DESENVOLVIMENTO DE UMA BIBLIOTECA PARA SENSOR ULTRASSÔNICO DO ARDUINO COM LÓGICA PARACONSISTENTE

Kauan Victor Aguiar da Cruz[1], Franklyn Brito Mourao de Oliveira[2], Samuel Pinheiro Gonçalves[3], Thiago Reis da Silva[4]

RESUMO

O projeto envolveu a criação de uma biblioteca destinada à plataforma Arduino, com o propósito de integrar sensores ultrassônicos aos princípios da lógica paraconsistente. Esta inovação visa oferecer uma abordagem eficiente para lidar com dados paradoxais ou inconsistentes no contexto do Arduino, um obstáculo frequente em aplicações do mundo real. A biblioteca é pertinente a diversos cenários, incluindo robótica e automação, proporcionando uma solução que melhora a precisão e a confiabilidade das medições dos sensores. Ademais, a aplicação desta biblioteca em competições de robótica pode proporcionar uma vantagem competitiva considerável, ao otimizar a navegação e a capacidade de detecção de obstáculos dos robôs.

PALAVRAS-CHAVE: Biblioteca; Sensor Ultrassônico; Lógica Paraconsistente.

FINANCIAMENTO: Fundação de Amparo à Pesquisa do Maranhão (FAPEMA).

1. INTRODUÇÃO

Nos últimos anos, houve muitos avanços tecnológicos, fomentados por necessidades de sistemas inteligentes e adaptativos, indispensáveis para o tratamento de informações complexas e, muitas vezes, contraditórias, oriundas de ambientes dinâmicos do mundo real. Nesse contexto, a computação embarcada ocupa um papel estratégico,

[1] Estudante bolsista – Curso de Tecnologia em Redes de Computadores – IFMA/Campus São João dos Patos; kauana@acad.ifma.edu.br
[2] Técnico de Laboratório de Informática Esp. – Membro do Projeto/Orientador – IFMA/Campus São João dos Patos; E-mail: franklyn.oliveira@ifma.edu.br
[3] Professor de Mecânica/Automoção Industrial Me – Membro do Projeto/Orientador – IFMA/Campus São João dos Patos; E-mail: samuelg.itz@ifma.edu.br
[4] Professor de Informática Dr – Coordenador do Projeto/Orientador – IFMA/Campus São João dos Patos; E-mail: thiago.reis@ifma.edu.br

fornecendo soluções a baixo custo e alta versatilidade para uma ampla gama de aplicações. Dentre as plataformas disponíveis, o Arduino é reconhecido pela facilidade de uso em projetos eletrônicos (MCROBERTS, 2011; EVANS; NOBLE e HOCHENBAUM, 2013). A versatilidade e modularidade fazem do Arduino um ambiente adequado para muitos projetos, desde dispositivos de controle simples a sistemas de monitoramento e interação com o ambiente (COUTINHO JÚNIOR, 2021; FURTADO e PAIVA, 2023).

Apesar da ampla adoção do Arduino, seu processamento lógico geralmente depende da lógica clássica, limitada pelo princípio do terceiro excluído. Essa abordagem não lida bem com dados paradoxais ou inconsistentes, comuns em cenários reais, levando ao problema do princípio da explosão, em que uma contradição torna todo o sistema lógico inválido. Assim, a lógica clássica mostra-se frágil e pouco prática para aplicações em ambientes dinâmicos, como monitoramento, controle industrial e robótica autônoma.

Diante desafios semelhantes, a lógica paraconsistente se configura como uma forma de pensar que ultrapassa a lógica clássica. Ao contrário da lógica clássica, a lógica paraconsistente foi pensada para enfrentar informações inconsistentes ou paradoxais de forma que permita a coexistência de informações contraditórias sem que isso conduza a uma trivialização do sistema (NASCIMENTO, 2021; OLIVEIRA et al., 2023). Este modelo oferece uma forma de pensar problemas em que a informação pode ser duvidosa ou conflituosa. Algumas de suas várias aplicações incluem áreas como inteligência artificial, sistemas de diagnóstico médico, engenharia de software e, de forma marcante, robótica, em que pode ser utilizada a fim de administrar incertezas e imprecisões sensoriais (OLIVEIRA *et. al*. 2023).

A proposta de uma biblioteca para Arduino que compreendesse os conceitos da lógica paraconsistente é válida pelos seguintes motivos: o fato de que se pode lidar com informações contraditórias ou incertas sem eliminá-las, preservando a consistência do sistema (SILVA FILHO, 1999). O projeto desenvolveu uma biblioteca para a plataforma Arduino objetivando integrar sensores ultrassônicos com conceitos de lógica paraconsistente e, assim, lidar com informações paradoxais ou inconsistentes neste microcontrolador.

2. METODOLOGIA

A condução dos estudos seguiu uma abordagem aplicada, fundamentada em pesquisa tecnológica, com foco no desenvolvimento de uma solução voltada à integração da lógica paraconsistente em sistemas embarcados utilizando o Arduino. Conforme

ressalta Marcondes *et. al.* (2017), a pesquisa aplicada busca gerar conhecimento direcionado à solução de problemas práticos, característica que norteou todas as etapas deste trabalho. Inicialmente, procedeu-se a uma revisão bibliográfica sobre os fundamentos da lógica paraconsistente, contemplando não apenas seus princípios teóricos, mas também suas aplicações em contextos computacionais. Nesse processo, destacou-se a Lógica Paraconsistente Anotada de Anotação com Dois Valores (LPA2v), que serviu como base conceitual para o projeto. Foram pesquisados aspectos como os graus de crença (μ1) e descrença (μ2) e sua interpretação no Quadrado Unitário do Plano Cartesiano (QUPC), importante para o entendimento da dinâmica lógica adotada. Em paralelo, analisou-se a estrutura e o funcionamento do sensor ultrassônico HC-SR04, utilizado em projetos de Arduino para medições de distância. Sendo assim, o foi organizado o projeto nas fases destacadas a seguir.

Na Fase 1, denominada Desafio Técnicos, foram identificados desafios técnicos. As bibliotecas existentes para Arduino, por aderirem à lógica clássica, apresentavam limitações na capacidade de processamento lógico para lidar com informações inconsistentes. Adicionalmente, o código básico do sensor HC-SR04 apresentava restrições ao tratar medições inconsistentes ou ruídos ambientais. A transformação de medições físicas em proposições lógicas, fundamental para a aplicação da lógica paraconsistente, também se apresentou como um desafio, visto que o algoritmo padrão do HC-SR04 realizava apenas cálculos lineares sem avaliar a qualidade dos dados. A solução para esses desafios foi o desenvolvimento de uma biblioteca paraconsistente específica para o Arduino, focada na integração com sensores ultrassônicos, baseando-se nos conceitos matemáticos dos graus de crença (μ1) e descrença (μ2) para quantificar a confiabilidade das medições.

Na Fase 2, foi realizado o Desenvolvimento da Biblioteca Paraconsistente para Arduino, iniciou-se com a definição dos componentes de hardware necessários. Foram selecionadas placas Arduino UNO, o sensor ultrassônico HC-SR04, jumpers para conexões elétricas, resistores para controle de corrente e uma base de prototipagem (*breadboard*) para montagem dos circuitos.

O ambiente de desenvolvimento de software adotado foi o Arduino IDE. Este ambiente proporcionou a flexibilidade e as ferramentas necessárias para a implementação e depuração do código da biblioteca.

Com base no ambiente de desenvolvimento e nos componentes definidos, a implementação da biblioteca paraconsistente foi estruturada em três etapas principais. A

primeira etapa envolveu a construção de classes utilizando os princípios da Programação Orientada a Objetos (POO). O objetivo dessa abordagem foi encapsular as funcionalidades da lógica paraconsistente em uma estrutura fácil integração com projetos baseados em Arduino.

A criação da classe *ultra_s* exemplifica essa etapa, estabelecendo uma estrutura para lidar com contradições e incertezas, e para verificar a consistência das medições. Esta classe inclui membros públicos e privados, como o construtor ultra_s(int trig, int echo) para inicializar os pinos do sensor, e funções como *measureDistance* e *paraconsistentMeasureDistance* para as operações de medição e análise lógica.

A fase 3 foi implementada as funções de Medição e Análise Lógica. Nesta fase focou na implementação das funções responsáveis pela medição da distância e pelo processamento dos parâmetros obtidos pelo sensor.

A função *measureDistance* é a base para a leitura do sensor ultrassônico. Ela configura o pino *_trig* para emitir um pulso de ultrassom: primeiro, um breve período LOW (2 microssegundos) para garantir um sinal limpo, seguido por um pulso HIGH de 10 microssegundos para a emissão sonora. Após a emissão, o pino *_trig* retorna a LOW. O tempo de resposta do sinal refletido é então capturado pelo pino _echo utilizando a função *pulseIn*, que armazena a duração do pulso. Essa duração é convertida em distância pela fórmula *duration* * 0.034 / 2, onde 0.034 representa a velocidade do som em centímetros por microssegundo, e a divisão por 2 compensa a viagem de ida e volta do som. A função permite selecionar a unidade de medida (centímetros ou polegadas) através do parâmetro *char opt*.

A função *paraconsistentMeasureDistance* melhora a precisão das medições ao validar a consistência dos valores obtidos pelo sensor. Ela recebe duas medições de distância e utiliza uma função auxiliar, *isConsistent*, para verificar se a diferença entre elas está dentro de uma tolerância pré-definida. Se as medições forem consistentes, a função retorna a média aritmética dos valores, proporcionando um resultado mais confiável. Caso contrário, se houver uma contradição (diferença acima da tolerância), a função retorna -1, indicando uma medição inconsistente.

A função *isConsistent* é fundamental para essa validação. Ela recebe duas distâncias (*distance1*, *distance2*) e uma *tolerance* (padrão de 1.0), retornando *true* se o valor absoluto da diferença entre as duas distâncias for menor ou igual à tolerância, e *false* caso contrário. Isso permite filtrar dados inconsistentes gerados por ruído ou imprecisões.

A integração da lógica paraconsistente no sistema de medição ultrassônica foi realizada para tratar incertezas e contradições inerentes à aquisição de dados sensoriais. Para superar as limitações do código do HC-SR04, a biblioteca foi fundamentada nos conceitos matemáticos dos graus de crença (μ1) e descrença (μ2), que quantificam a confiabilidade das medições. O cálculo desses graus baseia-se na variação entre múltiplas leituras do sensor, permitindo uma análise dos dados. A implementação foi realizada através da classe LPA2v, que encapsula os algoritmos para análise paraconsistente, incluindo os cálculos dos graus de certeza (Gc) e incerteza (Gi).

Por fim, a fase 4 foi realizada a Integração do Sistema e Classificação de Medições, envolvendo a integração da biblioteca desenvolvida a um código funcional de sensor ultrassônico. Essa integração demonstrou a aplicabilidade da biblioteca em cenários reais, onde a presença de ruído e dados contraditórios é comum. O sistema resultante oferece ao usuário a capacidade de classificar as medições em quatro estados distintos: verdadeiro, falso, inconsistente ou indefinido, em conformidade com os princípios da lógica paraconsistente.

Essa abordagem permitiu que um dispositivo originalmente baseado em lógica booleana passasse a avaliar suas medições com base em estados complexos, superando a limitação das bibliotecas que aderem estritamente à lógica clássica.

3. RESULTADOS E DISCUSSÕES

3.1 Lógica Paraconsistente

A lógica paraconsistente é um campo da lógica que teve sua origem devido às limitações da lógica clássica para lidar com informações contraditórias ou inconsistentes. Enquanto a lógica clássica se fundamenta no princípio do terceiro excluído, que afirma que uma proposição deve ser verdadeira ou falsa, a lógica paraconsistente permite a coexistência de informações contraditórias (ABE, KAMA e NAKAMATSU, 2015). Um dos fundamentos da lógica paraconsistente é o que se denomina o “princípio da tolerância à contradição”, que admite a possibilidade de que contradições possam ser verdadeiras (NASCIMENTO et al, 2021). Isso quer dizer que, em certos contextos, o fato de duas proposições contraditórias coexistirem não implica uma inconsistência lógica. Uma das maneiras em como a lógica paraconsistente tem sido trabalhada é através da implementação de sistemas lógicos paraconsistentes, que desenvolvem uma forma de lidar com informações inconsistentes de maneira coerente (OLIVEIRA e COLS, 2023).

3.2. Lógica Paraconsistente Anotada de Anotação com Dois Valores (LPA2v)

A Lógica Paraconsistente Anotada com Dois Valores (LPA2v) é uma classe de lógica paraconsistente inspirada na Lógica Evidencial, que trabalha com premissas oferecendo apenas evidências parciais às conclusões. Nessa abordagem, cada proposição recebe uma anotação representada por um par ordenado (μ1,μ2)(μ1, μ2)(μ1,μ2), em que μ1μ1μ1 expressa o grau de crença e μ2μ2μ2 o grau de descrença, ambos variando independentemente no intervalo [0,1]. Essa independência permite representar situações de inconsistência (alta crença e alta descrença) ou indefinição (baixa crença e baixa descrença), superando limitações de sistemas que consideram esses valores como complementares (SILVA FILHO, 1999).

3.3 Arduino

O Arduino é uma plataforma aberta de prototipagem eletrônica que se destaca pela acessibilidade, versatilidade e facilidade de uso. Com placas baseadas em microcontroladores programáveis em linguagem C/C++, permite a interação com o mundo físico por meio de sensores, motores e outros dispositivos. Sua interface simplificada facilita o aprendizado e o desenvolvimento, sendo aplicado em contextos que vão desde a educação até a automação e prototipagem de produtos comerciais. (MCROBERTS, 2011; EVANS, NOBLE e HOCHENBAUM, 2013).

3.4. Biblioteca

A biblioteca para Arduino desenvolvida para integrar sensores ultrassônicos com lógica paraconsistente foi estruturada em torno da classe *ultra_s*, seguindo os princípios da POO. Essa abordagem permitiu o encapsulamento das funcionalidades e proporciona uma integração em projetos baseados na plataforma Arduino. O construtor da classe, *ultra_s*(int trig, int echo), é responsável por inicializar os pinos do sensor ultrassônico, configurando o pino de disparo (*_trig*) como saída e o pino de eco (*_echo*) como entrada, preparando assim o hardware para as operações de medição. Foram implementadas três funções principais nesta classe para gerenciar as medições e a análise lógica:

```
class ultra_s
{
public:
   ultra_s(int trig, int echo);
   float measureDistance(char opt);
   // Função para lidar com contradições ou incertezas
   float paraconsistentMeasureDistance(float distance1, float distance2);
private:
```

```
    int _trig;
    int _echo;
    // Função para verificar a consistência das medições
    bool isConsistent(float distance1, float distance2, float tolerance = 1.0);
};
```

A segunda etapa consistiu na implementação do construtor dentro da biblioteca, um passo para definir sua estrutura e funcionalidade. Essa implementação possibilita a correta inicialização da biblioteca, permitindo sua utilização em dois estados definidos e estáveis, assegurando o funcionamento do sistema. Essa função é destacada a seguir:

```
ultra_s::ultra_s(int trig, int echo) : _trig(trig), _echo(echo)
{
    pinMode(_trig, OUTPUT);
    pinMode(_echo, INPUT);
}
```

A terceira etapa consistiu na formulação das funções responsáveis pela medição da distância e pelo processamento dos parâmetros obtidos pelo sensor. Para isso, a função *MeasureDistance* foi implementada, iniciando-se com a configuração do pino trigger (*trig*) para emitir um pulso de ultrassom. Após um breve intervalo, o tempo de resposta do sinal refletido é capturado pelo pino echo (*echo*) por meio da função *pulseIn*, armazenando a duração do pulso, apresentando a seguir.

```
// Implementação da função de medição de distância
float ultra_s::measureDistance(char opt)
{
    digitalWrite(_trig, LOW);
    delayMicroseconds(2);
    digitalWrite(_trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(_trig, LOW);
    long duration = pulseIn(_echo, HIGH);
    float distance = duration * 0.034 / 2;
    if (opt == 'c')
// Se a opção for 'c', retorna a distância em centímetros
    {
        return distance;
    }
    else if (opt == 'i')
// Se a opção for 'i', retorna a distância em polegadas
    {
```

```
        return distance / 2.54;
    }
    else
    {
        return -1; // Retorna -1 se a opção for inválida
    }
}
```

Nesta etapa, foi desenvolvida a função *MeasureDistance*, que aprimora a precisão das medições ao validar a consistência dos valores obtidos pelo sensor. A função recebe duas medições de distância e utiliza a função auxiliar *isConsistent* para verificar se a diferença entre elas está dentro de uma tolerância pré-definida.

Caso as medições sejam consistentes, a função retorna a média aritmética dos valores. No entanto, se houver uma contradição entre os valores (ou seja, se a diferença ultrapassar a tolerância permitida), a função retorna -1, indicando uma medição inconsistente. Esse mecanismo destaca a confiabilidade do sistema, minimizando erros e garantindo maior precisão na obtenção da distância.

```
// Implementação da função paraconsistente
float ultra_s::paraconsistentMeasureDistance(float distance1, float distance2)
{
    if (isConsistent(distance1, distance2))
    {
        return (distance1 + distance2) / 2;
// Retorna a média se as medições forem consistentes
    }
    else
    {
        return -1;
// Retorna -1 se houver uma contradição (medições inconsistentes)
    }
}
```

Com toda a elaboração, um resultado é estimado (grau de crença) ao valor esperado para gerar um resultado iminente ao de dois índices limiares *true and false*, onde se dá ao método um resultado proeminente da estrutura paraconsistente, finalizando assim a biblioteca que será implementada a um código de sensor ultrassônico com fatores de mediação a capacidade de longitude delimitada pelo sensor.

```
bool ultra_s::isConsistent(float distance1, float distance2, float tolerance)
{
    return abs(distance1 - distance2) <= tolerance;
```

4. CONCLUSÃO

Esta pesquisa demonstrou com sucesso a viabilidade e a funcionalidade da Lógica Paraconsistente Anotada de Anotação com Dois Valores (LPA2v) no contexto de sistemas embarcados, especificamente por meio do desenvolvimento de uma biblioteca Arduino para sensores ultrassônicos, destacando um potencial entre conceitos teóricos avançados da lógica não clássica e aplicações práticas de engenharia, superando as limitações inerentes à lógica clássica no tratamento de informações inconsistentes e paradoxais [1, 1]. As principais contribuições incluem:

- **Desenvolvimento de uma Biblioteca Arduino:** A criação de uma biblioteca para Arduino que integra sensores ultrassônicos com LPA2v, oferecendo uma alternativa para aumentar a precisão e confiabilidade das leituras sensoriais em ambientes ruidosos e com dados contraditórios;
- **Elaboração do Algoritmo Para-Analisador:** A descrição e a implementação do algoritmo, que traduz os conceitos teóricos da LPA2v em um modelo computacional prático. Este algoritmo permite a manipulação e o raciocínio com conhecimento incerto, classificando as situações em 12 estados lógicos distintos (extremos e não-extremos) com base nos graus de crença, descrença, certeza e incerteza;
- **Projeto do Controlador Lógico Para-Control (Software e Hardware):** A materialização do Para-Analisador em um software funcional (em linguagem C) e em um projeto de Circuito Integrado (CI). O Para-Control oferece flexibilidade para ajustes externos, aplicação de operadores lógicos (NOT, OR, AND) e saídas analógicas e discretas, tornando-o uma ferramenta versátil para diversas aplicações.
- **Demonstração da Aplicabilidade:** A validação da LPA2v em áreas críticas como robótica autônoma, sistemas especialistas em Inteligência Artificial e automação industrial, mostrando sua capacidade de lidar com incertezas e inconsistências de forma não trivial.
- **Concepção de Sistemas Híbridos ("Para-Fuzzy"):** A proposição de um controlador híbrido que combina a LPA2v com a Lógica Fuzzy, permitindo o tratamento simultâneo de contradições e imprecisões. Este avanço representa um passo significativo para a criação de sistemas mais robustos e confiáveis em ambientes complexos.

Este projeto, por dar acesso a capacidades lógicas muito mais avançadas do que é atualmente possível em uma plataforma popular como o Arduino, não só contribui para o avanço do estado-da-arte científico de campos em computação embarcada e lógica não clássica, mas também democratiza as ferramentas para o desenvolvimento da nova geração de sistemas inteligentes e adaptativos. A habilidade de lidar com inconsistências de maneira razoável torna o desafio uma oportunidade para a inovação e o avanço tecnológico.

AGRADECIMENTOS

Os autores agradecem a FAPEMA, pelo suporte financeiro a este projeto, ao IFMA através do Edital PRPGI nº 14/2024 – PIBITI Ensino Superior 2024/2025 – e, em especial, ao Campus São João dos Patos, por toda infraestrutura oferecida.

REFERÊNCIAS

ABE, J. M.; AKAMA, S.; NAKAMATSU, K. ***Introduction to Annotated Logics***. Springer International Publishing, v. 88, 2015.

COUTINHO JÚNIOR, A. L.; MONTEIRO, J. A.; LOPES, J. L.; et al. **Uma proposta experimental de eletricidade com o uso da placa de prototipagem Arduino para o ensino de física**. Research, Society and Development, v. 10, n. 2, p. e11110212302, 2021.

SILVA FILHO, J. I. ***Métodos de aplicações da lógica paraconsistente anotada de anotação com dois valores-LPA2v com construção de algoritmo e implementação de circuitos eletrônicos***. Universidade de São Paulo, 1999.

EVANS, M.; NOBLE, J.; HOCHENBAUM, J. ***Arduino em Ação***. São Paulo: Novatec, 2013.

MCROBERTS, M. ***Arduino Básico***. São Paulo: Novatec, 2011.

MARCONDES, R. C.; MIGUEL, L. A. P.; FRANKLIN, M. A.; PEREZ, G. (2017). Metodologia para trabalhos práticos e aplicados. São Paulo: Editora Mackenzie.

NASCIMENTO, S. S.; NAÄS, I. A.; ABE, J. M.; OLIVEIRA, C. C.; FORÇAN, L. R. **Instrumento de Avaliação de Competências Aplicando a Lógica Paraconsistente Anotada Evidencial** Eτ. *Research, Society and Development*, v. 10, n. 4, 2021.

OLIVEIRA, J. B.; GINO, J. V. S. R.; LIMA, C. J.; SILVA FILHO, J. I. **Do código à confiabilidade: Lógica Paraconsistente impulsionada por Python para a detecção de alarmes na rede de energia**. *GeSec: Revista de Gestão e Secretariado*, v. 14, n. 10, 2023.