**UNIVERSIDADE SANTA CECÍLIA**
**PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA MECÂNICA**

**PAULINO MACHADO GOMES**

# SISTEMA NEURAL ARTIFICIAL PARACONSISTENTE DE APRENDIZADO POR DEMONSTRAÇÃO IMPLEMENTADO EM MICROCONTROLADOR

**SANTOS/SP**
**2022**

**PAULINO MACHADO GOMES**

# SISTEMA NEURAL ARTIFICIAL PARACONSISTENTE DE APRENDIZADO POR DEMONSTRAÇÃO IMPLEMENTADO EM MICROCONTROLADOR

Dissertação apresentada a Universidade Santa Cecília como parte dos requisitos para obtenção de título de mestre no Programa de Pós-Graduação em Engenharia Mecânica, sob a orientação do Prof. Dr. João Inácio da Silva Filho e Prof. Dr. Germano Lambert-Torres

**SANTOS/SP**

**2022**

Autorizo a reprodução parcial ou total deste trabalho, por qualquer que seja o processo, exclusivamente para fins acadêmicos e científicos.

511.31 GOMES, P. M.
Sistema Neural Artificial Paraconsistente de Aprendizado por Demonstração Implementado em Microcontrolador.
Paulino Machado Gomes/2022
92 páginas

Orientador(es): Prof. Dr. João Inácio da Silva Filho e Prof. Dr. Germano Lambert-Torres

Dissertação (Mestrado) - Universidade Santa Cecília, Programa de Pós-Graduação em Engenharia Mecânica, Santos, SP, 2022.

1. Célula Neural artificial. 2. Lógica Paraconsistente. 3. Algoritmo 4. Inteligência Artificial. 5. Aprendizado por Demonstração

I. Da Silva Filho, J. I., e Lambert-Torres, G.

**DEDICATÓRIA**

*Dedico este trabalho à minha esposa,*
*que teve paciência para suportar a minha ausência.*

# AGRADECIMENTOS

A Deus, por me guiar e encontrar o caminho certo e no final, estar onde devo estar.

Agradeço a Universidade Santa Cecília e todo o corpo docente, por me preparar e assistir durante esse período de aprendizagem da Pesquisa.

Agradeço ao meu orientador, Prof. Dr. João Inácio da Silva Filho, por compartilhar o seu conhecimento técnico, durante o caminho da minha Pesquisa.

Agradeço ao meu orientador, Prof. Dr. Germano Lambert-Torres, por acompanhar a elaboração da minha Pesquisa.

Agradeço em especial, ao meu amigo Leonardo Do Espírito Santos, por me motivar na minha Pesquisa.

## RESUMO

A presente dissertação tem por objetivo, apresentar uma modelagem baseada em Lógica Paraconsistente, para compor uma Célula Neural Artificial Paraconsistente de Aprendizagem – CNAPap, como um dispositivo físico implementado em um microcontrolador, e efetuar ensaios para investigar aplicações em robótica, utilizando técnicas de aprendizagem por demonstração. Desse modo, uma CNAPap foi construída a partir de uma Célula Paraconsistente básica, representada pelo algoritmo que descreve o reticulado associado à Lógica Paraconsistente Anotada, denominado de “Para-Analisador”, e inserido o seu modelo matemático originado do algoritmo, em um único *Chip*. Inicialmente o algoritmo da CNAPap, foi implementado por uma linguagem de programação, e configurado especialmente para um microcontrolador. Depois, foram efetuados testes na CNAPap onde, a partir dos resultados, foi então estruturada uma configuração programada em rede em um microcontrolador, compondo assim um Sistema Neural artificial paraconsistente de aprendizagem por demonstração - SNAPApdem. O programa computacional residente SNAPApdem, foi aplicado em um protótipo de um braço robótico articulado, com movimentação harmônica acionado por cinco motores CC, com o objetivo de, através do aprendizado por demonstração, dotar o dispositivo robótico da capacidade de aprender e efetuar tarefas. Diversos ensaios foram feitos, com resultados que demonstraram que a CNAPap foi capaz de ser programada em redes de recorrência, criando um SNAPApdem para aprender e guardar padrões, com características similares as do cérebro humano, como também fazer o braço robótico reproduzir tarefas, que foram anteriormente demonstradas. Estes resultados apresentados pelo SNAPApdem, abrem um amplo campo de pesquisa, visando o melhoramento desta técnica na área de robótica, e sistemas autônomos no que se trata, de inovações na técnica, da aprendizagem de máquinas por demonstração.



**Palavras Chave:** Célula Neural Artificial Paraconsistente, Lógica Paraconsistente, Algoritmo, Inteligência Artificial, Robótica, Aprendizagem por Demonstração.

## ABSTRACT

This dissertation aims to present a modeling based on Paraconsistent Logic, to compose a Paraconsistent Learning Artificial Neural Cell - CNAPap, as a physical device implemented in a microcontroller, and to carry out tests to investigate applications in robotics, using learning techniques by demonstration. Thus, a CNAPap was built from a basic Paraconsistent Cell, represented by the algorithm that describes the lattice associated with the Annotated Paraconsistent Logic, called “Para-Analyzer”, and inserted its mathematical model originated from the algorithm in a single Chip. Initially, the CNAPap algorithm was implemented by a programming language, and specially configured for a microcontroller. Afterwards, tests were carried out at CNAPap where, from the results, a programmed configuration in a network in a microcontroller was then structured, thus composing a paraconsistent artificial neural system for learning by demonstration - SNAPApdem. The resident computer program SNAPApdem was applied to a prototype of an articulated robotic arm, with harmonic movement driven by five DC motors, with the objective of, through demonstration learning, providing the robotic device with the ability to learn and perform tasks. Several tests were carried out, with results that showed that the CNAPap was able to be programmed in recurrence networks, creating a SNAPApdem to learn and store patterns, with characteristics similar to those of the human brain, as well as to make the robotic arm reproduce tasks, which were previously demonstrated. These results presented by SNAPApdem, open a wide field of research, aiming at the improvement of this technique in the area of robotics, and autonomous systems in terms of innovations in the technique, of machine learning by demonstration.



**Keywords:** Paraconsistent Artificial Neural Cell, Paraconsistent Logic, Algorithm, Artificial Intelligence, Robotics, Demonstration Learning.

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

## LISTA DE GRÁFICOS

## LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| *P* | Proposição |
| LP | Lógica Paraconsistente |
| LPA | Lógica Paraconsistente Anotada |
| V | Verdadeiro |
| F | Falso |
| LPA2v | Lógica Paraconsistente Anotada com anotação de dois valores |
| $G_C$ | Grau de Certeza |
| $G_{CT}$ | Grau de Contradição |
| $G_{CR}$ | Grau de Certeza Real |
| D | Distância |
| NAP | Nó de Análise Paraconsistente |
| CNAP | Célula Neural Artificial Paraconsistente |
| CNAPap | Célula Neural Artificial Paraconsistente de Aprendizagem |
| $F_{tCt}$ | Fator de tolerância a Contradição |
| $F_{tC}$ | Fator de tolerância a Certeza |
| $F_{tD}$ | Fator de tolerância a Decisão |
| $F_A$ | Fator de Aprendizagem |
| CNAPb | Célula Neural Artificial Paraconsistente |
| S1 | SAÍDA 1 |
| S2 | SAÍDA 2 |
| L*f*D | *Learning from Demonstration* |

## LISTA DE SÍMBOLOS

| | |
|---|---|
| $\top$ | Inconsistente |
| V | Verdadeiro |
| F | Falso |
| $\perp$ | Paracompleto ou Indeterminado |
| $\mu$ | Grau de evidência favorável |
| $\lambda$ | Grau de evidência desfavorável |
| $\varphi$ | Intervalo de certeza |
| $\mu_{\mathrm{E}}$ | Intervalo de evidência resultante |
| $\mu_{\mathrm{ER}}$ | Grau de evidência resultante real |
| $\mu_{\mathrm{CTR}}$ | Grau de contradição normalizado |
| $\sum$ | Totalizador de repetições |
| ⎍ | Dispositivo de acionamento |

**SUMÁRIO**

# 1. INTRODUÇÃO

Através da biociência compreendemos que, o cérebro humano é um emaranhado de conexões entre prolongamentos, de células nervosas chamadas neurônios. Esta extensa rede de células existente no cérebro, é muito complexa, mas de uma maneira simplificada, pode-se afirmar que o cérebro humano, é formado basicamente de conjuntos de células, que além da sua função biológica normal, possuem propriedades que as capacitam de processar, e transmitir informações (DA SILVA FILHO, 2008).

A unidade básica do cérebro humano, é denominada de Neurônio.
Dessa forma, podemos definir um neurônio biológico, como sendo uma célula especializada na transmissão de informações, que possui a capacidade de condução de mensagens nervosas, trocando informações entre si. Para as funções mentais, existem vários tipos de neurônios, no entanto, um neurônio típico pode ser estudado, dividindo-o em três partes principais, que podem ser vistas na Figura 1 (BRAGA, et al. 2000) (DA SILVA FILHO, 2008).

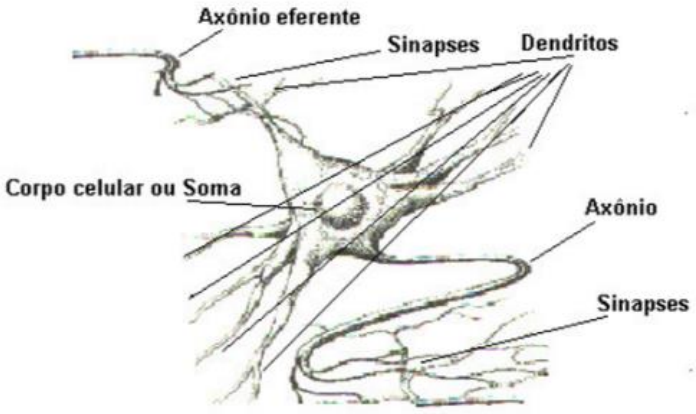



**Figura 1. Neurônio Biológico Típico**
**Fonte: Da Silva Filho (2008)**

a) A árvore dendrital, formada por dendritos que são prolongamentos filamentares, que fazem a recepção dos sinais de informações para as células;

b) O corpo celular ou soma, onde estão situados o núcleo celular e o citoplasma, sua função é reunir as informações recebidas pelos dendritos;

c) O axônio, que é uma projeção filamentar de diâmetro relativamente uniforme, e pode se apresentar com diversos comprimentos.

No cérebro humano, há uma complexa rede composta de bilhões de neurônios interconectados. O contato para a passagem de informação, entre os axônios e os dendritos, é devido às estruturas chamadas de Sinapses. Regiões com atividades eletroquímicas, capazes de fazer a conexão para transmissão das informações (HAYKIN, 1999) (DA SILVA FILHO, 2008).

A Sinapse, representada na Figura 2, pode ser dividida em duas partes separadas por uma região, chamada fenda sináptica. A parte anterior, que é composta pela membrana do axônio, pelo qual, chega o sinal de informação (pré-sináptica) na forma de um pulso elétrico, e uma parte posterior, que é composta pela membrana do dendrito (pós-sináptica), que receberá a informação. A informação, chegando na forma de um impulso elétrico, na membrana pré-sináptica, faz com que apareçam vesículas com mediadores químicos, chamados de neurotransmissores (BRAGA, et al. 2000) (HAYKIN, 1999).

Os neurotransmissores, alterando os seus formatos tridimensionais, difundem-se através da fenda sináptica, e agem nos canais químicos receptores da membrana pós-sináptica, iniciando assim uma série de eventos para a ocorrência da conexão (Da Silva Filho, 2008) (HAYKIN, 1999).

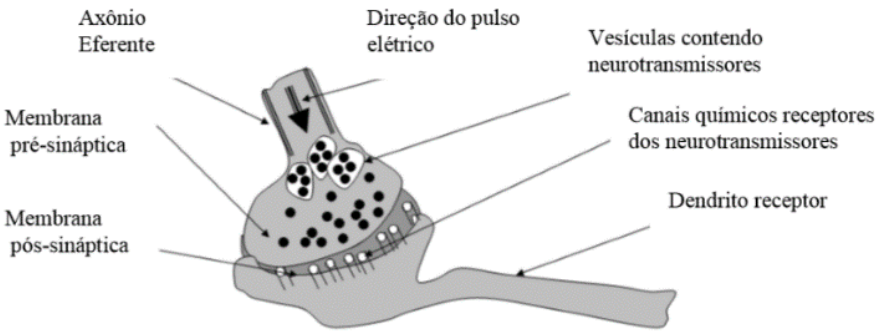



**Figura 2. Representação esquemática da Sinapse de recepção**
**Fonte: Da Silva Filho (2008)**

O neurônio biológico, recebe e envia informações através de conexões sinápticas, que ligam a árvore dendrital aos axônios, que trazem informações de outras células. A Sinapse propaga as informações em uma única direção, e dependendo dos tipos de neurotransmissores, podem facilitar ou inibir a formação de um *potencial de ação* no axônio do neurônio receptor.

O *potencial de ação,* é o responsável pelo envio de informações entre os neurônios, na forma de um impulso elétrico. Para gerar o chamado *potencial de ação,* na membrana axonal, os neurônios possuem propriedades físicas e eletroquímicas, que criam a depolarização ou a hiperpolarização, imposta pelos potenciais pré-sinápticos em cada Sinapse. Quando a membrana sofre uma depolarização suficientemente acentuada, á um certo limiar de disparo, gera impulso elétrico.

O Gráfico 1, mostra a forma de onda de um *potencial de ação*, que é um pulso gerado no axônio de um neurônio típico. A duração varia de um microsegundo, a um milisegundo.

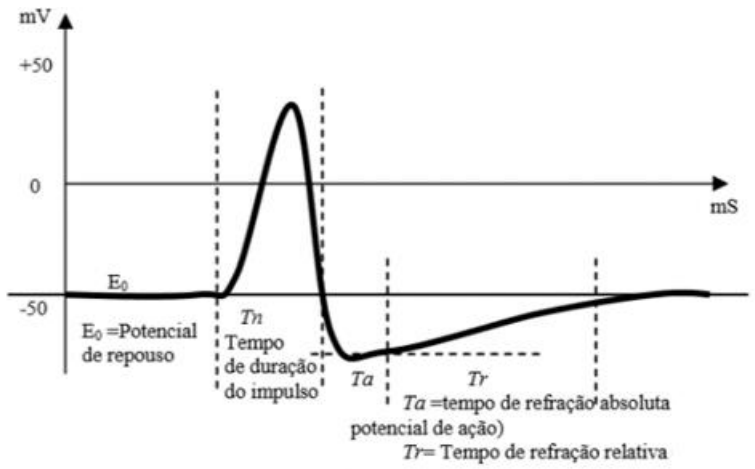



**Gráfico 1. Forma de onda do *potencial de ação***
**Fonte: Da Silva Filho (2008)**

No complexo processo eletroquímico que ocorre no interior do neurônio, quando os neurotransmissores se ligam aos receptores, provocam condições para que haja troca de cargas elétricas. Dependendo do processo eletroquímico, a absorção das informações pode variar, reforçando ou enfraquecendo a recepção. Este processo eletroquímico de variação na recepção de informações, que ocorre no núcleo celular, ainda não é completamente entendido, mas por seus efeitos, é considerado como uma memorização do neurônio.

As características funcionais, que modelam os dois princípios e outros efeitos, como a memorização, são determinadas por propriedades físico-químicas de cada célula, portanto, cada neurônio, Sinapse e neurotransmissor, também vão apresentar características próprias e particulares (HAYKIN, 1999) (DA SILVA FILHO, 2008).

## 1.1 Importância

Atualmente tem se investigado técnicas relacionadas ao aprendizado de máquina, que também é conhecida na área da Inteligência Artificial, de inteligência computacional. Este estudo tem sido direcionado a diversas áreas, como robótica, tradução automática, redes sociais, e-commerce, e até em áreas como a medicina e saúde, tem apresentado avanços significativos (BENGIO & LECUN, 2007).

A pesquisa em aprendizado de máquina, é um campo de estudo dentro da pesquisa em inteligência artificial. A sua ação, é buscar métodos de fornecer conhecimento aos computadores, através de dados, determinadas regras, algoritmos, observações e interações com o mundo. Assim, com esse conhecimento adquirido, os computadores ou máquinas robóticas, poderão aprender e reconhecer padrões, melhorando a sua interação, com os novos eventos e configurações (BENGIO &LECUN, 2007).

## 1.2 Problemática

O *aprendizado por demonstração*, também conhecido como "*programação por demonstração*", "*aprendizado por imitação*" e "*ensino por exibição*", recebeu atenção significativa na montagem automática de robôs nos últimos 20 anos. O objetivo, era substituir a programação manual de um robô, por um processo de programação automática, conduzido exclusivamente o mostrar ao robô, a tarefa de montagem por um especialista (CHERNOVA, 2009).

Em um sistema para o controle de robôs, algoritmos com alta complexidade, são criados e implementados, de modo que se realize movimentos com precisão. Entretanto, a cada mudança na trajetória do robô, ocorre a necessidade da intervenção de especialistas, para determinar um novo movimento, inviabilizando a flexibilidade para novas tarefas (Levine, 2016).

Dessa forma, a Aprendizagem por Demonstração, seria um método que proporcionaria maior rapidez na atribuição de movimentos a uma máquina, e fundamenta utilizar os conceitos de aprendizagem dos seres humanos, onde normalmente se obtém habilidades por meio de intervenção direta, observação, emulação de objetivos, imitação e outras interações sociais (KARLSSON, 2017).

Os estudos sobre Aprendizagem por Demonstração, se intensificaram quando, verificou-se que, com um alto nível de inovação, os métodos de aprendizado de máquina, mudariam somente através de um conjunto de informações anteriores sobre sua origem, e da existência de uma série de algoritmos, de onde se poderiam extrair uma estrutura de dados, e assim, minimizar o critério de erro.

Analisando, uma tarefa de aprendizagem geralmente seria bem menos definida, pois tem-se como objetivo, aprender as ações apropriadas em resposta a um estado percebido, afim de orientar um sistema dinâmico para realizar uma tarefa. Como a tarefa geralmente é um índice de desempenho arbitrário, não existem dados de treinamentos diretos, que possam ser usados para aprender (SCHAAL, 1997).

Para dificultar, no índice de desempenho definido, no comportamento de longo prazo da tarefa, surge um problema de como atribuir, creditar ou culpar as ações no passado, pelo desempenho atual. Nesse cenário, o aprendizado de uma tarefa do zero, pode exigir uma quantidade de tempo significativa, na exploração do espaço de ação do estado, para encontrar uma boa solução. Por esse problema, o aprendizado sem conhecimento prévio, é uma abordagem raramente adotada (SCHAAL, 1997).

Na tentativa reduzir a quantidade de tempo, na exploração do espaço de ação de estado, estudos especificos para este problema, mostraram que, o conhecimento de como abordar uma nova tarefa, pode ser transferido de tarefas aprendidas anteriormente, ou extraído do desempenho de um professor (SCHAAL, 1997).

Aprendizagem por Demonstração, é uma das técnicas conhecidas que apresenta novos métodos, trazendo inovação nos processos industriais automatizados, e assim, propõe que um robô aprenda com as demonstrações feitas por um “professor” (SCHAAL, 2008) (CHERNOVA, 2009).

Com o objetivo de tornar, novas implementações, relacionadas à programação de máquinas na área da indústria produtiva, o método da aprendizagem por demonstração, quando aplicada com outras técnicas já consolidadas, ligadas a Inteligência Artificial (IA), demonstra sucesso em robótica e automação (EWERTON, 2013) (HAYKIN, 2008).

## 1.3 Justificativa

O tema adotado para esta pesquisa, abrange um dos problemas mais importantes da área da Inteligência Artificial, que é a investigação de novos métodos de aprendizagem de máquinas, e as formas de sua aplicabilidade na área da robótica.

Portanto, a pesquisa teve seu foco, em investigar novas técnicas para aplicações dos processos de aprendizagem, por meio de demonstrações, e para isto, foram utilizados algoritmos fundamentados em Lógica Paraconsistente – LP, que formaram os tipos de células de aprendizagem.

A relevância e a inovação da pesquisa, está no fato de que se utilizou, algoritmos fundamentados em uma lógica não clássica, que é a Lógica Paraconsistente, com capacidade de tratar sinais incompletos e contraditórios.

São apresentadas novas alternativas em controle de automação, que requerem aprendizado de máquina para agregar um maior nível de eficiência em L*f*D, por meio da implementação de estruturas algorítmicas, baseadas em Lógica Paraconsistente – LPA.

Esta nova abordagem apresentou resultados compensadores, que poderão contribuir, para esta importante área de IA.

## 1.4 Objetivos

### 1.4.1 Objetivo Principal

Esta pesquisa teve como objetivo principal, apresentar um estudo de modelagem com Lógica Paraconsistente, para compor uma Célula Neural Artificial Paraconsistente de Aprendizagem – CNAPap, como um dispositivo físico implementado em um microcontrolador, e proporcionar condições técnicas para a aprendizagem por demonstração, em ensaios utilizando um braço robótico.

### 1.4.2 Objetivos Secundários

Como objetivos secundários, a pesquisa visou a implementação, via programação de algoritmos em um microcontrolador, para configurar um Sistema Neural Artificial Paraconsistente de Aprendizagem por demonstração - SNAPApdem.

Esta estrutura foi programada no microcontrolador, criando um dispositivo capaz de ser utilizado nos acionamentos de robôs, do tipo braço articulado. Dotando-os de capacidade para cumprir tarefas, após serem treinados, através de técnicas de aprendizagem por demonstração.

Portanto, como objetivos secundários, teve-se a construção de um sistema de decisão, capaz de aprender padrões por meio de repetições, ou de demonstrações, e atuar em um braço robótico, para efetuar as tarefas aprendidas interpretando estes dados, com funcionamento considerado similar, ao do cérebro humano.

## 1.5 Fundamentação Teórica

Neste capitulo serão apresentadas, as considerações fundamentais sobre a Lógica paraconsistentes, equações e seus algoritmos. Ainda neste capitulo, serão feitas as considerações sobre os métodos de aprendizagem por demonstração, e descritos os detalhes utilizados na pesquisa.

### 1.5.1 A Lógica Paraconsistente

A Lógica Clássica, iniciada na Grécia Antiga pelo filósofo Aristóteles, é atualmente a base para os modernos sistemas eletrônicos digitais, e computadores. A Lógica Clássica é estritamente binária, e aceita apenas dois estados possíveis, para uma proposição *P*. A sua aplicação em uma análise, obriga que a sua conclusão, aponte que a proposição é verdadeira (V), ou é falsa (F), não havendo espaço para meio termos ou contradições, que possam gerar incertezas.

No mundo real, as contradições, ambiguidades ou inconsistências, podem aparecer ao se tentar descrever situações, que se deseja analisar, assim, nestas situações, a Lógica Clássica se torna inadequada (DA SILVA FILHO, 2006).

As Lógicas Paraconsistentes (LP), surgiram da necessidade de se desenvolver ferramentas e metodologias, para tratar situações contraditórias ou inconsistentes. Estas lógicas, apresentam resultados que permitem considerar, as ambiguidades e contradições, em sua estrutura de maneira não trivial (DA SILVA FILHO, 2009).

### 1.5.2 Lógica Paraconsistente Anotada

A Lógica Paraconsistente Anotada – LPA, pertence a uma família de Lógicas Paraconsistentes, que possui a propriedade de ser associada a um reticulado, que pode ser o diagrama de Hasse, de quatro vértices. Nesse tipo de associação, são representados em seus vértices, os estados lógicos extremos de uma proposição (*P*), acompanhada por anotações pertencentes a um reticulado finito (DA COSTA e ABE, 1991) (SUBRAHMANIAN, 1987).

Cada anotação, está relacionada a um estado lógico extremo da proposição *P,* que por sua vez, está localizado no vértice do reticulado, tal que:
$\top$= Inconsistente, V= Verdadeiro, F= Falso e $\perp$= Paracompleto ou Indeterminado.

### 1.5.3 Lógica Paraconsistente Anotada com anotação de dois valores LPA2v

Na Lógica Paraconsistente Anotada, com Anotação de dois valores (LPA2v), a representação de evidências em um reticulado no plano real, é formada por pares de graus de evidência, que compõem a anotação (μ, λ), permitindo atingir um maior poder de representação.

Sendo assim, a proposição *P* é suportada pelo grau de evidência favorável, simbolizado por (μ), e pelo grau de evidência desfavorável, ou contrário a proposição *P,* simbolizado por (λ). Um operador "~", é introduzido e definido como segue (DA COSTA & ABE,1991):

$$\sim: |\tau| \rightarrow |\tau|, \text{ onde } \tau = \{(\mu, \lambda)\} \mid \epsilon\ \mu, \lambda\ [0,1] \subset \Re.$$

A associação de um par (μ, λ), para uma proposição *P, significa* que o grau de evidência favorável em *P* é μ, e o grau de evidência desfavorável é λ, conforme as anotações no reticulado (Da Costa; Abe,1991), (DA Cruz, 2015):

(1,0) → indica existência de evidência favorável total, e evidência desfavorável igual a zero, assinalando uma conotação lógica verdadeira para a proposição *P*.

(0,1) → indica a existência de evidência favorável igual a zero, e evidência desfavorável total, dando uma conotação de falsidade lógica para a proposição *P*.

(1,1) → indica a existência de ambos, evidência favorável e desfavorável total, atribuindo uma conotação lógica de inconsistência à proposição *P*.

(0,0) → indica a existência de ambos, evidência favorável e desfavorável zero, atribuindo uma conotação lógica de indeterminação para a proposição *P* (Da Costa e Abe, 1991).

A Figura 3, apresenta um reticulado representativo de quatro vértices, associado com a LPA2v (DA SILVA FILHO, 2007).

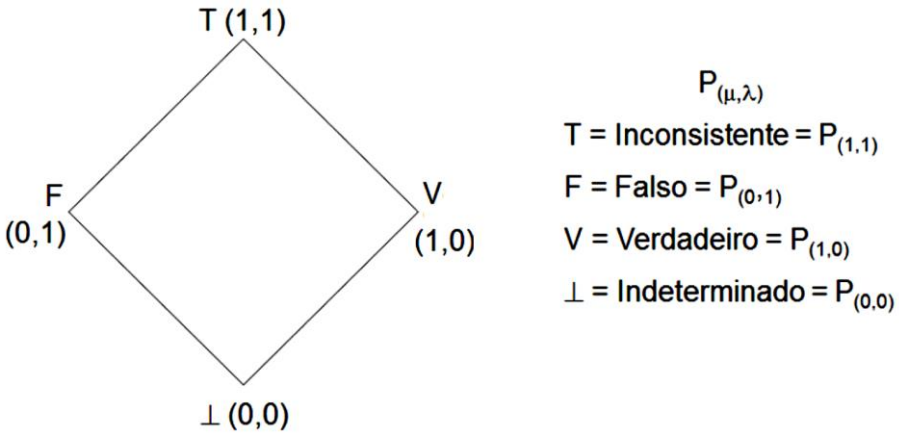



**Figura 3. Reticulado finito de Hasse associado à LPA2v.**
**Fonte: De Carvalho Jr, A., 2017**

Através de transformações lineares, entre um Quadrado Unitário no Plano Cartesiano (QUPC), e o reticulado associado à LPA, é possível se chegar às seguintes equações matemáticas, relacionadas à LPA2v (DA SILVA FILHO, 2007) apresentadas a seguir:

$$\lambda = 1 - \mu 2 \quad (1)$$

$$G_C = \mu 1 - \lambda \quad (2)$$

$$G_{CT} = \mu 1 + \lambda - 1 \quad (3)$$

$$\mu_E = \frac{G_C + 1}{2} \Rightarrow \mu_E = \frac{\mu - \lambda + 1}{2} \quad (4)$$

$$\mu_{CTR} = \frac{\mu + \lambda}{2} \quad (5)$$

Onde:

$\mu 1$ é o grau de evidência favorável, proveniente da primeira evidência;

$\lambda$ é o grau de evidência desfavorável, como o complemento da segunda evidência;

$G_C$ é o grau de certeza e;

$G_{CT}$ é o grau de contradição.

$\mu_E$ é o grau de evidência resultante de saída.

$\mu_{CTR}$ é o grau de contradição normalizado.

A análise paraconsistente, gera valores de $G_C$ e $G_{CT}$ entre 1 e -1. Para ser consistente com as entradas (μ, λ), em aplicações práticas eles precisam ser normalizados, e seus valores limitados novamente entre 0 e 1, para respeitar os limites da lógica paraconsistente. Portanto, o $\mu_E$ é o grau de evidência resultante, sendo a saída normalizada de $G_C$, enquanto que, $\mu_{CRT}$ é a saída normalizada de $G_{CT}$ (DA SILVA FILHO, 2007) (DE CARVALHO JR, 2017).

Conforme visto na equação (1), e equação (8), pode-se obter um resultado mais preciso, extraindo-se os efeitos da contradição em sucessivas análises, de modo a obter o grau de certeza resultante real $(G_{CR})$, como indicado na Figura 4, e apresentado nas equações matemáticas a seguir (DA SILVA FILHO, 2007):

$$D = \sqrt{(1 - |G_C|)^2 + {G_{CT}}^2} \tag{6}$$

$$\text{Se } G_C > 0 \rightarrow G_{CR} = 1 - D;$$
$$\text{Se } \quad G_C < 0 \rightarrow G_{CR} = D - 1; \tag{7}$$

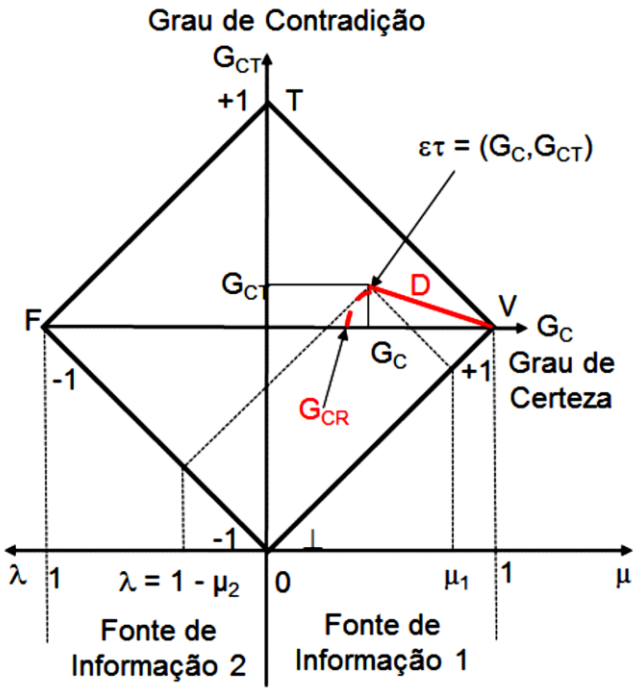



**Figura 4. Reticulado LPAv2 com Grau de certeza (GC), Grau de Contradição (GCT) e Grau de Certeza real (GCR). Fonte: De Carvalho Jr, A., 2017**

Sendo que, D é a distância entre o estado lógico paraconsistente - $\varepsilon\tau(G_C, G_{CT})$, e o vértice V (se $G_C > 0$) ou F (se $G_C < 0$); então $G_{CR}$, é o grau de certeza real; $\varphi$ é o intervalo de certeza; $\mu_{ER}$, é o grau de evidência resultante real ($G_{CR}$), normalizado com valores entre (0,1)); $\mu_{CTR}$, é o grau de contradição normalizado e $\varphi_E$, é o intervalo de evidência resultante.

Na equação (7), $G_{CR}$ é obtido após os efeitos da contradição serem removidos, sendo o resultado considerado, o de valor puro. O sinal de saída de um sistema paraconsistente de tratamento de incertezas, fornecerá o resultado de Grau de Certeza Resultante Real ($G_{CR}$), sendo representado pelos valores ($G_{CR}, \varphi(\pm)$). O grau de evidência resultante real ($\mu_{ER}$), é um valor mais consistente do que o grau de evidência resultante ($\mu_E$), conforme os efeitos da contradição são extraídos da análise.

Se o $G_{CT}$ tende a inconsistente, $\varphi$ é marcado como "+"; e se $G_{CT}$ tende a indeterminado, $\varphi$ é marcado como "-".

### 1.5.4 A Célula Neural Artificial Paraconsistente Padrão

Conforme apresentado em (DA SILVA FILHO, 2006), pode-se escrever o conjunto de equações matemáticas, da Lógica Paraconsistente Anotada, e suas interpretações para formar um algoritmo denominado de Nó de Análise Paraconsistente (NAP), sendo este o bloco básico de uma Célula Neural Artificial Paraconsistente (CNAP) (DE CARVALHO JR, 2017).

A Figura 5, mostra a representação do algoritmo de análise paraconsistente – LPA2v.

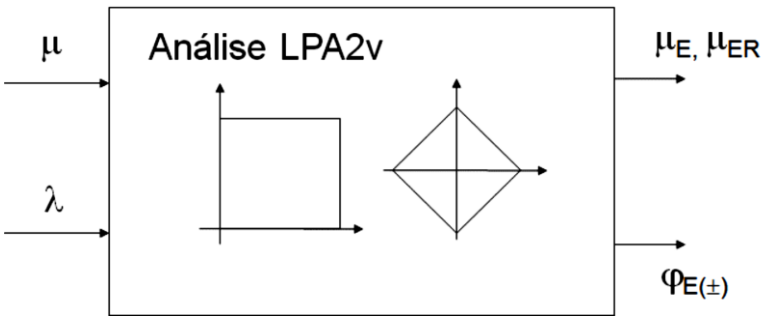



**Figura 5. Representação do algoritmo de Análise Paraconsistente LPA2v.**
**Fonte: Da Silva Filho, 2007**

A Figura 6, monstra o símbolo da CNAP padrão. São apresentadas as entradas ($F_{tCt}$ – Fator de Tolerância à Contradição, $F_{tC}$ – Fator de Tolerância à Certeza, $F_{tD}$ – Fator de Tolerância à Decisão, $F_A$ – Fator de Aprendizagem) são valores externos de ajuste entre (0,1), permitindo atribuir outros valores de limites superior e inferior para $G_C$, $G_{CT}$, $\varphi$ e taxa de aprendizagem. Estes fatores podem ou não serem utilizados, dependendo de como se aplica a CNAP e estão bem detalhados em (DA SILVA FILHO, 2007). O valor “1” é máximo e “0” é mínimo quando aplicado a cada fator.

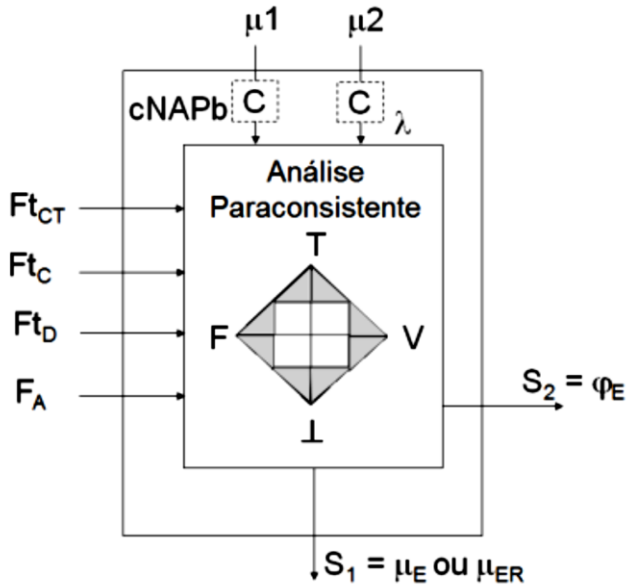



**Figura 6. Símbolo da CNAP padrão.**
**Fonte: De Carvalho Jr, A., 2017**

### 1.5.5 Célula Neural Artificial Paraconsistente de Aprendizagem

A célula Neural Artificial Paraconsistente de Aprendizagem – CNAPap, consiste basicamente de uma CNAPap, cuja saída $\mu_E$ é aplicada à entrada do grau de evidência desfavorável – $\lambda$ (DA CRUZ, 2015) (MARIO, 2007).

Dado que, a célula está sempre realizando a análise entre o grau de evidência favorável ($\mu$) atual, com o grau de evidência resultante ($\mu_E$), (ou grau de evidência resultante real ($\mu_{ER}$) anterior, há um sentido de relógio e a célula levará um tempo para apresentar o resultado final, que dependerá da precisão (casas decimais) desejada (DE CARVALHO JR, 2017).

A CNAPap recebe o nome de célula de aprendizagem, porque a saída tende a responder ou aprender, qualquer valor real no intervalo fechado [0,1], aplicado à entrada $\mu$.

### 1.5.6 O Fator de Aprendizagem FA

Ainda no processo de aprendizagem da CNAPap, é introduzido o fator de aprendizado $F_A$ , que é ajustado externamente.

O fator de aprendizagem $F_A$ ,funciona como um ajuste (ou ganho) do sinal realimentado, permitindo que a célula tenha uma aprendizagem mais rápida, ou mais lenta, conforme equações (11) a (12).

Dependendo do valor de $F_A$, será proporcionado uma aprendizagem mais rápida, ou mais lenta a CNAPap.

No processo de aprendizagem, considera-se uma equação para cálculos, de valores de graus de evidência resultantes sucessivos $\mu_{E(K)}$, até chegar ao valor 1. Portanto, para um valor de grau de evidência $\mu_{E(K)}$ inicial, são obtidos valores $\mu_{E(K+1)}$ até que $\mu_{E(K+1)}$ = 1.

Considerando um processo de aprendizagem do *padrão de verdade,* a equação de aprendizagem e obtida através da equação do cálculo do Grau de Evidencia resultante ficando (DE CARVALHO JR, 2017):

$$\mu_{E(K+1)} = \frac{\{\mu 1 - (\mu_{E(K)C})F_A\} + 1}{2} \qquad (8)$$

onde: $$\mu_{E(K)}C = 1 - \mu_{E(K)} \qquad (9)$$

sendo: $$0 \leq F_A \leq 1 \qquad (10)$$

Considera-se a Célula completamente treinada quando:

$$\mu_{E(K+1)} = 1. \qquad (11)$$

O fator de aprendizagem $F_A$ , é um valor real, dentro do intervalo fechado [0,1], atribuído arbitrariamente por ajustes externos. Conforme é visto pela equação do cálculo do Grau de Evidencia resultante $\mu_{E(K+1)}$, quanto maior é o seu valor, maior é a rapidez, de aprendizado da Célula.

O fluxograma e algoritmo de aprendizagem é apresentado na Figura 7.

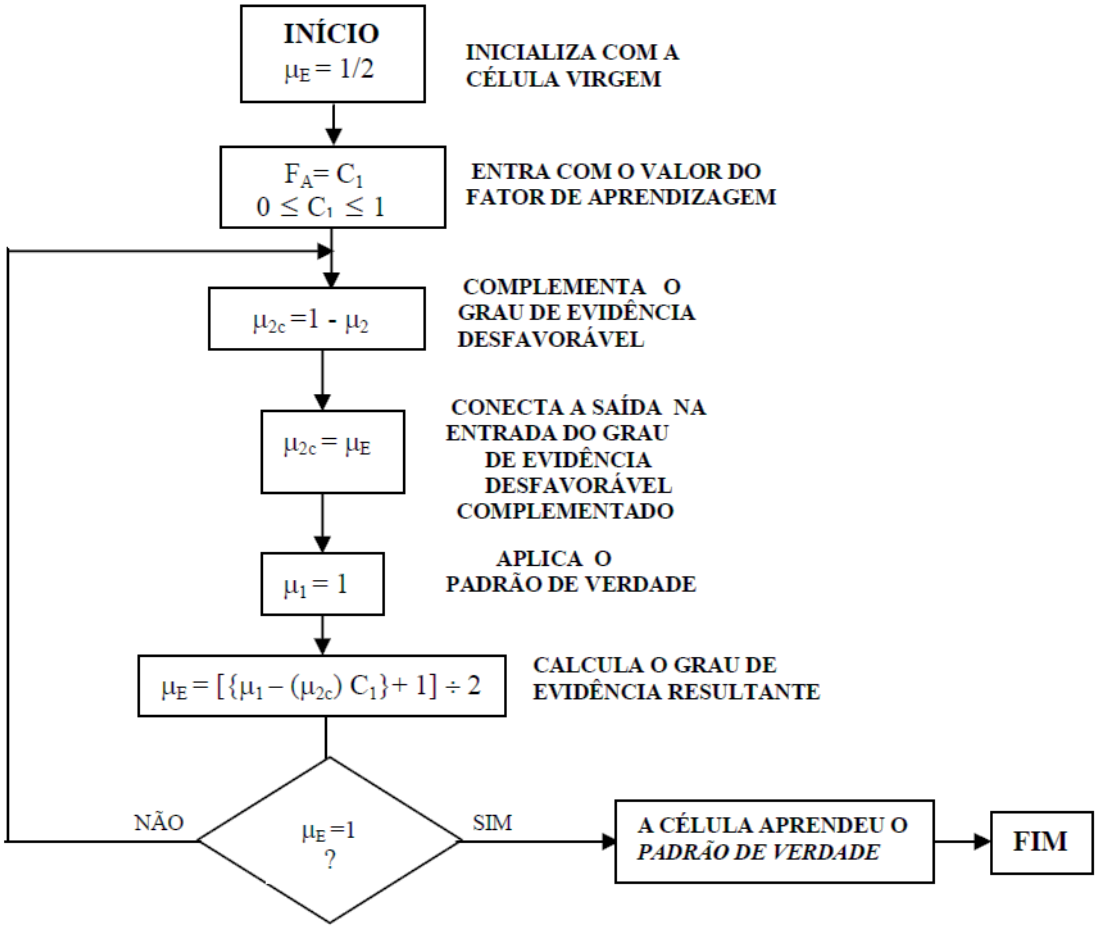



**Figura 7. Fluxograma para aprendizagem do *padrão de Verdade* da CNAPap**
**Fonte: Da Silva Filho, 2008**

Conforme as características, e os conceitos básicos da Logica Paraconsistente Anotada, no processo de aprendizagem do *padrão de verdade,* o padrão aplicado, é o de valor 1. As diferenças entre os valores da entrada, e da saída, são as contradições que somente se anularão, quando o Grau de Evidencia resultante chegar ao valor 1.

Portanto, este *padrão de verdade* sendo aplicado sucessivamente na entrada, permite que a equação de determinação do Grau de Evidencia resultante, vá diluindo as diferenças até que o seu valor na saída, chegue ao seu valor máximo 1.

A Figura 8, apresenta o símbolo da Célula, e os valores finais nas entradas e saída.

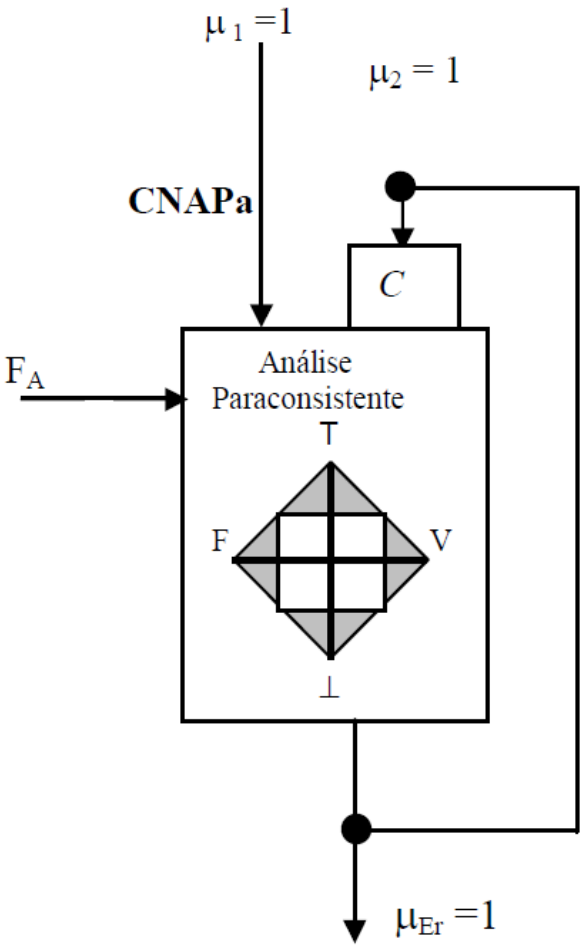



**Figura 8. Símbolo da Célula Neural Artificial Paraconsistente de Aprendizagem CNAPap**
**Fonte: Da Silva Filho, 2008**

O algoritmo que elabora a aprendizagem do *padrão de verdade*, onde os passos de 2 ao 5, são extraídos do Algoritmo que representa a Célula Neural Artificial Paraconsistente padrão CNAPp, e apresentado a seguir.

### 1.5.7 Algoritmo de aprendizagem da Célula Neural Artificial Paraconsistente – CNAPap (padrão de verdade)

**1-** Condição inicial

$\boldsymbol{\mu 1 = 1/2 \ \ e \ \ \mu_E = \mu 2 = 1/2}$ (12)

**2-** Entre com o valor do Fator de Aprendizagem

$\boldsymbol{F_A = C4}$ */ Fator de Aprendizagem $\boldsymbol{0 \leq \ F_A \ \leq 1}$ */ (13)

**3-** Transforme o Grau de Evidência 2 em Grau de Evidência Desfavorável

$\boldsymbol{\lambda = 1 - \mu_2}$ */ Grau de Evidência Desfavorável $\boldsymbol{0 \leq \ \lambda_2 \ \leq 1}$ */ (14)

**4-** Entre com o Grau de Evidência de entrada 1

$\boldsymbol{\mu_1 \ = 1}$ */ Grau de Evidência 1 (15)

**5-** Calcule o Grau de Evidência Resultante

$\boldsymbol{\mu_E = \frac{(\mu_1 + \lambda_2)+1}{2}}$ (16)

**6-** Considere a condicional

Se $\boldsymbol{\mu_E \neq 1}$ faça $\boldsymbol{\mu_E = 2}$ e retorne ao passo 3 (17)

**7-** Pare

### 1.5.8 Resultados consolidados obtidos no treinamento de uma Célula Neural Artificial Paraconsistente de Aprendizagem – CNAPap

Uma célula de aprendizagem, pode ser treinada para aprender *padrões de Verdade*. Para o aprendizado do *padrão de Verdade*, a equação é:

$$\mu_{E(K+1)} = \frac{\{\mu 1 - (\mu_{E(K)C})F_A\}+1}{2} \quad (18)$$

onde: $\mu_{E(K)}C = 1\text{-}\ \mu_{E(K)}$ (19)

sendo: $0 \leq F_A \leq 1$ (20)

**Tabela 1. Tabela resultante da Aplicação do Algoritmo da CNAPap.**

| Passos | Padrão= $\mu_1$ | $G_c=\mu_1-\mu_{2c}$ | $G_{ct}=(\mu_1+\mu_{2c})-1$ | $\mu_E=\{(G_c \times C_1)+1\} \div 2$ | observações |
|---|---|---|---|---|---|
| **0** | | **0,0000000000** | **-1,000000000** | **0,5000000000** | **Início** |
| 1 | 1,0 | 0,5000000000 | 0,500000000 | 0,7500000000 | |
| 2 | 1,0 | 0,7500000000 | 0,250000000 | 0,8750000000 | Aprendizagem |
| 3 | 1,0 | 0,8750000000 | 0,125000000 | 0,9375000000 | |
| 4 | 1,0 | 0,9375000000 | 0,062500000 | 0,9687500000 | |
| 5 | 1,0 | 0,9687500000 | 0,031250000 | 0,9843750000 | |
| 6 | 1,0 | 0,9843750000 | 0,015625000 | 0,9921875000 | |
| 7 | 1,0 | 0,9921875000 | 0,007812500 | 0,9960937500 | |
| 8 | 1,0 | 0,9960937500 | 0,003906250 | 0,9984687500 | |
| 9 | 1,0 | 0,9984687500 | 0,001953125 | 0,9992343750 | |
| 10 | 1,0 | 0,9992343750 | 0,000976562 | 0,9996171870 | |
| | **1,0** | **1,0000000000** | **0,000000000** | **1,0000000000** | **Término** |

**Fonte: Da Silva Filho, 2008**

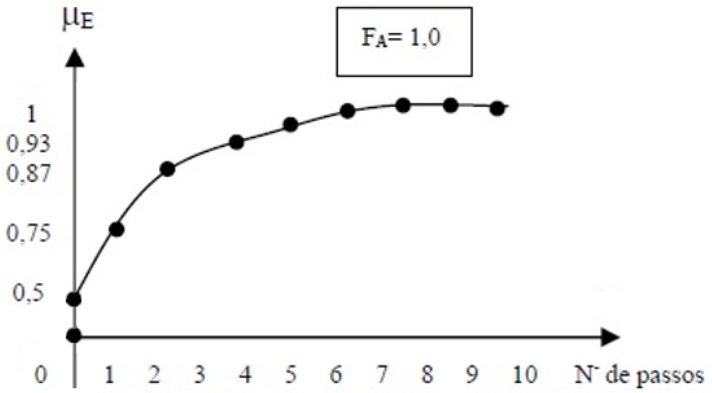



**Gráfico 2. Curva gráfica representando os Valores Obtidos de Aprendizagem da CNAPap. Fonte: Da Silva Filho, 2008.**

O Gráfico 2, mostra que o comportamento do sinal de saída aplicado, tem um padrão de repetição na entrada. No instante t0 a saída é indefinida, com Grau de Evidência resultante valendo 1/2. Do instante t0 até o instante t, acontece na saída, um comportamento monotônico, para então, o Grau de Evidência resultante ficar constante, no instante t2. Do instante t3, até o instante t4, ocorre a inversão do padrão na entrada, resultando em um decréscimo no sinal de saída, para então, no instante t4 confirmar o novo padrão aprendido.

Nesta pesquisa, a CNAPap foi implementada em um microcontrolador e depois, um aplicativo com uma programação especial, compôs um Sistema de aprendizagem por demonstração, atuando em um braço robótico. Na próxima seção, serão descritas as técnicas para os procedimentos, de uma aprendizagem por demonstração, aplicados em robôs.

### 1.5.9 Formas de Estruturas cognitivas para aprendizagem de robôs

Em aplicações de robótica, que envolvem a aprendizagem por repetições, é frequente se encontrar uma estrutura cognitiva, como base para o acionamento dos robôs (LANGLEY, 2009).

Segundo D'arpino, (2017) a arquitetura para aprendizagem de imitação de robô, possui capacidades cognitivas e motoras, que são: percepção, gerenciamento de conhecimento, aprendizagem e acionamento de comando motor.

No estudo apresentado em Breazeal, (1999) uma arquitetura foi desenvolvida para dois robôs, demostrando que o trabalho de aprendizagem dos robôs, não estava evidente em um componente específico, mas em diversos componentes, sendo que um sistema regula as preferências de atenção, com base em estados motivacionais, e são transferidos para o sistema de acionamento do robô.

Em Breazeal, (1999) a implementação foi realizada em um braço robótico equipado com uma câmera, e utilizou uma arquitetura composta por três componentes principais, que são:

a) O Subconsciente, responsável pelo sistema sensor de visão e processamento, com o objetivo de extrair informações, e controlar o sistema robótico.

b) A Área Conceitual, responsável por organizar as informações fornecidas pela área do Subconsciente.

c) O Idioma Simbólico, utilizado para representar os dados dos sensores na área de linguística.

Em Gienger, (2010) os autores apresentaram uma arquitetura em três camadas, com o objetivo de fornecer soluções para problemas de generalização, cujo processo, foi capaz de avaliar um conjunto de exemplos de treinamento, identificando características importantes comuns a esses exemplos, de modo a realizar após o aprendizado, tarefas em diferentes contextos, que ainda não foram observados. A camada denominada reativa, foi responsável por manipular percepções do sistema, já a camada Memória de Objeto, foi usada como *interface* entre o sistema e o mundo real. A terceira camada, denominada Sequência, tinha a função de ser uma memória do processo, para as sequências de movimentos.

Em Demiris, (2006) foi utilizado um robô com uma câmera, sendo está o único elemento sensor. Uma vez que o robô observa a execução de algo, todos os estados de ação, foram entregues para os modelos disponíveis do sistema. Desta forma, os comandos de motores correspondentes, que representavam as hipóteses, foram entregues ao modelo relacionado, para que ele pudesse prever o próximo movimento.

Em D'arpino, (2017) um método denominado *C-LEARN*, foi criado com o objetivo reduzir as restrições geométricas, geradas em demonstrações, com modelos generalistas. Este método tinha como característica, aprender multipassos das tarefas demonstradas, com uma sequência de quadros-chave, baseado em sistema com *Design* Auxiliado por Computador.

Para o funcionamento deste sistema, foi utilizada a técnica de tele operação, onde um operador demonstrava ao robô os movimentos a serem executados, e todos os dados foram coletados e transferidos para o ambiente 3D *CAD*. O sistema *C-LEARN* proporcionou também que, após um robô aprender uma nova habilidade, esse conhecimento pudesse ser transferido automaticamente para outros robôs.

### 1.5.10 Forma de Aprendizagem em Redes Neurais

O aprendizado de máquinas com redes Neurais, pode ser exemplificado de duas formas (HAYKIN, 2008);

1) aprendizado sem um professor ou;

2) aprendizado com o professor.

1) aprendizado sem um professor

Na aprendizagem sem professor, o processo ocorre sem a supervisão, e dessa forma, não possuem exemplos a serem seguidos pela rede neural.

Este tipo de aprendizagem, pode ser dividida em:

(1) (a) aprendizagem por reforço e;

(b) aprendizagem auto organizada.

1(a) – A aprendizagem por reforço ocorre de forma contínua, com o objetivo de minimizar um índice desempenho.

O sistema possui a característica de aprender, com a observação de uma sequência temporal de estímulos recebidos, que normalmente são resultados da geração de resultados, do sinal de reforço relativos.

A Figura 9, mostra este processo através de um diagrama de blocos, onde o sistema e ambiente, estão dentro de um *Loop* (HAYKIN, 2008).

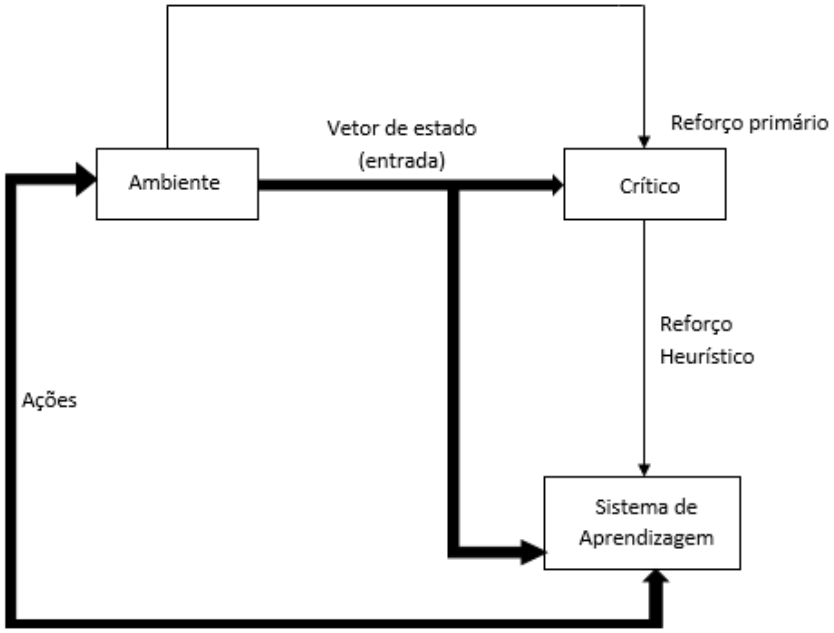



**Figura 9** – **Diagrama em blocos de aprendizagem por reforço**
**Fonte: (Haykin, 2008)**

1(b) – Na aprendizagem auto organizada, também denominada de Não Supervisionada, o professor não está presente. Dessa forma, medidas independentes são fornecidas ao sistema, para a tarefa em que a rede necessita aprender, ocorra através da otimização dos parâmetros da rede, com base nas medidas adotadas.

Se a rede neural sintonizou, as regularidades estatísticas dos dados de entrada, independente da tarefa, ela desenvolve a capacidade de formar representações, para codificar os dados de entrada, e assim criar novas classes automaticamente (HAYKIN, 2008).

2) aprendizado com o professor

Na aprendizagem com professor, também denominada como aprendizagem supervisionada, conceitua-se que o professor detém o conhecimento do ambiente, com base em um conjunto de entrada e saída.
A aprendizagem supervisionada, considerada como L*f*D, com o apoio de um sistema composto por redes neurais artificiais, pode ser classificada em 3 fases: Demonstração, Aprendizagem e Imitação, conforme detalhadas a seguir (Schaal, S. 2006), (Niekum et al. 2015):

Fase 1. Demonstração – É a fase onde o Tutor, realiza as atividades que deseja, que o robô aprenda.

Fase 2. Aprendizagem - É a fase onde ocorre a aprendizagem supervisionada.

Fase 3. Imitação – É a fase na qual, a condição de aprendizagem está completa, e a rede neural artificial, finalizou seu processo de treinamento, transferindo o conhecimento obtido do Tutor, para o Aprendiz.

Em uma etapa de demonstração, considera-se para a rede neural, que o ambiente é desconhecido, mas se o professor e a rede ficarem expostos em um vetor de treinamento, que é um exemplo extraído do ambiente, o professor consegue fornecer a rede neural uma resposta adequada para o vetor.

Desta forma, os parâmetros da rede, são ajustados com base na influência do vetor de treinamento, e do sinal de erro, que é a diferença entre a resposta real, e a desejada pela rede (Haykin, 2008).

Na Figura 10, é possível observar este conceito.

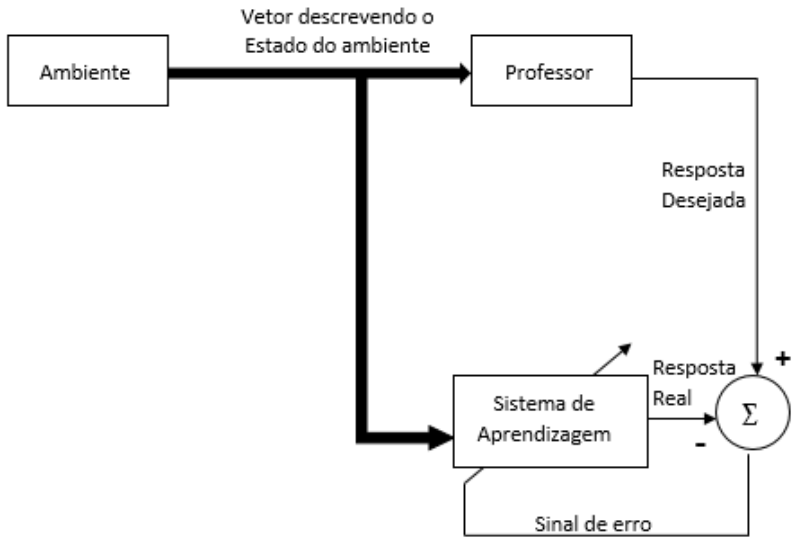



**Figura 10 – Diagrama em blocos de aprendizagem supervisionada**
**Fonte: (Haykin, 2008)**

Ao concluir a aprendizagem, considera-se que o professor não é mais necessário atuar no sistema e, portanto, considera-se que a rede neural é capaz de reconhecer e lidar com o ambiente sem intervenção externa. Esse processo recebe o nome de Imitação, caracterizado pela interação da rede neural com o ambiente de maneira independente como o que foi ensinado.

### 1.5.11 Abordagens das Etapas de Demonstração

Em geral, três tipos de abordagem são usados para o estágio de demonstração (Schaal, S. 2006), (Niekum et al. 2015).

São estes:

Tipo 1. Cinestésico - O Tutor demonstra movendo fisicamente o robô, ou a máquina, através dos movimentos, ou trajetórias desejadas.

Tipo 2. Tele operação – O Tutor não entra em contato direto com o Aprendiz, que neste caso, é o robô.

Tipo 3. Observações Passivas **-** O Tutor realiza a tarefa, usando seu próprio corpo, que às vezes pode ser instrumentado por sensores adicionais, para facilitar o rastreamento.

### 1.5.12 Modelagens de Dados para o Aprendizado

A técnica de L*f*D, traz inúmeros desafios para sua aplicação, e um dos principais problemas está no modo de modelagem de dados, para a extração de características possíveis, de serem aprendidas pelo Aprendiz.

Nesta fase, apresenta-se duas principais questões; sendo a primeira, a de como interpretar e compreender os dados decorrentes dos comportamentos observados, e a segunda, a de como integrar os sistemas de percepção e controle do movimento, para reconstruir o que é observado (Ekvall, et al. 2008).

Outro fato, é que os modelos de sistemas de controle que vão atuar no método L*f*D, devem conter facilidades de generalização, uma vez que um movimento demonstrado, deve ser possível para modelar diferentes posições de meta (Ekvall, et al. 2008).

Em (Liu & Lemeire, 2017), modelos ocultos de Markov (HMMs) foram usados para codificar e generalizar, os exemplos de demonstração. Em outros trabalhos, conforme visto em (Pastor et al.,2013) e (Schaal, 2006), os autores consideraram, o fato de que, sistemas dinâmicos com estabilidade assintótica global, definem uma função que representa, um mapa de navegação global, onde todas as trajetórias espaciais, convergem para o alvo. Uma alternativa para o sistema ser robusto, a essas perturbações temporais, é que ele seja autônomo, ou sendo, independente do tempo (Nicolescu & Mataric, 2003).

### 1.5.13 Modos de execução da Aprendizagem por Demonstração

Segundo Pook, (1993), em uma etapa da aprendizagem alguns modos de executar a demonstração, podem ser utilizados para a sua execução e registro. Para detalhar este caso, foram destacadas duas formas de L*f*D, que têm em comum a utilização:

1. Sombra.
2. Tele operação.

1. Sombra.

Na técnica de demonstração Sombra, o robô imita os movimentos demonstrados pelo professor, e grava através de seus próprios sensores.

Deste modo, o robô aprendiz registra sua própria execução imitadora, e as ações do professor são indiretamente codificadas, de um conjunto de dados.

1. Tele operação.

A técnica de tele operação, possui a característica de transferir diretamente as informações, no qual, sequências de movimentos são definidas e operadas pelo professor, enquanto o aprendiz grava os dados, a partir de seus próprios sensores.

A tele operação, é recomendada em controles de máquinas industriais, onde pode facilmente ser utilizada, devido as características de movimentação, deste tipo de aplicação (POOK, 1993).

Atualmente, a tele operação está sendo utilizada em diversas outras aplicações, como em voo de um helicóptero robótico, em movimentos de roteamento robótico em (COATES, 1994), objeto agarrando em (SWEENEY, 2007), tarefas de montagem do braço robótico (CHEN, 2003).

Todas estas aplicações usaram um *joystick,* para realizar a demonstração. A tele operação na demonstração, pode ser feita de diversas outras maneiras, como pela utilização de *Smartphone.* ou através de comandos de voz, no qual o robô é informado especificamente quais ações executar, conforme visto em (DEMIRIS, 2002).

Ao comparar a técnica de Sombra com a tele operação, a de sombra requer um algoritmo adicional, que tem como função rastrear, e se somar de maneira ativa, diferentemente da tele operação, que ocorre de forma passiva.

### 1.5.14 Programação

Nesta pesquisa, utilizamos uma programação em Linguagem C++, para implementação das CNAPaps, no *software* do microcontrolador. A Linguagem C++, é uma linguagem de programação orientada a objetos, utilizada como linguagem de máquina, ou como linguagem para softwares. Se aplica ao desenvolvimento de sistemas de alta performance, e no ensino sobre orientação, a objetos na programação. Desenvolvida em 1983, pelo dinamarquês Bjarne Stroustrup na *Nokia Bell Labs*, se aplica na programação de computadores, onde se estabelece uma disciplina de desenvolvimento de algoritmos, independente da sua complexidade, facilitando a compreensão da solução, através de um número restrito de mecanismos de codificação (ZIVIANI, 2012).

A Figura 11, representa as principais características da Programação:

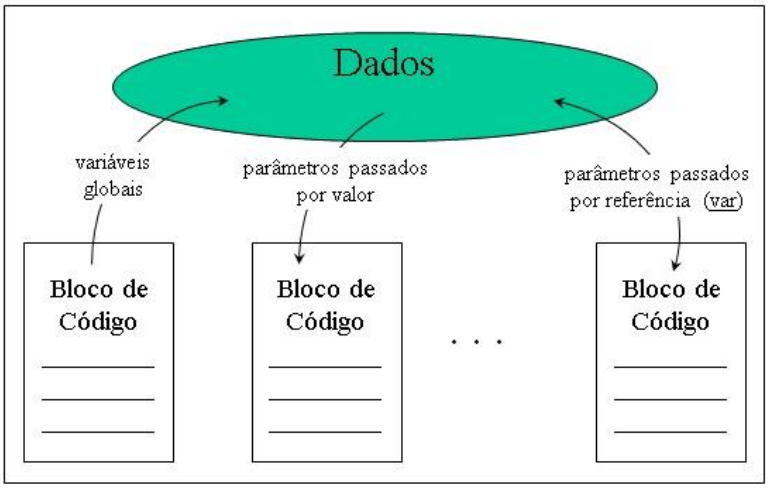



**Figura 11: Principais características da Programação**
**Fonte: Ziviani, 2012**

### 1.5.15 Dispositivo de Acionamento Manual de Padrões de Aprendizagem

Apesar de existirem equipamentos simples, e até outras alternativas com tecnologias altamente desenvolvidas, para apoiar movimentação de máquinas, o *joystick* ainda continua sendo, o dispositivo de entrada, muito utilizado para acionar equipamentos. Neste trabalho, utilizou-se no primeiro teste, um *joystick* para inserir manualmente, os padrões que a CNAPap aprendeu, em sua fase de treinamento. No segundo teste, foram utilizados potenciômetros, para esta finalidade. O Anexo A, apresenta o potenciômetro, servo motor e o *chip* ATMEGA 328p, utilizados na modelagem física, e as características do *joystick* e o seu funcionamento básico, relacionado ao tipo que foi utilizado nesta pesquisa.

### 1.5.16 Robôs Industriais

Os robôs são essencialmente, máquinas capazes de realizar os mais diversos movimentos programados, adaptando-se às necessidades operacionais de determinadas tarefas, e empregando garras e/ou ferramentas, conforme o trabalho, ou tarefa a que está destinado (GIENGER et al, 2010) (DA SILVA FILHO et al, 2006).

### 1.5.17 Classificação dos Robôs Industriais

Os Robôs Industriais podem ser classificados, conforme a sua estrutura mecânica, pois diversas combinações de elementos (juntas e elos), podem ser adicionadas, para se obter uma configuração desejada (SCHIAVICCO & SICILIANO,1995) (CRAIG, 1986) (BORODIN, 1988), (CUTKOSKY,1989) (GIENGER et al., 2010). Segundo (SCHIAVICCO & SICILIANO, 1995), as principais configurações básicas, quanto à estrutura mecânica de um robô Industrial, são:

a) Robô de Coordenadas Cartesianas/Pórtico. Este tipo de robô, possui três juntas prismáticas (PPP), resultando em um movimento composto de três translações, cujos eixos de movimento, são coincidentes com um sistema de coordenadas de referência cartesiano. A sua representação geométrica está mostrada, na Figura 12(a), onde se destaca a sua geometria e espaço de trabalho, com as coordenadas cartesianas (d1 [x], d2 [y], d3 [z]), definindo a posição da garra com relação à base.

b) Robô de Coordenadas Cilíndrica. Este tipo de robô, possui os eixos de movimento, que podem ser descritos, no sistema de coordenadas de referência cilíndrica. A sua estrutura mecânica, é formada por duas juntas prismáticas e uma de rotação (RPP), compondo movimentos de duas translações, e uma rotação. A sua representação geométrica está mostrada, na Figura 12(b), onde se destaca a sua geometria e espaço de trabalho, com as coordenadas cilíndricas (θ1, d2, d3), definindo a posição da garra em relação à base.

c) Robô de Coordenadas Esféricas. Neste tipo de robô, os eixos de movimento formam um sistema de coordenadas de referência polar, através de uma junta prismática e duas de rotação (RRP), compondo movimentos de uma translação, e duas rotações.

A sua representação geométrica, está mostrada na Figura 12 (c), onde se destaca a sua geometria e espaço de trabalho, com as coordenadas esféricas (θ1, θ2, d3), que definem a posição da garra. Os três eixos Z0, Z1 e Z2, são mutuamente perpendiculares.

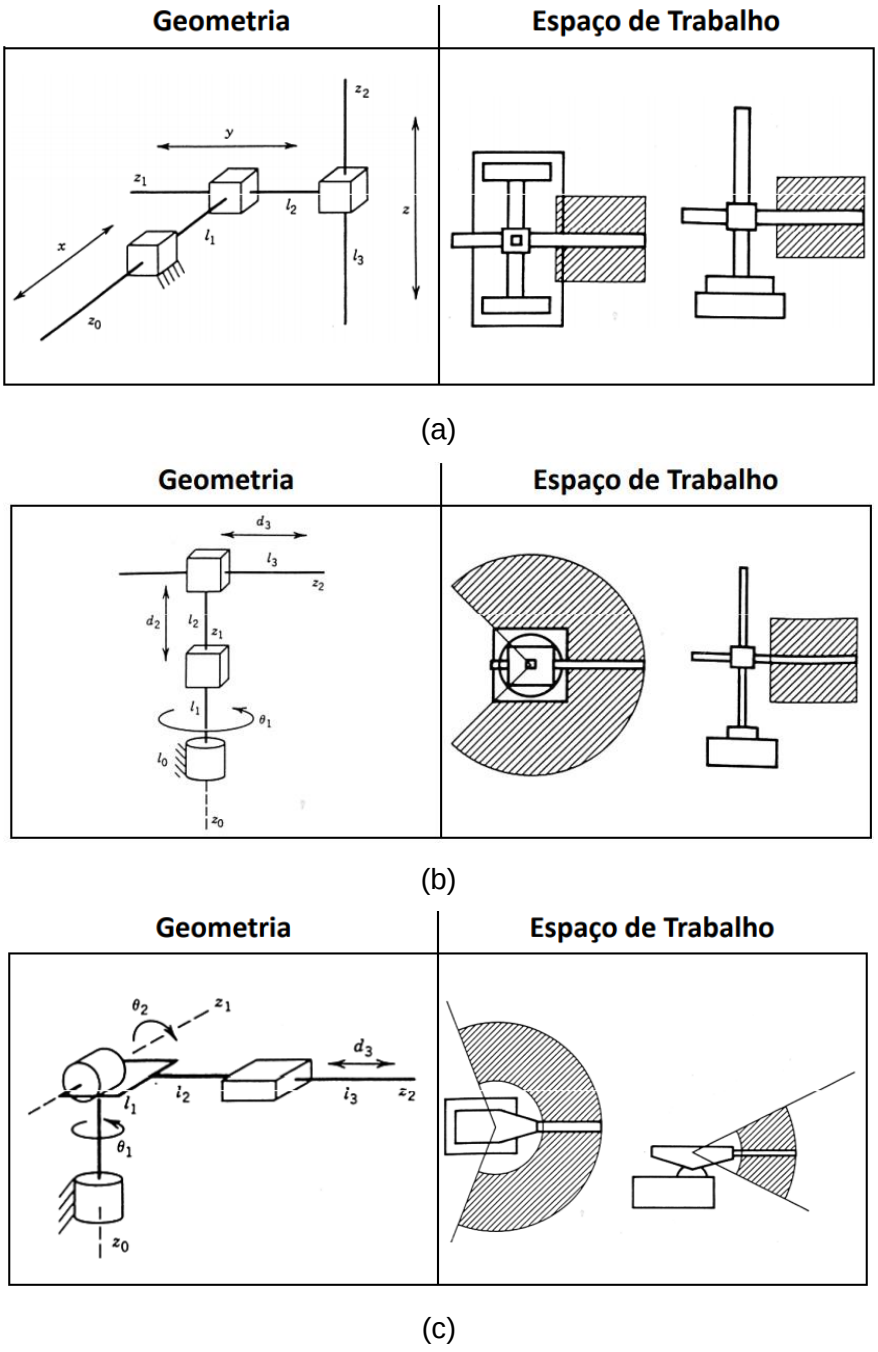



**Figura 12. Três principais configurações básicas quanto à estrutura mecânica de Robô Industrial. a) Robô de Coordenadas Cartesianas/Pórtico. b) Robô de Coordenadas Cilíndrica. c) Robô de Coordenadas Esféricas.**

### 1.5.18 Braço Robótico ou Manipulador Mecânico

Na robótica industrial, o Braço Robótico também denominado Manipulador Mecânico, refere-se principalmente ao aspecto mecânico e estrutural do robô.

Os robôs manipuladores industriais possuem mecanismos compostos de ligamentos rígidos, conectados por articulações que permitem o movimento relativo entre os ligamentos vizinhos.

Nas articulações, são adicionados sensores de posição, que fornecem a posição entre os ligamentos vizinhos. No caso de articulações de revolução, ou de rotação, estes sensores medem posição angular.

Alguns robôs contém articulações de translação, ou prismáticas, nas quais o deslocamento relativo entre os ligamentos, segue uma linha reta.

O número de graus de liberdade que um robô manipulador possui, é igual ao número de variáveis de posição independentes, que devem ser especificadas para localizar todas as partes do mecanismo.

Como definição, esta estrutura, consiste da combinação de elementos estruturais rígidos (corpos ou elos), conectados entre si através de articulações (juntas), sendo o primeiro corpo denominado base, e o último extremidade terminal, onde será vinculado o componente efetuador (garra ou ferramenta).

A estrutura mecânica do Robô de coordenadas esféricas, está mostrada na Figura 12(c). O seu aspecto geométrico, e espaço de trabalho, é mais utilizada para construção do manipulador mecânico (SCHIAVICCO & SICILIANO,1995) (CRAIG, 1986) (BORODIN, 1988), (CUTKOSKY,1989) (GIENGER et al., 2010).

Nesta pesquisa iremos utilizar, um Robô de coordenadas esféricas.

# 2. MATERIAIS E MÉTODOS

Esta pesquisa foi desenvolvida em duas etapas, onde na primeira, foi feita a programação do algoritmo da CNAPap e testes de aderência e, na segunda etapa, se deu a programação e geração do *software* da CNAPap, com a montagem e testes de funcionamento da CNAPap Física. As ações nesta primeira parte, estão descritas a seguir.

## 2.1 Programação do algoritmo da CNAPap e testes de aderência

Inicialmente, foram utilizadas funções da programação em linguagem C++, para implementar o modelo matemático originado do algoritmo da Célula Neural Artificial Paraconsistente de Aprendizagem (CNAPap), para efetuação de um estudo computacional.

Nesta implementação inicial, foram feitas as investigações de modo computacional, sobre os limites impostos pelo algoritmo da LPA2v, e o reconhecimento das condições necessárias, para se fazer as comparações das ações, que deveriam ser implementadas em *hardware*.

O código completo do algoritmo implementado, na linguagem de programação, utilizado nesta primeira etapa, são mostrados no APÊNDICE A.

Observou-se, que foram inseridos nas linhas do programa, pontos com pausas para verificação dos dados, e assim, obter comportamentos dos sinais, em diversas fases do aprendizado.

Com essa primeira implementação, foram realizados ensaios, para verificação das compatibilizações das respostas esperadas, pelo algoritmo original e, possíveis adequações da estrutura algorítmica na programação utilizada.

Os principais ensaios, foram realizados, com 4 valores do Fator de Aprendizado diferentes: Fa = 0,25, Fa = 0,5, Fa = 0,75 e Fa=1,0.

Os resultados dos ensaios desta implementação inicial da CNAPap, são mostrados no capitulo referente aos resultados.

## 2.2 Montagem e Funcionamento da CNAPap Física

Após a implementação inicial com a linguagem de programação, compondo o algoritmo da CNAPap, foi preparada uma modelagem física baseada nos resultados obtidos na primeira etapa.

A modelagem física para utilização do *chip,* foi composta por uma *interface (Arduíno Uno)* com uma estrutura de testes própria para a programação do *chip,* constituída de um microcontrolador, e de conexões dos terminais com o meio físico externo.

Para a inserção dos sinais de entrada, foi utilizado um *joystick*, e o movimento da sua alavanca, foi representado em um sistema de coordenadas angulares, com dois eixos, (x, y), onde os componentes x e y, são correspondentes a simulação de comando de rotação, e translação do *joystick*.

Considerou-se neste trabalho que um comando de controle de (1) equivalia à rotação para a frente ou para a esquerda, e um comando de controle de (-1) equivalia à reversão máxima ou rotação para a direita. Um comando de (0) não especificaria nenhum movimento.

Para estudar as ações resultantes do Sistema paraconsistente, foi utilizado um braço manipulador robótico, composto de uma estrutura onde ocorre as ações de cinco servo motores de acionamento.

### 2.2.1 Princípio de Funcionamento

O sistema construído em *hardware (Arduíno Uno)*, tem os sinais de aprendizagem da célula, inseridos através do *joystick* que, ao ser acionado, a sua manopla, obtém como resultado a movimentação da garra do braço robótico, representado pela energização de seu servo motor CC.

Portanto, o movimento do *joystick,* gera as informações para o aprendizado por Demonstração da célula CNAPap, que através da programação do microcontrolador, ficou residente no *hardware*.

Após completar o ciclo de aprendizagem, isto é, efetuadas as iterações necessárias para que a saída da CNAPap, apresente seu valor máximo (1,0), uma chave foi acionada, havendo então, a liberação dos sinais de saída do microcontrolador.

A programação do sistema, fez com que o microcontrolador ao receber este comando, acionasse o servo motor da garra do braço robótico, para efetuar as ações, conforme foram aprendidas através das demonstrações.

A Figura 13, mostra uma representação geral da montagem efetuada, e sequencias de funcionamento da aprendizagem por demonstração. A Figura 14, o diagrama eletrônico, das conexões utilizadas na Interface *(Arduíno Uno)*.

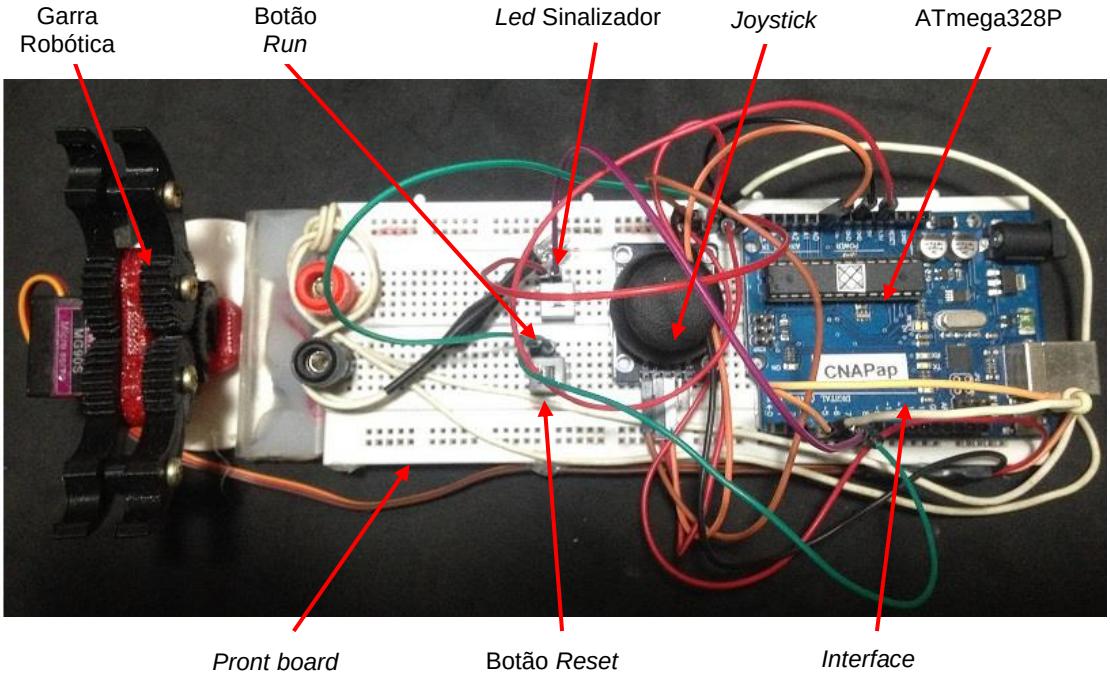



**Figura 13. Descrição dos componentes utilizados no *Arduíno Uno*, onde uma célula de aprendizagem CNAPap foi implementada no microcontrolador ATmega 328P**

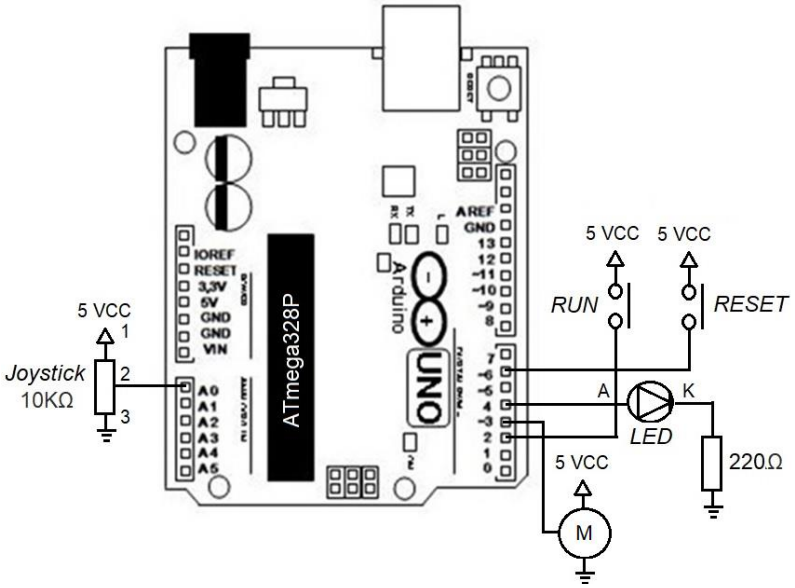


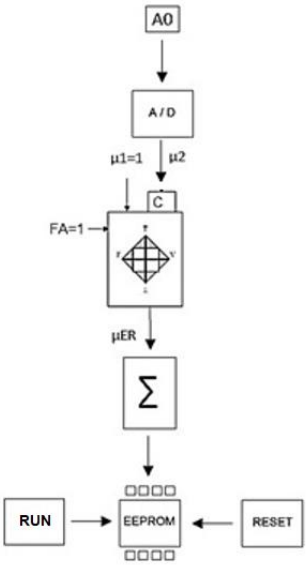

**Figura 14. Diagrama eletrônico das conexões utilizadas no *Arduíno Uno*, onde uma célula de aprendizagem CNAPap foi implementada no microcontrolador ATmega 328P**

### 2.2.2 Aprendizagem por demonstração e processo de repetição utilizando uma única CNAPap

No aprendizado por demonstração que utilizou o *hardware (Arduíno Uno),* $\mu_1$ foi um dos graus de evidência do algoritmo, que representou o padrão a ser aprendido. No programa, o seu valor foi igual a 1 e assim, sinais analógicos de voltagem entre 0V e 5V, foram inseridos através do *joystick*, e aplicados no terminal de entrada do *chip* ATmega 328P (A0), correspondente a CNAPap. Um conversor interno, transformou os sinais analógicos para digitais, e o conectou em μ2, que foi outro grau de evidência. Desse modo, $\mu_2$ utilizou variáveis entre os limites 0 e 1.

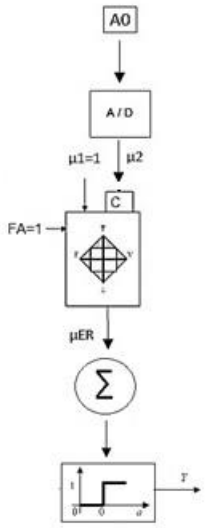
Para o acionamento do aprendizado por demonstração, o Fator de aprendizagem $F_A$, utilizado para determinar a velocidade de aprendizado da CNAPap, foi ajustado na programação com valor igual 1. Conforme a inserção dos sinais de entrada (A0), na CNAPap no microcontrolador, que aconteceram através das repetições dos acionamentos do *joystick*, que variou os sinais na saída da CNAPap (μE) - grau de evidência resultante (pino 3). A variação do valor da saída, observada pela manipulação do *joystick*, como consequência, atuava o servo motor instalado no Braço Robótico, que, por sua vez, movimentava a garra.

No processo de aprendizagem por demonstração, os dados inseridos na CNAPap, foram lidos e armazenados na decorrência de 10 repetições sequenciadas, conforme os padrões de aprendizagem da CNAPap, sem espaços para desvios. Ao final dessas repetições, a CNAPap sinalizou que aprendeu, através da ativação de uma saída analógica (pino 4), conectada a um *LED*. Os sinais analógicos convertidos em dados, eram armazenados em uma área de memória da CNAPap. Para executar a leitura dos sinais memorizados, era necessária a ativação do botão denominado *RUN*, conectado a uma entrada do chip (pino 2). Esse passo quando foi executado, acionava então o servo motor (μE) (pino 3), que por sua vez movimentava a garra, sem a necessidade de manipular o *joystick* ($\mu_2$) (A0). Para finalizar e limpar os dados inseridos na área de memória da CNAPap, o botão denominado *RESET* era ativado (pino 6), completando assim o ciclo de aprendizagem e repetição. Nas Figuras 15, 16 são mostrados os fluxogramas dos sinais no funcionamento da aprendizagem por demonstração, com uma única CNAPap.




| | |
|---|---|
| A0 | Entrada Analógica |
| A / D | Conversor analógico digital |
| µ1 | Grau de evidência com valor = 1 |
| µ2 | Grau de evidência com valor variável |
| FA | Fator de aprendizagem com valor = 1 |
| µER | Grau de evidência resultante |
| ∑ | Totalizador de Repetições |
| *RUN* | Botão para execução do Programa |
| EEPROM | Área de armazenamento |
| *Reset* | Botão de Parada/Apagamento do Programa |

**Figura 15. Fluxograma representando o funcionamento da CNAPap Física.**




| | |
|---|---|
| A0 | Entrada Analógica |
| µ1 | Grau de evidência com valor = 1 |
| µ2 | Grau de evidência com valor variável |
| FA | Fator de aprendizagem com valor = 1 |
| µER | Grau de evidência resultante |
| ∑ | Totalizador de Repetições |
|  | Dispositivo de Acionamento |

**Figura 16. Fluxograma representando a totalização das repetições e o acionamento.**

### 2.2.3 Aprendizagem por Demonstração e Processo de Repetição utilizando cinco CNAPaps na forma de um SNAPApdem

Neste ensaio, $\mu_{1}$, $\mu_{1b}$, $\mu_{1c}$, $\mu_{1d}$ e $\mu_{1e}$ foram os graus de evidência do algoritmo, portanto, os padrões que foram aprendidos no programa. Cada um deles teve o seu valor igual a 1. Quando ativado o botão *RUN* (pino 4), do *hardware (Arduíno Uno),* mostrado na Figura 22 e 23, onde agora um braço robótico com cinco pequenos motores CC, foi instalado no lugar da garra, um *LED* (pino 11) começou a acender e apagar, sinalizando que o sistema estava pronto para o início da aprendizagem, habilitando a movimentação.

### 2.2.4 Braço Robótico utilizado

O Braço Robótico utilizado nos testes, tem a sua estrutura mecânica classificada, em um tipo de coordenadas esféricas, mostrado na Figura 12(c), com algumas adaptações construtivas. Os testes com o Sistema Neural Artificial Paraconsistente de aprendizagem por demonstração – SNAPApdem, foram para acionar os cinco principais motores de movimentação das ações de:

1. Rotação da cintura.
2. Rotação do ombro.
3. Rotação do cotovelo.
4. Rotação do punho.
5. Rotação da Garra.

A Figura 17, mostra o desenho em destaque das ações de rotação das partes do Braço Robótico, que foram atingidas pelo processo de aprendizagem por demonstração, através do SNAPApdem.

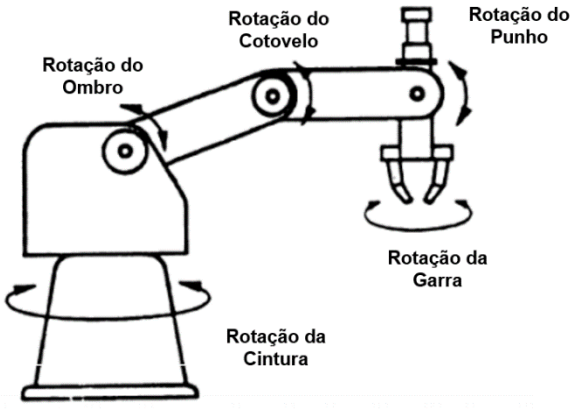



**Figura 17. Partes do Braço Robótico que serão atingidas pelo processo de aprendizagem por demonstração através do SNAPApdem.**

Cada parte do Braço Robótico que será atingida pelo processo de aprendizagem por demonstração tem o seu correspondente motor de acionamento. Portanto nesta pesquisa o SNAPApdem terá a sua ação em cinco motores CC. Para estabelecer condições para os testes, os valores de voltagem CC que aparecem nas saídas de cinco potenciômetros são considerados como comparativos ao ângulo de deslocamento real provocado na movimentação dos eixos através do acionamento de cada motor CC.

A Figura 18, mostra o Braço articulado, com cinco servo motores.

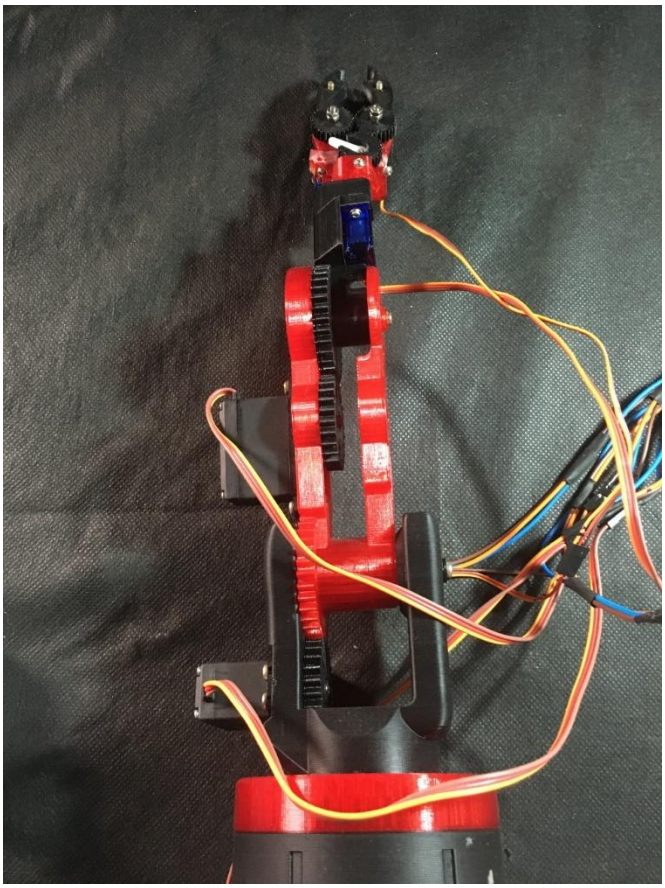


**Figura 18. Braço articulado com cinco servo motores, modelagem do suporte e engrenagens fabricados em PVC através de impressora 3D.**

### 2.2.5 Procedimentos iniciais de ajustes

Inicialmente, uma linha de referência para cada eixo de rotação, de cada parte do Braço Robótico, foi considerada e feito o alinhamento com o valor de zero *volt*, correspondente a saída de cada um dos potenciômetros geradores, dos padrões que foram aprendidos pelas CNAPaps, componentes do SNAPApdem.

Os ajustes iniciais corresponderam, ao Intervalo de Interesse de 0 até 120 graus de deslocamento, para cada parte do Braço Robótico, atingida pelo processo de aprendizagem por demonstração.

Dessa forma, uma variação do potenciômetro, que resultasse entre 0 e 5 *Volts* CC, representava uma variação correspondente, a rotação de 0 a 120 graus, nas partes do Braço Robótico, que foram atingidas, pelo processo de aprendizagem por demonstração.

A monitoração das ações de movimentação do Braço robótico, foram feitas, através de respostas obtidas por sensores instalados nas juntas, e na garra do Braço Robótico. O valor da resistência, a partir de uma referência de ângulo, correspondeu ao ângulo de deslocamento, de cada junta percorrida pelo Braço robótico, portanto, as informações da resposta de aprendizagem, nas juntas e na garra, foram obtidas com base, em sua própria resistência (*Ohms*).

Para cada junta, e para a garra, foram considerados inicialmente, o ângulo máximo de deslocamento, que foi aprendido pelo robô, em relação ao ângulo Mínimo considerado como referência.

Cada parte do Braço Robótico, que foi atingida pelo processo de aprendizagem por demonstração, teve o seu correspondente, motor de acionamento. Portanto, nesta pesquisa o SNAPApdem teve a sua ação em cinco servo motores CC.

Para estabelecer condições para os testes, os valores de voltagem CC, que aparecem nas saídas dos cinco potenciômetros, foram considerados como, comparativos aos ângulos de deslocamento reais, provocados na movimentação dos eixos, através do acionamento de cada servo motor CC.

Conforme exposto na Figura 19, relacionou-se o ângulo máximo, com a resistência máxima do potenciômetro ΩMax, e considerou-se o ângulo mínimo, igual a resistência nula do potenciômetro ΩMin=0, que serviu como referência.

A partir da determinação dos ângulos, se relacionou os ângulos de deslocamento com a resistência, para qualquer medida obtida no potenciômetro (ΩMedido), obtendo o grau de evidência, fazendo a relação:

$$\text{Se } \Omega\text{medido} \geq \Omega\text{Max}, \text{ então } \mu_2 = 1$$
$$\text{Se } \Omega\text{medido} \leq \Omega\text{Min}, \text{ então } \mu_2 = 0$$
$$\text{Se } \Omega\text{Min} \leq \Omega\text{medido} \leq \Omega\text{Max} \text{ então } \mu_2 = \frac{\Omega\text{medido - }\Omega\text{Min}}{\Omega\text{Max - }\Omega\text{Min}}$$

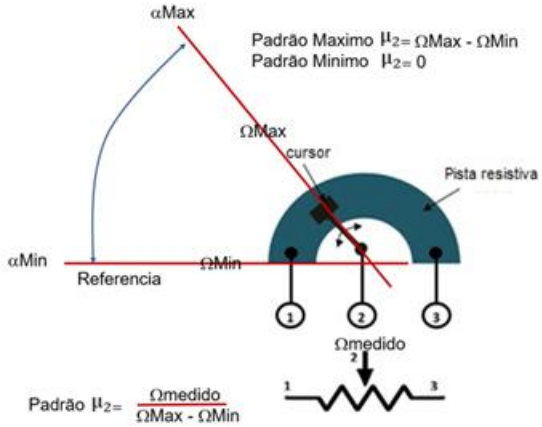



**Figura 19. Relação dos ângulos limites definidos na movimentação desejada das juntas e da garra, com a geração do grau de evidencia padrão μ2, que será aplicado na CNAPap para o aprendizado por demonstração.**

Havendo similaridades, considerou-se a excursão $\Delta\alpha UD = 90^o$ que correspondia a 10 KΩ para todos os 20 movimentos estudados. Definidos os ângulos de deslocamento máximos (αMax), e mínimos (αMin) de cada movimentação das juntas, foram calculados os limites do Universo de Discurso, e inseridos na programação do microcontrolador, para o cálculo do padrão μ2 a ser aprendido. Dessa forma, o cálculo para o Fator de aprendizagem aprendido, foi feito por:

$$\mu_2 = \frac{\text{Ângulo } \alpha\text{medido - }\Omega\text{Min}}{\Delta\alpha} = \frac{\text{Ângulo } \alpha\text{medido em graus}(^o)}{90^o} = \frac{\text{Resistência Medida em } \Omega}{10K\Omega}$$

Os padrões de treinamento escolhidos para as análises, foram definidos a partir de cada grau de deslocamento escolhido, para a aprendizagem de cada uma das cinco juntas do Robô.

### 2.2.6 Testes com o SNAPApdem

No início da aprendizagem, o botão *RUN*, foi ativado e o *LED* (pino 11) acendeu e apagou, sinalizando o início da aprendizagem. Considerados os ajustes iniciais, com os sinais analógicos de voltagem entre 0V e 5V, inseridos através de cinco potenciômetros, foram então aplicados aos terminais de entrada do *chip* ATmega328P (A0, A1, A3, A4 e A5), os padrões de aprendizagem. Conversores internos, transformaram os sinais analógicos para digitais, e os conectaram em $\mu_{C2}$, $\mu_{C2b}$, $\mu_{C2c}$, $\mu_{C2d}$ e $\mu_{C2e}$, que são os graus de evidência, que utilizaram variáveis entre os limites 0 e 1. O Fator de aprendizagem de cada CNAPap, têm os símbolos $C_4$, $C_{4b}$, $C_{4c}$, $C_{4d}$, e $C_{4e}$, e foram utilizados para determinar a velocidade de aprendizagem de cada célula, sendo ajustado na programação, o valor igual a 1. Conforme eram inseridos os sinais analógicos, nas entradas das CNAPaps (A0, A1, A3, A4 e A5), também ocorriam variações de sinais nas saídas correspondentes (µER grau de evidência resultante (pino 3), µErb grau de evidência resultante (pino 5), µERc grau de evidência resultante (pino 6), µERd grau de evidência resultante (pino 9) e µEre grau de evidência resultante (pino 10). Estas variações nas saídas, indicaram que as células estavam aprendendo pela demonstração. Isso foi observado, pela manipulação de cada potenciômetro, e a atuação de cada servo motor, que movimentou cada articulação correspondente no braço robótico. Verificou-se que neste caso, os potenciômetros substituíram os *joysticks*, na movimentação das articulações do braço Robótico, nas correspondentes inserções de sinais analógicos. Como foi observado na Figura 24, os dados inseridos em cada CNAPap, foram lidos e armazenados na decorrência de 10 repetições sequenciadas, sem espaços para desvios. Na CNAPap Física, ao final dessas repetições, o botão *RUN* (pino 4), foi novamente ativado, e o *LED* (pino 11) se apagou, sinalizando o final da aprendizagem. Os sinais analógicos convertidos em dados, foram então armazenadas em uma área de memória da CNAPap Física. Para execução da leitura dos sinais aprendidos, foi necessária a ativação do botão denominado *EXE*, conectado a uma entrada do *chip* (pino 2). Esse passo quando executado, acionava os servos motores conectados aos ATmega328P (pino3, pino5, pino6, pino8, pino9 e pino 10), que movimentavam suas articulações correspondentes, na sequência em que foram acionadas durante a aprendizagem, sem a necessidade de manipular os potenciômetros (A0, A1, A3, A4 e A5).

Para finalizar e parar a execução, o botão denominado *EXE* foi novamente ativado, parando a execução e completando o ciclo. Nas Figuras 20, 21, 22 e 23, foram mostradas as montagens nos testes efetuados com o SNAPApdem.

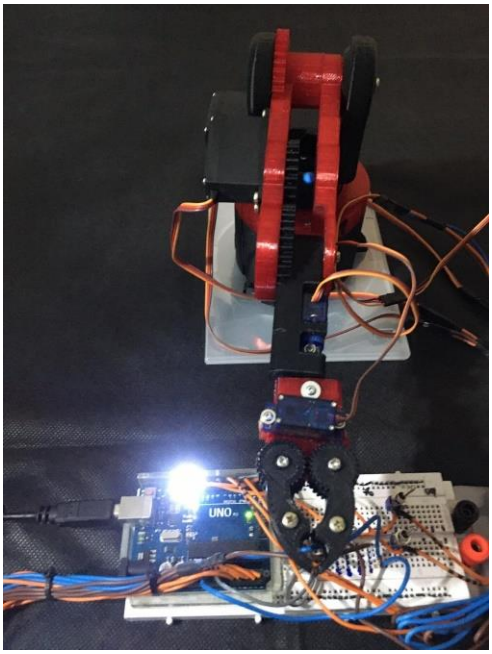


**Figura 20. Braço articulado com cinco servo motores, modelagem do suporte e engrenagens fabricados em PVC com impressora 3D, *Interface (Arduíno Uno) chip* ATmega328P fixado em acrílico PVC.**

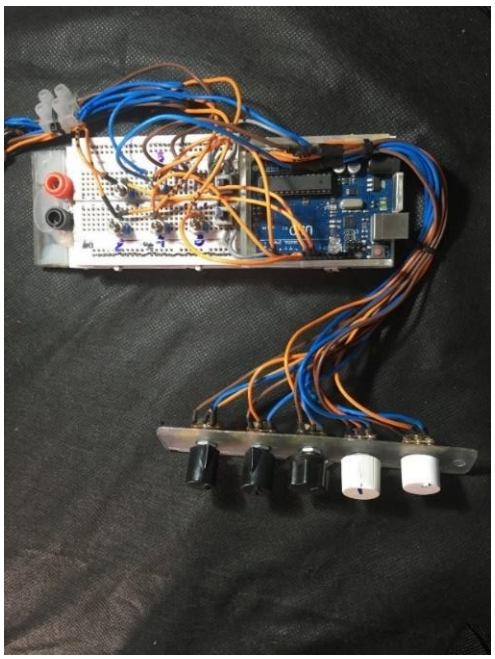


**Figura 21. *Interface (Arduíno Uno) chip* ATmega328P em acrílico PVC, para conexões de fiação e botões, cinco potenciômetros fixados na barra de acrílico em PVC, substituem os *joysticks* na inserção de sinais analógicos nas CNAPaps**

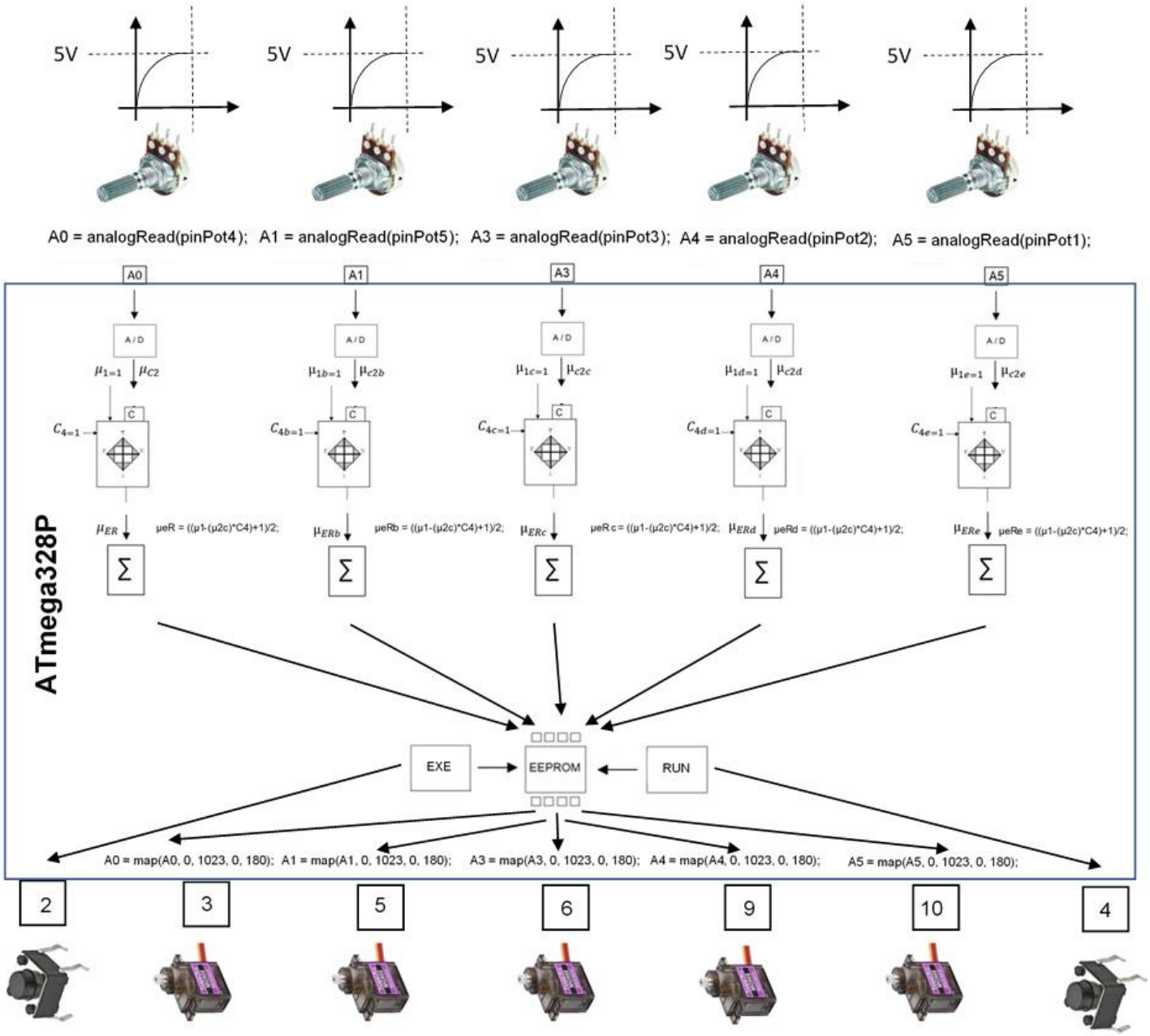



**Figura 22. Diagrama dos pinos do *chip* ATmega328P entradas: cinco potenciômetros, substituem os *joysticks* (sinais analógicos de voltagem) dois botões: (Movimento e execução do programa) cinco saídas: Servo motores (movimento articulado)**

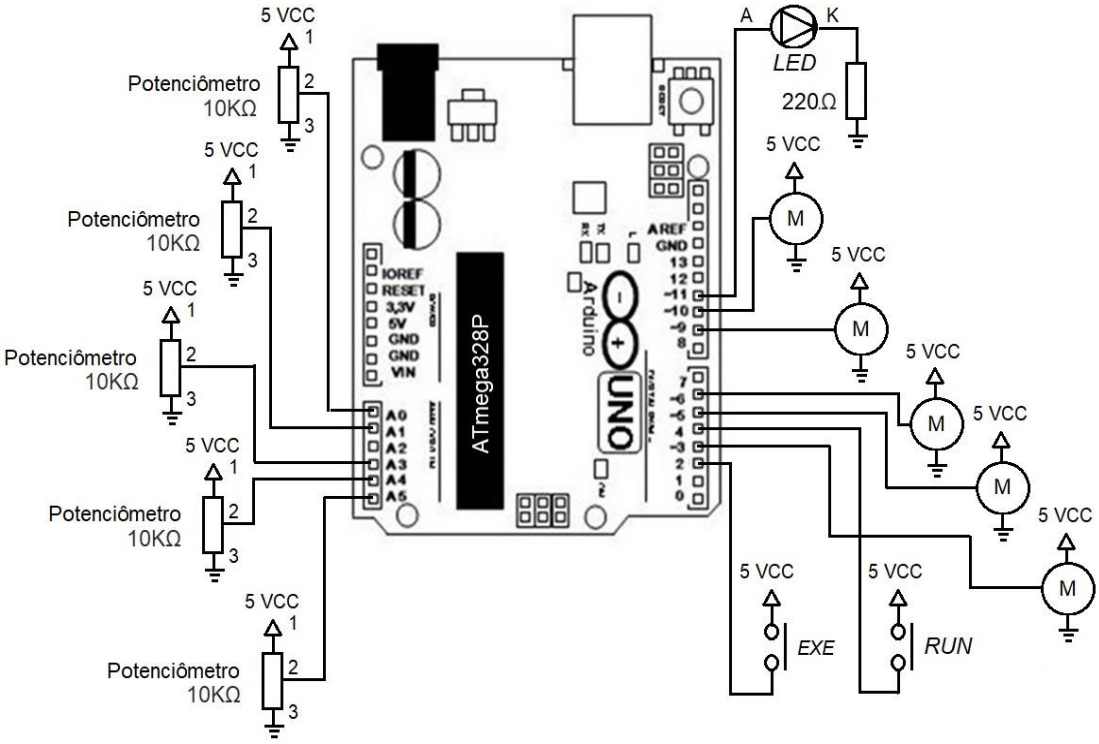



**Figura 23. Diagrama eletrônico da interface *(Arduíno Uno)* e pinos do *chip* ATmega328P entradas: cinco potenciômetros, substituem os *joysticks* (sinais analógicos de voltagem) dois botões: (Movimento e execução do programa) cinco saídas: Servo motores (movimento articulado)**

## 3. RESULTADOS e DISCUSSÃO

Inicialmente, foram feitos os testes com a investigação sobre os limites impostos pelo algoritmo da LPA2v, e o reconhecimento das condições necessárias. Deste modo, pode-se fazer as comparações das ações que deveriam ser implementadas em *hardware (Arduíno Uno)*.

No código do algoritmo implementado na linguagem C++, foram inseridos nas linhas do programa, pontos com pausas para verificação dos dados, e assim obter comportamentos dos sinais, em diversas fases do aprendizado.

Foram feitos ensaios, para verificação das compatibilizações das respostas esperadas pelo algoritmo original, e possíveis adequações da estrutura algorítmica na programação utilizada. Os ensaios iniciais utilizando a linguagem de programação C++, cujos resultados foram apresentados nas Tabelas 2, 3, 4, 5 e 6; Gráficos 3, 4, 5, 6 e 7, foram feitos utilizando os limites correspondentes aos consolidados obtidos com o algoritmo da LPA2v, apresentados na Tabela 1 e mostrados no Gráfico 2. Nestes ensaios, foram utilizados 4 valores diferentes do Fator de Aprendizado: Fa = 0,25, Fa = 0,5, Fa = 0,75 e Fa=1,0.

O fator de aprendizagem, funcionou como um ganho do sinal, e dependendo do valor de Fa, foi proporcionado uma aprendizagem mais rápida, ou mais lenta da CNAPap.

**Tabela 2. Dados Obtidos com a Implementação com Linguagem de Programação C++, Fator de aprendizagem 0,25.**

| Passos | FA | μ1 | Gc | Gct | μE |
|---|---|---|---|---|---|
| 0 | 0,250000000 | 1,000000000 | 0,000000000 | 0,500000000 | 0,000000000 |
| 1 | 0,250000000 | 1,000000000 | 0,500000000 | 0,500000000 | 0,937500000 |
| 2 | 0,250000000 | 1,000000000 | 0,937500000 | 0,062500000 | 0,992187500 |
| 3 | 0,250000000 | 1,000000000 | 0,992187500 | 0,007812500 | 0,999023438 |
| 4 | 0,250000000 | 1,000000000 | 0,999023438 | 0,000976562 | 0,999877930 |
| 5 | 0,250000000 | 1,000000000 | 0,999877930 | 0,000122070 | 0,999984741 |
| 6 | 0,250000000 | 1,000000000 | 0,999984741 | 0,000015259 | 0,999998093 |
| 7 | 0,250000000 | 1,000000000 | 0,999998093 | 0,000001907 | 0,999999762 |
| 8 | 0,250000000 | 1,000000000 | 0,999999762 | 0,000000238 | 1,000000000 |

A Tabela 2 mostra os dados, utilizando o Fator de aprendizagem 0,25.

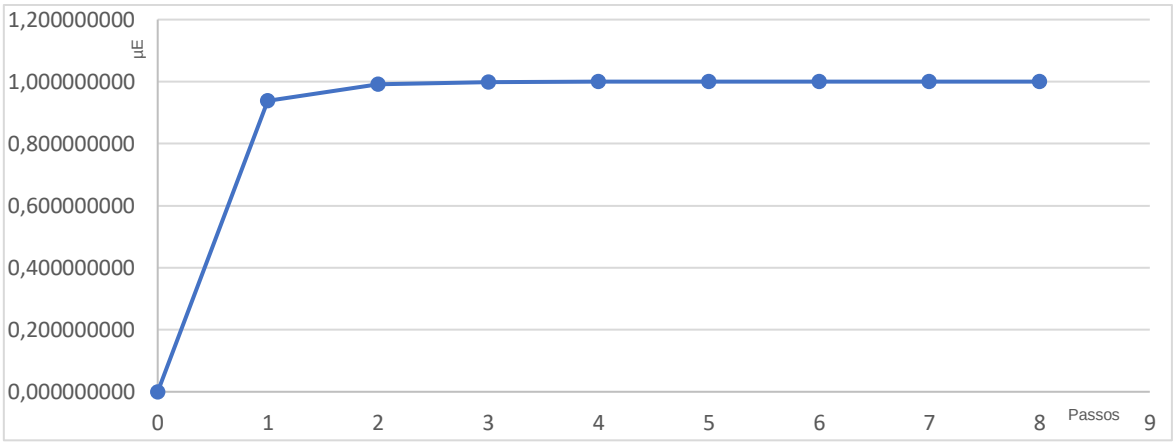



**Gráfico 3. Dados Obtidos com a Implementação com Linguagem de Programação C++, Fator de aprendizagem 0,25.**

No Gráfico 3, é mostrado o comportamento de aprendizagem, obtido no treinamento da CNAPap.

**Tabela 3. Dados Obtidos com a Implementação com Linguagem de Programação C++. Fator de aprendizagem 0,5.**

| Passos | FA | µ1 | Gc | Gct | µE |
|---|---|---|---|---|---|
| 0 | 0,500000000 | 1,000000000 | 0,000000000 | 0,500000000 | 0,000000000 |
| 1 | 0,500000000 | 1,000000000 | 0,500000000 | 0,500000000 | 0,875000000 |
| 2 | 0,500000000 | 1,000000000 | 0,875000000 | 0,125000000 | 0,968750000 |
| 3 | 0,500000000 | 1,000000000 | 0,968750000 | 0,031250000 | 0,992187500 |
| 4 | 0,500000000 | 1,000000000 | 0,992187500 | 0,007812500 | 0,999511719 |
| 5 | 0,500000000 | 1,000000000 | 0,999511719 | 0,001953125 | 0,999877930 |
| 6 | 0,500000000 | 1,000000000 | 0,999877930 | 0,000488281 | 0,999969820 |
| 7 | 0,500000000 | 1,000000000 | 0,999969820 | 0,000122070 | 0,999998093 |
| 8 | 0,500000000 | 1,000000000 | 0,999998093 | 0,000007629 | 0,999999523 |
| 9 | 0,500000000 | 1,000000000 | 0,999999523 | 0,000001907 | 0,999999881 |
| 10 | 0,500000000 | 1,000000000 | 0,999999881 | 0,000000477 | 1,000000000 |
| 11 | 0,500000000 | 1,000000000 | 1,000000000 | 0,000000119 | 1,000000000 |

A Tabela 3 mostra os dados obtidos, utilizando o Fator de aprendizagem 0,5.

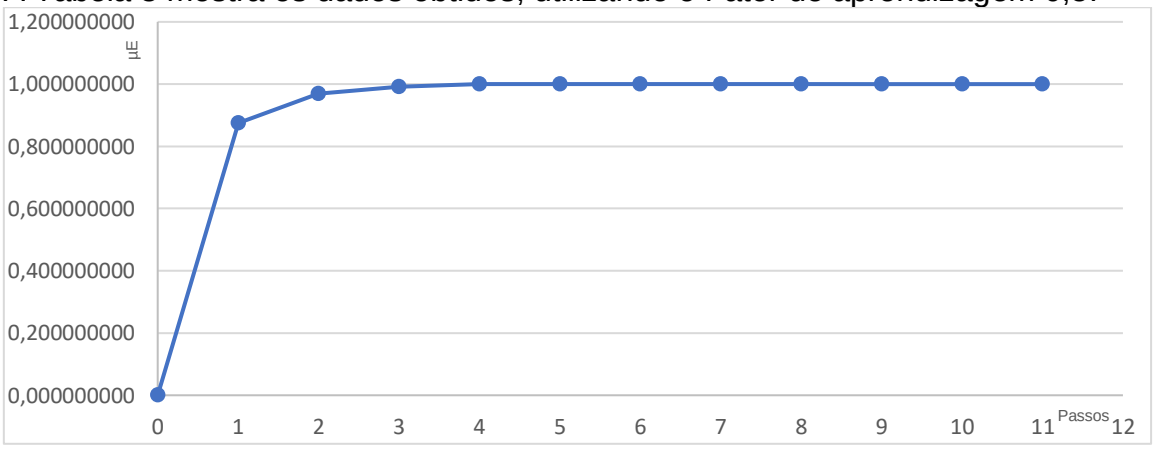



**Gráfico 4. Dados Obtidos com a Implementação com Linguagem de Programação C++. Fator de aprendizagem 0,5.**

No Gráfico 4, é mostrado o comportamento de aprendizagem obtido no treinamento da CNAPap.

Verificou-se que, a célula aprendeu, após 11 passos.

**Tabela 4. Dados Obtidos com a Implementação com Linguagem de Programação C++. Fator de aprendizagem 0,75.**

| Passos | FA | µ1 | Gc | Gct | µE |
|---|---|---|---|---|---|
| 0 | 0,750000000 | 1,000000000 | 0,000000000 | 0,500000000 | 0,000000000 |
| 1 | 0,750000000 | 1,000000000 | 0,500000000 | 0,500000000 | 0,812500000 |
| 2 | 0,750000000 | 1,000000000 | 0,812500000 | 0,187500000 | 0,929687500 |
| 3 | 0,750000000 | 1,000000000 | 0,929687500 | 0,070312500 | 0,973632812 |
| 4 | 0,750000000 | 1,000000000 | 0,973632812 | 0,026367180 | 0,990112305 |
| 5 | 0,750000000 | 1,000000000 | 0,990112305 | 0,009887695 | 0,996292114 |
| 6 | 0,750000000 | 1,000000000 | 0,996292114 | 0,003707886 | 0,998609543 |
| 7 | 0,750000000 | 1,000000000 | 0,998609543 | 0,001390457 | 0,999478579 |
| 8 | 0,750000000 | 1,000000000 | 0,999478579 | 0,000521421 | 0,999804497 |
| 9 | 0,750000000 | 1,000000000 | 0,999804497 | 0,000195503 | 0,999926686 |
| 10 | 0,750000000 | 1,000000000 | 0,999926686 | 0,000073314 | 0,999972522 |
| 11 | 0,750000000 | 1,000000000 | 0,999972522 | 0,000027478 | 0,999989688 |
| 12 | 0,750000000 | 1,000000000 | 0,999989688 | 0,000010312 | 0,999996126 |
| 13 | 0,750000000 | 1,000000000 | 0,999996126 | 0,000003874 | 0,999998569 |
| 14 | 0,750000000 | 1,000000000 | 0,999998569 | 0,000001431 | 0,999999464 |
| 15 | 0,750000000 | 1,000000000 | 0,999999464 | 0,000000536 | 0,999999821 |
| 16 | 0,750000000 | 1,000000000 | 0,999999821 | 0,000000179 | 0,999999940 |
| 17 | 0,750000000 | 1,000000000 | 0,999999940 | 0,000000060 | 1,000000000 |

A Tabela 4 mostra os dados obtidos, utilizando o Fator de aprendizagem 0,75.

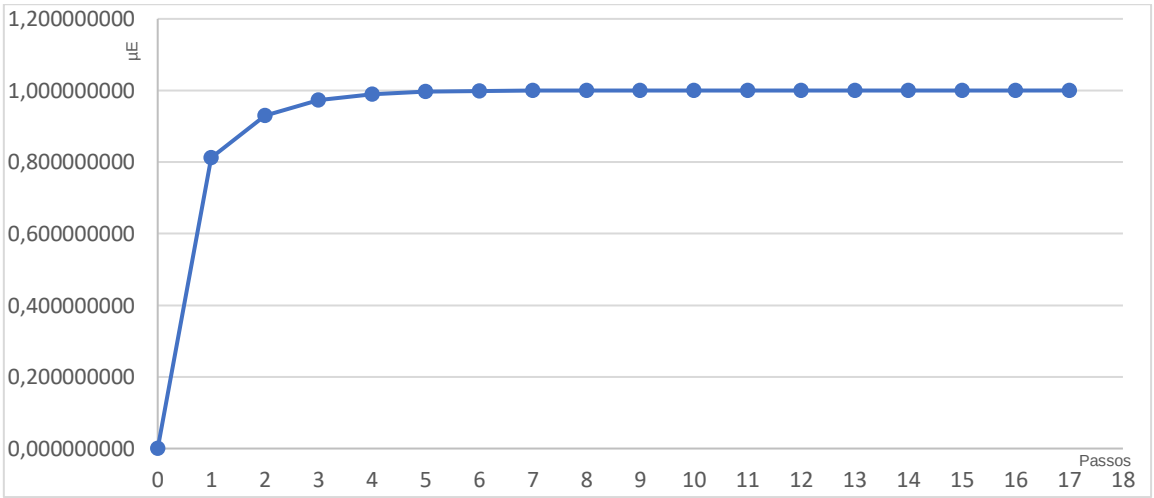



**Gráfico 5. Dados Obtidos com a Implementação com Linguagem de Programação C++. Fator de aprendizagem 0,75.**

No Gráfico 5, é mostrado o comportamento de aprendizagem obtido no treinamento da CNAPap.

Verificou-se que, a célula aprendeu em 17 passos.

**Tabela 5. Dados Obtidos com a Implementação com Linguagem de Programação C++. Fator de aprendizagem 1,00.**

| Passos | FA | µ1 | Gc | Gct | µE |
|---|---|---|---|---|---|
| 0 | 1,000000000 | 1,000000000 | 0,000000000 | 0,500000000 | 0,000000000 |
| 1 | 1,000000000 | 1,000000000 | 0,500000000 | 0,500000000 | 0,750000000 |
| 2 | 1,000000000 | 1,000000000 | 0,750000000 | 0,250000000 | 0,875000000 |
| 3 | 1,000000000 | 1,000000000 | 0,875000000 | 0,125000000 | 0,937500000 |
| 4 | 1,000000000 | 1,000000000 | 0,937500000 | 0,062500000 | 0,968750000 |
| 5 | 1,000000000 | 1,000000000 | 0,968750000 | 0,031250000 | 0,984375000 |
| 6 | 1,000000000 | 1,000000000 | 0,984375000 | 0,015625000 | 0,992187500 |
| 7 | 1,000000000 | 1,000000000 | 0,992187500 | 0,007812500 | 0,996093750 |
| 8 | 1,000000000 | 1,000000000 | 0,996093750 | 0,003906250 | 0,998046875 |
| 9 | 1,000000000 | 1,000000000 | 0,998046875 | 0,001953125 | 0,999023438 |
| 10 | 1,000000000 | 1,000000000 | 0,999023438 | 0,000976562 | 0,999511719 |
| 11 | 1,000000000 | 1,000000000 | 0,999511719 | 0,000488281 | 0,999755859 |
| 12 | 1,000000000 | 1,000000000 | 0,999755859 | 0,000000000 | 1,000000000 |

A Tabela 5 mostra os dados obtidos, utilizando o Fator de aprendizagem 1,00.

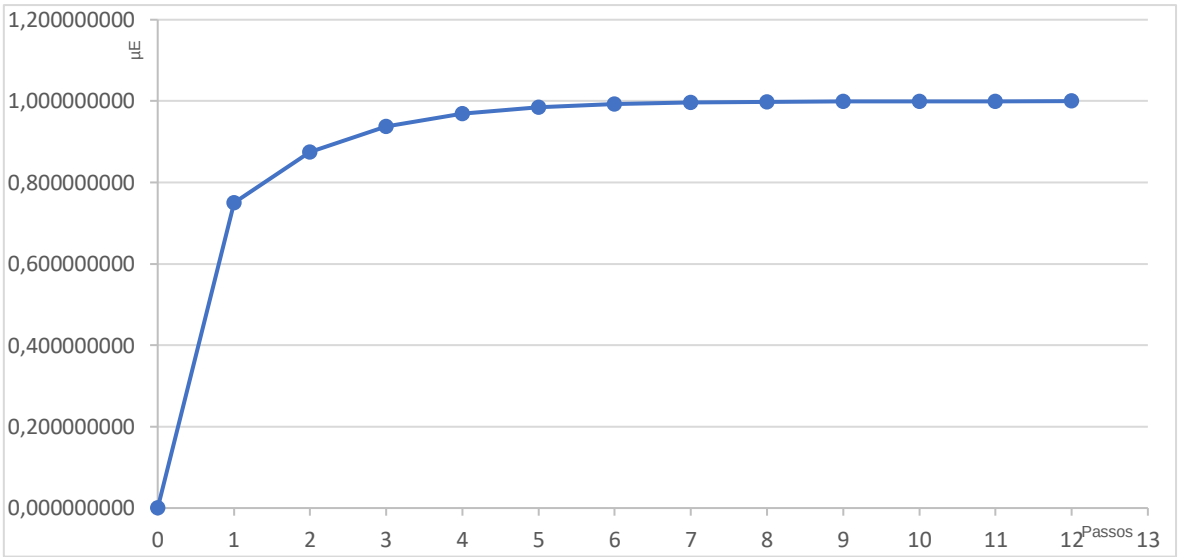



**Gráfico 6. Dados Obtidos com a Implementação da Linguagem de Programação C++. Fator de aprendizagem 1,00**

No Gráfico 6, é mostrado o comportamento de aprendizagem obtido, no treinamento da CNAPap.

Verificou-se, que a célula aprendeu em 12 passos.

**Tabela 6. Dados Obtidos com o algoritmo LPA2v, aplicado na modelagem com uma célula de aprendizagem, implementada no *chip* ATmega328P, 10 passos e Fator de aprendizagem 1,00.**

| Passos | FA | µ1 | Gc | Gct | µRE |
|---|---|---|---|---|---|
| 0 | 1,000000000 | 1,000000000 | 0,000000000 | 0,500000000 | 0,000000000 |
| 1 | 1,000000000 | 1,000000000 | 0,500000000 | 0,500000000 | 0,750000000 |
| 2 | 1,000000000 | 1,000000000 | 0,750000000 | 0,250000000 | 0,875000000 |
| 3 | 1,000000000 | 1,000000000 | 0,875000000 | 0,125000000 | 0,937500000 |
| 4 | 1,000000000 | 1,000000000 | 0,937500000 | 0,062500000 | 0,968750000 |
| 5 | 1,000000000 | 1,000000000 | 0,968750000 | 0,031250000 | 0,984375000 |
| 6 | 1,000000000 | 1,000000000 | 0,984375000 | 0,015625000 | 0,992187500 |
| 7 | 1,000000000 | 1,000000000 | 0,992187500 | 0,007812500 | 0,996093750 |
| 8 | 1,000000000 | 1,000000000 | 0,996093750 | 0,003906250 | 0,998046875 |
| 9 | 1,000000000 | 1,000000000 | 0,998046875 | 0,001953125 | 0,999023438 |
| 10 | 1,000000000 | 1,000000000 | 0,999755859 | 0,000000000 | 1,000000000 |

A Tabela 6, mostra os dados obtidos com a Implementação no *chip* ATmega 328P, utilizando o Fator de aprendizagem 1,00.

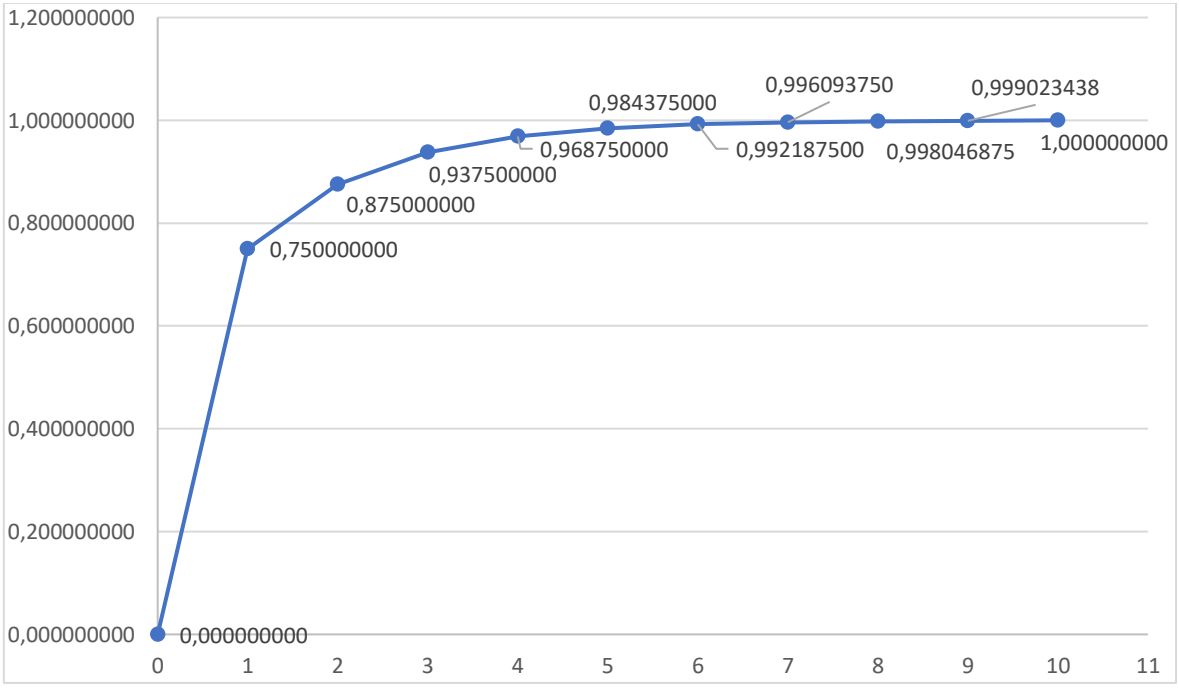



**Gráfico 7. Dados Obtidos com o algoritmo LPA2v, aplicado na modelagem com uma célula de aprendizagem implementada, no *chip* ATmega328P, 10 passos e Fator de aprendizagem 1,00.**

Verificou-se, que a célula aprendeu em 10 passos.

Os valores correspondentes resultantes, para os potenciômetros, que estão relacionados aos ângulos de movimentação, das juntas do Braço Robótico, ficaram:

O ângulo da aprendizagem, na Rotação da Cintura $\Delta\alpha = 80^o$, onde a resistência correspondente do potenciômetro foi de 8,8 KΩ, resultando no Grau de evidência aplicado como Padrão μ2 = 0,88.

O ângulo da aprendizagem, na Rotação do Ombro $\Delta\alpha = 70^o$, onde a resistência correspondente do potenciômetro foi de 7,7 KΩ, resultando no Grau de evidência aplicado como Padrão μ2 = 0,77.

O ângulo da aprendizagem, na Rotação do Cotovelo $\Delta\alpha = 60^o$, onde a resistência correspondente do potenciômetro foi de 6,6 KΩ, resultando no Grau de evidência aplicado como Padrão μ2 = 0,66.

O ângulo da aprendizagem, na Rotação do Punho $\Delta\alpha = 45^o$, onde a resistência correspondente do potenciômetro foi de 5,0 KΩ, resultando no Grau de evidência aplicado como Padrão μ2 = 0,50.

O ângulo da aprendizagem, na Rotação da Garra $\Delta\alpha = 30^o$, onde a resistência correspondente do potenciômetro foi de 3,3 KΩ, resultando no Grau de evidência aplicado como Padrão μ2 = 0,33.

Os resultados destas simulações são apresentados no gráfico 8.

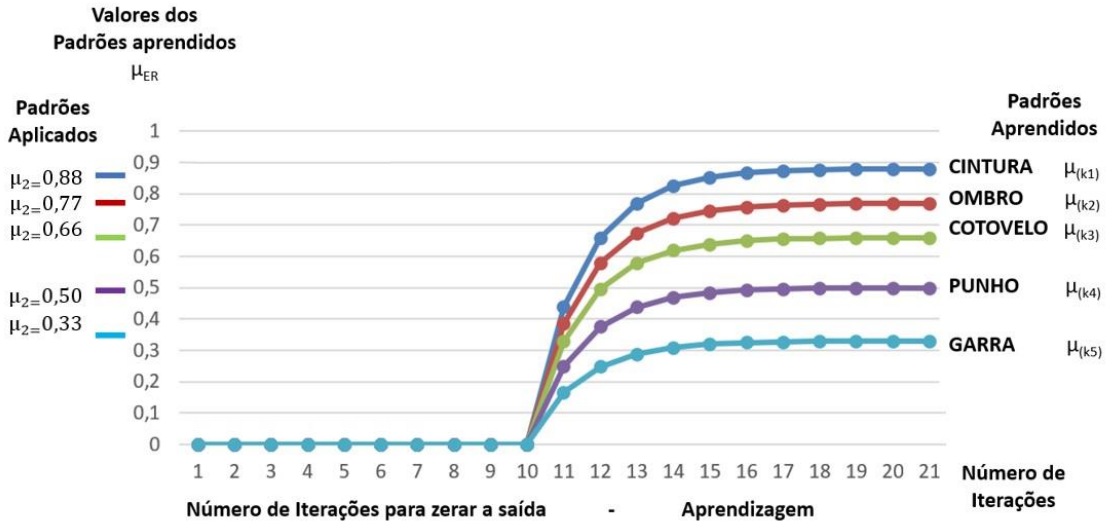



**Gráfico 8. Resultados gráficos dos valores de aprendizagem, com as aplicações dos padrões escolhidos para os testes do Robô Manipulador.**

Através da técnica de Tele operação, com o acionamento do potenciômetro, estabeleceu-se o estado de “Aprendizado por demonstração”, onde foi aplicado *n* vezes o padrão $\mu 2$ na entrada da CNAPap, até o aprendizado ser completado. Após este processo, o Braço Robótico foi colocado no estado de “Imitação”, onde os dispositivos acionaram, os servos motores das juntas, movimentando-as nos ângulos limites que foram estipulados.

Para verificação da eficiência do aprendizado por demonstração, pelo sistema construído, durante o processo de Imitação, foram medidos os valores dos ângulos, e corresponderam aos esperados.

Na Tabela 7, foram mostrados os valores médios das respostas, após 20 verificações dos resultados das imitações, com a correspondente porcentagem de acertos, encontradas para as cinco juntas e a Garra do Braço Robótico.

**Tabela 7. Valores obtidos nos testes, onde o total de demonstrações Nd=10 e o total de verificações da imitação *N*I=20.**

| Parte do Robô: **CINTURA** | | | | | |
|---|---|---|---|---|---|
| Rotação de aprendizagem | Sensor Resistivo | Voltagem Realimentação | Padrão de aplicação | Acertos nos limites de erro $\Delta\alpha \pm 10\%$ | Eficiência |
| $\Delta\alpha$ 80º | 8,8 KΩ | 4,4 V | $\mu 2 = 0,88$ | 15 | 75 % |
| | | | | | |
| Parte do Robô: | **OMBRO** | | | | |
| Rotação de aprendizagem | Sensor Resistivo | Voltagem Realimentação | Padrão de aplicação | Número de acertos nos limites $\Delta\alpha \pm 10\%$ | Eficiência |
| $\Delta\alpha$ 70º | 7,7 KΩ | 3,85 V | $\mu 2 = 0,77$ | 18 | 90 % |
| | | | | | |
| Parte do Robô: | **COTOVELO** | | | | |
| Rotação de aprendizagem | Sensor Resistivo | Voltagem Realimentação | Padrão de aplicação | Número de acertos nos limites $\Delta\alpha \pm 10\%$ | Eficiência |
| $\Delta\alpha$ 60º | 6,6 KΩ | 3,3 V | $\mu 2 = 0,66$ | 14 | 70 % |
| | | | | | |
| Parte do Robô: | **PUNHO** | | | | |
| Rotação de aprendizagem | Sensor Resistivo | Voltagem Realimentação | Padrão de aplicação | Número de acertos nos limites $\Delta\alpha \pm 10\%$ | Eficiência |
| $\Delta\alpha$ 45º | 5,0 KΩ | 2,5 V | $\mu 2 = 0,50$ | 17 | 85 % |
| | | | | | |
| Parte do Robô: | **GARRA** | | | | |
| Rotação de aprendizagem | Sensor Resistivo | Voltagem Realimentação | Padrão de aplicação | Número de acertos nos limites $\Delta\alpha \pm 10\%$ | Eficiência |
| $\Delta\alpha$ 30º | 3,3 KΩ | 1,65 V | $\mu 2 = 0,33$ | 15 | 75 % |
| | | | | | |

Os ensaios iniciais utilizando a Linguagem C++, cujos resultados foram apresentados nas Tabelas 2, 3, 4, 5 e 6 e Gráficos 3, 4, 5, 6 e 7 foram feitos utilizando os limites correspondentes aos consolidados nas Tabela 1 e mostrados no Gráfico 2.

Verificou-se, que os resultados mostraram valores, que foram considerados suficientemente compatíveis, para que, a modelagem abrangendo cinco CNAPaps, pudesse ser inserida em um modelo microprocessado, com aplicações em condições reais.

Desse modo, os resultados obtidos na implementação inicial, com a linguagem de programação do algoritmo da CNAPap, em Linguagem C++, permitiu assegurar, que a implementação física, fosse elaborada.

A implementação física para utilização do *chip,* foi composta por uma *interface* própria (*Arduíno Uno*), para a programação do *chip*. A Placa de testes, foi constituída de um microcontrolador de tipo ATmega328P, com instalações de conexões, com o meio físico externo.

Os resultados apresentados na Tabela 6 e Gráfico 7, se mostraram compatíveis para o funcionamento correto, do microcontrolador ATmega328P. Dessa forma, foi selecionado este tipo de microcontrolador, para implementar o algoritmo da CNAPap, por sua programação ser em linguagem C++. Verificou-se que para inserção dos sinais de entrada, foi utilizado um *joystick*, conforme a descrição e estrutura montada (Figura 15 e 16).

Este procedimento permitiu, que o sistema construído em *hardware(Arduíno Uno),* obtivesse os sinais de aprendizagem da célula, inseridos através do *joystick*, de forma que, ao se acionar a sua manopla, como resultado ocorria, a movimentação da garra robótica, representando assim a energização do servo motor.

A Tabela 6 e o Gráfico 7, mostraram os resultados otimizados com valores próprios, para estudar as ações resultantes, do Sistema Paraconsistente. Neste caso, (Figura 24 e 25) cinco potenciômetros substituíram os *joysticks*, e foram usados, para inserção dos sinais analógicos de entrada.

Dois botões de comando, do tipo Liga/Desliga, atribuíram as funções de *RUN* e *EXE*, e um braço articulado, acionado através dos sinais de saída do ATmega328P, em cinco servos motores, executaram a movimentação das articulações.

A modelagem física para utilização do *chip* foi composta por uma *interface*, (*Arduíno Uno*), necessária para a programação do *chip*, através de microcomputador e conexão dos terminais com o meio físico externo.

Conforme mostrado no gráfico 8, os resultados das simulações efetuadas, na CNAPap, programada no microcontrolador ATmega328P, se mostraram muito próximos aos valores esperados, para todos os cinco diferentes padrões aplicados, nas partes constituintes do Robô Manipulador.

Essa proximidade dos resultados obtidos, com os valores esperados, não aconteceu quando os testes foram efetuados, diretamente no Robô Manipulador, através da Tele operação.

Verificou-se, pelos valores resultantes contidos na Tabela 7, que a Cintura e a Garra do Robô, mostraram o menor número de acertos, com uma eficiência de aprendizagem de 75 %. Entre as partes constituintes do Robô Manipulador, a que resultou em maior eficiência, foi o Ombro, que mostrou a 90 % de acertos.

Verificou-se, que as condições estruturais do Robô Manipulador, influenciaram nos resultados dos testes, com a Tele operação.

O modelo do Braço Robótico, apresentou uma estrutura oscilante, e dificultou a adaptações dos sensores nas juntas, ocasionando erros nos padrões de aplicação, por meio da Tele operação.

Este fator, ligado a estrutura mecânica do Robô, e dos componentes sensores, foi o mais influente nos resultados obtidos, no processo de aprendizagem dos graus de abertura das juntas.

Este problema apresentado, na etapa do aprendizado, refletiu na etapa da Imitação. No entanto, tanto na aprendizagem, como na Imitação, o Robô mostrou que possui condições, de recuperação da trajetória ideal, o que mostra robustez a distúrbios.

Os resultados apresentados nas tabelas e gráficos, mostraram que o Sistema Paraconsistente de aprendizado por demonstração, que foi implementado com sucesso no *chip* ATmega328P, se mostrou compatível para a inserção, através de programação o algoritmo da CNAPap.

## 4. CONCLUSÕES

Esta pesquisa teve como principal objetivo, efetuar modelagens baseadas em Lógica Paraconsistente, utilizando o algoritmo da Célula Neural Artificial Paraconsistente de Aprendizagem – CNAPap, como um dispositivo físico, para permitir ensaios e testes de aplicações em robótica, utilizando técnicas de aprendizagem por demonstração.

Considerando este como objetivo principal, o algoritmo da CNAPap, foi inserido em um microcontrolador ATmega328P, sendo representado em um único *chip*. Após, o programa computacional da CNAPap, ser implementado no microcontrolador, foi aplicado em um protótipo de um braço robótico articulado.

Acionando por meio de aprendizagem por demonstração, cinco servo motores CC, complementando o aprendizado por demonstração, e dotando o dispositivo robótico da capacidade de aprender e efetuar tarefas.

A sua implementação mostrada neste trabalho, apresentou um novo método, com propriedades de ser programado em redes de recorrência, criando um Sistema Neural Artificial Paraconsistente de aprendizagem por demonstração - SNAPApdem.

Verificou-se que, o SNAPApdem com a sua capacidade de aprender e guardar padrões, como também fazer, o braço robótico reproduzir tarefas, que foram anteriormente demonstradas, apresentou características similares as do cérebro humano.

Estes resultados apresentados pelo SNAPApdem, abrem um amplo campo de pesquisa, visando o melhoramento deste método, na área de robótica e sistemas autônomos, no que trata de inovações, na técnica da aprendizagem de máquinas por demonstração.

## 4.1 Trabalhos Futuros

A modelagem de um algoritmo da Lógica Paraconsistente, e a sua Inserção em um microcontrolador do tipo ATmega328P, que foi testado na movimentação, de cinco servo motores, em um braço articulado, contribui de forma muito impactante, com a Inteligência Artificial.

Novas estruturas e configurações, podem ser implementados a partir dos resultados, apresentados nesta dissertação. Um dos trabalhos passiveis de futuramente serem desenvolvidos, trata-se de módulos de apoio, que capacitem o SNAPApdem de guardar informações, para que seja feito o monitoramento em tempo real, da efetividade das tarefas, que o equipamento está desenvolvendo no meio físico.

Este módulo teria a função de confirmar, se o Robô está cumprindo a tarefa que recebeu, através do aprendizado por demonstração.

Outro tipo de trabalho que se sugere, para uma futura pesquisa, seria um módulo de apoio para estudo, e atuação de ajustes na ação do robô.

Isto se deve, porque, a modelagem apresentada na dissertação, durante as demonstrações para o treinamento da CNAPap Física, não considerou desvios, sendo, portanto, um caso a ser estudado, em uma futura aplicação.

## REFERÊNCIAS


BENGIO, Y. and LECUN, Y. (2007) **Scaling Learning Algorithms towards AI. Large Scale Kernel Machines**, MIT Press.

BRAGA, A.; LUDERMIR, T. B.; CARVALHO, A. C. P. L. F. **Redes neurais artificiais: teoria e aplicações**. Rio de Janeiro: Livros Técnicos e Científicos Editora S.A., 2000. 101

BORODIN, N., **Machine Design**, 1 ed., MIR Publishers, Moscow,1988.

CHEN J. and A. Zelinsky. **Programing by demonstration: Coping with suboptimal teaching actions**. The International Journal of Robotics Research, 22(5):299–319, 2003.

CRAIG, J., **Introduction to Robotics: Mechanics & Control,** Addison-Wesley Publishing Co., 1 ed., Massachusetts, 1986.

CUTKOSKY, M. R., "**On Grasp Choice, Grasp Models, and the Design of Hands for Manufacturing Tasks**", IEEE Trasactions on Robotics and Automation, v. 5, n.3, pp. 269-279 ,1989.

Da Silva Filho, J. I.. **Métodos de Aplicações da Lógica Paraconsistente Anotada de anotação com dois valores-LPA2v.** Revista Seleção Documental, ISSN 1809-0648, Santos, São Paulo – Brasil, Número 1, Ano 1, p. 18 – 25, Março 2006.

Da Silva Filho, J. I.. **Análises de Sinais de Informações em Lógica Paraconsistente Anotada.** Revista Seleção Documental, ISSN 1809-0648, Santos, São Paulo – Brasil
Número 14, Ano 4, p. 22 – 26, Junho 2009.

Da Costa, N. C. A.; Abe, J. M.; Subrahmanian, V.S. **"Remarks on annotated logic, Zeitschrift f. math. Logik und Grundlagen d. Math".** Vol.37, pp 561-570, 1991.

Da Cruz, C. M. et al. **"Application of Paraconsistent Artificial Neural network in Statistical Process Control acting on voltage level monitoring in Electrical Power Systems – Intelligent System Application to Power Systems (ISAP)",** pp. (1-6), Porto–PT, DOI 10.1109/ISAP.2015.7325579, September, 2015.

Da Silva Filho, J. I.. **"Introdução às Células Neurais Artificiais Paraconsistentes"** Revista Seleção Documental, No 8, Dezembro, 2007.

Da Silva Filho, J. I.; Da Cruz, C. M.; Rocco, A.; Garcia, D. V.; Ferrara, L.F. P. Onuki, A. S.; Mario, . C.; Abe, J. M.. M. A. **"Paraconsistent Artificial Neural Network for structuring Statistical Process Control in Electrical Engineering".** Towards Paraconsistent En- gineering Book, pp 77-102, Springer International Publishing, ISBN 978-3-319-40418-9, 2016.

Da Silva Filho, J. .I. **"Treatment of Uncertainties with Algorithms of the Paraconsistent Annotated Logic",** Journal of Intelligent Learning Systems and Applications, 2012, 4, 144-15. Published Online May 2012 at http://dx.doi.org/10.4236/jilsa.2012.42014.

Da Silva Filho, J. I.; Torres, C. R.; Abe, J. M. **"Robô Móvel Autônomo Emmy: Uma Aplicação eficiente da Lógica Paraconsist Paraconsistente Anotada"** Revista Seleção Documental, No 3, Setembro, 2006, SP. Brasil. ISBN 18090648.

Da Silva Filho, J. I.; Abe, M. J.; Torres, G. L. **Livro: Inteligência Artificial com Redes de Análise Paraconsistentes Teoria e aplicações** Rio de Janeiro – LTC, 2008. ISBN 978-85-216-1631-3.

D'ARPINO, C. Pérez e Shah J.A., **C-LEARN: Learning geometric constraints from demonstrations for multi-step manipulation in shared autonomy** - Robotics and Automation (ICRA), IEEE, 2017

DEMIRIS, Y and G. Hayes**. Imitation as a dual-route process featuring predictive and learning components: a biologically-plausible computational model**. In Dautenhahn and Nehaniv , chapter 13, 2002.

DEMIRIS, Y. & KHADHOURI, B. (2006). **Hierarchical attentive multiple models for execution and recognition of actions**. Robotics and autonomous systems, 54(5), 361–369.

HAYKIN S (1999) **Neural Networks: A Comprehensive Foundation**. Prentice-Hall Inc., New York, 842p.

De Carvalho Jr, A.; Mario, M. C.; Cortês, H. M.; Da Silva Filho, J. I. CNAP Célua Neural Artificial Paraconsistente **Unisanta Science and Technology, 2017, 6, July Published Online 2017 Vol.6 No1** pp 52-54; http://periodicos.unisanta.br/index.php/sat).

Schaal, S. **Learning From Demonstration, sschaal @cc .gatech.edu;** http://www.cc.gatech.edulfac/Stefan.Schaal College of Computing, Georgia Tech, 801 Atlantic Drive, Atlanta, GA 30332-0280 ATR Human Information Processing, 2-2 Hikaridai, Seiko-cho, Soraku-gun, 619-02 Kyoto pp 1041-1044. 1997.

LANGLEY, P., LAIRD, J. E., & ROGERS, S. (2009). **Cognitive architectures: Research issues and challenges**. Cognitive Systems Research, 10(2), 141–160.

Mario, M. C.; Ferrara, L. F. P.; Da Silva Filho, J. I. **"Treinamento de uma Célula Neural Artificial Paraconsistente de Aprendizagem".** Seleção Documental Magazine, No 6, pp 11 – 16, 2007, SP. Brazil. ISBN 18090648.

BREAZEAL, C. & SCASSELLATI, B., **A context-dependent attention system for a social robot**. (pp. 1146–1151), 1999.

GIENGER, M., MUHLIG, M., & STEIL, J. J. **Imitating object movement skills with robots - a task-level approach exploiting generalization and invariance**. In Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on (pp. 1262–1269). IEEE., 2010.

HAYKIN S. **Neural Networks and Learning Machines** - 3 Ed, Bookman, pages 902. 2008.

POOK P. K. and Ballard D. H. **Recognizing teleoperated manipulations**. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'93), 1993.

SCHIAVICCO, L., SICILIANO, B., **Robotica Industriale - Modellistica e Controllo di Manipolatori,** 1 ed., McGraw-Hill Inc., Milano, 1995.

SWEENEY J. D. and R. A. Grupen. **A model of shared grasp affordances from demonstration**. In Proceedings of the IEEE-RAS International Conference on Humanoids Robots, 2007.

Subrahmanian, V. S. **"On the semantics of quantitative logic programs. Proc. 4th. IEEE Symposium on Logic Programming",** Computer Society Press, Washington D.C., 1987.

Ziviani, N. **DEVMEDIA – Plataforma para programadores, 2012 Introdução a Programação Estruturada** https://www.devmedia.com.br/introducao-a-programacao-estruturada/24951 pp 1-5

Todd H.et al. **Deep Q-learning from Demonstrations** Submitted on 12 Apr 2017 (v1), last revised 22 Nov 2017 (this version, v4)]22/10/2017 Published at AAAI 2018. Previously on arxiv as **"Learning from Demonstrations for Real World Reinforcement Learning"** Artificial Intelligence (cs.AI); Machine Learning (cs.LG) arXiv:1704.03732 [cs.AI] (or arXiv:1704.03732v4 [cs.AI] for this version)

Ekvall, S., & Kragic, D. (2008). **Robot learning from demonstration: A task-level planning approach**. International Journal of Advanced Robotic Systems, 5(3):223234.

Liu, T., & Lemeire, J. (2017) **Efficient and Effective Learning of HMMs Based on Identification of Hidden States**. Mathematical Problems in Engineering, vol. 2017,Article ID 7318940, 26 pages, 2017. https://doi.org/10.1155/2017/7318940

Nicolescu, M. N., & Mataric, M. J. (2003). **Natural methods for robot task learning: Instructive demonstrations, generalization and practice**. In Proceedings of the second international joint conference on Autonomous agents and multiagent systems, 2003, 241–248. ACM.

Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., & Barto, A. G. (2015) **Learning grounded finite-state representations from unstructured demonstrations**. The International Journal of Robotics Research. 2015; 34(2):131-157. 10.1177/0278364914554471

Pastor, P., Kalakrishnan, M., Meier, F., Stulp, F., Buchli, J., Theodorou, E., & Schaal, S. (2013). **From dynamic movement primitives to associative skill memories**. Robotics and Autonomous Systems, 2013, 61(4), 351–361.

Schaal, S. (2006) **Dynamic movement primitives-a framework for motor control in humans and humanoid robotics**. in Adaptive Motion of Animals and Machines. Springer, 2006, pp. 261–280.

## APÊNDICE A – Sketch do programa com a definição de variáveis e conversores

```
#define led 4
#define c5
int x=0;
#include <Servo.h>
Servo myservo;
float m1;
void setup(){
Serial.begin(9600);
myservo.attach(3);}
void loop(){
float me;
float m2;
float C4;
float m2c;
float A0 = analogRead(A0);
float A1 = A0;
float A2 = A1;
A2 = (A1)/1024;
m1 = 0,5;
m2c = (1 - A2);
m1 = 1;
C4 = 1;
me = ((m1-(m2c)*C4)+1)/2;
A0 = map(A0, 0 , 1023, 0, 180);
myservo.write(A0);
A0 = analogRead(A0);
A1 = map(A0, 0, 1023, 0, 1);
Serial.println(me);
if(me<0.6){
x++;
if(x==20){
digitalWrite(led,HIGH);
delay(700);
digitalWrite(led,LOW);
x=0;
while(me>0.74){
}
}
Serial.println(x);
delay(150);
        }
        }
```

## ANEXO A – *Joystick*

Os *joysticks* são muito utilizados em equipamentos de movimentação como; prensas não automáticas, ou acionamentos manuais funcionando como *by-pass* de sistemas automáticos (VISHNU et al., 2010). Em jogos de computadores, verifica-se muito a aplicação do *joystick,* como equipamento de inserção de dados. O estudo de mapeamento de *joystick,* é uma pesquisa que ainda está em desenvolvimento, conforme apresentado em Vishnu et al., (2010), onde os autores utilizam um algoritmo que emprega recursos naturais de imagem, para navegação em corredores internos. Este algoritmo é então, adicionado com a entrada de *joystick* padrão do usuário, para assistência progressiva e correção de trajetória. A maioria dessas *interfaces,* tem controles pré-determinados, para movimentar o equipamento que está sob o controle manual. Basicamente, os *joystick*s atuais, têm um *hardware* adicional para coletar sinais, e permite apenas quatro comandos diferentes (dianteiro, traseiro, esquerdo ou direito), em uma velocidade predefinida (VISHNU et al., 2010). Um controlador de *joystick,* é uma ferramenta usada para manobrar um equipamento, convertendo o movimento da mão em símbolos matemáticos. Quando o usuário começa a impulsionar a manopla, o *joystick* recebe as coordenadas cartesianas (x, y) e as converte em coordenadas polares (distância e ângulo). O valor recebido para o eixo x, aumenta à medida que a manopla se move para a direita, e o valor do eixo y, fica maior, à medida que a manopla se move para longe do usuário. Após o processo de controle, ele reconverte novamente as saídas, em uma tensão analógica (VISHNU et al., 2010).

Neste trabalho, foi usado para os testes iniciais em uma CNAPap, um módulo *Joystick* do tipo KY023, conforme mostrado na Figura 24.

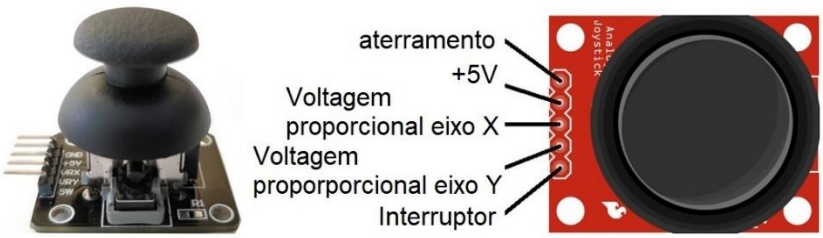



**Figura 24: Módulo *Joystick* utilizado para inserir os sinais de aprendizagem na CNAP.**
**Fonte: (*Datasheet*) (*Joy* – IT /KY-023/1)**

O Módulo *joystick* KY023, foi desenvolvido para aplicações em projetos DIY baseados em microcontroladores, e controle de robôs. Consiste em dois potenciômetros, cada um para um eixo (X e Y). Ambos os potenciômetros de 10kΩ, são independentes para se moverem em suas direções específicas.

O pino interruptor, é conectado a um botão internamente.

A Figura 25, apresenta o diagrama geral do funcionamento, retirado da folha de dados do Módulo *joystick* KY023:

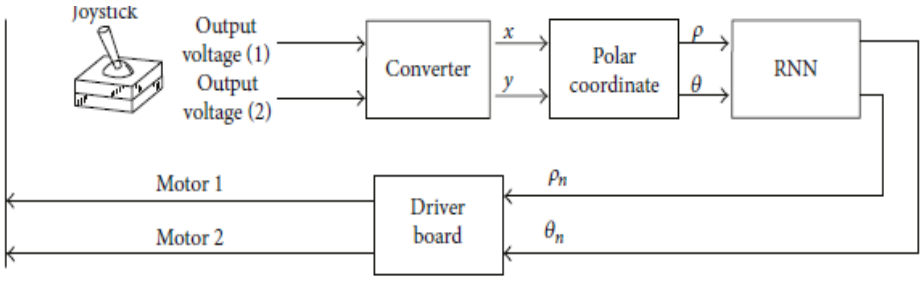



**Figura 25. Visão geral de acionamento por *joystick*.**

**Fonte: (*Datasheet*) (*joystick* KY023)**

### A.1.1 Potenciômetro

Os potenciômetros, são resistores que podem fornecer uma resistência variável, através da rotação de uma haste, no topo de seu corpo. São classificados, com base em dois parâmetros principais, a sua própria resistência (*Ohms*), e a sua potência (*Watts*).

A resistência em Ohms, define a oposição ao fluxo de corrente. Quanto maior o valor do resistor, menor será o fluxo de corrente. Alguns valores para um potenciômetro são: 500Ω, 1KΩ, 2KΩ, 5KΩ, 10KΩ, 22KΩ, 47KΩ, 50KΩ, 100KΩ, 220KΩ, 470KΩ, 500KΩ, 1MΩ.

Os resistores têm dois terminais, mas um potenciômetro tem três terminais (Figura 26).

**Figura 26. Potenciômetro 10KΩ**

**Fonte: (*Datasheet*) (*Model* P232)**

Quanto maior a potência nominal (potência definida em tabela pelo fabricante do componente), maior o resistor, e consequentemente a corrente que circula em seu corpo. Os potenciômetros são usados em circuito de baixa circulação de corrente. Sua construção física, pode ser observada na Figura 27.

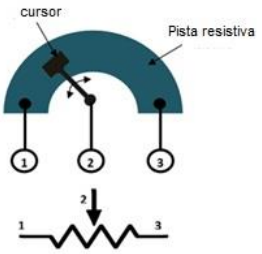



**Figura 27. Diagrama da Pinagem do Potenciômetro 10kΩ**
**Fonte: (*Datasheet*) (*Model* P232)**

O diagrama simboliza, as partes internas de um potenciômetro. Uma pista resistiva de resistência total, corresponde ao valor de resistência nominal do potenciômetro. Como representado no diagrama da Figura 27, um potenciômetro é um resistor com uma extremidade variável. Em um potenciômetro de 10kΩ, medindo a resistência entre o terminal 1 e o terminal 3, será obtido um valor de 10kΩ. Estes terminais são extremidades fixas do potenciômetro. Posicionando o cursor em 25%, do terminal 1, como demonstrado na Figura 27, e executando a medição de resistência, entre o terminal 1 e o terminal 2, o valor obtido será de 25% de 10kΩ, que corresponde à 2,5KΩ. Executando a medição através dos terminais 2 e 3, o valor obtido, será de uma resistência de 7,5KΩ. Os terminais 1 e 2, ou os terminais 2 e 3, podem ser usados para obter a resistência variável e o cursor acoplado à haste, pode ser usado para variar a resistência e definir o valor necessário.

A Figura 28, apresenta o diagrama das dimensões do potenciômetro.

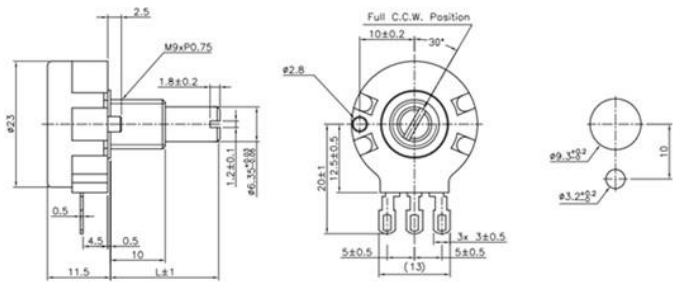



**Figura 28. Diagrama das Dimensões do Potenciômetro 10kΩ**
**Fonte: (*Datasheet*) (*Model* P232)**

### A.1.2 Servo Motor

Servo motor é um dispositivo eletromecânico, utilizado para movimentar com precisão, um objeto, assim permitindo-o girar em ângulos ou distâncias específicas, garantindo o posicionamento e a velocidade. (Figura 29).

**Figura 29. Servo Motor Digital**
**Fonte: (*Datasheet*) (MG996R)**

Um motor elétrico rotativo, acoplado a um sensor que passa a condição de seu posicionamento, permite o controle preciso da velocidade, aceleração e da posição angular. O motor pode ser de corrente contínua, ou de corrente alternada.

Possui este nome por, não ter rotação livre, de forma contínua, como um motor convencional, e responde a um comando pré estabelecido.

Se aplica em sistemas de coordenadas angulares, utilizados em automação (industrial, aeroespacial, agrícola, defesa, médica), em máquinas diversas, braços robóticos, drones, aeromodelos de helicópteros e aviões, e entre muitas outras.

A Figura 30, apresenta o diagrama de ligação do servo motor.

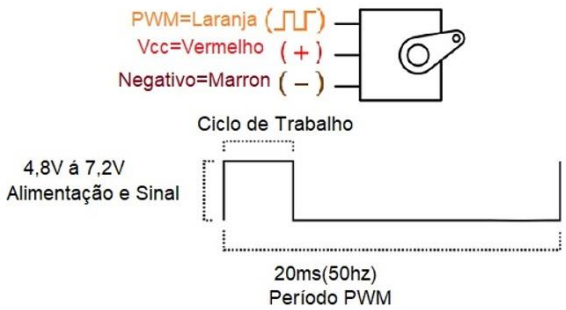



**Figura 30. Diagrama de Ligação do Servo Motor Digital**
**Fonte: (*Datasheet*) (MG996R)**

O servo motor digital MG996R, possui engrenagens de metal, resultando em 10kg extra, de alto torque, em um corpo minúsculo. A engrenagem e o motor desse modelo, foram atualizados para melhorar a largura de banda morta e centralização.

A unidade vem completa, com fio de 30 cm, e conector fêmea tipo 'S' de 3 pinos, que se encaixa na maioria dos receptores (Figura 35).

Este servo motor padrão de alto torque, pode girar aproximadamente 120 graus (60 graus em cada direção). É adaptável a qualquer código de servo, *hardware* ou biblioteca para controle. Recomendado para um controlador de motor com *feedback* e caixa de engrenagens (Figura 31).

Os principais dados técnicos, são apresentados a seguir:

| | |
|---|---|
| Peso: | 55g |
| Dimensão: | 40,7 x 19,7 x 42,9 mm aprox. |
| Torque de parada: | 9,4 kgf · cm (4,8 V), 11 kgf · cm (6 V) |
| Velocidade de operação: | 0,17 s / 60º (4,8 V), 0,14 s / 60º (6 V) |
| Tensão de operação: | 4,8 V a 7,2 V |

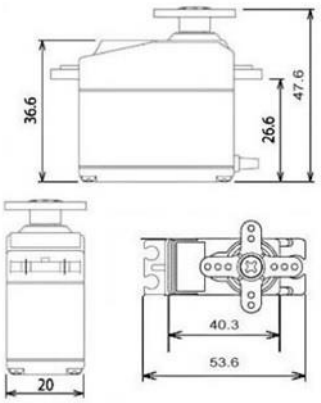



**Figura 31. Diagrama das Dimensões do Servo Motor Digital**
**Fonte: (*Datasheet*) (MG996R)**

### A.1.3 O Microcontrolador ATmega328P

O circuito em *hardware*, do Sistema de Aprendizagem por demonstração com algoritmos da LP, será implementado em um microcontrolador do tipo ATmega328P. As descrições técnicas, e principais propriedades do *chip*, serão descritos a seguir. O componente ATmega328P, pertence a linha de microcontroladores de 8 *bits CMOS*, que se destaca por ser um microcontrolador, com um custo benefício alto.

Esta vantagem do ATmega328P, se deve principalmente, levando em conta a sua alta *performance,* onde consegue realizar 1 Milhão de instruções por segundo por Mega *Hertz* (ATmega328P [*Datasheet*] 7810D-AVR-01/15).

A Figura 32, mostra a apresentação física do *chip* ATmega328P, e sua pinagem correspondente.

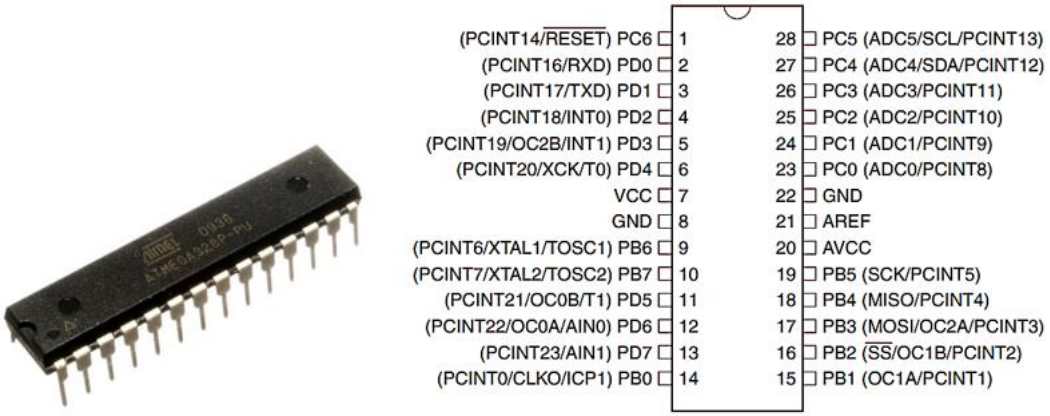



**Figura 32: Apresentação física do *chip* ATmega328P e sua respectiva pinagem.**
**Fonte: (ATmega328P [*Datasheet*] 7810D-AVR-01/15)**

Os principais dados Técnicos do ATmega328P, são apresentados a seguir:

| | |
|---|---|
| Tensão de operação: | 5V |
| Tensão de alimentação máxima: | 5,5 V |
| Entradas e saídas digitais: | 23 |
| Memória *Flash*: | 32 KB |
| Memória SRAM: | 2 KB |
| Memória EEPROM: | 1 KB |
| Velocidade de *clock*: | 0 - 20 MHZ |

**ANEXO B – Programa completo do Protótipo**

```
#include <VarSpeedServo.h>
#include <EEPROM.h>
#define espacoMemoria 199
#define tempoPausaEntreMovimentos 1000
#define pinServo1  3
#define pinServo2  5
#define pinServo3  6
#define pinServo4  9
#define pinServo5  10
#define pinBotao1     2
#define pinBotao2     4
#define pinLedA       12
#define pinLedB       13
#define pinPot1      A5
#define pinPot2      A4
#define pinPot3      A3
#define pinPot4      A0
#define pinPot5      A1
#define pinPot6      A2
VarSpeedServo servo1;
VarSpeedServo servo2;
VarSpeedServo servo3;
VarSpeedServo servo4;
VarSpeedServo servo5;
byte pinBotao1Modo();
bool pinBotao2Retencao();
void pinLedAPisca(bool reset = false);
```

```
void setMemoria(byte posicao, byte servo, byte valor);
byte readMemoria(byte posicao, byte servo);
int ultMemoria;
void setup() {
  Serial.begin(9600);
  servo1.attach(pinServo1);
  servo2.attach(pinServo2);
  servo3.attach(pinServo3);
  servo4.attach(pinServo4);
  servo5.attach(pinServo5);
  pinMode(pinLedA, OUTPUT);
  pinMode(pinLedB, OUTPUT);
  pinMode(pinBotao1, INPUT_PULLUP);
  pinMode(pinBotao2, INPUT_PULLUP);
  ultMemoria = EEPROM.read(0);
void loop() {
static byte modo = 0;
static byte modoAnt;
static byte movimento = 0;
static byte posMemoria = 0;
static unsigned long delayPausa;
if (modo == 0) {
    digitalWrite(pinLedA, LOW);
    if (pinBotao2Retencao()) {
       digitalWrite(pinLedB, HIGH);
         if (movimento == 0) {
         byte velocidade = map(analogRead(pinPot6),0,1023,0,255);
         servo1.slowmove(readMemoria(posMemoria,0), velocidade);
         servo2.slowmove(readMemoria(posMemoria,1), velocidade);
```

```
      servo3.slowmove(readMemoria(posMemoria,2), velocidade);
      servo4.slowmove(readMemoria(posMemoria,3), velocidade);
      servo5.slowmove(readMemoria(posMemoria,4), velocidade);
      movimento = 1;
    }
      if (movimento == 1) {
      if ( (servo1.read() == readMemoria(posMemoria,0)) &&
           (servo2.read() == readMemoria(posMemoria,1)) &&
           (servo3.read() == readMemoria(posMemoria,2)) &&
           (servo4.read() == readMemoria(posMemoria,3)) &&
           (servo5.read() == readMemoria(posMemoria,4)) ) {
        posMemoria++;
        if (posMemoria > ultMemoria) { posMemoria = 0; }
        delayPausa = millis();
        movimento = 2;
      }
    }
       if (movimento == 2) {
       if ((millis() - delayPausa) > tempoPausaEntreMovimentos) {
          movimento = 0;
       }
    }
  } else {
    digitalWrite(pinLedB, LOW);
          if (pinBotao1Modo() == 2) {
       modo = 1;
    }
  }
}
```

```
if (modo == 1) {
if (modoAnt == 0) {
    pinLedAPisca(true);
    ultMemoria = -1;
    EEPROM.write(0, ultMemoria);
 }
 digitalWrite(pinLedB, LOW);
 pinLedAPisca();
 if (pinBotao1Modo() == 2) {
    modo = 0;
 }
servo1.write( map(analogRead(pinPot1),0,1023,0,179) );
int x = 0;
float m1;
float me;
float m2;
float C4;
float m2c;
float A5 = analogRead(pinPot1);
float A1a = A5;
float A2a = A1a;
A2a = (A1a)/1024;
m1 = 0.5;
m2c = (1 - A2a);
m1 = 1;
C4 = 1;
A5 = map(A5, 0, 1023, 0, 180);
A5 = analogRead(pinPot1);
me = ((m1-(m2c)*C4)+1)/2;
```

```
if (me > 0.51 ) {
x++;
if (x==5){
digitalWrite(4,LOW);
delay(100);
}else{
delay(0.1);
digitalWrite(4,HIGH);
x = 0;
}
}
if (me > 0.99 ) {
x++;
if (x==5){
digitalWrite(4,LOW);
delay(100);
}else{
delay(0.1);
digitalWrite(4,HIGH);
x = 0;
}
}
servo2.write( map(analogRead(pinPot2),0,1023,0,179) );
int xb = 0;
float m1b;
float meb;
float m2b;
float C4b;
float m2cb;
```

```
float A4 = analogRead(pinPot2);
float A1b = A4;
float A2b = A1b;
A2b = (A1b)/1024;
m1b = 0.5;
m2cb = (1 - A2b);
m1b = 1;
C4b = 1;
A4 = map(A4, 0, 1023, 0, 180);
A4 = analogRead(pinPot2);
meb = ((m1b-(m2cb)*C4b)+1)/2;
if (meb > 0.51 ) {
xb++;}
if (xb==1){
digitalWrite(4,LOW);
delay(1);
}else {
  delay(1);
digitalWrite(4,HIGH);
xb = 0;
}
if (meb > 0.99 ) {
xb++;}
if (xb==1){
digitalWrite(4,LOW);
delay(1);
}else {
  delay(1);
digitalWrite(4,HIGH);
```

```
xb = 0;
}
{
}
servo3.write( map(analogRead(pinPot3),0,1023,0,179) );
int xc = 0;
float m1c;
float mec;
float m2c2;
float C4c;
float m2cc;
float A3 = analogRead(pinPot3);
float A1c = A3;
float A2c = A1c;
A2c = (A1c)/1024;
m1c = 0.5;
m2c2 = (1 - A2c);
m1c = 1;
C4c = 1;
A3 = map(A3, 0, 1023, 0, 180);
A3 = analogRead(pinPot3);
mec = ((m1c-(m2cc)*C4c)+1)/2;
if (mec > 0.51 ) {
xc++;}
if (xc==1){
digitalWrite(4,LOW);
delay(0.1);
}else {
  delay(0.1);
```

```
digitalWrite(4,HIGH);
xc = 0;
}
if (mec > 0.99 ) {
xc++;}
if (xc==1){
digitalWrite(4,LOW);
delay(0.1);
}else {
  delay(0.1);
digitalWrite(4,HIGH);
xc = 0;
}
{
}
servo4.write( map(analogRead(pinPot4),0,1023,0,179) );
int xd = 0;
float m1d;
float med;
float m2d;
float C4d;
float m2cd;
float A0 = analogRead(pinPot4);
float A1d = A0;
float A2d = A1d;
A2d = (A1d)/1024;
m1d = 0.5;
m2d = (1 - A2d);
m1d = 1;
```

```
C4d = 1;
 A0 = map(A0, 0, 1023, 0, 180);
A0 = analogRead(pinPot4);
med = ((m1d-(m2cd)*C4d)+1)/2;
if (med > 0.51 ) {
xd++;}
if (xd==1){
digitalWrite(4,LOW);
delay(0.1);
}else {
  delay(0.1);
digitalWrite(4,HIGH);
xd = 0;
}
if (med > 0.99 ) {
xd++;}
if (xd==1){
digitalWrite(4,LOW);
delay(0.1);
}else {
  delay(0.1);
digitalWrite(4,HIGH);
xd = 0;
}
{
}
servo5.write( map(analogRead(pinPot5),0,1023,0,179) );
int xe = 0;
float m1e;
```

```
float mee;
float m2e;
float C4e;
float m2ce;
float A1 = analogRead(pinPot5);
float A1e = A1;
float A2e = A1e;
A2e = (A1e)/1024;
m1e = 0.5;
m2e = (1 - A2e);
m1e = 1;
C4e = 1;
A1 = map(A1, 0, 1023, 0, 180);
A1 = analogRead(pinPot5);
mee = ((m1e-(m2ce)*C4e)+1)/2;
if (mee > 0.51 ) {
xe++;}
if (xe==1){
digitalWrite(4,LOW);
delay(0.1);
}else {
  delay(0.1);
digitalWrite(4,HIGH);
xe = 0;
}
if (mee > 0.99 ) {
xd++;}
if (xe==1){
digitalWrite(4,LOW);
```

```
    delay(0.1);
    }else {
      delay(0.1);
    digitalWrite(4,HIGH);
    xe = 0;
    }
    {
    }
      if (pinBotao2Apertado()) {
        ultMemoria++;
        EEPROM.write(0, ultMemoria);
        setMemoria(ultMemoria, 0, map(analogRead(pinPot1),0,1023,0,179));
        setMemoria(ultMemoria, 1, map(analogRead(pinPot2),0,1023,0,179));
        setMemoria(ultMemoria, 2, map(analogRead(pinPot3),0,1023,0,179));
        setMemoria(ultMemoria, 3, map(analogRead(pinPot4),0,1023,0,179));
        setMemoria(ultMemoria, 4, map(analogRead(pinPot5),0,1023,0,179));
        digitalWrite(pinLedB, HIGH);
        delay(250);
        digitalWrite(pinLedB, LOW);
        if ( ultMemoria == (espacoMemoria - 1)) {
            modo = 0;
        }
     }
  }
  modoAnt = modo;
}
byte pinBotao1Modo() {
#define tempoDebounce 40
#define tempoBotao    1500
```

```
bool estadoBotao;
static bool estadoBotaoAnt;
static byte estadoRet = 0;
static unsigned long delayBotao = 0;
static byte enviado = 0;
if ( (millis() - delayBotao) > tempoDebounce ) {
    estadoBotao = digitalRead(pinBotao1);
    if (estadoRet == 0) {
       if ( !estadoBotao && (estadoBotao != estadoBotaoAnt) ) {
          estadoRet = 1;
          delayBotao = millis();
       }
    }
    if (estadoRet == 1) {
       if ( estadoBotao && (estadoBotao != estadoBotaoAnt) ) {
          estadoRet = 0;
          delayBotao = millis();
       }
    }
    estadoBotaoAnt = estadoBotao;
}
if (estadoRet == 1) {
   if ((millis() - delayBotao) > tempoBotao) {
      estadoRet = 2;
      delayBotao = millis();
    }
}
if (estadoRet == 2) {
   enviado++;
```

```
      if (enviado >= 2) {
        estadoRet = 0;
        delayBotao = millis();
        enviado = 0;
      }
   }
      return estadoRet;
}
bool pinBotao2Retencao() {
#define tempoDebounce 40
 bool estadoBotao;
   static bool estadoBotaoAnt;
   static bool estadoRet = false;
   static unsigned long delayBotao = 0;
   if ( (millis() - delayBotao) > tempoDebounce ) {
      estadoBotao = digitalRead(pinBotao2);
      if ( !estadoBotao && (estadoBotao != estadoBotaoAnt) ) {
         estadoRet = !estadoRet;
         delayBotao = millis();
      }
      if ( estadoBotao && (estadoBotao != estadoBotaoAnt) ) {
         delayBotao = millis();
      }
      estadoBotaoAnt = estadoBotao;
   }
   return estadoRet;
}
bool pinBotao2Apertado() {
#define tempoDebounce 40
```

```
  bool estadoBotao;
  static bool estadoBotaoAnt;
  static bool estadoRet;
  static unsigned long delayBotao = 0;
    estadoRet = false;
  if ( (millis() - delayBotao) > tempoDebounce ) {
      estadoBotao = digitalRead(pinBotao2);
      if ( !estadoBotao && (estadoBotao != estadoBotaoAnt) ) {
        estadoRet = true;
        delayBotao = millis();
      }
      if ( estadoBotao && (estadoBotao != estadoBotaoAnt) ) {
        delayBotao = millis();
      }
      estadoBotaoAnt = estadoBotao;
  }
  return estadoRet;
  }
void pinLedAPisca(bool reset) {
static unsigned long delayPisca = 0;
  if (reset) {
     delayPisca = millis();
  } else {
    if ((millis() - delayPisca) < 500) {
       digitalWrite(pinLedA, LOW);
    } else {
       digitalWrite(pinLedA, HIGH);
    }
    if ((millis() - delayPisca) >= 1000) {
```

```
        delayPisca = millis();
     }
   }
}
void setMemoria(byte posicao, byte servo, byte valor) {
int posMem;

    posMem = ((posicao * 5) + servo) + 1;
    EEPROM.write(posMem, valor);
}
byte readMemoria(byte posicao, byte servo) {
int posMem;
    posMem = ((posicao * 5) + servo) + 1;
    return EEPROM.read(posMem);
    }
```