

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
GRADUAÇÃO EM ENGENHARIA MECATRÔNICA

GUSTAVO BRUNO OLIVEIRA VASCONCELOS

Uso de Técnicas de Inteligência Artificial para o Desenvolvimento de Jogos
Sérios Aplicados a Estrutura Robótica para Reabilitação do Punho Humano

Uberlândia – MG

2023

GUSTAVO BRUNO OLIVEIRA VASCONCELOS

Uso de Técnicas de Inteligência Artificial para o Desenvolvimento de Jogos Sérios Aplicados
a Estrutura Robótica para Reabilitação do Punho Humano

Trabalho de Conclusão de Curso apresentado à
Faculdade de Engenharia Mecânica da Universidade
Federal de Uberlândia como requisito parcial para
obtenção do título de Engenheiro Mecatrônico.

Orientador: Prof. Dr. Rogério Sales Gonçalves

Uberlândia – MG

2023

GUSTAVO BRUNO OLIVEIRA VASCONCELOS

Uso de técnicas de inteligência artificial para o desenvolvimento de jogos sérios
aplicados a estrutura robótica para reabilitação do punho humano

Trabalho de Conclusão de Curso apresentado à
Faculdade de Engenharia Mecatrônica da
Universidade Federal de Uberlândia como requisito
parcial para obtenção do título de Engenheiro
Mecatrônico.

Uberlândia, 07 de Dezembro de 2023

Banca Examinadora:

Prof. Dr. Rogério Sales Gonçalves – UFU (Orientador)

Profa. Dra. Guênia Mara Vieira Ladeira

Mestre Gabriella Faina Garcia

AGRADECIMENTOS

Gostaria de expressar minha mais profunda gratidão, em primeiro lugar a minha mãe Nilma Rubia que é a pessoa mais maravilhosa do mundo, que fez tudo isso ser possível e cada palavra que eu poderia escrever aqui seria apenas uma pequena tentativa de capturar a magnitude da sua importância em minha vida. Moldou meu mundo desde nosso primeiro encontro e que meu sonho é conseguir ser uma fração da pessoa que ela é.

Além disso, é com grande apreço que desejo agradecer ao meu orientador, Rogério Sales Gonçalves, pela sua orientação, apoio inestimável e expertise ao longo de todo o projeto. Os valiosos insights e o incentivo constante que ele me proporcionou desempenharam um papel fundamental na moldagem deste trabalho.

Não posso deixar de agradecer aos meus amigos que me ajudaram de alguma forma com o projeto, Adilson Araújo, Alan Nascimento, Gabriella Garcia, João Barcelos, Matheus Freitas e Rafael Passos, por todo apoio e que me estenderam a mão para essa jornada. Deixo um agradecimento especial ao meu melhor amigo que considero um irmão, Geovani Estevan, pelo apoio imensurável em toda essa caminhada e me acolher em sua casa para que eu pudesse ficar em Uberlândia e conseguir terminar o curso.

Agradeço também às minhas madrinhas, pelo incentivo e pelo auxílio financeiro durante essa fase importante da minha vida. Deixo um agradecimento especial a minha namorada, Irene Lustosa, pelo apoio principalmente psicológico nessa reta final. Gratidão também aos meus animais que já se foram e aos que ainda estão comigo pelos momentos de alegria e paz que me trouxeram.

Não posso deixar de expressar minha gratidão aos respeitados membros do corpo docente da Faculdade de Engenharia Mecânica (FEMEC) e da Universidade Federal de Uberlândia (UFU). Sua dedicação e compartilhamento de conhecimentos ao longo da minha jornada acadêmica foram verdadeiramente enriquecedores. Um agradecimento também aos meus professores do ensino fundamental e médio.

Por fim, mas não menos importante gostaria de agradecer ao CNPq pela bolsa dentro do projeto 408013/2022-2. Juntamente a todas as fontes de conhecimento, autores e pesquisadores cujas obras foram consultadas e citadas nesta pesquisa. E a todos que contribuíram para este projeto, o meu mais sincero agradecimento. Vocês foram fundamentais para o meu crescimento acadêmico e profissional.

“O sucesso é a soma de pequenos
esforços repetidos dia após dia.”

Robert Collier

RESUMO

Este Trabalho de Conclusão de Curso (TCC) envolve a temática relacionada ao uso de técnicas de inteligência artificial para o desenvolvimento de jogos sérios aplicados a uma estrutura robótica para reabilitação do punho humano. O objetivo principal é utilizar a inteligência artificial para deixar os pacientes mais engajados, ajustando o nível de dificuldade dos jogos sérios em função das ações do paciente.

Na primeira fase do projeto foi realizado o estudo de um dispositivo de reabilitação para o punho humano, que representa um avanço significativo na integração de componentes eletrônicos e estruturais para promover eficazmente a recuperação física. Ele será utilizado não só como um controle *Joystick* para enviar sinais de comando ao personagem dos jogos, mas também como um dispositivo de auxílio à movimentação do paciente.

Na segunda fase do projeto, a atenção foi direcionada para o desenvolvimento de jogos sérios, uma abordagem que visa combinar entretenimento com objetivo terapêutico. Nesse contexto, uma ênfase especial foi colocada na criação de três jogos distintos na plataforma Unity 3D visando os principais movimentos do punho humano. Cada jogo foi concebido com uma mecânica envolvente, fundamentada na premissa de que o personagem principal deve desviar-se de obstáculos. Esse aspecto não apenas oferece uma experiência lúdica estimulante, mas também incorpora uma abordagem estratégica, promovendo o desenvolvimento cognitivo e motor dos usuários durante o processo de reabilitação.

Na terceira fase do projeto, focou-se na implementação do projeto, foi estabelecida uma comunicação entre o Arduino e o Matlab, consolidando a interação entre os componentes físicos e o sistema de controle. Além disso, a integração entre o Matlab e o ambiente do Unity ocorreu de uma forma que possibilitava a troca bidirecional de informações sobre a movimentação do jogador nos jogos.

Os resultados deixam claro que o projeto foi um sucesso visto que todo planejamento do controle da IA em cada jogo ocorreu como o esperado, assim o jogo se adequa a cada paciente dificultando ou facilitando os jogos, dessa forma deixando a experiência com a reabilitação mais interativa e mantendo a eficácia do tratamento.

Palavras-chave: reabilitação do punho humano, jogos sérios, inteligência artificial, estrutura robótica, protocolo de comunicação.

ABSTRACT

This Bachelor's Thesis (TCC) revolves around the use of artificial intelligence techniques for the development of serious games applied to a robotic structure for wrist rehabilitation. The main objective is to employ artificial intelligence to enhance patient engagement by adjusting the difficulty level of serious games based on the patient's actions.

In the first phase of the project, a study was conducted on a wrist rehabilitation device, representing a significant advancement in the integration of electronic and structural components to effectively promote physical recovery. It serves not only as a joystick control to send command signals to game characters but also as an aid device for patient movement.

The second phase focused on the development of serious games, aiming to combine entertainment with therapeutic objectives. Three distinct games were created on the Unity 3D platform, targeting key movements of the human wrist. Each game was designed with engaging mechanics, based on the premise that the main character must navigate obstacles. This not only provides a stimulating gaming experience but also incorporates a strategic approach, promoting cognitive and motor development during the rehabilitation process.

In the third phase, the project implementation centered on establishing communication between Arduino and Matlab, consolidating interaction between physical components and the control system. Additionally, integration between Matlab and the Unity environment allowed bidirectional exchange of information about player movements in the games.

The results make it clear that the project was a success, as the entire planning of AI control in each game unfolded as expected. Consequently, the game adjusts to each patient, presenting challenges or facilitating interactions tailored to individual needs. This enhances the interactive nature of the rehabilitation experience while maintaining the effectiveness of the treatment.

Keywords: human wrist rehabilitation, serious games, artificial intelligence, robotic structure, communication protocol.

LISTA DE ILUSTRAÇÕES

Figura 1 –	Machine Learning	16
Figura 2 –	Plano frontal (AA'), plano sagital (BB')	20
Figura 3 –	Movimentos do punho humano. (a) Flexão e Extensão; (b) Desvios radial e ulnar; (c) supinação e pronação.	21
Figura 4 –	Dispositivo de Reabilitação	23
Figura 5 –	Box Adventure	25
Figura 6 –	Space Adventure	26
Figura 7 –	Bird Adventure	26
Figura 8 –	Taxa de Aceleração nos jogos	28
Figura 9 –	Taxa de Desaceleração por morte nos jogos	29
Figura 10 –	Resultados do Box Adventure	31
Figura 11 –	Resultados do Space Adventure	32
Figura 12 –	Resultados do Bird Adventure	33

LISTA DE EQUAÇÕES

Equação 1 – Cálculo da jogabilidade do paciente para classificação da IA

28

LISTA DE ABREVIATURAS E SIGLAS

CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
UFU	Universidade Federal de Uberlândia
CAD	<i>Computer Aided Design</i>
PLA	Ácido Polilático
TCP-IP	Protocolo de Controle de Transmissão - Protocolo de Internet
UDP	Protocolo de datagramas do usuário
IDE	<i>Arduino Integrated Development Environment</i>
AVE	Acidente Vascular Encefálico
IA	Inteligência Artificial
ML	<i>Machine Learning</i>

SUMÁRIO

1	INTRODUÇÃO	14
2	OBJETIVOS.....	15
3	REVISÃO BIBLIOGRÁFICA.....	16
3.1.	Machine Learning.....	16
3.2.	Deep Learning.....	17
3.3.	Jogos Sérios no software Unity 3D	17
3.4.	Equipamento Robótico para Reabilitação.....	17
3.5.	Fundamentos para Reabilitação.....	18
3.6.	Revisão Cinesiológica do Punho	19
4	METODOLOGIA	22
4.1.	Dispositivo de Reabilitação	22
4.1.1	Motor Bosch 24 V L006WM0310	23
4.1.2	Ponte H BTS7960B	23
4.1.3	Encoder Rotativo E40S6-1000	23
4.1.4	Torquímetro MKDC-100	24
4.1.5	Arduino Mega	24
4.2.	Jogos Sérios	24
4.3.	Comunicações entre o Dispositivo de Reabilitação e o Jogo Sério	27
4.4.	Implementação da Inteligência Artificial	28
5	RESULTADOS.....	31
5.1.	Box Adventure	31
5.2.	Space Adventure	31
5.3.	Bird Adventure	32
6	DISCUSSÃO	34
7	CONCLUSÕES	35
8	REFERÊNCIAS	36
	Apêndices	38
	Apêndice A – Códigos relacionados com a comunicação dos softwares, controle, dados e IA.....	39
	Apêndice B – Códigos dos jogos.....	55

1 INTRODUÇÃO

A reabilitação do corpo humano é uma área de grande importância e que tem recebido cada vez mais atenção. Com o desenvolvimento de novas tecnologias, especialmente as estruturas robóticas, os tratamentos de reabilitação tornaram-se mais eficazes e personalizados (CECCARELLI e ROMDHANE, 2010). Nesse contexto, os jogos sérios são uma ferramenta de motivação muito utilizada para manter os pacientes engajados durante as sessões de reabilitação (MASSON, 2015).

Porém, nem todos os jogos sérios são igualmente eficazes para todos os pacientes. Cada indivíduo tem características e necessidades específicas que devem ser consideradas na escolha dos jogos e na definição do nível de dificuldade. Para solucionar esse problema, tem-se compreendido o uso de técnicas de inteligência artificial, que permitem personalizar ainda mais os jogos sérios para cada paciente (GONÇALVES et al., 2015).

Neste projeto de conclusão de curso, será explorado o uso de técnicas de inteligência artificial para o desenvolvimento de jogos sérios aplicados a uma estrutura robótica para reabilitação do punho humano. O objetivo principal é utilizar a inteligência artificial para deixar os pacientes mais engajados, ajustando o nível de dificuldade dos jogos em função das ações do paciente. Além disso, a inteligência artificial permitirá o acompanhamento mais preciso da evolução do paciente, a partir dos dados coletados durante as sessões de reabilitação.

Serão abordados aspectos teóricos e práticos sobre a utilização de técnicas de inteligência artificial na área de reabilitação, bem como a aplicação dessas técnicas na criação de jogos sérios. O projeto será focado na reabilitação do punho humano com o intuito de particularizar as sessões de reabilitação para cada paciente maximizando os possíveis ganhos.

Em resumo, este trabalho tem como objetivo explorar o uso de técnicas de inteligência artificial no desenvolvimento de jogos sérios aplicados à estrutura robótica para reabilitação do punho humano. Espera-se que os resultados deste projeto possam contribuir para uma maior eficácia e personalização dos tratamentos de reabilitação, trazendo benefícios para os pacientes e profissionais envolvidos.

2 OBJETIVOS

A reabilitação é um processo contínuo e em constante evolução que visa ajudar indivíduos a se adaptarem a novas realidades após situações traumáticas, sejam elas de natureza física ou cognitiva. Inicialmente, o foco principal da reabilitação era a restauração da função motora afetada, mas pesquisas e estudos posteriores demonstraram que o objetivo mais amplo é permitir que os indivíduos vivam com a maior liberdade e autonomia possíveis, adaptando-se às suas capacidades e recursos disponíveis (LANG et al. 2016). Nesse contexto, o treinamento robótico surge como uma alternativa promissora na reabilitação, oferecendo vantagens como repetibilidade, controle da assistência ou resistência, e a capacidade de medir quantitativamente o desempenho do paciente (CECCARELLI e ROMDHANE, 2010).

Além disso, a abordagem de treinamento intensivo e direcionado para tarefas tem se mostrado eficaz na promoção da aprendizagem de movimentos e na restauração da funcionalidade. No caso específico da reabilitação do punho humano, compreender a anatomia dessa articulação, que permite uma variedade de movimentos de supinação, pronação, flexão e extensão desvios radial e ulnar é fundamental para projetar intervenções eficazes.

O objetivo geral deste trabalho é, portanto, o estudo, análise e implementação de técnicas de inteligência artificial aplicadas a novos jogos sérios associados a equipamento robótico para a reabilitação do punho humano. Essas técnicas visam personalizar as sessões de reabilitação para cada paciente, considerando suas necessidades e capacidades individuais. Ao integrar a inteligência artificial com equipamentos robóticos e jogos sérios, busca-se maximizar os possíveis ganhos de reabilitação, tornando o processo mais envolvente, adaptável e eficaz.

O trabalho proposto não apenas se concentra na restauração da movimentação do punho, mas também na melhoria da qualidade de vida do paciente, permitindo que eles retomem atividades cotidianas com mais facilidade e confiança. Portanto, a implementação de técnicas de inteligência artificial em jogos sérios para a reabilitação do punho humano é uma abordagem inovadora e promissora que visa atender às necessidades individuais de cada paciente e melhorar seu processo de reabilitação de maneira significativa.

3.2. Deep Learning

Deep Learning, ou Aprendizado Profundo, é uma subárea do *Machine Learning* que se baseia em redes neurais artificiais com várias camadas, também conhecidas como redes neurais profundas. O termo "profundo" refere-se ao número substancial de camadas ocultas em uma rede neural, que permite o aprendizado de representações complexas dos dados. Essas redes são capazes de aprender automaticamente características hierárquicas dos dados, o que as torna particularmente adequadas para tarefas de alto nível, como visão computacional, processamento de linguagem natural e reconhecimento de padrões. O *Deep Learning* revolucionou o campo do *Machine Learning*, permitindo avanços significativos em áreas como reconhecimento de imagem, tradução automática, carros autônomos e muito mais (LIU; LANG, 2019).

3.3. Jogos Sérios no software Unity 3D

Os Jogos Sérios no Unity 3D representam uma abordagem inovadora e poderosa para o desenvolvimento de experiências interativas com finalidades educacionais, de treinamento e simulação. O Unity 3D é uma das principais *engines* de desenvolvimento de jogos usada nesse contexto devido à sua flexibilidade, capacidades de criação 3D e suporte multiplataforma (MACHADO, 2014). Jogos Sérios são projetados com a intenção de fornecer treinamento prático, simulações ou experiências educacionais que imitam aspectos do mundo real. Eles são amplamente utilizados em campos como educação, saúde, indústria, segurança e defesa para fornecer treinamento eficaz e prático a alunos, profissionais e equipes (MASSON, 2015).

3.4. Equipamento Robótico para Reabilitação

O Equipamento robótico para reabilitação é uma classe de dispositivos projetados para auxiliar indivíduos na recuperação física após lesões, cirurgias ou condições médicas que afetam a mobilidade. Esses sistemas combinam tecnologia robótica e princípios de fisioterapia para oferecer tratamentos personalizados e eficazes. Esses dispositivos podem variar desde exoesqueletos que auxiliam na reabilitação de membros até máquinas de treinamento para membros superiores e inferiores. Equipamentos robóticos para reabilitação são projetados para fornecer treinamento intensivo e tarefas direcionadas, permitindo que os pacientes recuperem a

força, a amplitude de movimento e a coordenação (GONÇALVES et al., 2015; POLI et al., 2013).

3.5. Fundamentos para Reabilitação

A diminuição das capacidades físicas após um Acidente Vascular Encefálico (AVE) está diretamente ligada a danos cerebrais. A reabilitação tradicional, com terapeutas, mostra resultados, mas é limitada em tempo e esforço. Para superar essas limitações, a reabilitação robótica introduziu métodos eficazes, permitindo movimentos repetitivos no membro afetado e avaliação quantitativa do progresso (HIDLER et al., 2005).

A reabilitação convencional, com terapeutas que realizam movimentos assistidos e repetitivos no membro afetado, mostrou resultados promissores na redução de deficiências e melhoria da função motora. No entanto, essa forma de terapia demanda tempo e esforço, limitando a quantidade de tratamento que um paciente pode receber. Em busca de técnicas de reabilitação mais eficazes, o campo evoluiu para métodos de reabilitação robótica. Estudos indicam que o movimento repetitivo no membro comprometido resulta em melhoria da função motora, e a abordagem robótica proporciona uma maneira eficaz de permitir aos pacientes praticar esses movimentos, além de oferecer uma avaliação quantitativa do progresso (HIDLER et al., 2005).

É importante destacar que a introdução da robótica na reabilitação não busca substituir o terapeuta, mas sim colaborar com ele. A intenção é que a robótica seja utilizada em conjunto, ajudando em movimentos pesados, desafiadores ou repetitivos, enquanto o terapeuta continua fornecendo informações essenciais para avaliar as estratégias gerais de movimentação do paciente. A reabilitação é um processo dinâmico que requer uma equipe multiprofissional trabalhando em conjunto. Seu objetivo é treinar para compensar funções específicas perdidas, adaptando-se às necessidades individuais de cada paciente, variando em intensidade, duração e metas (BIAA, 2019). A lesão cerebral pode resultar em fraqueza ou paralisia dos membros, tornando essencial a movimentação e estimulação para prevenir contraturas, manter a amplitude de movimento das articulações, tônus muscular e força.

O processo de reabilitação visa capacitar o paciente a recuperar habilidades naturais do corpo e promover a reaprendizagem cerebral, permitindo uma recuperação mais eficiente. Seu foco é possibilitar que as pessoas realizem suas atividades diárias com segurança e independência (BIAA, 2019).

3.6. Revisão Cinesiológica do Punho

O dispositivo robótico utilizado neste projeto visa contribuir para a reabilitação do punho humano.

A mão possui várias funções, primariamente de sensação e preensão e secundariamente, expressão, comunicação e equilíbrio, sendo que todas proporcionam papéis fundamentais para execução de atividades de vida diária (HUNTER e CROME, 2002). A estabilidade dinâmica das articulações proximais do membro superior, ombro e tronco influenciam o alcance, ou posicionamento eficiente da mão para a função (HUNTER e CROME, 2002). Por essas razões, a mão e o braço são considerados como uma unidade funcional para a função do membro superior (HUNTER e CROME, 2002).

O punho, uma articulação distal do membro superior, conecta a mão ao antebraço, apresentando dois graus de liberdade para movimentos de abdução-adução e flexão-extensão. A rotação da mão ocorre nas articulações rádio-ulnares, pertencentes ao cotovelo e ao antebraço, fornecendo um terceiro grau de liberdade ao punho, correspondendo à pronação e supinação. Assim, o punho pode ser comparado a uma articulação esférica com três graus de liberdade para rotação, restritos pelas amplitudes dos movimentos associados ao punho (KAPANDJI, 2000).

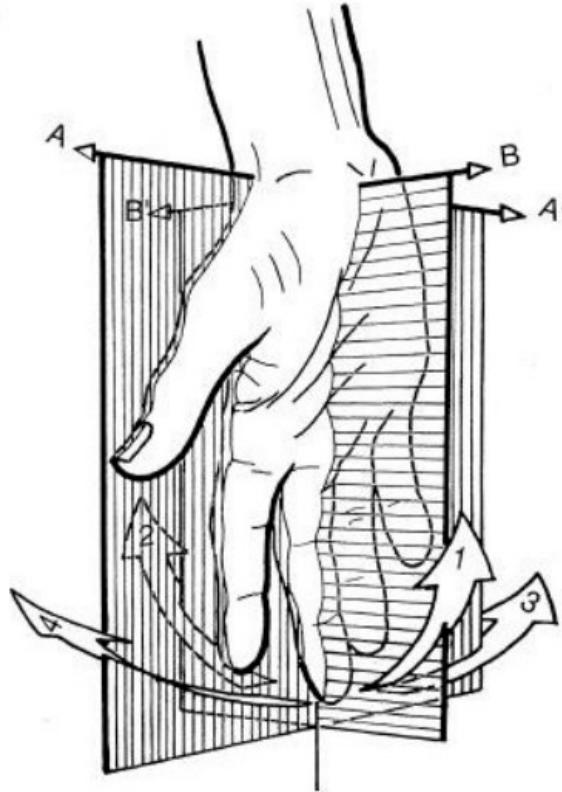
Para estudar o punho, é essencial definir as referências geométricas da mão. O eixo longitudinal da mão é definido pelo terceiro metacarpo em posição reta, e os planos frontal e sagital são estabelecidos conforme a Figura 2. Os movimentos do punho ocorrem em torno de dois eixos, com a mão em posição anatômica, ou seja, com a máxima supinação alinhada ao eixo longitudinal do braço.

O eixo AA', transversal e pertencente ao plano frontal, condiciona os movimentos de flexão-extensão no plano sagital. A flexão (seta 1 na Fig. 2) é a aproximação da superfície palmar da mão à superfície anterior do antebraço, enquanto a extensão (seta 2 na Fig. 2) é caracterizada pela aproximação da superfície dorsal da mão à superfície posterior do antebraço (KAPANDJI, 2000).

O eixo BB', ântero-posterior e pertencente ao plano sagital, condiciona os movimentos de adução-abdução no plano frontal. A adução (seta 3 na Fig. 2) ocorre quando a mão se aproxima do eixo do corpo e do lado interno (lado do dedo mínimo), formando um ângulo obtuso para dentro com o antebraço. A abdução (seta 4 na Fig. 2) envolve o afastamento da

mão do eixo do corpo e do lado externo, formando um ângulo obtuso com o antebraço (KAPANDJI, 2000).

Figura 2: – Plano frontal (AA'), plano sagital (BB')



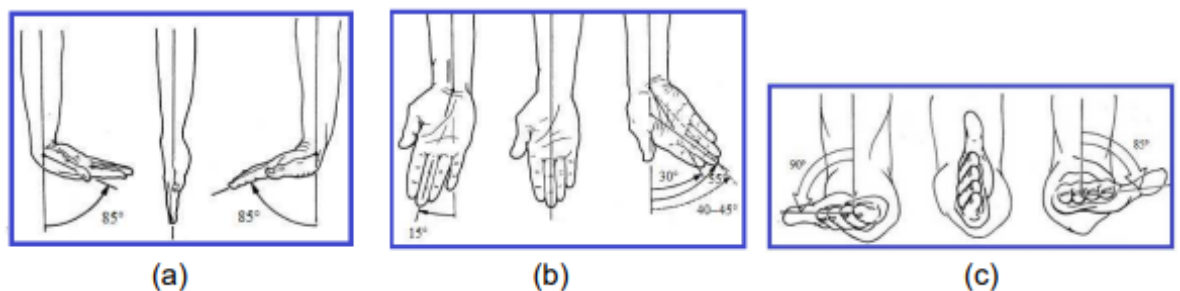
Fonte: Kapandji, 2010.

Em resumo, os movimentos de flexão e extensão do punho ocorrem no plano sagital, com a palma da mão movendo-se em direção à superfície anterior do antebraço na flexão e a superfície dorsal da mão em direção à superfície posterior do antebraço na extensão (Fig. 3). A amplitude desses movimentos é de 85°, sendo que o ângulo máximo ocorre na ausência de movimentos de abdução-adução e na posição de supinação.

A abdução e adução, ou desvio radial e ulnar, ocorrem no plano frontal em torno do eixo ântero-posterior. O desvio radial tem uma amplitude de até 15°, enquanto o desvio ulnar possui diferentes amplitudes dependendo do referencial adotado (Fig. 3). A adução atinge de 25 a 30° quando a mão está em pronação, sendo que as amplitudes são mínimas durante a flexão-extensão devido à tensão nos ligamentos do carpo.

O movimento de pronação e supinação ocorre nas articulações radioulnares proximal e distal do antebraço, envolvendo a rotação do rádio em torno da ulna. A supinação ocorre com a palma da mão para cima e o polegar para fora, enquanto a pronação acontece com a palma para baixo e o polegar para dentro. A posição intermediária é determinada pela direção do polegar para cima e da palma para dentro, com amplitudes de 90° para supinação e 85° para pronação (Fig. 3).

Figura 3: – Movimentos do punho humano. (a) Flexão e Extensão; (b) Desvios radial e ulnar; (c) supinação e pronação.



Fonte: Kapandji, 2010.

Para a reabilitação do punho, alguns protocolos são seguidos para orientar quais movimentos devem ser executados. Os movimentos de flexão, extensão, desvio radial e desvio ulnar do punho, geralmente utilizados não só para promover sua recuperação, mas também otimizar a saúde e a destreza do punho, proporcionando aos indivíduos uma jornada de reabilitação abrangente.

4 METODOLOGIA

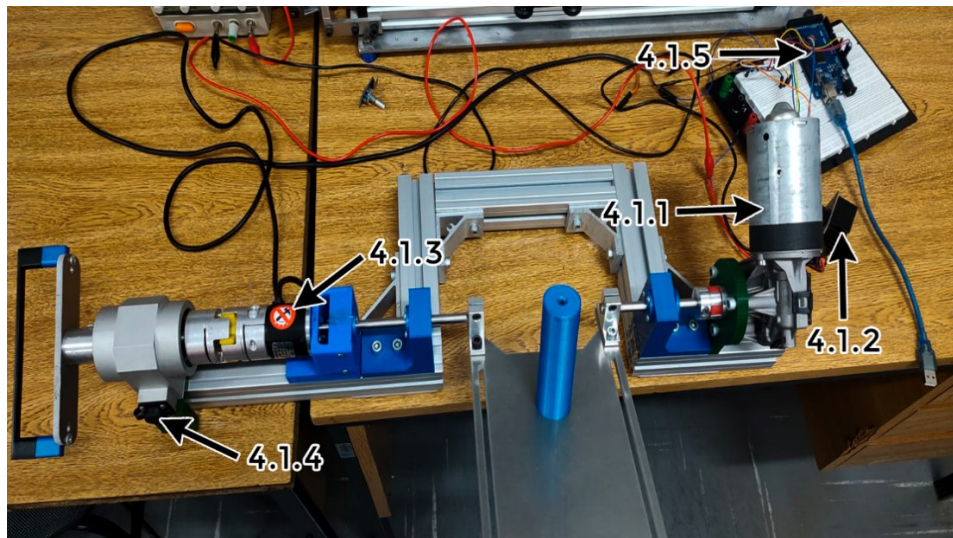
A metodologia adotada para o desenvolvimento deste trabalho consiste em uma abordagem sistemática que envolve diferentes etapas. O objetivo principal é reabilitar o punho de uma forma interativa e eficaz, inicialmente foram desenvolvidos os jogos sérios, o dispositivo de reabilitação foi utilizado no projeto, a interação com o computador via Arduino, após esse momento a comunicação Arduino-Matlab e Matlab-Unity, o controle sendo feito com os dados no Matlab, dessa forma finalizando com a implementação da inteligência artificial para ajustar variáveis nos jogos.

4.1. Dispositivo de Reabilitação

O dispositivo de reabilitação (Fig. 4) representa um avanço significativo na integração de componentes eletrônicos e estruturais para poder promover eficazmente a recuperação física.

A base do dispositivo é formada por perfis extrudados de Alumínio V-slot 40x40 complementadas por elementos de impressão 3D fabricados com PLA. O mecanismo possui um motor, uma ponte H, uma fonte de alimentação de tensão e corrente contínua, um encoder para feedback de posição angular, um torquímetro para medição do torque feito pelo paciente/equipamento, e uma placa Arduino como microcontrolador acoplado a um notebook. A mão do paciente é fixa a uma plataforma móvel que permite os principais movimentos individuais do punho apresentados na seção 3.6. O dispositivo pode adaptar-se às necessidades individuais de reabilitação, proporcionando uma plataforma versátil e personalizável para a promoção da recuperação física. O desenvolvimento deste equipamento não é objetivo deste projeto.

Figura 4: Dispositivo de Reabilitação



Fonte: Autor, 2023

A seguir são detalhados os principais componentes do protótipo utilizado neste projeto.

4.1.1 Motor Bosch 24 V L006WM0310

O motor Bosch, juntamente com uma caixa de redução é um componente essencial para o protótipo, porém não foi utilizado no projeto por conta que o foco é o desenvolvimento e controle dos jogos utilizando inteligência artificial, com isso foi decidido que o paciente teria livre movimento nesta etapa do projeto.

4.1.2 Ponte H BTS7960B

A ponte H permite controlar o sentido e o torque do motor, isso tudo por conta que ela controla o sentido da corrente nos enrolamentos do motor, seu funcionamento é baseado em quatro transistores que operam como chaves eletrônicas, dessa forma enviando o sentido e a força para que o motor consiga movimentar a plataforma.

4.1.3 Encoder Rotativo E40S6-1000

O encoder rotativo é utilizado para mensurar a posição angular da plataforma móvel, dessa forma todo controle depende de seus dados, visto que esse dado será a variável que irá

controlar a movimentação do personagem dentro dos jogos, além de saber o quanto o paciente está conseguindo movimentar a plataforma.

4.1.4 Torquímetro MKDC-100

O motor possui alta redução e bloqueia o controle manual da plataforma pelo paciente, o torquímetro será necessário para saber se o paciente está tentando movimentar a plataforma e dessa forma envia um sinal que é trabalhado e devolvido ao motor, porém como o motor não foi utilizado, não houve necessidade da utilização do torquímetro também neste projeto.

4.1.5 Arduino Mega

O Arduino Mega é utilizado por ter uma fácil programação e interpretação de dados, conseguindo dessa forma comunicar com o torquímetro, ponte H e o encoder. Assim ele receberá os dados do torquímetro e do encoder enviando-os ao software Matlab para trabalhar os dados e enviando o que o motor precisa fazer à ponte H.

4.2. Jogos Sérios

Para o início do desenvolvimento do projeto foi realizado um estudo sobre as funcionalidades de cada parte do projeto, como o Unity 3D, IDE Arduino e Matlab, para dessa forma conseguir fazer o planejamento dos jogos e assim conseguir chegar em um resultado otimizado e divertido para o paciente conseguir a reabilitação.

Para o primeiro jogo foi escolhido um jogo de plataforma cujo objetivo é o jogador, que comanda um cubo, desviar das armadilhas que caem do céu. Este jogo foi designado para reabilitação dos movimentos de Supinação e Pronação. O modelo base foi inspirado em um tutorial no youtube do canal Fredi, playlist 'Unity' (@FrediDev). No modelo do jogo (Fig. 5), o paciente comanda o bloco preto com detalhes em amarelo tendo que se esquivar dos blocos em vermelho com espinhos que podem surgir aleatoriamente entre as posições -10 e 10 do eixo X, surgindo aleatoriamente 1 a 5 caixas de uma vez, a parte visual foi feita utilizando pacotes de vetores gratuitos do site 'quaternius' (@FrediDev), vale ressaltar que outros sites também possuem pacotes gratuitos e cada projetista pode escolher o pacote que mais lhe agrada para a ambientação de seu jogo.

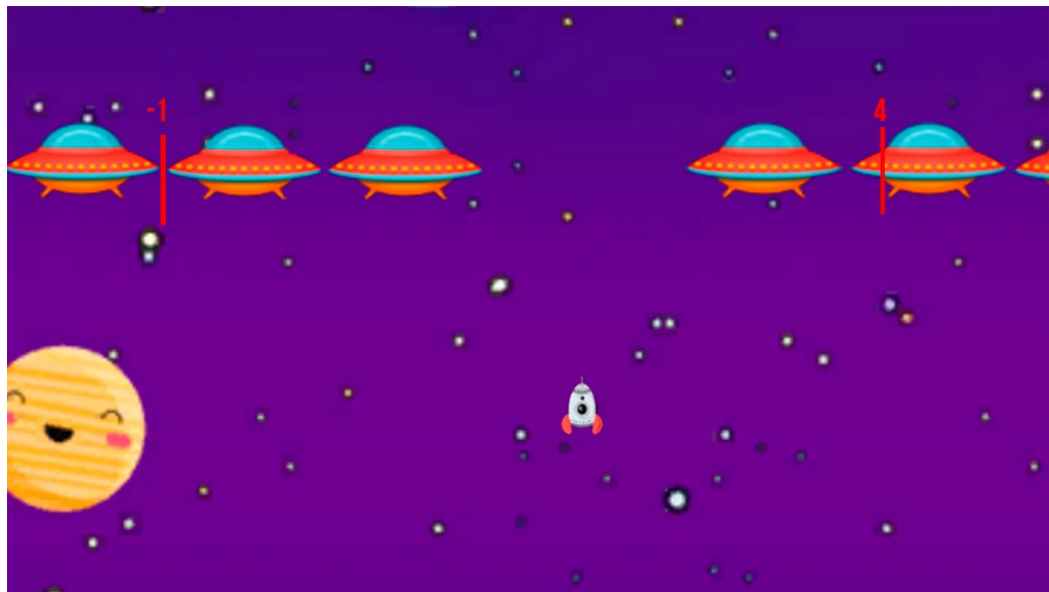
Figura 5: Box Adventure



Fonte: Elaborado pelo autor, 2023.

Para o jogo 2, que será designado para o movimento de desvios radial e ulnar, o jogo será do estilo de *infinity run*, tendo como exemplos mais famosos, jogos como *Temple Run* e *Subway Surfers*, porém tem-se que considerar que existem apenas duas variáveis para esse movimento, então a mecânica do jogo será desviar de obstáculos apenas lateralmente, sem ser incrementado o movimento de pulo e rolamento. Por ter seu funcionamento parecido com o jogo 3, que foi desenvolvido primeiro, foi-se aproveitado vários códigos fazendo ajustes mecânicos. No jogo (Fig. 6), o objetivo é o jogador conseguir passar entre os obstáculos, naves alienígenas, que surgem aleatoriamente entre as posições -1 a 4 no eixo X.

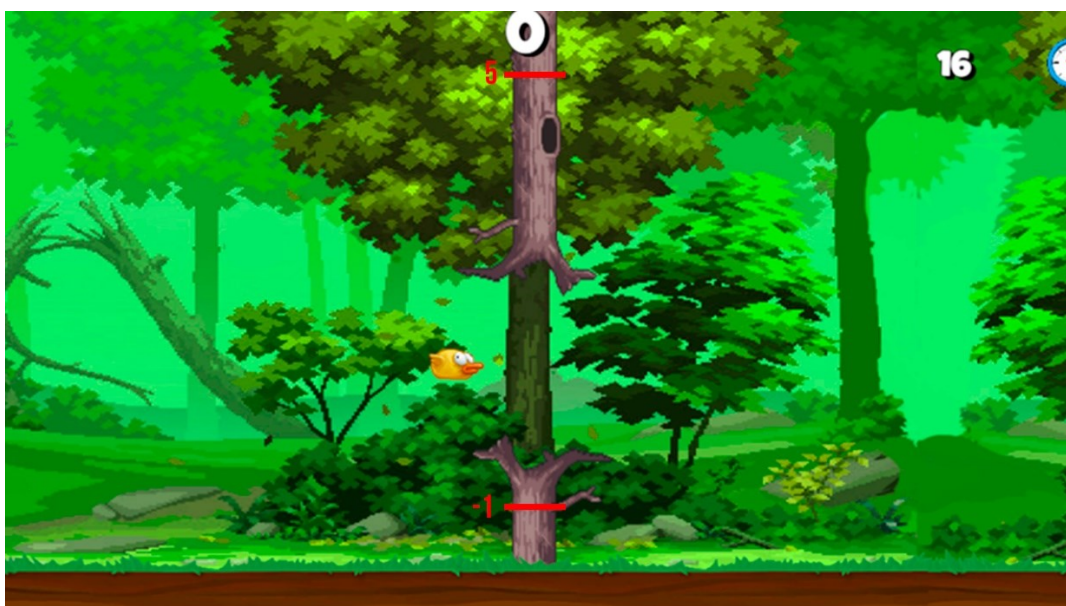
Figura 6: Space Adventure



Fonte: Elaborado pelo autor, 2023.

O jogo 3 será designado aos movimentos de Flexão e Extensão, ele será um jogo de ação estilo arcade, podemos considerar como um *infinity fly*, pois os movimentos serão verticais, com uma velocidade constante na horizontal, um jogo utilizado como referência foi o *Flappy Bird*. No jogo (Fig. 7), o objetivo é o jogador conseguir passar entre os obstáculos, árvores, que surgem aleatoriamente entre as posições -1 a 5 no eixo Y.

Figura 7: Bird Adventure



Fonte: Elaborado pelo autor, 2023.

Durante o desenvolvimento dos jogos, adotou-se uma escala universal com o valor "1" para facilitar a comparação de variáveis, proporcionando uma base consistente para ajustes e análises. Essa abordagem visa não apenas simplificar a interpretação de dados, mas também garantir coesão entre os diferentes parâmetros do jogo, resultando em uma experiência mais equilibrada. Além disso, todas as imagens utilizadas foram disponibilizadas gratuitamente ou criadas.

4.3. Comunicações entre o Dispositivo de Reabilitação e o Jogo Sérico

Na implementação do projeto, estabeleceu-se uma comunicação entre o Arduino e o Matlab, utilizando o ambiente de desenvolvimento integrado (IDE). Para facilitar a transmissão de dados, optou-se pela comunicação serial, permitindo que o Arduino enviasse informações para o controle do personagem. Especificamente, o valor do encoder, que representa a posição angular da plataforma foi transmitido ao Matlab.

Além disso, a integração entre o Matlab e os ambientes UNITY foi realizada por meio do protocolo UDP. Essa escolha estratégica permitiu que o Matlab enviasse informações sobre a movimentação, de acordo com o valor do encoder, do jogador principal nos jogos, utilizando uma porta serial dedicada. Por sua vez, os jogos Unity responderam fornecendo dados essenciais, como a posição do jogador, o número de mortes, o score e informações sobre a passagem pelos obstáculos. Essa comunicação bidirecional entre o Matlab e os jogos UNITY possibilita a sincronização e monitoramento preciso das atividades do paciente no ambiente virtual, concentrando assim todo o controle do projeto no Matlab.

Dessa forma, o sistema global é integrado de maneira coesa, com o Matlab atuando como o ponto central de controle. A utilização de diferentes protocolos de comunicação, como TCP-IP para Arduino e UDP para UNITY, demonstra a adaptação estratégica para atender às necessidades específicas de cada componente do projeto, garantindo uma comunicação robusta e eficaz entre eles. A descrição destes protocolos está detalhada no Apêndice A.

Em todos os três jogos, o Arduino envia ao Matlab o valor do encoder que é na casa das centenas, o Matlab envia ao Unity esse valor e por conta que a movimentação do personagem é na casa das unidades, esse valor recebido é dividido por um número diferente em cada jogo para dar maior estabilidade ao personagem principal, para o jogo 1 o valor foi 175, para os outros jogos o valor foi 100. Outro ponto em comum nos jogos são os dados que são

enviados ao Matlab, todos os jogos enviam a quantidade de morte e o score total desde que o jogo foi iniciado, tratando os dados e com isso irá devolver ao jogo a velocidade que será somada a velocidade padrão dos obstáculos.

4.4. Implementação da Inteligência Artificial

A implementação da Inteligência Artificial foi feita no Matlab, seguindo um algoritmo relacionado a lógica Fuzzy (TANAKA e Wang, 2001). Foi criada uma função matemática que calcula o nível de jogabilidade do paciente, conforme a Equação (1):

$$IA = \frac{2 * Mortes^2}{Pontuação} \quad (1)$$

O valor da variável IA tem a seguinte lógica:

- a) Se $IA < 0,5$ – Jogador excelente
- b) Se $0,5 \leq IA < 0,65$ – Jogador bom
- c) Se $0,65 \leq IA < 0,8$ – Jogador regular
- d) Se $0,8 \leq IA < 1$ – Jogador ruim
- e) Se $1 < IA$ – Jogador péssimo

Dependendo dessa classificação, a Inteligência Artificial varia a constante de aumento de velocidade dentro do jogo, por exemplo, um jogador classificado como ótimo terá uma taxa de aceleração dentro do jogo cinco ou seis vezes maior que a taxa de aceleração de um jogador classificado como péssimo, conforme Figura (8).

Figura 8: Taxa de Aceleração nos jogos



Fonte: Elaborado pelo autor, 2023.

Sabendo que o gráfico da aceleração é uma reta de primeiro grau, logo a velocidade é aumentada de acordo com uma parábola de segundo grau. Assim, a Inteligência Artificial consegue com o passar do tempo adaptar a variável classificatória para nivelar o jogo, dessa forma facilitando o jogo para o paciente que possui maior dificuldade de movimentação, enquanto dificulta o jogo para o paciente que possui maior mobilidade.

Vale ressaltar que cada morte reduz a velocidade dos obstáculos, essa redução depende da velocidade que o obstáculo estava antes da morte, conforma na figura (9).

Figura 9: Taxa de Desaceleração por morte nos jogos



Fonte: Elaborado pelo autor, 2023.

- a) Se $V < -25$ – Muito Lento
- b) Se $-25 \leq IA < -10$ – Lento
- c) Se $-10 \leq IA < 10$ – Regular
- d) Se $10 \leq IA < 25$ – Rápido
- e) Se $25 < IA$ – Muito Rápido

Como o jogador que estará com os obstáculos em uma maior velocidade morrerá menos, ele é mais penalizado por cada morte, assim o jogo também vai se ajustando ao nível motor do paciente.

Dito isso a Inteligência Artificial programada não possui redes neurais e com isso ela não consegue ajustar as variáveis automaticamente, mas caso necessário um programador consegue fazer ajustes nessas variáveis, dessa forma o algoritmo obedece a uma lista de regras para tratar os dados e fornecer uma boa experiência com a reabilitação. A inserção de redes neurais é uma implementação futura para o projeto.

Vale mencionar que no jogo 1, a quantidade de caixas que podem surgir depende se a velocidade dos obstáculos estiver muito lento ou lento, assim surgindo de 1 a 3 caixas, se estiver regular será de 2 a 4 caixas, enquanto se estiver rápido ou muito rápido surgirá de 3 a 5 caixas, ou seja a inteligência artificial tem uma influência indireta na quantidade de caixas que podem surgir.

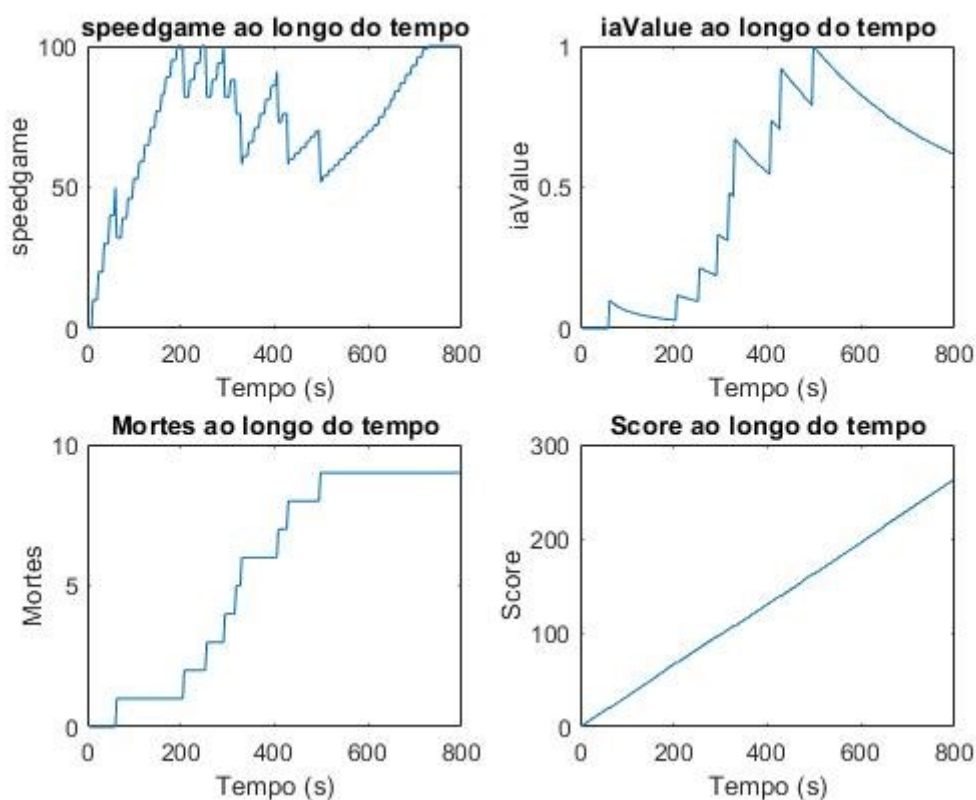
5 RESULTADOS

Esta seção apresenta os dados coletados dos jogos sérios desenvolvidos para poder tirar conclusões preliminares sobre a usabilidade dos jogos para a reabilitação do punho. A cada três segundos foram coletados os dados do score total, quantidade de mortes, o valor da equação da Inteligência Artificial e a velocidade adicional em cada jogo.

5.1. Box Adventure

Conforme a Figura (10) tem-se que a velocidade do jogo está de fato aumentando constantemente de acordo com a faixa escolhida do valor da função da Inteligência Artificial, o score sempre aumenta constantemente, e a quantidade de mortes interfere significativamente no valor da função da IA.

Figura 10: Resultados do Box Adventure

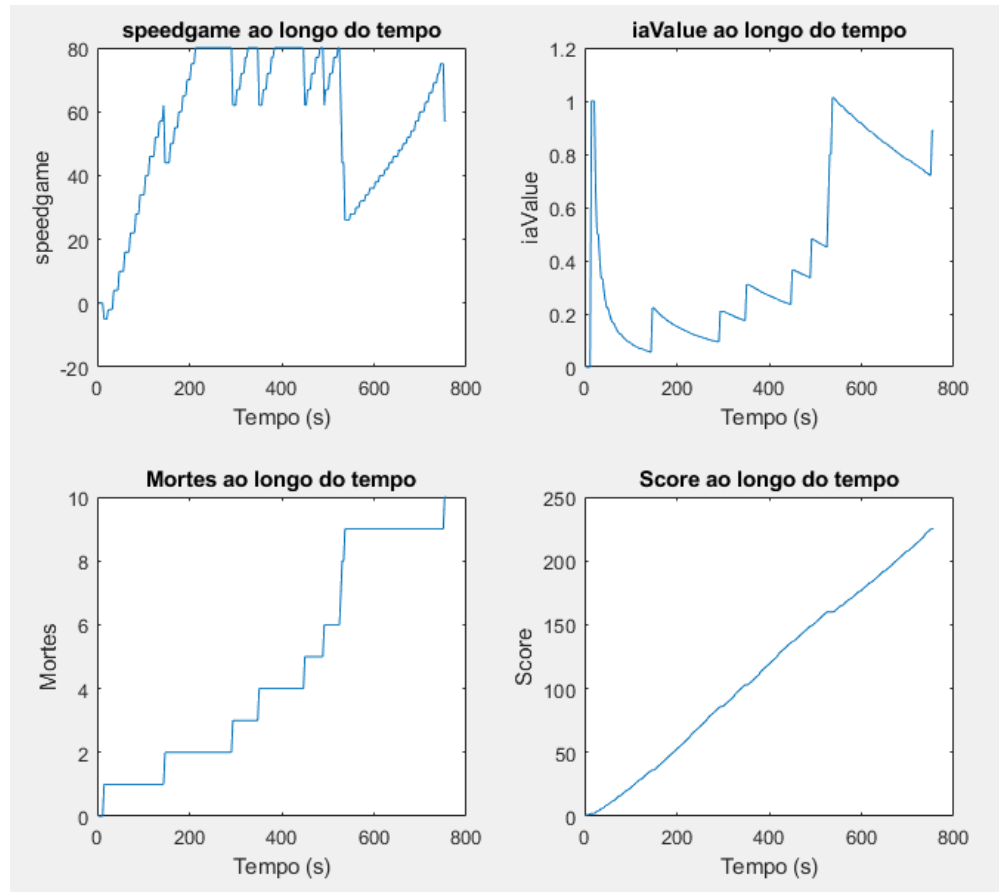


Fonte: Elaborado pelo autor, 2023.

5.2. Space Adventure

Foi realizado as medições do jogo Space Adventure mostrado na figura (11), confirmando que a mecânica planejada foi devidamente executada, a velocidade do jogo aumenta constantemente de acordo com a faixa classificatória do valor da função da IA para o paciente, a quantidade de mortes do jogador interfere consideravelmente no valor da função da Inteligência Artificial e o score vai aumentando constantemente.

Figura 11: Resultados do Space Adventure

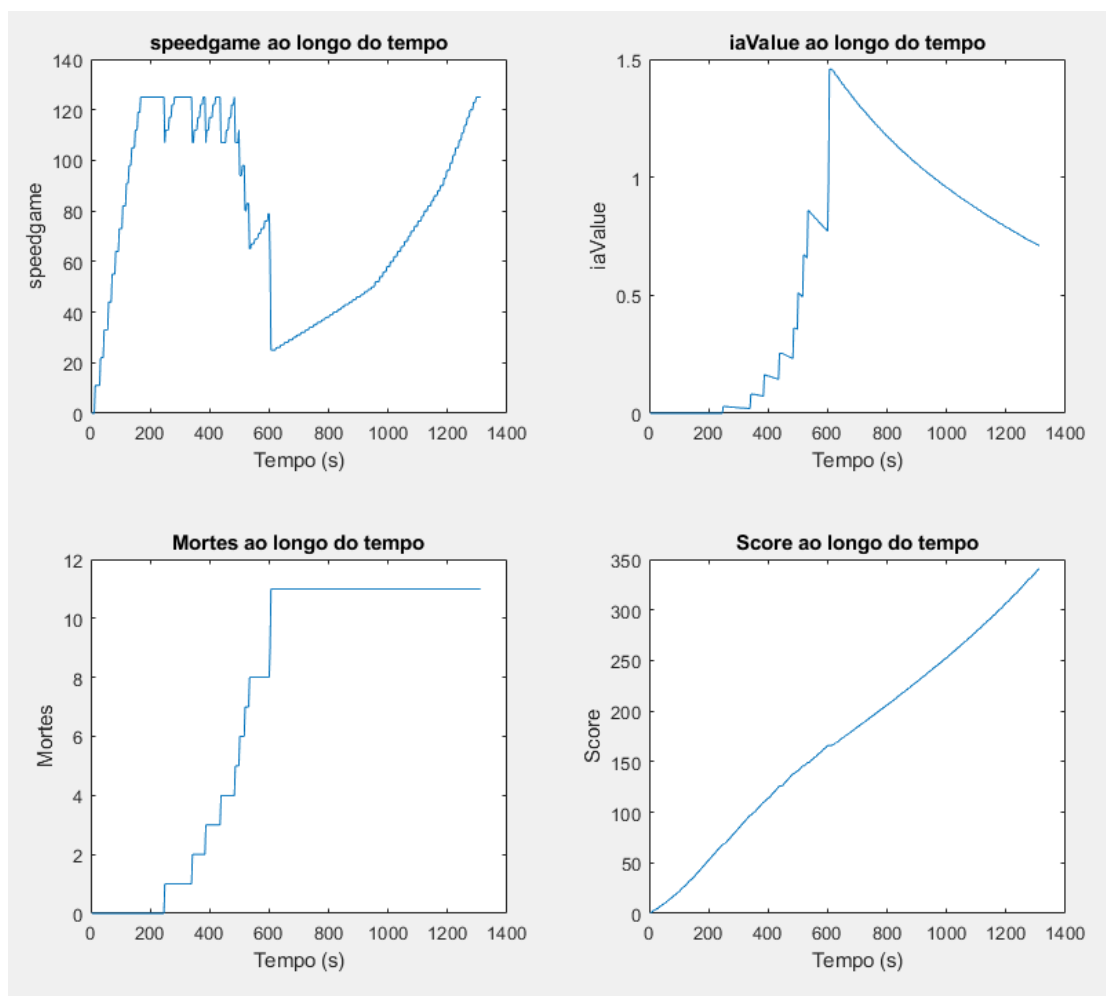


Fonte: Elaborado pelo autor, 2023.

5.3. Bird Adventure

As medições do jogo Bird Adventure mostrado na figura (12), estabelecendo que a execução está de acordo com o planejado, a velocidade do jogo subindo variando com a classificação do valor da função da IA para o paciente, a quantidade de mortes do jogador interfere substancialmente no valor da função da Inteligência Artificial e o score vai aumentando continuamente.

Figura 12: Resultados do Bird Adventure



Fonte: Elaborado pelo autor, 2023.

6 DISCUSSÃO

A implementação de técnicas de inteligência artificial (IA) no desenvolvimento de jogos sérios voltados para a reabilitação do punho humano poderá ajudar na personalização e melhora do paciente. A aplicação dessas tecnologias pode proporcionar benefícios tanto em termos da experiência menos monótona com a utilização dos jogos sérios, quanto na reabilitação, destacando-se como uma abordagem inovadora na promoção da recuperação e no aprimoramento da qualidade de vida para pacientes com comprometimento motor no punho.

Os resultados preliminares demonstraram que os sistemas robóticos integrados a jogos sérios podem ser capazes de oferecer uma experiência controlada e personalizada para os pacientes durante o processo de reabilitação.

Entretanto, algumas limitações foram identificadas, destacando a necessidade de aprimoramentos no sistema de aprendizado com redes neurais e o controle do motor com o torquímetro via inteligência artificial devido suas complexidades. Isso acarretou serem consideradas áreas de investigação futura para melhorarias no projeto.

Este estudo representa uma contribuição para o avanço do conhecimento no campo da reabilitação por meio da integração de IA em jogos sérios aplicados ao punho humano. Os conceitos apresentados fornecem uma base para o desenvolvimento de soluções eficazes e interativas no âmbito da robótica para a reabilitação.

Acredita-se que este projeto possa não apenas inspirar investigações futuras, mas também motivar a implementação prática dessas tecnologias em larga escala, promovendo melhorias substanciais na reabilitação do punho humano e contribuindo para a interação entre as áreas da saúde e robótica.

7 CONCLUSÕES

De acordo com os resultados preliminares e as discussões apresentadas, foi possível concluir que o uso de técnicas de inteligência artificial para o desenvolvimento de jogos sérios aplicados em estrutura robótica para a reabilitação do punho humano é um projeto promissor que utiliza robótica na área da fisioterapia.

O projeto mostrou-se adequado para posteriormente ser utilizado na reabilitação do punho humano com os jogos que podem se adequar as limitações de cada paciente, devido ao uso da IA. Essa forma interativa ajuda aliviar a tensão envolvida com a reabilitação criando um ambiente mais agradável e trazendo um objetivo a mais para a movimentação já necessária para recuperação.

Por fim, este trabalho reforça a importância do uso de técnicas robóticas para a evolução da tecnologia aplicada em diversas áreas, assim mostrando a importância da engenharia mecatrônica para o contexto atual da humanidade. A utilização de inteligência artificial, jogos sérios e estrutura robótica para a reabilitação, são exemplos de aplicações que podem trazer benefícios significativos para a área da saúde.

8 REFERÊNCIAS

BIAA – Brain Injury Association of America. **Brain Injury Rehabilitation**. Disponível em: <<https://www.biausa.org/brain-injury/about-brain-injury/treatment/brain-injury-rehabilitation>>. Acesso em 12 de novembro de 2023.

CAILLOIS, Roger. **Os jogos e os homens**. Lisboa – Portugal: Editora Cotovia, 1990. 232 p.

CECCARELLI, M.; ROMDHANE, L., **Design Issues for Human-Machine Platform Interface in Cable Based Parallel Manipulators for Physiotherapy Applications**, Journal of Zhejiang University SCIENCE A, 2010.

Disponível em: <https://doi.org/10.1631/jzus.A1000027>

CHO, S. et al. **Learning from machine learning in accounting and assurance**. [S.l.]: American Accounting Association, 2020.

GONÇALVES, R. S.; CARVALHO, J. C. M.; RIBEIRO, J.F; SALIM, V. V., **Cable-Driven Robot for Upper and Lower Limbs Rehabilitation. Handbook of Research on Advancements in Robotics and Mechatronics**. 1ed.: IGI Global, p. 284-315, 2015.

Disponível em: <https://doi.org/10.4018/978-1-4666-7387-8.ch011>

HIDLER, J.; NICHOLS, D.; PELLICCIO, M.; BRADY, K. **Advances in the understanding and treatment of stroke impairment using robotic devices**. Topics in Stroke rehabilitation, vol. 12, pp. 22-35, 2005.

Disponível em: <<https://doi.org/10.1310/RYT5-62N4-CTVX-8JTE>>.

HUNTER, S.; CROME, P. **Hand function and stroke**. Rev. Clin. Gerontol London, v.12, n.1, p.68-81, 2002.

KAPANDJI, A. I. **Fisiologia Articular**, 5 ed., vol. 1., Pinheiros, SP: Panamericana, 2000.

LANG, C. E. et al. **Dose response of task-specific upper limb training in people at least 6 months poststroke: A phase II, single-blind, randomized, controlled trial.** *Annals of urology*, v. 80, n. 3, p. 342–354, set. 2016.

Disponível em: <https://doi.org/10.1002/ana.24734>

LIU, H.; LANG, B. **Machine learning and deep learning methods for intrusion detection systems: A survey.** *Applied Sciences*, v. 9, n. 20, 2019. ISSN 2076-3417.

Disponível em: <https://www.mdpi.com/2076-3417/9/20/4396>.

MASSON, D. B. **A importância de investir em treinamento.** 2015.

Disponível em: <http://www.mascaropessoas.com.br/artigo/a-importancia-de-investir-em-treinamento.html>. Acesso: 05-08-2023.

POLI, P.; MORONE, G.; ROSATI, G.; MASIERO, S. **Robotic Technologies and Rehabilitation: New Tools for Stroke Patients' Therapy.** *BioMed Research International*, v.2013, 2013.

Tanaka, K.; Wang, H. **Fuzzy Control Systems Design and Analysis: A Linear Matrix Inequality Approach.** John Wiley & Sons, New York, NY. 2001

@FredDev. **Unity do Zero.** Disponível em: <https://www.youtube.com/@FredDev>
Acesso:14-04-2023.

Apêndices

Apêndice A – Códigos relacionados com a comunicação dos softwares, controle, dados e IA.

Código 1: Código do IDE Arduino de leitura do encoder e comunicação com o Matlab

```
#include <Encoder.h>

Encoder myEncoder(2, 3); // Substitua 2 e 3 pelos pinos conectados ao encoder.

void setup() {
  Serial.begin(9600); // Inicializa a comunicação serial
}

void loop() {
  long encoderValue = myEncoder.read(); // Lê o valor do encoder

  Serial.println(encoderValue); // Envia o valor pela porta serial

  // Aguarde um curto período de tempo para evitar sobrecarregar a porta serial
  delay(100);
}
```

Código 2: Código do Matlab de comunicação com o IDE e Unity, controle, IA e apuração de dados para o jogo Box Adventure

```
clear all;

% Inicialização e configuração de comunicação
dadosParaUnityAnterior = ''; % Variável para armazenar o valor anterior
s = serial('COM4', 'BaudRate', 9600);
fopen(s); % Abre a porta serial antes do loop

% Configuração do servidor UDP no MATLAB
unityReceivePort = 35000; % Porta para receber dados do Unity
unitySendPort = 52000; % Porta para enviar dados ao Unity
udpServer = udp('127.0.0.1', unityReceivePort, 'LocalPort', unitySendPort);
fopen(udpServer);

mortes = 0;
score = 0;
scorecont = 0;
speedgame = 0.00;
mortescont = 0;

% Inicializa vetores para armazenar dados para o gráfico
tempo = [];
```

```

speedgameHistory = [];
iaValueHistory = [];
mortesHistory = [];
scoreHistory = [];

% Contador de tempo para gravação
tempoAnterior = tic;
intervaloGravacao = 3; % Intervalo de 3 segundos
jogoRodando = true;
time = 0;

% Loop principal
while jogoRodando

    %recebe do arduino posição da plataforma
    while s.BytesAvailable > 0
        fread(s, s.BytesAvailable);
    end
    valorAtual = str2double(fgetl(s)); % Lê como string e converte para
número

    % Receber dados do Unity
    data = fread(udpServer, 1024, 'char');
    dataString = char(data);
    dataValues = str2double(strsplit(dataString, '!'));

    % Verificar se há elementos suficientes em dataValues
    if length(dataValues) >= 2
        mortes = dataValues(1)
        score = dataValues(2)
    else
        % Lidar com o caso em que dataValues não tem elementos suficientes
        figure;

        subplot(2, 2, 1);
        plot(tempo, speedgameHistory);
        title('speedgame ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('speedgame');

        subplot(2, 2, 2);
        plot(tempo, iaValueHistory);
        title('iaValue ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('iaValue');

        subplot(2, 2, 3);
        plot(tempo, mortesHistory);
        title('Mortes ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('Mortes');

        subplot(2, 2, 4);
        plot(tempo, scoreHistory);
        title('Score ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('Score');
        jogoRodando = false;
    end
end

```

```

end

if (speedgame < -25 && mortes > mortescont)
    speedgame = speedgame - 3;
    mortescont = mortes
elseif (speedgame < -10 && mortes > mortescont)
    speedgame = speedgame - 4;
    mortescont = mortes
elseif (speedgame < 10 && mortes > mortescont)
    speedgame = speedgame - 5;
    mortescont = mortes
elseif (speedgame < 25 && mortes > mortescont)
    speedgame = speedgame - 10;
    mortescont = mortes
elseif (mortes > mortescont)
    speedgame = speedgame - 18;
    mortescont = mortes
end

% Calcula a IA (ajuste conforme necessário)
iaValue = (mortes * mortes * 2) / (score);

% ajuste para velocidade ótima
if (mortes == 0 && score>scorecont+3)
    if (speedgame < 45)
        speedgame = speedgame + 10;
        scorecont = score;
    elseif (speedgame < 85)
        speedgame = speedgame + 9;
        scorecont = score;
    else
        speedgame = speedgame + 8;
        scorecont = score;
    end
end

% jogando bem
elseif (iaValue < 0.5 && score>scorecont+3)

    if (speedgame < 50)
        speedgame = speedgame + 7;
        scorecont = score;
    else
        speedgame = speedgame + 6;
        scorecont = score;
    end

elseif (0.5 <= iaValue && iaValue < 0.65 && score>scorecont+3)

    speedgame = speedgame + 5;
    scorecont = score;

elseif (0.65 <= iaValue && iaValue < 0.8 && score>scorecont+3)

    speedgame = speedgame + 3;
    scorecont = score;

elseif (0.8 <= iaValue && iaValue < 1.0 && score>scorecont+3)

```

```

        speedgame = speedgame + 2;
        scorecont = score;

elseif (iaValue > 1.0 && score>scorecont+3)

        speedgame = speedgame + 1;
        scorecont = score;

end

if speedgame > 100;
    speedgame = 100;
elseif speedgame < -50
    speedgame = -50;
else
end
fwrite(udpServer, uint8(sprintf('%d!%d', valorAtual, speedgame)));
if toc(tempoAnterior) >= intervaloGravacao
    % Gravação das variáveis
    time = time + intervaloGravacao;
    fprintf('Gravando: speedgame=%f, iaValue=%f, mortes=%d, score=%d\n',
speedgame, iaValue, mortes, score);

    % Adiciona dados ao histórico
    tempo = [tempo, time];
    speedgameHistory = [speedgameHistory, speedgame];
    iaValueHistory = [iaValueHistory, iaValue];
    mortesHistory = [mortesHistory, mortes];
    scoreHistory = [scoreHistory, score];

    % Reinicia o contador de tempo
    tempoAnterior = tic;
end
end

fclose(udpServer);
delete(udpServer);
fclose(s); % Feche a porta com o Arduino

```

Código 3: Código do Matlab de comunicação com o IDE e Unity, controle, IA e apuração de dados para o jogo Space Adventure

```

clear all;

% Inicialização e configuração de comunicação
dadosParaUnityAnterior = ''; % Variável para armazenar o valor anterior
s = serial('COM4', 'BaudRate', 9600);
fopen(s); % Abre a porta serial antes do loop

% Configuração do servidor UDP no MATLAB

```

```

unityReceivePort = 35000; % Porta para receber dados do Unity
unitySendPort = 52000; % Porta para enviar dados ao Unity
udpServer = udp('127.0.0.1', unityReceivePort, 'LocalPort', unitySendPort);
fopen(udpServer);

mortes = 0;
score = 0;
scorecont = 0;
speedgame = 0.00;
mortescont = 0;

% Inicializa vetores para armazenar dados para o gráfico
tempo = [];
speedgameHistory = [];
iaValueHistory = [];
mortesHistory = [];
scoreHistory = [];

% Contador de tempo para gravação
tempoAnterior = tic;
intervaloGravacao = 3; % Intervalo de 3 segundos
jogoRodando = true;
time = 0;

% Loop principal
while jogoRodando

    %recebe do arduino posição da plataforma
    while s.BytesAvailable > 0
        fread(s, s.BytesAvailable);
    end
    valorAtual = str2double(fgetl(s)); % Lê como string e converte para
número

    % Receber dados do Unity
    data = fread(udpServer, 1024, 'char');
    dataString = char(data);
    dataValues = str2double(strsplit(dataString, '!'));

    % Verificar se há elementos suficientes em dataValues
    if length(dataValues) >= 2
        mortes = dataValues(1)
        score = dataValues(2)
    else
        % Lidar com o caso em que dataValues não tem elementos suficientes
        figure;

        subplot(2, 2, 1);
        plot(tempo, speedgameHistory);
        title('speedgame ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('speedgame');

        subplot(2, 2, 2);
        plot(tempo, iaValueHistory);
        title('iaValue ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('iaValue');
    end
end

```

```

        subplot(2, 2, 3);
        plot(tempo, mortesHistory);
        title('Mortes ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('Mortes');

        subplot(2, 2, 4);
        plot(tempo, scoreHistory);
        title('Score ao longo do tempo');
        xlabel('Tempo (s)');
        ylabel('Score');
        jogoRodando = false;
    end

    if (speedgame < -25 && mortes > mortescont)
        speedgame = speedgame - 3;
        mortescont = mortes
    elseif (speedgame < -10 && mortes > mortescont)
        speedgame = speedgame - 4;
        mortescont = mortes
    elseif (speedgame < 10 && mortes > mortescont)
        speedgame = speedgame - 5;
        mortescont = mortes
    elseif (speedgame < 25 && mortes > mortescont)
        speedgame = speedgame - 10;
        mortescont = mortes
    elseif (mortes > mortescont)
        speedgame = speedgame - 18;
        mortescont = mortes
    end
    % Calcula a IA (ajuste conforme necessário)
    iaValue = (mortes * mortes * 2) / (score);

    % ajuste para velocidade ótima
    if (mortes == 0 && score > scorecont+2)
        if (speedgame < 45)
            speedgame = speedgame + 11;
            scorecont = score;
        elseif (speedgame < 85)
            speedgame = speedgame + 9;
            scorecont = score;
        else
            speedgame = speedgame + 7;
            scorecont = score;
        end
    end

    % jogando bem
    elseif (iaValue < 0.5 && score > scorecont+2)

        if (speedgame < 50)
            speedgame = speedgame + 6;
            scorecont = score;
        else
            speedgame = speedgame + 5;
            scorecont = score;
        end
    end

```

```

elseif (0.5 <= iaValue && iaValue < 0.65 && score>scorecont+2)

    speedgame = speedgame + 4;
    scorecont = score;

elseif (0.65 <= iaValue && iaValue < 0.8 && score>scorecont+2)

    speedgame = speedgame + 3;
    scorecont = score;

elseif (0.8 <= iaValue && iaValue < 1.0 && score>scorecont+2)

    speedgame = speedgame + 2;
    scorecont = score;

elseif (iaValue > 1.0 && score>scorecont+2)

    speedgame = speedgame + 1;
    scorecont = score;

end

if speedgame > 80
    speedgame = 80;
elseif speedgame < -50
    speedgame = -50;
else
end
end
fwrite(udpServer, uint8(sprintf('%d!%d', valorAtual, speedgame)));

if toc(tempoAnterior) >= intervaloGravacao
    % Gravação das variáveis
    time = time + intervaloGravacao;
    fprintf('Gravando: speedgame=%f, iaValue=%f, mortes=%d, score=%d\n',
speedgame, iaValue, mortes, score);

    % Adiciona dados ao histórico
    tempo = [tempo, time];
    speedgameHistory = [speedgameHistory, speedgame];
    iaValueHistory = [iaValueHistory, iaValue];
    mortesHistory = [mortesHistory, mortes];
    scoreHistory = [scoreHistory, score];

    % Reinicia o contador de tempo
    tempoAnterior = tic;
end

end

fclose(udpServer);
delete(udpServer);
fclose(s); % Fecha a porta com o Arduino

```

Código 4: Código do Matlab de comunicação com o IDE e Unity, controle, IA e apuração de dados para o jogo Bird Adventure

```
% Inicialização e configuração de comunicação
dadosParaUnityAnterior = ''; % Variável para armazenar o valor anterior
s = serial('COM4', 'BaudRate', 9600);
fopen(s); % Abre a porta serial antes do loop

% Configuração do servidor UDP no MATLAB
unityReceivePort = 35000; % Porta para receber dados do Unity
unitySendPort = 52000; % Porta para enviar dados ao Unity
udpServer = udp('127.0.0.1', unityReceivePort, 'LocalPort', unitySendPort);
fopen(udpServer);

mortes = 0;
score = 0;
scorecont = 0;
speedgame = 0.00;
mortescont = 0;

% Inicializa vetores para armazenar dados para o gráfico
tempo = [];
speedgameHistory = [];
iaValueHistory = [];
mortesHistory = [];
scoreHistory = [];

% Contador de tempo para gravação
tempoAnterior = tic;
intervaloGravacao = 3; % Intervalo de 3 segundos
jogoRodando = true;
time = 0;

% Loop principal
while jogoRodando

    %recebe do arduino posição da plataforma
    while s.BytesAvailable > 0
        fread(s, s.BytesAvailable);
    end
    valorAtual = str2double(fgetl(s)); % Lê como string e converte para
número

    % Receber dados do Unity
    data = fread(udpServer, 1024, 'char');
    dataString = char(data);
    dataValues = str2double(strsplit(dataString, '!'));

    % Verificar se há elementos suficientes em dataValues
    if length(dataValues) >= 2
        mortes = dataValues(1)
        score = dataValues(2)
    else
        % Lidar com o caso em que dataValues não tem elementos suficientes
```

```

figure;

subplot(2, 2, 1);
plot(tempo, speedgameHistory);
title('speedgame ao longo do tempo');
xlabel('Tempo (s)');
ylabel('speedgame');

subplot(2, 2, 2);
plot(tempo, iaValueHistory);
title('iaValue ao longo do tempo');
xlabel('Tempo (s)');
ylabel('iaValue');

subplot(2, 2, 3);
plot(tempo, mortesHistory);
title('Mortes ao longo do tempo');
xlabel('Tempo (s)');
ylabel('Mortes');

subplot(2, 2, 4);
plot(tempo, scoreHistory);
title('Score ao longo do tempo');
xlabel('Tempo (s)');
ylabel('Score');
jogoRodando = false;
end

if (speedgame < -25 && mortes > mortescont)
    speedgame = speedgame - 3;
    mortescont = mortes
elseif (speedgame < -10 && mortes > mortescont)
    speedgame = speedgame - 4;
    mortescont = mortes
elseif (speedgame < 10 && mortes > mortescont)
    speedgame = speedgame - 5;
    mortescont = mortes
elseif (speedgame < 25 && mortes > mortescont)
    speedgame = speedgame - 10;
    mortescont = mortes
elseif (mortes > mortescont)
    speedgame = speedgame - 18;
    mortescont = mortes
end
% Calcula a IA (ajuste conforme necessário)
iaValue = (mortes * mortes * 2) / (score);

% ajuste para velocidade ótima
if (mortes == 0 && score > scorecont+2)
    if (speedgame < 45)
        speedgame = speedgame + 11;
        scorecont = score;
    elseif (speedgame < 85)
        speedgame = speedgame + 9;
        scorecont = score;
    else
        speedgame = speedgame + 7;
    end
end

```

```

        scorecont = score;
    end

    % jogando bem
    elseif (iaValue < 0.5 && score>scorecont+2)

        if (speedgame < 50)
            speedgame = speedgame + 6;
            scorecont = score;
        else
            speedgame = speedgame + 5;
            scorecont = score;
        end

    elseif (0.5 <= iaValue && iaValue < 0.65 && score>scorecont+2)

        speedgame = speedgame + 4;
        scorecont = score;

    elseif (0.65 <= iaValue && iaValue < 0.8 && score>scorecont+2)

        speedgame = speedgame + 3;
        scorecont = score;

    elseif (0.8 <= iaValue && iaValue < 1.0 && score>scorecont+2)

        speedgame = speedgame + 2;
        scorecont = score;

    elseif (iaValue > 1.0 && score>scorecont+2)

        speedgame = speedgame + 1;
        scorecont = score;

    end

    if speedgame > 125
        speedgame = 125;
    elseif speedgame < -75
        speedgame = -75;
    else
    end
    fwrite(udpServer, uint8(sprintf('%d!%d', valorAtual, speedgame)));

    if toc(tempoAnterior) >= intervaloGravacao
        % Gravação das variáveis
        time = time + intervaloGravacao;
        fprintf('Gravando: speedgame=%f, iaValue=%f, mortes=%d, score=%d\n',
speedgame, iaValue, mortes, score);

        % Adiciona dados ao histórico
        tempo = [tempo, time];
        speedgameHistory = [speedgameHistory, speedgame];
        iaValueHistory = [iaValueHistory, iaValue];
        mortesHistory = [mortesHistory, mortes];
        scoreHistory = [scoreHistory, score];
    end
end

```

```

        % Reinicia o contador de tempo
        tempoAnterior = tic;
    end

end

fclose(udpServer);
delete(udpServer);
fclose(s); % Feche a porta com o Arduino

```

Código 5: Código do Unity jogo Box Adventure, comunicação com o Malab

```

using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using UnityEngine;

public class UDPCommunication : MonoBehaviour
{
    private UdpClient udpClient;
    private IPEndPoint matlabEndPoint;
    private int unityReceivePort = 35000; // Porta para receber dados do MATLAB
    public static int mortes;
    public static int scoreplus;
    public static float BoxControl;
    public static float speedgame;
    public GameController gamecontrol;
    public Jogador jogador;
    public static UDPCommunication Instance;

    private float communicationDelay = 0.12f; // Atraso de 0.12 segundos
    private float lastCommunicationTime;

    void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }

    void Start()
    {
        udpClient = new UdpClient();
        matlabEndPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 52000); //
Endereço e porta do servidor MATLAB
        udpClient.Client.Bind(new IPEndPoint(IPAddress.Any, unityReceivePort)); //
Configurar a porta de recebimento do Unity
    }
}

```

```

}

void Update()
{
    // Adicione um atraso entre as comunicações
    if (Time.time - lastCommunicationTime < communicationDelay)
    {
        return;
    }

    // Atualize o tempo da última comunicação
    lastCommunicationTime = Time.time;

    // Resto do código de Update() permanece inalterado

    // Enviar dados para o MATLAB
    scoreplus = GameController.scoreplus;
    mortes = GameController.mortes;

    string dataToSend = $"{mortes}!{scoreplus}";
    byte[] dataBytes = Encoding.ASCII.GetBytes(dataToSend);
    udpClient.Send(dataBytes, dataBytes.Length, matlabEndPoint);

    // Receber dados do MATLAB
    IPEndPoint receivedEndPoint = new IPEndPoint(IPAddress.Any, unityReceivePort);
    byte[] receivedData = udpClient.Receive(ref receivedEndPoint);
    string receivedString = Encoding.ASCII.GetString(receivedData);

    // Debug para visualizar os dados recebidos no console do Unity
    Debug.Log("Dados recebidos do MATLAB: " + receivedString);

    // Divida a string recebida em partes usando o delimitador "!"
    string[] dataValues = receivedString.Split('!');

    // Verifique se há elementos suficientes em dataValues
    if (dataValues.Length >= 2)
    {
        BoxControl = float.Parse(dataValues[0]);
        BoxControl = -BoxControl / 175;

        speedgame = float.Parse(dataValues[1]);
        speedgame = speedgame / 100;
        Debug.Log("Dados recebidos do MATLAB:: " + BoxControl + ", " + speedgame);
        // Agora você pode usar 'BirdControl' e 'speedgame' como necessário no seu
código
    }
    else
    {
        BoxControl = 0;
        speedgame = 0;
        // Lidar com o caso em que dataValues não tem elementos suficientes
        Debug.LogError("Erro: Não há elementos suficientes em dataValues.");
    }
}
}

```

Código 6: Código do Unity jogo Space Adventure, comunicação com o Malab

```
using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using UnityEngine;

public class UDPCommunication : MonoBehaviour
{
    private UdpClient udpClient;
    private IPEndPoint matlabEndPoint;
    private int unityReceivePort = 35000; // Porta para receber dados do MATLAB
    private int numDecimalPlaces = 2;
    public float hBird;
    public static float spawnYPosition;
    public static int mortes;
    public static int scoreplus;
    public static float BirdControl;
    public static float speedgame;
    public GameControl gamecontrol;
    public Bird bird;
    public static UDPCommunication Instance;

    private float communicationDelay = 0.12f; // Atraso de 0.12 segundos
    private float lastCommunicationTime;

    void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }

    void Start()
    {
        udpClient = new UdpClient();
        matlabEndPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 52000); //
        Endereço e porta do servidor MATLAB
        udpClient.Client.Bind(new IPEndPoint(IPAddress.Any, unityReceivePort)); //
        Configurar a porta de recebimento do Unity
    }

    void Update()
    {
        // Adicione um atraso entre as comunicações
        if (Time.time - lastCommunicationTime < communicationDelay)
        {
            return;
        }

        // Atualize o tempo da última comunicação
    }
}
```

```

lastCommunicationTime = Time.time;

// Resto do código de Update() permanece inalterado

// Enviar dados para o MATLAB
scoreplus = GameController.scoreplus;
mortes = GameController.mortes;

string dataToSend = $"{mortes}!{scoreplus}";
byte[] dataBytes = Encoding.ASCII.GetBytes(dataToSend);
udpClient.Send(dataBytes, dataBytes.Length, matlabEndPoint);

// Receber dados do MATLAB
IPEndPoint receivedEndPoint = new IPEndPoint(IPAddress.Any, unityReceivePort);
byte[] receivedData = udpClient.Receive(ref receivedEndPoint);
string receivedString = Encoding.ASCII.GetString(receivedData);

// Debug para visualizar os dados recebidos no console do Unity
Debug.Log("Dados recebidos do MATLAB: " + receivedString);

// Divida a string recebida em partes usando o delimitador "!"
string[] dataValues = receivedString.Split('!');

// Verifique se há elementos suficientes em dataValues
if (dataValues.Length >= 2)
{
    BirdControl = float.Parse(dataValues[0]);
    BirdControl = BirdControl / 125;

    speedgame = float.Parse(dataValues[1]);
    speedgame = speedgame / 100;
    Debug.Log("Dados recebidos do MATLAB:: " + BirdControl + ", " +
speedgame);
    // Agora você pode usar 'BirdControl' e 'speedgame' como necessário no seu
código
}
else
{
    BirdControl = 0;
    speedgame = 0;
    // Lidar com o caso em que dataValues não tem elementos suficientes
    Debug.LogError("Erro: Não há elementos suficientes em dataValues.");
}
}
}

```

Código 7: Código do Unity jogo Bird Adventure, comunicação com o Malab

```

using System;
using System.Text;
using System.Net;
using System.Net.Sockets;
using UnityEngine;

public class UDPCommunication : MonoBehaviour
{
    private UdpClient udpClient;

```

```

private IPEndPoint matlabEndPoint;
private int unityReceivePort = 35000; // Porta para receber dados do MATLAB
private int numDecimalPlaces = 2;
public float hBird;
public static float spawnYPosition;
public static int mortes;
public static int scoreplus;
public static float BirdControl;
public static float speedgame;
public GameControl gamecontrol;
public Bird bird;
public static UDPCommunication Instance;

private float communicationDelay = 0.12f; // Atraso de 0.12 segundos
private float lastCommunicationTime;

void Awake()
{
    if (Instance == null)
    {
        Instance = this;
        DontDestroyOnLoad(gameObject);
    }
    else
    {
        Destroy(gameObject);
    }
}

void Start()
{
    udpClient = new UdpClient();
    matlabEndPoint = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 52000); //
Endereço e porta do servidor MATLAB
    udpClient.Client.Bind(new IPEndPoint(IPAddress.Any, unityReceivePort)); //
Configurar a porta de recebimento do Unity
}

void Update()
{
    // Adicione um atraso entre as comunicações
    if (Time.time - lastCommunicationTime < communicationDelay)
    {
        return;
    }

    // Atualize o tempo da última comunicação
    lastCommunicationTime = Time.time;

    // Resto do código de Update() permanece inalterado

    // Enviar dados para o MATLAB
    scoreplus = GameControl.scoreplus;
    mortes = GameControl.mortes;

    string dataToSend = $"{mortes}!{scoreplus}";
    byte[] dataBytes = Encoding.ASCII.GetBytes(dataToSend);
    udpClient.Send(dataBytes, dataBytes.Length, matlabEndPoint);

    // Receber dados do MATLAB

```

```

IPEndPoint receivedEndPoint = new IPEndPoint(IPAddress.Any, unityReceivePort);
byte[] receivedData = udpClient.Receive(ref receivedEndPoint);
string receivedString = Encoding.ASCII.GetString(receivedData);

// Debug para visualizar os dados recebidos no console do Unity
Debug.Log("Dados recebidos do MATLAB: " + receivedString);

// Divida a string recebida em partes usando o delimitador "!"
string[] dataValues = receivedString.Split('!');

// Verifique se há elementos suficientes em dataValues
if (dataValues.Length >= 2)
{
    BirdControl = float.Parse(dataValues[0]);
    BirdControl = BirdControl / 100;

    speedgame = float.Parse(dataValues[1]);
    speedgame = speedgame / 100;
    Debug.Log("Dados recebidos do MATLAB:: " + BirdControl + ", " +
speedgame);
    // Agora você pode usar 'BirdControl' e 'speedgame' como necessário no seu
código
}
else
{
    BirdControl = 0;
    speedgame = 0;
    // Lidar com o caso em que dataValues não tem elementos suficientes
    Debug.LogError("Erro: Não há elementos suficientes em dataValues.");
}
}
}

```

Apêndice B – Códigos dos jogos

Código 8: Código de permanência do gameobject que carrega a comunicação independente de resetar a cena em todos jogos

```
// PersistentObject.cs
using UnityEngine;

public class PersistentObject : MonoBehaviour
{
    private static PersistentObject instance;

    private void Awake()
    {
        if (instance == null)
        {
            instance = this;
            DontDestroyOnLoad(gameObject);
        }
        else
        {
            Destroy(gameObject);
        }
    }
}
```

Código 9: Código do jogo Box Adventure da caixa controlada pelo jogador

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Jogador : MonoBehaviour
{
    public float vel = 4f; // velocidade
    public static float boxControl;

    private GameController gameController;
    private Rigidbody rb;
    public UDPCommunication udp;

    // Start is called before the first frame update
    void Start()
    {
        rb = GetComponent<Rigidbody>();
        gameController = GameObject.FindObjectOfType<GameController>();
    }

    // Update is called once per frame
    void Update()
    {

```

```

        // float movimentoHorizontal = Input.GetAxis("Horizontal"); // Obtém a entrada
horizontal (seta esquerda/direita)
        boxControl = UDPCommunication.BoxControl;
        // Calcula o vetor de movimento
        Vector3 movimento = new Vector3(boxControl, 0.0f, 0.0f) * vel *
Time.deltaTime;

        // Aplica o movimento ao objeto do personagem
        transform.Translate(movimento);
    }

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Perigo"))
        {
            gameController.ReiniciarCena();
        }
    }
}

```

Código 10: Código do jogo Box Adventure da caixa obstáculo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Perigo : MonoBehaviour
{
    private GameController gameController;

    private void Start()
    {
        gameController = GameObject.FindObjectOfType<GameController>();
    }

    private void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Ilha"))
        {
            Destroy(gameObject);
        }
    }
}

```

Código 11: Código do jogo Box Adventure da mecânica do jogo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

```

```

public class GameController : MonoBehaviour
{
    public GameObject jogadorPrefab;
    public GameObject perigofab;
    public UDPCommunication udp;
    public int nperigo;
    public int nperigomin;
    public static float speedgame;
    public int score = 0;
    public static int scoreplus = 1;
    public static int mortes;
    public Text scoreText;

    void Start()
    {
        // Procura o objeto Text na cena se não estiver associado no Inspector.
        if (scoreText == null)
        {
            scoreText = GameObject.Find("ScoreText").GetComponent<Text>();
        }

        // Configura a posição do texto na tela (parte superior central).
        RectTransform textRectTransform = scoreText.GetComponent<RectTransform>();
        textRectTransform.anchorMin = new Vector2(0.5f, 1f);
        textRectTransform.anchorMax = new Vector2(0.5f, 1f);
        textRectTransform.pivot = new Vector2(0.5f, 1f);
        textRectTransform.anchoredPosition = new Vector2(0f, -10f); // Ajuste conforme
necessário.

        scoreText.text = "Score: " + score;
        SpawnJogador();
        StartCoroutine(SpawnPerigo());
    }

    void SpawnJogador()
    {
        Instantiate(jogadorPrefab, new Vector3(0, 0, 0), Quaternion.identity);
    }

    private IEnumerator SpawnPerigo()
    {
        while (true)
        {
            yield return new WaitForSeconds(3f);
            speedgame = UDPCommunication.speedgame;
            if (speedgame >= 0.5f)
            {
                nperigo = 5;
                nperigomin = 2;
            }
            else if (speedgame <= 0.1f)
            {
                nperigo = 3;
                nperigomin = 1;
            }
            else
            {
                nperigo = 4;
                nperigomin = 2;
            }
        }
    }
}

```

```

        var qtdspwn = Random.Range(nperigomin, nperigo);

        for (int i = 0; i < qtdspwn; i++)
        {
            var x = Random.Range(-10, 10);
            var drag = Random.Range(1f-speedgame, 2f-speedgame);

            Instantiate(perigofab, new Vector3(x, 13, 0),
Quaternion.identity).GetComponent<Rigidbody>().drag = drag;
        }

        score++;
        scoreplus++;
        scoreText.text = "Score: " + score;
    }
}

public void ReiniciarCena()
{
    mortes++;
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
}
}

```

Código 12: Código do jogo Space Adventure da linha de obstáculos, naves alienígenas e no Bird Adventure da coluna de obstáculo, árvores

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Column : MonoBehaviour
{
    private void OnTriggerEnter2D(Collider2D other)
    {
        if (other.GetComponent<Bird>() != null)
        {
            GameControl.instance.Scored();
        }
    }
}

```

Código 13: Código do jogo Space Adventure e Bird Adventure para o surgimento do novo obstáculo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ColumnPool : MonoBehaviour

```

```

{
    public static ColumnPool instance;
    public GameObject columnPrefab;
    public float spawnRate = 8f;

    private GameObject column;
    private Vector2 objectPoolPosition = new Vector2(-15f, -25f);
    private float initialYPosition;

    void Awake()
    {
        instance = this;
    }

    void Start()
    {
        column = (GameObject)Instantiate(columnPrefab, new Vector2(8f, 0f),
Quaternion.identity);
        initialYPosition = column.transform.position.y;
    }

    public void SpawnColumn(float spawnYPosition, float spawnXPosition)
    {
        column.transform.position = new Vector2(spawnXPosition, spawnYPosition);
    }

    public void ResetColumns()
    {
        column.transform.position = new Vector2(column.transform.position.x + 8f,
initialYPosition);
    }
}

```

Código 14: Código do jogo Space Adventure e Bird Adventure para as músicas de fundo de cada jogo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SoundTrackEnvironment : MonoBehaviour {

    AudioSource soundtrack;

    // Use this for initialization
    void Start () {
        soundtrack.GetComponent<AudioSource>();
    }

    // Update is called once per frame
    void Update () {

    }

}

```

Código 15: Código do jogo Space Adventure e Bird Adventure das telas iniciais de cada jogo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;

public class startGame : MonoBehaviour {

    //public AudioClip selectClip;
    //public AudioSource selectSource;

    private AsyncOperation asyncOperation;
    public GameObject[] UI;
    public Text txtProgress;

    // Use this for initialization
    void Start () {
        //selectSource.clip = selectClip;
    }

    // Update is called once per frame
    private bool isLoadingLevel = false;

    void Update()
    {
        float verticalInput = Input.GetAxis("Vertical");

        if (verticalInput != 0 && !isLoadingLevel)
        {
            isLoadingLevel = true; // Indica que o nível está sendo carregado
            LoadLevel();
        }
    }

    public void LoadLevel()
    {
        UI[0].SetActive(false);
        UI[1].SetActive(true);
        StartCoroutine(LoadScene());
    }

    IEnumerator LoadScene()
    {
        yield return null;

        //Begin to load the Scene you specify
        asyncOperation = SceneManager.LoadSceneAsync(1);

        //Don't let the Scene activate until you allow it to
        asyncOperation.allowSceneActivation = false;

        //When the load is still in progress, output the Text and progress bar
        while (!asyncOperation.isDone)
        {
            //Output the current progress
            txtProgress.text = (asyncOperation.progress * 100).ToString("0") + "%";
        }
    }
}
```

```

        if (asyncOperation.progress >= 0.9f)
        {
            txtProgress.text = "100%";
            //Active Scene
            asyncOperation.allowSceneActivation = true;
        }

        yield return null;
    }
}
}

```

Código 16: Código do jogo Space Adventure e Bird Adventure da movimentação dos obstáculos em direção ao personagem de cada jogo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ScrollingObject : MonoBehaviour
{
    private Rigidbody2D rb;
    private GameController gameControl;
    public bool isRising;
    public bool notMovableY;

    private bool cont = false;

    // Use this for initialization
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        // Encontrar o objeto que tem o script GameController.cs
        GameObject gameControlObject = GameObject.FindWithTag("GameController");

        // Verificar se encontramos o objeto e o componente GameController
        if (gameControlObject != null)
        {
            gameControl = gameControlObject.GetComponent<GameController>();
        }
    }

    // Update is called once per frame
    void Update()
    {
        float verticalInput = Input.GetAxis("Vertical");

        if (verticalInput != 0)
        {
            gameControl.SelectPlayer();
            cont = true;
        }

        if (cont == true)
        {

```

```

        rb.velocity = new Vector2(GameControl.instance.scrollSpeed, 0);
    }

    if (GameControl.instance.gameOver == true)
    {
        rb.velocity = new Vector2(0, 0);
    }
}
}

```

Código 17: Código do jogo Space Adventure controle da nave

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Bird : MonoBehaviour
{
    public GameObject txtStart;
    //public GameObject feather;
    public GameObject wave;

    Vector2 pos;

    private bool isDead = false;
    private Rigidbody2D rb;
    private Animator anim;

    //public AudioClip FlyClip;
    //public AudioSource FlySource;

    public Text txtTimer;
    public bool startTimer;
    private float timer = 0.0f;
    public float timeLimit;
    public static float birdControl;
    public UDPCommunication udp;

    // Use this for initialization
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();

        birdControl = 0;
    }

    // Update is called once per frame
    void Update()
    {
        if (isDead == false)
        {
            birdControl = UDPCommunication.BirdControl;

```

```

//int birdControl = udp.BirdControl;
//Debug.Log("Valor de BirdControl: " + birdControl);
txtStart.SetActive(false);
float verticalInput = Input.GetAxisRaw("Vertical");

rb.velocity = new Vector2(rb.velocity.x, birdControl);
anim.SetTrigger("Fly"); // Aciona a animação de voo

//Mantém personagem em 0 no eixo X, em qualquer situação
pos = transform.position;
pos.x = -1;
transform.position = pos;
gameObject.GetComponent<Transform>();

//sprite voar
anim.SetTrigger("Fly");

//penas
//GameObject feathers = Instantiate(feather);
Vector3 positions = transform.position + new Vector3(0, 0.9f, 0);
//feathers.transform.position = positions;

//som voar
//FlySource.Play();

if (!startTimer)
    startTimer = true;

transform.rotation = Quaternion.Euler(0, 0, rb.velocity.y * 3); //rotação

if (startTimer)
    timer += Time.deltaTime;

txtTimer.text = timer.ToString("0");

if (timer >= timeLimit)
    GameController.instance.scrollMovable = true;
}
}

private void OnCollisionEnter2D(Collision2D collision)
{
    if (gameObject.transform.localScale.x == 2f)
    {
        if (collision.collider.tag == "Obstacle")
        {
            var posit = transform.position;
            posit.x -= 8f;
            collision.transform.position = posit;
            GameController.instance.Scored();
        }
    }
    else
    {
        rb.velocity = new Vector2(0, 0);
        isDead = true;
        anim.SetTrigger("Die");
        GameController.instance.Died();
    }
}

```

```

        GameControl.instance.RestartGame();
    }
}

```

Código 18: Código do jogo Space Adventure controle da mecânica do jogo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using Random = UnityEngine.Random;
using System.Net.Sockets;
using System.Text;
using System.Net;

[Serializable]
public class PlayerData
{
    public int score;
    public int charIndex;
}

public class GameControl : MonoBehaviour
{
    public static GameControl instance;
    public GameObject txtGameOver;
    public Color targetColor;
    public Text txtScore;
    public bool gameOver = false;
    public float scrollSpeed = -1f;

    private int lastScore;
    public Image[] selectImages;
    public Color deselectColor;
    public GameObject[] characters;
    private int charIndex;
    public GameObject[] HUDs;
    private string filePath;
    public Text txtLastScore;
    public Image imgLastChar;
    public GameObject uiLast;
    public bool scrollMovable;

    private float timeSpawn = 0f;
    private float spawnXPosition = 5f;
    public float spawnRate = 6f;
    public static float spawnYPosition;
    public static int mortes;
    public static int score;
    public static int scoreplus = 1;
}

```

```

public UDPCommunication udp;
public static float speedgame;

void Start()
{
    score = 0;
    spawnYPosition = 0;
    txtScore.text = "0";
    spawnXPosition = 1f;
    ColumnPool.instance.enabled = true;
}

void Awake()
{
    if (instance == null)
    {
        instance = this;
    }
    else if (instance != this)
    {
        Destroy(gameObject);
    }

    filePath = Application.persistentDataPath + "/playerInfo.dat";

    if (File.Exists(filePath))
    {
        Load();
        LastMatch();
    }
}

public void Save()
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream file = File.Create(filePath);

    PlayerData data = new PlayerData();
    data.score = score;

    bf.Serialize(file, data);
    file.Close();
}

private void Load()
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream file = File.Open(filePath, FileMode.Open);

    PlayerData data = (PlayerData)bf.Deserialize(file);
    file.Close();

    lastScore = data.score;
    charIndex = data.charIndex;
}

void Update()
{
    speedgame = UDPCommunication.speedgame;
    scrollSpeed = -1.5f - speedgame;
}

```

```

public void Scored()
{
    if (gameOver)
    {
        return;
    }
    score++;
    scoreplus++;
    txtScore.text = score.ToString();
    spawnXPosition = 5f;
    int spawnYPosition = Random.Range(-1, 4);
    //Debug.Log("Spawn Y Position: " + spawnYPosition);
    ColumnPool.instance.SpawnColumn(spawnYPosition, spawnXPosition);
}

public void Died()
{
    txtGameOver.SetActive(true);
    gameOver = true;
    spawnYPosition = 0;
    Save();
}

private void LastMatch()
{
    if (!uiLast.activeSelf)
        uiLast.SetActive(true);

    txtLastScore.text = lastScore.ToString();
}

public void SelectPlayer()
{
    HUDs[0].SetActive(false);
    HUDs[1].SetActive(true);
}

public void SetSelectedChar(int index)
{
    charIndex = index;
}

public void RestartGame()
{
    mortes++; // Incrementa a contagem de mortes.
    //Debug.Log("Mortes: " + mortes); // Imprime a contagem de mortes no console.
    spawnYPosition = 0;
    SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
}
}

```

Código 19: Código do jogo Space Adventure reinicialização da tela de fundo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;

```

```

public class Scrolling : MonoBehaviour
{
    public float speed;
    public Renderer background;
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        background.material.mainTextureOffset += new Vector2(speed * Time.deltaTime,
0);
    }
}

```

Código 20: Código do jogo Bird Adventure controle do pássaro

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Bird : MonoBehaviour
{
    public GameObject txtStart;
    //public GameObject feather;
    public GameObject wave;

    Vector2 pos;

    private bool isDead = false;
    private Rigidbody2D rb;
    private Animator anim;

    //public AudioClip FlyClip;
    //public AudioSource FlySource;

    public Text txtTimer;
    public bool startTimer;
    private float timer = 0.0f;
    public float timeLimit;
    public static float birdControl;
    public UDPCommunication udp;

    // Use this for initialization
    void Start()
    {
        rb = GetComponent<Rigidbody2D>();
        anim = GetComponent<Animator>();

        birdControl = 0;
    }
}

```

```

// Update is called once per frame
void Update()
{
    if (isDead == false)
    {
        birdControl = UDPCommunication.BirdControl;
        //int birdControl = udp.BirdControl;
        //Debug.Log("Valor de BirdControl: " + birdControl);
        txtStart.SetActive(false);
        float verticalInput = Input.GetAxisRaw("Vertical");

        rb.velocity = new Vector2(rb.velocity.x, birdControl);
        anim.SetTrigger("Fly"); // Aciona a animação de voo

        //Mantém personagem em 0 no eixo X, em qualquer situação
        pos = transform.position;
        pos.x = 0;
        transform.position = pos;
        gameObject.GetComponent<Transform>();

        //sprite voar
        anim.SetTrigger("Fly");

        //penas
        //GameObject feathers = Instantiate(feather);
        Vector3 positions = transform.position + new Vector3(0, 0.9f, 0);
        //feathers.transform.position = positions;

        //som voar
        //FlySource.Play();

        if (!startTimer)
            startTimer = true;

        transform.rotation = Quaternion.Euler(0, 0, rb.velocity.y * 3); //rotação

        if (startTimer)
            timer += Time.deltaTime;

        txtTimer.text = timer.ToString("0");

        if (timer >= timeLimit)
            GameController.instance.scrollMovable = true;
    }
}

private void OnCollisionEnter2D(Collision2D collision)
{
    if (gameObject.transform.localScale.x == 2f)
    {
        if (collision.collider.tag == "Obstacle")
        {
            var posit = transform.position;
            posit.x -= 8f;
            collision.transform.position = posit;
            GameController.instance.Scored();
        }
    }
}

```

```

        }
    }
    else
    {
        rb.velocity = new Vector2(0, 0);
        isDead = true;
        anim.SetTrigger("Die");
        GameControl.instance.Died();
        GameControl.instance.RestartGame();
    }
}
}

```

Código 21: Código do jogo Bird Adventure controle da mecânica do jogo

```

using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;
using System;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;
using Random = UnityEngine.Random;
using System.Net.Sockets;
using System.Text;
using System.Net;

[Serializable]
public class PlayerData
{
    public int score;
    public int charIndex;
}

public class GameControl : MonoBehaviour
{
    public static GameControl instance;
    public GameObject txtGameOver;
    public Color targetColor;
    public Text txtScore;
    public bool gameOver = false;
    public float scrollSpeed = -1.5f;

    private int lastScore;
    public Image[] selectImages;
    public Color deselectColor;
    public GameObject[] characters;
    private int charIndex;
    public GameObject[] HUDs;
    private string filePath;
    public Text txtLastScore;
    public Image imgLastChar;
    public GameObject uiLast;
    public bool scrollMovable;
}

```

```

private float timeSpawn = 0f;
private float spawnXPosition = 8f;
public float spawnRate = 8f;
public static float spawnYPosition;
public static int mortes;
public static int score;
public static int scoreplus = 1;
public UDPCommunication udp;
public static float speedgame;

void Start()
{
    score = 0;
    spawnYPosition = 0;
    txtScore.text = "0";
    spawnXPosition = 1f;
    ColumnPool.instance.enabled = true;
}

void Awake()
{
    if (instance == null)
    {
        instance = this;
    }
    else if (instance != this)
    {
        Destroy(gameObject);
    }

    filePath = Application.persistentDataPath + "/playerInfo.dat";

    if (File.Exists(filePath))
    {
        Load();
        LastMatch();
    }
}

public void Save()
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream file = File.Create(filePath);

    PlayerData data = new PlayerData();
    data.score = score;

    bf.Serialize(file, data);
    file.Close();
}

private void Load()
{
    BinaryFormatter bf = new BinaryFormatter();
    FileStream file = File.Open(filePath, FileMode.Open);

    PlayerData data = (PlayerData)bf.Deserialize(file);
    file.Close();

    lastScore = data.score;
}

```

```

        charIndex = data.charIndex;
    }

    void Update()
    {
        speedgame = UDPCommunication.speedgame;
        scrollSpeed = -1.5f - speedgame;
    }

    public void Scored()
    {
        if (gameOver)
        {
            return;
        }
        score++;
        scoreplus++;
        txtScore.text = score.ToString();
        spawnXPosition = 8f;
        int spawnYPosition = Random.Range(-1, 5);
        //Debug.Log("Spawn Y Position: " + spawnYPosition);
        ColumnPool.instance.SpawnColumn(spawnYPosition, spawnXPosition);
    }

    public void Died()
    {
        txtGameOver.SetActive(true);
        gameOver = true;
        spawnYPosition = 0;
        Save();
    }

    private void LastMatch()
    {
        if (!uiLast.activeSelf)
            uiLast.SetActive(true);

        txtLastScore.text = lastScore.ToString();
    }

    public void SelectPlayer()
    {
        HUDs[0].SetActive(false);
        HUDs[1].SetActive(true);
    }

    public void SetSelectedChar(int index)
    {
        charIndex = index;
    }

    public void RestartGame()
    {
        mortes++; // Incrementa a contagem de mortes.
        //Debug.Log("Mortes: " + mortes); // Imprime a contagem de mortes no console.
        spawnYPosition = 0;
        SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex);
    }
}

```

Código 22: Código do jogo Bird Adventure reinicialização da tela de fundo

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class RepeatingBackground : MonoBehaviour {

    private BoxCollider2D groundCollider;
    private float groundHorizontal;

    // Use this for initialization
    void Start () {
        groundCollider = GetComponent<BoxCollider2D>();
        groundHorizontal = groundCollider.size.x;
    }

    // Update is called once per frame
    void Update () {
        if(transform.position.x < -groundHorizontal)
        {
            Reposition();
        }
    }

    private void Reposition()
    {
        Vector2 groundMove = new Vector2(groundHorizontal * 1.977f, 0);
        transform.position = (Vector2)transform.position + groundMove;
    }
}
```

ANEXO 1 – Datasheet do encoder rotativo E40S6 - 1000

E40 Series

Diameter ø40mm Shaft type/Hollow type/Built-in type Incremental Rotary Encoder

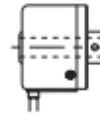
■ Features

- 12-24VDC power supply of line driver output(Line-up)
- Easy installation at narrow space
- Small moment of inertia
- Power supply : 5VDC, 12-24VDC ±5%
- Various output types

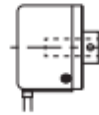
⚠ Please read "Caution for your safety" in operation manual before using.



E40S Series



E40H Series



E40HB Series

■ Ordering information

E40	H	8	—	5000	—	3	—	N	—	24	—	
-----	---	---	---	------	---	---	---	---	---	----	---	--

Series	Shaft type	Hollow type	Pulse/1Revolution	Output phase	Output	Power supply	Cable
S: Shaft type	External	Inner	Series	2: A, B	T: Totem pole output	5 : 5VDC ±5% 24: 12-24VDC ±5%	No mark: Cable type C: Connector cable type(※)
H: Hollow type	6: ø6mm	8: ø8mm		3: A, B, Z	N: NPN open collector output		
HB: Hollow built-in type	8: ø8mm	10: ø10mm		4: A, $\bar{A}$, B, $\bar{B}$	V: Voltage output		
		12: ø12mm		6: A, $\bar{A}$, B, $\bar{B}$, Z, Z	L: Line driver output		

※Standard: E40S6-PULSE-3-N-24, E40H8-PULSE-3-N-24 ※Standard: A, B, Z

※Cable length : 250mm

■ Specifications

Item	Diameter ø40mm shaft/hollow shaft/hollow built-in type of incremental rotary encoder	
Resolution(P/R) ^{※1}	*1, *2, *5, 10, *12, 15, 20, 23, 25, 30, 35, 40, 45, 50, 60, 75, 100, 120, 150, 192, 200, 240, 250, 256, 300, 360, 400, 500, 512, 600, 800, 1000, 1024, 1200, 1500, 1800, 2000, 2048, 2500, 3000, 3600, 5000 (Not indicated resolution is customizable.)	
Electrical specification	Output phase	A, B, Z phase(Line driver A, $\bar{A}$, B, $\bar{B}$, Z, Z phase)
	Phase difference of output	Phase difference between A and B : $\frac{T}{4} \pm \frac{T}{8}$ (T=1 cycle of A phase)
	Control output	Totem pole output
		NPN open collector output
		Voltage output
		Line driver output
	Response time (Rise/Fall)	Totem pole output
		NPN open collector output
		Voltage output
		Line driver output
Mechanical specification	Max. Response frequency	300kHz
	Power supply	* 5VDC ±5%(Ripple P-P : Max. 5%) * 12-24VDC ±5%(Ripple P-P : Max. 5%)
	Current consumption	Max. 80mA(disconnection of the load), Line driver output : Max. 50mA(disconnection of the load)
	Insulation resistance	Min. 100MΩ(at 500VDC megger between all terminals and case)
	Dielectric strength	750VAC 50/60Hz for 1 minute(Between all terminals and case)
	Connection	Cable type, 250mm connector cable type
	Starting torque	Shaft type : Max. 40gf·cm(0.004N·m), Hole type : Max. 50gf·cm(0.005N·m)
	Moment of inertia	Max. 40g·cm ² (4×10 ⁻⁶ kg·m ²)
	Shaft loading	Radial : Max. 2kgf, Thrust : Max. 1kgf
	Max. allowable revolution ^{※2}	5000rpm
Vibration	1.5mm amplitude or 300m/s ² at frequency of 10 to 55Hz(for 1 min.) in each of X, Y, Z directions for 2 hours	
Shock	Approx. Max. 50G	
Environment	Ambient temperature	-10 to 70°C, storage : -25 to 85°C
	Ambient humidity	35 to 85%RH, storage : 35 to 90%RH
Protection	IP50(IEC standard) ※Option type is available for IP64 (IEC standard).	
Cable	ø5, 5-wire, Length : 2m, Shield cable(Line driver output : ø5, 8-wire) (AWG24, Core diameter: 0.08, Number of cores: 40, Insulator out diameter: ø1)	
Accessory	* Shaft type : ø6mm coupling standard, ø8mm coupling(Sold separately) * Hole type : Bracket	
Approval	CE (Except line driver output)	
Unit weight	Approx. 160g	

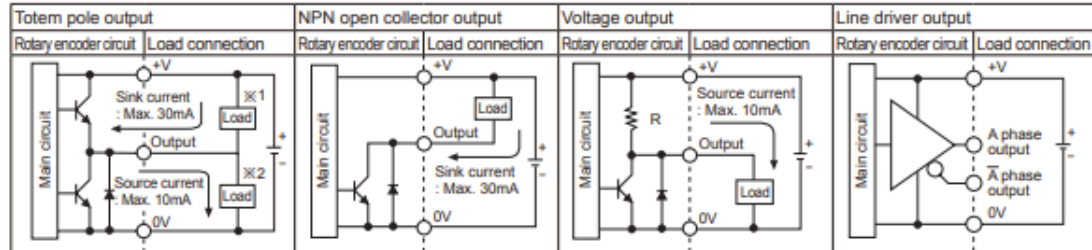
※1: "V" pulse is only for A, B phase(Line Driver output is for A, $\bar{A}$, B, $\bar{B}$ phase)

※2: Make sure that Max response revolution should be lower than or equal to max. allowable revolution when selecting the resolution.

$$[\text{Max. response revolution(rpm)}] = \frac{\text{Max. response frequency}}{\text{Resolution}} \times 60 \text{ sec} \quad \text{※Environment resistance is rated at no freezing or condensation.}$$

Incremental ø40mm Shaft/Hollow Shaft/Built-in type

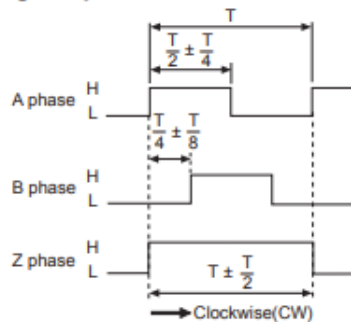
Control output diagram



- Totem pole output type can be used for NPN open collector output type(※1) or Voltage output type(※2).
- All output circuits of A, B, Z phase are the same. (Line driver output is A, $\bar{A}$, B, $\bar{B}$, Z, $\bar{Z}$)

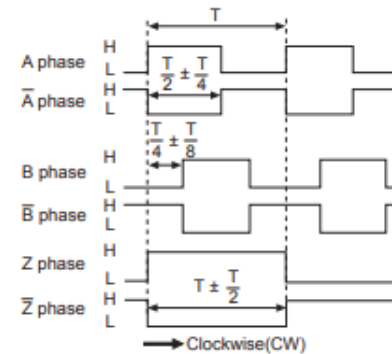
Output waveform

- Totem pole output / NPN open collector output / Voltage output



- ※Z phase output is option.
- ※CW : Right turn as from the shaft.

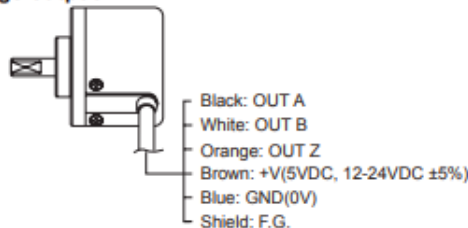
- Line driver output



Connections

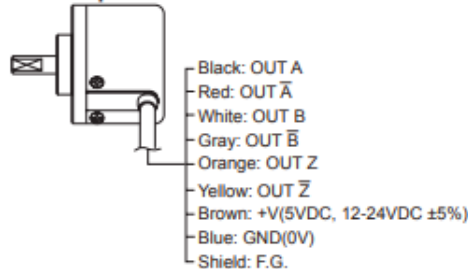
Cable type

- Totem pole output / NPN open collector output / Voltage output



- ※Unused wires must be insulated.
- ※The metal case and shield wire of encoder should be grounded(F.G.).

- Line driver output



Connector cable type

- Totem pole output / NPN open collector output / Voltage output
- Line driver output



Totem pole output/ NPN open collector output/ Voltage output			Line driver output		
Pin No	Function	Cable color	Pin No	Function	Cable color
①	OUT A	Black	①	OUT A	Black
②	OUT B	White	②	OUT $\bar{A}$	Red
③	OUT Z	Orange	③	+V	Brown
④	+V	Brown	④	GND	Blue
⑤	GND	Blue	⑤	OUT B	White
⑥	F.G.	Shield	⑥	OUT $\bar{B}$	Gray
			⑦	OUT Z	Orange
			⑧	OUT $\bar{Z}$	Yellow
			⑨	F.G.	Shield

- ※F.G.(Field Ground): It should be grounded separately.

(A)	Photo electric sensor
(B)	Fiber optic sensor
(C)	Door/Area sensor
(D)	Proximity sensor
(E)	Pressure sensor
(F)	Rotary encoder
(G)	Connector/Socket
(H)	Temp. controller
(I)	SRP/Power controller
(J)	Counter
(K)	Timer
(L)	Panel meter
(M)	Tachol/Speed/Pulse meter
(N)	Display unit
(O)	Sensor controller
(P)	Switching mode power supply
(Q)	Stepper motor& Driver&Controller
(R)	Graphic/Logic panel
(S)	Field network device
(T)	Software
(U)	Other

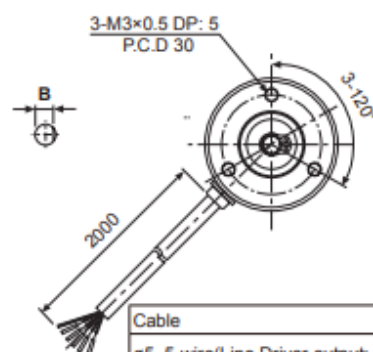
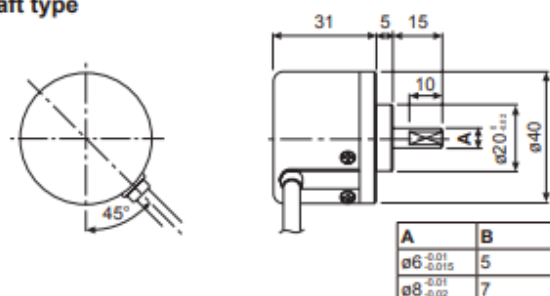
E40 Series

■ Dimensions

(unit: mm)

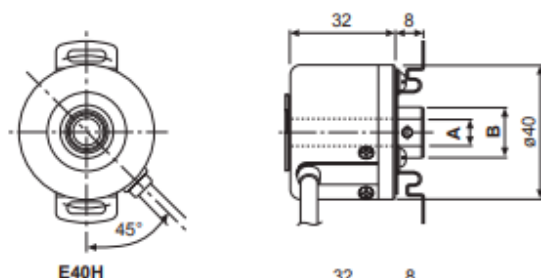
◎ Cable type

● Shaft type

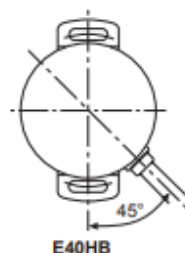


Cable
ø5, 5-wire(Line Driver output: 8-wire),
Length:2000, Shield cable

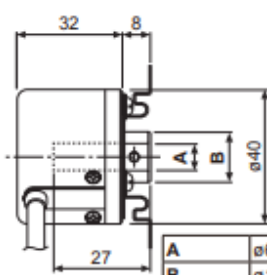
● Hollow shaft / Hollow shaft built-in type



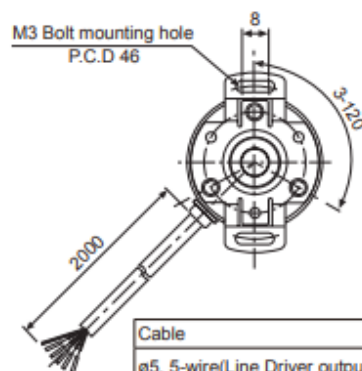
E40H



E40HB

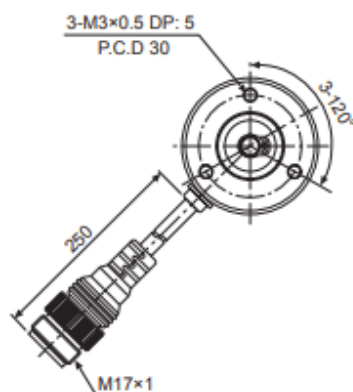


A	ø6	ø8	ø10	ø12
B	ø15	ø17		
Tolerance	$+0.015$ S			



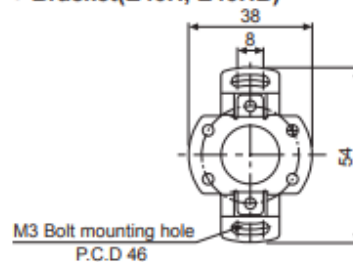
Cable
ø5, 5-wire(Line Driver output: 8-wire),
Length: 2000, Shield cable

◎ Connector cable type



※Connector cable is customizable and refer to the G-10 for specifications.

● Bracket(E40H, E40HB)

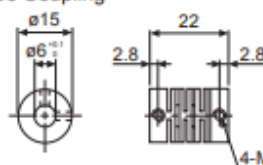


M3 Bolt mounting hole
P.C.D 46

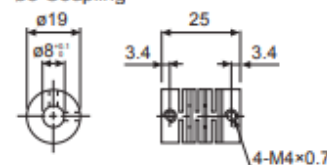


● Coupling(E40S)

● ø6 Coupling



● ø8 Coupling



- Parallel misalignment : Max. 0.25mm
- Angular misalignment: Max. 5°
- End-play: Max. 0.2mm

- ※For parallel misalignment, angular misalignment, end-play terms, refer to the F-78 page.
- ※For flexible coupling(ERB Series) information, refer to the F-71 page.