

LUIZ FERNANDO NOSCHANG

**EVOLUINDO O BRINQUEDO ROPE PARA SE TORNAR UM
SMART TOY**

Itajaí (SC), dezembro de 2020



UNIVERSIDADE DO VALE DO ITAJAÍ

**PRÓ-REITORIA DE PÓS-GRADUAÇÃO,
PESQUISA, EXTENSÃO E CULTURA**

**PROGRAMA DE MESTRADO ACADÊMICO EM
COMPUTAÇÃO APLICADA**

**EVOLUINDO O BRINQUEDO ROPE PARA SE TORNAR UM
SMART TOY**

por

Luiz Fernando Noschang

Dissertação apresentada como requisito parcial à
obtenção do grau de Mestre em Computação
Aplicada
Orientador(a): André Luis Alice Raabe, PhD.

Itajaí (SC), dezembro de 2020

EVOLUINDO O BRINQUEDO ROPE PARA SE TORNAR UM SMART TOY

Luiz Fernando Noschang

Dezembro/2020

Orientador: André Luís Alice Raabe, Dr.

Área de Concentração: Computação Aplicada

Linha de Pesquisa: Informática na Educação

Palavras-chave: Smart Toys, brinquedo inteligente, RoPE, IoT

Número de páginas: 62

RESUMO

Smart Toys são uma nova modalidade de brinquedos que agregam componentes eletrônicos e microprocessadores permitindo que brinquedos se tornem sistemas embarcados. Desta forma pode se agregar ao projeto de um brinquedo características inteligentes, dotá-lo de acesso à internet e de sensores e atuadores diversos. A forma de interação com a criança pode ser amplamente enriquecida, adaptando-se às suas necessidades. O uso de reconhecimento de fala, processamento de linguagem natural, inteligência artificial e computação em nuvem podem proporcionar oportunidades de interação até então muito pouco exploradas. Neste contexto emergem aspectos éticos e riscos a privacidade e exposição de crianças que precisam ser seriamente discutidas a fim de garantir segurança para evolução deste conceito de brinquedo. Esta dissertação busca viabilizar que o brinquedo educacional RoPE, brinquedo programável educacional de baixo custo produzido no LITE - UNIVALI, possa se tornar um Smart Toy . O RoPE, apesar de ser um sistema embarcado, não pode ser considerado um Smart Toy, pois não é capaz de reagir e se adaptar de forma personalizada às interações dos usuários. Este trabalho propõe-se a modificar o RoPE para conectá-lo à Internet e implementar um sistema de comunicação usando o protocolo MQTT para permitir que ele seja controlado e monitorado remotamente. Espera-se que essas modificações sirvam como base para que transformar o RoPE em um Smart Toy e que futuras aplicações possam ser desenvolvidas explorando esta capacidade. O trabalho fundamenta a escolha pelo controlador ESP8266, descreve e detalha o projeto da alteração no robô e descreve a aplicação a ser desenvolvida como prova de conceito/estudo de caso.

LISTA DE ILUSTRAÇÕES

Figura 1. Evolução da carenagem do RoPE (da esquerda para a direita)	9
Figura 2. O brinquedo MOYA (esquerda) e seu mecanismo de detecção de movimentos	15
Figura 3. Brinquedo TinkerBot conectado a um <i>tablet</i>	16
Figura 4. Brinquedo Smartibot conectado ao <i>smartphone</i>	16
Figura 5. ScratchJr e LightBot, exemplos de <i>Smart Toys</i> não tangíveis.....	17
Figura 6. O brinquedo de programar RoPE e sua interface de programação.....	18
Figura 7. Tapetes pedagógicos do RoPE	19
Figura 8. O brinquedo Kibo	21
Figura 9. Módulo ESP-01	23
Figura 10. Exemplo de código para atualização OTA	25
Figura 11. Rede SofAP e página web hospedada no ESP8266	26
Figura 12. Código para configuração do RoPE	26
Figura 13. Arquitetura do MQTT	28
Figura 14. Conexão do ESP-01 com o RoPE.....	32
Figura 15. Trecho de código para configuração da rede Wi-Fi	33
Figura 16. Roteamento das mensagens de controle remoto.....	34
Figura 17. Sequência para envio de comandos para o RoPE.....	34
Figura 18. Encaminhamento das mensagens de evento.....	35
Figura 19. Sequência para o envio de eventos para os dispositivos	36
Figura 20. Protótipo da tela inicial do aplicativo de testes	49
Figura 21. Protótipo da tela de teste da unidade do aplicativo	50
Figura 22. Tapete para avaliação do pensamento computacional.....	51
Figura 23. Tela principal do gerenciador de testes	53
Figura 24. Execução de um teste em andamento	54
Figura 25. Os testes já executados ficam registrados.....	55
Figura 26. Um mesmo teste sendo executado novamente	56
Figura 27. Diagrama ER do banco de dados do aplicativo Smart RoPE	57

LISTA DE TABELAS

Tabela 1. Principais componentes do RoPE	19
Tabela 2. Comparação de hardware entre o ATmega328P e o ESP8266	27
Tabela 3. Requisitos funcionais do Smart Rope	32
Tabela 4. Temas do Smart RoPE	33
Tabela 5. Comando MOVE	37
Tabela 6. Comando TURN	37
Tabela 7. Comando TOGGLE_SOUND	38
Tabela 8. Comando TUNE_SOUND	38
Tabela 9. Comando PLAY_SOUND	38
Tabela 10. Comando TOGGLE_LED	39
Tabela 11. Comando INSERT_INSTRUCTION	39
Tabela 12. Comando REMOVE_INSTRUCTION	39
Tabela 13. Comando RESIZE_MEMORY	40
Tabela 14. Comando CLEAR_MEMORY	40
Tabela 15. Comando EXECUTE_PROGRAM	40
Tabela 16. Comando ABORT_PROGRAM	40
Tabela 17. Comando WAIT	41
Tabela 18. Comando PRESS_BUTTON	41
Tabela 19. Comando ADD_EVENT_LISTENER	41
Tabela 20. Comando REMOVE_EVENT_LISTENER	42
Tabela 21. Evento MOVED	42
Tabela 22. Evento TURNED	42
Tabela 23. Evento BATTERY_CHARGE_CHANGED	43
Tabela 24. Evento SOUND_TOGGLED	43
Tabela 25. Evento SOUND_TUNED	43
Tabela 26. Evento SOUND_PLAYED	44
Tabela 27. Evento LED_TOGGLED	44
Tabela 28. Evento INSTRUCTION_INSERTED	45
Tabela 29. Evento INSTRUCTION_REMOVED	45
Tabela 30. Evento MEMORY_RESIZED	46
Tabela 31. Evento MEMORY_CLEARED	46
Tabela 32. Evento PROGRAM_STARTED	46
Tabela 33. Evento INSTRUCTION_EXECUTED	47
Tabela 34. Evento PROGRAM_FINISHED	47
Tabela 35. Evento BUTTON_PRESSED	48
Tabela 36. Evento SWITCHED_ON	48
Tabela 37. Enunciados para a avaliação do pensamento computacional	52
Tabela 38. Variáveis do pensamento computacional	53

LISTA DE ABREVIATURAS

EEPROM	<i>Electrically-Erasable Programmable Read-Only Memory</i>
ER	Entidade-Relacionamento
IoT	<i>Internet of Things</i>
LCD	<i>Liquid Crystal Display</i>
LED	<i>Light Emmiting Diode</i>
MQTT	<i>Message Queuing Telemetry Transport</i>
NAT	<i>Network Address Translation</i>
OTA	<i>Over The Air</i>
PCI	Placa de Circuito Impresso
RAM	<i>Random Access Memory</i>
TLS	<i>Transport Layer Security</i>
UART	<i>Universal Asynchronous Receiver-Transmitter</i>
URL	<i>Uniform Resource Locator</i>

SUMÁRIO

1	INTRODUÇÃO.....	8
1.1	PROBLEMA DE PESQUISA.....	9
1.1.1	Solução Proposta.....	9
1.1.2	Delimitação de Escopo.....	10
1.1.3	Justificativa.....	10
1.2	OBJETIVOS.....	11
1.2.1	Objetivo Geral.....	11
1.2.2	Objetivos Específicos.....	11
1.3	METODOLOGIA.....	12
1.4	ESTRUTURA DA DISSERTAÇÃO.....	12
2	FUNDAMENTAÇÃO TEÓRICA.....	14
2.1	SMART TOYS.....	14
2.2	O BRINQUEDO ROPE.....	17
2.3	TRABALHOS RELACIONADOS.....	20
2.3.1	Tactic-Kibo.....	20
3	DESENVOLVIMENTO.....	22
3.1	MÓDULO WIFI ESP-01.....	22
3.2	PROTOCOLO MQTT.....	27
3.3	ARQUITETURA DO PROJETO.....	31
3.3.1	Requisitos funcionais.....	31
3.3.2	Conexão Wi-Fi.....	32
3.3.3	Roteamento das mensagens.....	33
3.3.4	Mensagens de controle.....	37
3.3.5	Mensagens de evento.....	42
3.4	VALIDAÇÃO.....	48
3.4.1	Aplicativo de testes.....	48
3.4.2	Estudo de caso.....	51
4	CONSIDERAÇÕES FINAIS.....	58

1 INTRODUÇÃO

O projeto RoPE (Robô Programável Educacional) vem sendo desenvolvido pelo Laboratório de Inovação Tecnológica na Educação (LITE) da Univali desde 2013. O projeto promove o desenvolvimento de brinquedos de programar que possibilitam auxiliar o desenvolvimento do Pensamento Computacional desde a Educação Infantil, mais especificamente para crianças a partir dos 4 anos de idade. O projeto é um caso de sucesso da relação entre ensino, pesquisa e extensão e tem recebido bastante atenção da mídia, conforme ilustra o website do projeto¹.

A partir de 2017 o projeto RoPE estabeleceu uma parceria com o município de Balneário Camboriú para a entrega de 30 brinquedos para os Núcleos de Educação Infantil atendendo a aproximadamente 1000 crianças (RAABE et al., 2017). No ano de 2020, o projeto foi ampliado para atender também às séries iniciais do ensino fundamental. Para isso, mais 52 robôs foram fabricados e entregues às escolas do município. Além dos robôs, vários materiais de apoio ao trabalho do professor foram confeccionados em parcerias com pesquisas da área de Educação, bem como foi realizada a formação 400 professores para o uso pedagógico do brinquedo de programar.

Desde sua primeira versão desenvolvida no ano de 2013 muitas melhorias já foram incorporadas no projeto do RoPE. A carenagem foi redesenhada diversas vezes (Figura 1), os materiais e processos de fabricação mudaram amplamente, a placa controladora foi redesenhada diversas vezes, o sistema de alimentação foi otimizado, o Firmware do robô passou também a ser frequentemente atualizado. Estas modificações, em grande parte, foram fundamentadas nas avaliações que foram realizados com as crianças e a partir dos relatos dos professores que usam o RoPE nas escolas. O vídeo denominado Rope Design Timeline², disponível no canal do YouTube do LITE ilustra muitas destas melhorias.

Com a motivação de prosseguir evoluindo o brinquedo, busca-se neste trabalho incluir suporte à internet Wi-Fi no robô. Esta evolução visa permitir incorporar novas funcionalidades que possam ampliar as possibilidades pedagógicas do RoPE, facilitar o processo de atualização do *firmware* do robô e a criação de aplicações que possam coletar e enviar dados automaticamente a partir da interação com os usuários.

¹ <http://lite.acad.univali.br/pt/clipagem/>

² <https://www.youtube.com/watch?v=mwQqM5nec8A&list=PLy2Oi5TbC70tMXnD5x36Td6WChNOPjMqO&index=2>



Figura 1. Evolução da carenagem do RoPE (da esquerda para a direita)

Aliado a esta perspectiva está a emergente área de pesquisa relacionada ao uso de Internet das Coisas (IoT) na concepção de fabricação de brinquedos. A expressão Internet of Toys, difundida em (MASCHERONI; HOLLOWAY, 2019), é uma especialização do conceito de Smart Toy que exploram as potencialidades e riscos de brinquedos que acessam a Internet. Neste sentido, a incorporação de capacidade de acesso à Internet no brinquedo RoPE irá permitir que ele se torne um *Smart Toy* e impulse um novo ramo de pesquisas a serem desenvolvidas no âmbito do LITE.

1.1 PROBLEMA DE PESQUISA

De acordo com (RAABE et al., 2017) “[...] os pesquisadores que construíram o RoPE priorizaram decisões que pudessem reduzir o custo e tornar o brinquedo aderente a realidade dos núcleos de educação infantil brasileiros”. Nesse sentido, esse trabalho busca responder à seguinte pergunta: **é possível incluir a funcionalidade de conexão Wi-Fi a um brinquedo sem onerar demasiadamente o custo de produção?**

1.1.1 Solução Proposta

Este trabalho propõe a modificação do *hardware* e do *software* do RoPE para conectá-lo à Internet através de uma rede Wi-Fi. O trabalho propõe-se também a implementar um sistema de comunicação entre o RoPE e outros dispositivos através da Internet, permitindo que o RoPE seja controlado e monitorado remotamente, com o objetivo de possibilitar a sua evolução para um *Smart Toy* em trabalhos futuros.

O sistema a ser desenvolvido possibilitará que os dispositivos enviem comandos pré-definidos para que o RoPE os execute como, por exemplo: andar para frente e para trás, girar para ambos os lados e reproduzir notas musicais. O sistema possibilitará também que os dispositivos tenham acesso a dados do RoPE relacionados a seu funcionamento como, por exemplo: o programa que está

armazenado na memória e a carga da bateria. Por último, o sistema possibilitará que os dispositivos sejam notificados sobre eventos importantes que ocorrerem com o brinquedo como, por exemplo: o pressionamento físico de um botão pelo usuário do RoPE.

1.1.2 Delimitação de Escopo

Este trabalho limita-se em habilitar o RoPE a acessar à Internet através da Wi-Fi e a desenvolver um sistema de troca de mensagens para controle e monitoramento remoto do RoPE, bem como o desenvolvimento de um aplicativo de testes para validar a implementação do sistema.

A validação do sistema será realizada com unidades específicas do RoPE em um contexto restrito e controlado. Sendo assim, na parte de segurança apenas serão aproveitados os recursos oferecidos pelas tecnologias usadas no desenvolvimento do projeto. Não será escopo deste trabalho implementar uma camada de segurança rígida para garantir a proteção das informações coletadas ou impedir a invasão do brinquedo por terceiros. Esta tarefa deverá ser realizada em trabalhos futuros antes que o sistema seja integrado ao RoPE de forma definitiva.

1.1.3 Justificativa

Um dos benefícios educacionais mais significativos oferecidos pelos brinquedos conectados à Internet é a possibilidade de individualização do conteúdo. Estando conectado à Internet o *software* do brinquedo pode coletar informações e, com isso, se adaptar às necessidades e ao progresso individual de cada criança, favorecendo o desenvolvimento de todo o seu potencial (HOLLOWAY; GREEN, 2016).

De acordo com (HOLLOWAY; GREEN, 2016) grandes líderes da indústria de brinquedos, como a Lego, Mattel e Vivid Toys, vêm investindo de forma pesada na pesquisa e desenvolvimento de brinquedos inteligentes conectados à Internet. Um exemplo desse tipo de investimento é o brinquedo “Hello Barbie” lançado pela Mattel em 2015. O brinquedo consiste em uma versão da Barbie que usa Wi-Fi para consumir serviços de reconhecimento de fala e inteligência artificial na nuvem e compreender o que a criança diz, respondendo e interagindo de forma personalizada de acordo com o que foi dito pela criança. Com empresas deste porte investindo nos brinquedos conectados, espera-se que a busca por este tipo de brinquedo cresça rapidamente daqui para frente.

Com base no que foi visto até aqui, habilitar o RoPE para acessar à Internet possibilita que trabalhos futuros expandam as suas funcionalidades e transformem-no em um brinquedo inteligente através do uso das tecnologias aqui mencionadas, como reconhecimento de fala e inteligência artificial. Tais mudanças trazem um impacto significativo e muito positivo para o RoPE, do ponto de vista educacional, comercial e científico.

No âmbito educacional porque, ao personalizar o *feedback* para cada criança, a experiência de brincar com o RoPE é enriquecida e se torna mais agradável, pois a criança pode escolher seu modo de aprendizado e aprender no seu próprio ritmo (GORDON, 2014).

No âmbito comercial porque, ao incorporar esse tipo de tecnologia o brinquedo se moderniza e passa a acompanhar a tendência de mercado, tornando-se mais atrativo tanto para os consumidores quanto para possíveis investidores. No âmbito científico porque viabiliza a coleta de dados para a realização de pesquisas em diversas áreas, incluindo a do pensamento computacional, conforme é proposto através de um estudo de caso ao final deste trabalho.

1.2 OBJETIVOS

Esta seção formaliza os objetivos do trabalho, conforme descrito a seguir.

1.2.1 Objetivo Geral

Evoluir o *hardware* e o *software* do brinquedo RoPE para conectá-lo à Internet através de uma rede Wi-Fi e permitir que seja monitorado e controlado remotamente

1.2.2 Objetivos Específicos

1. Analisar trabalhos relacionados;
2. Selecionar um módulo Wi-Fi e conectá-lo ao RoPE para realizar o acesso à Internet;
3. Implementar um módulo Wi-Fi no brinquedo RoPE;
4. Realizar um estudo de caso usando a versão evoluída do RoPE para auxiliar na avaliação do pensamento computacional;

1.3 METODOLOGIA

Este trabalho utilizou levantamentos bibliográficos de artigos científicos e análises de trabalhos similares envolvendo brinquedos inteligentes para a realização da fundamentação teórica do projeto.

A escolha do módulo Wi-Fi e do protocolo de comunicação usado na implementação do sistema foi feita levando em conta as necessidades e limitações do RoPE, as quais estão descritas e justificadas no texto dos capítulos 3.1 e 3.2.

Na etapa de projeto do sistema foram definidos os requisitos funcionais, o conjunto de comandos que poderão ser executados remotamente no RoPE e o conjunto de dados que poderão ser monitorados. Todos esses itens foram relacionados e documentados no texto do capítulo 3.3.

Na etapa de projeto do aplicativo de testes foram criados protótipos de telas demonstrando o funcionamento desejado. Foi também criado o projeto do banco de dados usado para dar suporte ao estudo de caso envolvendo a avaliação do pensamento computacional. O protótipo do banco de dados está documentado em um diagrama ER no final do capítulo 3.4.2 .

1.4 ESTRUTURA DA DISSERTAÇÃO

Este documento foi dividido em três capítulos: (i) Introdução, (ii) Fundamentação teórica e (iii) Desenvolvimento.

No Capítulo 1, Introdução, o projeto foi apresentado e contextualizado sucintamente, sendo abordados temas como problematização e solução proposta, os objetivos a serem atingidos, além da metodologia utilizada para sua execução.

No Capítulo 2, Fundamentação Teórica, apresentou-se a revisão bibliográfica sobre: brinquedos inteligentes (*Smart Toys*). Também foi feita uma breve introdução ao brinquedo RoPE, descrevendo seus recursos e como ele tem sido aplicado na educação infantil. Foram também apresentados alguns trabalhos relacionados envolvendo brinquedos inteligentes e IoT

No capítulo 3, foram abordadas todos itens relacionados à implementação do sistema, como: escolha do módulo Wi-Fi, escolha do protocolo de comunicação, levantamento dos requisitos funcionais e protótipos de tela e banco de dados do aplicativo de validação.

No quarto e último capítulo são discutidos os riscos do projeto e o que pode ser feito para mitigá-los. São apresentados também os objetivos do trabalho que já foram alcançados e também o cronograma planejado para a conclusão do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como objetivo definir o conceito de *Smart Toys* e o que torna um brinquedo inteligente. Em seguida é feita uma apresentação do RoPE, o brinquedo que será trabalhado durante essa pesquisa. Por fim, são analisados alguns trabalhos similares que poderão ajudar no desenvolvimento do projeto.

2.1 SMART TOYS

Os *Smart Toys*, também conhecidos como brinquedos tecnológicos ou brinquedos inteligentes, são um novo tipo de brinquedo que combinam elementos tecnológicos com brinquedos tradicionais (YANG; LU; WU, 2018). Em geral, são brinquedos que fazem o uso de componentes eletrônicos e microprocessadores controlados por *software* para interagir com o usuário de forma personalizada (MASCHERONI; HOLLOWAY, 2017). Segundo (RAFFERTY et al., 2017), podem ser definidos também como, um dispositivo constituído por um brinquedo físico que se conecta a um ou mais serviços de computação na nuvem através de sensores e de uma rede, a fim de aumentar a funcionalidade de um brinquedo tradicional.

Os brinquedos inteligentes podem usar os mais variados tipos de componentes e sensores, incluindo, câmeras, microfones, acelerômetros, sensores de proximidade, motores, giroscópios e dispositivos radiotransmissores, como módulos de Bluetooth e Wi-Fi. Com o auxílio destes componentes os brinquedos são capazes de processar uma maior quantidade de informação para se adaptar às ações do usuário podendo, para isso, empregar estratégias como reconhecimento de padrões em imagens e reconhecimento de fala (MASCHERONI; HOLLOWAY, 2019).

Os *Smart Toys* podem ser separados em dois grupos, conectados e não conectados. Brinquedos conectados são aqueles que podem ser controlados remotamente pela infraestrutura de rede, por exemplo, por meio de smartphones e tablets, ou que incorporam tecnologias da Internet, como reconhecimento e ativação de fala, para reagir à interação das crianças. São brinquedos que costumam coletar informações da criança através de sensores e enviá-las para plataformas baseadas em nuvem para serem processadas em tempo real (MASCHERONI; HOLLOWAY, 2017).

Os brinquedos conectados oferecem algumas vantagens se comparados aos demais. Um exemplo é o uso de inteligência artificial (IA) para melhorar a interação com o usuário. Geralmente

isso não é possível já que o uso de IA requer um grande poder computacional e os brinquedos costumam ter um *hardware* bastante limitado. Porém, nos brinquedos conectados as informações podem ser enviadas para que esse processamento seja realizado na nuvem. Em contrapartida, a conexão do brinquedo com a Internet gera um problema de segurança já que os dados do usuário podem ser capturados e/ou alguém não autorizado assumir o controle do brinquedo (VALENTE; CARDENAS, 2017).

Contudo, (MASCHERONI; HOLLOWAY, 2017) enfatiza que há uma distinção entre brinquedos inteligentes e brinquedos conectados e que é importante estar ciente desta diferença. Segundo o autor, um brinquedo pode ser inteligente, mas não ser conectado. Um exemplo é o brinquedo MOYA (Figura 2), desenvolvido por (AHN et al., 2018) para auxiliar crianças a desenvolverem suas habilidades linguísticas. Este brinquedo usa técnicas avançadas de classificação de imagem e detecção de movimento, porém todo o processamento é realizado dentro do *hardware* do próprio brinquedo, sem que haja a necessidade de enviar os dados via Internet para processamento na nuvem.



Figura 2. O brinquedo MOYA (esquerda) e seu mecanismo de detecção de movimentos

Da mesma maneira, um brinquedo pode ser conectado, mas não ser inteligente. Um exemplo é o brinquedo de programar TinkerBot³. Este brinquedo conecta-se através de *bluetooth* ao *tablet* e ao *smartphone* onde pode ser programado (Figura 3), no entanto, não fornece nenhum *feedback* personalizado ao usuário. Existem ainda os brinquedos que se enquadram em ambas as categorias, como é o caso do Smartibot⁴, que se conecta ao *smartphone* e usa o poder computacional do aparelho

³ <https://www.generationrobots.com/en/403314-tinkerbots-my-first-robot.html>

⁴ <https://thecraftyrobot.net/>

para executar um algoritmo de inteligência artificial e reconhecer animais de estimação, pessoas e veículos (Figura 4).



Figura 3. Brinquedo TinkerBot conectado a um *tablet*

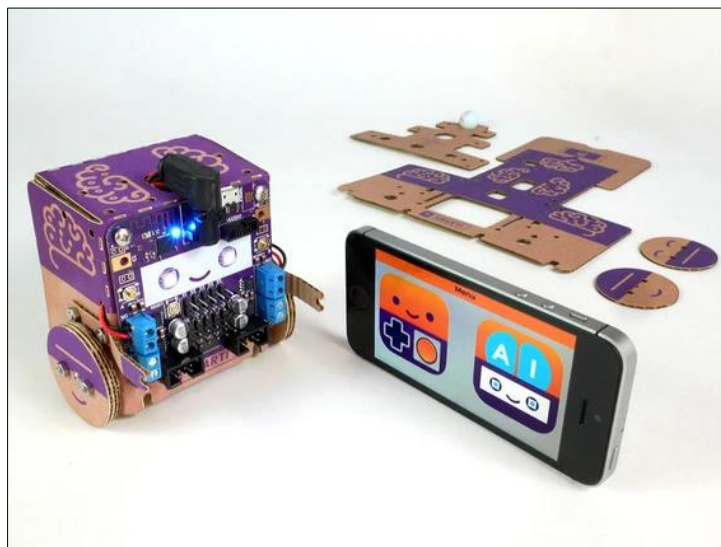


Figura 4. Brinquedo Smartibot conectado ao *smartphone*

Do ponto de vista da sua construção, os brinquedos tecnológicos são classificados em três categorias: brinquedos visuais (ou não tangíveis), brinquedos tangíveis e brinquedos híbridos (LIN, 2015). Os brinquedos visuais são restritos ao mundo virtual, são programas de *software* que executam no computador, tablet ou smartphone como, por exemplo, o ScratchJr⁵ e o LightBot⁶ (Figura 5). Por outro lado, os brinquedos tangíveis, são aqueles que permitem a interação com o usuário de forma

⁵ <https://www.scratchjr.org/>

⁶ <https://lightbot.com/>

física. Já os brinquedos híbridos são a combinação dos dois anteriores, de tal forma que, a interação do usuário com o brinquedo no mundo físico reflete também em uma interação no mundo virtual.

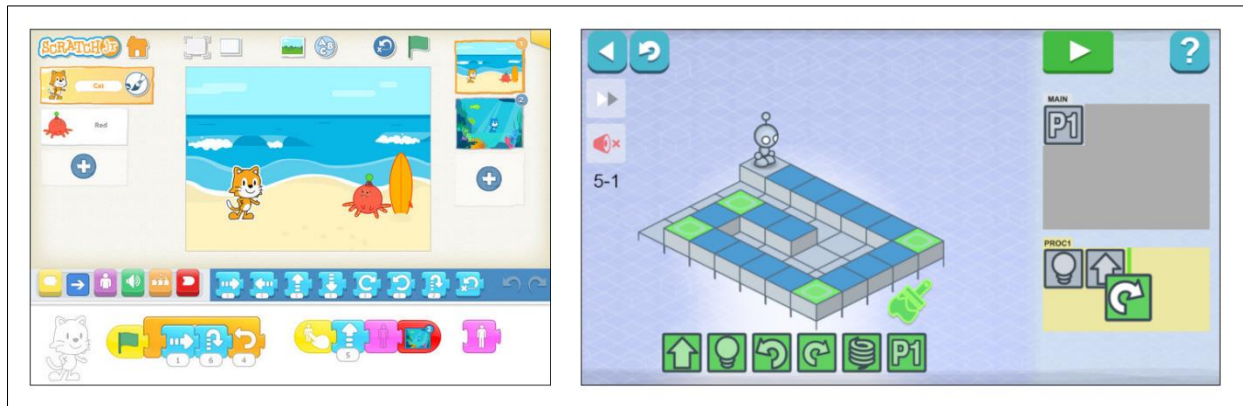


Figura 5. ScratchJr e LightBot, exemplos de *Smart Toys* não tangíveis

2.2 O BRINQUEDO ROPE

O RoPE, **Robô Programável Educacional**, é um brinquedo de programar de baixo custo que vem sendo desenvolvido dentro da Universidade do Vale do Itajaí (UNIVALI). Ele surgiu como resultado de mais de três anos de pesquisas envolvendo aproximadamente 20 pesquisadores de várias disciplinas: Educação, Ciência da Computação, Design e Engenharias Mecânica, Computação e Arquitetura (RAABE et al., 2017). Suas principais fontes de inspiração foram, a linguagem Logo (PAPERT, 1980) e o Bee-Bot, um brinquedo de programar usado pelo grupo de pesquisa em um experimento para identificar o impacto deste tipo de brinquedo na aprendizagem infantil (RAABE; VIEIRA; ROSÁRIO, 2015).

Segundo (RAABE et al., 2015), “brinquedos de programar são brinquedos que podem executar sequencias de instruções definidas por crianças. Normalmente estes brinquedos apresentam-se na forma de um veículo com rodas e assumem aparências diversas como carro, tanque, abelha, e outras figuras representativas do imaginário infantil. As instruções que executam estão relacionadas a movimentação e rotação do veículo tais com andar para frente ou para trás alguns centímetros e girar 90 graus para direita ou para esquerda”.

Seguindo essa mesma linha de raciocínio, o RoPE foi projetado para ensinar programação de forma lúdica e tangível para crianças na faixa dos cinco anos, empregando um visual amigável e

colorido. Ele permite que as crianças programem seus movimentos através de botões localizados em sua parte superior, os quais emitem um *feedback* imediato através de luzes e sons.

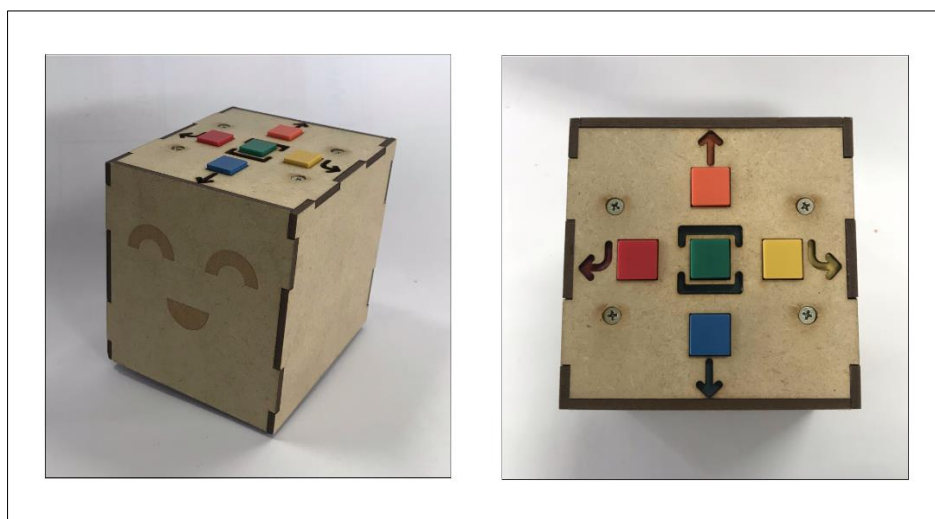


Figura 6. O brinquedo de programar RoPE e sua interface de programação

Em seu trabalho, (SARAMA; CLEMENTS, 2002) discutem o conceito de micromundos e os definem como “um ambiente computacional pequeno e coerente consistindo de ferramentas, estruturas e/ou atividades, que refletem ou incorporam um domínio da Matemática ou da Ciência, e, portanto, promove o aprendizado por meio de exploração, proposição e resolução de problema”.

A experiência de brincar com o RoPE é enriquecida através de um conjunto de tapetes pedagógicos (Figura 7) que introduzem esse conceito de micromundos para criar um contexto lúdico e interdisciplinar. Os micromundos de histórias infantis, mapas de cidade, desafios de lógica (*puzzles*), letras e muito mais podem ser usados por pais, terapeutas e professores para trabalhar os mais diversos conteúdos com as crianças, como: lógica, matemática e lateralidade.

Os tapetes pedagógicos são baratos e fáceis de produzir, podendo ser confeccionados pelos próprios professores. Um exemplo disso é o Núcleo Educacional Bom Sucesso, de Balneário Camboriú, SC. Em uma entrevista com (SANTOS, 2019), a supervisora relata que, desde que o RoPE foi adquirido em 2017 a equipe da escola já desenvolveu cerca de 18 tapetes, integrando o brinquedo em todas as disciplinas. O destaque ficou por conta de um tapete elaborado para trabalhar o comportamento das crianças. Segundo a gestora da escola “Ao usar o RoPE no tapete de boas maneiras, por exemplo, a criança aprende melhor o que pode ou não fazer, diferente de só colar um cartaz na parede”.

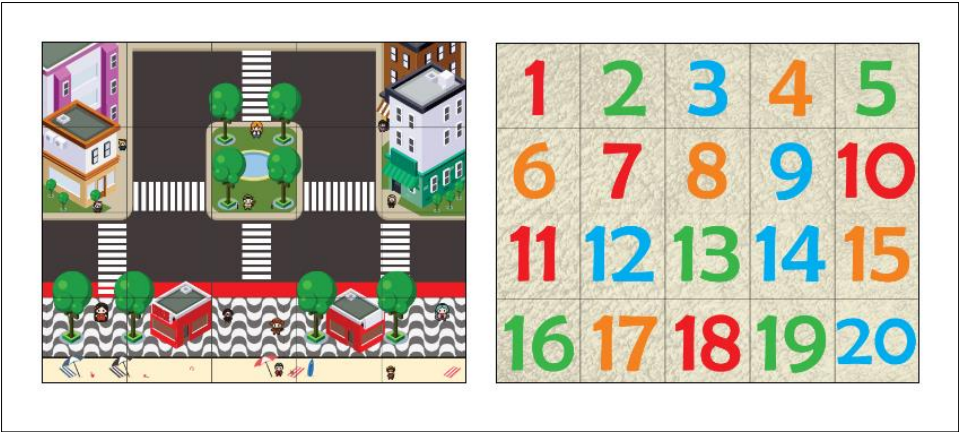


Figura 7. Tapetes pedagógicos do RoPE

Do ponto de vista técnico “[...] os pesquisadores que construíram o RoPE priorizaram decisões que pudessem reduzir o custo e tornar o brinquedo aderente a realidade dos núcleos de educação infantil brasileiros” (RAABE et al., 2017). Por isso, o RoPE é composto por componentes eletrônicos baratos e que possuem uma boa disponibilidade no mercado. A Tabela 1 mostra uma relação dos principais componentes eletrônicos do RoPE.





Imagem	Componente	Imagem	Componente
	2x Motor de passo Modelo 28byj-48		1x Microcontrolador ATmega328P-AU
	1x Driver para motor de passo Modelo ULN2803AFWG		1x Buzzer Modelo KC-1206

Tabela 1. Principais componentes do RoPE

O microcontrolador ATmega328P presente no RoPE é o mesmo chip embarcado na plataforma de desenvolvimento Arduino Nano e que é normalmente utilizada no desenvolvimento de protótipos por conta de seus recursos e facilidade de programação. Esse microcontrolador pode ser

programado usando a linguagem C/C++ e possui um conjunto vasto de bibliotecas para trabalhar com os mais variados tipos de componentes, incluindo: motores, sensores, *displays* de LCD, entre outros.

Embora o RoPE faça o uso de componentes eletrônicos, ele não pode ser considerado um *Smart Toy*, pois ele não é capaz de reagir e se adaptar de forma personalizada às interações dos usuários. Também não pode ser considerado um brinquedo conectado, pois não possui acesso à Internet. Espera-se que ao final deste trabalho as modificações realizadas no *hardware* e *software* do RoPE transformem-no em um brinquedo conectado e viabilizem que trabalhos futuros façam dele brinquedo inteligente.

2.3 TRABALHOS RELACIONADOS

Esta seção tem como objetivo analisar alguns trabalhos similares que podem contribuir com o desenvolvimento do sistema e com os objetivos propostos na pesquisa.

2.3.1 Tactic-Kibo

O Kibo é um brinquedo de programar desenvolvido pelo grupo de pesquisa DevTec da universidade de Tufts University e comercializado pela KinderLab Robotics. Ele foi projetado para trabalhar conceitos fundamentais de engenharia e programação com crianças de 4 a 7 anos (ELKIN; SULLIVAN; BERS, 2016).

Ao contrário de outros brinquedos de programar como o RoPE, o Kibo não possui botões em sua estrutura. Ao invés disso, os algoritmos são desenvolvidos usando blocos de madeira que representam instruções e devem ser conectados em sequência um ao lado do outro. O Kibo permite criar programas mais complexos, pois além das instruções básicas como mover-se e girar, ele suporta a programação de laços de repetição.

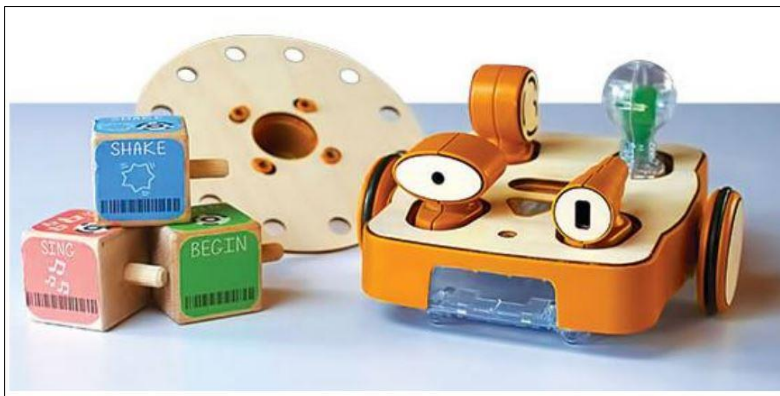


Figura 8. O brinquedo Kibo

Além dos blocos de programar, o Kibo possui um conjunto de componentes robóticos que podem ser acoplados a ele para trabalhar os conceitos de engenharia. Dentre os componentes estão, rodas, motores, luzes e vários tipos de sensores. Além disso, o kit também contém adesivos que permitem às crianças personalizá-lo (SULLIVAN et al., 2015).

Em seu trabalho, (EMILY RELKIN et al., 2018), usou o Kibo como ferramenta para a aplicação de um conjunto de métricas de avaliação do pensamento computacional. O projeto foi batizado de Tactic-Kibo (Tufts Assessment of Computational Thinking In Children – Kibo version) e apresentou resultados satisfatórios, sugerindo que é sim possível usar um brinquedo de programar para realizar esse tipo de avaliação em crianças. O trabalho realizado com o Tactic-Kibo é relevante para este projeto porque ele serve como uma referência para identificar quais são as variáveis que devem ser coletadas pelo RoPE para possibilitar a avaliação do pensamento computacional no estudo de caso que será realizado.

3 DESENVOLVIMENTO

Este capítulo detalha a solução proposta para transformar o RoPE em um *Smart Toy*. A solução consiste em fazer alterações de *hardware* e *software* no brinquedo a fim de: (i) conectá-lo com a Internet através da rede Wi-Fi, (ii) permitir que o brinquedo seja controlado remotamente e (iii) permitir que os dados do brinquedo sejam coletados para a realização de pesquisas e experimentos.

Na parte de *hardware* a solução consiste em: (i) integrar um módulo Wi-Fi ao RoPE. Na Parte de *software* a solução consiste em: (i) escolher um protocolo de comunicação, (ii) modificar o *firmware* para executar comandos enviados remotamente (iii) modificar o *firmware* para coletar dados relevantes do funcionamento do brinquedo e enviá-los através da Internet.

3.1 MÓDULO WIFI ESP-01

Um dos objetivos deste trabalho é adaptar o RoPE para que ele possa se conectar à internet através de uma rede WiFi e interagir com outros dispositivos e aplicativos remotamente. Para viabilizar essa conexão é necessário fazer alterações no *hardware* do brinquedo, pois o microcontrolador ATmega328P (família AVR da Atmel), usado atualmente, não oferece essa funcionalidade.

Uma das possibilidades seria substituir o ATmega328P por um outro chip que possua essa função, no entanto, isso não é viável no momento, pois envolve ter que recodificar todo o firmware do RoPE para o novo microcontrolador, tarefa essa que poderá ser realizada em um trabalho futuro. Por esse motivo, faz-se necessário usar no projeto um módulo que possa ser acoplado ao brinquedo e se comunicar com o microcontrolador principal via comunicação serial UART (*Universal Asynchronous Receiver-Transmitter*).

Para melhor compreensão, é importante salientar que o termo módulo é utilizado para descrever uma placa de circuito impresso (PCI) que integra um chip principal aos seus periféricos como, por exemplo: cristal, memória flash, antena e outros componentes demandados. O módulo escolhido para o desenvolvimento do Smart RoPE foi o ESP-01, o qual integra o chip ESP8266⁷ da

⁷ Após uma revisão de arquitetura o ESP8266 passou a se chamar ESP8266EX. No entanto, ambos os nomes se referem ao mesmo chip e possuem os mesmos recursos, sendo mais comum encontrar o nome ESP8266 na literatura

Espressif (MICROSHIP, 2020a) e é amplamente utilizado em projetos IoT em conjunto com o Arduino (baseadas em microcontroladores da família AVR da Atmel). A Figura 9 ilustra um exemplar do ESP-01 fabricado pela AIThinker.

O ESP8266 integrado no ESP-01 possui a funcionalidade de comunicação serial necessária ao projeto, além de vários outros recursos úteis que serão abordados nesse capítulo. Além disso, dentro do contexto de *Smart Toys* um módulo com o ESP8266 já foi empregado com sucesso na construção do Dolphin Sam (COLOMBO et al., 2016), um brinquedo concebido com o intuito de auxiliar no desenvolvimento de crianças com deficiência cognitiva.

De acordo com (RAABE et al., 2017) “[...] os pesquisadores que construíram o RoPE priorizaram decisões que pudessem reduzir o custo e tornar o brinquedo aderente a realidade dos núcleos de educação infantil brasileiros”. Por esse motivo, é importante manter a proposta original dos idealizadores do RoPE e optar por uma solução de baixo custo. O ESP-01 se enquadra nesse requisito pois, segundo (MEHTA, 2015), ele pode ser encontrado por menos de \$5,00 enquanto outros módulos chegam a custar entre \$30,00 e \$60,00.

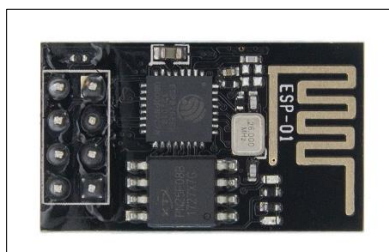


Figura 9. Módulo ESP-01

Outra principal preocupação deste trabalho é manter um baixo consumo de energia. O RoPE é um brinquedo amplamente utilizado em sala de aula em atividades que duram de 30 a 60 minutos em uso contínuo, portanto, é necessário garantir a autonomia da bateria durante este período. Atualmente o RoPE consegue manter essa autonomia acionando a função *Power-Down* presente no chip ATmega328P (MICROSHIP, 2020b), nessa função o chip desliga a maior parte de seus recursos e aguarda até que seja recebida uma interrupção externa para então ligar e reiniciar sua atividade. No caso do RoPE essa interrupção ocorre quando um de seus botões é pressionado.

De forma similar, o ESP8266 também oferece funções de economia de energia, que conforme apresentado no trabalho de (MEHTA, 2015) podem reduzir o consumo de corrente para valores

abaixo de 0.012mA. Este consumo é muito mais baixo do que do próprio ATmega328P, que segundo o datasheet da (MICROSHIP, 2020b) consome entre 1.2mA a 2.7mA em modo de espera, quando conectado a uma fonte de energia de 5V e ativado o cristal interno de 8MHz, configuração usada pelo RoPE. As capacidades de economia de energia do ESP8266 já foram testadas por (MESQUITA et al., 2018) e os experimentos demonstraram que sob certas condições ele pode ser capaz de funcionar por até 4 dias consecutivos usando uma bateria de 1000mAh. Sendo assim, no quesito de consumo energia o ESP8266 se mostra uma ótima opção e deve atender à necessidade do projeto.

A partir do ano de 2017, em parceria com a Secretaria de Educação de Balneário Camboriú, pelo menos 30 unidades do RoPE foram construídas e distribuídas para vários núcleos de educação infantil da região, (RAABE et al., 2017). Desde então, com seu uso extensivo em sala de aula, os professores identificaram uma série de falhas de *software* que necessitavam de correção para que as unidades pudessem continuar operando. Ao identificar essas falhas o procedimento adotado para correção consistia em enviar as unidades defeituosas de volta ao setor de manufatura, aguardar a gravação de um *firmware* de correção e, após, reenviar as unidades para o núcleo de educação. Esse processo causa alguns problemas: (i) gera um custo de logística; (ii) aumenta o tempo de espera para a correção e; (iii) inviabiliza que a atualização seja aplicada nas demais unidades antes que os defeitos ocorram.

A partir da implementação do ESP8266 estes problemas poderiam ser solucionados com a inclusão do recurso de atualização *Over The Air* (OTA), o qual permite baixar e gravar uma nova versão do *firmware* a partir da nuvem. O uso desse recurso no ESP8266 é relativamente simples, pois a comunidade desenvolve e mantém um conjunto de bibliotecas no Github que implementam essa funcionalidade⁸. Um exemplo de código para atualização OTA pode ser visto na Figura 10. Outra facilidade disponibilizada pela atualização OTA é a gravação simultânea de firmware em múltiplas unidades, apropriado para o processo de manufatura em grande escala.

⁸ <https://github.com/esp8266/Arduino>


```

ESP8266WiFiMulti WiFiMulti;

void setup() {
  WiFi.mode(WIFI_STA);
  WiFiMulti.addAP("REDE-WIFI", "SENHA-WIFI");
}

void loop() {
  if ((WiFiMulti.run() == WL_CONNECTED)) {
    ESPhttpUpdate.onStart([]() {
      Serial.println("Iniciando atualização do firmware");
    });

    ESPhttpUpdate.onProgress([](int progress, int total) {
      Serial.printf("Instalando firmware, progresso: %f", (progress / (total / 100)));
    });

    ESPhttpUpdate.onEnd([]() {
      Serial.println("Atualização do firmware finalizada");
    });

    WiFiClient client;
    t_httpUpdate_return ret = ESPhttpUpdate.update(client, "http://smartfun.com.br/rope/firmware/latest");

    if (ret == HTTP_UPDATE_FAILED) {
      Serial.printf("Erro: ao atualizar: %s", ESPhttpUpdate.getLastErrorMessage().c_str());
    }
  }
}

```

Figura 10. Exemplo de código para atualização OTA

Em sua programação atual o RoPE possui alguns parâmetros de funcionamento que precisam ser ajustados de forma única para cada unidade produzida, devido à sutis diferenças existentes em seus componentes. Um destes componentes é o *buzzer* que, dependendo da unidade, não é capaz de reproduzir com clareza e intensidade suficiente as notas que compõem as melodias do RoPE. Isso torna necessário usar uma combinação de notas musicais diferentes para cada unidade produzida. Outro componente problemático é o motor de passo modelo 28byj-48 que, por ser de baixo custo, apresenta uma folga em seu eixo, gerando uma imprecisão quando o RoPE realiza um movimento de giro. Para corrigir isso, existe um parâmetro dentro do *firmware* que modifica o modo de acionamento dos motores a fim de compensar essa diferença.

O maior problema com a configuração destes parâmetros é que eles precisam ser alterados diretamente no código fonte do brinquedo e requer que o *firmware* seja regravado em cada unidade. Este problema pode ser resolvido usando o recurso de *SoftAP* do ESP8266 (ESPRESSIF, 2020). Com este recurso ativo é criada uma rede WiFi que permite conectar-se ao módulo e acessar via navegador um conjunto de páginas *Web* hospedadas dentro do mesmo. Os parâmetros podem então ser ajustados através desta interface visual e são armazenados em uma memória interna não volátil (*EEPROM* - *Electrically-Erasable Programmable Read-Only Memory*) para serem lidos durante a inicialização

do *firmware*. Assim, é possível manter um único *firmware* para todas as unidades e efetuar o ajuste posteriormente de forma independente, sem necessidade de regravação.

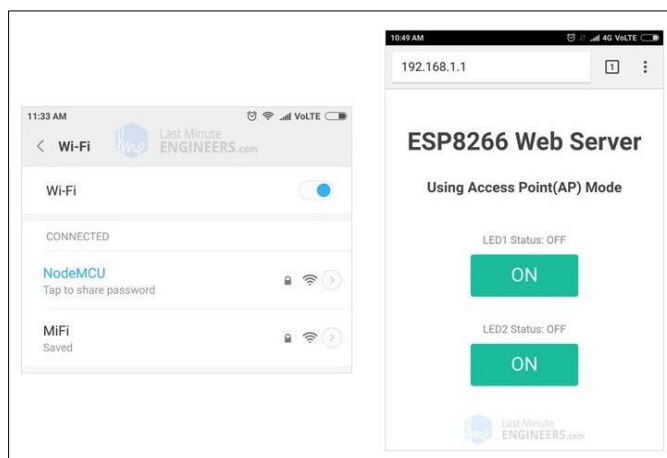


Figura 11. Rede SofAP e página web hospedada no ESP8266

A Figura 11 demonstra em exemplo de página de configuração hospedada no ESP8266 e uma rede denominada “NodeMCU” usada para a conexão. Já na Figura 12 é possível visualizar um trecho de código simplificado que pode ser usado para criar uma página semelhante no RoPE e gravar os parâmetros na *EEPROM*.

```
ESP8266WebServer server(80);

void homePage() {
    String html = "<html><head><title>RoPE</title></head><body></body></html>";
    server.send(200, "text/html", html);
}

void saveConfig(){
    if (server.arg("motor") != "") {
        EEPROM.write(0, toInt(server.arg("motor"))); // Salva o valor do parâmetro
        EEPROM.commit();
    }
}

void setup() {
    IPAddress ipAndGateway(192, 168, 1, 1);
    IPAddress subnet(255, 255, 255, 0);

    WiFi.softAP("NodeMCU", "SENHA");
    WiFi.softAPConfig(ipAndGateway, ipAndGateway, subnet); // Inicializa a rede SoftAP

    server.on("/", homePage);
    server.on("/save", saveConfig);
    server.begin();

    int motor = EEPROM.read(0); // Lê o valor do parâmetro salvo
}

void loop() {
    server.handleClient();
}
```

Figura 12. Código para configuração do RoPE

Por fim, além dos recursos já apresentados, o ESP8266 conta com um hardware de maior desempenho, em comparação com o embarcado atualmente no RoPE, oferecendo um clock de CPU mais elevado, maior capacidade de memória RAM (*Random-access Memory*) e memória flash. Neste trabalho o ESP-01 será usado apenas em conjunto com o chip ATmega328P, mas em trabalhos futuros o microcontrolador atual poderá ser totalmente substituído pelo ESP8266, ampliando as capacidades do RoPE e possibilitando a execução de tarefas relativamente mais complexas e que demandem maior desempenho. A Tabela 2 mostra uma comparação entre o hardware atual do RoPE e do ESP-01, dados extraídos dos *datasheets* oficiais (MICROSHIP, 2020b) e (MICROSHIP, 2020a) respectivamente.

	RoPE (ATmega328p)	ESP-01 (ESP8266)
CPU	Padrão: 16 MHz ⁹ Máximo: 20 MHz	Padrão: 80 MHz Máximo: 160 MHz
RAM	2 kB	36 kB
Flash	32 kB	Padrão: 1 MB ¹⁰ Máximo: 16 MB

Tabela 2. Comparação de hardware entre o ATmega328P e o ESP8266

3.2 PROTOCOLO MQTT

Ao trabalhar com IoT uma das etapas do desenvolvimento é escolher o protocolo de comunicação que será adotado para a troca de mensagens entre os dispositivos na camada de aplicação. Para este projeto, foi escolhido o protocolo MQTT, desenvolvido pela IBM. Neste capítulo é apresentado o funcionamento deste protocolo e os motivos que levaram à sua escolha.

O MQTT é um baseado na arquitetura *publisher/subscriber* (editor/assinante) que permite a comunicação entre dispositivos através de uma rede sem fio (KASHYAP; SHARMA; GUPTA, 2018). Nesse modelo, os *editores* são os dispositivos que publicam informações e os *assinantes* são os dispositivos que as consomem. A troca de informações é coordenada por um servidor denominado *broker*, que recebe as mensagens dos *editores* e as encaminha para os *assinantes* correspondentes.

⁹ No RoPE, a frequência foi reduzida para 8 MHz a fim de economizar energia

¹⁰ O módulo fabricado pela AIThinker vem com 1 MB, módulos genéricos costumam variar entre 512 kB e 4 MB

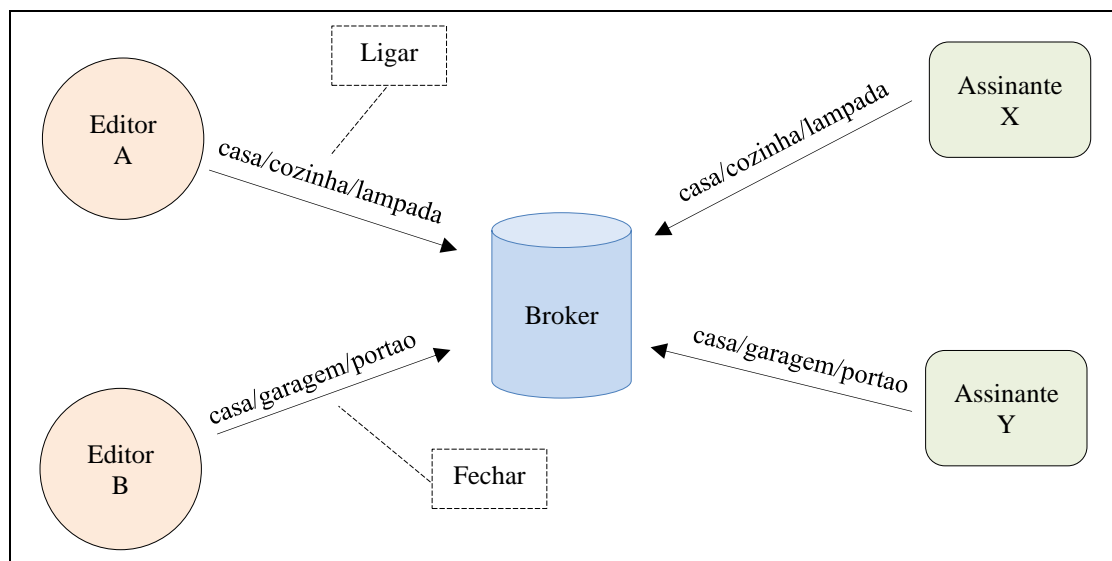


Figura 13. Arquitetura do MQTT

O encaminhamento das mensagens é realizado adotando um conceito denominado *topic* (tema), que nada mais é do que uma cadeia de caracteres contida no pacote MQTT, similar a uma URL, e que funciona como um canal de comunicação. Quando desejam transmitir uma informação os *editores* devem enviar uma mensagem do tipo *publish* ao *broker* com o conteúdo da mensagem (*payload*) e o *tema* no qual desejam publicar. Já os *assinantes* devem enviar uma mensagem do tipo *subscribe* contendo o *tema* no qual desejam se inscrever, assim receberão o conteúdo que os *editores* tiverem publicado neste mesmo *tema*.

Para uma melhor compreensão, o diagrama da Figura 13 ilustra uma aplicação hipotética em MQTT para automação residencial. Nesta aplicação o *editor A* publica uma mensagem no *tema* “casa/cozinha/lampada” para ligar a lâmpada da cozinha. O *assinante X*, que se inscreveu neste mesmo *tema*, recebe esta mensagem e atua para executar a ação, ligando a lâmpada. O *assinante Y*, por sua vez, não recebe a mensagem, pois está inscrito em outro *tema*. É importante salientar que um dispositivo pode se inscrever e/ou publicar em vários *temas* ao mesmo tempo, além disso, pode atuar simultaneamente como *editor* e *assinante* (SAHADEVAN et al., 2017).

De acordo com (MANANDHAR, 2017), ao empregar este modelo o MQTT permite que a comunicação ocorra de forma assíncrona, sem necessitar que *editor* e *assinante* estejam conectados no mesmo instante ou tenham conhecimento da existência um do outro. A possibilidade de funcionar de forma assíncrona é importante para este trabalho porque viabiliza a comunicação de dispositivos remotos com o RoPE em ambientes onde a infraestrutura de rede é precária e apresenta instabilidades,

como é o caso de algumas escolas públicas. Ainda, segundo o autor, a intermediação do *broker* elimina a necessidade de os dispositivos conhecerem os endereços IP uns dos outros, permitindo que a comunicação ocorra mesmo em redes que mascaram os endereços IPs usando NAT.

Durante o encaminhamento das mensagens o MQTT permite escolher entre 3 níveis de QoS, conforme mencionado por (SHINHO LEE et al., 2013). No primeiro nível (level 0), as mensagens são transmitidas apenas uma vez e não há verificação de entrega, havendo possibilidade de perda. No segundo nível (level 1), as mensagens são enviadas no mínimo uma vez e há verificação de entrega, no entanto, pode haver duplicação caso o pacote contendo a confirmação do recebimento se perca. Já no terceiro nível (level 2), o *broker* usa uma verificação de 4 vias para garantir que as mensagens sejam entregues apenas uma vez, ao custo de uma maior latência na comunicação.

No contexto do RoPE a confiabilidade na entrega das mensagens é necessária. Considere um cenário no qual um dispositivo se conecta remotamente ao RoPE com o intuito de lhe enviar a seguinte sequência de comandos de movimento: 1) andar para frente; 2) girar à esquerda; 3) andar para trás. Se não houver um mecanismo que garanta a entrega das mensagens o RoPE irá executar uma sequência que não corresponde ao programa original, podendo haver falta de comandos ou comandos duplicados. O QoS level 2 do MQTT resolve este problema. No entanto, (HWANG; PARK; SHON, 2016) menciona que o MQTT não garante a ordem das mensagens. O autor sugere uma solução que consiste em acrescentar ao *payload* um campo que indica a sequência da mensagem. Esta estratégia, ou outra similar, poderá ser usada no trabalho para resolver o problema da ordenação.

A complexidade é outro ponto relevante para o projeto, pois interfere no tempo necessário para o desenvolvimento, além de impactar na manutenção do *firmware* conforme o RoPE for evoluindo. Neste quesito, o MQTT também se mostra uma ótima opção, pois segundo (ELHADI et al., 2018) é um protocolo fácil de ser implementado e independentemente da linguagem de programação. É possível encontrar bibliotecas de MQTT em diversas linguagens, incluindo C e PHP que serão usadas nesse trabalho. A implementação em PHP será útil para desenvolver uma aplicação de estudo de caso ao final do trabalho. Já a implementação em C será usada no ESP8266 e conta com mais de 10 opções de bibliotecas, conforme apontado por (OLIVEIRA et al., 2018).

Um dos objetivos do trabalho é permitir que o RoPE tenha seus dados coletados e seja controlado remotamente através da Internet por outros dispositivos, porém é preciso garantir que somente dispositivos autorizados sejam capazes de obter esse acesso. Para auxiliar nessa tarefa o

MQTT possui um mecanismo de segurança baseado em usuário e senha (SONI; MAKWANA, 2017) com suporte a criptografia TLS dependendo do *broker* utilizado. Nesse modelo de segurança, os usuários precisam estar registrados no *broker* e os dispositivos devem enviar as suas credenciais na fase de estabelecimento da conexão.

O mecanismo de autenticação via usuário senha é o suficiente para o desenvolvimento deste projeto, no entanto, apresenta alguns problemas. Segundo (BHAWIYUGA; DATA; WARDA, 2017), a necessidade de envio das credenciais a cada conexão facilita a sua captura por possíveis *sniffers*¹¹ conectados à rede. Ainda de acordo com o autor, pelo fato de as credenciais nunca expirarem, uma vez que tenham sido capturadas elas podem ser usadas indefinidamente para ataques até que sejam alteradas no *broker*. A literatura apresenta alguns trabalhos que podem ser usados como referência para melhorar a camada de segurança do RoPE em trabalhos futuros: (SINGH et al., 2015), (SHIN et al., 2016), (PERRONE et al., 2017) e (PATEL; DOSHI, 2020).

Como já foi mencionado, ao usar o protocolo MQTT é necessário também a configuração de um *broker* para intermediar a comunicação entre os dispositivos. Dentre as diversas implementações de *broker* disponíveis, optou-se por usar nesse trabalho o *Mosquitto*, um *broker* de código aberto, escrito na linguagem C e mantido pela *Eclipse Foundation*. O *Mosquitto* se mostra uma boa escolha pois possui bom desempenho em relação a uso de CPU e memória (TORRES; ROCHA; DE SOUZA, 2016), além de suportar todos os níveis de QoS e a possibilidade de criação dinâmica de *temas* (SONI; MAKWANA, 2017).

Outro fator que pesou na escolha desse *broker* é o fato de ele já ter sido usado com sucesso em um trabalho similar. Em seu trabalho, (MARTINS, 2019) desenvolveu um sistema de automação residencial utilizando os mesmos componentes deste projeto, um módulo com o ESP8266 em conjunto com o protocolo MQTT e o *Mosquitto* atuando como *broker*.

¹¹ *Sniffer* é um software ou hardware que permite ao usuário “farejar” ou monitorar o tráfego da rede

3.3 ARQUITETURA DO PROJETO

Existem três objetivos específicos que devem ser alcançados neste trabalho para transformar o RoPE em um *smart toy*: (i) conectar o RoPE à Internet através de uma rede Wi-Fi; (ii) permitir que o RoPE receba comandos e seja controlado remotamente através da Internet; (iii) permitir que o RoPE seja monitorado e tenha seus dados coletados durante o seu uso. Neste capítulo será apresentada a arquitetura que foi elaborada para alcançar esses objetivos.

3.3.1 Requisitos funcionais

O primeiro passo para elaborar a arquitetura do sistema foi fazer um levantamento dos requisitos funcionais do projeto, os quais relacionam o conjunto de comandos que poderão ser executados remotamente no RoPE e conjunto de dados que poderão ser monitorados. Estes requisitos estão enumerados na Tabela 3 e servirão como norte para o desenvolvimento.

Requisitos funcionais do Smart RoPE	
RF1	Os dispositivos externos devem conseguir mover o RoPE para frente e para trás
RF2	Os dispositivos externos devem conseguir girar o RoPE em sentido horário e anti-horário
RF3	Os dispositivos externos devem conseguir ligar e desligar o som do RoPE
RF4	Os dispositivos externos devem conseguir alterar a tonalidade do som do RoPE
RF5	Os dispositivos externos devem conseguir reproduzir sons no RoPE
RF6	Os dispositivos externos devem conseguir ligar e desligar os LEDs do RoPE
RF7	Os dispositivos externos devem conseguir inserir instruções na memória do RoPE
RF8	Os dispositivos externos devem conseguir remover instruções da memória do RoPE
RF9	Os dispositivos externos devem conseguir alterar o tamanho da memória de instruções do RoPE
RF10	Os dispositivos externos devem conseguir limpar completamente a memória de instruções do RoPE
RF11	Os dispositivos externos devem conseguir executar um programa armazenado na memória do RoPE
RF12	Os dispositivos externos devem conseguir interromper um programa que esteja sendo executado
RF13	Os dispositivos externos devem conseguir instruir o RoPE a aguardar um intervalo de tempo
RF14	Os dispositivos externos devem conseguir simular o pressionamento dos botões do RoPE
RF15	Os dispositivos externos devem ser notificados quando o RoPE se mover
RF16	Os dispositivos externos devem ser notificados quando o RoPE girar
RF17	Os dispositivos externos devem ser notificados quando o nível de bateria do RoPE mudar
RF18	Os dispositivos externos devem ser notificados quando o som do RoPE for ligado ou desligado
RF19	Os dispositivos externos devem ser notificados quando a tonalidade dos sons do RoPE for alterada
RF20	Os dispositivos externos devem ser notificados quando um som for reproduzido no RoPE
RF21	Os dispositivos externos devem ser notificados quando um LED do RoPE for ligado ou desligado

Requisitos funcionais do Smart RoPE	
RF22	Os dispositivos externos devem ser notificados quando uma instrução for inserida na memória do RoPE
RF23	Os dispositivos externos devem ser notificados quando uma instrução for removida da memória do RoPE
RF24	Os dispositivos externos devem ser notificados quando a memória do RoPE for redimensionada
RF25	Os dispositivos externos devem ser notificados quando a memória do RoPE for apagada
RF26	Os dispositivos externos devem ser notificados quando o RoPE iniciar a execução de um programa
RF27	Os dispositivos externos devem ser notificados do progresso da execução de um programa
RF28	Os dispositivos externos devem ser notificados quando o RoPE finalizar a execução de um programa
RF29	Os dispositivos externos devem ser notificados quando um botão do RoPE for pressionado
RF30	Os dispositivos externos devem ser notificados do estado inicial do RoPE quando ele for ligado
RF31	Os dispositivos externos devem conseguir obter uma lista com o número de série de todas as unidades do RoPE conectadas ao <i>broker</i>
RF32	Os dispositivos externos devem conseguir registrar um <i>tema</i> personalizado e requisitar que o RoPE publique seus eventos nesse <i>tema</i>
RF33	Os dispositivos externos devem conseguir remover um <i>tema</i> personalizado registrado anteriormente para que o RoPE pare de publicar os eventos nesse <i>tema</i>

Tabela 3. Requisitos funcionais do Smart Rope

3.3.2 Conexão Wi-Fi

Para cumprir o objetivo de conectar o RoPE à Internet é necessário acoplar o módulo ESP-01 à placa do brinquedo. A conexão é bem simples, como pode ser visto na Figura 14, pois a PCI do RoPE já expõe um conjunto de pinos para a conexão com módulos externos usando a interface serial (UART).

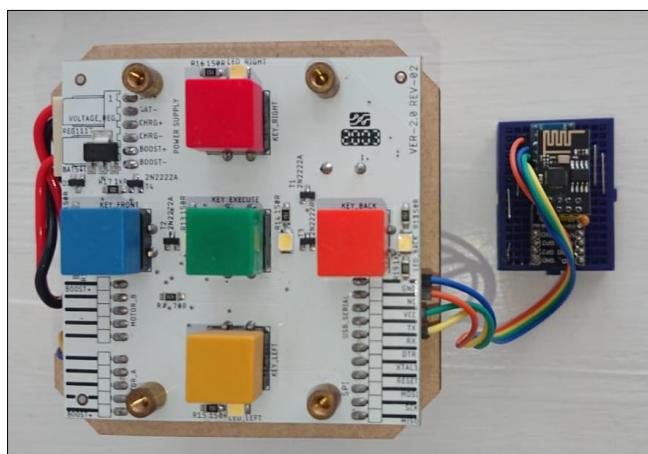


Figura 14. Conexão do ESP-01 com o RoPE

Após a ligação do módulo, a segunda etapa é realizar a configuração da rede Wi-Fi. Esse processo é realizado programaticamente fornecendo o SSID da rede e a senha da rede que se deseja

conectar, conforme ilustrado na Figura 15. Uma vez conectado à rede Wi-Fi o ESP-01 já estará pronto para transmitir e receber dados através da Internet.

```
void setup(void) {
  WiFi.begin("SSID", "SENHA");

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
  }
}
```

Figura 15. Trecho de código para configuração da rede Wi-Fi

3.3.3 Roteamento das mensagens

O Smart Rope funcionará usando o protocolo MQTT, abordado no capítulo 3.2. Conforme é estabelecido nesse protocolo, para que ocorra a comunicação entre o RoPE e os demais dispositivos é necessário que sejam definidos os *temas* nos quais eles deverão publicar e se inscrever. Para esse projeto foram definidos três temas, os quais estão relacionados na Tabela 4 juntamente com seu propósito.

Tema	Descrição
rope/list	Tema destinado ao registro das unidades. Através deste tema os dispositivos externos podem obter o número de série das unidades conectadas (RF31)
rope/{serial}/control	Tema destinado às mensagens para o controle remoto do RoPE. Através deste tema os dispositivos externos podem enviar comandos como: mover, girar ou ligar um LED
rope/{serial}/events	Tema destinado à publicação de eventos ocorridos no RoPE. Através deste tema os dispositivos externos podem obter informações como: carga da bateria, botões pressionados e alterações na memória

Tabela 4. Temas do Smart RoPE

No recebimento de comandos para controle remoto do RoPE as mensagens são originadas no dispositivo externo e enviadas ao *broker*, que as encaminha para o RoPE. Estas mensagens são recebidas pelo módulo ESP-01 que as retransmite ao microcontrolador ATmega328P através da interface serial (*UART*) para serem então processadas. O caminho percorrido pelas mensagens pode ser visto de maneira simplificada a seguir na Figura 16.

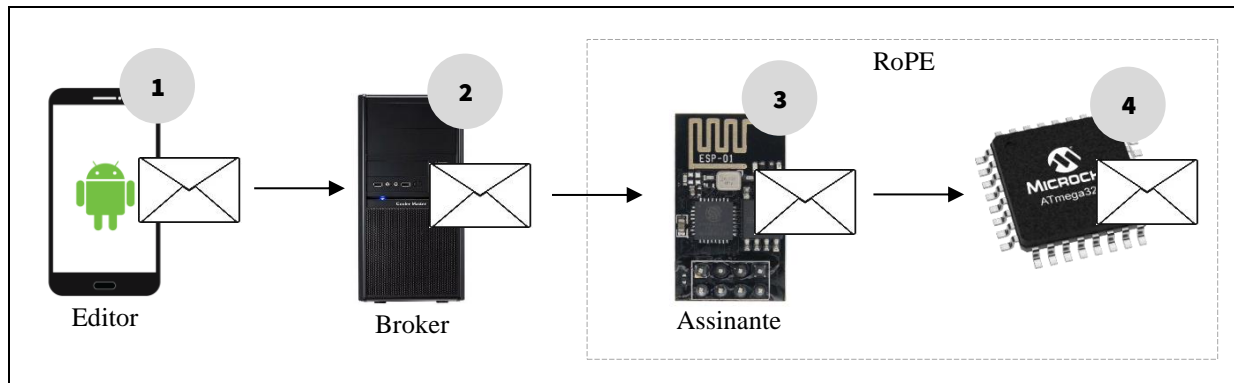


Figura 16. Roteamento das mensagens de controle remoto

Para que as mensagens percorram esse caminho ocorrer a seguinte sequência de eventos. Primeiro o RoPE se conecta ao *broker* e publica uma mensagem contendo o seu número de série no *tema* “rope/list”. Ao mesmo tempo, o dispositivo que deseja enviar comandos ao RoPE se conecta ao *broker* e se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o número de série do RoPE para o dispositivo, que pode armazená-lo para uso futuro. Em seguida, o dispositivo publica uma mensagem contendo o comando que deve ser executado pelo RoPE no *tema* “rope/{serial}/control”, onde “{serial}” é o número de série do RoPE. Ao mesmo tempo, o RoPE se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o comando ao RoPE, que pode então executá-lo. Para uma melhor compreensão, a Figura 17 apresenta um diagrama de sequência exemplificando esse processo.

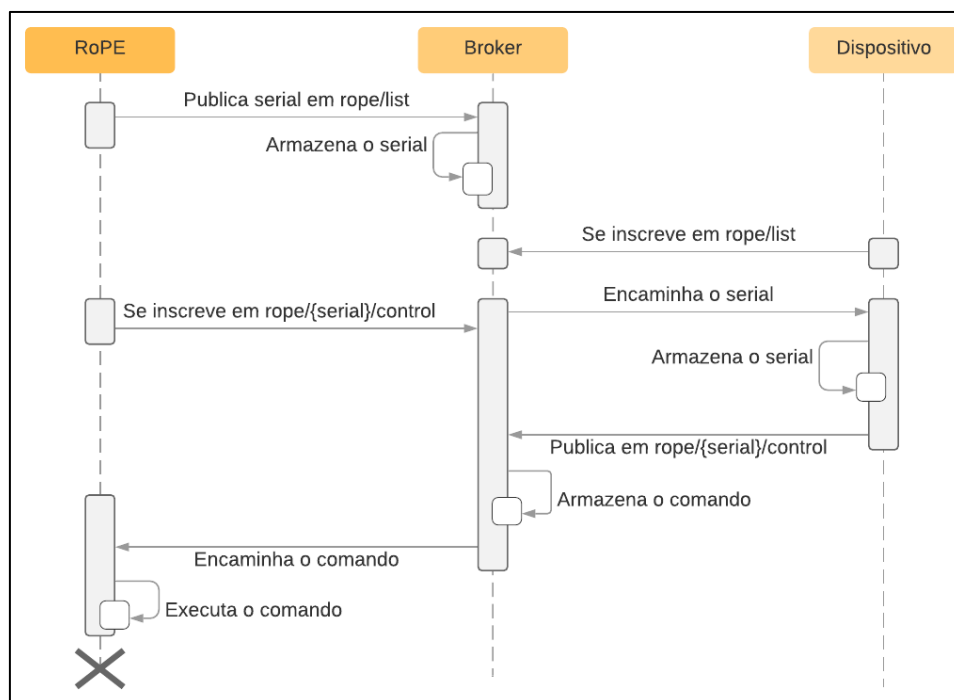


Figura 17. Sequência para envio de comandos para o RoPE

Para o monitoramento e coleta de dados do RoPE, foi elaborado um mecanismo baseado em eventos, no qual, toda vez que um determinado evento ocorre no RoPE uma mensagem contendo informações relevantes sobre o acontecimento é publicada no *broker*. Os eventos são gerados tanto para ações executadas por usuários como para ações executadas remotamente por dispositivos. Por exemplo, o pressionamento de um botão, pode ser físico ou pode ser simulado por um dispositivo através de um comando remoto. Em ambos os casos será gerado um evento de pressionamento de botão. Esse sistema permite que um dispositivo monitore as ações que estão sendo executados no RoPE por intermédio de outro dispositivo.

No envio de eventos aos dispositivos, o caminho é o inverso do anterior. As mensagens são originadas dentro do RoPE pelo microcontrolador ATmega328P, que as transmite ao módulo ESP-01 através da interface serial (*UART*). No módulo ESP-01 as mensagens são enviadas para o *broker*, que por sua vez as encaminha para o dispositivo a fim de serem então processadas. O caminho percorrido pelas mensagens pode ser visto de maneira simplificada a seguir na Figura 18.

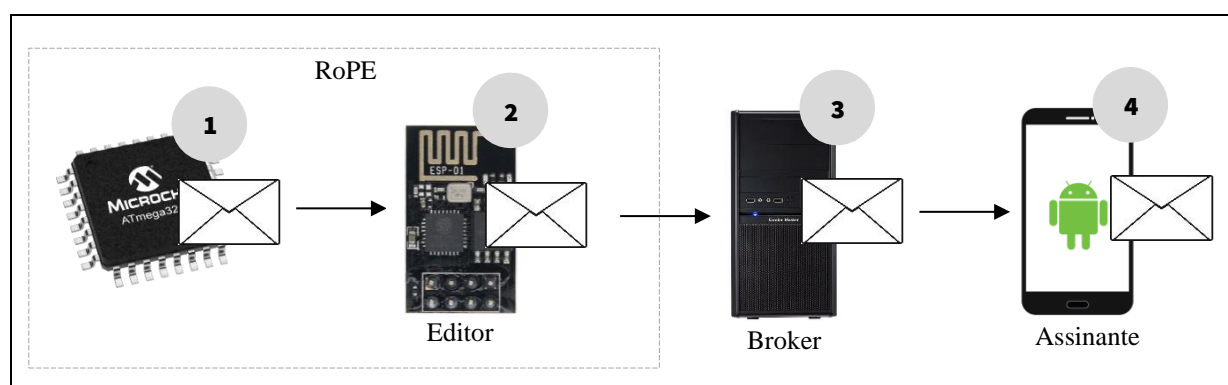


Figura 18. Encaminhamento das mensagens de evento

Para que as mensagens percorram esse caminho ocorrer a seguinte sequência de eventos. Primeiro o RoPE se conecta ao *broker* e publica uma mensagem contendo o seu número de série no *tema* “rope/list”. Ao mesmo tempo, o dispositivo que deseja observar os eventos do RoPE se conecta ao *broker* e se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o número de série para o dispositivo, que pode armazená-lo para uso futuro. Em seguida, o RoPE publica uma mensagem contendo o evento ocorrido no *tema* “rope/{serial}/events”, onde “{serial}” é o número de série do RoPE. Ao mesmo tempo, o dispositivo externo se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o evento ao dispositivo, que pode então processá-lo. Para uma melhor compreensão, a Figura 19 apresenta um diagrama de sequência exemplificando esse processo.

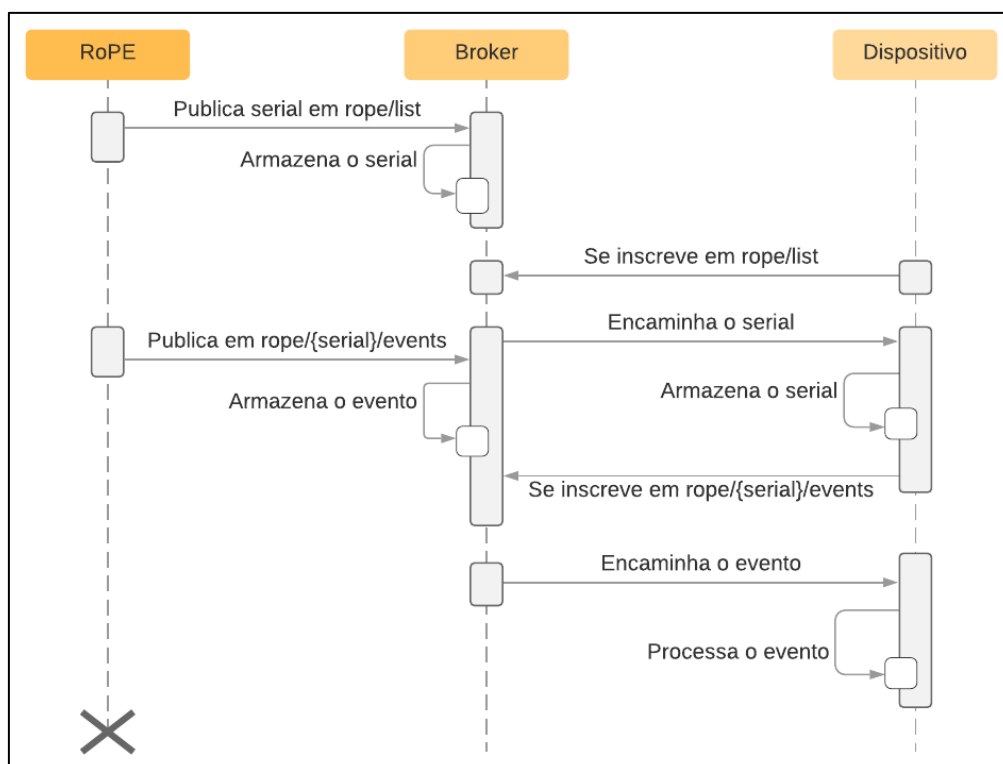


Figura 19. Sequência para o envio de eventos para os dispositivos

O próximo passo é definir o formato das mensagens que serão enviadas durante a comunicação. Quanto à isso, o protocolo MQTT não impõe um formato para o conteúdo (*payload*) das mensagens enviadas, desde que o seu tamanho não ultrapasse 256MB (OASIS MQTT TECHNICAL COMMITTEE, 2014, p. 1), podendo conter desde informações textuais simples até dados binários, como os *bytes* de uma imagem, por exemplo.

No contexto do Smart RoPE, todas as mensagens possuem uma estrutura bem definida. As mensagens de controle remoto são compostas por um comando e por uma lista de parâmetros necessários à sua execução. Por exemplo, o comando para ligar um LED requer que seja informado qual dos cinco LEDs existentes será ligado. Da mesma forma, as mensagens de eventos são compostas pelo tipo de evento e por um conjunto de dados relacionados a esse evento. Por exemplo, o evento de uma nota sendo reproduzida requer que seja identificada qual foi a nota e qual foi a sua duração.

Tendo isso em mente optou-se por usar o formato JSON no trabalho, pois ele permite representar facilmente a estrutura das mensagens, além de ser um formato leve, independente de linguagem de programação e amplamente usado na Web (WEHNER; PIBERGER; GOHRINGER, 2014).

3.3.4 Mensagens de controle

A seguir estão relacionadas todas as mensagens de controle definidas para o projeto (Tabela 5 até Tabela 18) e os requisitos funcionais que elas implementam.

Mensagem de controle		
Comando	MOVE	
Descrição	Move uma determinada distância em milímetros (RF1)	
Parâmetros		
Direction	A direção para qual o RoPE vai se mover	String: [FORWARD BACKWARDS]
Distance	A distância a ser movida (em milímetros)	Float: > 0.0
Exemplos		
{ "command": "MOVE", "parameters": { "direction": "FORWARD", "distance": 10.0 } }		Move 10.0 milímetros para a frente
{ "command": "MOVE", "parameters": { "direction": "BACKWARDS", "distance": 30.0 } }		Move 30.0 milímetros para trás

Tabela 5. Comando MOVE

Mensagem de controle		
Comando	TURN	
Descrição	Gira uma determinada quantidade de graus (RF2)	
Parâmetros		
Direction	O sentido no qual o RoPE vai girar	String: [CLOCKWISE COUNTERCLOCKWISE]
Degrees	A quantidade de graus a girar	Float: > 0.0
Exemplos		
{ "command": "TURN", "parameters": { "direction": "CLOCKWISE", "degrees": -75.0 } }		Gira 75.0 graus em sentido horário
{ "command": "TURN", "parameters": { "direction": "COUNTERCLOCKWISE", "degrees": 90.0 } }		Gira 90.0 graus em sentido anti-horário

Tabela 6. Comando TURN

Mensagem de controle		
Comando	TOGGLE_SOUND	
Descrição	Liga ou desliga o som (RF3)	
Parâmetros		
State	O novo estado do som	String: [ON OFF]
Exemplos		
{ "command": "TOGGLE_SOUND", "parameters": { "state": "ON" } }		Liga o som
{ "command": "TOGGLE_SOUND", "parameters": { "state": "OFF" } }		Desliga o som

Tabela 7. Comando TOGGLE_SOUND

Mensagem de controle		
Comando	TUNE_SOUND	
Descrição	Aplica uma variação de semitons nas notas emitidas pelo <i>buzzer</i> (RF4)	
Parâmetros		
Variation	A quantidade de semitons que será variada	Int: [-36..36]
Exemplos		
{ "command": "TUNE_SOUND", "parameters": { "variation": 2 } }		Aumenta 2 semitons, deixa mais agudo A4 → B4
{ "command": "TUNE_SOUND", "parameters": { "variation": -3 } }		Diminui 3 semitons, deixa mais grave A4 → Gb4

Tabela 8. Comando TUNE_SOUND

Mensagem de controle		
Comando	PLAY_SOUND	
Descrição	Reproduz um som no <i>buzzer</i> (RF5)	
Parâmetros		
Mode	Define o modo de reprodução do som	String: [NOTE FREQUENCY]
Frequency	A frequência a ser reproduzida (se estiver no modo FREQUENCY)	Float: > 0.0
Note	A nota musical a ser reproduzida (se estiver no modo NOTE)	String: [C Cs Db D Ds Eb E F Fs Gb G Gs Ab A As Bb B]
Octave	A oitava da nota a ser reproduzida (se estiver no modo NOTE)	Int: [0..8]
Duration	A duração da nota em segundos	Float: [0.0..60.0]
Pause (Opcional)	Uma pausa em segundos que será acrescentada após reproduzir a nota	Float: [0.0..60.0]
Exemplos		
{ "command": "PLAY_SOUND", "parameters": { "mode": "NOTE", "note": "C", "octave": 5, "duration": 1.5, "pause": 0.5 } }		Reproduz a nota C5 (523.25 Hz) por 1 segundo e meio, logo após, dá uma pausa de meio segundo
{ "command": "PLAY_SOUND", "parameters": { "mode": "NOTE", "note": "E", "octave": 3, "duration": 2.0 } }		Reproduz a nota E3 (164.81 Hz) por 2 segundos, sem dar nenhuma pausa após a reprodução
{ "command": "PLAY_SOUND", "parameters": { "mode": "FREQUENCY", "frequency": 215.78, "duration": 1.0 } }		Reproduz a frequência 215.78 Hz por 1 segundo, sem dar nenhuma pausa após a reprodução

Tabela 9. Comando PLAY_SOUND

Mensagem de controle	
Comando	TOGGLE_LED
Descrição	Liga ou desliga um determinado LED (RF6)
Parâmetros	

Led	O LED que deverá ser alternado	String: [FRONT LEFT RIGHT BACK CENTER]
State	O novo estado do LED	String: [ON OFF]
Exemplos		
{ "command": "TOGGLE_LED ", "parameters": { "led": "FRONT", "state": "ON" } }		Liga o LED frontal
{ "command": "TOGGLE_LED", "parameters": { "led": "BACK", "state": "OFF" } }		Desliga o LED traseiro

Tabela 10. Comando TOGGLE_LED

Mensagem de controle		
Comando	INSERT_INSTRUCTION	
Descrição	Insere uma instrução na memória do RoPE (RF7)	
Parâmetros		
Instruction	A instrução a ser inserida na memória	String: [MOVE_FORWARD MOVE_BACKWARDS TURN_LEFT TURN_RIGHT]
Index (Opcional)	A posição da memória (começando em 0) na qual a instrução será inserida. Se já houver uma instrução nessa posição ela será substituída. Se nenhum valor for informado insere na primeira posição que está vazia	Int: [0..?]
Exemplos		
{ "command": "INSERT_INSTRUCTION", "parameters": { "instruction": "TURN_RIGHT", "index": 2 } }		Insere a instrução “Girar à esquerda” na terceira posição da memória, substituindo a instrução existente nessa posição
{ "command": " INSERT_INSTRUCTION ", "parameters": { "instruction": "MOVE_FORWARD" } }		Varre a memória e insere a instrução “Mover para frente” na primeira posição que estiver vazia

Tabela 11. Comando INSERT_INSTRUCTION

Mensagem de controle		
Comando	REMOVE_INSTRUCTION	
Descrição	Remove uma instrução da memória do RoPE (RF8)	
Parâmetros		
Index (Opcional)	A posição da memória (começando em 0) da qual a instrução será removida. Se nenhum valor for informado remove a instrução que estiver mais próxima do final da memória. As demais posições de memória não são afetadas	Int: [0..?]
Exemplos		
{ "command": "REMOVE_INSTRUCTION", "parameters": { "index": 4 } }		Remove a instrução na quinta posição da memória
{ "command": " REMOVE_INSTRUCTION" }		Remove a instrução que estiver mais próxima do final da memória

Tabela 12. Comando REMOVE_INSTRUCTION

Mensagem de controle		
-----------------------------	--	--

Comando	RESIZE_MEMORY	
Descrição	Redimensiona a memória de instruções (RF9)	
Parâmetros		
Size	O novo tamanho da memória de instruções. Se o tamanho for menor que o atual, descarta as instruções ao final da memória	Int: [1..50]
Exemplos		
{ "command": "RESIZE_MEMORY", "parameters": { "size": 20 } }		Redimensiona a memória para 20 instruções
{ "command": "RESIZE_MEMORY", "parameters": { "size": 45 } }		Redimensiona a memória para 45 instruções (valor padrão)

Tabela 13. Comando RESIZE_MEMORY

Mensagem de controle		
Comando	CLEAR_MEMORY	
Descrição	Limpa a memória de instruções do RoPE (RF10)	
Exemplo		
{ "command": "CLEAR_MEMORY" }		Limpa a memória, removendo todas as instruções e preservando o tamanho da memória

Tabela 14. Comando CLEAR_MEMORY

Mensagem de controle	
Comando	EXECUTE_PROGRAM
Descrição	Executa o programa armazenado na memória (RF11)
Exemplo	
{ "command": "EXECUTE_PROGRAM " }	

Tabela 15. Comando EXECUTE_PROGRAM

Mensagem de controle	
Comando	ABORT_PROGRAM
Descrição	Interrompe o programa que está sendo executado (RF12)
Exemplo	
{ "command": "ABORT_PROGRAM " }	

Tabela 16. Comando ABORT_PROGRAM

Mensagem de controle		
Comando	WAIT	
Descrição	Pausa o RoPE por um determinado intervalo de tempo (RF13)	
Parâmetros		
Interval	O intervalo de tempo (em segundos) que o RoPE deve aguardar	Float: > 0.0

Exemplos	
<code>{"command": "WAIT", "parameters": {"interval": 2.5}}</code>	Pausa o RoPE por 2 segundos e meio
<code>{"command": "WAIT", "parameters": {"interval": 0.1}}</code>	Pausa o RoPE por 100 milissegundos

Tabela 17. Comando WAIT

Mensagem de controle		
Comando	PRESS_BUTTON	
Descrição	Simula o pressionamento de um botão pelo usuário (RF14)	
Parâmetros		
Button	O botão que será pressionado	String: [MOVE_FORWARD MOVE_BACKWARDS TURN_LEFT TURN_RIGHT EXECUTE]
Exemplos		
{ "command": "PRESS_BUTTON", "parameters": { "button": "TURN_LEFT" } }		Pressiona o botão “Girar para a esquerda”
{ "command": "PRESS_BUTTON", "parameters": { "button": "TURN_RIGHT" } }		Pressiona o botão “Girar para a direita”

Tabela 18. Comando PRESS_BUTTON

Mensagem de controle		
Comando	ADD_EVENT_LISTENER	
Descrição	Registra um <i>tema</i> personalizado para receber eventos do RoPE (RF32)	
Parâmetros		
Topic	O <i>tema</i> no qual o RoPE deve publicar os eventos	String
Exemplos		
{ "command": "ADD_EVENT_LISTENER", "parameters": { "topic": "ct-assessment/session-1/test-1"}}		Adiciona o <i>tema</i> “ct-assessment/session-1/test-1” à lista de <i>temas</i> que terão os eventos publicados. Quando um evento ocorrer no RoPE ele será publicado no <i>tema</i> padrão “rope/{serial}/events” e no <i>tema</i> adicional “rope/{serial}/events/ct-assessment/session-1/test-1”

Tabela 19. Comando ADD_EVENT_LISTENER

Mensagem de controle		
Comando	REMOVE_EVENT_LISTENER	
Descrição	Remove um <i>tema</i> previamente registrado para receber eventos do RoPE (RF33)	
Parâmetros		
Topic	O <i>tema</i> no qual o RoPE deve parar de publicar os eventos	String
Exemplos		
{ "command": "REMOVE_EVENT_LISTENER", "parameters": { "topic": "ct-assessment/session-1/test-1" } }		Remove o <i>tema</i> “ct-assessment/session-1/test-1” previamente registrado na lista de <i>temas</i> que terão os eventos

	publicados. Quando um evento ocorrer no RoPE ele não será mais publicado no tema padrão tema adicional “rope/{serial}/events/ct-assessment/session-1/test-1”, somente no tema padrão “rope/{serial}/events”
--	---

Tabela 20. Comando REMOVE_EVENT_LISTENER

3.3.5 Mensagens de evento

A seguir estão relacionadas todas as mensagens de evento definidas para o projeto (Tabela 21 até Tabela 36) e os requisitos funcionais que elas implementam.

Mensagem de evento		
Evento	MOVED	
Descrição	Evento disparado quando o RoPE se move em alguma direção (RF15)	
Dados		
Direction	A direção para a qual o RoPE se moveu	String: [FORWARD BACKWARDS]
Distance	A distância que o RoPE se moveu (em milímetros)	Float: > 0.0
Exemplos		
{ "event": "MOVED", "data": { "direction": "FORWARD", "distance": 10.0 } }		Evento indicando que o RoPE se moveu 10.0 milímetros para a frente
{ "event": "MOVED", "data": { "direction": "BACKWARDS", "distance": 30.0 } }		Evento indicando que o RoPE se moveu 30.0 milímetros para trás

Tabela 21. Evento MOVED

Mensagem de evento		
Evento	TURNED	
Descrição	Evento disparado quando o RoPE gira em algum sentido (RF16)	
Dados		
Direction	O sentido no qual o RoPE girou	String: [CLOCKWISE COUNTERCLOCKWISE]
Degrees	A quantidade de graus que o RoPE girou	Float: > 0.0
Exemplos		
{ "event": "TURNED", "data": { "direction": "CLOCKWISE", "degrees": -75.0 } }		Evento indicando que o RoPE girou 75.0 graus em sentido horário
{ "event": "TURNED", "data": { "direction": "COUNTERCLOCKWISE", "degrees": 90.0 } }		Evento indicando que o RoPE girou 90.0 graus em sentido anti-horário

Tabela 22. Evento TURNED

Mensagem de evento	
Evento	BATTERY_CHARGE_CHANGED
Descrição	Evento disparado quando a carga da bateria se altera (RF17)

Dados		
Previous charge	O nível de carga anterior da bateria (em percentual)	Float: [0.0..100.0]
Current charge	O nível de carga atual da bateria (em percentual)	Float: [0.0..100.0]
Critical Treshold	O limiar de carga antes da bateria entrar em estado crítico (em percentual)	Float: [0.0..100.0]
Exemplo		
<pre>{ "event": "BATTERY_CHARGE_CHANGED", "data": { "previous_charge": 63.0, "current_charge": 59.0, "critical_treshold": 5.0 } }</pre>		Evento indicando que o nível de carga da bateria diminuiu de 63.0% para 59.0%. O nível está acima do limiar de bateria crítica (5.0%), portanto a bateria não precisa ser recarregada no momento

Tabela 23. Evento BATTERY_CHARGE_CHANGED

Mensagem de evento		
Evento	SOUND_TOGGLED	
Descrição	Evento disparado quando o som é ligado ou desligado (RF18)	
Dados		
Previous State	O estado anterior do som	String: [ON OFF]
Current State	O estado atual do som	String: [ON OFF]
Exemplo		
{ "event": "SOUND_TOGGLED", "data": { "previous_state": "OFF", "current_state": "ON" } }		Evento indicando que o som estava desligado e agora foi ligado

Tabela 24. Evento SOUND_TOGGLED

Mensagem de evento		
Evento	SOUND_TUNED	
Descrição	Evento disparado quando a tonalidade do <i>buzzer</i> é alterada (RF19)	
Dados		
Previous Variation	A variação de semitons anterior	Int: [-36..36]
Current Variation	A variação de semitons atual	Int: [-36..36]
Exemplo		
{ "event": "SOUND_TUNED", "data": { "previous_variation": 1, "current_variation": 7 } }		Evento indicando que a variação de tonalidade foi alterada de 1 semitom para 7 semitons

Tabela 25. Evento SOUND_TUNED

Mensagem de evento		
Evento	SOUND_PLAYED	
Descrição	Evento disparado quando um som é reproduzido no <i>buzzer</i> (RF20)	
Dados		
Mode	O modo de usado para reproduzir o som	String: [NOTE FREQUENCY]
Note	A nota musical que foi reproduzida	String: [C Cs Db D Ds Eb E F Fs Gb G Gs Ab A As Bb B]

Octave	A oitava da nota que foi reproduzida	Int: [0..8]
Duration	A duração da nota em segundos	Float: [0.0..60.0]
Pause (Opcional)	A pausa (em segundos) que foi gerada após reproduzir a nota (se houver)	Float: [0.0..60.0]
Exemplos		
{"event": "SOUND_PLAYED", "data": {"mode": "NOTE", "note": "C", "octave": 5, "frequency": 523.25, "duration": 1.5, "pause": 0.5}}		Evento indicando que a nota C5 foi reproduzida por 1 segundo e meio, com uma pausa de meio segundo após a reprodução. A frequência, 523.25 Hz, foi inferida a partir da nota
{"event": "SOUND_PLAYED", "data": {"mode": "FREQUENCY", "frequency": 523.25, "note": "C", "octave": 5, "duration": 1.5, "pause": 0.5}}		Evento indicando que a frequência 523.25 Hz foi reproduzida por 1 segundo e meio, com uma pausa de meio segundo após a reprodução. A nota, C5, foi inferida a partir da frequência
{"event": "SOUND_PLAYED", "data": {"mode": "FREQUENCY", "frequency": 510.17, "duration": 1.5, "pause": 0.5}}		Evento indicando que a frequência 510.17 Hz foi reproduzida por 1 segundo e meio, com uma pausa de meio segundo após a reprodução. A nota não pôde ser inferida pois a frequência não representa uma nota válida

Tabela 26. Evento SOUND_PLAYED

Mensagem de evento		
Evento	LED_TOGGLED	
Descrição	Evento disparado quando um LED é ligado ou desligado (RF21)	
Dados		
Led	O LED que foi alternado	String: [FRONT LEFT RIGHT BACK CENTER]
Previous State	O estado anterior do LED	String: [ON OFF]
Current State	O estado atual do LED	String: [ON OFF]
Exemplos		
{"event": "LED_TOGGLED", "data": { "led": "FRONT", "previous_state": "ON", "current_state": "OFF"}}		Evento indicando que o LED frontal estava ligado e agora foi desligado
{"event": "LED_TOGGLED", "data": { "led": "BACK", "previous_state": "OFF", "current_state": "ON"}}		Evento indicando que o LED traseiro estava desligado e agora foi ligado

Tabela 27. Evento LED_TOGGLED

Mensagem de Evento		
Evento	INSTRUCTION_INSERTED	
Descrição	Evento disparado quando uma instrução é inserida na memória do RoPE (RF22)	
Dados		
Instruction	A instrução que foi inserida na memória	String: [MOVE_FORWARD MOVE_BACKWARDS TURN_LEFT TURN_RIGHT]

Index	A posição da memória (começando em 0) na qual a instrução foi inserida	Int: [0..?]
Memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		
<pre>{ "event": "INSTRUCTION_INSERTED", "data": { "instruction": "TURN_RIGHT", "index": 2, "memory": { "previous_state": { "size": 4, "used": 1, "free": 3, "insertions": 1, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "", "", ""], } "current_state": { "size": 4, "used": 2, "free": 2, "insertions": 2, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "", "TURN_RIGHT", ""] } } }}</pre>		Evento indicando que a instrução “Girar para a esquerda” foi inserida na terceira posição da memória. A memória, que continha apenas 1 instrução “Mover para frente”, agora tem 2 instruções

Tabela 28. Evento INSTRUCTION_INSERTED

Mensagem de evento		
Evento	INSTRUCTION_REMOVED	
Descrição	Evento disparado quando uma instrução é removida da memória do RoPE (RF23)	
Dados		
Instruction	A instrução que foi removida da memória	String: [MOVE_FORWARD MOVE_BACKWARDS TURN_LEFT TURN_RIGHT]
Index	A posição da memória (começando em 0) da qual a instrução foi removida	Int: [0..?]
Memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		
<pre>{ "event": "INSTRUCTION_REMOVED", "data": { "instruction": "MOVE_FORWARD", "index": 0, "memory": { "previous_state": { "size": 4, "used": 2, "free": 2, "insertions": 2, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "", "TURN_RIGHT", ""]}, "current_state": { "size": 4, "used": 1, "free": 3, "insertions": 2, "removals": 1, "cleanings": 0, "resizings": 0, "content": ["", "", "TURN_RIGHT", ""]} } }}</pre>		Evento indicando que uma instrução “Mover para frente” foi removida da primeira posição da memória. A memória, que continha 2 instruções, agora contém apenas 1 instrução “Girar para a direita”

Tabela 29. Evento INSTRUCTION_REMOVED

Mensagem de evento		
Evento	MEMORY_RESIZED	
Descrição	Evento disparado quando a memória do RoPE é redimensionada (RF24)	
Dados		
Memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		
{ "event": "MEMORY_RESIZED", "data": { "memory": {		Evento indicando que a memória foi redimensionada de 2 para 4 instruções.

<pre> "previous_state": { "size": 2, "used": 1, "free": 1, "insertions": 1, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["", "TURN_RIGHT"] }, "current_state": { "size": 4, "used": 1, "free": 3, "insertions": 1, "removals": 0, "cleanings": 0, "resizings": 1, "content": ["", "TURN_RIGHT", "", ""] } } </pre>	A instrução “Girar para a direita”, que se encontrava na posição 1, não foi afetada
--	---

Tabela 30. Evento MEMORY_RESIZED

Mensagem de evento		
Evento	MEMORY_CLEARED	
Descrição	Evento disparado quando a memória do RoPE é apagada (RF25)	
Dados		
Memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		
<pre>{ "event": "MEMORY_CLEARED", "data": { "memory": { "previous_state": { "size": 4, "used": 2, "free": 2, "insertions": 2, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["", "TURN_RIGHT", "", "TURN_LEFT"] }, "current_state": { "size": 4, "used": 0, "free": 4, "insertions": 2, "removals": 0, "cleanings": 1, "resizings": 0, "content": ["", "", "", ""] } } }}</pre>		Evento indicando que a memória foi apagada, liberando 2 posições que estavam ocupadas por instruções

Tabela 31. Evento MEMORY_CLEARED

Mensagem de evento		
Evento	PROGRAM_STARTED	
Descrição	Evento disparado quando o RoPE inicia a execução de um programa (RF26)	
Dados		
Execution	Objeto contendo informações sobre a execução atual	Object
Memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		
<pre>{ "event": "PROGRAM_STARTED ", "data": { "execution": { "progress": 0.0, "statistics": { "started": 2, "completed": 1, "aborted": 0 }}, "memory": { "current_state": { "size": 3, "used": 3, "free": 0, "insertions": 3, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "TURN_RIGHT", "TURN_LEFT"]} } } }}</pre>		Evento indicando que um programa foi iniciado. O programa que está sendo executado é o que está armazenado na memória

Tabela 32. Evento PROGRAM_STARTED

Mensagem de evento

Evento	INSTRUCTION_EXECUTED	
Descrição	Evento disparado quando a próxima instrução do programa é executada (RF27)	
Dados		
Execution	Objeto contendo informações sobre a execução atual	Object
Memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		
<pre>{ "event": "INSTRUCTION_EXECUTED", "data": { "execution": { "instruction": 1, "progress": 66.0, "statistics": { "started": 2, "completed": 1, "aborted": 0 }}, "memory": { "current_state": { "size": 3, "used": 3, "free": 0, "insertions": 3, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "TURN_RIGHT", "TURN_LEFT"]} } } }}</pre>		Evento indicando que a segunda instrução do programa foi executada, representando um progresso de 66.0% na execução do programa. O programa que está sendo executado é o que está armazenado na memória

Tabela 33. Evento INSTRUCTION_EXECUTED

Mensagem de evento		
Evento	PROGRAM_FINISHED	
Descrição	Evento disparado quando um programa é finalizado (RF28)	
Dados		
Reason	Indica o motivo pelo qual o programa finalizou	String: [COMPLETED ABORTED]
Execution	Objeto contendo informações sobre a execução atual	Object
Memory	Objeto contendo informações úteis sobre a memória	Object
Exemplos		
<pre>{ "event": "PROGRAM_FINISHED", "data": { "reason": "ABORTED", "execution": { "instruction": 0, "progress": 33.0, "statistics": { "started": 2, "completed": 1, "aborted": 1 }}, "memory": { "current_state": { "size": 3, "used": 3, "free": 0, "insertions": 3, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "TURN_RIGHT", "TURN_LEFT"]} } } }}</pre>		Evento indicando que o programa finalizou após ser abortado enquanto executava a primeira instrução. O programa completou apenas 33.0% da execução
<pre>{ "event": "PROGRAM_FINISHED", "data": { "reason": "COMPLETED", "execution": { "instruction": 2, "progress": 100.0, "statistics": { "started": 2, "completed": 1, "aborted": 1 }}, "memory": { "current_state": { "size": 3, "used": 3, "free": 0, "insertions": 3, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "TURN_RIGHT", "TURN_LEFT"] } } }}</pre>		Evento indicando que o programa finalizou após ter completado a execução corretamente, ou seja, após ter executado 100.0% das instruções

Tabela 34. Evento PROGRAM_FINISHED

Mensagem de evento		
Evento	BUTTON_PRESSED	
Descrição	Evento disparado quando um botão do RoPE é pressionado (RF29)	
Dados		
Button	O botão que foi pressionado	String: [MOVE_FORWARD MOVE_BACKWARDS TURN_LEFT TURN_RIGHT EXECUTE]
Exemplos		
{ "event": " BUTTON_PRESSED", "data": { "button": "TURN_LEFT" } }		Evento indicando que o botão “Girar para a esquerda” foi pressionado
{ "event": "BUTTON_PRESSED", "data": { "button": "TURN_RIGHT" } }		Evento indicando que o botão “Girar para a direita” foi pressionado

Tabela 35. Evento BUTTON_PRESSED

Mensagem de evento		
Evento	SWITCHED_ON	
Descrição	Evento disparado quando o RoPE é ligado (RF30)	
Dados		
Battery	Objeto contendo informações iniciais da bateria	Object
Memory	Objeto contendo informações iniciais da memória	Object
Sound	Objeto contendo informações iniciais do som	Object
Exemplo		
{ "event": "SWITCHED_ON", "data": { "battery": { "charge": 75.0, "critical_treshold": 5.0 }, "memory": { "size": 3, "used": 0, "free": 3 }, "sound": { "state": "ON", "variation": 0 } }		Evento indicando que o RoPE foi recém ligado. A mensagem contém as configurações iniciais do brinquedo

Tabela 36. Evento SWITCHED_ON

3.4 VALIDAÇÃO

Nesta seção são discutidas estratégias que serão usadas para validar a implementação realizada e verificar se a solução cumpre com o que foi proposto. A validação será feita através de um aplicativo de testes a ser desenvolvido e está dividida duas etapas: (i) testar a execução de comandos remotos e a coleta de dados e, (ii) estudo de caso usando a coleta de dados para auxiliar em um experimento de avaliação do pensamento computacional.

3.4.1 Aplicativo de testes

O aplicativo de testes a ser desenvolvido vai possuir 3 telas. A tela inicial (Figura 20), consiste em uma lista de todas as unidades do RoPE que estão ligadas e conectadas à Internet. As unidades

estarão identificadas pelo seu número de série e informações de propriedade (previamente gravadas no firmware). A tela deve possuir um campo para pesquisa, permitindo buscar unidades específicas do RoPE a partir dessas informações. O objetivo desta tela é permitir que seja feita a seleção da unidade do RoPE com a qual os testes serão realizados.

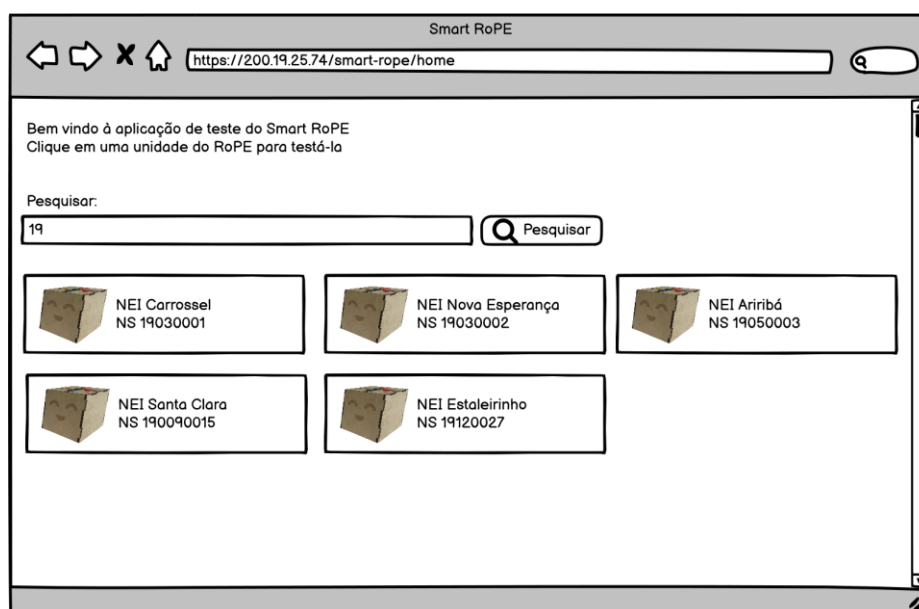


Figura 20. Protótipo da tela inicial do aplicativo de testes

Ao clicar em uma unidade do RoPE na tela principal o aplicativo deve ser redirecionado à tela de testes da unidade (Figura 21). Nesta tela, são apresentadas novamente as informações básicas da unidade, a fim de identificá-la, bem como dois painéis de gerenciamento que serão usados para conduzir os testes.

O teste de controle remoto será realizado através do primeiro painel, denominado “Controle”. Este painel deve apresentar elementos visuais que correspondem às funcionalidades do RoPE que podem ser controladas. Os elementos devem ser interativos, respondendo às ações do usuário. Dentre os elementos estão:

- **Teclado direcional:** permite programar o RoPE simulando o pressionamento dos botões físicos e/ou enviar comandos para movimentá-lo instantaneamente na direção escolhida
- **Painel de LEDs:** permite enviar comandos para ligar e desligar os LEDs correspondentes do RoPE. Os LEDs são da mesma cor dos LEDs do RoPE e arranjos no mesmo layout do LEDs físicos do brinquedo

- **Controles de som:** permite ligar e desligar o som do RoPE e alterar a tonalidade dos sons reproduzidos pelo *buzzer*
- **Teclado musical:** um teclado que imita um piano com pelo menos duas oitavas musicais. Permite reproduzir no *buzzer* do RoPE a nota musical correspondente à tecla

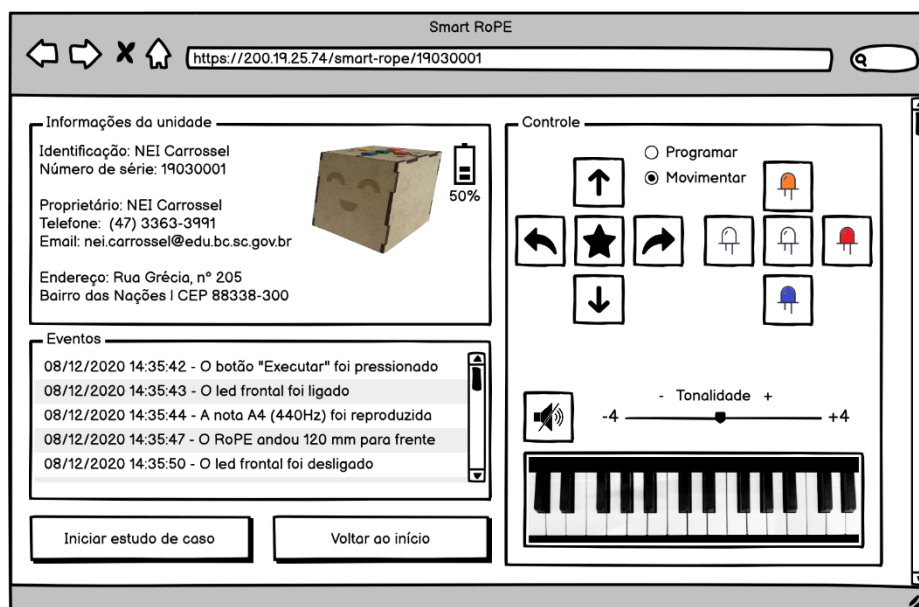


Figura 21. Protótipo da tela de teste da unidade do aplicativo

A validação do controle remoto é feita da seguinte forma:

1. O usuário escolhe um controle do painel, por exemplo, um LED
2. O usuário interage com o controle, por exemplo, clicando no LED
3. O usuário observa o comportamento da unidade do RoPE
 - a. O RoPE executa a ação correspondente, validando o comando
 - b. O RoPE não executa nenhuma ação ou executa uma ação diferente, invalidando o comando
4. O processo é repetido escolhendo outro controle

O teste de coleta de dados do RoPE será feito através do painel “Eventos”. Este painel consiste em uma listagem de todos os eventos ocorridos com o brinquedo, sejam por interação física com o RoPE ou através do painel de controle. A validação é realizada da seguinte forma:

1. O usuário escolhe um controle do painel, por exemplo, um LED
2. O usuário interage com o controle, por exemplo, clicando no LED

3. O usuário observa a lista de eventos
 - a. Um novo item é acrescentado na lista, correspondente à ação realizada no RoPE. O evento é validado
 - b. Nenhum item é acrescentado na lista ou o item acrescentado não corresponde à ação realizada no RoPE. O evento é invalidado
4. O processo é repetido escolhendo outro controle

Os resultados dos testes serão registrados em um relatório para ser incluído nos anexos do trabalho após a sua conclusão.

3.4.2 Estudo de caso

Além de validar a implementação do projeto em si, é desejável validar se o RoPE como um *Smart Toy* e seu sistema de coleta de dados podem ser usados como ferramenta para auxiliar em pesquisas. Para isso, será realizado um estudo caso com um teste de avaliação do pensamento computacional que está sendo desenvolvido pelo grupo de pesquisa em informática na educação da UNIVALI.

O teste em questão faz uso de um tapete do RoPE que foi elaborado especificamente com o propósito de permitir a avaliação do pensamento computacional (Figura 22). O tapete consiste em uma grade onde cada célula contém a gravura de um animal e uma célula especial com a gravura do RoPE, que determina o ponto inicial onde o brinquedo deve ser posicionado antes de cada teste.



Figura 22. Tapete para avaliação do pensamento computacional

O teste é dividido em várias etapas que deverão ser cumpridas em sequência. Cada etapa é composta por um ou mais enunciados no qual é requisitado que a criança execute um algoritmo para cumprir um objetivo específico. Conforme a criança avança nos testes o nível de dificuldade e complexidade dos algoritmos é aumentada através da introdução de condições e restrições. A Tabela 37 relaciona alguns exemplos de enunciados de teste. É importante ressaltar que os enunciados apresentados aqui apenas exemplificam o funcionamento dos testes e podem sofrer alterações, visto que o modelo de avaliação ainda está em fase de desenvolvimento e aprimoramento.

Etapas	Enunciado	Dificuldade
Etapa 1	Leve o RoPE até o coelho	Muito fácil
	Leve o RoPE até a baleia	Muito fácil
Etapa 2	Leve o RoPE até o rato	Muito Fácil
	Leve o RoPE até o elefante	Muito Fácil
Etapa 3	Leve o RoPE até o porco	Fácil
	Leve o RoPE até a vaca	Fácil
...
...
Etapa 8	Leve o RoPE até o gato passando pelo cavalo	Médio
	Leve o RoPE até o cachorro sem passar pelo coelho nem pela galinha	Médio

Tabela 37. Enunciados para a avaliação do pensamento computacional

O pensamento computacional engloba quatro pilares principais para a solução de problemas: (i) decomposição, (ii) reconhecimento de padrões, (iii) abstração e, (iv) algoritmos (BRACKMANN et al., 2016). Cada um dos testes foi elaborado visando avaliar o grau de desenvolvimento da criança nestes pilares. Essa avaliação é realizada através da análise de múltiplas variáveis que podem ser observadas durante os testes. A Tabela 38 relaciona algumas dessas variáveis a fim de exemplo (outras variáveis existem, mas não estão relacionadas aqui).

Variáveis do pensamento computacional
Número de instruções usadas para resolver o problema
Tempo total necessário para resolver o problema
Quantidade de erros cometidos
Quantidade de vezes que o teste precisou ser reiniciado
Quantidade de vezes que a criança interrompeu o programa no meio da execução
Se a criança concluiu o desafio usando uma solução válida diferente da esperada
Número de instruções a mais que foram usadas

Variáveis do pensamento computacional
Se a criança chegou ao objetivo, mas usando instruções proibidas pelo enunciado
Média de tempo entre cada instrução programada
Intervalo de tempo desde o início do teste até a programação da primeira instrução
Média de tempo para a solução dos enunciados

Tabela 38. Variáveis do pensamento computacional

Algumas dessas variáveis são facilmente observadas e mensuradas por um ser humano durante a execução dos testes como, por exemplo, o número de instruções usadas resolução do problema. Porém, outras variáveis são mais difíceis de medir e observar, principalmente nas etapas em que a complexidade dos algoritmos é maior e a contagem de instruções aumenta. Um exemplo é a média de tempo entre cada instrução programada. Medir esta variável requer cronometrar e anotar o intervalo entre cada pressionamento de botão do RoPE, o que pode se tornar inviável.

Dentro desse contexto, a proposta do trabalho é desenvolver uma ferramenta de apoio à realização dos testes, usando o mecanismo de propagação de eventos do *Smart RoPE* para facilitar e automatizar a mensuração dessas variáveis. A ferramenta será implementada dentro do aplicativo de testes proposto na seção 3.4.1 .

A tela principal da ferramenta (Figura 23) exibirá uma árvore com todas as etapas de teste e seus respectivos enunciados. O componente permitirá iniciar qualquer teste em qualquer ordem, ficando a cargo do operador selecionar os testes na ordem correta.

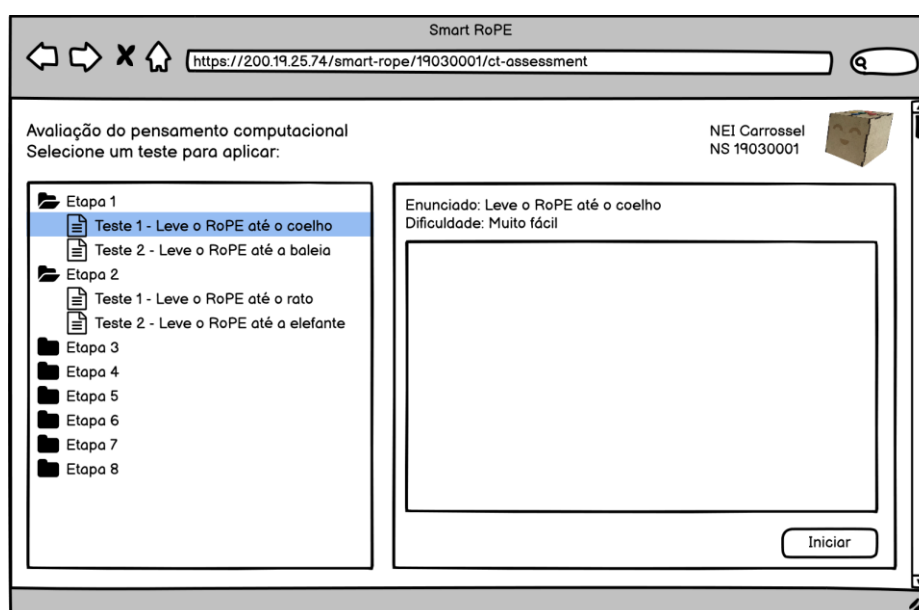


Figura 23. Tela principal do gerenciador de testes

A execução do experimento, envolverá 3 pessoas: a criança (sujeito do teste), um professor para supervisionar a criança e um avaliador para coordenar a execução dos testes no aplicativo. O experimento será conduzido da seguinte maneira:

1. O professor ligará o RoPE e aguardará até que ele esteja conectado na rede WiFi
2. O avaliador acessará o aplicativo, selecionará o RoPE correspondente e abrirá a tela de gerenciamento dos testes
3. O avaliador selecionará um teste na árvore e sinalizará ao professor quando estiver pronto para iniciar o experimento, informando ao professor qual teste foi escolhido
4. Ao ser sinalizado, o professor fará a preparação da criança da forma que achar necessário como, por exemplo, explicando o funcionamento do RoPE
5. Quando a criança estiver preparada, o professor (que terá uma cópia impressa dos testes a serem realizados) fará a leitura do enunciado para a criança e posicionará o brinquedo na posição inicial
6. Neste momento, o avaliador clicará no botão “Iniciar” para dar início à coleta das informações. O aplicativo começará a fazer o registro dos eventos enviados pelo RoPE e exibirá na tela essas informações para que o avaliador possa acompanhar (Figura 24). As informações que aparecerem na tela serão também registradas em um banco de dados para processamento posterior

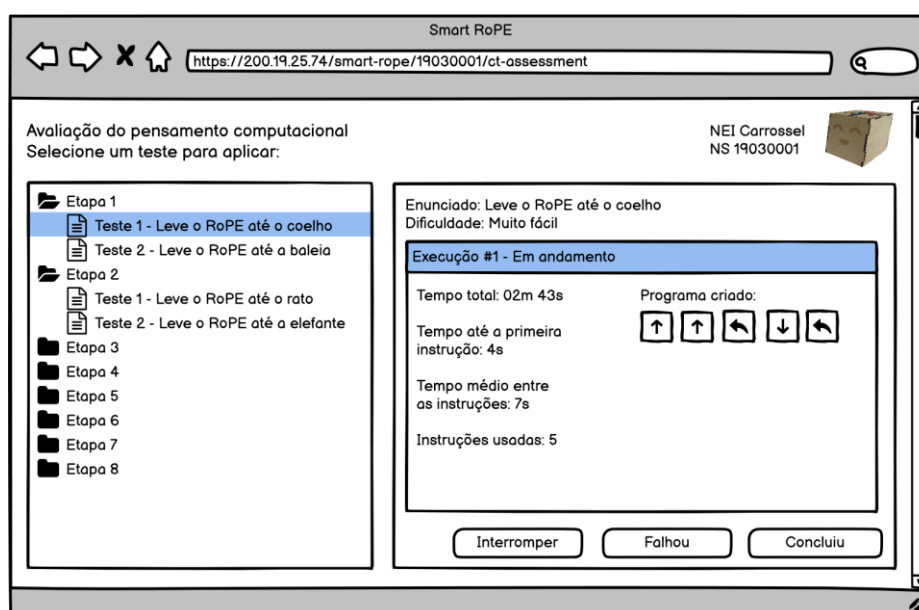


Figura 24. Execução de um teste em andamento

7. Ambos, o professor e o avaliador deixarão a criança livre para experimentar o brinquedo enquanto observam, devendo intervir somente nas seguintes situações:
 - a. quando a criança concluir o desafio com sucesso. Neste cenário, o avaliador clicará no botão “Concluir”, sinalizando o aplicativo de que o desafio foi concluído com sucesso pela criança
 - b. quando a criança desistir do desafio, quando a criança pedir para reiniciar o desafio ou quando se esgotar o tempo estipulado para a conclusão do desafio. Neste cenário, o avaliador clicará no botão “Falhou”, sinalizando o aplicativo de que a criança falhou em concluir o desafio
 - c. quando houver qualquer outro acontecimento que impeça a conclusão do desafio como, por exemplo: (i) a criança necessitou ir ao banheiro, (ii) os pais da criança vieram levá-la para casa, (iii) outros eventos não previstos. Neste cenário, o avaliador clicará no botão “Interromper”, sinalizando o aplicativo de que o teste precisou ser interrompido, mas que não houve falha nem sucesso
8. Em todos os cenários descritos no item anterior, o aplicativo fará o registro da execução e voltará ao estado inicial (Figura 25). Neste instante, o avaliador pode:
 - a. Clicar no registro da execução para visualizar os dados coletados
 - b. Clicar no botão “Iniciar” para começar uma nova execução do mesmo teste (Figura 26). Retorna ao item 6
 - c. Avança para o item 9

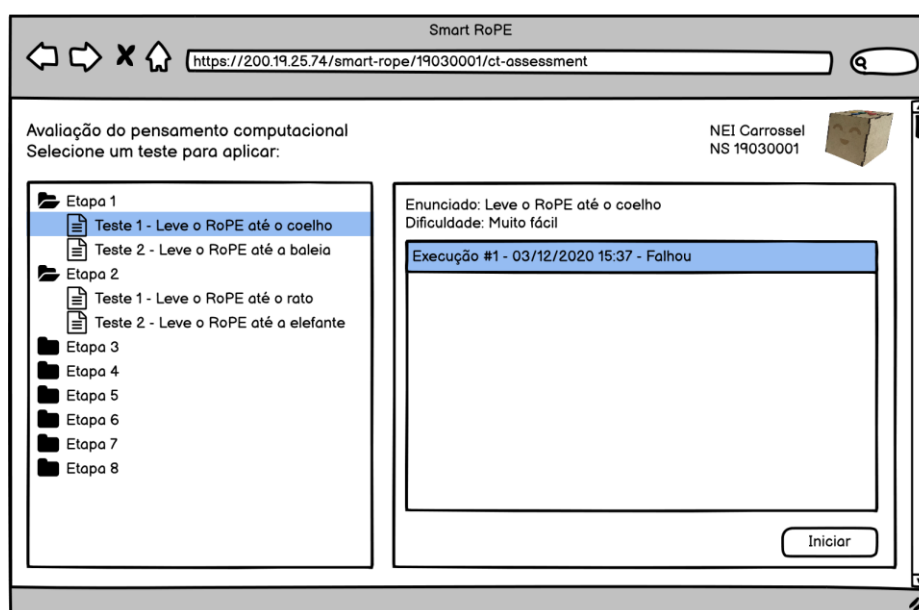


Figura 25. Os testes já executados ficam registrados

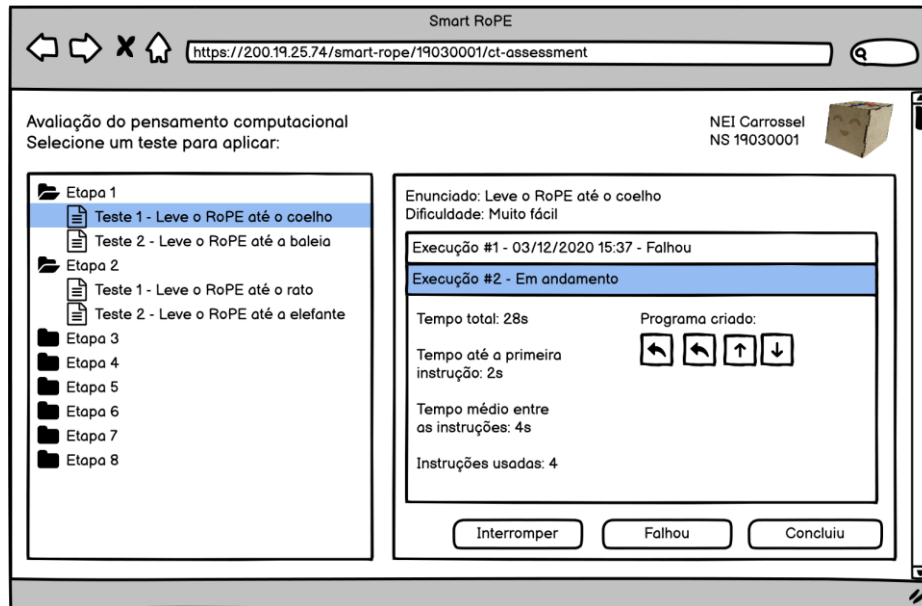


Figura 26. Um mesmo teste sendo executado novamente

9. Ambos, o professor e o avaliador, entram em consenso e a bateria de testes é finalizada

Após a finalização dos testes as informações coletadas pelo aplicativo estarão registradas em um banco de dados. Não está no escopo deste trabalho processar os dados a fim de avaliar o grau de desenvolvimento do pensamento computacional. Os dados, porém, ficarão à disposição do grupo de pesquisa para usar como achar mais conveniente.

É importante ressaltar que o passo a passo descrito até aqui consiste apenas em uma proposta para a execução dos testes e pode não ser a melhor estratégia. Uma falha desse design é que ele permite realizar os testes apenas com uma criança de cada vez. Trabalhos futuros poderão replanejar o experimento a fim de melhorá-lo.

Para melhor compreensão de como os dados serão armazenados foi modelado um diagrama de banco de dados (Figura 27). As tabelas “test_suite”, “test_stage” e “test_challenge” armazenam o teste do pensamento computacional, as etapas do teste e os enunciados de cada etapa, respectivamente. O registro das execuções dos testes, ilustrado anteriormente no protótipo da aplicação, são armazenadas na tabela “test_executions”. Por fim, os eventos recebidos do RoPE bem como as variáveis computadas a partir desses eventos, são armazenados nas tabelas “rope_event” e “test_variable”, respectivamente.

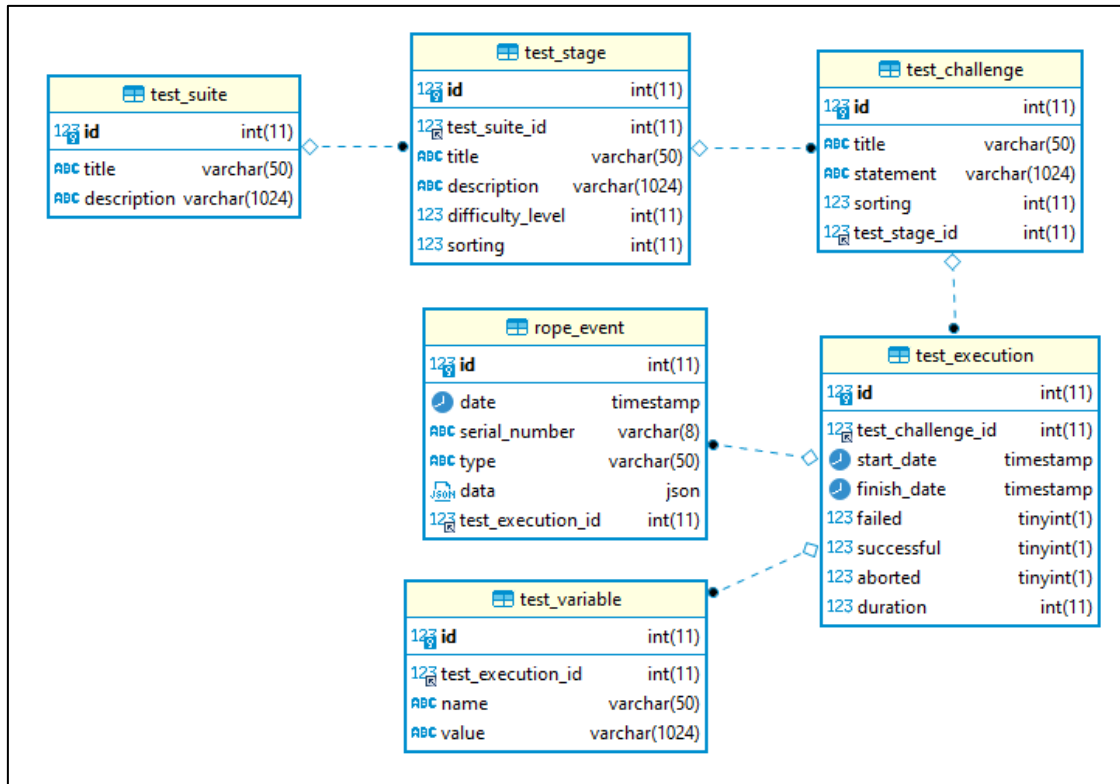


Figura 27. Diagrama ER do banco de dados do aplicativo Smart RoPE

4 CONSIDERAÇÕES FINAIS

Objetivos alcançados

Durante o andamento da pesquisa alguns dos objetivos já foram alcançados. O primeiro objetivo alcançado foi a escolha do módulo Wi-Fi e sua ligação no RoPEa ser usado no projeto. Optou-se por usar o módulo ESP-01 e os motivos dessa escolha são apresentados no capítulo 3.1. Também foi alcançado o objetivo de escolher o protocolo de comunicação a ser usado no sistema. Optou-se por usar o protocolo MQTT e a escolha está justificada no capítulo 3.2. Outro objetivo também já foi cumprido foi definir os comandos que o RoPE vai aceitar executar remotamente e os dados que ele vai permitir serem coletados/monitorados. As mensagens MQTT referentes a estes itens estão relacionadas nos capítulos 3.3.4 e 3.3.5 respectivamente. Por último, o objetivo de implementar o aplicativo de testes e validação foi parcialmente cumprido, pois o banco de dados apresentado no diagrama ER do capítulo 3.4.2 já foi implementado e testado.

Riscos do projeto

Este trabalho não apresenta nenhum risco que impeça a sua execução.

Plano de trabalho

Quadro 1. Cronograma de execução do projeto para (2021/1)

Atividade	Jan	Fev	Mar	Abr	Mai	Jun
Implementar o sistema de troca de mensagens	x	x				
Implementar o aplicativo de testes		x	x			
Testar a implementação e realizar o estudo de caso			x			
Documentar os resultados obtidos			x			
Redigir a versão final do trabalho para a defesa	x	x	x			

REFERÊNCIAS

AHN, J. Y. et al. **MOYA: Interactive AI toy for children to develop their language skills**. Proceedings of the 9th Augmented Human International Conference on - AH '18. **Anais...** In: THE 9TH AUGMENTED HUMAN INTERNATIONAL CONFERENCE. Seoul, Republic of Korea: ACM Press, 2018. Disponível em: <<http://dl.acm.org/citation.cfm?doid=3174910.3174957>>. Acesso em: 11 dez. 2020

BHAWIYUGA, A.; DATA, M.; WARDA, A. **Architectural design of token based authentication of MQTT protocol in constrained IoT device**. 2017 11th International Conference on Telecommunication Systems Services and Applications (TSSA). **Anais...** In: 2017 11TH INTERNATIONAL CONFERENCE ON TELECOMMUNICATION SYSTEMS SERVICES AND APPLICATIONS (TSSA). Lombok: IEEE, out. 2017. Disponível em: <<http://ieeexplore.ieee.org/document/8272933/>>. Acesso em: 6 dez. 2020

BRACKMANN, C. et al. **Computational thinking: Panorama of the Americas**. 2016 International Symposium on Computers in Education (SIIE). **Anais...** In: 2016 INTERNATIONAL SYMPOSIUM ON COMPUTERS IN EDUCATION (SIIE). Salamanca, Spain: IEEE, set. 2016. Disponível em: <<http://ieeexplore.ieee.org/document/7751839/>>. Acesso em: 10 dez. 2020

COLOMBO, S. et al. **Dolphin Sam: A Smart Pet for Children with Intellectual Disability**. Proceedings of the International Working Conference on Advanced Visual Interfaces - AVI '16. **Anais...** In: THE INTERNATIONAL WORKING CONFERENCE. Bari, Italy: ACM Press, 2016. Disponível em: <<http://dl.acm.org/citation.cfm?doid=2909132.2926090>>. Acesso em: 5 dez. 2020

ELKIN, M.; SULLIVAN, A.; BERS, M. U. Programming with the KIBO Robotics Kit in Preschool Classrooms. p. 169–186, set. 2016.

EMILY RELKIN et al. **Assessing Young Children's Computational Thinking Abilities**. [s.l.] Tufts University, 2018.

ESPRESSIF. **ESP8266EX Datasheet**Espressif, , 2020. . Disponível em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf>. Acesso em: 5 dez. 2020

GORDON, N. Flexible Pedagogies: technology-enhanced learning. p. 25, jan. 2014.

HOLLOWAY, D.; GREEN, L. The Internet of toys. **Communication Research and Practice**, v. 2, n. 4, p. 506–519, out. 2016.

HWANG, H. C.; PARK, J.; SHON, J. G. Design and Implementation of a Reliable Message Transmission System Based on MQTT Protocol in IoT. **Wireless Personal Communications**, v. 91, n. 4, p. 1765–1777, dez. 2016.

KASHYAP, M.; SHARMA, V.; GUPTA, N. Taking MQTT and NodeMcu to IOT: Communication in Internet of Things. **Procedia Computer Science**, v. 132, p. 1611–1618, 2018.

LIN, V. Computational Thinking and Technology Toys. p. 115, maio 2015.

MANANDHAR, S. MQTT based communication in IoT. p. 56, 31 maio 2017.

MARTINS, V. F. AUTOMAÇÃO RESIDENCIAL USANDO PROTOCOLO MQTT, NODE-RED E MOSQUITTO BROKER COM ESP32 E ESP8266. p. 53, 2019.

MASCHERONI, G.; HOLLOWAY, D. The Internet of Toys: A Report on Media and Social Discourses around Young Children and IoToys. p. 60, 2017.

MASCHERONI, G.; HOLLOWAY, D. (EDS.). **The Internet of Toys: Practices, Affordances and the Political Economy of Children's Smart Play**. Cham: Springer International Publishing, 2019.

MEHTA, M. ESP 8266: A BREAKTHROUGH IN WIRELESS SENSOR NETWORKS AND INTERNET OF THINGS. **International Journal of Electronics and Communication Engineering & Technology**, v. 6, n. 8, p. 7–11, ago. 2015.

MESQUITA, J. et al. **Assessing the ESP8266 WiFi module for the Internet of Things**. 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA). **Anais...** In: 2018 IEEE 23RD INTERNATIONAL CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA). Turin: IEEE, set. 2018. Disponível em: <<https://ieeexplore.ieee.org/document/8502562/>>. Acesso em: 5 dez. 2020

MICROSHIP. **ESP-01 WiFi Module Version 1.0** Microship, , 2020a.

MICROSHIP. **ATmega48A/PA/88A/PA/168A/PA/328/P megaAVR Data Sheet** Microship, , 2020b. . Disponível em: <<https://ww1.microchip.com/downloads/en/DeviceDoc/ATmega48A-PA-88A-PA-168A-PA-328-P-DS-DS40002061B.pdf>>. Acesso em: 4 dez. 2020

OASIS MQTT TECHNICAL COMMITTEE. **MQTT Version 3.1.1**. . Disponível em: <<https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html>>. Acesso em: 9 dez. 2020.

OLIVEIRA, G. M. B. et al. **Comparison Between MQTT and WebSocket Protocols for IoT Applications Using ESP8266**. 2018 Workshop on Metrology for Industry 4.0 and IoT. **Anais...** In: 2018 WORKSHOP ON METROLOGY FOR INDUSTRY 4.0 AND IOT. Brescia: IEEE, abr. 2018. Disponível em: <<https://ieeexplore.ieee.org/document/8428348/>>. Acesso em: 3 dez. 2020

PAPERT, S. Children, Computers, and Powerful Ideas. p. 11, 1 jan. 1980.

PATEL, C.; DOSHI, N. "A Novel MQTT Security framework In Generic IoT Model". **Procedia Computer Science**, v. 171, p. 1399–1408, 2020.

PERRONE, G. et al. **The Day After Mirai: A Survey on MQTT Security Solutions After the Largest Cyber-attack Carried Out through an Army of IoT Devices**: Proceedings of the 2nd International Conference on Internet of Things, Big Data and Security. **Anais...** In: 2ND INTERNATIONAL CONFERENCE ON INTERNET OF THINGS, BIG DATA AND SECURITY. Porto, Portugal: SCITEPRESS - Science and Technology Publications, 2017. Disponível em: <<http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006287302460253>>. Acesso em: 6 dez. 2020

RAABE, A. et al. **Brinquedos de Programar na Educação Infantil: Um estudo de Caso.** . In: XXI WORKSHOP DE INFORMÁTICA NA ESCOLA. Maceió, Alagoas, Brasil: 26 out. 2015. Disponível em: <<http://br-ie.org/pub/index.php/wie/article/view/4985>>. Acesso em: 10 dez. 2020

RAABE, A. et al. **RoPE - Brinquedo de Programar e Plataforma de Aprender.** . In: XXIII WORKSHOP DE INFORMÁTICA NA ESCOLA. Recife, Pernambuco, Brasil: 27 out. 2017. Disponível em: <<http://www.br-ie.org/pub/index.php/wie/article/view/7349>>. Acesso em: 5 dez. 2020

RAABE, A. L. A.; VIEIRA, M. F. V.; ROSÁRIO, T. A. M. DO. UM RELATO DE EXPERIÊNCIA COM O USO DO BRINQUEDO DE PROGRAMAR BEE-BOT NA EDUCAÇÃO INFANTIL COM CRIANÇAS DE 3 A 4 ANOS DE IDADE. v. 7, n. 13, p. 11, dez. 2015.

RAFFERTY, L. et al. **Towards a Privacy Rule Conceptual Model for Smart Toys**. Proceedings of the 50th Hawaii International Conference on System Sciences. **Anais...** In: HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES. 2017

SAHADEVAN, A. et al. **An Offline Online Strategy for IoT Using MQTT**. 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud). **Anais...** In: 2017 IEEE 4TH INTERNATIONAL CONFERENCE ON CYBER SECURITY AND CLOUD COMPUTING (CSCLOUD). New York, NY, USA: IEEE, jun. 2017. Disponível em: <<http://ieeexplore.ieee.org/document/7987225/>>. Acesso em: 3 dez. 2020

SANTOS, S. **Robótica além da lógica: o uso de brinquedos de programar no dia a dia escolar**, 18 nov. 2019. . Disponível em: <<http://lite.acad.univali.br/pt/18/11/2019/3848/>>

SARAMA, J.; CLEMENTS, D. H. Design of Microworlds in Mathematics and Science Education. **Journal of Educational Computing Research**, v. 27, n. 1, p. 1–5, jul. 2002.

SHIN, S. et al. **A security framework for MQTT**. 2016 IEEE Conference on Communications and Network Security (CNS). **Anais...** In: 2016 IEEE CONFERENCE ON COMMUNICATIONS AND NETWORK SECURITY (CNS). Philadelphia, PA, USA: IEEE, out. 2016. Disponível em: <<http://ieeexplore.ieee.org/document/7860532/>>. Acesso em: 6 dez. 2020

SHINHO LEE et al. **Correlation analysis of MQTT loss and delay according to QoS level**. The International Conference on Information Networking 2013 (ICOIN). **Anais...** In: 2013 INTERNATIONAL CONFERENCE ON INFORMATION NETWORKING (ICOIN). Bangkok:

IEEE, jan. 2013. Disponível em: <<http://ieeexplore.ieee.org/document/6496715/>>. Acesso em: 3 dez. 2020

SINGH, M. et al. **Secure MQTT for Internet of Things (IoT)**. 2015 Fifth International Conference on Communication Systems and Network Technologies. **Anais...** In: 2015 FIFTH INTERNATIONAL CONFERENCE ON COMMUNICATION SYSTEMS AND NETWORK TECHNOLOGIES (CSNT). Gwalior, India: IEEE, abr. 2015. Disponível em: <<http://ieeexplore.ieee.org/document/7280018/>>. Acesso em: 6 dez. 2020

SONI, D.; MAKWANA, A. A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS (IOT). p. 5, 2017.

SULLIVAN, A. et al. KIBO Robot Demo: Engaging Young Children in Programming and Engineering. p. 4, 2015.

TORRES, A. B. B.; ROCHA, A. R.; DE SOUZA, J. N. Análise de Desempenho de Brokers MQTT em Sistema de Baixo Custo. p. 12, 2016.

VALENTE, J.; CARDENAS, A. A. **Security & Privacy in Smart Toys**. Proceedings of the 2017 Workshop on Internet of Things Security and Privacy - IoTS&P '17. **Anais...** In: THE 2017 WORKSHOP. Dallas, Texas, USA: ACM Press, 2017. Disponível em: <<http://dl.acm.org/citation.cfm?doid=3139937.3139947>>. Acesso em: 11 dez. 2020

WEHNER, P.; PIBERGER, C.; GOHRINGER, D. **Using JSON to manage communication between services in the Internet of Things**. 2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC). **Anais...** In: 2014 9TH INTERNATIONAL SYMPOSIUM ON RECONFIGURABLE AND COMMUNICATION-CENTRIC SYSTEMS-ON-CHIP (RECOSOC). Montpellier, France: IEEE, maio 2014. Disponível em: <<http://ieeexplore.ieee.org/document/6861361/>>. Acesso em: 9 dez. 2020

YANG, J.; LU, Z.; WU, J. Smart-toy-edge-computing-oriented data exchange based on blockchain. **Journal of Systems Architecture**, v. 87, p. 36–48, jun. 2018.