UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO DE CIÊNCIAS, TECNOLOGIA E SAÚDE - CAMPUS ARARANGUÁ
CURSO DE GRADUAÇÃO EM ENGENHARIA DE COMPUTAÇÃO

Michele Rocha

SimpleBDI: Uma Proposta de Arquitetura para Agentes Inteligentes

Araranguá
2023

Michele Rocha

SimpleBDI: Uma Proposta de Arquitetura para Agentes Inteligentes

Trabalho de Conclusão de Curso do Curso de Graduação em Engenharia de Computação do Centro de Ciências, Tecnologia e Saúde - Campus Araranguá da Universidade Federal de Santa Catarina para a obtenção do título de Bacharel em Engenharia de Computação.
Orientador: Prof. Alison R. Panisson, Dr.

Araranguá
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Rocha, Michele
   SimpleBDI: Uma Proposta de Arquitetura para Agentes Inteligentes / Michele Rocha ; orientador, Alison R. Panisson, 2023.
   29 p.

   Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Araranguá, Graduação em Engenharia de Computação, Araranguá, 2023.

   Inclui referências.

   1. Engenharia de Computação. 2. Inteligência Artificial. 3. Sistemas Multiagentes. 4. Sistemas Embarcados. 5. Web 3.0. I. Panisson, Alison R. . II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Computação. III. Título.

Michele Rocha

SimpleBDI: Uma Proposta de Arquitetura para Agentes Inteligentes

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Bacharel em Engenharia de Computação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Computação.

Araranguá, 30 de Novembro de 2023.

Documento assinado digitalmente
Jim Lau
Data: 08/12/2023 11:57:38-0300
CPF: ***.464.702-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Jim Lau, Dr.
Coordenador do Curso

Banca Examinadora:

Documento assinado digitalmente
Alison Roberto Panisson
Data: 08/12/2023 11:50:26-0300
CPF: ***.887.420-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Alison R. Panisson, Dr.
Orientador

Documento assinado digitalmente
GIOVANI PARENTE FARIAS
Data: 08/12/2023 19:57:21-0300
CPF: ***.191.610-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Giovani Parente Farias, Dr.
Avaliador
Instituição FURG

Documento assinado digitalmente
Roberto Vito Rodrigues Filho
Data: 08/12/2023 12:21:29-0300
CPF: ***.470.901-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Roberto Vito Rodrigues Filho, Dr.
Avaliador
Instituição UFSC

SimpleBDI: Uma Proposta de Arquitetura para Agentes Inteligentes

Michele Rocha* Alison R. Panisson†

2023, November

Resumo

Este trabalho implementa uma simplificação da arquitetura BDI (*Belief-Desire-Intention*) que é utilizada para a implementação de agentes inteligentes. A arquitetura proposta pode ser facilmente implementada em qualquer linguagem e foi intitulada como SimpleBDI. Para demonstrar sua versatilidade, explorou-se sua aplicação tanto em ambientes com recursos computacionais limitados (por exemplo microcontroladores) quanto em cenários distribuídos (por exemplo nós de uma rede blockchain). Para avaliar a arquitetura proposta, implementamos 3 estudos de caso. O primeiro estudo de caso envolve hardware IoT, que é aprimorado pela arquitetura SimpleBDI, conferindo inteligência e adaptabilidade ao dispositivo. O segundo estudo de caso tem como aplicação a Web 3.0, expandindo uma biblioteca de código aberto, que possibilita implementar agentes como nós de rede, mas carece de aspectos de raciocínio, o que é solucionado com a integração da arquitetura SimpleBDI. O terceiro estudo de caso apresenta uma integração dos dois estudos de caso anteriores, conferindo um sistema multiagentes, integrando agentes na Web 3.0 e agentes embarcados em dispositivos IoT. O estudo resultante se alinha com o cenário tecnológico multifuncional e distribuído que estamos vivenciando na área da inteligência artificial. A arquitetura proposta pode ser implementada em diferentes contextos, de aplicação, linguagens, etc. Possibilitando ainda a integração desses agentes.

Palavras-chaves: Inteligência Artificial, Web 3.0, Blockchain, Agentes Autônomos Inteligentes, Sistemas Multiagentes, Sistemas Embarcados.

*mr.michelerocha@gmail.com
†alison.panisson@ufsc.br

SimpleBDI: An Architecture Proposal for Intelligent Agents

Michele Rocha* Alison R. Panisson†

2023, November

Abstract

This work implements a simplification of the BDI (*Belief-Desire-Intention*) architecture used for implementing intelligent agents. The proposed architecture can be easily implemented in any programming language and has been named SimpleBDI. To demonstrate its versatility, its application has been explored in environments with limited computational resources (e.g., microcontrollers) as well as in distributed scenarios (e.g., nodes in a blockchain network). To evaluate the proposed architecture, three case studies have been implemented. The first case study involves IoT hardware, which is enhanced by the SimpleBDI architecture, providing intelligence and adaptability to the device. The second case study applies to Web 3.0, expanding an open-source code library that enables the implementation of agents as network nodes but lacks reasoning aspects, which are resolved by integrating the SimpleBDI architecture. The third case study presents an integration of the two previous case studies, creating a multi-agent system that combines agents in Web 3.0 with agents embedded in IoT devices. The resulting study enhances the autonomy and intelligence of the developed systems and aligns with the multifunctional and distributed technological scenario we are experiencing in the field of artificial intelligence. The proposed architecture can be implemented in different application contexts, programming languages, etc., allowing for the integration of these agents.

Keywords: Artificial Intelligence, Web 3.0, Blockchain, Intelligent Autonomous Agents, Multi-Agent Systems, Embedded Systems.

*mr.michelerocha@gmail.com
†alison.panisson@ufsc.br

1 Introdução

A evolução da tecnologia tem proporcionado avanços significativos no campo da Inteligência Artificial (IA), permitindo o desenvolvimento de sistemas inteligentes distribuídos sofisticados e autônomos, nos quais múltiplos componentes de IA podem trabalhar em conjunto para realizar tarefas complexas (AHMED; ABOOD; HAMDI, 2021). Os sistemas multiagentes representam uma forma de implementar esses sistemas distribuídos sofisticados e autônomos, Nesse contexto, os agentes são entidades autônomas capazes de tomar decisões de forma independente (OSSOWSKI; MENEZES, 2006; WOOLDRIDGE; JENNINGS, 1995). Essa abordagem de sistemas inteligentes composto de muitos agentes permite a resolução de problemas complexos que exigem habilidades distribuídas, como por exemplo, controle de tráfego aéreo (TANG; LIU; PAN, 2022), a otimização de processos industriais (GURURAJAPATHY; MOKHLIS; ILLIAS, 2017), ou ainda a coordenação de robôs em uma linha de produção (WAN et al., 2020).

No contexto de sistemas multiagentes, a arquitetura BDI (*Belief-Desire-Intention*) (WOOLDRIDGE; JENNINGS, 1995), que modela agentes inteligentes em estados de crenças, desejos e intenções, é amplamente utilizada. Essa arquitetura é frequentemente empregada em conjunto com o PRS (*Procedural Reasoning System*) (LEE et al., 1994), uma estrutura para construir sistemas de raciocínio em tempo real, visando a otimização das tomadas de decisão pelos agentes. Um exemplo notável que ilustra a aplicação prática dessas teorias é o framework Jason (BORDINI; HÜBNER; WOOLDRIDGE, 2007). No entanto, é crucial abordar que, apesar dos avanços proporcionados por essas arquiteturas e frameworks, existem desafios significativos quando se trata de integrá-los em sistemas diversos, conforme evidenciado em estudos anteriores (SANTOS et al., 2022; JESUS et al., 2023), que mencionam a complexidade associada à implementação dessas tecnologias em hardware, bem como a exigência de lidar com a configuração de múltiplos Ambientes de Desenvolvimento Integrados (IDEs). Essa integração é particularmente complexa quando tentamos unir essas tecnologias com técnicas de Inteligência Artificial (IA) bem estabelecidas, que são frequentemente desenvolvidas em linguagens de programação modernas e populares, como Python, como destacado por Raschka, Patterson e Nolet (2020), que aponta o python como uma das tecnologias favoritas para a implementação de IA. Essa discrepância entre as linguagens e paradigmas de programação utilizados nas arquiteturas de sistemas multiagentes e nas implementações de IA contemporâneas pode gerar obstáculos na interoperabilidade e na eficiência da implementação, exigindo uma atenção especial dos desenvolvedores e pesquisadores para superar tais barreiras.

Baseado na arquitetura BDI, desenvolvemos uma arquitetura simples que pode ser implementada em múltiplas linguagens de programação que incorpora os princípios fundamentais para a criação de agentes inteligentes. Para avaliarmos a arquitetura proposta, foram implementados 3 estudos de caso. Os estudos de caso foram divididos em três partes distintas. No primeiro estudo de caso, será abordada a capacidade de aprimorar a inteligência de dispositivos com recursos limitados, onde um agente é implementado embarcado a um hardware, e o dispositivo passa a ser tratando como uma entidade inteligente no sistema. No segundo estudo de caso, utilizou-se uma biblioteca *open-sorce* de agentes inteligentes aplicada à Web 3.0. Essa biblioteca é umas das primeiras iniciativas de integração de agentes inteligentes à Web 3.0. Ela é fundamentada na tecnologia blockchain, que trata as informações como registros digitais em blocos de transações. O desenvolvimento proposto visa explorar as vantagens da integração entre Inteligência Artificial e blockchain (CALVARESI et al., 2018). Como a biblioteca *open-source* selecionada carece de aspectos

sofisticados de raciocínio e planejamento, incorporamos o raciocínio da arquitetura BDI (BORDINI; HÜBNER; WOOLDRIDGE, 2007) proposto nesse artigo, mantendo os aspectos já existentes de conceitualização baseados na Web 3.0. Por último, no terceiro estudo de caso, foi feita uma integração multiagente usando tanto os agentes embarcado em hardware com recursos limitado, como o agente desenvolvido junto à tecnologia baseada na Web 3.0 e blockchain. Nesse último estudo, o agente de software fundamentado na Web 3.0 funciona como dispositivo central que pode ter maior capacidade de processamento e memória, considerando que o mesmo pode ser executado em servidores ou computadores com elevado poder de processamento. Esses agentes podem então incorporar técnicas sofisticadas de inteligencia artificial, como visão computacional. No estudo de caso proposto, esses agentes criam planos de ação para o dispositivo embarcado, e então esse planos são encaminhados para o agente embarcado no hardware, os quais podem ser utilizados para atingir objetivos do sistema multiagente.

A arquitetura proposta nesse trabalho visa contribuir para o avanço do campo de sistemas multiagentes e inteligência artificial como um todo. A integração da arquitetura BDI em conjunto com o PRS oferece aos agentes inteligentes a capacidade, não apenas de realizar ações no ambiente e automatizar tarefas, mas também de adaptar seu comportamento com base nas suas percepções no ambiente e seus planos que podem ser dinamicamente atualizados. Ao utilizarmos desses componentes, trazemos essas características tanto para agentes embarcados, como também para a Web 3.0.

2 Fundamentação Teórica

Nesta seção, vamos introduzir os conceitos fundamentais relacionados às tecnologias empregadas na implementação deste trabalho. Abordaremos os princípios subjacentes às ferramentas e abordagens adotadas, fornecendo uma base conceitual dos temas que abrangem o trabalho. Além disso, discutiremos o sistema de raciocínio baseado em lógica procedural, contextualizando tudo dentro da nova geração da Internet, a Web 3.0.

2.1 Sistemas Multiagentes e a Arquitetura BDI

De acordo com Wooldridge e Jennings (1995), um agente inteligente é um sistema computacional capaz de perceber seu ambiente, tomar decisões e agir de forma autônoma para atingir seus objetivos ou realizar tarefas específicas. Um agente inteligente é projetado para interagir com seu ambiente, utilizando sensores para perceber informações relevantes e atuadores para executar ações (WOOLDRIDGE; JENNINGS, 1995). Existem diferentes tipos de agentes inteligentes, e sua utilização depende do domínio de aplicação. Por exemplo, um agente inteligente pode ser um programa de computador que joga xadrez e toma decisões com base em suas percepções do tabuleiro utilizando uma estratégia pré-definida. Outro exemplo seria um agente robótico autônomo que navega em um ambiente desconhecido, tomando decisões sobre quais ações executar para evitar obstáculos e alcançar um destino específico (RUSSELL, 2010). A inteligência de um agente está relacionada à sua capacidade de perceber e interpretar corretamente as informações do ambiente, tomar decisões racionais com base nessas informações e agir de maneira eficiente para alcançar seus objetivos. A inteligência também pode envolver aprendizado e adaptação, permitindo que o agente melhore seu desempenho ao longo do tempo (RUSSELL, 2010).

Sistemas multiagentes são sistemas computacionais compostos por múltiplos agentes inteligentes que interagem e colaboram entre si para alcançar objetivos comuns ou resolver

problemas complexos. Cada agente em um sistema multiagente é capaz de perceber seu ambiente, tomar decisões autônomas e agir de forma independente. Agentes em um sistema multiagente podem ter diferentes capacidades, conhecimentos, habilidades e objetivos. Eles podem ser projetados para interagir uns com os outros, trocando informações, coordenando suas ações, negociando recursos ou colaborando para alcançar resultados melhores do que se estivessem agindo isoladamente (WOOLDRIDGE; JENNINGS, 1995).

A arquitetura BDI (*Belief-Desire-Intention*, em português, Crença-Desejo-Intenção) é uma estrutura conceitual para modelar e implementar agentes inteligentes. Essa arquitetura foi proposta por Michael Georgeff e Amy L. Lansky em 1997, futuramente modificado por importantes desenvolvimentos realizados por Michael Bratman ao longo dos anos, e tem sido amplamente utilizada no campo de sistemas multiagentes (GEORGEFF; LANSKY, 1987; BRATMAN, 1987; RAO; GEORGEFF, 1997). A arquitetura BDI é baseada em uma abordagem psicológica para modelar o comportamento humano, buscando capturar os aspectos cognitivos e motivacionais dos agentes. Ela é composta por três componentes principais:

- Crenças (*Belief*): Representa o conhecimento do agente sobre o mundo. As crenças são as informações percebidas pelo agente a partir de seu ambiente, como observações, dados sensoriais, regras e fatos. Essas crenças são atualizadas conforme o agente interage com o ambiente.
- Desejos (*Desire*): Reflete os objetivos ou metas que o agente deseja alcançar. Os desejos são baseados nas crenças do agente e podem variar de acordo com o estado de suas crenças, descrevendo um contexto específico. Os desejos podem ser explícitos (declarados pelo agente) ou implícitos (inferidos a partir das crenças).
- Intenções (*Intention*): Representa os planos de ação que o agente seleciona para realizar seus desejos (objetivos). As intenções são formadas com base na avaliação das crenças e desejos atuais do agente, considerando também suas capacidades e restrições. Uma intenção é o compromisso do agente em agir de acordo com um plano específico.

A arquitetura BDI é frequentemente utilizada em sistemas multiagentes que requerem um alto grau de raciocínio deliberativo, planejamento e coordenação de ações em ambientes dinâmicos (PALANCA et al., 2023; STRINGER et al., 2020). Ela tem sido aplicada em diversas áreas, incluindo robótica, filtragem da água, e até mesmo em gerenciamento de trafego (RÜB; DUNIN-KEPLICZ, 2020; MENDOZA et al., 2021). Uma das vantagens da arquitetura BDI é a sua modularidade, o que permite uma fácil extensão e adaptação para diferentes tarefas e contextos. Além disso, ela é capaz de lidar com incerteza e incompletude das informações, permitindo que o agente tome decisões mesmo quando não possui informações completas sobre o ambiente. A arquitetura oferece uma solução promissora e adequada para a utilização em sistemas robóticos, permitindo a modelagem do comportamento dos agentes de forma modular, adaptativa e eficiente. Ao adotar essa arquitetura, os sistemas robóticos podem beneficiar-se de um maior grau de inteligência e capacidade de resposta, tornando-se mais eficazes em uma ampla variedade de tarefas e ambientes desafiadores (BRATMAN, 1987).

2.2 Arquitetura BDI e Sistema de Raciocínio PRS

A arquitetura BDI define componentes de um agente, baseados em componentes humanos, originários da psicologia, como crenças desejos e intenções. Utilizando esses componentes, existem algumas abordagens que propõem sistemas de raciocínios para tecnologias multiagentes, um dos mais conhecidos é o PRS (*Procedural Reasoning System*)(LEE et al., 1994). PRS utiliza um conjunto de regras e procedimentos para tomar decisões e resolver problemas de forma automatizada. Ele é baseado na ideia de decompor um problema complexo em etapas menores e definir ações específicas para cada etapa. Ele prevê ainda uma analise sobre a situação atual, avaliando as informações disponíveis para determinar a melhor sequência de ações a serem executadas para atingir um objetivo ou resolver um problema (LEE et al., 1994). O PRS oferece uma abordagem sistemática para resolver problemas, permitindo a adaptação e a flexibilidade em diferentes contextos, utilizando os componentes chave da arquitetura BDI (GEORGEFF; INGRAND, 1989).

O PRS opera através de um fluxo contínuo de informações e decisões, como mostrado na Figura 1, o que facilita o funcionamento dos agentes em um ambiente. Inicialmente, o sistema absorve entradas de sensores, representando informações do ambiente onde o agente atua. Essa percepção é crucial para a atualização das crenças do agente sobre o estado atual do mundo. Posteriormente, essas informações são transformadas em 'Crenças', retratando o entendimento do agente sobre o mundo ao seu redor. Com base nessas crenças e em outros aspectos internos, o agente configura seus 'Desejos', que simbolizam os objetivos ou estados desejados. Contudo, a escolha de perseguir ou não um desejo específico é influenciada por vários fatores, como capacidade do agente, prioridades e planos disponíveis. Essa estrutura de planos é o que diferencia o BDI e o PRS, e são armazenados em uma 'Biblioteca de Planos', que consiste em ações ou procedimentos designados para alcançar um desejo específico. A seleção do plano adequado é guiada pelas 'Intenções' do agente, que representam os desejos escolhidos para serem alcançados. Dentro deste sistema, o 'Interpretador' serve como o núcleo operacional, avaliando crenças, desejos e intenções e, com isso, selecionando a ação mais apropriada a ser executada. Finalmente, após esta avaliação, o agente desencadeia uma ação, visando alcançar ou concretizar suas intenções, marcando sua interação direta com o mundo exterior. Esta abordagem cíclica do PRS facilita a adaptabilidade e eficiência dos agentes em diversos ambientes.

O PRS possui algumas características como planos pré-compilados, que economizam tempo ao usar estratégias predefinidas; a possibilidade de incluir objetivos nos planos, tornando-os mais flexíveis; a representação das crenças, que define as verdades do agente; e a pilha de intenções, que mantém metas a serem alcançadas e direciona o agente a explorar planos alinhados com suas crenças para atingir essas metas. No processo de funcionamento do PRS, o agente começa selecionando planos que correspondem à meta no topo da pilha de intenções e que se alinham com suas crenças atuais. Em seguida, ele entra em uma fase de deliberação, escolhendo entre esses planos usando critérios como planos de meta-nível ou valores numéricos (utilidades). O plano escolhido é então executado, podendo implicar a adição de novos objetivos à pilha de intenções. Se o plano não for bem-sucedido, o agente é capaz de replanejar e selecionar um novo plano para atingir a mesma meta. Esse processo garante a adaptabilidade do agente na busca por seus objetivos (GEORGEFF; LANSKY, 1987; BORDINI; HÜBNER; WOOLDRIDGE, 2007).

Fonte: (BORDINI; HÜBNER; WOOLDRIDGE, 2007)

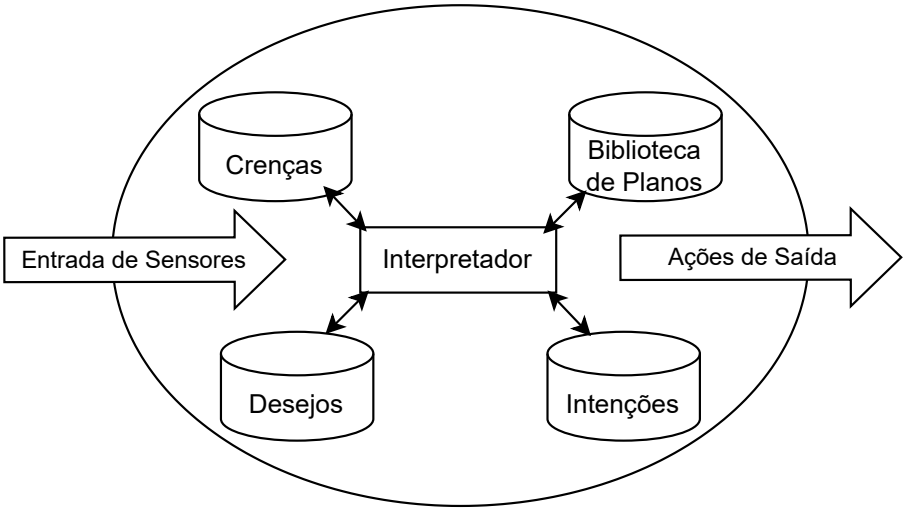

Figura 1 – Procedural Reasoning System.

2.3 Web 3.0 e Blockchain

A Web 3.0 é uma visão da evolução da internet que busca promover uma maior descentralização e autonomia para os usuários, criando um ambiente online mais seguro, transparente e eficiente (SHIVALINGAIAH; NAIK, 2008; ALABDULWAHHAB, 2018). Essa concepção se baseia em princípios fundamentais que buscam superar as limitações da Web 2.0 e alavancar a tecnologia blockchain (YAMAGUCHI; YAMAGUCHI, 2016). A tecnologia blockchain é uma forma de registro digital descentralizado e distribuído, que permite armazenar e verificar transações de forma segura, transparente e imutável (CAO, 2022). Ela foi originalmente desenvolvida para sustentar o funcionamento das criptomoedas, como o Bitcoin (NAKAMOTO, 2008), mas hoje também é aplicada em diversos outros setores, tornando-se uma ferramenta central para a nova visão da internet.

Um dos princípios descentralizados da Web 3.0 é a substituição de intermediários centralizados por protocolos peer-to-peer (YAMAGUCHI; YAMAGUCHI, 2016). Na Web 2.0, muitas atividades online são intermediadas por empresas e plataformas centralizadas, o que gera problemas como falta de privacidade, dependência de terceiros e riscos de censura (SHIVALINGAIAH; NAIK, 2008). Com a Web 3.0, a tecnologia blockchain possibilita a criação de protocolos descentralizados que permitem a interação direta entre os usuários, eliminando a necessidade de intermediários e colocando o controle dos dados e das transações nas mãos dos usuários.

Em sua essência, uma blockchain é um registro público composto por blocos de informações encadeados de forma cronológica como mostra na Figura 2. Cada bloco contém um conjunto de transações ou dados, além de um identificador único chamado "*hash*". O *hash* é uma sequência alfanumérica gerada por meio de um algoritmo criptográfico, e funciona como uma impressão digital do bloco, permitindo sua identificação única (WOLFSKEHL, 2018). O que torna a tecnologia blockchain especial é o fato de que ela opera em um sistema descentralizado, no qual as transações são validadas e registradas por uma rede de participantes, chamados de "nós" ou "*nodes*". Esses nós trabalham em conjunto para confirmar a autenticidade e a integridade das transações, utilizando mecanismos de consenso, como o algoritmo de prova de trabalho (PoW) ou prova de participação (PoS) (GERVAIS et al., 2016).

Fonte: O autor.

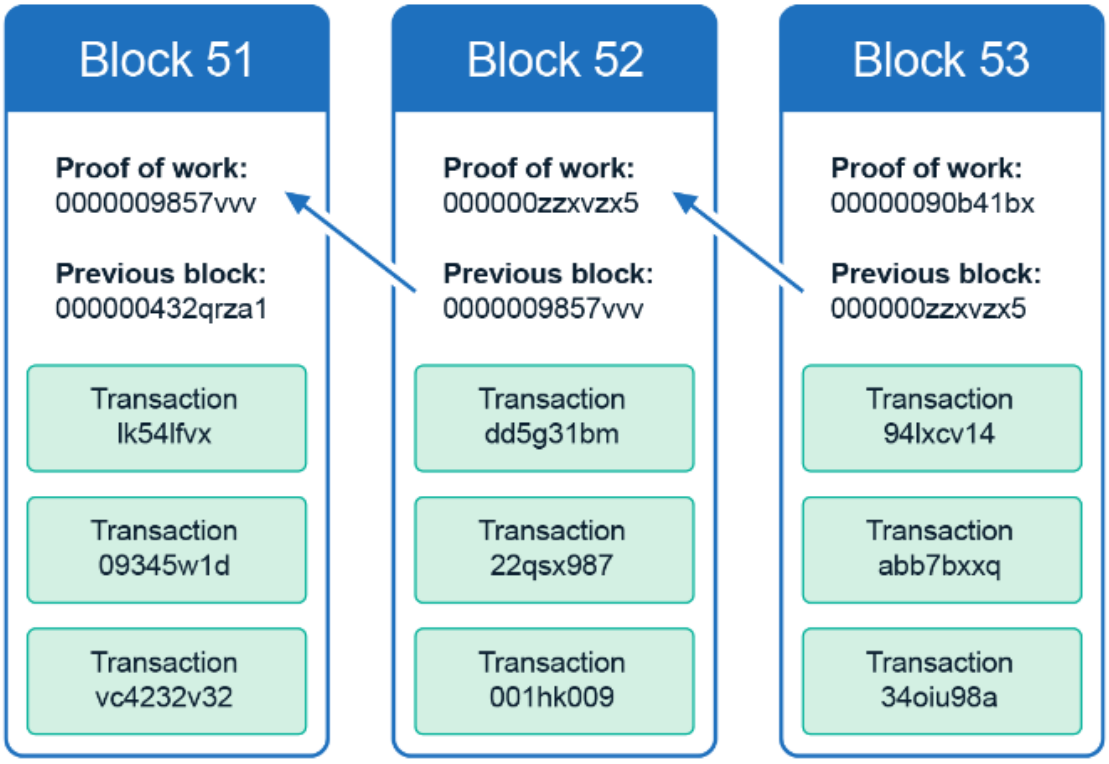

Figura 2 – Exemplo de Blockchain.

Uma vez que uma transação é validada por consenso e adicionada a um bloco, esse bloco é permanentemente adicionado à cadeia existente, formando um histórico sequencial e imutável de todas as transações que já ocorreram na rede. Isso significa que, uma vez que uma transação é registrada em uma blockchain, é extremamente difícil alterá-la ou apagá-la, o que garante a segurança e a integridade dos dados armazenados (GERVAIS et al., 2016; WOLFSKEHL, 2018).

Além da segurança e da imutabilidade, a transparência é outro aspecto chave da tecnologia blockchain. Como o registro é público e distribuído, qualquer pessoa pode acessar e verificar as transações realizadas. No entanto, embora as informações sejam públicas, a identidade dos envolvidos nas transações geralmente é protegida por meio de chaves criptográficas, garantindo a privacidade dos usuários (GERVAIS et al., 2016).

Por fim, a Web 3.0 promove a participação ativa e o empoderamento dos usuários. Com a descentralização utilizando da blockchain para sua segurança e integridade, as pessoas têm maior controle sobre seus dados e podem decidir como e quando compartilhá-los. Além disso, a Web 3.0 possibilita a criação de ecossistemas econômicos baseados em tokens, que permitem a colaboração direta entre os usuários e a valorização dos seus recursos digitais. O princípio-chave da Web 3.0 é a transparência e a auditabilidade. Através da tecnologia blockchain, é possível criar registros distribuídos e imutáveis de todas as transações e atividades na rede. Isso significa que qualquer pessoa pode verificar e auditar as transações de forma transparente, o que aumenta a confiança e a segurança nas interações online (SHIVALINGAIAH; NAIK, 2008).

A Web 3.0 está intimamente relacionada com a tecnologia blockchain. A descentra-

lização, a transparência, a segurança e o empoderamento dos usuários são elementos-chave proporcionados pelo uso da blockchain na Web 3.0. Essa combinação visa transformar a internet em um ambiente mais democrático, confiável e eficiente, onde os usuários têm maior controle sobre suas interações e seus dados, além de permitir o desenvolvimento de novos modelos de negócios e a criação de redes econômicas descentralizadas.

3 SimpleBDI: Uma proposta de arquitetura simples e adaptável para agentes inteligentes

Nesta seção, abordaremos a conceitualização do estudo para a implementação de agentes inteligentes. Discutiremos como essa implementação promove a capacidade de adaptação e inteligência, utilizando raciocínio lógico e procedural na tomada de decisão, baseado no BDI e PRS.

3.1 Modelagem Conceitual da Arquitetura SimpleBDI

A implementação dos agentes, conforme ilustrado na Figura 3, envolve a adoção de um contexto composto por crenças, desejos e intenções, conceitos já discutidos na Seção 2.1. Além disso, as **ações** realizadas pelos agentes (*action*) desempenham um papel importante na atualização das crenças (*update beliefs*). Essas ações geram informações (update perceptions) que são incorporadas às crenças dos agentes. As **crenças** desempenham um papel fundamental, representando o conhecimento que os agentes possuem sobre o mundo. Essas crenças são adquiridas por meio das percepções do agente (*update perceptions*) conforme ele age com o ambiente (*environment*), ou quando recebidas a partir de processos de comunicação. Esse processo forma o ciclo de raciocínio do agente, onde o agente faz uma nova ação, que pode gerar uma nova crença (em reação a nova ação) que criará uma nova percepção, criando assim um loop contínuo de ação percepção.

Em aspectos de implementação, crenças serão armazenadas em forma de dicionários JSON, ilustrado na Figura 3 pelo bloco "*belief_base*". Essa abordagem de ter um dicionários de crenças modular, ou seja, o bloco "*belief_base*", permite que os agentes atualizem seu

Fonte: o Autor.

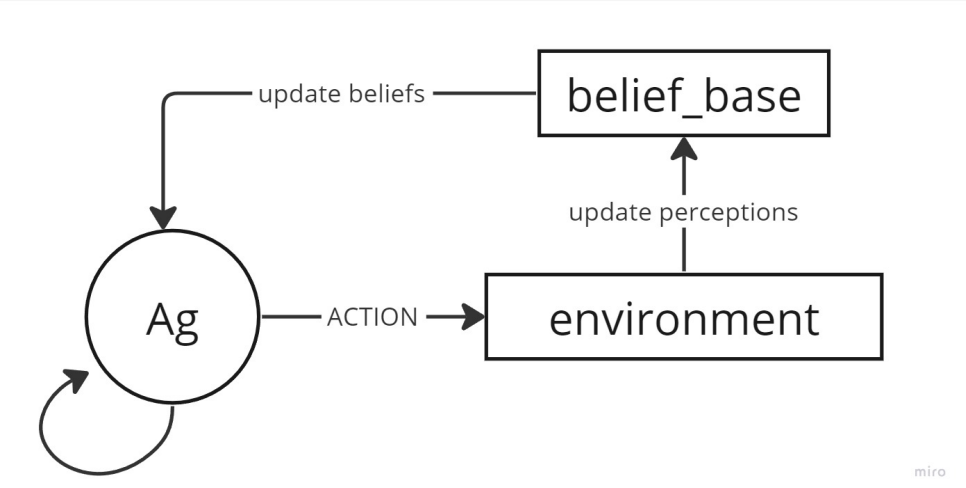

Figura 3 – Diagrama Conceitual SimpleBDI.

entendimento sobre o estado atual do ambiente, e esse estado de crenças seja persistente em termos de armazenamento enquanto o agente estiver ligado e operando no sistema. Essa atualização ocorre em um ciclo contínuo de raciocínio, em que os agentes têm acesso aos dados mentais por meio dessas crenças. Isso possibilita que os agentes ajustem suas ações e tomem decisões informadas com base nas informações atualizadas.

Os **desejos** são uma expressão dos objetivos e planos que os agentes têm em relação ao ambiente. Eles representam as preferências e percepções dos agentes em relação ao que eles desejam alcançar ou realizar. Os desejos dos agentes podem ser influenciados pelas suas crenças, bem como por estímulos externos ou internos. Observando a Figura 3, desejo não podem ser observados de forma explicita, pois na figura o desejo implicitamente está associado ao próprio agente sendo o 'Ag'. O desejo trata da vontade do agente de querer alcançar um estado no ambiente, estando contido internamente no raciocínio do agente, de acordo com a fundamentação teórica descrita na Seção 3.2, representando parte do raciocínio interno do agente.

Com base nas suas crenças e desejos, os agentes formam **intenções**. As intenções representam a determinação dos agentes em realizar ações específicas para atingir seus objetivos, representado na Figura 3 por "ACTION". Vale lembrar que as intenções dos agentes não são de fato suas ações no ambiente, mas sim seu planejamento de ações a serem executadas no ambiente. As intenções são a base para o planejamento das sequências de ações que agentes executarão no ambiente. Agindo no ambiente, as informações obtidas pelo agente a partir das suas **percepções** são interpretadas e assimiladas em seu conjunto de crenças, influenciando, por sua vez, suas futuras ações e decisões.

O raciocínio dos agentes ocorre de forma iterativa, em um ciclo contínuo de ações, e percepções. Os agentes executam ações com base em suas intenções e, posteriormente, atualizam suas crenças com base nas consequências dessas ações, criando um ciclo de raciocínio continuo. Essa retroalimentação das crenças permite que os agentes ajustem seu comportamento e tomem decisões adaptativas de acordo com as mudanças no ambiente.

3.2 Formalização Proposta para a Arquitetura

A seguir, apresenta-se a formalização da arquitetura simplificada proposta para o raciocínio do agente que será posteriormente utilizada na implementação de agentes nesse trabalho. A formalização é baseada na arquitetura BDI e no PRS, ambos descritos nas Seções 2.1 e 2.2. A formalização dá ênfaze para os elementos e as relações que definem a estrutura e o funcionamento da proposta de arquitetura, permitindo uma representação formal e precisa do processo de tomada de decisão baseado em crenças, desejos e intenções dos agentes.

Um agente é uma tupla, $\langle ag, B, D, I, A, ps \rangle$, onde ag é o nome do agente, B é a sua base de crenças, D são os desejos do agente, I é a intenção em execução pelo agente, A é o conjunto de ações que o agente é capaz de executar, e ps é a biblioteca de planos, onde cada plano possui o formato de $\langle g, ctx, [a_1, \ldots, a_n] \rangle$ com g o objetivo (desejo) alcançado pelo plano, ctx o contexto (pré condições) para execução do plano, e $[a_1, \ldots, a_n]$ a sequência de ações (ou subobjetivos) que implementam o plano. Abaixo é apresentado o pseudo código do ciclo de raciocínio principal do agente implementado

Como pode ser observado no pseudocódigo, o estado do agente é definido por conjuntos que correspondem a sua biblioteca de planos, crenças, desejos iniciais e um conjunto vazio de intenções, passando então para um ciclo contínuo de execução da pilha

```
ps ← {p_1, ..., p_n} ;                    /* inicializa a biblioteca de planos */
B ← {b_1, ..., b_n} ;                     /* inicializa a base de crenças */
D ← {d_1, ..., d_n} ;                     /* inicializa a os desejos */
I ← ∅ ;                                   /* inicializa a intenções vazia */
while true do
    B ← update(B, perception()) ;         /* atualiza as crenças */
    if I ≠ ∅ then
        α ← next(I) ;                     /* recupera próxima ação da intenção */
        executa(α) ;                      /* executa a próxima ação da intenção */
        I ← I \ α ;                       /* remove a ação da intenção */
    else
        γ ← next(D) ;                     /* recupera próximo desejo */
        I ← I ∪ plan(γ, ps, B) ;          /* recupera a sequência de ações para γ */
    end
end
```

de ações que compõe sua intenção atual. Caso ele não possua uma intenção em execução, como acontece na sua inicialização, o agente seleciona o próximo desejo/objetivo a ser alcançado, selecionando um plano aplicável para alcança-lo, transformando-o sua intenção atual. É importante enfatizar que as crenças podem ser constantemente atualizadas pela percepção do agente, e seus planos e desejos também podem ser atualizados por fatores externos, como percepções que vão disparar desejos específicos (de carácter reativo) e/ou mensagens recebidas, delegando novos objetivos ao agente.

Compreendemos que os agentes têm a habilidade de desenvolver **crenças** a respeito do ambiente e do estado do sistema à medida que interagem com o ambiente. Essas interações são registradas para que posteriormente possam ser acessadas. O sistema de crenças foi pensado para ser baseado em dicionários. Tais crenças são armazenadas e recebidas por meio de arquivos JSON, e estão representadas na formalização como $B \leftarrow \{b_1, \ldots, b_n\}$, que define as crenças do agente. A base de crenças do agente pode ser enriquecida pela operação $B \leftarrow update(B, perception())$. Estas crenças são continuamente atualizadas com base nas interações do agente com o ambiente. Em tempo real, são avaliadas pelo agente durante o processo de planejamento, utilizando crenças como pré condições para seleção de planos da biblioteca de planos, permitindo assim que o agente tome ações apropriadas em resposta às mudanças no ambiente.

Os **desejos** funcionam como vontade dos agentes, representando o que eles pretendem alcançar. Essas vontades podem ser estabelecidos pelos próprios agentes ou surgirem de interações e negociações com outros agentes. As ações desses agentes são orientadas por esses desejos, independentemente de serem ou não realizadas no ambiente, e as decisões são tomadas com base em como suas ações podem contribuir para a realização de seus objetivos. Conforme apresentado na Figura 3, discutida na seção 3.1, os agentes realizam ações no ambiente e, em seguida, as percepções resultantes dessas ações são usadas para atualizar as suas base de crenças. Essas ações são parte de planos escolhidos para alcançar os desejos do agente, normalmente representando interação com o ambiente. Independentemente de terem êxito ou não, esse anseio de alcançar estados específicos do ambiente são definidos como "desejos". Os desejos do agente são definidos pelo conjunto D, representado por uma sequência $(d_1, d_2, ..., d_n)$, onde d_i é um desejo específico. Para incorporar um novo desejo à lista, adicionamos um novo elemento ao final da sequência D, ou seja, d_{n+1}. E para acessar

um desejo, removemos o primeiro elemento d_1 de D e o retornamos (ou seja, ele respeita um implementação de fila), sendo assim $\gamma \leftarrow next(D)$.

As **intenções** de um agente são elementos fundamentais para direcionar suas ações e auxiliá-lo na realização de seus objetivos. As intenções correspondem a raciocínio de planos de ação que um agente elabora a partir de seus desejos atualizados e de suas crenças sobre o ambiente. Formalmente, estas são definidas como I e representadas por uma sequência $(\alpha_1, \alpha_2, ..., \alpha_n)$, onde α_n é uma ação ou meta/desejo. As intenções diferem dos desejos na medida em que um desejo representa a vontade de realizar algo (atingir um objetivo), enquanto a intenção é o planejamento para concretizar essa vontade. A intenção dá uma direção de ações à serem executadas e esse planejamento é descrito como $I \leftarrow I \cup plan(\gamma, ps, B)$. As intenções são formadas por sequências de ações e subobjetivos, definidos pelo plano escolhido, filtrado pelas crenças do agente. Após a escolha de um plano, o agente passa a executar as primeiras ações do plano no ambiente.

As **ações** dos agentes são conduzidas pelas suas intenções e desejos. Enquanto os desejos expressam a vontade dos agentes de realizar uma ação, independentemente do resultado ser sucesso ou fracasso, as intenções consideram estratégias para a concretização bem-sucedida desses desejos, operando como um elo entre o desejo e a ação. As ações, por sua vez, resultam em sucesso ou fracasso dentro do ambiente e podem ser categorizadas em duas: ações diretas no ambiente e ações baseadas em planos. Formalmente, ambas são representadas pelo conjunto A. Para adicionar uma sequência de ações pertencentes à A para a lista de ações correspondente a intenção I do agente, realizamos a operação de concatenação de sequências $I = I \cup A'$, com A' a lista de ações a serem adicionadas. Ao executar uma intenção no ambiente, removemos ela do topo da pilha representada por I. Se α for uma *ação*, então a executamos $executa(\alpha)$. Se for um desejo/objetivo, procura-se um plano correspondente para atingi-lo, ou seja, $I \leftarrow I \cup plan(\alpha, ps, B)$, adicionando a sequência de ações e subobjetivos daquele plano à intenção atual, e então aguardamos o próximo ciclo de raciocínio do agente para desencadear a ação inicial de sua intenção atualizada. As ações diretas, por outro lado, como o nome sugere, são realizadas sem a necessidade de novo planejamento, executadas diretamente no ambiente.

Como detalhado na Seção 2.2, os agentes utilizam lógica processual e inferencial para analisar suas crenças, desejos e intenções, utilizando o mecanismo de planejamento apresentado, conceitualizado na existência da biblioteca de planos apresentada. Conforme ilustrado na Figura 1, no elemento 'Biblioteca de Planos', para fornecer lógica ao sistema, modelamos a biblioteca de planos formalizada como ps. Cada elemento p_i pertencente a ps é uma tupla $\langle g, ctx, [a_1, \ldots, a_n] \rangle$. Para executar um plano diante de um cenário específico do ambiente, busca-se um plano correspondente a um objetivo g_1. Procuramos por um elemento $p_i = \langle g_1, ctx, A' \rangle \in ps$ tal que $ctx \subseteq B$. Se tal elemento for encontrado, retornamos A', que representa sequência de ações e subobjetivos que implementa aquele plano (que implementa uma forma de atingir aquele objetivo específico). De forma mais especifica, a função 'busca' procura, dentro dos planos, um plano para o objetivo g_1 com um contexto específico ctx satisfeito, retornando a sequência de ações A' implementada por aquele plano. Um plano será selecionado apenas se o contexto estiver presente nas crenças do agente, ou seja, $ctx \subseteq B$, estabelecendo uma precondição para a execução daquele plano.

Após o empilhamento da sequência de ações definidas por um plano, o agente passa a executar aquela sequência no formato de sua intenção. Após a execução de uma ação, haverá uma nova percepção do agente em relação ao ambiente, que será convertida em uma crença (nova ou atualização de uma já existente), fechando assim o ciclo de raciocínio

do agente.

Do ponto de vista do pseudocódigo apresentado, e sua formalização, podemos demonstrar que é possível implementar aspectos importantes relacionados ao funcionamento contínuo e autônomo de um agente, entre eles que os agentes que implementam esse pseudocódigo são capazes de executar tanto *achievement goals* como também *mantainance goals* (ELLIOT; THRASH, 2001; DUFF; THANGARAJAH; HARLAND, 2014)

Definição 3.1 (Achievement Goal) *São objetivos que são alcançados por um plano contendo uma sequência de ações e subobjetivos. Uma vez executado, o objetivo é dito ter sido alcançado.*

Teorema 3.1 (Achievement Goal) *Seja g_a um objetivo do tipo achievement goal, ele será alcançado através da seleção de um plano da biblioteca de planos $\langle g_a, ctx, [a_1, \ldots, a_n]\rangle$ com o contexto ctx satisfeito, ou seja $ctx \subseteq B$, empilhando a sequência de ações $[a_1, \ldots, a_n]$ como sua intenção atual, executando cada uma das ações e subobjetivos definidos pelo plano, alcançando assim o objetivo g_a.*

Definição 3.2 (Mantainance Goal) *São objetivos que estão relacionados à manter um estado do ambiente, e implementam um ciclo continuo de ações a serem executadas, repetidamente. São implementados por planos com sequências de ações e subobjetivos, onde recursivamente empilham o objetivo a ser mantido ao final da execução do plano.*

Teorema 3.2 (Mantainance Goal) *Seja g_m um objetivo do tipo mantainance goal, ele será inicializado através da seleção de um plano da biblioteca de planos $\langle g_m, ctx, [a_1, \ldots, a_n, g_m]\rangle$ com o contexto ctx satisfeito, ou seja $ctx \subseteq B$, empilhando a sequência de ações e objetivo $[a_1, \ldots, a_n, g_m]$ como sua intenção atual, executando cada uma das ações e subobjetivos definidos pelo plano, onde, ao final da execução, o objetivo g_m é recursivamente selecionado para execução.*

Perceba que, durante a execução do agentes, o estado de sua base de crenças B muda. Dessa forma, objetivos do tipo *mantainance goal* podem ser encerrados a partir de estratégias simples, implementando 2 planos $\langle g_m, ctx_1, [a_1, \ldots, a_n, g_m]\rangle$ e $\langle g_m, ctx_2, [a_1, \ldots, a_n]\rangle$, onde o objetivo g_m é mantido enquanto o contexto ctx_1 for satisfeito. A partir do momento em que o contexto ctx_1 não for mais satisfeito, o próximo plano com contexto ctx_2 poderá ser selecionado, implementando um plano que finaliza execução, sem empilhamento recursivo do objetivo g_m

4 Validação da Arquitetura nos Estudos de Caso

Além dos resultados teóricos apresentados acima, como parte da avaliação da formalização proposta, conduzimos a implementação de três estudos de caso distintos. O primeiro deles implementa abordagem proposta em um dispositivo embarcado com recursos computacionais limitados. O segundo implementa a abordagem proposta como um nó no contexto da Web 3.0. Por fim, o terceiro estudo de caso apresenta um cenário de interação entre ambas as implementações, ou seja, dispositivo embarcado e nó da Web 3.0.

4.1 Uma Implementação em Sistemas Embarcados

Nessa seção será descrito o primeiro estudo de caso, que possui o objetivo de implementar a abordagem proposta em um dispositivo de baixo poder computacional, tornando ele um dispositivo facilmente adaptável, de comportamento autônomo inteligente. Nas seções que seguem, será descrito o hardware utilizado no estudo, detalhando suas especificações, a implementação da abordagem proposta na linguagem C++, e por fim, os resultados alcançados.

4.1.1 O Hardware

Para demonstrar a abordagem proposta, o primeiro estudo de caso visa implementar a abordagem apresentada em hardware embarcado. Para esse estudo, o pseudocódigo foi implementado utilizando a linguagem de programação C++. Essa implementação foi realizada em um protótipo simples de veiculo equipado com um ESP8266 NodeMCU, conhecido pela sua documentação detalhada no site oficial da Espressif Systems (PARIHAR et al., 2019), acoplado com duas rodas motorizadas, e uma roda estática para rotação. Considerando o interesse de desenvolver dispositivos inteligentes conectados a rede de Internet, as estruturas de dados que implementam os componentes do agentes podem ser acessadas e atualizadas por requisições através de protocolos baseados em HTTP, utilizando as funcionalidades Wifi disponibilizadas pelo microcontrolador ESP8266, dentro de uma rede local. Permitindo assim que as crenças, desejos, intenções e biblioteca de planos dos agentes sejam acessíveis por meio do respectivo endereço IP atribuído pela rede ao ESP8266.

Figura 4 – O autor.

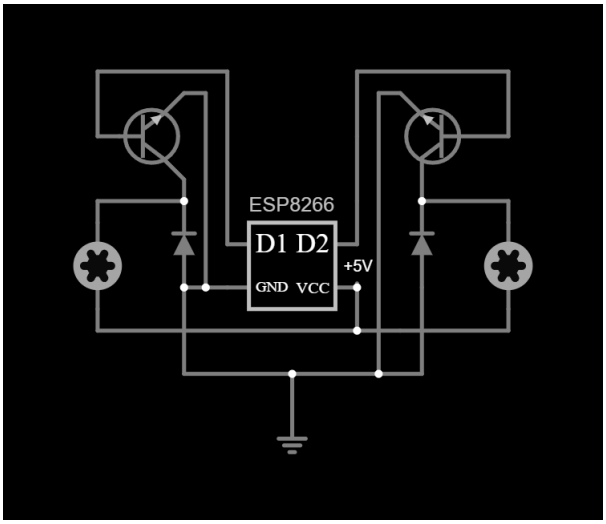

Figura 5 – Diagrama do Circuito.

Conforme ilustra o diagrama de circuito apresentado na Figura 5, uma versão simplificada do ESP8266 pode ser vista no centro da figura, apresentando somente as portas D1, D2, VCC e GND, todas devidamente especificadas na documentação da Espressif Systems (PARIHAR et al., 2019), Além disso, o circuito incorporou o uso de dois BC548 e dois diodos, fundamentais para o seu funcionamento pois as portas D1 e D2 precisam controlar dois motores que atuam como as rodas do veiculo.

O BC548 é um transistor bipolar NPN. Ele é usado como um interruptor controlado por corrente. Quando uma corrente pequena flui da base (conectada à porta do microcontrolador, como D1 ou D2) para o emissor, ele permite que uma corrente maior flua do coletor para o emissor. No contexto de um motor, o transistor é usado para controlar a alimentação do motor. Quando a porta do microcontrolador está em um estado apropriado, a corrente flui da base para o emissor, ativando o transistor e permitindo que a corrente do motor flua, fazendo o motor girar. Os diodos por sua vez são componentes eletrônicos que permitem o fluxo de corrente em uma direção enquanto bloqueiam em outra. Neste circuito, os diodos podem ser usados para proteger o circuito contra reversão de polaridade ou correntes induzidas pelo motor quando ele é desligado. Eles garantem que a energia flua apenas na direção desejada, ajudando a proteger o circuito e manter seu funcionamento adequado.

4.1.2 Agente Inteligente Embarcado

Para a implementação da abordagem proposta de agentes inteligentes utilizou-se a linguagem de programação C++, levando em consideração as peculiaridades da linguagem e suas bibliotecas disponíveis para o ESP8266. A implementação respeitou o pseudocódigo descrito na Seção 3.2 e está disponível em repositório do GitHub[1]

Na implementação do agente embarcado foi desenvolvido uma interface para o usuário em uma pagina HTML para a integração dos componentes principais da arquitetura BDI no ESP8266. Foram criados *endpoints* designados, que inclui três manipuladores: `/setplan`, `/setbeliefs` e `/setdesire`. Que recebem os parâmetros em formato de dicionario JSON, para serem analisados pelo algoritmo. As crenças são estruturadas no dicionários da seguinte maneira: `BB` (Base de Crenças) : $\{\texttt{chave}_1 : \texttt{valor}_1, \texttt{chave}_2 : \texttt{valor}_2, \ ..., \ \texttt{chave}_n : \texttt{valor}_n\}$. Por outro lado, os desejos e as intenções são representados como listas de strings, na forma $[\texttt{valor}_1, \texttt{valor}_2, ..., \texttt{valor}_n]$. Já os planos possuem uma estrutura JSON mais complexa, pois levam em consideração o contexto no qual o plano deve ser aplicado, ou seja, as crenças que o agente deve ter sobre o ambiente para executar o plano, além do nome do plano. Assim, a estrutura é a seguinte: $\{\texttt{meta} : \{\texttt{contexto} : \{\texttt{conjunto de crenças}\}, \texttt{plano} : [\texttt{sequência de ações}]\}\}$. Nessa estrutura, a base de crenças (BB) é uma coleção de pares chave-valor em formato JSON, representando as informações ou conhecimentos que o agente possui sobre o ambiente. Os desejos e intenções são listas de strings que indicam os desejos e intenções que o agente busca alcançar. Por fim, os planos são estruturados em um formato JSON mais elaborado, onde cada plano possui uma meta específica, contexto (conjunto de crenças necessárias para sua execução) e uma sequência de ações a serem realizadas para atingir esse objetivo.

Não nos ateremos às nuances da linguagem C++, uma vez que a implementação segue o pseudocódigo que foi apresentado na Seção 3.2. No entanto, é válido destacar as particularidades definidas no arquivo 'action.c', onde estão listadas as ações básicas que o agente pode executar no ambiente. Essas ações incluem `HighM1`, `HighM2`, `LowM1` e `LowM2`, cada uma com suas respectivas chamadas de função para atualizar as crenças do agente. Cada ação é chamada pela função `voidAction` ::, usando os comandos de $\texttt{analogWrite}(\texttt{pinM1}, 300)$; e $\texttt{setStatus}(\text{“}\texttt{OK}\text{”})$;, respectivamente esses comandos, configuram a porta D0, denominada por pinM1, definindo a largura do pulso do sinal PWM no

[1] <https://github.com/Mrmichelerocha/hard_bdi_motor>.

pino especificado para 300, Subsequente chama a função que atualiza a variável *belief* do agente que manda um dicionario json para o `/getbeliefs`.

Na implementação, também é possível observar um conjunto de *endpoints* que inclui três recursos: `/getplan`, `/getbeliefs` e `/getdesire`. Esses *endpoints* fornecem informações sobre as percepções do agente para qualquer pessoa que tenha o ip acesse eles, embora essa funcionalidade não tenha sido explorada no primeiro estudo de caso.

4.1.3 Estudo de Caso

O primeiro estudo de caso, embora simples, sua estrutura abarca uma lógica de ações e subplanos que permitem ao agente embarcado em um robô simples navegar no ambiente, com objetivo de somente percorrer o ambiente com planos e subplanos montados apartir de suas funcionalidades. Um breve video demonstrando o funcionamento do robô no ambiente com suas diretrizes pode ser visualizado no video demostrativo[2]. Primeiramente, a base de ações do agente é composta pelas ações `HighM1`, `HighM2`, `LowM1` e `LowM2`, essas ações ligam o motor um e dois, como os desligam. Essas ações combinadas compõem os planos, que são as funcionalidades do robô no ambiente, usadas para atingir objetivos básicos de movimentação, como por exemplo:

{"`virar_direita`", {"`m1`" : "`OK`"}, ["`HighM1`", "`HighM1`", "`HighM1`", "`LowM1`"]}

{"`virar_esquerda`", {"`m2`" : "`OK`"}, ["`HighM2`", "`HighM2`", "`HighM2`", "`LowM2`"]}

{"`andar_frente`", {"`m2`" : "`OK`", "`m1`" : "`OK`"}, ["`HighM2`", "`HighM1`", "`LowM1`", "`LowM2`"]}

Esses subplanos, quando ativados, possibilitam que o agente execute os movimentos essenciais. A partir desses subplanos, pode-se desenvolver planos mais complexos para a navegação do agente. Por exemplo:

{"`chegar_objetivo`", {"`plan`" : "`OK`"}, ["`virar_direita`", "`andar_frente`",
"`andar_frente`", "`virar_esquerda`", "`andar_frente`"]}.

Esse plano combina as ações fundamentais dos subplanos para permitir movimentos direcionados no ambiente até o objetivo. Essa composição permite a elaboração de forma incremental e gradual dos planos do agente, onde, no cenário proposto, permite alcançar uma variedade de movimentos no ambiente, como demonstrado no vídeo demonstrativo. Além disso, os planos do agente podem ser atualizados por meio de uma interface para o cliente simples em uma pagina HTML, ao receber esses planos de forma estruturada de dicionários JSON, internamente o agente usa a arquitetura SimpleBDI descrita na seção 3.2 para entender a situação e agir no ambiente.

Apesar dos componentes básicos apresentarem uma natureza simples, sua combinação e sequenciamento dão origem a um comportamento mais elaborado. Em outras palavras, a arquitetura empregada confirma a adaptabilidade do comportamento do agente, sendo necessário apenas manipular a estrutura de dados que implementa sua biblioteca de planos. No cenário proposto, como observado no video demostrativo, isso foi explorado através de diferentes configurações de objetivo de navegação, atualizando-se de forma dinâmica o plano de `chegar_objetivo`, com uma composição de acordo com o cenário de cada objetivo físico. Cada ação pode ser configurada pela rede, e essa versatilidade do sistema é possibilitada pela sua capacidade de ser reconfigurado em tempo real por meio da

[2] <https://youtu.be/KkcuBS6xCZU>.

rede. Observa-se assim que o agente pode ser facilmente atualizado, e no cenário proposto ele pode manobrar de forma adequada no ambiente, utilizando planos e subplanos de ação.

4.2 Uma Implementação na Web 3.0

Com o objetivo de integrar agentes inteligentes no contexto da Web 3.0, nesta seção, será proposta uma extensão das funcionalidades de uma biblioteca open-source disponível na internet para encapsular agentes inteligentes na Web 3.0. Assim, é possível a implementação de agentes inteligentes, usando a abordagem proposta nesse trabalho, no contexto da Web 3.0.

4.2.1 Desenvolvimento de Agentes na Web 3.0

Para integrar os agentes inteligentes no contexto da Web 3.0, procurou-se estender as funcionalidades de uma biblioteca open-source pública na internet, a Fetch.ia[3], que propôs a estrutura de agentes inteligentes com os benefícios que a tecnologia blockchain pode oferecer no contexto de sistemas multiagentes, além de ser uma das primeiras iniciativas no contexto. Por esses motivos escolhemos essa tecnologia como base para a implementação da abordagem proposta nesse trabalho no contexto da Web 3.0. A implementação foi realizada em Python utilizando a biblioteca da Fetch.ia (FETCH.AI, 2023). Essa biblioteca oferece recursos avançados de contratos na blockchain, aproveitando o potencial da Web 3.0 para automatizar e executar acordos e interações entre os agentes, porém carece de aspectos sofisticados de inteligencia, os quais foram incorporados a partir das extensões propostas neste trabalho.

Para aprofundar o estudo e formalização das Seções 2.3 e 3.2, recorremos ao registro dos agentes por meio de contratos no que foi denominado pela biblioteca como contrato "Almanaque", e se encontra detalhadamente na sua documentação (FETCH.AI, 2023). Um contrato inteligente é essencialmente um programa de computador que estabelece regras e condições para a execução de transações automatizadas, eliminando a necessidade de intermediários e aumentando a confiabilidade e segurança nas interações, conforme discutido em (DIAS et al., 2021). Para cada registro, os agentes do sistema devem fornecer prova de propriedade do endereço, assinando um número de sequência com sua chave privada e enviando essa assinatura para verificação no contrato. O número de sequência é incrementado a cada registro bem-sucedido e pode ser consultado automaticamente. O processo de registro é detalhadamente definido no seguinte código:

```
1    from uagents.setup import fund_agent_if_low
2    from uagents import Agent
3    agent = Agent(
4        name="alice",
5        port=8000,
6        seed="agent1 secret phrase",
7        endpoint=["http://127.0.0.1:8000/submit"])
8    fund_agent_if_low(agent.wallet.address())
```

O trecho de código apresentado acima ilustra a criação de uma instância de agente dentro do sistema. Neste exemplo, estamos nomeando o agente como "alice" e atribuindo

[3] <https://fetch.ai/>

a ele uma porta local de comunicação (8000). Além disso, fornecemos uma chave ("*agent1 secret phrase*") que será usada para a identificação do agente.

É importante notar que o agente pode ser configurado para se comunicar tanto em um ambiente local quanto em uma rede global pela Internet. O endereço do agente pode ser definido de forma flexível para atender às necessidades de distribuição e conectividade em rede. Quando um agente se registra no contrato "Almanaque", ele precisa especificar os "terminais de serviço" que está disposto a oferecer. Cada terminal de serviço representa uma função ou capacidade específica que o agente pode desempenhar no sistema. Para tornar a seleção de terminais de serviço mais flexível e relevante, o agente atribui a cada um deles um "parâmetro de peso". Esse parâmetro de peso reflete a importância ou prioridade relativa de cada terminal de serviço em relação aos outros. Quando outro agente tenta se comunicar com este agente registrado, o sistema utiliza as indicações fornecidas pelos terminais de serviço e seus parâmetros de peso para determinar qual terminal de serviço será selecionado para atender à solicitação. Se o agente que faz a solicitação não especificar um terminal de serviço em particular, o sistema selecionará aleatoriamente um dos terminais disponíveis, levando em consideração os pesos atribuídos. Isso garante uma alocação justa e eficiente dos recursos e capacidades oferecidas pelos agentes registrados no contrato "Almanaque".

O *token* gerado para o agente pelo contrato inteligente é o principal identificador do agente dentro da rede, fornecendo um alto nível de segurança. Outros agentes podem usar esse acesso criptografado para consultar as informações do agente no contrato Almanaque e estabelecer uma comunicação segura por meio do nó de rede. Essa criptografia do agente permite exclusivamente a troca de informações com agentes que possuam a chave correspondente, garantindo assim a privacidade e a segurança das comunicações. O código para acesso desas informações é:

```
1    from uagents import Agent
2    agent = Agent(name="agent 1")
3    print("uAgent address: ", agent.address)
```

A funcionalidade descrita acima desempenhou um papel fundamental na arquitetura proposta neste artigo. Ao implementar essa abordagem, garantimos que, ao criar novas crenças, desejos ou intenções para um agente no sistema, somente outros agentes que possuam a chave de acesso do agente correspondente possam acessá-las.

Essa medida de segurança visa proteger a privacidade e a confidencialidade das informações dos agentes, permitindo um compartilhamento seletivo e controlado de dados. Ao restringir o acesso apenas aos agentes autorizados, evitamos o acesso não autorizado ou o vazamento de informações sensíveis para agentes não confiáveis.

4.2.2 Agente Inteligente na Web 3.0

Seguindo a abordagem proposta na Seção 3.2, incorporamos ela à rede blockchain, implementando a abordagem proposta usando Python. Nesta seção será descrito os componentes principais dispostos nessa implementação, começando com a classe central do agente. Nessa classe é definido o contexto[4] (ctx) do agente em relação ao sistema, este

[4] É importante mencionar que lidamos com dois tipos de contexto representados por 'ctx'. No contexto da nossa arquitetura, temos o 'ctx ou CONTEXT' que se refere ao contexto do plano interno do agente. Além disso, existe um segundo 'ctx' que lida com o contexto do agente no sistema. Na biblioteca, esse contexto está relacionado ao nó de rede que informa ao sistema qual agente está em operação.

contexto do agente no sistema é crucial, pois indica se desejamos adicionar uma crença, um desejo ou um plano à estrutura de conhecimento do agente. Abaixo, apresentamos o código base para definição de contexto:

```
agent.belief(ctx, 'KEY', 'Value')
agent.desire(ctx, 'Value')
agent.intention(ctx, 'Value')
agent.set_plan_library(ctx,'GOAL',{'CONTEXT':'set of beliefs'},['PLAN'])
```

Essa estrutura JSON[5] só pode ser acessada pelo próprio agente ou por outros agentes que possuam a chave de acesso criptografada correspondente (*token*). Essa abordagem garante a segurança dos dados na blockchain, que é a tecnologia utilizada pelo sistema para operar de forma descentralizada na rede, as informações contidas na estrutura JSON permanecem protegidas contra acessos não autorizados. Isso promove a confidencialidade e a integridade dos dados, garantindo que apenas as partes confiáveis tenham permissão para visualizar e interagir com as informações contidas na estrutura JSON.

Um dos aspectos essenciais dos agente é seu raciocínio, no encapsulamento de agentes fornecido pela estrutura da Fetch.AI, os agentes possuem, originalmente, um ciclo de raciocínio disparado por: eventos ou períodos definidos. Funcionalidade que foi estendida para incorporar a arquitetura proposta nesse trabalho. Originalmente, um evento ocorre quando o agente é iniciado e colocado em execução, conhecido como evento de "*startup*". Por outro lado, a função "*on_interval*" é executado periodicamente de acordo com um período de tempo extipulado, que dá o ciclo de raciocínio do agente. Esse ciclo de raciocínio permite que o agente capture novas informações à medida que o tempo avança e essas informações se alteram. O código a seguir exemplifica esse ciclos do sistema:

```
@agent1.on_event('startup')
async def event_initil(ctx: Context):
 agent1.belief(ctx, 'KEY', 'Value')
 agent1.desire(ctx, 'Value')
 agent1.set_plan_library(ctx,'GOAL',{'CONTEXT':'set of beliefs'},['PLAN'])
```

```
@agent1.on_interval(period=10.5)
async def interval(ctx: Context):
 agent1.update_intention(ctx)
 action.action_environment(ctx)
  if agent1.contexto(ctx, {"CHAVE": "VALOR", "CHAVE2": "VALOR2"})
  else False
```

Nesse trecho de código, de forma genérica, pode-se inserir a implementação específica do ciclo de raciocínio do agente. Isso pode incluir a definição de eventos de inicialização e encerramento, bem como a lógica para o período de execução periódica. Foi através da implementação desses métodos que incorporou-se a arquitetura proposta nos agentes da Fetch.AI a arquitetura SimpleBDI descrita na seção 3.2. O código trata de maneira abrangente o uso da arquitetura BDI, e pode ser modulado para qualquer aplicação relevante. É importante observar que a implementação dos planos do agente ocorre dentro dos eventos

[5] <https://www.json.org/json-en.html>

mencionados. O evento de "*startup*" é usado para iniciar o agente com suas crenças, planos e desejos, como mostrado no código o "*agent*1", inicia adicionando uma belief genérica a sua BB(belief base), como um desejo e um plano e no "*on_interval*" executa periodicamente o ciclo de raciocínio do agente, com um "`period=10.5`" onde ele fica checando se houve uma nova percepção no agente em "`update_intention(ctx)`" e executa a ação "`action.action_environment(ctx)`" somente se o contexto "`contexto(ctx, {chave`$_1$`: valor`$_1$`,chave`$_2$`: valor`$_2$`})`" for satisfeito.

À medida que o agente é atualizado periodicamente no "*on_interval*", ele realiza ações no ambiente caso o contexto seja valido que podem afetar suas crenças, dando novas percepções e consequentemente executando novas ações. Essa abordagem permite que o agente trabalhe com um conjunto de planos pré-definidos para alcançar seus objetivos. O comportamento do agente se adapta a mudança de suas crenças e interações no ambiente. Conforme o tempo avança e o ambiente muda, o agente executa ações que têm impacto nas suas crenças, e trabalha em direção às metas desejadas. Embora tenhamos descrito apenas alguns aspectos da extensão proposta, sua implementação completa é mais detalhada no estudo de caso apresentado na seção 4.2.3. Nessa seção, disponibilizamos um código no GitHub e um exemplo para fornecer uma demonstração do estudo.

4.2.3 Estudo de caso

Um exemplo prático para ilustrar e solidificar os conceitos da arquitetura proposta foi criado e está disponível em repositório público[6]. O cenário proposto para este estudo de caso é o uso dos agentes em uma simulação de uma *smarthome*, onde o agente pode controlar a temperatura e a iluminação do ambiente de acordo com as preferencias do usuário. Para fornecer uma compreensão mais aprofundada do cenário de teste e facilitar a visualização prática do funcionamento dos agentes de smarthome, descrevemos o ambiente simulado com mais detalhes. A simulação é realizada em um ambiente virtual que replica as condições de uma casa inteligente, onde são definidos parâmetros como a localização dos cômodos, a temperatura padrão desejada e as condições de iluminação. Nesse ambiente, o agente opera com autonomia para tomar decisões baseadas em seu conjunto de crenças e objetivos. Por exemplo, se ele percebe que as condições de temperatura estão diferentes da preferência do usuários, ele ativa o sistema de aquecimento ou resfriamento para alcançar a temperatura desejada. Similarmente, ajusta a iluminação com base no horário do dia, intensificando a luz artificial conforme o crepúsculo avança. Este teste em um cenário controlado e simulado demonstra a capacidade do agente de executar tarefas de maneira autônoma e adaptativa, mostrando como as preferências do usuário são atendidas de maneira dinâmica e inteligente.

No cenário modelado, o agente inicialmente acredita que o usuário esta no quarto, que a temperatura do local é diferente de 24ºC (a preferência do usuário) e o horário é 19h. O agente da *smarthome* está equipado com planos para os seguintes objetivos na sua biblioteca de planos: `regular_iluminacao`, `regular_temperatura` e `regular_ambiente`. O plano `regular_iluminacao` está associado ao controle de iluminação em um determinado local, o `regular_temperatura` lida com o ajuste da temperatura em função de um valor específico de 24 graus, e o `regular_ambiente` é um plano mais amplo que engloba tanto o controle de iluminação quanto a temperatura, considerando também o horário e local. Dado isso, as medidas relevantes para manter o ambiente ajustado são utilizadas. Os planos são especificados em um arquivo JSON, sendo então incorporados à biblioteca de planos

[6] <https://github.com/Mrmichelerocha/ag-bdi>.

do agente. Abaixo, descreve-se a estrutura desse arquivo, que podem ser observada em maiores detalhes no código disponibilizado.

{“regular_iluminacao”, {“local” : “quarto”}, [“desligar”]}

{“regular_temperatura”, {“temperatura_diferente” : 24}, [“ajustar_temperatura”]}

{“regular_ambiente”, {“horario_passado” : 19, “local” : “quarto”},
[“regular_iluminacao”, “regular_temperatura”]}.

Com esse cenário, justifica-se o uso da Web 3.0 e da Blockchain quando consideramos que os dados utilizados pelos usuários de dispositivos IoT em casas inteligentes são frequentemente analisados e utilizados por empresas terceiras que fornecem o equipamento, sem conceder ao usuário autonomia sobre seus dados. Essa autonomia é um dos pilares da Web 3.0. Ao conferirmos essa autonomia sobre as preferências do usuário ao utilizar seus dispositivos IoT, precisamos garantir sua segurança, que é alcançada por meio da utilização da rede blockchain para armazenar esses dados, nesse exemplo caso o usuário queira compartilhar suas informações ele pode fazer por meio do compartilhamento do *token* de acesso a essas informações.

4.3 Um Estudo de Caso Combinando Agentes Embarcados e Agentes na Web 3.0

É importante enfatizar que o sistema desenvolvido neste estudo representa apenas uma pequena amostra do potencial da arquitetura proposta. Devido à sua fundamentação em crenças e planos, a abordagem proposta possui uma flexibilidade significativa, podendo ser implementada para vários tipos de cenários. Os agentes desenvolvidos, por sua vez, podem estar hospedados em múltiplos nós de uma rede e espalhados por diversos servidores, o que facilita a integração de várias técnicas, requerendo aspectos de comunicação entre eles. Destaca-se, no entanto, a capacidade de adaptação dos agentes, permitindo que eles evoluam e respondam de maneira eficiente às mudanças ao longo do tempo.

Nesse estudo de caso, temos o dispositivo em hardware, implementado para o primeiro estudo de caso, apresentado na Seção 4.1.1, e um agente central, ou também referenciado como “central de controle", que executa como nó da Web 3.0. O estudo elaborado de sistemas multiagentes integra ambas as implementações apresentadas nos estudos de caso anteriores. Foi desenvolvido como um sistema que recebe as coordenadas atuais e desejadas do robô embarcado. A central de controle elabora um plano correspondente e o transmite pela rede, o plano elaborado utiliza somente as coordenadas atuais e desejadas, criando automaticamente um plano ao receber as informações das coordenadas, permitindo que o robô se movimente com suas respectivas ações para chegar até seu objetivo desejado. O agente utiliza o plano personalizado desenvolvido pela central de controle para cada mudança nas coordenadas desejadas, e caso o objetivo desejado seja modificado (e consequentemente um plano diferente enviado), o agente reorganiza sua rota e segue em direção ao novo objetivo. O agente central possui uma estrutura de dados, que chamamos de “memória do agente” ou “*memory*”. Essa memoria foi criada para facilitar tanto o acesso de informações, como também permitir entradas do usuário ao sistema, no caso as coordenadas atuais e desejadas do agente, servindo como um meio de interação com o usuário do sistema. Dessa forma, considerando que o agente do primeiro estudo de caso não possui sensores ou visão para se localizar no ambiente, ele fica limitado as informações que são disponibilizadas para ele pela memoria do agente. Nesse sentido, o agente central tem o objetivo de gerar planos de ação para o agente de hardware e enviar para que o mesmo possa executá-los. Simulando uma situação em que um agente de planejamento

Fonte: O autor

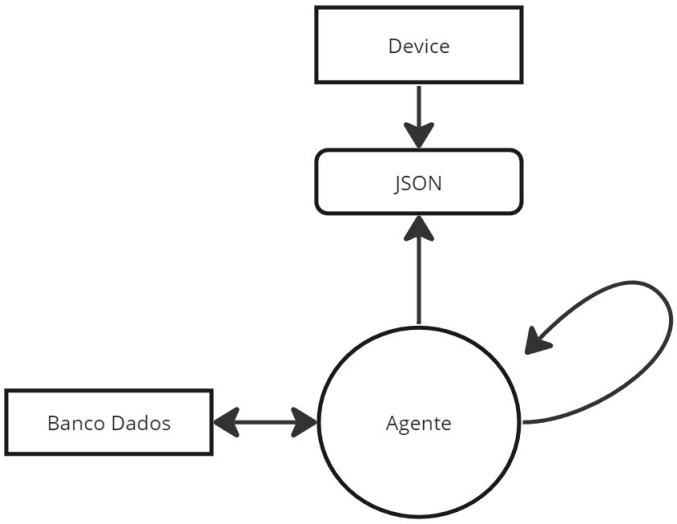

Figura 6 – Metodologia de Monitoramento dos Agentes.

pode instruir outros agentes sobre bons planos de ações para atingir seus objetivos. A Figura 6 ilustra o mecanismo através do qual foi baseada a implementação, em nossa metodologia o dispositivo compartilha dados JSON em uma página HTML.

Como estudo de caso deste projeto, temos um robô com coordenadas iniciais (x, y) e um objetivo com coordenadas desejadas (x', y'). Ambas as informações são setadas dentro da memoria do agente (banco de dados). A central de controle ao receber estas coordenadas, que são atualizadas manualmente pelo usuário, elabora um plano personalizado para o agente chegar até o seu objetivo, e transmite o plano para o agente embarcado no dispositivo de hardware (robô), descrito na Seção 4.1. Em posse deste plano, o agente cria uma intenção para chegar no objetivo proposto, passando assim a se movimentar em direção ao objetivo. Embora as coordenadas sejam atualizadas pelo usuário, houveram tentativas de implementar o estudo de caso usando computação visual, onde um terceiro agente, com essa capacidade de identificação visual, transmitiria as coordenadas para a central de controle automaticamente, como as coordenadas seriam atualizadas a medida que o agente se movimenta no ambiente, a movimentação do agente seria totalmente automática e autônoma a partir das suas crenças sobre o ambiente. Nesse estudo falho, houve sucesso na implementação do reconhecimento do agente e do objetivo por meio da computação visual, porém a baixa taxa de *frames* criou um problema na comunicação da central com o agente hardware, pois ao reconhecer o agente e o objetivo e mandar o primeiro plano para o agente se movimentar, o frame demorava muito para se atualizar, fazendo o agente novamente tomar a mesma ação no ambiente, pois a imagem não tinha sido atualizada fazendo o agente acreditar que ainda estava na posição anterior, e não na nova, tornando a otimização do frame inviável para o tempo restante trabalho. Diante disso incorporar um agente com visão computacional ficou como trabalho futuro desse projeto.

Como a utilização da visão computacional não teve exito foi usado o banco de dados SQLite. O banco de dados funciona como uma interface entre o usuário do sistema

e a central de controle, onde inputs de coordenadas mostram a localização do robô e do objetivo no ambiente. Com essas informações a central de controle pode elaborar um plano para que o agente de hardware seja capaz de se mover em direção ao objetivo, encaminhando esse plano para o agente de hardware, o mesmo é capaz de se mover e atingir o objetivo proposto. O hardware, por sua vez, recebe os planos transmitido pela central de controle através de uma solicitação HTTP POST, que contém informações relativas a crenças, planos e desejos. O robô inicia a execução do plano personalizado, que faz uso das ações básicas que o agente hardware possui, criando planos para se mover para frente, virar para esquerda/direita, melhor descritos na Seção 4.1.2. Este conjunto restrito de ações pode ser combinado de diferentes maneiras conforme forem atualizadas pela central de controle, permitindo uma movimentação coordenada e estratégica[7]. Os códigos da implementação foram divididos entre o hardware[8], agentes[9] e memória[10].

5 Trabalhos Relacionados

Na condução deste estudo, buscamos analisar a aplicação da arquitetura BDI em contextos que envolvem a integração com redes blockchain e dispositivos IoT (*Internet of Things*). Estabelecemos uma *string* de busca, que foi aplicada nas bases de dados científicas SPRINGER e IEEE. As palavras-chave selecionadas para a busca foram: "`hardware`", "`blockchain`", "`agents`", "`architecture BDI`", "`web3.0`" e "`embarcados`". A string de busca formulada foi: **`("BDI" OR "artificial intelligence" OR "BDI architecture" OR "belief desire intention" OR "Multi-Agentes" OR "agents") AND ("hardware" OR "embedded systems") AND ("blockchain" OR "Web3.0")`**. Essa busca resultou em mais de 12 mil artigos em ambas as bases cientifica.

Para aprimorar a seleção de artigos para nossa pesquisa, estabelecemos critérios claros de inclusão e exclusão. No que diz respeito aos critérios de exclusão, optamos por descartar qualquer artigo que não tenha sido escrito em inglês ou português, bem como aqueles publicados antes de 2018, excluindo ainda artigos com mais de 15 páginas de extensão ou que sejam *short papers* (com menos de 4 páginas). Também excluímos estudos que se concentram em livros e conference papers. Em contrapartida, os critérios de inclusão incluem as palavras-chave, a presença das palavras-chave no resumo ou nas palavras-chave do artigo, e a conformidade do artigo com o escopo de nossa pesquisa, garantindo assim a seleção de trabalhos pertinentes para nossa análise.

Com base nestes critérios de seleção, realizamos uma triagem nas bases de dados da SPRINGER e IEEE, identificando um total de 24 artigos dentre os 861 resultados obtidos na primeira base, e 19 artigos dentre os 1020 resultados da segunda, resultando 43 artigos selecionados. Essa seleção visa direcionar nossa análise de forma precisa e eficiente. Além disso, para melhor guiar nossa investigação, formulamos as questões de pesquisa que servirão para norteadores o nosso estudo, as quais são mostradas na Tabela 1.

A primeira questão teve como objetivo estabelecer critérios claros em nossa busca para identificar resultados que não apenas mencionam a arquitetura BDI, mas a utilizam efetivamente como uma estrutura de modelagem de casos de uso. Os resultados dessa análise estão representados na Tabela 1, que ilustra que, dos 43 artigos examinados, apenas 5 deles de fato implementam a arquitetura BDI em seus estudos.

[7] Um vídeo demonstrativo pode ser acessado em <https://youtu.be/KkcuBS6xCZU>
[8] O código está disponíveis em: <https://github.com/Mrmichelerocha/hard_bdi_motor>.
[9] O código está disponíveis em: <https://github.com/Mrmichelerocha/ag-bdi-central>.
[10] O código está disponíveis em: <https://github.com/Mrmichelerocha/ag-bdi-memory>.

Pergunta	**Sim**	**Não**
O trabalho cita a arquitetura BDI como um dos componentes principais	5	38
O trabalho utiliza sistemas multiagentes	17	26
O trabalho utiliza a arquitetura BDI em uma rede blockchain multiagentes	2	41
O trabalho utiliza a arquitetura BDI ou sistemas multiagentes em dispositivos inteligentes	9	34
O trabalho utiliza a rede blockchain em dispositivos inteligentes	25	18

Tabela 1 – Resultados da Análise

Já em relação à segunda questão, ela se concentra na utilização de sistemas multiagentes. A Tabela 1 evidencia que uma proporção significativa de artigos aborda sistemas multiagentes, sugerindo que muitos trabalhos se concentram nessa abordagem. No entanto, é importante notar que a maioria desses artigos não incorpora a arquitetura BDI em suas modelagens. Isso ocorre devido ao fato de que, embora esses sistemas envolvam a interação entre agentes, eles frequentemente carecem de uma inteligência cognitiva subjacente. Como resultado, os sistemas que efetivamente empregam a arquitetura BDI se destacam em meio a essa abordagem, demonstrando um diferencial.

A terceira questão foi formulada com o propósito de investigar se algum dos estudos encontrados nas bases científicas já havia incorporado a arquitetura BDI em contextos relacionados a blockchain. A intenção subsequente nessa análise era examinar as vantagens e perspectivas dessa integração. Observamos, no entanto, que o número de artigos que abordam essa questão é notavelmente escasso, como evidenciado na Tabela 1. Isso ressalta que o trabalho em questão se posiciona como uma das poucas iniciativas a explorar a integração desses dois domínios emergentes.

A quarta questão foi formulada com o objetivo de orientar nossa pesquisa na identificação de artigos que empregam a arquitetura BDI ou sistemas multiagentes em dispositivos inteligentes, contextualizando-os com o estudo de caso proposto no presente artigo. Este estudo visa a aplicação dessas tecnologias em dispositivos embarcados com conectividade à internet. Dos 43 artigos analisados, identificamos 9 que incorporam ou mencionam o uso de sistemas multiagentes em suas pesquisas, conforme ilustrado na Tabela 1. Embora, de acordo com nossa *string* de busca, haja uma aderência relativamente baixa dos sistemas multiagentes ao contexto do IoT, essa aderência não é escassa. A presença desses 9 artigos demonstra o significativo potencial que esses sistemas possuem nesse cenário, algo que exploraremos em detalhes neste artigo.

Por fim, a última questão tem como objetivo identificar os artigos que abordam a combinação de blockchain com IoT. Como revelado na Tabela 1, a maioria desses artigos menciona tanto IoT quanto blockchain como componentes essenciais para o futuro, o que demonstra que a ascensão dessas tecnologias é de grande interesse para o desenvolvimento das aplicações tecnológicas futuras.

Após realizar essas análises, notamos que nenhum dos artigos obteve respostas afirmativas em todas as questões, levando-nos a considerar para nossa revisão apenas aqueles que receberam 3 ou 4 respostas positivas. Diante desses critérios analisamos 5 artigos onde optamos por adotar o método de revisão rápida PECO(População; Exposição; Comparação; Resultados)(MORGAN et al., 2018) para avaliar as conclusões apresentadas nos artigos e compará-las com a relevância de nossos estudos de caso.

O primeiro trabalho relacionado (WILLIAM et al., 2022) analisa uma população de

sensores de baixa potência e destaca seu aumento de capacidade em eficiência energética e processamento. O trabalho faz uso de agentes autônomos do tipo crença-desejo-intenção em nós de sensores. O estudo apresentado envolveu um agente BDI de baixo consumo de energia para controlar a iluminação de um ambiente. Esse agente demonstrou comportamento reativo e proativo, respondendo rapidamente a mudanças nos objetivos do usuário e ajustando a iluminação quando necessário. Um ponto forte foi a eficiência energética, com ciclos de raciocínio extremamente rápidos, resultando em um baixo consumo de energia em comparação com abordagens baseadas apenas na percepção do ambiente. No entanto, o sistema possui limitações, como suporte para apenas um agente, o que restringe sua aplicação em áreas maiores. Nós acreditamos que com sensores semelhantes em nosso estudo os resultados poderiam ser equiparáveis, além do nosso estudo resolver desafios de comunicação mencionados no artigo.

O segundo trabalho (VACHTSEVANOU et al., 2023) também envolve uma população de sensores de baixa potência. Sensores de baixa potência são dispositivos projetados para detectar e responder a diferentes tipos de dados físicos — como luz, temperatura, movimento, umidade, pressão, entre outros — utilizando uma quantidade mínima de energia elétrica para funcionar. O artigo se concentra na exposição da integração de agentes autônomos na IoT com baixa potência. Nossa abordagem compartilha semelhanças com o estudo, pois visamos agentes eficientes para dispositivos de baixo consumo de energia.

O terceiro trabalho(SONG et al., 2022) trata de uma população de redes definidas por software em ambientes multi-domínio, destacando desafios de segurança. Neste estudo, introduziram uma arquitetura baseada em BDI (Crença-Desejo-Intenção) em um controlador de resiliência de SDN (Rede Definida por Software). A estrutura destaca um Controlador de Rede de Resiliência (RNC) com blockchain, incluindo seus principais componentes, como o publicador de eventos, executor de ações e adaptador. Ao comparar o trabalho mencionado com o estudo conduzido por esta pesquisa, concentramos nossa atenção na criação de sistemas inteligentes, com a ênfase em segurança, para segurança da informação obtidas e analisadas dos agentes.

Já o quarto trabalho (AMIRI et al., 2023) aborda ambientes distribuídos, com população tecnologias relacionadas à IoT. Isso engloba ambientes de nuvem, borda, neblina, Internet dos Drones (IoD) e Internet dos Veículos (IoV). Este estudo concentra-se no desenvolvimento e aprimoramento do controle de sistemas de *fog computing* interconectados e heterogêneos, com o objetivo principal de melhorar a eficiência energética, fusão de dados, *offloading* de computação, qualidade de serviço e suporte móvel para redes de *fog computing*. Reconhecemos a importância crítica da eficiência energética, especialmente em sistemas de névoa que frequentemente operam com dispositivos alimentados por baterias, e enfatizamos que otimizar o consumo de energia pode significativamente prolongar a vida útil desses sistemas. Em comparação com nossos estudo de caso, esse artigo tem seu objetivo atuando mais como um inspirador futuro do que uma comparação dado que nossos vieses são diferentes, mas ainda assim sua importância é notável.

Finalmente, o quinto trabalho (FRIKHA et al., 2023) parece se concentrar em aplicações que demandam recursos computacionais e de armazenamento de memória intensivos, incluindo aplicações baseadas em Inteligência Artificial, reconstrução 3D e Blockchain. Este estudo se dedica ao desenvolvimento e validação de uma abordagem baseada em sistemas multiagentes usando uma plataforma FPGA (Field-Programmable Gate Array). Utilizando a plataforma Xilinx ML 507, foram apresentadas duas arquiteturas: uma fixa e outra reconfigurável. Na arquitetura fixa, o Microblaze age como administrador

do sistema, com os agentes implementados em VHDL usando os componentes chave da arquitetura BDI. A aplicação trata o sistema de FPGA como varios agentes de hardware trabalhando entre si para otimizar o sistema, esse trabalho se diferencia e muito da nossa aplicação da arquitetura SimpleBDI onde implementamos a comunicação entre agentes de software e hardware diferente de tratar os componentes do hardware como agentes como o quinto artigo trata, a implementação de agentes como os componentes de hardware que precisam trabalhar entre si de maneira inteligente para otimizar o sistema é uma nova perspectiva de como tratar a arquitetura BDI não para tornar o sistema inteligente, mas cada parte de seus componentes como uma entidade capaz de tomar decisões.

6 Conclusão e Trabalhos Futuros

Este trabalho explorou a aplicação de sistemas multiagentes em dispositivos embarcados na era da Web 3.0, com um foco especial na implementação de uma arquitetura simplificada baseada na arquitetura BDI. Nossa pesquisa revelou a viabilidade e o potencial dessa abordagem, destacando os benefícios de criar agentes inteligentes capazes de tomar decisões autônomas e adaptativas em ambientes distribuídos.

Ao longo deste estudo, desenvolvemos um agente BDI para embarcados com poucos recursos computacionais, bem como promovemos a adaptabilidade do mesmo. Da mesma forma estendemos as funcionalidades de uma biblioteca de código aberto para aprimorar trabalhos já feitos na área de agentes e blockchain, fornecendo um aprimoramento de inteligencia ao sistema, que propôs um ambiente elaborado para um sistema que se comunicasse tanto em hardware como em software no contexto tecnológico atual.

Como em toda pesquisa, há espaço para aprimoramentos e trabalhos futuros. Uma área promissora é a integração dessa técnica com sistemas operacionais de tempo real, como o FreeRTOS, entre outros, para aproveitar o uso de *threads* e melhorar a eficiência e a capacidade de resposta dos dispositivos embarcados. Isso poderia abrir novas possibilidades de aplicação e tornar os agentes ainda mais eficazes em ambientes dinâmicos. Além disso, a migração de código é uma área de pesquisa em crescimento que pode ser explorada para aprimorar as capacidades dos dispositivos já implantados e estender suas funcionalidades. Isso permitiria a integração de dispositivos embarcados em sistemas mais amplos e aproveitaria o poder de processamento e armazenamento disponível, No entanto, o desenvolvimento futuro mais significativo, possivelmente o próximo passo, estará na área de comunicação, da qual no estudo não foi muito explorada entre os nodos, com a implementação de múltiplos nós de agentes distribuídos em servidores, visamos alcançar o uma distribuição inteligente de informações. Também, é possível explorar integrações com tecnologias de interação humano computador, incorporando trabalhos como (CUSTÓDIO et al., 2022; ENGELMANN et al., 2021) a abordagem apresentada nesse trabalho, bem como outras capacidades cognitivas, como a capacidade de modelar e raciocinar sobre teoria da mente (ROCHA et al., 2023; SILVA et al., 2023).

Em conclusão, este trabalho estabeleceu uma ponto de partida para a aplicação de sistemas multiagentes em dispositivos embarcados na era da Web 3.0 com blockchain, demonstrando seu potencial e identificando áreas para futuras melhorias e expansões. A pesquisa nesse campo continua a ser emocionante e promissora, oferecendo oportunidades significativas para o avanço da tecnologia e a criação de soluções mais inteligentes e adaptativas para diversos domínios de aplicação.

Referências

AHMED, A. S.; ABOOD, M. S.; HAMDI, M. M. Advancement of deep learning in big data and distributed systems. *2021 3rd international congress on human-computer interaction, optimization and robotic applications (HORA)*, p. 1–7, 2021. Citado na página [3].

ALABDULWAHHAB, F. A. Web 3.0: the decentralized web blockchain networks and protocol innovation. p. 1–4, 2018. Citado na página [7].

AMIRI, Z. et al. Resilient and dependability management in distributed environments: A systematic and comprehensive literature review. *Cluster Computing*, Springer, v. 26, n. 2, p. 1565–1600, 2023. Citado na página [25].

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. *Programming multi-agent systems in AgentSpeak using Jason*. Liverpool: John Wiley & Sons, 2007. 273 p. Citado (4) vezes nas páginas [3, 4, 6 e 7].

BRATMAN, M. Intention, plans, and practical reason. 1987. Citado na página [5].

CALVARESI, D. et al. Multi-agent systems and blockchain: Results from a systematic literature review. *Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity: The PAAMS Collection: 16th International Conference, PAAMS 2018, Toledo, Spain, June 20–22, 2018, Proceedings 16*, p. 110–126, 2018. Citado na página [3].

CAO, L. Decentralized ai: Edge intelligence and smart blockchain, metaverse, web3, and desci. *IEEE Intelligent Systems*, IEEE, v. 37, n. 3, p. 6–19, 2022. Citado na página [7].

CUSTÓDIO, M. d. S. et al. Rasa4jaca: Uma interface entre sistemas multiagentes e tecnologias chatbots open source. Araranguá, SC., 2022. Citado na página [26].

DIAS, R. et al. A blockchain-based platform for reliably tracing political contacts. *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, p. 1–6, 2021. Citado na página [17].

DUFF, S.; THANGARAJAH, J.; HARLAND, J. Maintenance goals in intelligent agents. *Computational Intelligence*, Wiley Online Library, v. 30, n. 1, p. 71–114, 2014. Citado na página [13].

ELLIOT, A. J.; THRASH, T. M. Achievement goals and the hierarchical model of achievement motivation. *Educational psychology review*, Springer, v. 13, p. 139–156, 2001. Citado na página [13].

ENGELMANN, D. et al. Dial4jaca–a communication interface between multi-agent systems and chatbots. *International conference on practical applications of agents and multi-agent systems*, p. 77–88, 2021. Citado na página [26].

FETCH.AI. *Fetch.AI - Developer Documentation*. 2023. <https://docs.fetch.ai/>. Acessado em: 04/04/2023. Citado na página [17].

FRIKHA, T. et al. Embedded decision support platform based on multi-agent systems. *Multimedia Tools and Applications*, Springer, p. 1–27, 2023. Citado na página [25].

GEORGEFF, M. P.; INGRAND, F. Decision-making in an embedded reasoning system. *International joint conference on artificial intelligence*, 1989. Citado na página [6].

GEORGEFF, M. P.; LANSKY, A. L. Reactive reasoning and planning. *AAAI*, v. 87, p. 677–682, 1987. Citado (2) vezes nas páginas [5 e 6].

GERVAIS, A. et al. On the security and performance of proof of work blockchains. *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, p. 3–16, 2016. Citado (2) vezes nas páginas [7 e 8].

GURURAJAPATHY, S. S.; MOKHLIS, H.; ILLIAS, H. A. Fault location and detection techniques in power distribution systems with distributed generation: A review. *Renewable and sustainable energy reviews*, Elsevier, v. 74, p. 949–958, 2017. Citado na página [3].

JESUS, V. Souza de et al. An ide to support the development of embedded multi-agent systems. *International Conference on Practical Applications of Agents and Multi-Agent Systems*, p. 346–358, 2023. Citado na página [3].

LEE, J. et al. Um-prs: An implementation of the procedural reasoning system for multirobot applications. *NASA. Johnson Space Center, Conference on Intelligent Robotics in Field, Factory, Service and Space (CIRFFSS 1994), Volume 2*, n. AIAA PAPER 94-1297-CP, 1994. Citado (2) vezes nas páginas [3 e 6].

MENDOZA, E. et al. Deliberative architecture for smart sensors in the filtering operation of a water purification plant. *Journal of Physics: Conference Series*, v. 1730, n. 1, p. 012088, 2021. Citado na página [5].

MORGAN, R. L. et al. Identifying the peco: a framework for formulating good questions to explore the association of environmental and other exposures with health outcomes. *Environment international*, NIH Public Access, v. 121, n. Pt 1, p. 1027, 2018. Citado na página [24].

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, p. 21260, 2008. Citado na página [7].

OSSOWSKI, S.; MENEZES, R. On coordination and its significance to distributed and multi-agent systems. *Concurrency and Computation: Practice and Experience*, Wiley Online Library, v. 18, n. 4, p. 359–370, 2006. Citado na página [3].

PALANCA, J. et al. Flexible agent architecture: Mixing reactive and deliberative behaviors in spade. *Electronics*, MDPI, v. 12, n. 3, p. 659, 2023. Citado na página [5].

PARIHAR, Y. S. et al. Internet of things and nodemcu. *journal of emerging technologies and innovative research*, v. 6, n. 6, p. 1085, 2019. Citado na página [14].

RAO, A. S.; GEORGEFF, M. P. Modeling rational agents within a bdi-architecture. *Readings in agents*, p. 317–328, 1997. Citado na página [5].

RASCHKA, S.; PATTERSON, J.; NOLET, C. Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. *Information*, Multidisciplinary Digital Publishing Institute, v. 11, n. 4, p. 193, 2020. Citado na página [3].

ROCHA, M. et al. Applying theory of mind to multi-agent systems: A systematic review. *Brazilian Conference on Intelligent Systems*, p. 367–381, 2023. Citado na página [26].

RÜB, I.; DUNIN-KEPLICZ, B. Basta: Bdi-based architecture of simulated traffic agents. *Journal of Information and Telecommunication*, Taylor & Francis, v. 4, n. 4, p. 440–460, 2020. Citado na página [5].

RUSSELL, S. J. *Artificial intelligence a modern approach*. London: Pearson Education, Inc., 2010. Citado na página [4].

SANTOS, M. M. d. et al. Programação orientada a agentes bdi em sistemas embarcados. 2022. Citado na página [3].

SHIVALINGAIAH, D.; NAIK, U. Comparative study of web 1.0, web 2.0 and web 3.0. INFLIBNET Center3, 2008. Citado (2) vezes nas páginas [7 e 8].

SILVA, H. H. da et al. Perspectivas da utilização de teoria da mente para o reconhecimento e intervenção de estresse ocupacional. *Anais do Computer on the Beach*, v. 14, p. 487–489, 2023. Citado na página [26].

SONG, Y. et al. Resilience network controller design for multi-domain sdn: A bdi-based framework. *2022 IEEE 95th Vehicular Technology Conference:(VTC2022-Spring)*, p. 1–5, 2022. Citado na página [25].

STRINGER, P. et al. Adaptable and verifiable bdi reasoning. *arXiv preprint arXiv:2007.11743*, 2020. Citado na página [5].

TANG, J.; LIU, G.; PAN, Q. Review on artificial intelligence techniques for improving representative air traffic management capability. *Journal of Systems Engineering and Electronics*, BIAI, v. 33, n. 5, p. 1123–1134, 2022. Citado na página [3].

VACHTSEVANOU, D. et al. Embedding autonomous agents into low-power wireless sensor networks. *International Conference on Practical Applications of Agents and Multi-Agent Systems*, p. 375–387, 2023. Citado na página [25].

WAN, J. et al. Artificial-intelligence-driven customized manufacturing factory: key technologies, applications, and challenges. *Proceedings of the IEEE*, IEEE, v. 109, n. 4, p. 377–398, 2020. Citado na página [3].

WILLIAM, J. et al. Increasing the intelligence of low-power sensors with autonomous agents. *Proceedings of the 20th ACM Conference on Embedded Networked Sensor Systems*, p. 994–999, 2022. Citado na página [24].

WOLFSKEHL, M. Why and how blockchain? *The Journal of The British Blockchain Association*, The British Blockchain Association, v. 1, n. 1, 2018. Citado (2) vezes nas páginas [7 e 8].

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. *The knowledge engineering review*, Cambridge University Press, v. 10, n. 2, p. 115–152, 1995. Citado (3) vezes nas páginas [3, 4 e 5].

YAMAGUCHI, Y.; YAMAGUCHI, K. Peer-to-peer public money system. *Japan Futures Research Center, Working Paper*, n. 02-2016, 2016. Citado na página [7].