Command                 = 0x37,
98   kVK_Shift                   = 0x38,
99   kVK_CapsLock                = 0x39,
100  kVK_Option                  = 0x3A,
101  kVK_Control                 = 0x3B,
102  kVK_RightCommand            = 0x36,
103  kVK_RightShift              = 0x3C,
104  kVK_RightOption             = 0x3D,
105  kVK_RightControl            = 0x3E,
106  kVK_Function                = 0x3F,
107  kVK_F17                     = 0x40,
108  kVK_VolumeUp                = 0x48,
109  kVK_VolumeDown              = 0x49,
110  kVK_Mute                    = 0x4A,
111  kVK_F18                     = 0x4F,
112  kVK_F19                     = 0x50,
113  kVK_F20                     = 0x5A,
114  kVK_F5                      = 0x60,
115  kVK_F6                      = 0x61,
116  kVK_F7                      = 0x62,
117  kVK_F3                      = 0x63,
118  kVK_F8                      = 0x64,
119  kVK_F9                      = 0x65,
120  kVK_F11                     = 0x67,
121  kVK_F13                     = 0x69,
122  kVK_F16                     = 0x6A,
123  kVK_F14                     = 0x6B,
124  kVK_F10                     = 0x6D,
125  kVK_F12                     = 0x6F,
126  kVK_F15                     = 0x71,
127  kVK_Help                    = 0x72,
128  kVK_Home                    = 0x73,
129  kVK_PageUp                  = 0x74,
130  kVK_ForwardDelete           = 0x75,
131  kVK_F4                      = 0x76,
132  kVK_End                     = 0x77,
133  kVK_F2                      = 0x78,
134  kVK_PageDown                = 0x79,
135  kVK_F1                      = 0x7A,
```

```
136   kVK_LeftArrow                 = 0x7B,
137   kVK_RightArrow                = 0x7C,
138   kVK_DownArrow                 = 0x7D,
139   kVK_UpArrow                   = 0x7E
140 };
141
142 /* ISO keyboards only*/
143 enum {
144   kVK_ISO_Section               = 0x0A
145 };
146
147 /* JIS keyboards only*/
148 enum {
149   kVK_JIS_Yen                   = 0x5D,
150   kVK_JIS_Underscore            = 0x5E,
151   kVK_JIS_KeypadComma           = 0x5F,
152   kVK_JIS_Eisu                  = 0x66,
153   kVK_JIS_Kana                  = 0x68
154 };
```

APÊNDICE C – CÓDIGO ARDUINO PARA GERAÇÃO DO DATASET COM BASE NO MPU6050

```
#include "Simple_MPU6050.h"
//https://github.com/ZHomeSlice/Simple_MPU6050
#define MPU6050_ADDRESS_AD0_LOW     0x68
#define MPU6050_ADDRESS_AD0_HIGH    0x69
#define MPU6050_DEFAULT_ADDRESS     MPU6050_ADDRESS_AD0_LOW
#define OFFSETS 0, 0, 0, 0, 0, 0
ENABLE_MPU_OVERFLOW_PROTECTION();

Simple_MPU6050 mpu;

void setup() {
  uint8_t val;
  Serial.begin(115200);
  while (!Serial);
  Serial.println("Start: ");
  mpu.SetAddress(MPU6050_ADDRESS_AD0_LOW).TestConnection(1);
  mpu.load_DMP_Image(OFFSETS);
  Serial.print("Setup Complete in ");
  mpu.PrintActiveOffsets();
  mpu.CalibrateGyro(6);
  mpu.CalibrateAccel(6);
  mpu.on_FIFO(loop_read);

  Serial.print("full calibration Complete in ");
  Serial.print(millis());
  Serial.println("Miliseconds");
  Serial.println("x_accel, y_accel, z_accel, w_motion, x_motion
     , y_motion, z_motion, pitch_motion, roll_motion,
     yaw_motion");
}
```

```
29 void loop() {
30   mpu.dmp_read_fifo();
31 }
32 int getSensorValues(int16_t *gyro, int16_t *accel, int32_t *
     quartenion, uint16_t SpamDelay = 100) {
33   Quaternion q;
34   VectorFloat gravity;
35   float ypr[3] = { 0, 0, 0 };
36   float xyz[3] = { 0, 0, 0 };
37
38     mpu.GetQuaternion(&q, quartenion);
39     mpu.GetGravity(&gravity, &q);
40     mpu.GetYawPitchRoll(ypr, &q, &gravity);
41     mpu.ConvertToDegrees(ypr, xyz);
42     Serial.print(accel[0], 3);
43     Serial.print(",");
44     Serial.print(accel[1], 3);
45     Serial.print(",");
46     Serial.print(accel[2], 3);
47     Serial.print(",");
48     Serial.print(quartenion[0], 3);
49     Serial.print(",");
50     Serial.print(quartenion[1], 3);
51     Serial.print(",");
52     Serial.print(quartenion[2], 3);
53     Serial.print(",");
54     Serial.print(quartenion[3], 3);
55     Serial.print(",");
56     Serial.print(xyz[1], 3);
57     Serial.print(",");
58     Serial.print(xyz[2], 3);
59     Serial.print(",");
60     Serial.print(xyz[0], 3);
61 }
62 void loop_read(int16_t *gyro, int16_t *accel, int32_t *quat,
     uint32_t *timestamp) {
63   uint8_t Spam_Delay = 100;
64   getSensorValues(gyro, accel, quat, Spam_Delay);
65 }
```