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

Aprendizado por Demonstração Aplicado em uma Plataforma de Robótica Móvel usando Redes Neurais Recorrentes

Autor: Luiz Henrique Nunes de Oliveira
Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda
Coorientador: M.Sc Mario Andrés Pastrana Triana

Brasília, DF
2023

Luiz Henrique Nunes de Oliveira

Aprendizado por Demonstração Aplicado em uma Plataforma de Robótica Móvel usando Redes Neurais Recorrentes

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

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Coorientador: M.Sc Mario Andrés Pastrana Triana

Brasília, DF

2023

Luiz Henrique Nunes de Oliveira

Aprendizado por Demonstração Aplicado em uma Plataforma de Robótica Móvel usando Redes Neurais Recorrentes/ Luiz Henrique Nunes de Oliveira. – Brasília, DF, 2023-

66 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Daniel Mauricio Muñoz Arboleda

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2023.

1. Aprendizado por demonstração. 2. Redes neurais recorrentes. I. Prof. Dr. Daniel Mauricio Muñoz Arboleda. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Aprendizado por Demonstração Aplicado em uma Plataforma de Robótica Móvel usando Redes Neurais Recorrentes

CDU 02:141:005.6

Luiz Henrique Nunes de Oliveira

Aprendizado por Demonstração Aplicado em uma Plataforma de Robótica Móvel usando Redes Neurais Recorrentes

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

Trabalho aprovado. Brasília, DF, 25 de julho de 2023 – Data da aprovação do trabalho:

Prof. Dr. Daniel Mauricio Muñoz Arboleda
Orientador

Prof. Dr. Carlos Humberto Llanos Quintero (ENM/UnB)
Convidado 1

Prof. Dr. Roberto de Souza Baptista (FGA/UnB)
Convidado 2

Brasília, DF
2023

Este trabalho é dedicado às crianças adultas que,
quando pequenas, sonharam em se tornar engenheiras.

Agradecimentos

Aos meus orientadores, Prof. Dr. Daniel Mauricio Muñoz Arboleda e M.Sc Mario Andrés Pastrana Triana, pela paciência e pela oportunidade de tê-los como meus orientadores. A ajuda e o carinho de vocês para com este projeto, e para que tudo fluísse de maneira leve e com êxito, não seriam possíveis sem vocês.

Ao meu orientador de estágio, Prof. Dr. Wellington Avelino do Amaral, agradeço pela oportunidade de ter sido introduzido a esse mundo da microeletrônica e pela confiança no meu trabalho neste meio. Também reservo esse espaço para os colegas que me ajudaram nessa reta final: Erick Rollemberg, Mateus Matchuano, Jose Mendoza, Santiago Guzmán. A ajuda e o companheirismo no LabMicro foram essenciais. Agradeço também aos meus colegas do Projeto Cedro e do Projeto Jambo, em breve estaremos trabalhando juntos novamente.

Por fim, gostaria de agradecer à minha família pelo apoio e pelo carinho, especialmente à minha amada mãe, Helane Cristina, por sempre me apoiar e me incentivar nos piores momentos. Ao meu pai, Luiz Antônio, pelos conselhos e ensinamentos nesta caminhada tão árdua. E à minha namorada, Ágatha Helene, que esteve comigo em todos os momentos, me acompanhou em cada etapa, me cobrou, me apoiou e incentivou a continuar. Obrigado pelo carinho e apoio. Sem vocês, essa conquista seria muito mais difícil.

Resumo

O aprendizado por demonstração é utilizado para ensinar robôs a fazerem tarefas complexas e viabiliza a programação de robôs móveis. Neste trabalho será utilizada uma plataforma robótica móvel previamente desenvolvida, chamada de robô MARIA, que faz uso da metodologia de aprendizado por demonstração a partir da composição de comportamentos simples, conhecidos como micro-comportamentos. Em particular, este trabalho visa explorar a capacidade de redes neurais artificiais recorrentes permitindo que o controlador neural use as saídas passadas, evitando mudanças bruscas entre os micro-comportamentos ensinados. Como resultados, um ambiente de simulação foi desenvolvido no intuito de validar a metodologia proposta para a aprendizagem de micro-comportamentos e cenários desconhecidos foram usados para testar a composição de micro-comportamentos. Adicionalmente, uma arquitetura de hardware da rede neural recorrente foi mapeada em um SoC FPGA e, através da técnica de *hardware-in-the-loop* (HIL), foi integrada ao simulador permitindo comparar numericamente o desempenho do controlador neural em software e em hardware. E por fim foram realizados testes físicos com a nova topologia de rede neural para os micro-comportamentos.

Palavras-chave: Aprendizado por demonstração. Robótica móvel. Redes neurais recorrentes. FPGA.

Abstract

Demonstration learning is used to teach robots to perform complex tasks and enables the programming of mobile robots. In this work, a previously developed mobile robotic platform called MARIA will be used, which employs the methodology of demonstration learning through the composition of simple behaviors known as micro-behaviors. Specifically, this work aims to explore the capability of recurrent artificial neural networks, allowing the neural controller to use past outputs, thus avoiding abrupt changes between the taught micro-behaviors. As a result, a simulation environment was developed to validate the proposed methodology for learning micro-behaviors, and unknown scenarios were used to test the composition of micro-behaviors. Additionally, a hardware architecture of the recurrent neural network was mapped onto an SoC FPGA, and through the hardware-in-the-loop (HIL) technique, it was integrated into the simulator, allowing for a numerical comparison of the performance of the neural controller in software and hardware. Finally, physical tests were conducted with the new neural network topology for the micro-behaviors.

Key-words: Learning from demonstration. Mobile robots. Recurrent neural networks. FPGA.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BRAM	*Block Random Acess Memory*
FF	*Flip-Flop*
FPGA	*Field Programmable Gate Arrays*
HIL	*Hardware in the loop*
LfD	*Learning from demonstration*
LSTM	*Long short term memory*
LUT	*Look-Up Table*
MC	*Micro comportamento*
MIMO	*Multi-Input, Multi-Output*
MSE	*Mean Squared Error*
PCI	Placa de circuito impresso
PI	*Proportional-Integral controller*
PSO	*Particle Swarm Optmization*
RNA	Rede Neural Artificial
RNACC	Rede Neural Artificial Completamente Conectada
RNARD	Rede Neural Artificial Recorrente
SLP	*Single Layer Perceptron*
SoC	*Systems on Chip*
TCC	Trabalho de conclusão de curso
VD	Velocidade da roda direita
VE	Velocidade da roda esquerda
Ve	Velocidade ensinada
Vi	Velocidade imitada

Sumário

1 Introdução

1.1 Contextualização

Atualmente a robótica móvel tem muita popularidade e tem sido implementada em diversas áreas, algumas destas áreas podem ser observadas na figura 1. Devido a sua popularidade a comunidade científica tem desenvolvido diversas pesquisas na área da robótica, modelando os robôs móveis como sistemas de múltiplas entradas e múltiplas saídas (sistemas MIMO do inglês *Multi-Input, Multi-Output*) (CHEN; BAI, 2022)(CARLUCHO; De Paula; ACOSTA, 2020), utilizando, portanto, mais de um sensor e mais de um atuador. Diferentes propostas para o controle destes sistemas tem sido propostos na literatura, tais como controle adaptativo (AN; WANG; WANG, 2023) (PETROV; GEORGIEVA, 2018), controle preditivo (FRANCO et al., 2022), controle baseado em modelos (LV et al., 2023)(WANG et al., 2019), controle inteligente (JUANG; LIN; LIU, 2008), controle neural (LIU; ZHU, 2022)(ZHANG; JING, 2020) e controle com lógica nebulosa (TREESATAYAPUN, 2022)(MOHANTA; KESHARI, 2019)(HACENE; MENDIL, 2019), entre outros.

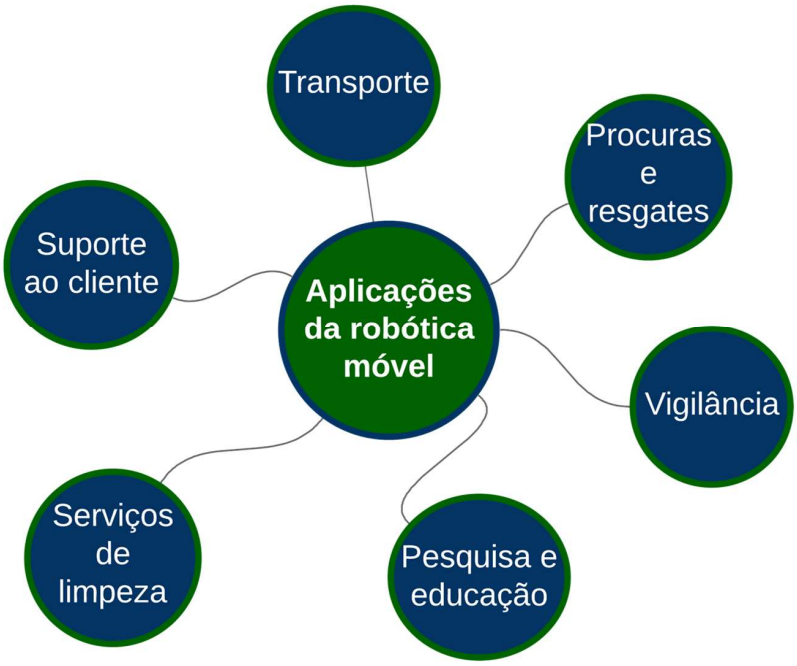


Figura 1 – Aplicações da robótica móvel.

As diferentes técnicas de controle de robôs móveis geralmente são desenvolvidas por pessoas com experiência na área de controle e modelagem de sistemas. Neste contexto, a metodologia de aprendizagem por demonstração (do inglês *Learning From Demonstration*,

LfD) facilita que pessoas sem conhecimento nem habilidades na área de controle possam programar robôs mediante a demonstração de comportamentos desejados (QIAN et al., 2022), os quais são posteriormente imitados pelo robô. Aprendizagem por demonstração é uma técnica de aprendizagem de máquina que permite aos robôs imitarem tarefas a partir das observações de demonstração humana (SAKR et al., 2022) (XU et al., 2019) (WANG et al., 2021).

Neste contexto, o projeto do robô MARIA busca explorar a técnica de LfD aplicada a redes neurais artificiais embarcadas em um Sistema em Chip (SoC, do inglês *System on Chip*). O robô MARIA se movimenta reativamente pela composição de três micro-comportamentos: virar a esquerda (MC3), virar a direita (MC2) e andar em linha reta (MC1). Além disso, a decisão de qual micro-comportamento o robô MARIA vai adotar é tomada por outra rede neural composta de um único neurônio, chamado de *Referee*. Na Figura 2 é mostrada a topologia geral do sistema que implementa os micro-comportamentos e o *Referee*. Observe que o neurônio que implementa o *Referee* seleciona através de um multiplexador (identificado como MUX na Figura 2), os pesos e vieses que vão entrar na rede neural que implementa os micro-comportamentos.

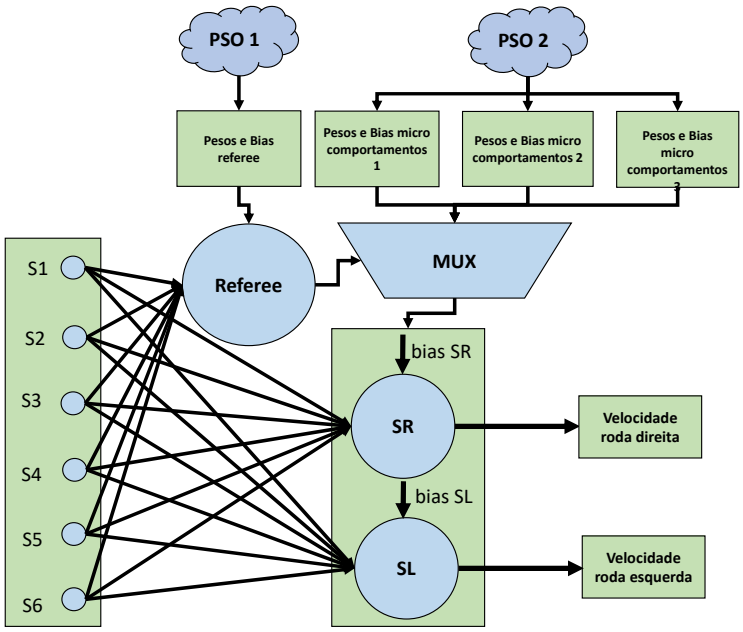


Figura 2 – Topologia da rede neural artificial que implementa os micro-comportamentos junto com o a rede neural do *Referee* .

A partir de testes no robô MARIA, foi observado que o robô apresentava comportamento brusco, ou seja, mudava de comportamento de forma repentina, o que ocasionava colisões em cenários desconhecidos. Com o intuito de solucionar este problema, neste trabalho é proposto a utilização de redes neurais recorrentes (RNR) e a utilização de 6 sensores infravermelhos.

1.2 Justificativa

O robô Maria possui seis sensores infravermelhos e dois motores de corrente contínua para realizar uma navegação puramente reativa. Adicionalmente, o robô usa um kit de desenvolvimento Minized com um SoC (*System on Chip*) FPGA da família Zynq 7007S da Xilinx, e um controle teleoperação com comunicação bluetooth criado para ser utilizado no telefone celular a fim de receber os comandos do operador humano e poder imitar comportamentos.

A topologia proposta visa melhorar a implementação da metodologia LfD em robôs móveis de pequeno porte. Os dois neurônios implementados terão como parâmetros de entrada os seis sensores infravermelhos (em metros) junto com as velocidades das rodas em instantes de tempo anteriores, produzindo assim novos valores de velocidade para a roda direita e esquerda. Um algoritmo de otimização por enxame de partículas (PSO do inglês *Particle Swarm Optimization*) será usado para treinamento do controlador neural recorrente.

Neste contexto, dispositivos FPGAs permitem mapear em *Hardware*, o sistema de controle neural, conseguindo otimizar o tempo de execução do processo de treinamento, explorando o paralelismo intrínseco dos algoritmos envolvidos.

1.3 Objetivo geral

Implementar a metodologia de aprendizagem por demonstração para composição de micro-comportamentos em uma plataforma robótica móvel diferencial, mediante a utilização de redes neurais recorrentes.

1.4 Objetivos específicos

- Propor uma rede neural recorrente para aplicar a metodologia de aprendizagem por demonstração para robôs móveis de pequeno porte usando a técnica de composição de micro-comportamentos proposta no trabalho de (TRIANA, 2022).
- Formular um problema de otimização matemática para treinamento da rede neural recorrente mediante otimização por enxame de partículas.
- Projetar e implementar em SoC FPGA uma arquitetura de hardware da rede neural recorrente treinada offline.
- Desenvolver um sistema *hardware in-the-loop* da metodologia de aprendizagem por demonstração mediante a integração do simulador CoppeliaSim, Python e um SoC FPGA.

- Usar a técnica de co-projeto hardware-software para embarcar o algoritmo de treinamento e a rede neural recorrente no SoC FPGA e validar a metodologia proposta usando o robô Maria.

1.5 Aspectos Metodológicos

A fim de validar a proposta de melhoria da rede neural, primeiramente buscou-se fazer simulações comportamentais utilizando o software *CoppeliaSim* e uma plataforma de linguagem de alto nível. Em seguida, é feita uma segunda simulação, porém utilizando a técnica *Hardware in the loop* (HIL), essa simulação se faz necessária pois o trabalho anterior tinha uma precisão de 27 bits em ponto flutuante, neste trabalho é proposta um redução para 16 bits em ponto flutuante, a fim de reduzir o consumo de recursos lógicos utilizados. Por fim, é feito um teste físico no robô MARIA com a nova topologia de rede neural.

1.6 Contribuição do trabalho

A principal contribuição do presente trabalho é a implementação da metodologia LfD com MCs e *Referee* utilizando uma topologia de rede neural recorrente, aplicado em hardware embarcado. Além de contribuir para analise de precisão de bits, como parâmetros em projetos futuros.

1.7 Organização do trabalho

O restante do presente trabalho contém 5 capítulos, organizados da seguinte maneira: No capítulo 2 é apresentada o fundamento teórico sobre aprendizagem por demonstração, redes neurais artificias recorrentes e FPGAs. No capítulo 3 é apresentado os procedimentos de simulação, o modelo de melhoria proposto, assim como os conceitos das técnicas utilizadas em simulação. No capítulo 4 são mostrados os resultados obtidos em ambiente de simulação, e no capítulo 5 é realizada a análise dos resultados obtidos e as considerações finais.

2 Referencial Teórico

2.1 Aprendizagem por Demonstração

Aprendizagem por demonstração (LfD) é uma técnica de aprendizagem utilizada principalmente na robótica e inspirada no comportamento humano e de outros animais em adquirir novas habilidades através da imitação (BILLARD; GROLLMAN, 2013). Para que um robô aprenda por LfD ele deve seguir os seguintes processos (BAKKER; KUNIYOSHI, 1996):

- Demonstração: que consiste em observar as orientações do professor (geralmente um operador humano). Nesta etapa um conjunto de dados de entradas e saídas são coletados.
- Treinamento: o robô utiliza técnicas de aprendizagem de máquina para obter um modelo matemático que represente o conjunto de dados de entradas e saídas. Por exemplo, atribui pesos e vieses nas camadas da rede neural artificial.
- Imitação: neste processo o robô reproduz aquilo que foi ensinado.

Para o processo de demonstração existem principalmente três técnicas:

- Cinestésico: onde o professor ensina manualmente a tarefa que o robô deve executar.
- Tele-operado: o professor ensina o robô através de um controle remoto, por exemplo.
- Observação direta do movimento: neste caso o robô deve contar com sensores de visão, por exemplo, de forma que ele consiga fazer um modelo do que o professor está lhe ensinando visualmente.

Neste trabalho será usado o método de tele-operação através de um aplicativo de celular, o qual se comunica com o robô móvel através de uma interface bluetooth BLE.

2.2 Redes Neurais Artificiais Recorrentes

Redes neurais artificiais (RNAs) são inspiradas no funcionamento dos neurônios no cérebro humano. A primeira proposta de rede neural artificial foi feita por Warren S. McCulloch e Walter Pitts em sua publicação *"A logical calculus of the ideas immanent in nervous activity"* (MCCULLOCH; PITTS, 1943).

Desde então foram feitas diversas pesquisas sobre essa temática em diversas aplicações. As redes neurais são aplicadas em problemas de regressão de funções, classificação de padrões e predição de estados e, portanto, podem ser encontradas em diversas áreas do conhecimento e da engenharia, tais como:

- Problemas de otimização;
- Processamento de imagens;
- Classificação de padrões;
- Controle de processos industriais;
- Processamento de sinais;
- Agrupamento e mineração de dados;
- Modelagem de séries temporais; e
- Robótica móvel e de manipuladores.

Na robótica as aplicações de RNAs também são diversas, as principais são: planejamento de trajetórias, controle de trajetória ou posição, controle de força, sensoriamento e percepção, ensinamento de tarefas, etc.

Os principais tipos de RNAs são as Perceptron de multi camada (MLP, do inglês Multi Layer Perceptron), um modelo matemático desse tipo de RNA é mostrado na Figura 3, observe que neste modelo tem apenas uma camada, porém pode se aplicar quantas camadas forem necessárias e quantas entradas e saídas fossem necessárias. Nas redes MLP o fluxo de informação vai em uma única direção, das entradas, passando pelas diversas camadas escondidas até alcançar a camada de saída, motivo pelo qual também são conhecidas como redes *feedforward*. A conexão entre os neurônios de uma camada e outra é realizada através de pesos sinápticos (ou pesos de conexão) que favorecem ou inibem a transmissão da informação.

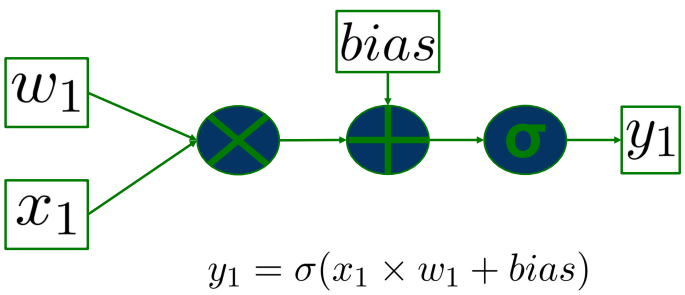


Figura 3 – Modelo matemático de RNA *Feedforward*, adaptado de Bravo e Sotelo (2010).

Outro tipo de RNA são as redes neurais recorrentes (RNAR). Esse tipo de rede implementa uma espécie de memória atrasando os sinais de saídas. Na Figura 4 é mostrado o modelo nas RNAs recorrentes, observe que diferente das RNAs *Feedforward*, a saída da RNA recorrente depende não somente mais das entradas, como também de saídas passadas. Uma das aplicações desse tipo de RNA é o reconhecimento de padrões que estão relacionados, ou seja, que são dependentes, já que essa RNA tem essa memória de curto prazo.

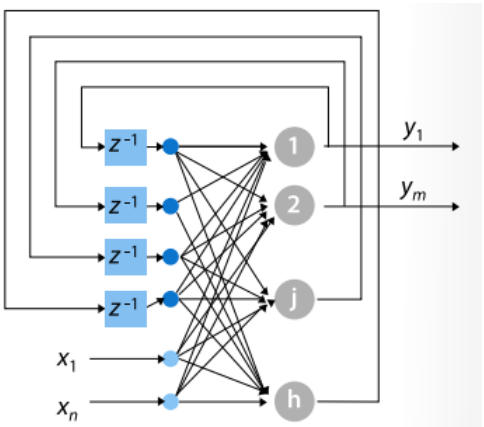


Figura 4 – Modelo matemático de RNA recorrentes, imagem retirada de Bravo e Sotelo (2010).

Caso seja necessário uma memória de longo prazo é utilizada a RNA do tipo LSTM (do inglês, *Long Short Term Memory*). Um modelo desse tipo de RNA é mostrado na Figura 5, onde x são as entradas atuais da LSTM e h_{t-1} é a saída anterior da LSTM. Na Figura 5 é possível observar dois tipos de função de ativação diferentes σ e ϕ.

Diferentes das outras arquiteturas, a arquitetura das RNAs LSTM permite que informações mais antigas se propaguem nas RNA sem a necessidade de colocar mais atrasos. As principais aplicações desse tipo de RNA são: modelagem de linguagem, tradução de idiomas, legendas em imagens, geração de texto, chatbots, entre outros.

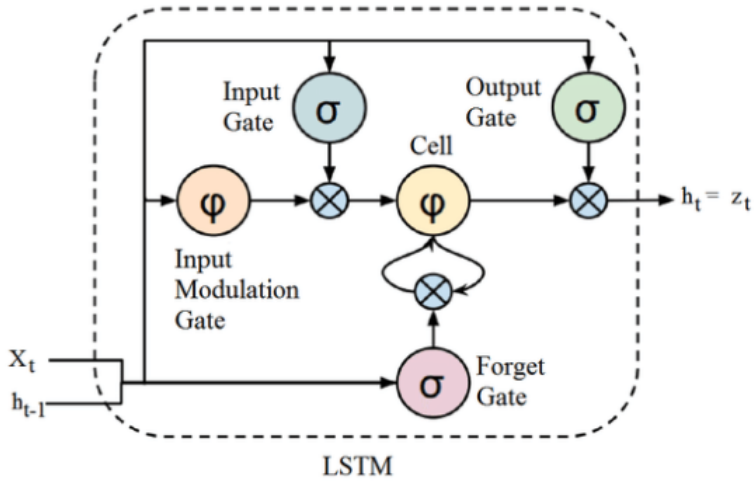


Figura 5 – Modelo matemático de RRNA LSTM, imagem retirada de Academy (2022).

Todas essas topologias de RNAs são treinadas para lidar com um problema específico utilizando um algoritmo de treinamento do qual depende em grande medida o desempenho da rede. Para treinamento de redes RNARs e LSTMs, os algoritmos mais comumente usados são baseados no cálculo do gradiente do erro entre as saídas estimadas pela rede e uma saída esperada. O algoritmo *Backpropagation* (HECHT-NIELSEN, 1992)(WYTHOFF, 1993), introduzido em 1974, por Werbos e John (1974), e variações do mesmo (AHMAD; ISMAIL; SAMAON, 2004), calculam gradientes, o que limita esse algoritmo apenas para funções contínuas onde existe a derivada, além de apresentar restrições de desempenho para funções não convexas, nas quais existem mínimos e máximos locais. Neste trabalho será apresentado uma outra abordagem de treinamento baseada em otimização por enxame de partículas (PSO), descrito na próxima seção.

2.3 Algoritmo de Otimização por Enxame de Partículas

Algoritmos bioinspirados, são sistemas que se inspiram em modelos biológicos. O PSO é inspirado no movimento de bandos de pássaros e de cardumes (KENNEDY; EBERHART, 1995). A vantagem de utilizar esse tipo de algoritmo, é que pode ser aplicado a funções não contínuas e multimodais (com vários mínimos locais) permitindo encontrar todos os pesos simultaneamente em um único vetor. A escolha do PSO foi devido à sua facilidade de implementação, baseado apenas em somas, subtrações, multiplicações e em geração de número aleatórios. Entretanto, pelo fato de ser uma meta-heurística, o algoritmo não garante convergência em um ótimo global.

Na Figura 6 apresenta-se um fluxograma das etapas do algoritmo PSO. Observe que neste algoritmo devemos criar uma função custo que deverá ser minimizada, por exemplo, o erro médio quadrático (ou MSE, do inglês Mean Square Error). Para calcular

o erro são utilizados os valores da saída da RNA com relação ao valor esperado na saída, ou seja, comparam-se os valores treinados em relação aos valores demonstrados.

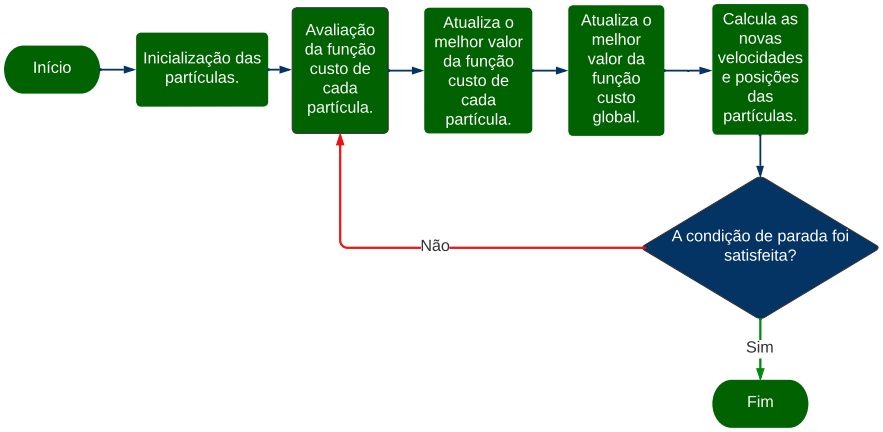


Figura 6 – Fluxograma do PSO simplificado, baseado no trabalho de Eberhart e Kennedy (1995).

Dessa forma, o PSO será utilizado para minimizar o MSE da velocidade das rodas do robô MARIA fornecidos pela RNA com relação as velocidades demonstradas, como mostrado na equação 2.1,

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y}_i)^2, \tag{2.1}$$

onde y_i são os valores da saída da RNA, $\bar{y}_i$ é o valor esperado para aquela saída e N é número de amostras.

O movimento das partículas do PSO é realizado conforme as equações 2.2 e 2.3, onde a equação 2.2 descreve a velocidade da partícula na i-ésima dimensão e a equação 2.3 descreve a posição da i-ésima partícula, para cada d-ésima dimensão.

$$v_{i,t+1}^d = wv_{i,t}^d + c_1 rand(p_{i,t}^d - x_{i,t}^d) + c_2 rand(p_{g,t}^d - x_{i,t}^d) \tag{2.2}$$

$$x_{i,t+1}^d = x_{i,t}^d + v_{i,t+1}^d \tag{2.3}$$

Observe que na equação 2.2, uma dois números aleatórios com distribuição uniforme multiplicam os termos $c_1(p_{i,t}^d - x_{i,t}^d)$ e $c_2(p_{g,t}^d - x_{i,t}^d)$, onde $p_{i,t}^d$ é a melhor posição individual da partícula para a dimensão d-ésima e $p_{g,t}^d$ é a melhor posição global do enxame.

Outros parâmetros importantes são o fator inercial w, o fator de aprendizado cognitivo c_1 e o fator de aprendizado social c_2. O fator inercial modula o tamanho do passo que pode dar uma partícula entre uma iteração e outra. O parâmetro c_1 é o grau

de confiança que a partícula tem na sua própria experiência, e o parâmetro c_2 é o grau de confiança que a partícula tem na solução encontrada pelo enxame, ou seja, se $c_1 > c_2$ as partículas ficam mais dispersas umas das outras, porém se $c_1 < c_2$ as partículas compartilham mais conhecimento global e tendem a ficar mais próximas (WANG; TAN; LIU, 2018).

2.4 Dispositivos FPGA

Field Programmable Gate Arrays (FPGA) são dispositivos utilizados em no desenvolvimento de circuitos lógicos digitais e em aplicações de hardware embarcado, em sua grande maioria, que exigem processamento em tempo real (CHU, 2008). FPGAs são dispositivos lógicos que contam com um *array* de células lógicas programáveis. Além dessas células, os FPGAs também contam com chaves programáveis que permitem conectas células lógicas para compor circuitos digitais complexos.

Na Figura 7 é mostrado uma estrutura conceitual das FPGAs, destacando-se os blocos lógicos, as chaves que estão representadas como *interconexão programável*, além de outras estruturas como blocos de IO, blocos de BRAMs, e blocos DSPs (*Digital Signal Processing*) que integram aos blocos lógicos configuráveis (CLB, do inglês Configurable Logic Block).

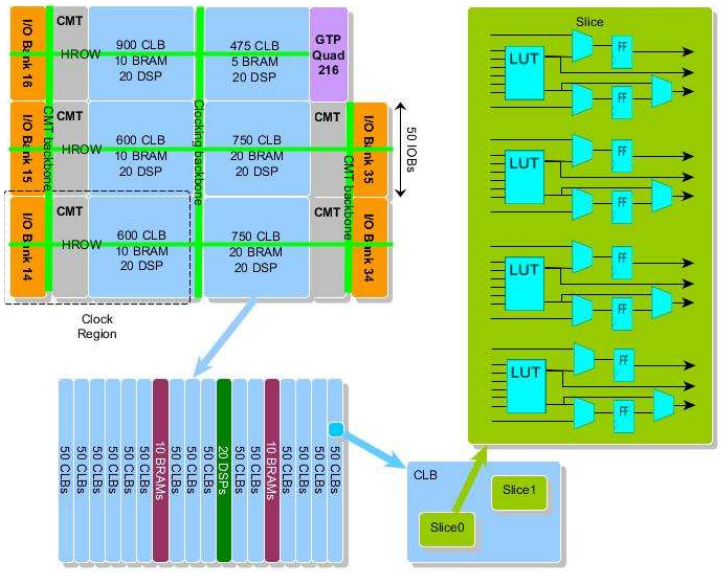


Figura 7 – Estrutura conceitual de uma FPGA.

A arquitetura interna do Zynq é uma plataforma integrada desenvolvida pela Xilinx, que combina um processador de aplicação ARM Cortex-A9 de núcleo duplo (ou, em versões mais recentes, Cortex-A53 ou Cortex-A72) com uma matriz programável de dispositivos lógicos (FPGA) da série Artix-7 ou Kintex-7. Essa combinação de processador e FPGA em um único chip é conhecida como System on Chip (SoC).

A arquitetura interna do Zynq pode ser dividida em três principais componentes: o processador ARM Cortex-A9 (ou Cortex-A53/A72), a matriz lógica programável FPGA e os recursos de interconexão.

O processador ARM Cortex-A9 é um processador de aplicação de núcleo duplo baseado na arquitetura ARMv7-A. Cada núcleo do Cortex-A9 possui sua própria unidade de controle, unidade de ponto flutuante, cache L1 separado (instrução e dados) e uma cache L2 compartilhada. O Cortex-A9 é responsável pelo processamento do sistema operacional, execução de código de aplicação e gerenciamento geral do sistema.

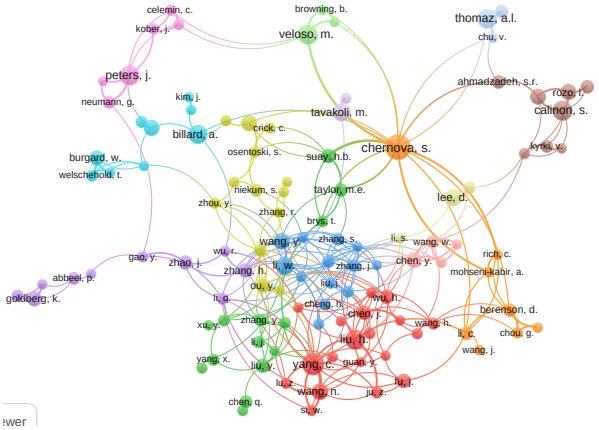
A matriz lógica programável FPGA é composta por blocos lógicos configuráveis, elementos de memória e outros recursos, que podem ser interconectados para implementar uma lógica digital personalizada. A Xilinx oferece diferentes variantes de FPGA (Artix-7 e Kintex-7) para diferentes modelos de Zynq, cada uma com capacidades e recursos variados. O FPGA permite a configuração e reconfiguração flexível da lógica digital para atender aos requisitos do sistema.

O Zynq possui uma variedade de recursos de interconexão para permitir a comunicação entre o processador e o FPGA, bem como com outros periféricos e interfaces externas. Isso inclui barramentos AXI (Advanced eXtensible Interface) para comunicação entre o processador e o FPGA, controladores de memória, controladores de periféricos, controladores de interrupção, interfaces de comunicação como Ethernet, USB, UART, entre outros.

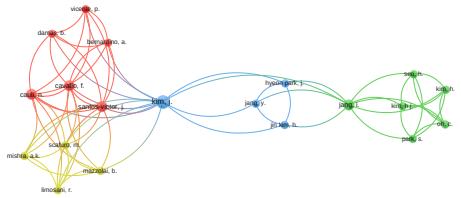
Os componentes do processador e do FPGA são conectados por meio de barramentos AXI, permitindo a transferência eficiente de dados entre eles. Essa integração de processador e FPGA no mesmo chip oferece uma solução poderosa para aplicações que requerem alto desempenho de processamento e flexibilidade de lógica programável.

2.5 Estado da Arte

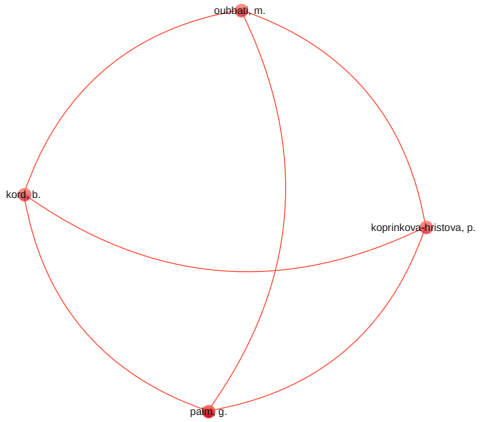
Nesta seção, será realizado um levantamento dos estudos relacionados a este trabalho específico. Foi conduzida uma pesquisa na base de dados do "Scopus"com o intuito de buscar trabalhos relacionados, utilizando as seguintes palavras-chave: "Learning from demonstration"; "Learning from demonstration"e "Mobile Robots"; "Learning from demonstration", "Mobile Robots"e "Recurrent neural network"; e "Learning from demonstration", "Mobile Robots", "Recurrent neural network"e "FPGA". Os trabalhos identificados estão apresentados na Figura 8, onde os *kernels* mostrados são os autores que publicaram artigos relacionado às palavras chaves e as ligações em diferentes cores são os trabalhos publicados. É importante observar que a figura não inclui trabalhos relacionados às palavras-chave: "Learning from demonstration", "Mobile Robots", "Recurrent neural network"e FPGA, pois não foi encontrado artigos relacionados.



(a) Palavras chaves utilizada : *Learning from demonstration*.



(b) Palavras chaves utilizada : *Learning from demonstration* e *Mobile Robots*.



(c) Palavras chaves utilizada : *Learning from demonstration*, *Mobile Robots* e *Recurrent neural network*.

Figura 8 – Artigos publicados nas áreas de *Learning from demonstration*, *Mobile Robots* e *Recurrent neural network*.

Na Tabela 1 mostra os trabalhos relacionados com o presente trabalho. Observe que os trabalhos mostrados na Tabela 1 estão relacionados, mas estes trabalhos utilizam técnicas diferentes. O trabalho de Mohamed et al. (2014) utiliza RNARs, mas não utiliza uma FPGA como plataforma de controle.

Tabela 1 – Comparação entre os trabalhos relacionados.

Autor	**Ano**	**Técnica de Aprendizagem**	**Atuadores**	**Sensores**	**Plataforma**	**Sistema de Locomoção**
D. Muñoz et all (MUñOZ et al., 2014)	2014	Redes Neurais	2 motores CC	5 Sensores infravermelhos	FPGA xc5vlx110t 30F6 (Eye-Sim and FPGA)	Diferencial
M. Oubbati et all (MOHAMED et al., 2014)	2014	Redes Neurais Recorrentes	2 motores de passo	8 Sensores infravermelhos	dsPIC 30F6014A (robô E-Puck)	Diferencial
H. Tan (TAN, 2015)	2015	Algoritmo computacional Evolucionário	2 motores CC	7 Sendores de luz	dsPIC 30F6014A (robô E-Puck)	Diferencial
N.Vuković et all (VUKOVIć; MITIć; MILJKOVIć, 2015)	2015	Redes Neurais Bioinspiradas	2 motores CC	5 Ultrasom	Motorola 68331 (robô Khepera II)	Diferencial
L. Jiang et al (JIANG et al., 2018)	2018	Controle de servo visual usando a teoria de controle de espaço não vetorial	4 motores CC	1 Câmera	Computador (robô)	Manipulador paralelo móvel
D.G Sillas et all (GARCIA et al., 2018)	2018	Processos Gaussianos	NA	Arquivo	ARM cortex A7 (simulação)	NA
J. Liu et all (LIU et al., 2021)	2020	Redes Neurais Spiking	2 motores CC	3 Sensores de distância	Computador (Simulação)	Diferencial
Omar Gamal et all (GAMAL; CAI; ROTH, 2020)	2020	Controle Nebuloso	2 motores CC	5 Infravermelho	Computador (Simulação)	Hackerman
F. Jefferson (BORGES, 2020)	2020	Redes Neurais	2 motores CC	5 Ultra-som	Arduino Uno e Arduino Mega	Diferencial
Y. Wei et all (WEI et al., 2021)	2021	Redes Neurais Convolucionias	NA	1 Sensor Lidar, odometria e sensor de inercia	Computador (ROS)	Diferencial
Mario Pastrana (TRIANA, 2022)	2022	SLP adaptativa + PSO	2 motores CC	4 sensores infravermelhos	SoC FPGA (robô MARIA)	Diferencial
Este trabalho	2023	SLP recorrente adaptativa + PSO	2 motores CC	4 sensores infravermelhos	SoC FPGA (robô MARIA)	Diferencial

No seguinte capítulo será fornecida uma visão geral do robô MARIA e quais mudanças foram propostas a fim de melhorar a performance do robô.

3 Metodologia

3.1 Robô MARIA

O robô MARIA (Figura 9) é uma plataforma de robótica móvel desenvolvida Universidade de Brasília por Triana (2022). O robô MARIA é um robô autônomo que usa sensores de distância para realizar a percepção do ambiente. Além disso, para tomar as decisões de fazer uma curva, continuar em frente e desviar de obstáculos, este robô conta com uma rede neural artificial embarcada em uma FPGA. A escolha de uma FPGA para tal tarefa é devido sua alta velocidade de processamento necessária durante a etapa de aprendizagem de comportamentos, assim como pela sua capacidade de reconfiguração. Na Figura 10 é mostrado como foram feitas as conexões e os componentes que constituem a parte eletrônica do robô MARIA.

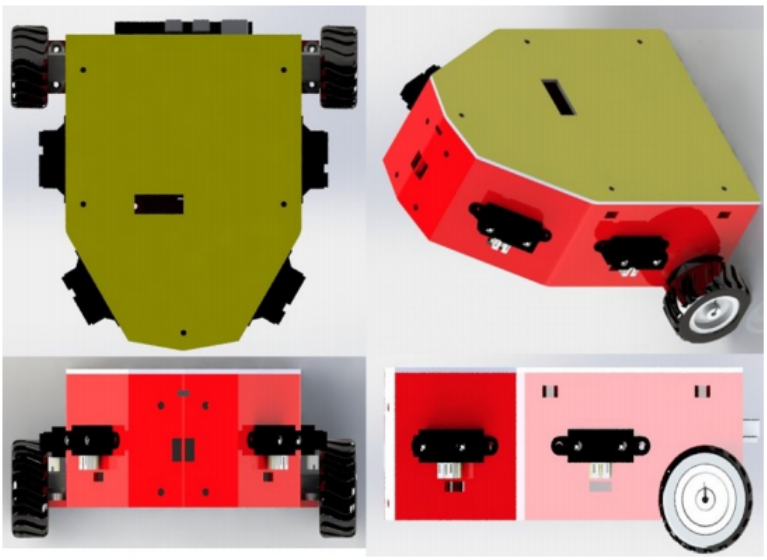


Figura 9 – Chassi do Robô Maria. (TRIANA, 2022)

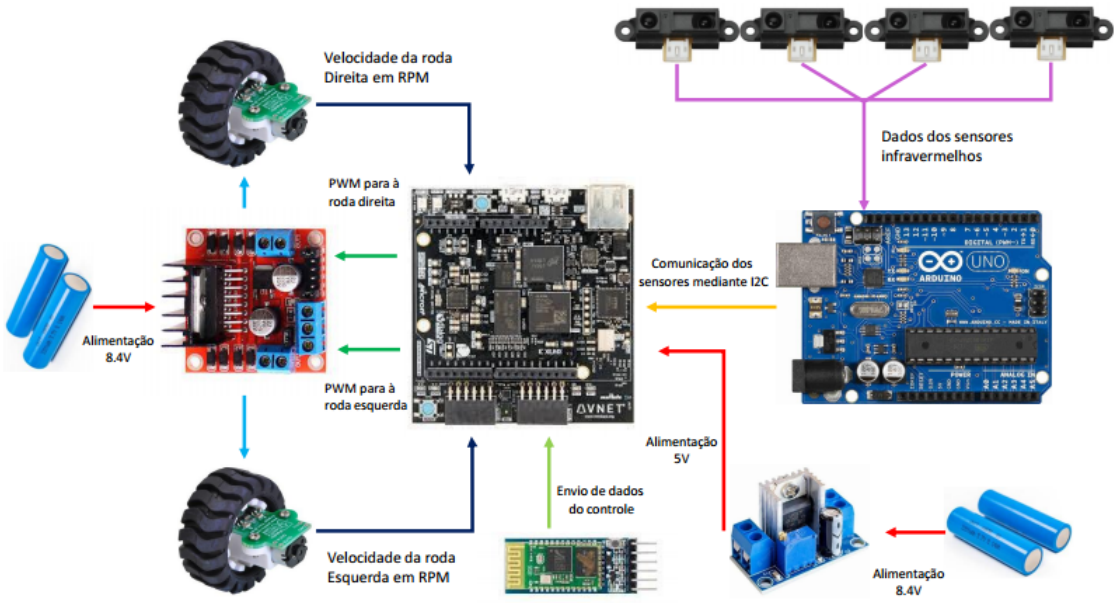


Figura 10 – Conexões eletrônicas do robô Maria. (TRIANA, 2022)

Observe que na figura 10, a alimentação do kit de desenvolvimento MiniZed é alimentado apenas pelo regulador de tensão, assim como o Arduíno. Assim, na próxima seção é discutida as mudanças nas conexões eletrônicas do robô MARIA.

3.1.1 Mudanças realizadas na plataforma

A fim de reduzir o consumo de energia do robô MARIA, forma implementadas as seguintes substituições nas conexões eletrônicas (vide figura 11):

- Multiplexador analógico: no lugar de lermos os sensores através das portas analógicas do Arduino UNO e transmiti-las via comunicação I2C, é proposto a utilização de um multiplexador analógico de 16 canais.
- Conversor CC-CC abaixador: a fim de melhorar a eficiência energética transmitida para o SoC FPGA, é proposto um conversor CC-CC abaixador de tensão exclusivo para alimento do kit de desenvolvimento MiniZed.
- Regulador de tensão: aproveitando o regulador de tensão do projeto anterior, é proposto que ele seja utilizado exclusivamente para alimentação dos sensores IR.

Por fim, na figura 11 é mostrado o novo diagrama de conexões eletrônicas do robô MARIA.

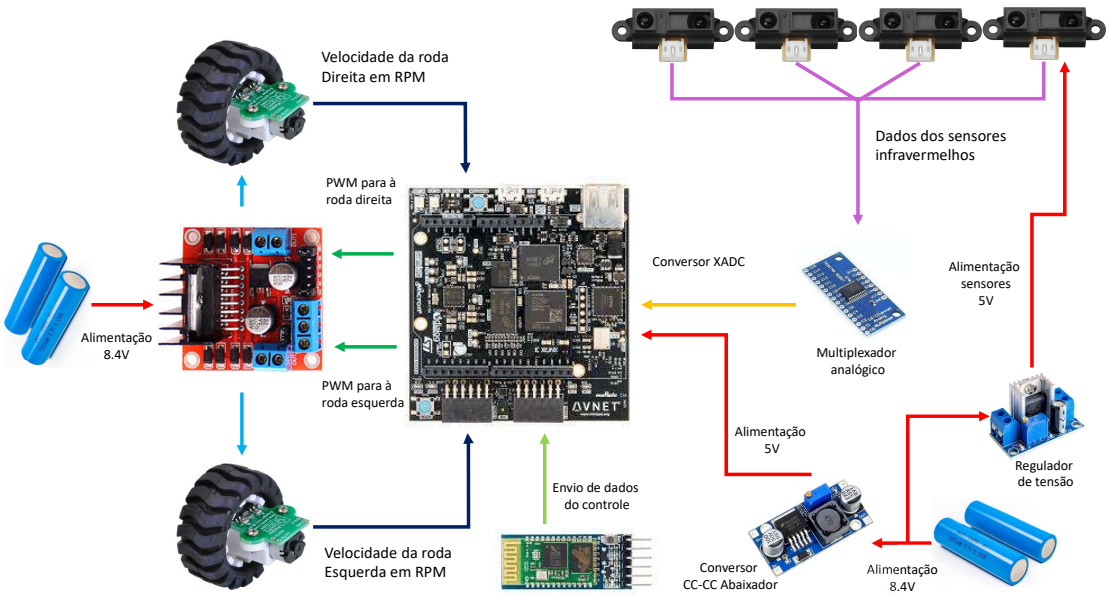


Figura 11 – Nova conexão proposta para o robô MARIA.

Devido a essas novas conexões foi necessário realizar novamente a calibração dos sensores, pois agora o conversor AD utilizado é o IP disponível na MiniZed chamado XADC.

3.1.2 Calibração do sensores

Com a nova proposta de conexões eletrônicas mostradas na figura 11, se faz necessário uma nova calibração dos sensores. Dessa forma, devido as características do sensor IR, foi feita uma calibração por partes, como mostrado na figura 12. Na figura, a curva em vermelho modela a saída do sensor de 10 a 45 cm e a curva em azul modelo a saída do sensor de 50 a 80 cm. Foi feita uma interpolação dos dados coletados.

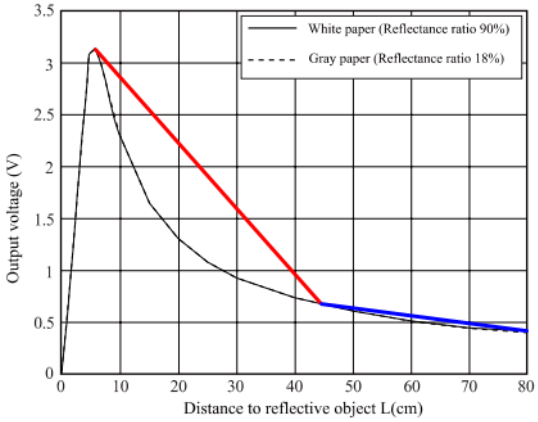


Figura 12 – Forma de onda da saída do sensor IR, adaptado de (SHARP, 2011).

Para coletar os dados do sensor foi feito um arranjo experimental como mostrado na figura 13. Observe que o robô MARIA fica fixo em uma posição e então, com o auxílio de uma régua, é modificada a posição do objeto de detecção, uma tábua de madeira, iniciando em 10 cm (distância miníma que o sensor pode medir) até 45 cm em passos de 1 cm. Após esse procedimento a posição do objeto e variada em passos de 5 cm até 80 cm.

Cada distância é medida 16 vezes, resultando em 16 amostras para cada ponto. Foi usada a mediana dos valores medidos em cada ponto e uma regressão linear foi realizada para obter um modelo aproximado em cada faixa de operação.

Por fim, é feito um teste medindo os mesmos pontos com o mesmo arranjo da figura 13 e calculado o erro de medida já com as retas encontradas na regressão linear e implementada em *hardware*.

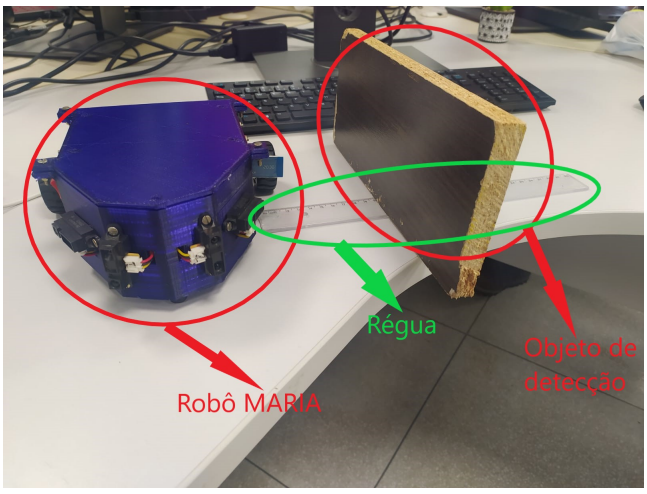


Figura 13 – Configuração para coleta de dados do sensor de distância IR.

3.2 Arquitetura da rede neural no robô MARIA

Até o momento tem sido apresentado as contribuições realizadas na plataforma do robô MARIA. Nesta seção será esclarecido qual arquitetura de rede neural o robô MARIA utiliza e como ela funciona.

O robô MARIA utiliza uma rede SLP (*single-layer perceptron*) adaptativa de apenas dois neurônios para decidir qual movimento fazer dependendo da informação dos sensores de medição de distância. Para que o robô aprenda é aplicada a técnica de aprendizagem por demonstração.

Na Figura 14 é mostrada a topologia de como a rede esta estruturada, suas entradas e saídas. Observe que a rede conta com um neurônio chamado *Referee*, que decide quais pesos vão entrar no controlador SLP que determina uma referência de velocidade das rodas do controlador PI (do inglês *Proportional–Integral controller*) que atua sobre o driver dos motores.

O uso do *Referee* permite que a rede SLP seja adaptativa, implementando um ou outro micro-comportamento em função das entradas dos sensores. Dessa forma, se a saída do *Referee* estiver entre que 0,28 e 0,23 os pesos e vieses do micro-comportamento 1 são selecionados, caso a saída do *Referee* seja maior que 0,28 os pesos e vieses do micro-comportamento 2 são selecionados, e por fim se a saída do *Referee* for menor que 0,23 os pesos e vieses do micro-comportamento 3 são implementados.

Os micro-comportamentos implementados em Triana (2022) são:

- Andar para frente no meio de um corredor (micro-comportamento 1).
- Girar no sentido horário em torno de um obstaculo de 2 metros de diâmetro (micro-comportamento 2).
- Girar no sentido anti-horário em torno de um obstaculo de 2 metros de diâmetro (micro-comportamento 3).

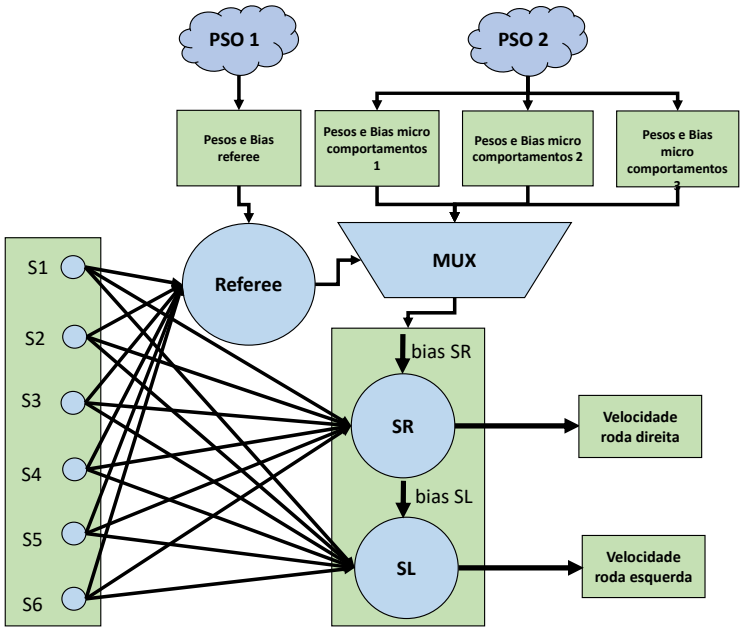


Figura 14 – Topologia de controle do robô Maria. (TRIANA, 2022)

Os resultados obtidos dessa topologia foram satisfatórios, como podemos observar nas Figuras 15, 16 e 17, onde é mostrado o resultado de simulação da etapa de demonstração e imitação para cada micro-comportamento.

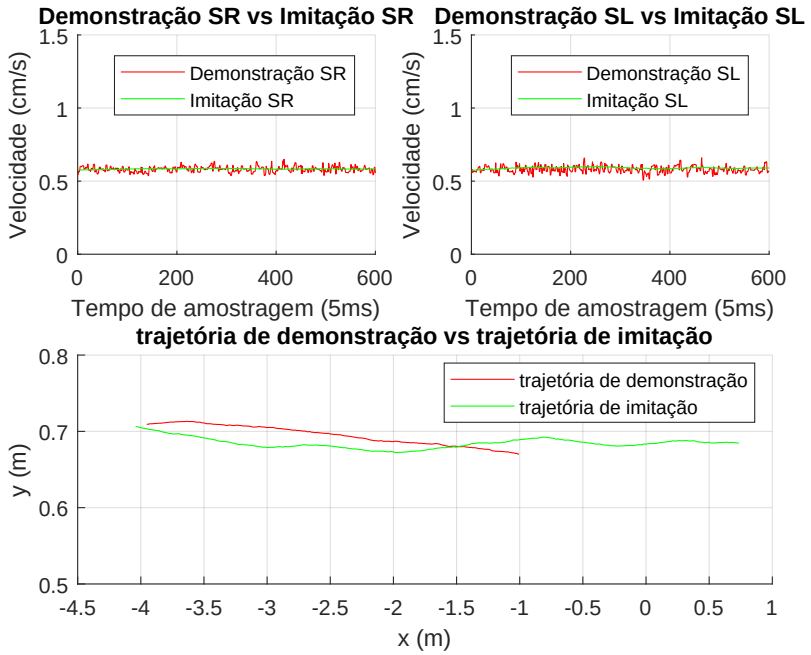


Figura 15 – Trajetórias ensinadas na fase de demonstração e resultados obtidos na fase de imitação para o micro-comportamento 1, seguindo em linha reta.. (TRIANA, 2022)

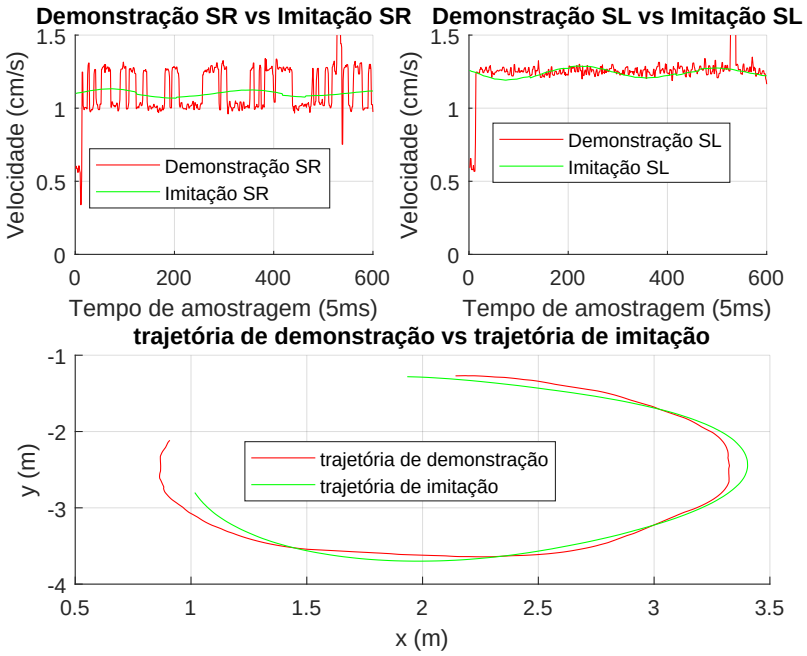


Figura 16 – Trajetórias ensinadas na fase de demonstração e resultados obtidos na fase de imitação para o micro-comportamento 2, girando em sentido horário. (TRIANA, 2022)

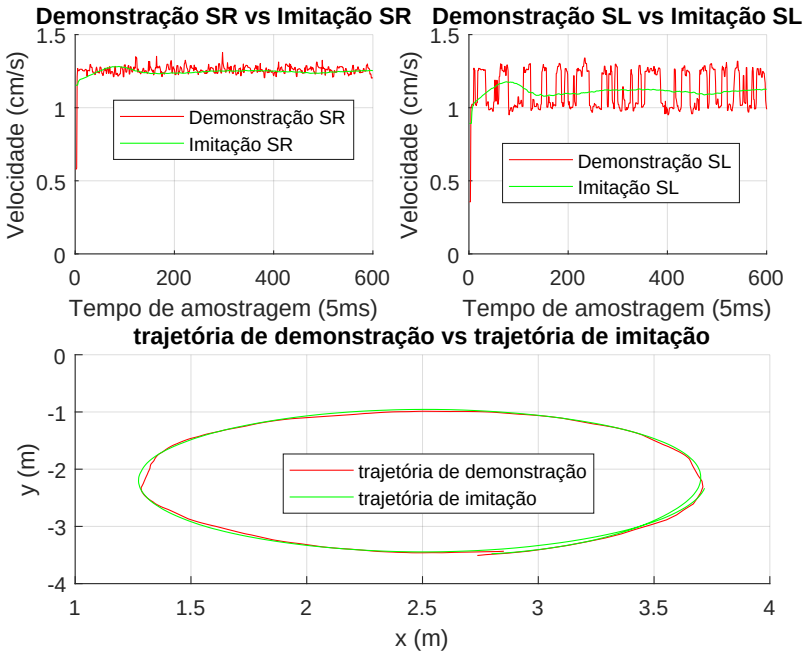


Figura 17 – Trajetórias ensinadas na fase de demonstração e resultados obtidos na fase de imitação para o micro-comportamento 3, girando em sentido anti-horário. (TRIANA, 2022)

O consumo de recursos lógicos de um neurônio da rede SLP implementada no robô MARIA é mostrado na Tabela 2. Essa arquitetura apresentada por Triana (2022), tem 27

bits de precisão em ponto flutuante, implementado na SoC FPGA Zynq-7007 disponível no *kit* de desenvolvimento MiniZed.

Tabela 2 – Consumo de recurso lógicos do neurônio implementado no SoC FPGA Zynq-7007 com 27 bits ponto flutuante.

Neurônio	**Slice LUTs (14400)**	**DSP (66)**	**Flip Flops (28800)**
	Slice LUTs%	**DSP%**	**Flip Flops%**
Somador 1	334	0	51
	2,32%	0%	0,18%
Somador 2	1.232	0	51
	8,56%	0%	0,18%
Multiplicador	229	1	30
	1,59%	1,52%	0,1%
Sigmoide	334	1	82
	2,32%	1,54%	0,28%
Total	2307	2	1668
	16,02%	3,03%	5,79%

Dessa forma, o presente trabalho buscar aplicar redes neurais recorrentes nos micro-comportamentos de modo a observar se o robô apresenta melhores resultados comparado ao sistema desenvolvido por Triana (2022) e se o robô apresenta menor reatividade, com essa nova topologia.

3.2.1 Modelo de melhoria proposto para rede neural

Na intenção de melhorar o comportamento do Robô Maria, são propostas duas topologias diferentes. A primeira topologia é mostrada na Figura 18, esta topologia é chamada de Rede neural artificial recorrente diagonal (RNARD). Neste trabalho, essa topologia será referenciada como Topologia 1.

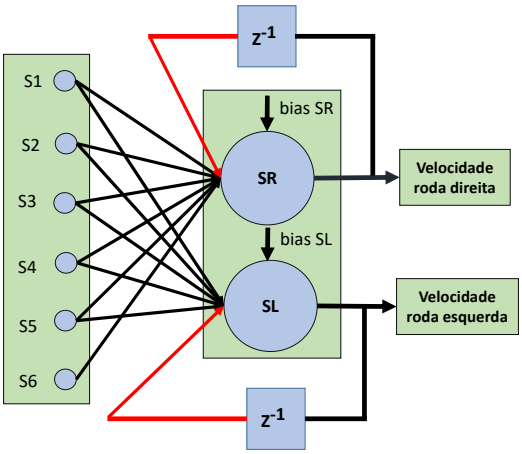


Figura 18 – Topologia 1 proposta para modificação da topologia original.

A outra topologia proposta é a mostrada da Figura 19. Essa topologia é chamada de Rede neural artificial recorrente completamente conectada (RNARCC). Essa topologia será chamada neste trabalho de Topologia 2.

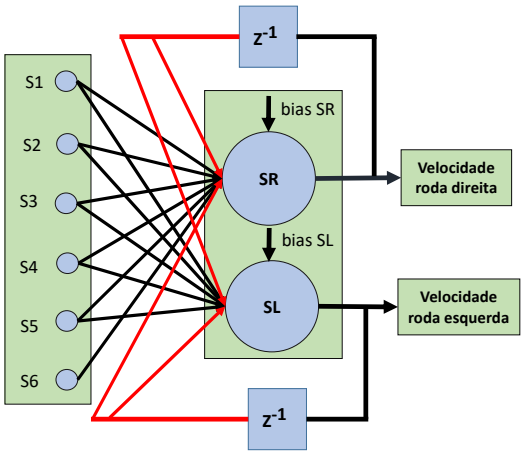


Figura 19 – Topologia 2 proposta para modificação da topologia original.

Segundo Ku e Lee (1995), essas duas topologias conseguem fazer um mapeamento dinâmico em comparação com RNA *Feedforward*, porém necessitam de mais pesos que a RNA *Feedforward*. Já a comparação entre a RNARD e RNARCC é que a RNARCC tem mais graus de liberdade para interpolar as funções aprendidas, porém as RNARCCs precisam do dobro de pesos de realimentação do que a RNARD e mais memória. Um desvantagem desses dois tipos de RNAR é que a convergência não é garantida, mas para sistemas dinâmicos elas apresentam bons resultados.

3.3 Metodologia para validação das topologias

De forma a explorar as capacidades das RNAR foram realizadas simulações utilizando o software *CoppeliaSim*, onde contém o modelo do robô MARIA e adicionalmente foi utilizado o a linguagem de alto nível *Python*, para realizar a interface entre o modelo do robô MARIA e o modelo das RNAR propostas.

Assim para validar as topologias de RNAR, foram feitos os seguintes passos:

1. Simulações utilizando o *CoppeliaSim* e *Python*.
2. Simulação mista (hardware/software), chamada *Hardware-in-the-Loop* utilizando o *CoppeliaSim* e um Soc FPGA.
3. Teste dos micro-comportamentos utilizando o robô físico.

Contudo, foi implementado em *Hardware-in-the-Loop* e no robô físico apenas a topologia da RNARD, devido a sua simplicidade e melhor convergência, em relação a topologia de RNARCC.

Os FPGAs utilizados nesse trabalho são os dos kits de desenvolvimento SoC Pynq-z2 e Minized. O primeiro será utilizado para fazer a simulação em *Hardware-in-the-loop*, pois este *kit* conta com uma portal *Ethernet* necessária para realizar a comunicação entre o computador e o SoC FPGA. Já o segundo dispositivo, MiniZed, é utilizado no projeto do robô MARIA, já mencionado anteriormente. Uma tabela comparativa entre os dois dispositivos é mostrada na Tabela 3.

Tabela 3 – Comparativos da quantidade de dispositivos lógicos disponíveis nas FPGAs dos kits de desenvolvimento miniZed e Pynq-Z2.

Ferramentas	**MiniZed**	**PYNQ-Z2**
FPGA	Xilinx Zynq SoC XC7Z007S	Zynq-7000 SoC XC7Z020-1CLG400C
Memoria	Micron 512 MB DDR3L; Micro 128 Mb QSPI Flash; Micron 8 GB eMMC;	512 Mbyte DDR3 com barramento de 16-bit @ 1050 Mbps; 128 Mbit Quad-SPI Flash; Conector de cartão Micro SD;
Interfaces padrão	USB 2.0; Compatibilidade: com Arduino e Pmod;	USB 2.0; Compatibilidade com: Arduino, Pmod e Raspberry Pi;
Outros periféricos	Wi-Fi 802.11b/g/n; Bluetooth 4.1 plus EDR and BLE (Bluetooth Low Energy); Microfone; Sensor de temperatura; Conversor XADC de 12 bits.	HDMI sink port (entrada); HDMI source port (saída); Interface I2S com 24 bits DAC;

Na Tabela 4 é mostrada a quantidade de tabelas verdades (LUTs, do inglês Look-Up Tables), *flip-flops* (FF), blocos de memória RAM (BRAM) e dispositivos de processamento de sinais (DSPs, do inglês Digital Signal Processing) de cada FPGA disponível nos *kits* de desenvolvimento, MiniZed e Pynq-Z2.

Tabela 4 – Comparativos da quantidade de dispositivos lógicos disponíveis nas FPGAs dos kits de desenvolvimento miniZed e Pynq-Z2.

FPGA	**LUTs**	**FFs**	**BRAMs**	**DSPs**
Xilinx Zynq SoC XC7Z007S	14400	28800	50 (1.8Mb)	66
Zynq-7000 SoC XC7Z2020-1CLG400C	53200	106400	140 (4.9Mb)	220

3.3.1 *Hardware-in-the-Loop* (HIL)

A técnica HIL consiste em simular um modelo em tempo real, permitindo que as FPGAs acelerem o tempo de simulação devido à sua alta capacidade de processamento (CHERRAGUI; HILAIRET; GIURGEA, 2015). Na Figura 20 é mostrada a configuração do sistema para fazer a simulação HIL.

Observe na Figura 20 que o modelo do sistema que está sendo trabalhando pode estar no processador de tempo real ou em um simulador desenvolvido em outro dispositivo, como um computador. No caso deste trabalho é utilizado a interface de comunicação *Ethernet* e o modelo do sistema está no *software CoppeliaSim*.

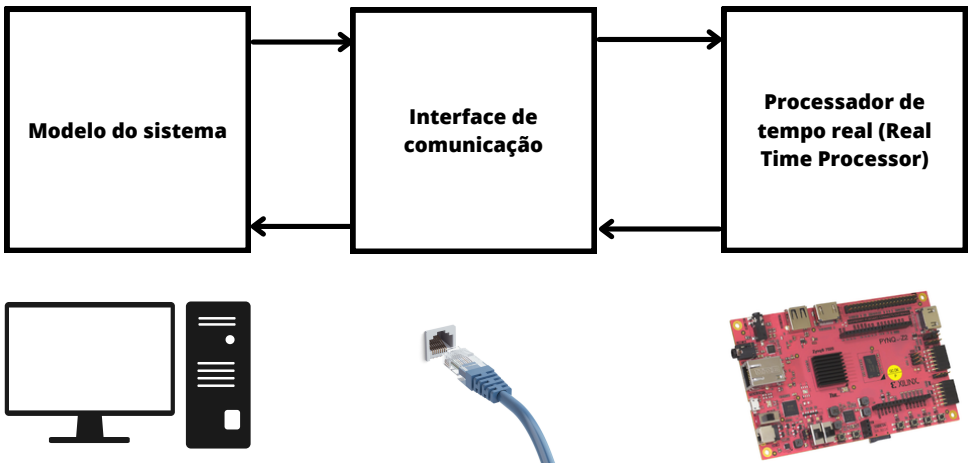


Figura 20 – Configuração do sistema para simulação HIL.

Para viabilizar esse tipo de simulação foi utilizado um IP chamado Comblock, desenvolvido pelo Centro Internacional de Física Teórica (ICTP, Italia). Este IP implementa a interface entre a parte lógica do SoC FPGA e o computador onde está armazenado o modelo do sistema que se deseja testar, no caso deste projeto os componentes mecânicos do robô MARIA e os cenários usados para treinar os micro-comportamentos.

3.3.1.1 Modelo implementado nos SoCs FPGAS

O modelo do *hardware* implementado é o mostrado na Figura 21. Observe que o numero de bits foi reduzido em relação ao trabalho de Triana (2022). Outro ponto a ser observado é que a realimentação não está sendo implementada em *hardware*, a saída anterior da RNAR é enviada pelo computador onde está o modelo do robô MARIA. Na

seção posterior será mostrado os resultados obtidos com essa topologia de *hardware* na simulação HIL.

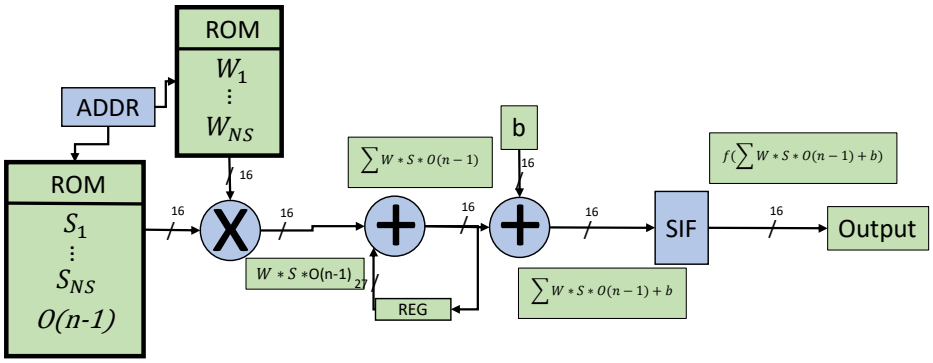


Figura 21 – Modelo do neurônio implementado em *hardware*.

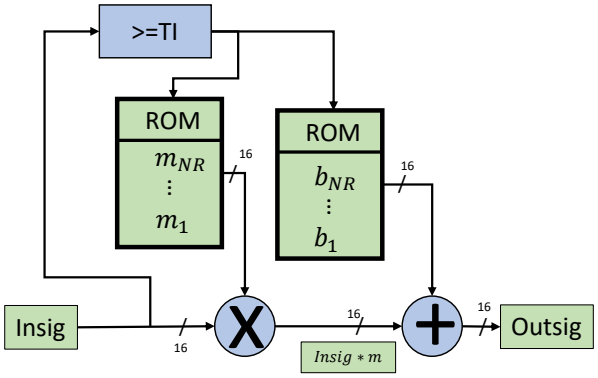


Figura 22 – Modelo da função Sigmoide implementado em *hardware*.

3.4 Procedimento para o treinamento das RNAR

Os procedimentos descritos nesta seção serão utilizados tanto na fase de simulações, quanto aplicados em *hardware*. Dessa forma, inicialmente é simulado o Cenário 1, que é um cenário criado no *software CoppeliaSim*, que conta com duas paredes em paralelo, simulando um corredor, como mostrado na Figura 23. Após coletar os dados de velocidade desse cenário, o robô é treinado para andar em linha reta. Então é simulado novamente o Cenário 1 com os pesos obtidos no treinamento e é avaliado visualmente se o robô imitou corretamente o que foi ensinado.

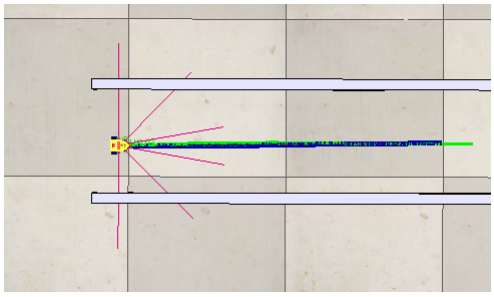


Figura 23 – Cenário para ensinar o micro-comportamento 1 a andar em linha reta.

Para o Cenário 2 segue-se o mesmo procedimento feito no Cenário 1, contudo neste cenário será ensinado o robô a andar em sentido horário. O Cenário 2 é mostrado na Figura 24.

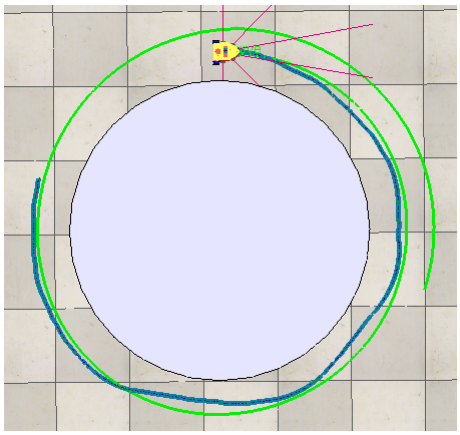


Figura 24 – Cenário para ensinar o micro-comportamento 2 a andar em sentido horário.

Por fim, o mesmo procedimento é feito para o Cenário 3 e ensinado ao robô a andar em sentido anti-horário, como mostrado na Figura 25.

Coletados os dados de cada cenário, pode ser feito o treinamento de cada micro-comportamento individualmente.

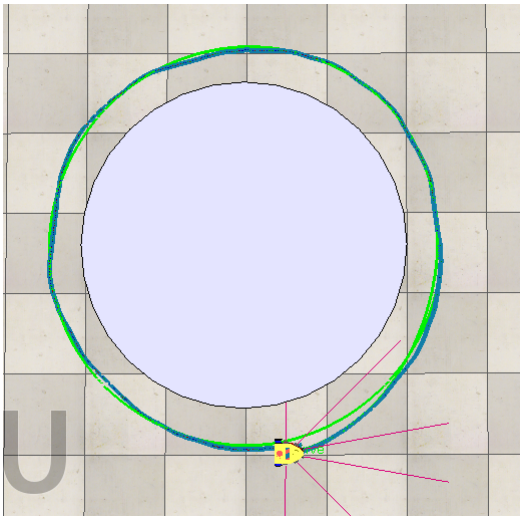


Figura 25 – Cenário para ensinar o micro-comportamento 3 a andar em sentido anti-horário.

Em resumo o procedimento feito para ensinar e treinar o robô é mostrado no fluxograma da Figura 26. Este procedimento é feito para todos os cenários e topologias. Ao fim desses passos, são implementados os pesos e vieses encontrados na fase de treinamento e então calculado o MSE da velocidade imitada com relação a velocidade ensinada, assim como o MSE da trajetórias para o caso da implementação dos micro-comportamentos.

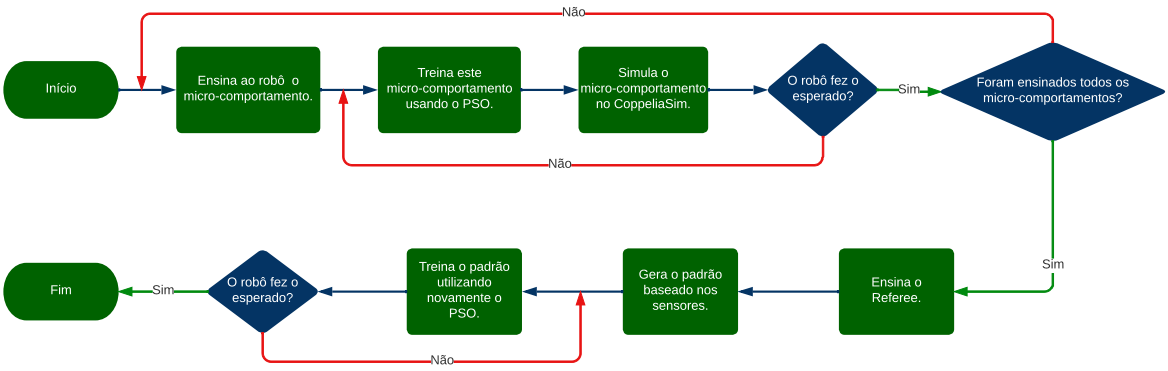


Figura 26 – Procedimento para o treinamento para cada micro-comportamento, independente da topologia.

3.5 Planejamento dos experimentos com o robô MARIA

Para realizar a avaliação da metodologia LFD proposta foram desenhados 3 cenários distintos, implementados com os três micro-comportamentos ensinados, conforme mostrado na tabela 5. A fim de observar a trajetória realizada é proposta a metodologia já implementada no trabalho de Triana (2022), que consiste em colocar uma marcador ArUco na tampa superior do robô MARIA, como mostrado na figura 27 e rastreá-lá utilizando a técnica de visão computacional apresentada em (PHAM et al., 2021).

Tabela 5 – Cenários de treinamento dos micro-comportamentos para a metodologia LFD, tabela adaptada de Triana (2022).

Num	Cenário	Gráfica	Observações
1	Micro-comportamento 1: Avançar para frente		- O robô tenta ir reto em um corredor de 62 cm de comprimento
2	Micro-comportamento 2: Girar no sentido horário		-Gira no sentido horário em um cilindro de 70 cm de diâmetro
3	Micro-comportamento 3: Girar no sentido anti-horário		-Gira no sentido anti-horário em um cilindro de 70 cm de diâmetro

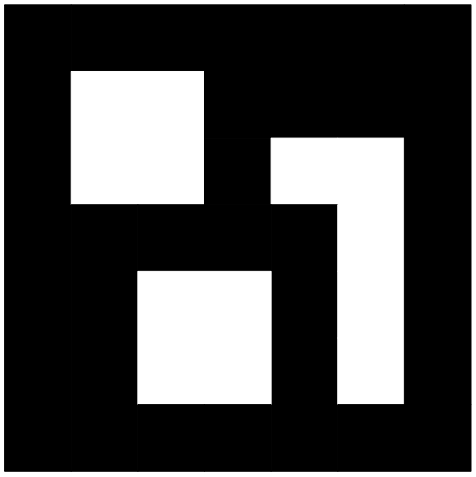


Figura 27 – Marcador ArUco colocado na parte superior do robô MARIA (TRIANA, 2022).

Dessa forma, os seguintes procedimentos foram desenvolvidos para validar o comportamento do circuito:

1. Verifica-se o nível de tensões das baterias.
2. Testa os sensores IR e os encoders.
3. Monta o cenário a ser testado.
4. Posiciona o robô MARIA a 20 cm de distância dos obstáculos.
5. Liga o robô MARIA.

6. Usando um câmera, se inicia a gravação do cenário de teste.

4 Resultados

4.1 Análise da calibração dos sensores

Ao realizar os procedimentos descritos na seção 3.1.2, foram obtidos os resultados mostrados na figura 28. Observa-se que a curva em verde é o resultado da interpolação e os pontos em vermelho são as medianas das medidas dos sensores. Os coeficientes encontrados na interpolação são mostrados na equação 4.1, onde D_{10a45} é função que modela o sensor de 10cm a 45cm, D_{50a80} é função que modela o sensor de 50 cm a 80 cm e x é o valor lido do conversor XADC em binário.

$$D_{10a45} = -0,0005084x + 48,23 \tag{4.1}$$

$$D_{50a80} = -0,0022108x + 75,22 \tag{4.2}$$

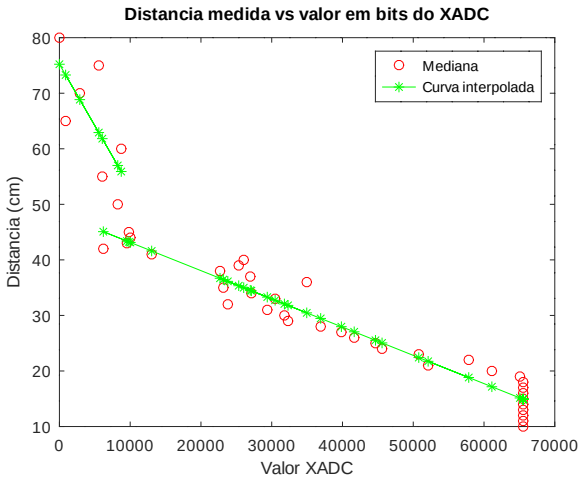


Figura 28 – Comparação entre o a curva interpolada e a mediana das medidas realizadas para fazer a interpolação.

A curva interpolada tem um coeficiente de correlação quadrático igual a 0,93 para as distâncias entre 10 cm a 45 cm. Contudo para as distâncias entre 50 cm a 80 cm o coeficiente de correlação é igual a 0,49. Na figura 29 é mostrado o resultado das medidas feitas pelo sensor usando os modelos da interpolação em azul e a media dessa medidas em vermelho.

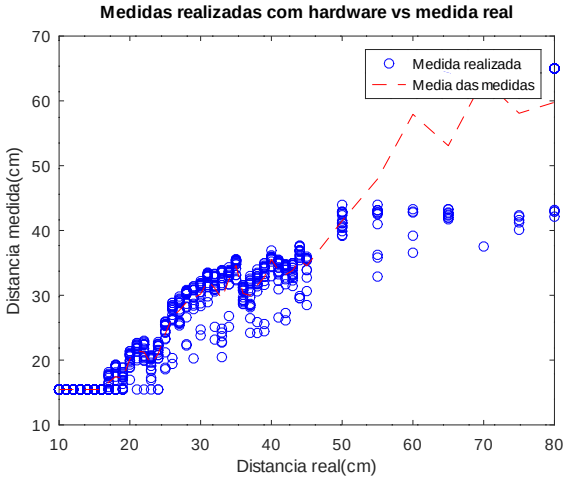


Figura 29 – Comparação entre as medidas realizadas com o modelo implementado e as medidas reais.

4.2 Resultados obtidos em simulação

4.2.1 Simulação dos micro-comportamentos

4.2.1.1 Topologia 1 - RNARD

Dada a topologia apresentada na Figura 18, foi feito o ensinamento dos 3 micro-comportamentos e comparado com os valores imitados das velocidades da roda direita (VD) e da roda esquerda (VE) em metros por segundo (m/s). Esses resultados são mostrados na Tabela 6.

Tabela 6 – Erros médios quadráticos das velocidades imitadas com relação as velocidade demonstradas para cada micro-comportamento.

	VD[m/s]	**VE [m/s]**
Micro-comportamento 1	1.16e-03	1.14e-03
Micro-comportamento 2	1.83e-04	9.48e-05
Micro-comportamento 3	6.85e-04	6.41e-04

Nas Figuras 30, 31 e 32 são mostradas as trajetórias e velocidades (ensinadas e imitadas) dos micro-comportamentos 1, 2 e 3 respectivamente, para a topologia 1.

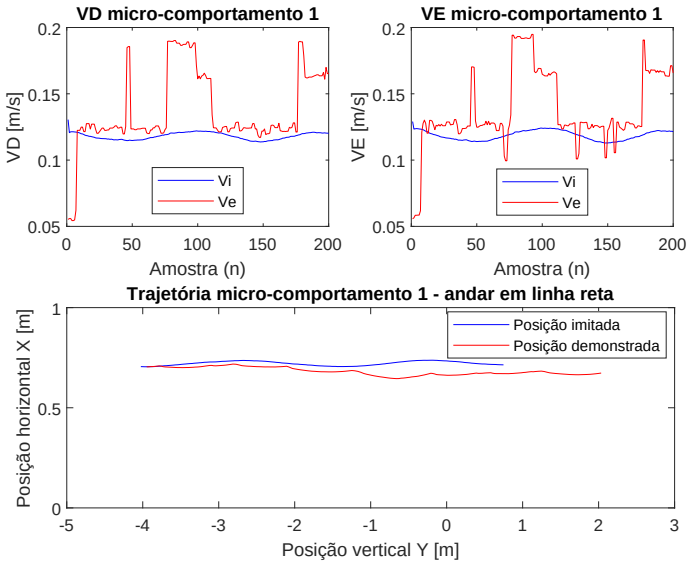


Figura 30 – Resultados de simulação do micro-comportamento 1, onde Vi é a velocidade imitada e Ve é a velocidade ensinada, para a topologia 1.

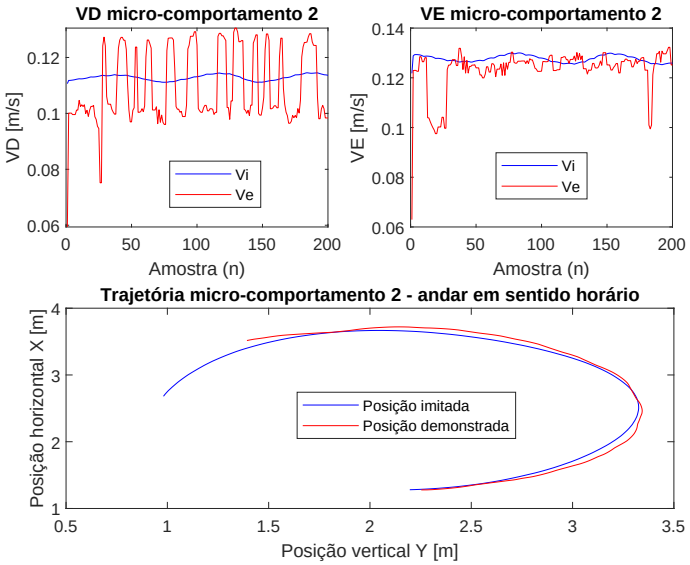


Figura 31 – Resultados de simulação do micro-comportamento 2, onde Vi é a velocidade imitada e Ve é a velocidade ensinada, para a topologia 1.

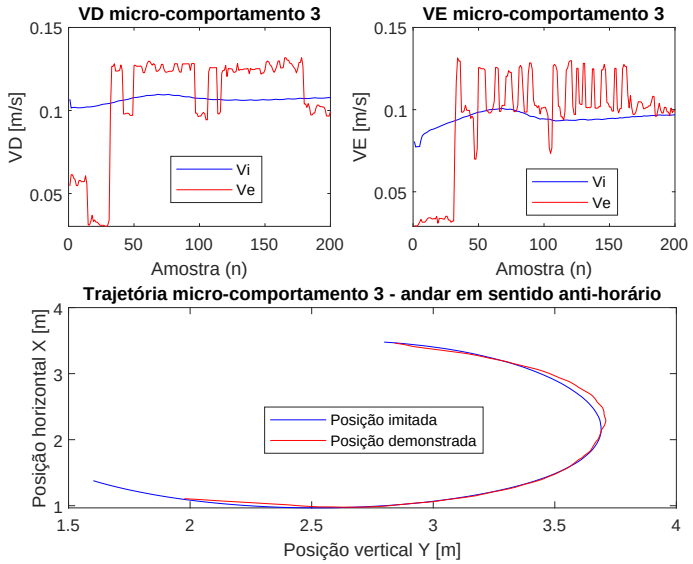


Figura 32 – Resultados de simulação do micro-comportamento 3, onde Vi é a velocidade imitada e Ve é a velocidade ensinada, para a topologia 1.

Os resultados mostrados acima foram obtidos utilizando os pesos mostrados nas Tabelas 7, 8 e 9.

Tabela 7 – Pesos e vieses do micro- comportamento 1 para a topologia 1.

Pesos roda direita	-0.5461	-0.9347	0.0929	-0.2810	-0.4359	-0.3825	-0.6436
Pesos roda esquerda	-0.5047	-0.5418	0.7710	0.3195	-0.8693	-0.5106	-0.3460
Bias roda direita	-0.2239						
Bias roda esquerda	-0.6227						

Tabela 8 – Pesos e vieses micro-comportamento 2 para a topologia 1.

Pesos roda direita	-0.0503	-0.3915	-0.0825	-0.3969	-0.7159	-0.3301	0.0996
Pesos roda esquerda	-0.7695	-0.4915	-0.0612	0.8202	-0.6141	0.0538	0.5201
Bias roda direita	-0.6699						
Bias roda esquerda	-0.4271						

Tabela 9 – Pesos e vieses micro-comportamento 3 para a topologia 1.

Pesos roda direita	-0.7738	0.0173	-0.6268	-0.5769	-0.4432	0.0598	-0.5260
Pesos roda esquerda	-0.9016	-0.2836	-0.5728	-0.0304	-0.4273	-0.5026	-0.5559
Bias roda direita	-0.5174						
Bias roda esquerda	-0.5738						

4.2.1.2 Topologia 2 - RNARCC

Feito a simulação para a topologia 2 (Figura 19) nos cenários 1, 2 e 3, foram obtidos os resultados mostrados na Tabela 10.

Tabela 10 – Erros médios quadráticos das velocidades imitadas com relação as velocidade demonstradas para cada micro-comportamento.

	VD$[(cm/s)^2]$	**VE** $[(cm/s)^2]$
Micro-comportamento 1	7.34e-04	7.87e-04
Micro-comportamento 2	2.30e-04	1.49e-04
Micro-comportamento 3	2.20e-04	2.24e-04

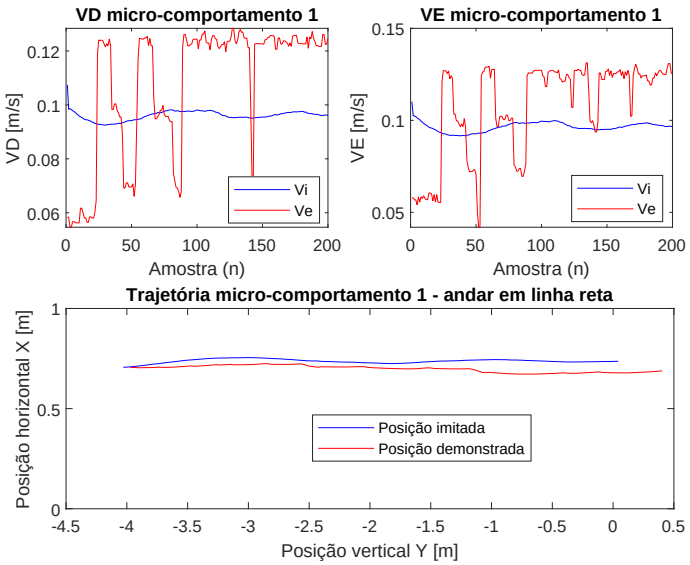


Figura 33 – Resultados de simulação do micro-comportamento 1, onde Vi é a velocidade imitada e Ve é a velocidade ensinada, para a topologia 2.

Os resultados mostrados acima foram obtidos utilizando os pesos mostrados na Tabela.

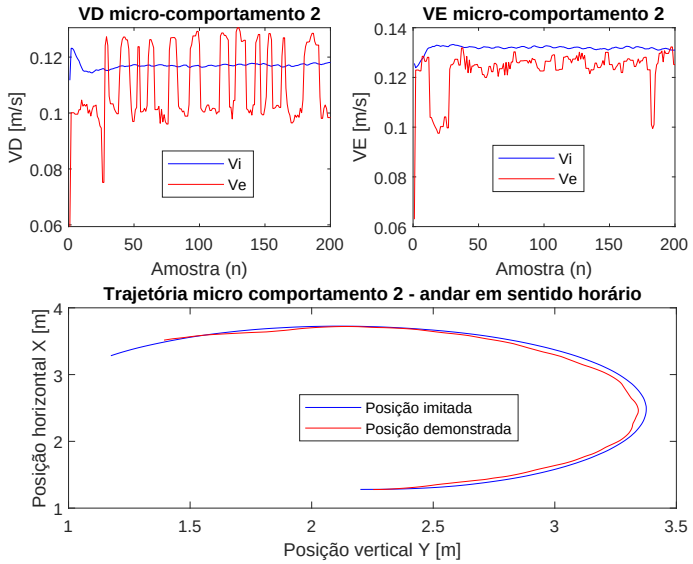


Figura 34 – Resultados de simulação do micro-comportamento 2, onde Vi é a velocidade imitada e Ve é a velocidade ensinada, para a topologia 2.

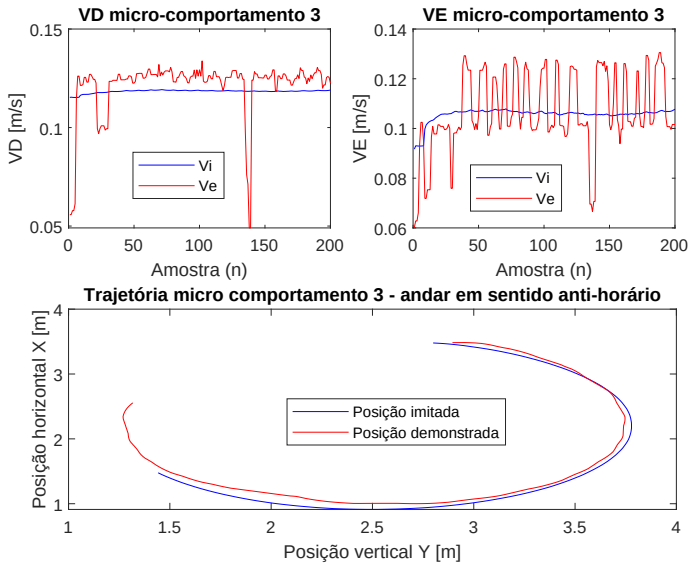


Figura 35 – Resultados de simulação do micro-comportamento 3, onde Vi é a velocidade imitada e Ve é a velocidade ensinada, para a topologia 2.

Os resultados mostrados acima foram obtidos utilizando os pesos mostrados nas Tabelas 11, 12 e 13.

Tabela 11 – Pesos e vieses micro-comportamento 1 topologia 2.

Pesos roda direita	-0.4967	-0.7326	0.0681	-0.5305	-0.3532	-0.3091	-0.4930	-0.3389
Pesos roda esquerda	-0.5963	-0.9847	-0.1012	0.4601	-0.1057	-0.6673	-0.6206	0.4298
Viés roda direita	-0.6341							
Viés roda esquerda	-0.6151							

Tabela 12 – Pesos e vieses micro-comportamento 2 topologia 2.

Pesos roda direita	-0.0558	-0.7436	-0.3313	-0.4065	0.1887	-0.9192	0.4050	0.5071
Pesos roda esquerda	-0.9210	-0.8614	0.2535	0.8321	0.0208	-0.6204	-0.1685	0.5178
Viés roda direita	-0.5496							
Viés roda esquerda	-0.0352							

Tabela 13 – Pesos e vieses micro-comportamento 3 topologia 2.

Pesos roda direita	-0.5438	-0.0335	-0.6556	-0.6268	-0.8380	-0.6041	0.0205	0.0054
Pesos roda esquerda	-0.6429	-0.3039	-0.8165	-0.6456	0.0790	-0.2826	0.1982	0.1676
Viés roda direita	0.5803							
Viés roda esquerda	-0.3792							

4.2.1.3 Simulação com o *referee*

Como a topologia da rede neural do *referee* não foi modificada, podemos utilizar os mesmos pesos e vieses para as duas topologias de redes neurais do micro-comportamentos. Nas Figuras 36 e 37 é mostrado a trajetória ensinada, a trajetória imitada e o padrão gerado para o Referee, respectivamente.

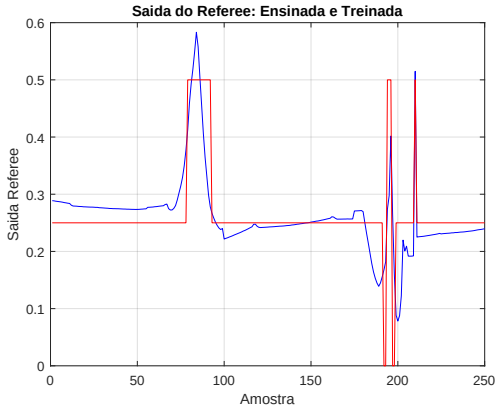
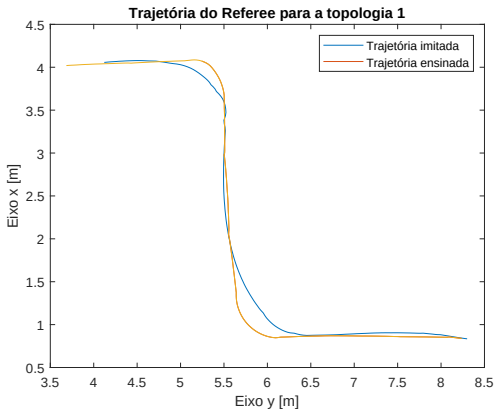
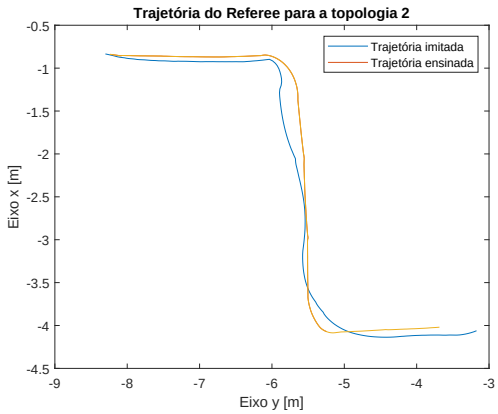


Figura 36 – Padrão gerado e o padrão treinado.



(a) Trajetória do referee para a topologia 1.



(b) Trajetória do referee para a topologia 2.

Figura 37 – Trajetórias do referee ensinadas e imitadas em cada topologia de rede neural recorrente.

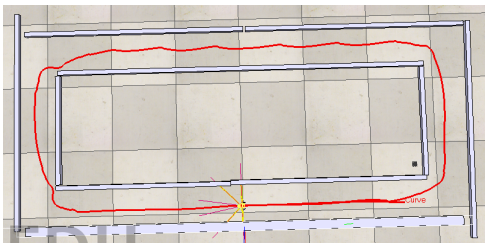
Os resultados mostrados acima foram obtidos utilizando os pesos mostrados na Tabela 14.

Tabela 14 – Pesos e vieses do neurônio do *referee*.

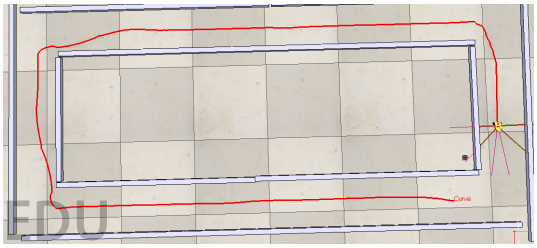
Peso	1.0156	-0.8691	0.2829	1.7362	-3.6098	2.3651
Viés	-0.7229					

4.2.1.4 Simulação em cenários desconhecidos

Nas Figuras 38, 39, 40, 41 e 42 são mostrados os resultados da topologia 1 e 2 em cenários desconhecidos. Observe que no primeiro cenário (Figura 38) as duas topologias não colidem com as paredes, mas já nos cenários das Figuras 39, 40 e 42, o robô colide com as paredes e obstáculos. No cenário da Figura 40 a topologia 1 chega ao final do corredor em "V", mas colide. Contudo para este mesmo cenário a topologia 2 faz o caminho inverso e não colide.

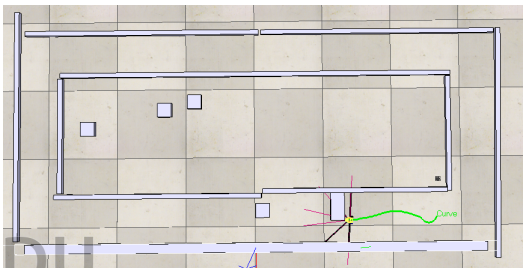


(a) Simulação do cenário desconhecido 1, para a topologia 1.

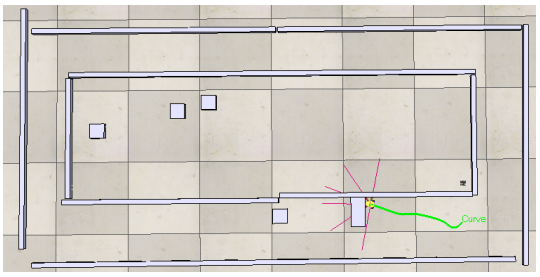


(b) Simulação do cenário desconhecido 1, para a topologia 2.

Figura 38 – Resultados das simulações, para as duas topologias no cenário desconhecido 1.

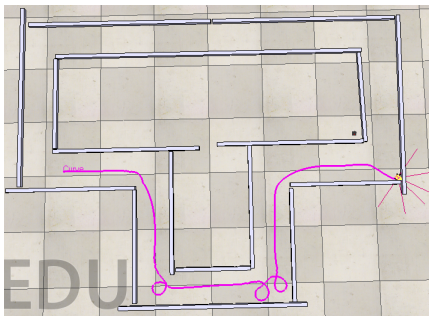


(a) Simulação do cenário desconhecido 2, para a topologia 1.

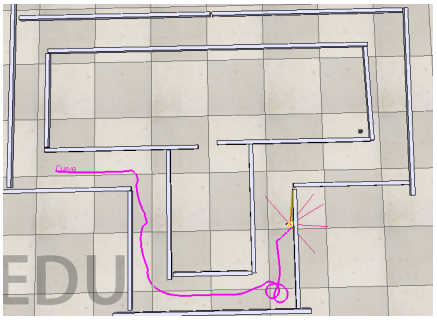


(b) Simulação do cenário desconhecido 2, para a topologia 2.

Figura 39 – Resultados das simulações, para as duas topologias no cenário desconhecido 2.

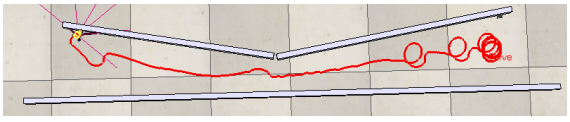


(a) Simulação do cenário desconhecido 3, para a topologia 1.

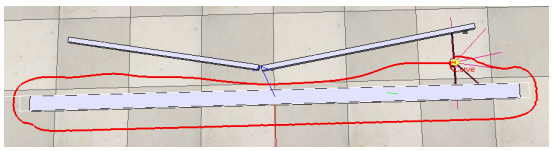


(b) Simulação do cenário desconhecido 3, para a topologia 2.

Figura 40 – Resultados das simulações, para as duas topologias no cenário desconhecido 3.

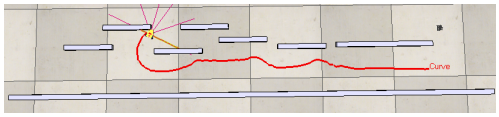


(a) Simulação do cenário desconhecido 4, para a topologia 1.

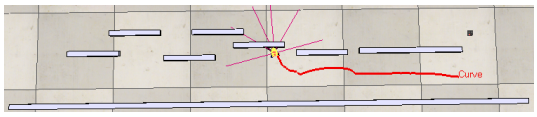


(b) Simulação do cenário desconhecido 4, para a topologia 2.

Figura 41 – Resultados das simulações, para as duas topologias no cenário desconhecido 4.



(a) Simulação do cenário desconhecido 5, para a topologia 1.



(b) Simulação do cenário desconhecido 5, para a topologia 2.

Figura 42 – Resultados das simulações, para as duas topologias no cenário desconhecido 5.

4.3 Hardware-in-the-loop

Utilizando a topologia de RNA recorrente mostrada na Figura 21 e usando a configuração mostrada na Figura 43 para a realização da simulação HIL, foram obtidos os resultados mostrados na Tabela 15.

Em comparação com os resultados mostrados na Tabela 6, o resultado do erro médio quadrático da simulação em software é 10 vezes menor, para os micro-comportamentos 2 e 3. Já no micro-comportamento 1 temos a mesma ordem de grandeza. Essa diferença, vem do fato que o computador tem uma precisão de 32 bits em ponto flutuante, mas em hardware foi implementado uma precisão de 16 bits em ponto flutuante. Contudo, em termos práticos a diferença não tem significância, sendo demonstrado que o robô apresentou bons resultados em simulação.

Tabela 15 – Erro médio quadrático entre a velocidade imitada e a velocidade ensinada na simulação HIL.

	VD$[(cm/s)^2]$	**VE**$[(cm/s)^2]$
Micro-comportamento 1	2.14e-3	2.68e-3
Micro-comportamento 2	2.67e-3	2.67e-3
Micro-comportamento 3	1.79e-3	8.15e-05

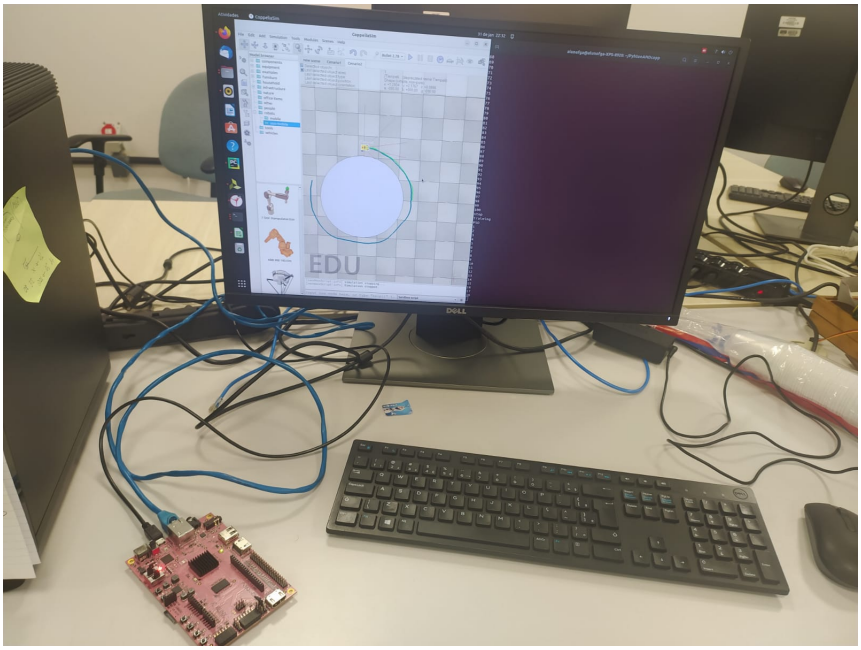


Figura 43 – Configuração para realização da simulação HIL.

4.4 Resultados dos testes físicos no robô MARIA

4.4.1 Implementação e testbench da RNARD

A fim de validar a topologia proposta, foi desenvolvido um IP, utilizando os IPs de ponto flutuante de ARBOLEDA (2012). Esses IPs tem a vantagem de serem parametrizados, podendo dessa forma mudar a precisão em bits da aplicação.

Para implementar a RNARD foi utilizado 16 bits em ponto flutuante, com 8 bits de expoente e 7 de mantissa, como citado anteriormente. Assim, desenvolveu-se um arquivo de teste para validar o IP desenvolvido em VHDL, o resultado é mostrado na Figura 44.

Observe na Figura 44, que temos o sinal de *clock* representado por *clk*, o sinal de saída representado por *y_out* e o sinal de entrada representado por *x_in*. Além disso, pode-se observar em destaque a latência do IP que é igual a 200ns, e o *througput* de 5 MFLOPS aproximadamente, para um clock de 100 MHz.

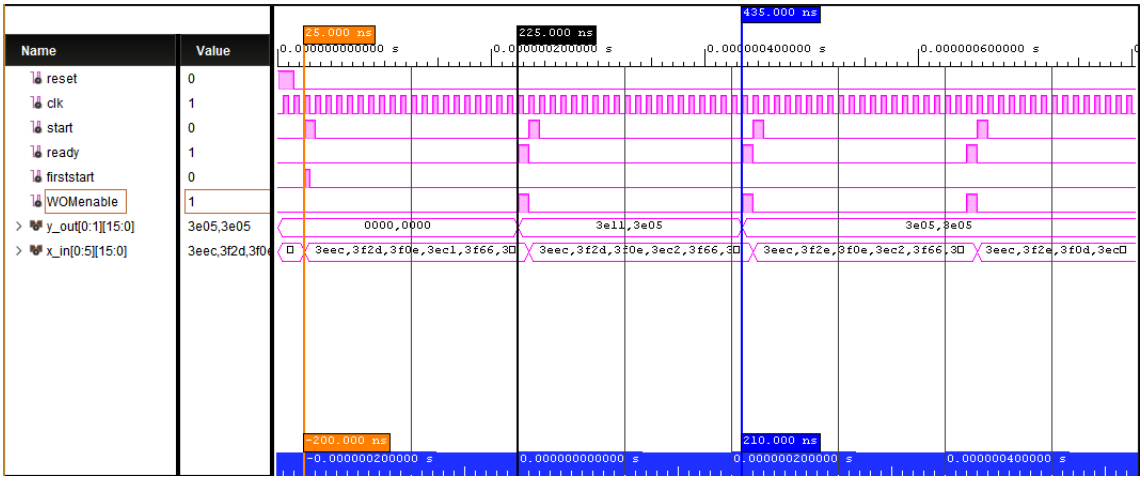


Figura 44 – Resultado da simulação temporal do IP da RNARD.

Para essa simulação foram usados 100 dados dos sensores obtidos da simulação descrita na seção 4.2.1 do micro-comportamento 1. O resultados das saídas foram armazenadas em um arquivo .txt para análise dos dados. Na figura 45 é mostrado o gráfico do erro médio quadrático de cada amostra processada em 16 bits, com relação aos resultados obtidos na simulação descrita na seção 4.2.1 a qual tem uma precisão numérica de 64 bits. O gráfico verde é mostra o erro do neurônio da roda direita e em vermelho o erro do neurônio da roda esquerda. O erro quadrático médio total para a roda direita é igual a 7.8094×10^{-5} e o erro total para a roda esquerda é igual a 5.4605×10^{-5}.

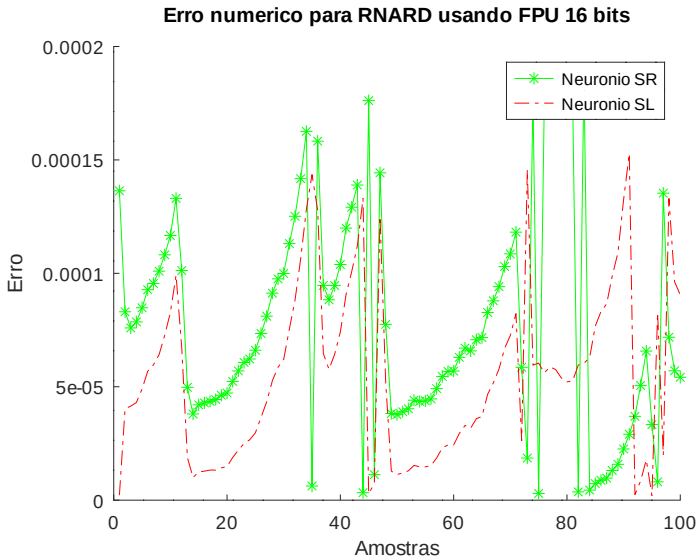


Figura 45 – Erro numérico da topologia RNARD com 16 *bits* em ponto flutuante.

Isto posto, o IP da RNARD foi implementado fisicamente substituindo a SLP do projeto desenvolvido anteriormente. Na próxima seção será descrito os resultados obtidos da implementação física do robô MARIA nos cenários de micro-comportamentos.

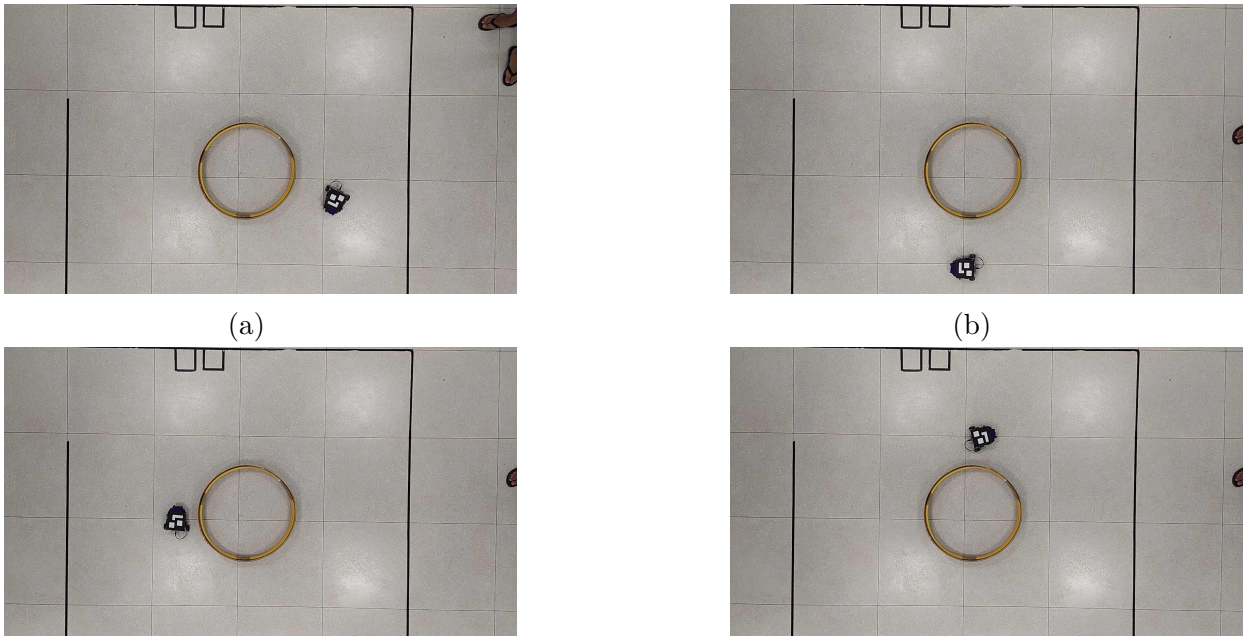
4.5 Implementação dos micro-comportamentos

Após as simulações e análise dos resultados em projeto em hardware, foram implementados os micro-comportamentos de andar no meio de um corredor, fazer curva no sentido horário e fazer curva no sentido anti-horário usando a RNARD. Os resultados obtidos são mostrados na figura 46, 47 e 48. Observe que o robô consegue executar as tarefas com exito, validando a topologia proposta neste trabalho e a capacidade de replicabilidade do trabalho de Triana (2022).

(a) (b)

(c) (d)

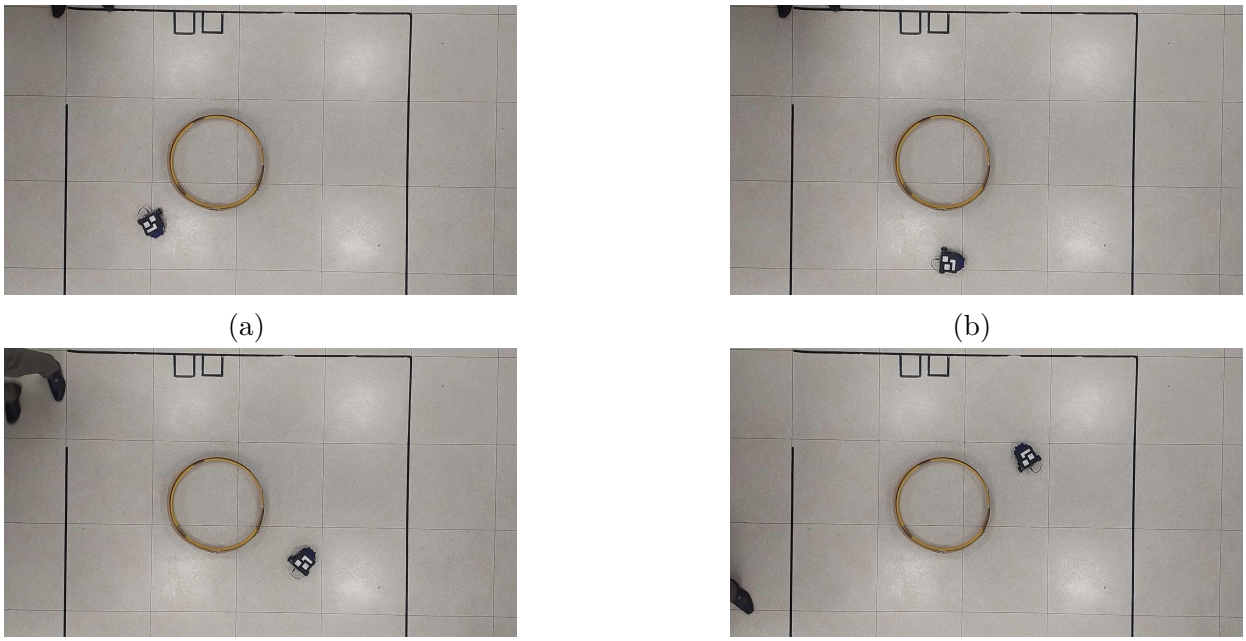
Figura 46 – Teste físico com o micro-comportamento 1.



(a) (b)

(c) (d)

Figura 47 – Teste físico com o micro-comportamento 2.



(a) (b)

(c) (d)

Figura 48 – Teste físico com o micro-comportamento 3.

Os pesos implementados nos testes das figuras 46, 47 e 48 foram os adquiridos na simulação (seção 4.2.1), com exceção do teste da figura 48, onde também foram implementados pesos obtidos em simulação contudo diferentes do mostrado na seção 4.2.1, os pesos implementados nesse teste é mostrado na tabela 16.

Tabela 16 – Pesos e vieses micro-comportamento 3 topologia 1, implementados no robô MARIA.

Pesos roda direita	-0.7866	0.0606	-0.7995	-0.4527	-0.0619	-0.0374	0.1146
Pesos roda esquerda	-0.7156	-0.4113	-0.6775	-0.6663	-0.7815	0.1511	-0.5757
Bias roda direita	-0.8046						
Bias roda esquerda	-0.1403						

4.5.1 Consumo de recursos e potência

Feita a implementação de todos os blocos necessários para o funcionamento do robô MARIA: controlador PI, leitor dos *encoders* de velocidade, conversor AD e do IP da RNARD descrito anteriormente, foram obtidos os valores de consumo de recursos lógicos mostrados na tabela 17.

Tabela 17 – Consumo de recursos lógicos pós-implementação em hardware.

MÓDULO	**LUTs (14.400)** %	**DSP (66)** %	**Flip-Flops (28.800)** %
PI-R	872 6,06%	0 0,0%	403 1,40%
PI-L	872 6,06%	0 0,0%	403 1,40%
REFEREE	1.631 11,33%	0 0,0%	882 3,06%
PWM-R	99 0,69%	0 0.0%	193 0,67%
PWM-L	99 0,69%	0 0.0%	193 0,67%
RNARD	3.539 24,58%	0 0,0%	2.190 7,60%
ENCONDER SR	139 0,97%	0 0,0%	278 0,97%
ENCONDER SL	139 0,97%	0 0,0%	278 0,97%
XADC	146 1.01%	0 0,0%	237 0,82%
OUTROS	1.247 8,65%	0 0,0%	836 2,88%
TOTAL	8.794 61,07%	0 0,0%	5.893 20,46%

Em comparação, o trabalho de Triana (2022) tem um consumo total de recursos lógicos de 94,88%, ou seja, reduzindo a precisão de bits de 27 pra 16 em ponto flutuante, tivemos uma redução de consumos lógicos de 33,81% de pontos percentuais. Além disso, o trabalho de Triana (2022) consome 18,18% de DSPs, contudo neste trabalho não foram

usados DSPs, as operações foram feitas todas em LUTs e Flip-Flops. O consumo de Flip-Flops também foi reduzido de 31,23% para 20,46%.

O consumo de potência estimado foi de 1.319W, sendo 1.200W de potência dinâmica e 0.118W de potência estática. Onde 0.007W é para geração de *clock*, 0.007W para sinais, 0.006W para a parte lógica, menos que 0.001W para portas de entrada e saída, 0.001W para o XADC e 1.178W para o ARM.

Com isso, se obteve um redução de consumo de potência total igual a 0,017W, ambos com baixa confiança de estimação. Na figura 49 é mostrado o *layout* final implementado na FPGA, onde o *referee* está em verde, os dois controladores PI das duas rodas estão em amarelo e a RNARD em rosa.

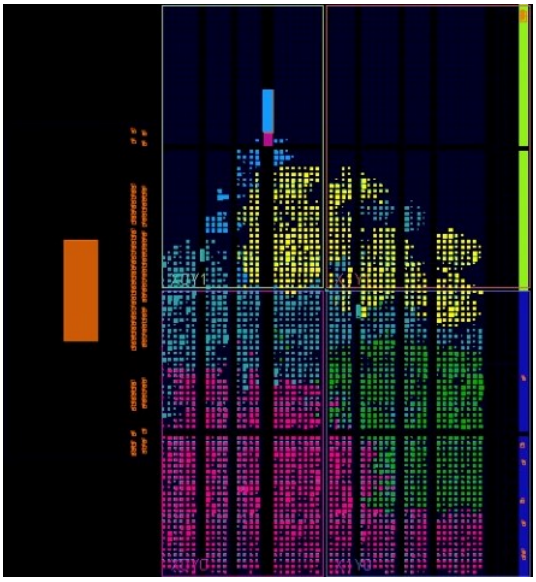


Figura 49 – *Layout* final do circuito implementado na FPGA.

4.6 Análise de resultados

4.6.1 Comparação entre os erros de velocidades imitadas em simulação

Na tabela 18 é mostrada a diferença em porcentagem entre o MSE obtido no trabalho de Triana (2022) com relação as duas topologias de RNAR avaliadas em simulação.

É possível observar que o erro de velocidade entre a topologia de RNA apresentada por Triana (2022) e as topologias avaliadas em simulação (RNARD e RNARCC), foi obtida uma redução percentual de erro de aproximadamente 90,00% para os micro-comportamentos 2 e 3, ou seja, para esses micro-comportamentos as RNARs foram superiores a RNA. Entretanto, as para o micro-comportamento 1, as RNARs tiverem um aumento de erro, para a topologia de RNARD de mais de 100% para a roda esquerda e 82,38% para a roda direita. Já para a topologia de RNARCC o erro foi aumentado em 15,40% para a roda direita e 97,24% para a roda esquerda.

Tabela 18 – Erro relativo entre os erros mostrados na tabela 6 e 10 , com relação ao erro do trabalho de Triana (2022).

	RNARD		RNACC	
	VD	VE	VD	VE
Micro-comportamento 1	-82,38%	-185,71%	-15,40%	-97,24%
Micro-comportamento 2	98,98%	99,66%	98,72%	99,47%
Micro-comportamento 3	96,29%	76,17%	98,81%	91,67%

4.6.2 Comparação entre o consumo de recursos lógicos

Como citado anteriormente, o trabalho de Triana (2022) tem 27 bits de precisão em ponto flutuante sendo 1 bits de sinal, 8 bits de expoente e 18 bits de mantissa, assim a fim de reduzir o consumo de recursos, este trabalho implementou todas as operações em 16 bits de precisão em ponto flutuante, sendo 1 bit de sinal, 8 de expoente e 7 de mantissa.

Dessa forma, pode-se reduzir o consumo de recursos em vários blocos como mostrado na tabela 19. Podendo se destacar para a redução de consumo de DSPs em todos os blocos e o consumo de LUTs e FFs nos blocos que são utilizados RNAs, ou seja, REFEREE e RNA, como mostrado na tabela 19.

Tabela 19 – Comparação entre o consumo de recursos lógicos obtidos no trabalho de Triana (2022) e neste trabalho.

	Triana (2022)			Este trabalho			Redução de consumo		
Módulos	LUTs	DSPs	FFs	LUTs	DSPs	FFs	LUTs	DSPs	FFs
PI-R	1.304	3	504	872	0	403	432	-3	101
PI-L	1.301	3	504	872	0	403	432	-3	101
REFEREE	2.526	2	1.146	1.631	0	882	895	-2	264
PWM-R	99	0	207	99	0	193	0	0	14
PWM-L	99	0	207	99	0	193	0	0	14
RNA	4.542	4	3.336	3.539	0	2.190	1.003	-4	1.146
ENCONDER SR	142	0	279	139	0	278	3	0	1
ENCONDER SL	142	0	279	139	0	278	3	0	1
XADC	NA	NA	NA	146	0	237	NA	NA	NA
OUTROS	3.046	0	2.066	1.247	0	836	1.799	0	1.230
Total	13.129	12	8.528	8.794	0	5.893	4.335	-12	2.635

5 Conclusões finais

As redes neurais artificiais oferecem uma ampla gama de possibilidades que podem ser exploradas, incluindo diferentes configurações e métodos de aprendizado. No entanto, quando se trata de aplicá-las em sistemas embarcados, surge o desafio de aproveitar essas topologias com recursos limitados, como demonstrado neste trabalho.

Portanto, é necessário utilizar topologias de redes neurais mais simples, uma vez que os sistemas embarcados têm recursos de processamento de dados limitados. Nesse sentido, neste trabalho foi abordada uma topologia simples de rede neural artificial recorrente, que se adapta bem a sistemas dinâmicos devido à sua realimentação das saídas.

Um ponto importante a ser destacado são os avanços em relação ao trabalho anterior. No trabalho anterior, a topologia implementada, juntamente com outros blocos, como o controlador PI, I2C e conversor serial, consumiam 90% dos recursos lógicos disponíveis no kit de desenvolvimento MiniZed. No entanto, neste trabalho, devido à redução do número de bits de precisão para 16 bits em ponto flutuante, apenas 60% dos recursos lógicos da MiniZed foram ocupados.

Com isso, foi possível explorar o paralelismo das redes neurais artificiais. Além disso, é importante ressaltar que não houve consumo de DSPs, pois os IP utilizados possuíam precisão de 16 bits, enquanto as DSPs disponíveis no SoC FPGA Zynq-XC7Z007S possuem 25x18 bits. Dessa forma, todas as operações foram implementadas com LUTs e Flip-Flops.

Outro avanço importante foi a utilização da técnica de simulação HIL, que permitiu validar a topologia proposta antes de integrá-la ao sistema. Essa técnica pode ser aproveitada em outros trabalhos, especialmente quando a planta na qual se deseja atuar não está disponível ou é inacessível para realizar testes com o controlador proposto. No caso deste trabalho, a planta foi modelada no software de simulação *CoppeliaSim*, o que possibilitou a validação da topologia utilizando HIL.

Alguns dos desafios encontrados foram principalmente na implementação da topologia 1 (RNARD) no robô físico utilizando os pesos obtidos em simulação. Isso se deve a alguns fatores: em simulação os sensores são lineares, o que não é verdade na implementação física, outro ponto de divergência se deve ao atrito do robô com o ambiente que também não é modelado na simulação, justificando assim, a dificuldade em implementar os pesos obtidos em simulação no robô físico.

Em suma, os resultados alcançados neste trabalho mostram que é viável utilizar topologias simples de redes neurais artificiais em sistemas embarcados, mesmo com recur-

sos limitados. Além disso, a utilização de técnicas como a redução da precisão dos bits e a simulação HIL mostraram-se eficientes para otimizar o uso dos recursos e validar a topologia proposta. Essas contribuições podem ser aplicadas em futuros trabalhos e ajudar no avanço da área de redes neurais aplicadas à robótica e sistemas embarcados.

5.0.1 Trabalhos futuros

Com o objetivo de aprimorar o controle do robô MARIA, é possível implementar uma topologia na qual as entradas são realimentadas por meio de uma FIFO. Além disso, constatou-se, no trabalho anterior, a dificuldade de controlar o robô MARIA durante a fase de ensino. Para solucionar esse desafio, é viável considerar a aplicação da técnica de teleoperação assistida ou a realização de uma otimização do aplicativo utilizado para controlar o robô MARIA.

Devido a redução de consumo de recursos lógicos, em consequência da redução do numero de bits de precisão, pode ser implementado em trabalhos futuros redes neurais mais complexas, explorando diferentes topologias e funções de ativação.

Por fim, neste trabalho não foi explorado a capacidade de reconfiguração dinâmica das FPGAs, em trabalhos futuros isso poderia ser implementado de forma que cada configuração execute uma tarefa, por exemplo: um *bitstream* para o treinamento da rede neural e outra para a implementação da rede neural.

Referências

ACADEMY, D. S. *Deep Learning Book*. [s.n.], 2022. Disponível em: <https://www.deeplearningbook.com.br/arquitetura-de-redes-neurais-long-short-term-memory/>. Citado 2 vezes nas páginas 8 e 22.

AHMAD, A.; ISMAIL, S.; SAMAON, D. Recurrent neural network with backpropagation through time for speech recognition. In: *IEEE International Symposium on Communications and Information Technology, 2004. ISCIT 2004.* [S.l.: s.n.], 2004. v. 1, p. 98–102 vol.1. Citado na página 22.

AN, H.; WANG, Y.; WANG, G. Event-triggered adaptive control of hypersonic vehicles subject to actuator nonlinearities. *Aerospace Science and Technology*, v. 133, p. 108119, 2023. ISSN 1270-9638. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1270963823000160>. Citado na página 15.

ARBOLEDA, D. M. M. *OTIMIZAC¸ AO POR INTELIG ~ ENCIA DE ENXAMES ^ USANDO ARQUITETURAS PARALELAS PARA APLICAC¸ OES EMBARCADAS*. Tese (Doutorado), Brasília, BR, 2012. Citado na página 53.

BAKKER, P.; KUNIYOSHI, Y. Robot see, robot do : An overview of robot imitation. *AISB96 Workshop on Learning in Robots and Animals*, 05 1996. Citado na página 19.

BILLARD, A.; GROLLMAN, D. Robot learning by demonstration. *Scholarpedia*, v. 8, n. 12, p. 3824, 2013. Revision #138061. Citado na página 19.

BORGES, J. *Redes neurais artificiais aplicadas em aprendizagem de trajetória em robótica móvel*. Dissertação (Mestrado) — Universidade de Brasília, Brasília, 12 2020. An optional note. Citado na página 27.

BRAVO, E. F. C.; SOTELO, J. A. L. *Redes neuronales artificiales*. 1st. ed. Cali, Colombia: Programa Editorial Universidad del Valle, 2010. ISBN 000-000-0000-00-0. Citado 3 vezes nas páginas 8, 20 e 21.

CARLUCHO, I.; De Paula, M.; ACOSTA, G. G. An adaptive deep reinforcement learning approach for mimo pid control of mobile robots. *ISA Transactions*, v. 102, p. 280–294, 2020. ISSN 0019-0578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0019057820300781>. Citado na página 15.

CHEN, S.; BAI, W. Performance analysis of typical linear augmented observers for a class of mimo systems with nonlinear uncertainty. *ISA Transactions*, v. 128, p. 316–327, 2022. ISSN 0019-0578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0019057821005930>. Citado na página 15.

CHERRAGUI, H.; HILAIRET, M.; GIURGEA, S. Hardware-in-the-loop simulation of a boost converter with the xilinx system generator from matlab/simulink. In: *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*. [S.l.: s.n.], 2015. p. 001837–001842. Citado na página 38.

CHU, P. P. *FPGA prototyping by VHDL examples*. United States of America: WILEY-INTERSCIENCE, 2008. ISBN 978-0-470-18531-5. Citado na página 24.

EBERHART, R.; KENNEDY, J. A new optimizer using particle swarm theory. In: *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. [S.l.: s.n.], 1995. p. 39–43. Citado 2 vezes nas páginas 8 e 23.

FRANCO, R. A. et al. Mimo auto-regressive modeling-based generalized predictive control for grid-connected hybrid systems. *Computers & Electrical Engineering*, v. 97, p. 107636, 2022. ISSN 0045-7906. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0045790621005620>. Citado na página 15.

GAMAL, O.; CAI, X.; ROTH, H. Learning from fuzzy system demonstration: Autonomous navigation of mobile robot in static indoor environment using multimodal deep learning. In: *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*. [S.l.: s.n.], 2020. p. 218–225. Citado na página 27.

GARCIA, D. et al. Learning from demonstration with gaussian process approach for an omni-directional mobile robot. *IEEE Latin America Transactions*, v. 16, n. 4, p. 1250–1255, 2018. Citado na página 27.

HACENE, N.; MENDIL, B. Fuzzy behavior-based control of three wheeled omnidirectional mobile robot. *International Journal of Automation and Computing*, v. 16, p. 163–185, 2019. ISSN 1751-8520. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494619301826>. Citado na página 15.

HECHT-NIELSEN, R. Iii.3 - theory of the backpropagation neural network**based on "nonindent" by robert hecht-nielsen, which appeared in proceedings of the international joint conference on neural networks 1, 593–611, june 1989. © 1989 ieee. In: WECHSLER, H. (Ed.). *Neural Networks for Perception*. Academic Press, 1992. p. 65–93. ISBN 978-0-12-741252-8. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780127412528500108>. Citado na página 22.

JIANG, L. et al. Personalize vison-based human following for mobile robots by learning from human-driven demonstrations. In: *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. [S.l.: s.n.], 2018. p. 726–731. Citado na página 27.

JUANG, J.-G.; LIN, R.-W.; LIU, W.-K. Comparison of classical control and intelligent control for a mimo system. *Applied Mathematics and Computation*, v. 205, n. 2, p. 778–791, 2008. ISSN 0096-3003. Special Issue on Advanced Intelligent Computing Theory and Methodology in Applied Mathematics and Computation. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0096300308003561>. Citado na página 15.

KENNEDY, J.; EBERHART, R. Particle swarm optimization. In: *Proceedings of ICNN'95 - International Conference on Neural Networks*. [S.l.: s.n.], 1995. v. 4, p. 1942–1948 vol.4. Citado na página 22.

KU, C.-C.; LEE, K. Diagonal recurrent neural networks for dynamic systems control. *IEEE Transactions on Neural Networks*, v. 6, n. 1, p. 144–156, 1995. Citado na página 36.

LIU, J. et al. Spiking neural network-based multi-task autonomous learning for mobile robots. *Engineering Applications of Artificial Intelligence*, v. 104, p. 104362, 2021. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197621002104>. Citado na página 27.

LIU, Y.; ZHU, Q. Adaptive neural network asymptotic control design for mimo nonlinear systems based on event-triggered mechanism. *Information Sciences*, v. 603, p. 91–105, 2022. ISSN 0020-0255. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0020025522004005>. Citado na página 15.

LV, Y. et al. Inverse-model-based iterative learning control for unknown mimo nonlinear system with neural network. *Neurocomputing*, v. 519, p. 187–193, 2023. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231222014254>. Citado na página 15.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, v. 5, p. 115–133, 1943. Citado na página 19.

MOHAMED, O. et al. Learning of embodied interaction dynamics with recurrent neural networks: some exploratory experiments. *J Neural Eng*, 2014. Disponível em: <https://pubmed.ncbi.nlm.nih.gov/24658453/>. Citado 2 vezes nas páginas 26 e 27.

MOHANTA, J.; KESHARI, A. A knowledge based fuzzy-probabilistic roadmap method for mobile robot navigation. *Applied Soft Computing*, v. 79, p. 391–409, 2019. ISSN 1568-4946. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1568494619301826>. Citado na página 15.

MUñOZ, D. M. et al. Hardware opposition-based pso applied to mobile robot controllers. *Engineering Applications of Artificial Intelligence*, v. 28, p. 64–77, 2014. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197613002376>. Citado na página 27.

PETROV, P.; GEORGIEVA, V. Adaptive velocity control for a differential drive mobile robot. In: *2018 20th International Symposium on Electrical Apparatus and Technologies (SIELA)*. [S.l.: s.n.], 2018. p. 1–4. Citado na página 15.

PHAM, L. H. et al. A mobile vision-based system for gap and flush measuring between planar surfaces using aruco markers. In: *2021 International Conference on Electronics, Information, and Communication (ICEIC)*. [S.l.: s.n.], 2021. p. 1–4. Citado na página 41.

QIAN, K. et al. Environment-adaptive learning from demonstration for proactive assistance in human–robot collaborative tasks. *Robotics and Autonomous Systems*, v. 151, p. 104046, 2022. ISSN 0921-8890. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0921889022000185>. Citado na página 16.

SAKR, M. et al. Quantifying demonstration quality for robot learning and generalization. *IEEE Robotics and Automation Letters*, v. 7, n. 4, p. 9659–9666, 2022. Citado na página 16.

SHARP. *GP2Y0A21YK0F*. [S.l.], 2011. Citado 2 vezes nas páginas 8 e 31.

TAN, H. Applying an extension of estimation of distribution algorithm (eda) for mobile robots to learn motion patterns from demonstration. In: *2015 IEEE International Conference on Systems, Man, and Cybernetics*. [S.l.: s.n.], 2015. p. 2829–2834. Citado na página 27.

TREESATAYAPUN, C. Data-driven fault-tolerant control with fuzzy-rules equivalent model for a class of unknown discrete-time mimo systems and complex coupling. *Journal of Computational Science*, v. 63, p. 101827, 2022. ISSN 1877-7503. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877750322001892>. Citado na página 15.

TRIANA, M. A. P. *Controle Baseado em Aprendizagem por Demonstração Aplicado a Robótica Móvel usando Hardware Reconfigurável*. Brasília: Universidade de Brasília, 2022. Citado 18 vezes nas páginas 8, 9, 11, 17, 27, 28, 29, 32, 33, 34, 35, 38, 41, 42, 55, 57, 58 e 59.

VUKOVIć, N.; MITIć, M.; MILJKOVIć, Z. Trajectory learning and reproduction for differential drive mobile robots based on gmm/hmm and dynamic time warping using learning from demonstration framework. *Engineering Applications of Artificial Intelligence*, v. 45, p. 388–404, 2015. ISSN 0952-1976. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0952197615001499>. Citado na página 27.

WANG, D.; TAN, D.; LIU, L. Particle swarm optimization algorithm: an overview. *Soft Comput*, v. 22, p. 387– 408, 2018. Disponível em: <https://doi.org/10.1007/s00500-016-2474-6>. Citado na página 24.

WANG, H. et al. Robust h attitude tracking control of a quadrotor uav on so(3) via variation-based linearization and interval matrix approach. *ISA Transactions*, v. 87, p. 10–16, 2019. ISSN 0019-0578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0019057818304518>. Citado na página 15.

WANG, Y. et al. Optimised learning from demonstrations for collaborative robots. *Robotics and Computer-Integrated Manufacturing*, v. 71, p. 102169, 2021. ISSN 0736-5845. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0736584521000533>. Citado na página 16.

WEI, Y. et al. Acquiring robot navigation skill with knowledge learned from demonstration. In: *2021 IEEE International Conference on Development and Learning (ICDL)*. [S.l.: s.n.], 2021. p. 1–6. Citado na página 27.

WERBOS, P.; JOHN, P. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. 01 1974. Citado na página 22.

WYTHOFF, B. J. Backpropagation neural networks: A tutorial. *Chemometrics and Intelligent Laboratory Systems*, v. 18, n. 2, p. 115–155, 1993. ISSN 0169-7439. Disponível em: <https://www.sciencedirect.com/science/article/pii/016974399380052J>. Citado na página 22.

XU, S. et al. Robot trajectory tracking control using learning from demonstration method. *Neurocomputing*, v. 338, p. 249–261, 2019. ISSN 0925-2312. Disponível em: <https://www.sciencedirect.com/science/article/pii/S0925231219300785>. Citado na página 16.

ZHANG, W.; JING, Y. Active queue management algorithm based on rbf neural network controller. In: *2020 Chinese Control And Decision Conference (CCDC)*. [S.l.: s.n.], 2020. p. 2289–2293. Citado na página 15.