LUIZ FERNANDO NOSCHANG

EVOLUINDO O BRINQUEDO ROPE PARA SE TORNAR UM SMART TOY

Itajaí (SC), dezembro de 2020

UNIVERSIDADE DO VALE DO ITAJAÍ

CURSO 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: André Luis Alice Raabe, PhD.

Itajaí (SC), dezembro de 2020

EVOLUINDO O BRINQUEDO ROPE PARA SE TORNAR UM SMART TOY

Luiz Fernando Noschang

Setembro/2021

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: 119

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 ESP32, descreve e detalha o projeto da alteração no robô e descreve as aplicações que foram criadas a fim de validar o funcionamento do sistema desenvolvido.

EVOLVING THE TOY ROPE TO BECOME A SMART TOY

Luiz Fernando Noschang

September/2021

Advisor: André Luís Alice Raabe, Dr.

Area of Concentration: Applied Computer Science

Research Line: Informatics on Education

Palavras-chave: Smart Toys, RoPE, IoT

Número de páginas: 119

ABTRACT

Smart Toys are a new modality of toys that add electronic components and microprocessors allowing toys to become embedded systems. In this way, intelligent features can be added to the design of a toy, providing it with internet access and with various sensors and actuators. The way of interacting with the child can be greatly enriched, adapting to its needs. The use of speech recognition, natural language processing, artificial intelligence and cloud computing can provide opportunities for interaction that have so far been largely unexplored. In this context, ethical aspects and risks to the privacy and exposure of children emerge, that need to be seriously discussed in order to guarantee safety for the evolution of this concept of toys.This dissertation seeks to enable the educational toy RoPE, a low cost educational programmable toy produced at LITE - UNIVALI, to become a Smart Toy. RoPE, despite being an embedded system, cannot be considered a Smart Toy, as it is not capable of reacting and adapting in a personalized way to user interactions. This work proposes to modify RoPE to connect it to the Internet and to implement a communication system using the MQTT protocol to allow it to be controlled and monitored remotely.It is hoped that these changes will serve as a basis for turning RoPE into a Smart Toy and that future applications can be developed exploring this capability. The work justifies the choice for the ESP32 controller, describes and details the design of the change in the robot and describes the applications created to validate the proper working of the system developed.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTA DE ABREVIATURAS

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

SUMÁRIO

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 projeto[1].

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 Timeline[2], 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

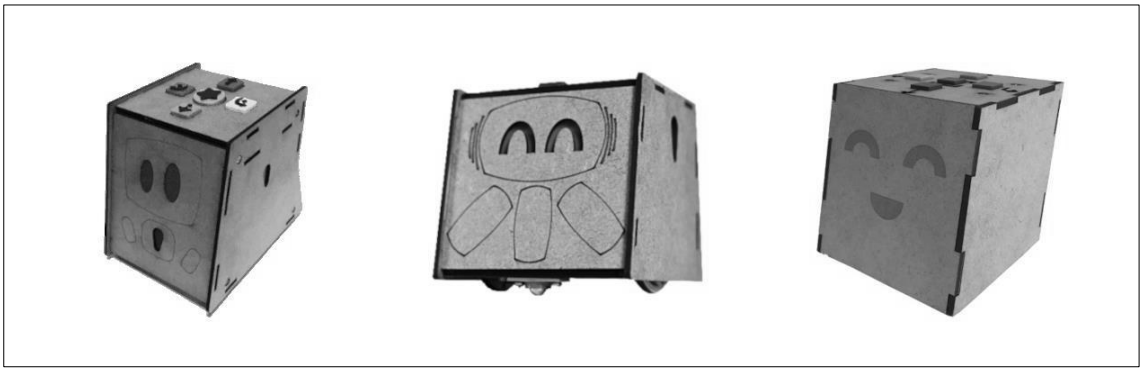


Figura 1. Evolução da carenagem do RoPE (da esquerda para a direita)
Fonte: O autor

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 impulsione 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 microcontrolador com suporte à Wi-Fi para usar no RoPE;
3. Implementar no firmware do RoPE a conexão com à Internet via rede Wi-Fi

4. Implementar no forware do RoPE a troca de mensagens de controle e monitoramento usando o protocolo MQTT;
5. Implementar um aplicativo de testes que possa ser usado para verificar o funcionamento da implementação do firmware;
6. 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 protocolo de comunicação usado na implementação do sistema e do microcontrolador com suporte à Wi-Fi foi feita levando em conta as necessidades e limitações do RoPE, as quais estão descritas e justificadas no texto dos capítulos 2.4 e 3.1, respectivamente.

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 testes foram criados dois aplicativos para validar o funcionamento do sistema implementado, os quais estão detalhados nos capítulos 4.1 e 4.2.

1.4 ESTRUTURA DA DISSERTAÇÃO

Este documento foi dividido em cinco capítulos: (i) Introdução, (ii) Fundamentação teórica, (iii) Desenvolvimento, (iv) Resultados e (v) Conclusões.

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, Desenvolvimento, foram abordados todos os itens relacionados à implementação do sistema, como: escolha do módulo Wi-Fi, levantamento dos requisitos funcionais e definição das mensagens de controle e eventos.

No capítulo 4, Resultados, são apresentados os aplicativos criados para testar e validar a implementação do mecanismo de controle e monitoramento desenvolvido.

No capítulo 5, Conclusões, é realizada uma síntese do trabalho, descrevendo os resultados alcançados e sugestões para trabalhos futuros.

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. Na sequência, são analisados alguns trabalhos similares que poderão ajudar no desenvolvimento do projeto. Por fim, é feita uma breve introdução ao protocolo MQTT, necessário para a implementação da troca de mensagens via Internet.

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 à 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 avanças 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.

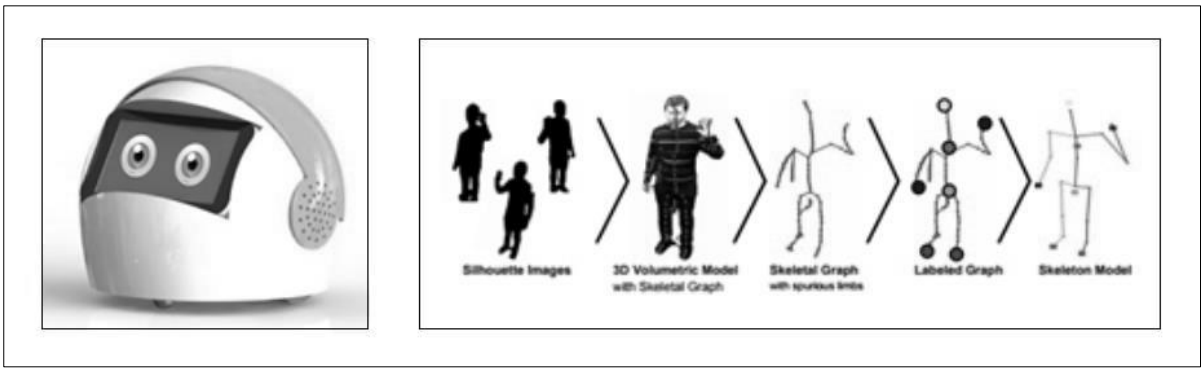


Figura 2. O brinquedo MOYA (à esquerda) e seu mecanismo de detecção de movimentos
Fonte: Adaptado de (AHN et al., 2018)

Da mesma maneira, um brinquedo pode ser conectado, mas não ser inteligente. Um exemplo é o brinquedo de programar TinkerBot[3]. 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,

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

como é o caso do Smartibot[4], que se conecta ao *smartphone* e usa o poder computacional do aparelho para executar um algoritmo de inteligência artificial e reconhecer animais de estimação, pessoas e veículos (Figura 4).

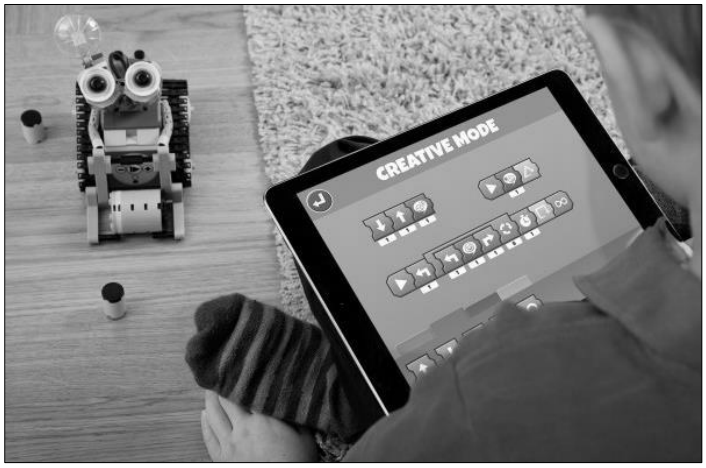


Figura 3. Brinquedo TinkerBot conectado a um *tablet*
Fonte: https://www.generationrobots.com/en/403314-tinkerbots-my-first-robot.html

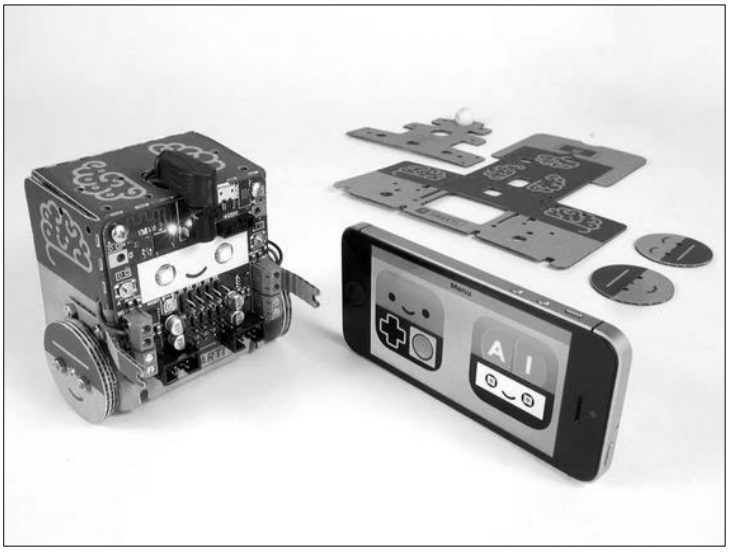


Figura 4. Brinquedo Smartibot conectado ao *smartphone*
Fonte: https://thecraftyrobot.net/

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

[4] https://thecraftyrobot.net/

no computador, tablet ou smartphone como, por exemplo, o ScratchJr[5] e o LightBot[6] (Figura 5). Por outro lado, os brinquedos tangíveis, são aqueles que permitem a interação com o usuário de forma 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.

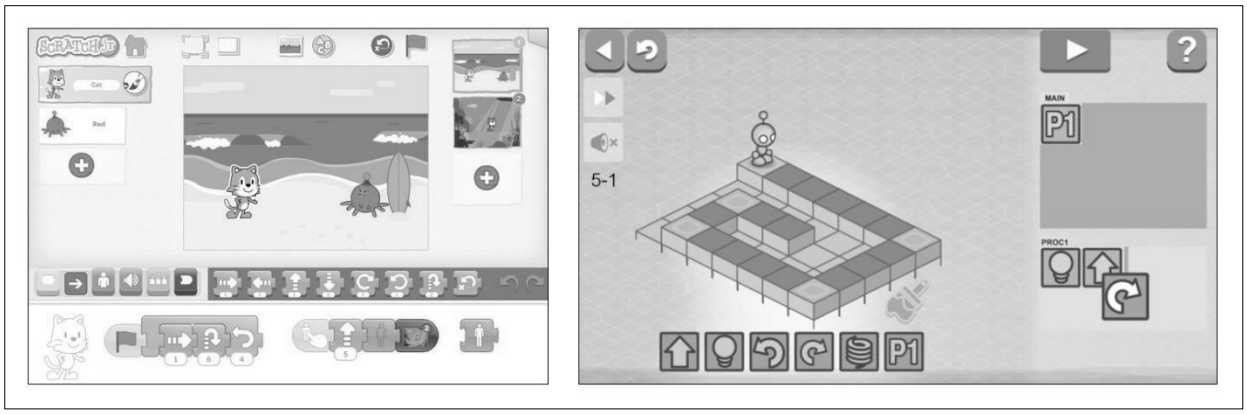


Figura 5. ScratchJr e LightBot, exemplos de *Smart Toys* não tangíveis

Fontes: Google - https://educacaoeinformatica.files.wordpress.com/2018/04/scratchjr.jpg e https://lightbot.com/

2.2 O BRINQUEDO ROPE

O RoPE, **Ro**bô **P**rogramável **E**ducacional, 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

[5] https://www.scratchjr.org/
[6] https://lightbot.com/

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 (Figura 6) 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.

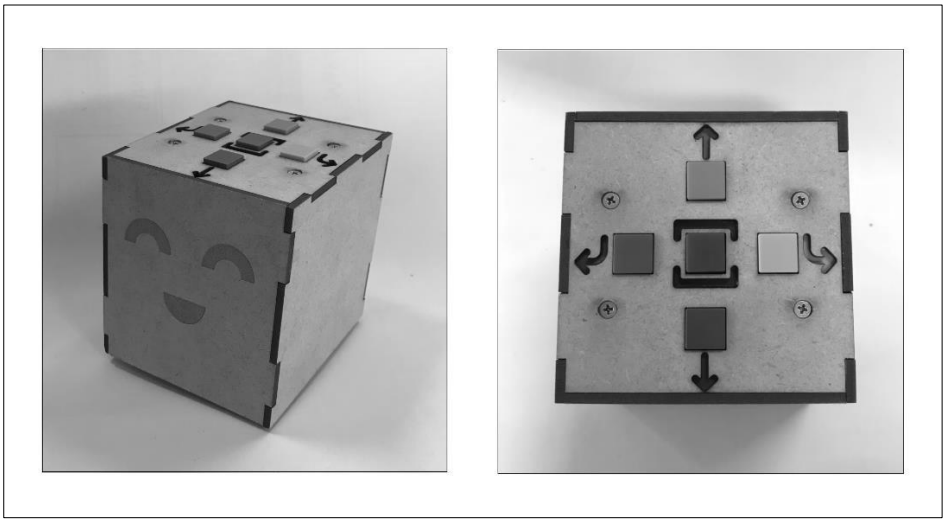


Figura 6. O brinquedo RoPE (à esquerda) e sua interface de programação
Fonte: O autor

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”.

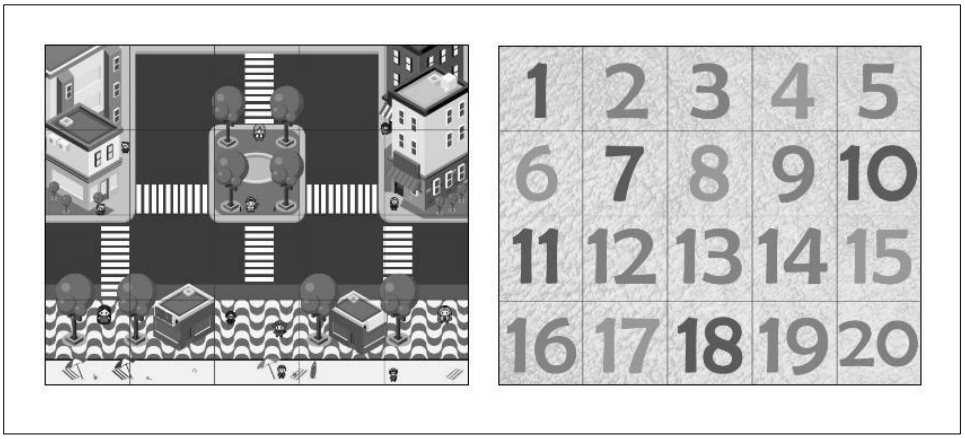


Figura 7. Tapetes pedagógicos do RoPE
Fonte: Adaptado do site do brinquedo - http://smartfunbrasil.com/

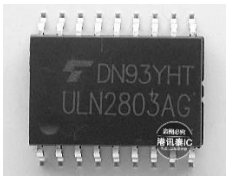
Do ponto de vista técnico “[...] os pesquisadores que construíram o RoPE priorizaram decisões que pudessem reduzir o custo e tornar o brinquedo aderente a realidade dos núcleos de educação infantil brasileiros” (RAABE et al., 2017). Por isso, o RoPE é composto por componentes eletrônicos baratos e que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

RoPE transformem-no em um brinquedo conectado e viabilizem que trabalhos futuros façam dele brinquedo inteligente.

Quadro 1. Principais componentes 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

Fonte: O autor

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 (Figura 8) 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.

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).

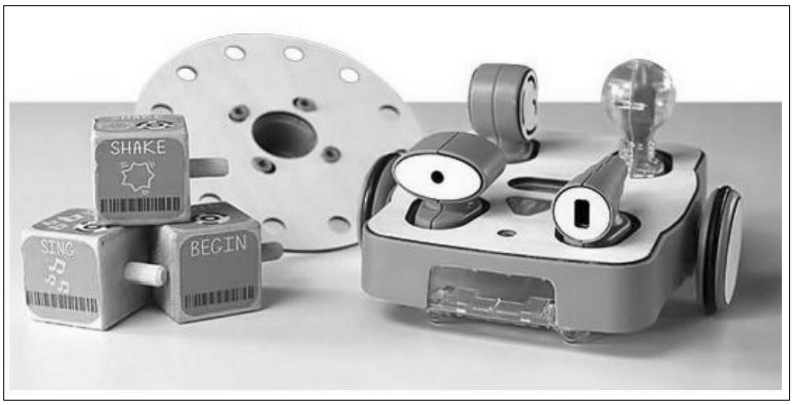


Figura 8. O brinquedo Kibo
Fonte: Google - https://www.clubscikidzmd.com/wp-content/uploads/2018/11/kibo-toy-1_orig.jpg

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.

2.4 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.

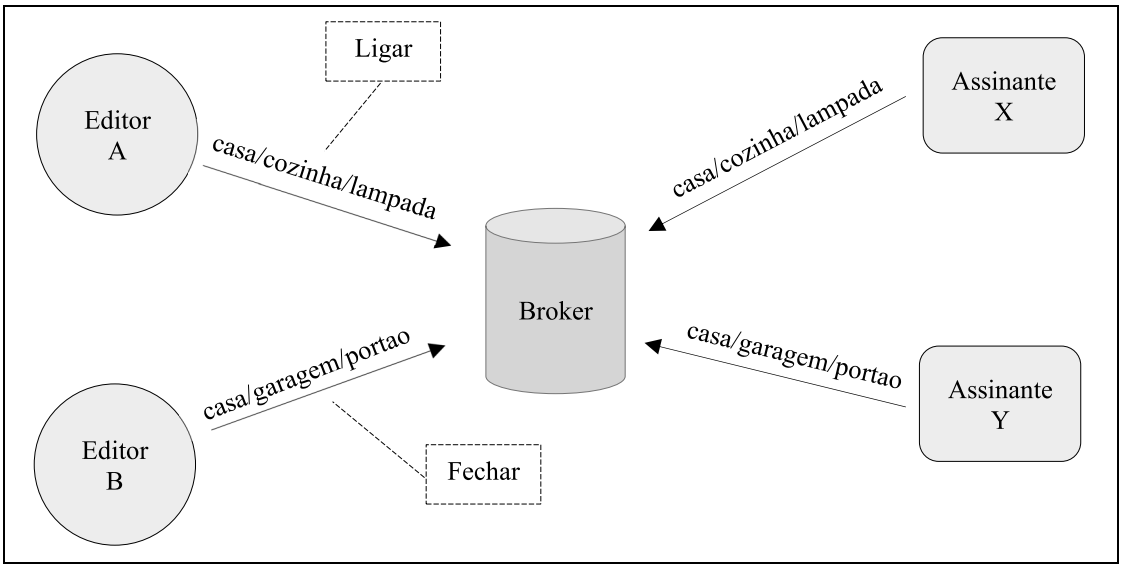


Figura 9. Arquitetura do MQTT

Fonte: O autor

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 9 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 Javascript que serão usadas nesse trabalho. A implementação em Javascript será útil para desenvolver

uma aplicação de estudo de caso ao final do trabalho. Já a implementação em C será usada no ESP32 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*[7] 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. Neste trabalho optou-se por utilizar o *HiveMQ*, um *broker* de alta qualidade que conta com uma série recursos: é facilmente escalável, suporta segurança TLS, é totalmente compatível com todas as versões do MQTT, é compatível com diversas bibliotecas de cliente MQTT e possui um painel de administração que permite acompanhar em tempo real os clientes conectados e as mensagens transmitidas, conforme ilustrado na Figura 10 (HIVEMQ, 2021).

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

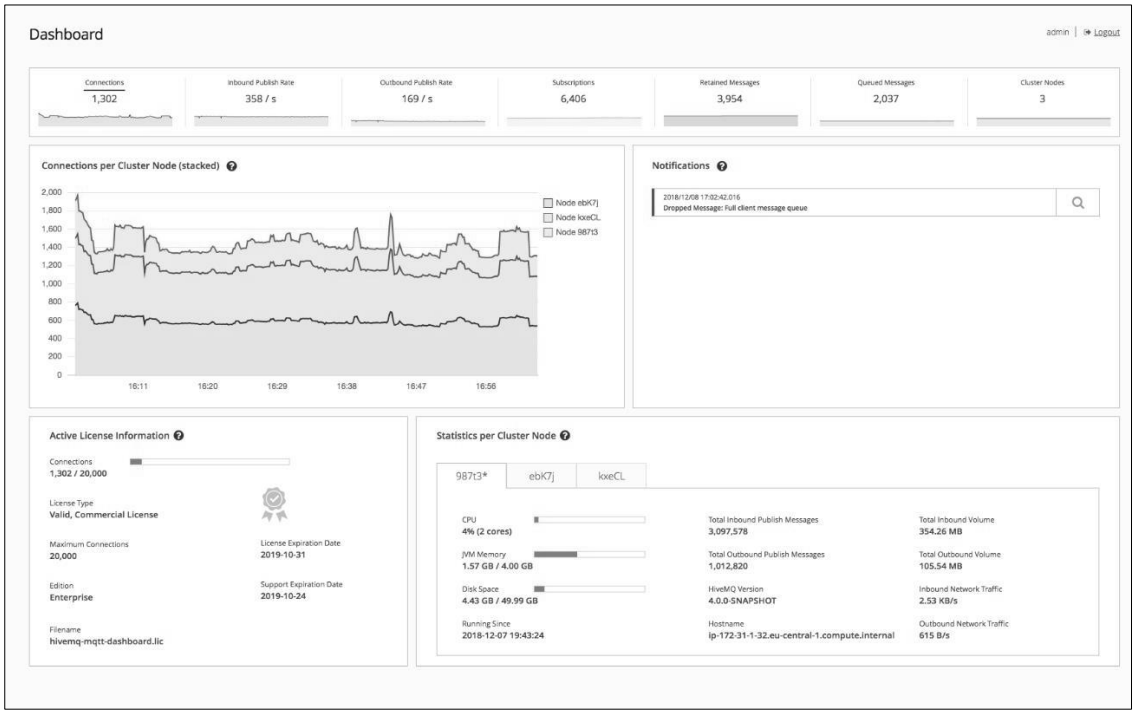


Figura 10. Painel de administração do HiveMQ

Fonte: HiveMQ - https://www.hivemq.com/img/cc-dashboard-view.gif

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) modificar a placa do RoPE para incluir um microcontrolador com suporte à Wi-Fi. Na Parte de *software* a solução consiste em: (i) modificar o firmware para estabelecer uma conexão com a Internet usando o recurso de Wi-Fi do novo hardware, (ii) modificar o *firmware* para executar comandos enviados para o RoPE a partir de dispositivos e aplicações remotas (iii) modificar o *firmware* do RoPE para coletar dados relevantes do funcionamento do brinquedo e enviá-los através da Internet aos dispositivos e aplicações remotas.

3.1 MICROCONTROLADOR ESP32

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 foi necessário alterar o microcontrolador usado na placa do brinquedo, pois o microcontrolador ATmega328P (família AVR da Atmel), usado até o momento, não oferecia essa funcionalidade.

Para o desenvolvimento da nova placa do RoPE foi escolhido o microcontrolador ESP32 da Espressif. O principal critério para a sua escolha foi o fato de o ESP32 ter compatibilidade com a plataforma Arduino oferecida de forma oficial pela fabricante através do plugin *Arduino core for ESP32* (BABIUCH; FOLTYNEK; SMUTNY, 2019). Este fator foi decisivo pois, o firmware original do RoPE foi escrito usando as bibliotecas dessa plataforma. Sendo assim, a adoção do ESP32 diminuiu consideravelmente o impacto na migração do firmware para o novo hardware. Além deste já citado, outros critérios precisaram ser observados na hora da escolha e são detalhados a seguir.

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, foi importante manter a proposta original

dos idealizadores do RoPE e optar por uma solução de baixo custo. Na data em que este trabalho foi escrito, uma versão do ESP32 podia ser encontrada por R$ 34,90 na loja Curto Circuito (Figura 11), enquanto uma versão do ATMega328p (microcontrolador atual do RoPE) podia ser encontrada por R$ 32,88 na loja Baú da Eletrônica (Figura 12). A diferença de preço entre os dois chips é mínima, compensando o investimento pelos recursos que o ESP32 possui.

Figura 11. Custo do ESP32 em uma loja de eletrônica
Fonte: Loja Curto Circuito (https://is.gd/Im13p5)

Figura 12. Custo do ATmega329p em uma loja de eletrônica
Fonte: Loja Baú da Eletrônica (https://is.gd/YWrRCB)

Outro requisito deste trabalho foi 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. A versão anterior do RoPE era capaz de manter essa autonomia acionando a função *Power-Down* presente no chip ATmega328P (MICROSHIP, 2020), 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. Sendo assim, o microcontrolador escolhido também precisava oferecer um consumo de energia baixo e funções de economia de energia similares às do chip anterior.

De forma similar, o ESP32 também oferece funções de economia de energia através de um coprocessador dedicado de baixo consumo, o qual é capaz de realizar operações de conversão analógico-digital e cálculos enquanto está em modo *deep sleep*[8]. No estudo conduzido por (GATIAL; BALOGH; HLUCHY, 2020), o ESP32 foi testado em diferentes modos de operação em uma aplicação usando MQTT, a fim de verificar o consumo de energia do chip. Os testes mostraram que, se aplicadas as estratégias corretas em conjunto com os modos de economia de energia presentes no chip, o consumo pode chegar próximo aos 0.8mAh. Para fins de comparação, o consumo atual do RoPE quando está em modo *deep sleep* é de aproximadamente 20mAh. Portanto, o ESP32 está apto a ser usado no RoPE.

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.

[8] Modo de economia de energia em que os componentes principais são desligados e somente alguns componentes secundários ficam ativos

Com o uso do ESP32 surge a possibilidade destes problemas serem solucionados de forma mais rápida, pois o ESP32 conta com um recurso de atualização *Over The Air* (OTA), o qual permite baixar e gravar uma nova versão do *firmware* a partir da nuvem. Com isso, versões atualizadas e corrigidas do firmware podem ser disponibilizadas para múltiplas unidades o RoPE ao mesmo tempo via Internet, eliminando a necessidade de logística e diminuindo o tempo necessário para a correção de falhas.

Por fim, além dos recursos já apresentados, o ESP32 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, possibilitando a execução de tarefas relativamente mais complexas e que demandem maior desempenho. O Quadro 2, construído a partir dos dados extraídos dos *datasheets* oficiais (MICROSHIP, 2020) e (ESPRESSIF, 2021), mostra uma comparação entre o hardware atual do RoPE e do ESP32.

Quadro 2. Comparação de hardware entre o ATmega328P e o ESP32

	RoPE (ATmega328p)	**ESP32**
CPU	Padrão: 16 MHz[9] Máximo: 20 MHz	Padrão: 240 MHz Máximo: 240 MHz
RAM	2 kB	520 kB
Flash	32 kB	Padrão: 4 MB[10] Máximo: 64 MB

Fonte: O autor

3.2 BIBLIOTECA H4PLUGINS

Durante o processo de migração do firmware do RoPE para o ESP32 foram enfrentadas algumas dificuldades. Uma delas foi realizar a portabilidade do código responsável pela reprodução de sons. No ATMega328p essa funcionalidade era implementada com base na biblioteca “tune” do Arduino, No entanto, para o ESP32 não foi encontrada uma biblioteca similar entre as bibliotecas

[9] No RoPE, a frequência foi reduzida para 8 MHz a fim de economizar energia
[10] A maior parte dos módulos fabricados costumam vir com 4 MB, mas isto pode variar dependendo do fabricante

oficiais, fato este que levou a uma pesquisa a fim de encontrar uma biblioteca alternativa que pudesse auxiliar nessa tarefa. Durante essa pesquisa, foi encontrada uma biblioteca chamada H4Plugins, que além do recurso desejado, oferecia muitas outras soluções úteis para o projeto. Nesta seção será apresentada brevemente essa biblioteca, com o objetivo de auxiliar outros pesquisadores a desenvolverem seus trabalhos com o ESP32 de forma mais fácil e rápida no futuro.

Um dos principais problemas enfrentados durante o desenvolvimento de um firmware para o ESP32 é lidar com o seu funcionamento de natureza assíncrona e concorrente, principalmente para quem está migrando de um microcontrolador baseado na plataforma Arduino. O Arduino possui apenas um núcleo de processamento e funciona de forma sequencial e síncrona. Desta forma, um firmware escrito para o Arduino não precisa se preocupar com eventos ocorrendo de forma concorrente dentro do microcontrolador, salvo nas situações em que se está usando o recurso de interrupções presente no chip.

O ESP32 em contrapartida possui dois núcleos de processamento e conta com conectividade Wi-Fi e conectividade Bluetooth. Estes recursos fazem com que o ESP32 tenha uma natureza naturalmente concorrente, onde vários eventos podem ocorrer de forma paralela dentro do microcontrolador. Por exemplo, um pacote de dados pode ser recebido via Bluetooth ao mesmo tempo em que um cliente faz uma requisição à uma página Web que está sendo servida via Wi-Fi. Se estes eventos não forem tratados da forma correta eles podem levar ao mal funcionamento do firmware.

Considere o famoso código “blink” do Arduino, ilustrado no quadro x, o qual é responsável por piscar um LED. Este código faz uso da função “delay” para determinar a frequência com a qual o LED vai piscar e funciona perfeitamente no Arduino, porém no ESP32 ele é uma fonte certa de problemas, pois bloqueia o fluxo de execução do programa. Se um pacote for recebido na Wi-Fi durante a chamada dessa função ele pode ser perdido ou em casos mais extremos pode até causar o reinício do microcontrolador de forma aleatória e sem motivo aparente.

Quadro 3. Programa “blink” do Arduino

```
void setup() {
  pinMode(LED_BUILTIN, OUTPUT);
}

void loop() {
  digitalWrite(LED_BUILTIN, HIGH);
  delay(1000);
  digitalWrite(LED_BUILTIN, LOW);
  delay(1000);
}
```

Fonte: Adaptado de https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink

Outra questão que geralmente não é levada em conta ao se trabalhar com o ESP32 é referente ao enfileiramento dos eventos, pacotes e mensagens. Como garantir que as requisições serão tratadas na mesma ordem em que foram solicitadas, por exemplo? Segundo (BOWLES, 2021), autor da biblioteca H4Plugins, o gerenciamento de múltiplas tarefas simultâneas em um ambiente embarcado requer conhecimentos avançados de programação em baixo nível e da arquitetura do microcontrolador em questão sendo, portanto, uma tarefa extremamente difícil, principalmente para iniciantes.

Dentro desse contexto, a biblioteca H4Plugins surge como uma alternativa capaz de solucionar boa parte desses problemas. O H4Plugins é na verdade muito mais que uma biblioteca, é um framework completo projetado especificamente para facilitar o desenvolvimento de aplicações IoT usando o ESP32 e o seu “irmão mais novo” o ESP8266. O funcionamento do framework é bem simples e se baseia no conceito de agendamento de tarefas usando *timers* e funções de *callback* que serão executadas após o tempo estipulado. O Quadro 4 ilustra o mesmo exemplo “blink” usando a biblioteca H4Plugins.

Quadro 4. Programa “blink” na biblioteca H4Plugins

```
#include<H4.h>

// Automatically starts Serial for you if speed provided
H4 h4(115200);

void myCallback(){
    // invert pin state
    digitalWrite(LED_BUILTIN, !digitalRead(LED_BUILTIN));
}

// do the same type of thing as the standard setup
void h4setup(){
    pinMode(LED_BUILTIN,OUTPUT);


    // All times are milliseconds, 1000=1 second
    h4.every(1000,myCallback);
}
```

Fonte: Adaptado de https://github.com/philbowles/H4

Ao todo o framework conta com 32 plugins diferentes para trabalhar desde os tópicos mais básicos, como a manipulação de botões e acionamento de LEDs, até tópicos mais avançados, como a atualização OTA. No repositório oficial da biblioteca[11][12] é possível encontrar a relação completa dos recursos oferecidos. Neste trabalho, foram usados os plugins de som, Wi-Fi e MQTT, os quais serão descritos brevemente a seguir.

3.2.1 Plugin de som

O plugin de som da biblioteca H4Plugins permite reproduzir sons no ESP32 através da geração de pulsos PWM. Este processo de modulação é transparente ao usuário, que precisa apenas incluir as classes “H4P_Voice” e “H4P_ToneController” no programa e chamar a função “play”. O Quadro 5 ilustra um exemplo de melodia sendo reproduzida no H4Plugins.

[11] https://github.com/philbowles/h4plugins/blob/master/docs/overview.md
[12] https://github.com/philbowles/h4plugins/blob/master/docs/index.md

Quadro 5. Reprodução de melodia no H4Plugins

```
#include<H4Plugins.h>

#define BUZZER_PIN 27

H4_USE_PLUGINS(115200,H4_Q_CAPACITY,false)

H4P_Voice vox_t(BUZZER_PIN);
H4P_ToneController h4tc; // defaults to metronome = 60 b.p.m.

// Take On Me - Aha
std::string tom_tc = "F#4q |F#4q |DN4q |BN3q |R  q |BN3q |R  q |EN4q " \
              "R  q |EN4q |R  q |EN4q |G#4q |G#4q |AN4q |BN4q " \
              "AN4q |AN4q |AN4q |EN4q |R  q |DN4q |R  q |F#4q " \
              "R  q |F#4q |R  q |F#4q |EN4q |EN4q |F#4q |EN4q ";


void h4setup(){
  h4tc.metronome(165); // Beats per minute (BPM)
  vox_t.play(tom_tc);
}
```

Fonte: Adaptado do exemplo oficial (https://is.gd/sfcHiw)

Ao contrário da biblioteca “tune” do Arduino, que trabalha diretamente com as frequências de som, o H4Plugins trabalha com o conceito de notas e tempos musicais. Para reproduzir uma melodia deve-se informar uma *string* com o conjunto de notas musicais a serem tocadas e separadas pelo caractere “|”. Cada nota deve ter exatamente 5 caracteres, onde cada caractere representa um aspecto da nota a ser reproduzida, conforme descrito no Quadro 6.

Ao reproduzir uma melodia, a biblioteca permite ajustar o volume em até 9 níveis, variando entre 0 e 8, no qual 0 significa ausência total de volume, 8 significa 100% do volume e os demais valores representam frações intermediárias do volume, 25%, 50%. Na prática percebeu-se que o comportamento do volume não é linear, havendo uma diferença de volume mais drástica entre os níveis 4 e 5 e uma diferença quase imperceptível entre os níveis 7 e 8, por exemplo.

Outro parâmetro ajustável é a velocidade de reprodução das melodias, permitindo deixar uma melodia mais rápida ou mais devagar sem alterar as notas musicais individualmente. Essa velocidade de reprodução é definida em batidas por minuto (BPM). Não existe um limite de velocidade imposto pela biblioteca, mas nos testes realizados durante o desenvolvimento do projeto percebeu-se que a maioria dos *buzzers* consegue reproduzir melhor nas velocidades entre 30 bpm e 300 bpm.

Alem da velocidade, é possível especificar um valor de transposição a ser aplicado na reprodução das melodias, permitindo deixar as melodias mais graves ou mais agudas. A transposição é definida em semitons, onde valores negativos transpõem as notas para frequências mais baixas, tornando-as mais graves e, valores positivos transpõem as notas para frequências mais altas, tornando-as mais agudas. O valor 0 reproduz a melodia com suas notas originais, sem nenhuma transposição. Não existe um limite de transposição imposto pela biblioteca, mas nos testes realizados durante o desenvolvimento do projeto percebeu-se que a maioria dos *buzzers* conseguiu reproduzir melhor na faixe de -24 e +12 semitons, tomando como base a nota C4.

Quadro 6. Formato da nota musical no H4Plugins

Nota	C (Dó) até B (Sí); R (Pausa)
Tipo de nota	N = Normal; # = Sustenido
Oitava	2 até 7
Duração	d = Fusa (1/32) D = Fusa pontuada (1/32) + (1/64) s = semicolcheia (1/16) S = semicolcheia pontuada (1/16) + (1/32) q = colcheia (1/8) Q = colcheia pontuada (1/8) + (1/16) c = semínima (1/4) C = semínima pontuada (1/4) + (1/8) m = mínima (1/2) M = mínima pontuada (1/2) + (1/4) b = semibreve (1) B = semibreve pontuada (1) + (1/2)
Reservado	Este campo é reservado para uso interno da biblioteca e geralmente é inserido como um espaço em branco

Fonte: O autor

Por fim, a biblioteca permite usar múltiplos *buzzers* ao mesmo tempo no ESP32, onde cada *buzzer* atua como um canal de áudio independente para compor melodias mais complexas. É possível, por exemplo reproduzir a melodia principal de uma música em um *buzzer* e o seu acompanhamento em outro. No RoPE esse recurso não foi usado, pois a inclusão de novos *buzzers* aumentaria o custo de produção do brinquedo. O APÊNDICE G – REPRODUÇÃO DE SONS, ao final do trabalho,

ilustra através de um diagrama de classes a arquitetura e o funcionamento da reprodução de sons no RoPE.

3.2.2 Plugins de Wi-Fi e MQTT

A biblioteca H4Plugins também possui dois plugins para conexão Wi-Fi e conexão MQTT. Ambos os plugins são facilmente configuráveis com poucas linhas de código e possuem o recurso de se reconectarem automaticamente em caso de falha/perda da conexão. Como pode ser visto no código de exemplo do Quadro 7, para configurar a WiFi, basta informar o SSID da rede, a senha e um *hostname*[13]. Uma vez conectada a Wi-Fi não é necessário fazer nenhum tratamento adicional como *polling*[14], por exemplo, pois a partir daí toda a comunicação passa a ser gerenciada pela biblioteca, que realiza o enfileiramento das requisições e as encaminha para os demais plugins.

Para a conexão MQTT, basta informar o endereço IP ou nome de domínio do servidor *broker* e a porta que se deseja conectar. Opcionalmente podem ser informados um nome de usuário e senha, nos casos em que um sistema de autenticação estiver sendo utilizado. Uma vez estabelecida a conexão, a função de *callback* "onMqttConnect" é chamada automaticamente, possibilitando realizar a inscrição do dispositivo nos *temas* desejados. Ao inscrever o dispositivo em um *tema* do MQTT, o H4Plugins automaticamente acrescenta o *hostname* como um prefixo, facilitando o gerenciamento de múltiplos dispositivos conectados. No exemplo ilustrado no Quadro 7, o dispositivo é inscrito no tema "my-device/my-topic".

No momento da inscrição em um *tema*, deve ser informada uma função de *callback*, que será executada automaticamente pela biblioteca quando uma mensagem for recebida neste *tema* específico. Dentro desta função, o *payload* da mensagem MQTT pode ser facilmente extraído usando a macro "H4PAYLOAD" disponibilizada pela biblioteca. O *payload* sempre é disponibilizado no formato de uma *string* devendo ser convertida manualmente para o formato esperado pelo firmware.

[13] Rótulo atribuído a um dispositivo conectado a uma rede a fim de facilitar sua identificação. Fonte: Wikipedia - https://en.wikipedia.org/wiki/Hostname
[14] Processo no qual o um dispositivo é constantemente monitorado ou requisitado para avaliar se existem informações disponíveis para serem consumidas. Fonte: Wikipedia - https://en.wikipedia.org/wiki/Polling_(computer_science)

Quadro 7. Conexão Wi-Fi e MQTT no H4Plugins

```
#include<H4Plugins.h>

#define WIFI_SSID "my-wifi"
#define WIFI_PASSWORD "123456"
#define HOST_NAME "my-device"

H4_USE_PLUGINS(115200,H4_Q_CAPACITY,false)

H4P_WiFi h4wifi(WIFI_SSID, WIFI_PASSWORD, HOST_NAME);
H4P_AsyncMQTT h4mqtt("http://192.168.1.4:1883");

uint32_t myCallback(std::vector<std::string> vs){
  Serial.printf("Message received: %s\n", H4PAYLOAD.c_str());

  if (H4PAYLOAD == "good"){
      h4mqtt.publishDevice("other-topic", "Valid payload!");
      return H4_CMD_OK;
  }else {
      h4mqtt.publishDevice("other-topic", "Invalid payload!");
      return H4_CMD_PAYLOAD_FORMAT;
  }
}

void onMqttConnect(){
 h4mqtt.subscribeDevice("my-topic", myCallback);
}

void onMqttDisconnect(){}

void h4pGlobalEventHandler(const std::string& svc,H4PE_TYPE t,const
std::string& msg){
    switch(t){
        H4P_DEFAULT_SYSTEM_HANDLER
        case H4PE_SERVICE:
            H4P_SERVICE_ADAPTER(Mqtt);
            break;
    }
}
```

Fonte: Adaptado do exemplo oficial (https://is.gd/AOuvZz)

No firmware do RoPE estes dois plugins foram usados no envio dos eventos e no recebimento dos comandos remotos via MQTT. Nos capítulos **3.3.3 Mensagens de controle** e **3.3.4 Mensagens de evento** são apresentados diagramas de classes que ilustram a arquitetura e o funcionamento do envio de mensagens e sua relação com os plugins da biblioteca H4Plugins.

3.2.3 Considerações finais

O uso da biblioteca H4Plugins facilitou muita a implementação do novo firmware do RoPE, pois resolve naturalmente problemas de concorrência, enfileiramento de pacotes e reconexão com a Wi-Fi que de outra forma teriam que ser implementados como parte do projeto. Isto permitiu focar os esforços na implementação das funcionalidades do RoPE, sem ter que se preocupar com esses detalhes.

A biblioteca conta com uma vasta documentação e exemplos no seu repositório do Github[15], um canal no Youtube[16] com vários tutoriais em vídeos e uma comunidade ativa no Facebook[17] onde são discutidos problemas encontrados e a adição de novas funcionalidades. Durante o desenvolvimento do projeto, houve momentos em que os recursos de suporte disponíveis não foram suficientes para resolver os problemas enfrentados no contexto específico do RoPE, porém, o próprio autor da biblioteca Phill Bowles se disponibilizou pessoalmente, fornecendo explicações mais aprofundadas sobre o funcionamento da biblioteca e, em alguns casos, auxiliando a corrigir *bugs*[18] pontuais.

3.3 ARQUITETURA DO PROJETO

Existem três objetivos 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 no Quadro 8 e serviram como norte para o desenvolvimento.

---

[15] https://github.com/philbowles/h4plugins
[16] https://www.youtube.com/channel/UCYi-Ko76_3p9hBUtleZRY6g/featured
[17] https://www.facebook.com/groups/h4plugins
[18] Defeitos não previstos no código de um software que levam ao seu mau funcionamento

Quadro 8. 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 os parâmetros de reprodução de som do RoPE, tais como tonalidade, volume e velocidade
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 simular o pressionamento dos botões do RoPE
RF14	Os dispositivos externos devem ser notificados quando o RoPE se mover
RF15	Os dispositivos externos devem ser notificados quando o RoPE girar
RF16	Os dispositivos externos devem ser notificados quando o nível de bateria do RoPE mudar
RF17	Os dispositivos externos devem ser notificados quando o som do RoPE for ligado ou desligado
RF18	Os dispositivos externos devem ser notificados quando os parâmetros de reprodução de som do RoPE forem alterados
RF19	Os dispositivos externos devem ser notificados quando um som for reproduzido no RoPE
RF20	Os dispositivos externos devem ser notificados quando um LED do RoPE for ligado ou desligado
RF21	Os dispositivos externos devem ser notificados quando uma instrução for inserida na memória do RoPE

RF22	Os dispositivos externos devem ser notificados quando uma instrução for removida da memória do RoPE
RF23	Os dispositivos externos devem ser notificados quando a memória do RoPE for redimensionada
RF24	Os dispositivos externos devem ser notificados quando a memória do RoPE for apagada
RF25	Os dispositivos externos devem ser notificados quando o RoPE iniciar a execução de um programa
RF26	Os dispositivos externos devem ser notificados do progresso da execução de um programa
RF27	Os dispositivos externos devem ser notificados quando o RoPE finalizar a execução de um programa
RF28	Os dispositivos externos devem ser notificados quando um botão do RoPE for pressionado
RF29	Os dispositivos externos devem ser notificados do estado inicial do RoPE quando ele for ligado
RF30	Os dispositivos externos devem conseguir obter uma lista com o número de série de todas as unidades do RoPE conectadas ao *broker*
RF31	Os dispositivos externos devem conseguir requisitar ao RoPE que publique seu estado

Fonte: O autor

3.3.2 Roteamento das mensagens

O Smart Rope funciona usando o protocolo MQTT, abordado no capítulo 2.4. 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 no Quadro 9 juntamente com seu propósito.

Quadro 9. Temas do Smart RoPE

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 (**RF30**)
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

Fonte: O autor

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 e são recebidas pelo microcontrolador ESP32 para serem então processadas. O caminho percorrido pelas mensagens pode ser visto de maneira simplificada a na Figura 13.

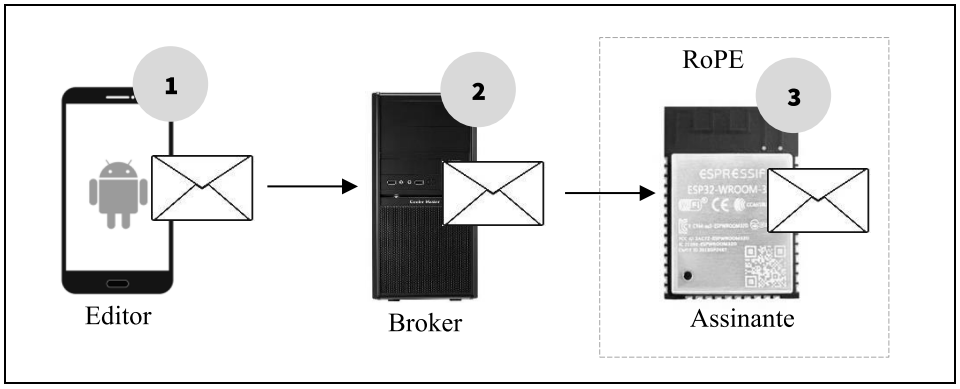


Figura 13. Roteamento das mensagens de controle remoto

Fonte: O autor

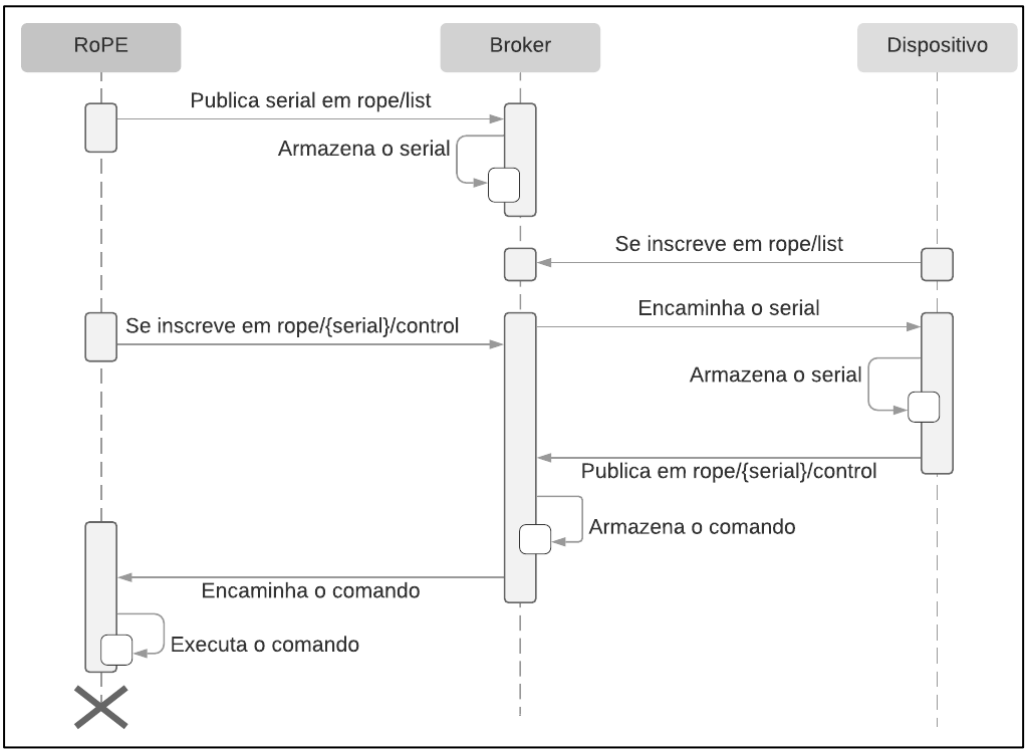


Figura 14. Sequência para envio de comandos para o RoPE

Fonte: O autor

Para que as mensagens percorram esse caminho ocorre 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 14 apresenta um diagrama de sequência exemplificando esse processo.

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 ESP32 e enviadas ao *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 na Figura 15.

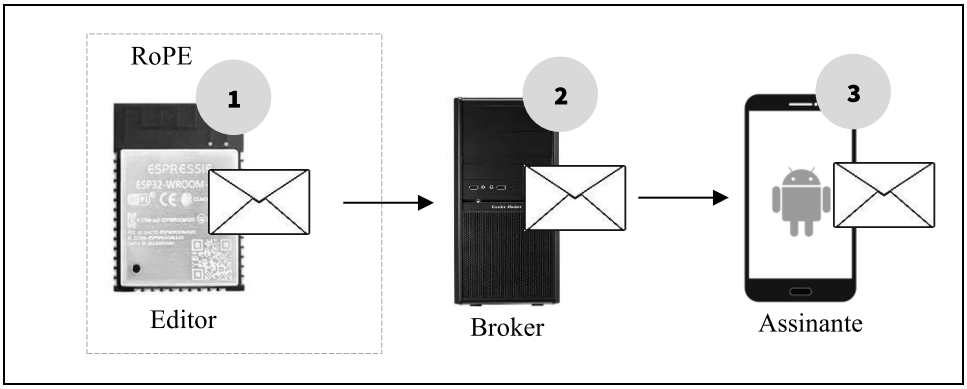


Figura 15. Encaminhamento das mensagens de evento
Fonte: O autor

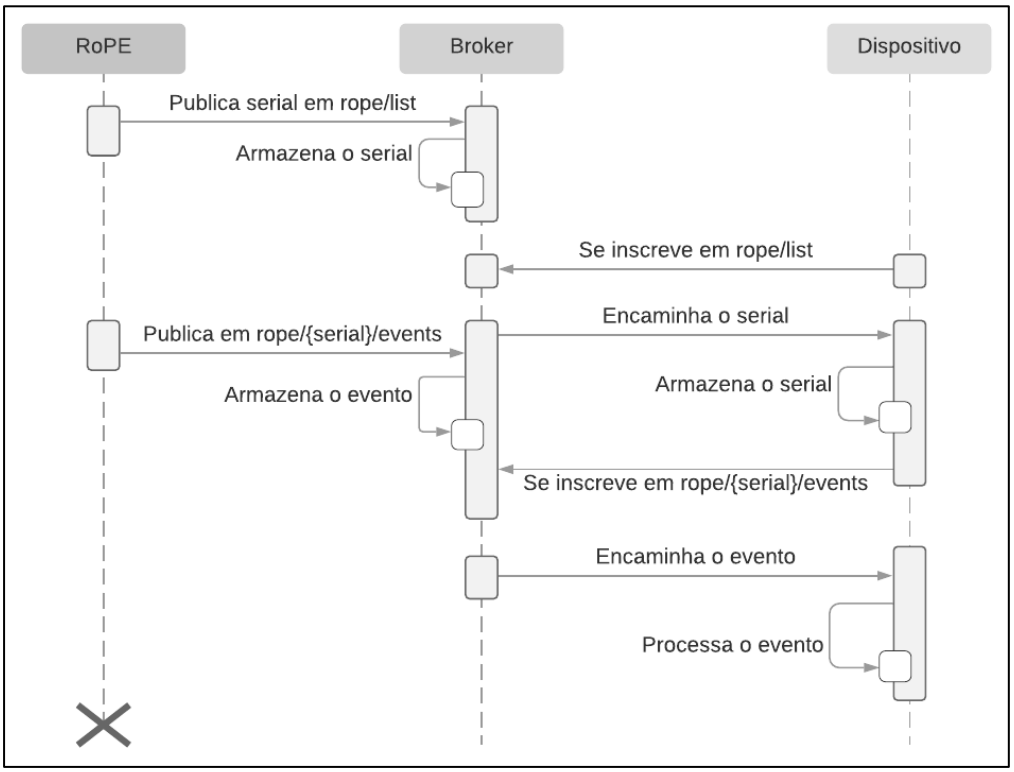


Figura 16. Sequência para o envio de eventos para os dispositivos
Fonte: O autor

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 16 apresenta um diagrama de sequência exemplificando esse processo.

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).

3.3.3 Mensagens de controle

As mensagens de controle são as mensagens usadas para enviar comandos ao RoPE a partir dos dispositivos remotos. A seguir estão relacionadas todas as mensagens de controle definidas para o projeto (Quadro 10 até Quadro 24) e os requisitos funcionais que elas implementam.

Quadro 10. Comando MOVE

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 (opcional)	A distância a ser movida (em milímetros). Se não for informada, ou for informado um valor inválido, é assumida a distância padrão de movimento do RoPE: 120mm	Float: > 0.0
speed (opcional)	A velocidade com que o RoPE deve se mover, em milímetros por segundo. Se não for informada, ou se for informado um valor inválido, o RoPE irá se mover na velocidade máxima permitida pelos motores: 54.04 mm/s	Float: > 0.0
forever (opcional)	Indica ao RoPE que ele deve mover-se indefinidamente, até que seja enviado um comando para interromper a movimentação. Este parâmetro tem prioridade sobre o parâmetro “distance”	Boolean: [true | false]
turnOnLed (opcional)	Indica ao RoPE que deve ligar um LED durante a execução do movimento. Se não for informado, assume o valor padrão: true	Boolean: [true | false]
playSound (opcional)	Indica ao RoPE que deve reproduzir uma melodia durante a execução do movimento. Se não for informado, assume o valor padrão: true	Boolean: [true | false]
Exemplos		
`{"command": "MOVE", "parameters": { "direction": "FORWARD", "distance": 10.0}}`		Move 10.0 milímetros para a frente na velocidade máxima
`{"command": "MOVE", "parameters": { "direction": "BACKWARDS", "distance": 30.0, "speed": 5.0, "playSound": true,}}`		Move 30.0 milímetros para trás na velocidade de 5 mm/s

Fonte: O autor

Quadro 11. Comando ROTATE

Mensagem de controle		
Comando	**ROTATE**	
Descrição	Rotaciona uma determinada quantidade de graus (**RF2**)	
Parâmetros		
direction	O sentido no qual o RoPE vai rotacionar	String: [CLOCKWISE | COUNTERCLOCKWISE]
degrees (opcional)	A quantidade de graus a rotacionar. Se não for informado, ou for informado um valor inválido, é assumido o ângulo de rotação padrão do RoPE: 90 graus	Float: > 0.0
speed (opcional)	A velocidade com que o RoPE deve rotacionar, em graus por segundo. Se não for informada, ou se for informado um valor inválido, o RoPE irá rotacionar na velocidade máxima permitida pelos motores	Float: > 0.0
forever (opcional)	Indica ao RoPE que ele deve rotacionar indefinidamente, até que seja enviado um comando para interromper a rotação. Este parâmetro tem prioridade sobre o parâmetro “degrees”	Boolean: [true | false]
turnOnLed (opcional)	Indica ao RoPE que deve ligar um LED durante a execução do movimento. Se não for informado, assume o valor padrão: true	Boolean: [true | false]
playSound (opcional)	Indica ao RoPE que deve reproduzir uma melodia durante a execução do movimento. Se não for informado, assume o valor padrão: true	Boolean: [true | false]
Exemplos		
`{"command": "ROTATE", "parameters": { "direction": "CLOCKWISE", "degrees": 75.0}}`		Gira 75.0 graus em sentido horário, na velocidade máxima
`{"command": " ROTATE ", "parameters": { "direction": "COUNTERCLOCKWISE", "degrees": 90.0, "speed": 30.0, "turnOnLed": false }}`		Gira 90.0 graus em sentido anti-horário, na velocidade de 30 graus por segundo

Fonte: O autor

Quadro 12. Comando STOP_ROPE

Mensagem de controle		
Comando	**STOP_ROPE**	
Descrição	Interrompe qualquer movimento do RoPE que esteja em execução	
Exemplos		
`{"command": "STOP_ROPE"}`		Interrompe imediatamente qualquer movimento do RoPE que esteja em execução, seja uma rotação ou um movimento para frente/trás

Fonte: O autor

Quadro 13. Comando TUNE_SOUND

Mensagem de controle		
Comando	**TUNE_SOUND**	
Descrição	Altera os parâmetros de reprodução de som (**RF3 e RF4**)	
Parâmetros		
state (opcional)	Alterna o estado do som entre ligado e desligado	String: [ON | OFF]
tempo (opcional)	Alterna a velocidade com que as notas são reproduzidas no *buzzer*. Pode ser usado para acelerar ou desacelerar o andamento das músicas reproduzidas. A velocidade é definida em BPM (Beats Per Minute), batidas por minuto	Int: [30..600]
volume (opcional)	Alterna o volume dos sons reproduzidos pelo RoPE. O valor é informado em um nível de 0 a 8, definidos pela biblioteca H4Plugins. O valor 0 faz com que nenhum som seja reproduzido, tendo efeito similar ao desligamento do som	Int: [0..8]
transposition (opcional)	Alterna a transposição das notas reproduzidas, fazendo com que fiquem mais graves ou mais agudas. A transposição é definida em semitons	Int: [-36..36]
Exemplos		
`{"command": " TUNE_SOUND", "parameters": { "state": "ON"}}`		Liga o som
`{"command": " TUNE_SOUND", "parameters": { "state": "OFF"}}`		Desliga o som
`{"command": "TUNE_SOUND", "parameters": { "transposition": 2}}`		Aumenta 2 semitons, deixa mais agudo A4 → B4
`{"command": "TUNE_SOUND", "parameters": { "transposition": -3}}`		Diminui 3 semitons, deixa mais grave A4 → Gb4
`{"command": "TUNE_SOUND", "parameters": { "tempo": 120, "volume": 4}}`		Ajusta a velocidade de reprodução para 120 bpm e o volume no nível 4 (50%)

Fonte: O autor

Quadro 14. Comando PLAY_SOUND

Mensagem de controle		
Comando	**PLAY_SOUND**	
Descrição	Reproduz um som no *buzzer* (**RF5**)	
Parâmetros		
melody	Define o conjunto de notas que deve ser reproduzido pelo RoPE, no formato adotado pela biblioteca H4Plugins	String
repetitions (opcional)	Define a quantidade de vezes que o som deve ser reproduzido. Se não for informado, o som é reproduzido apenas uma vez	Int: > 0
forever (opcional)	Indica que o RoPE deve reproduzir o som indefinidamente, até que seja enviado um comando para interromper a reprodução. Este parâmetro tem prioridade sobre o parâmetro “repetitions”	Boolean: [true | false]
tempo **volume** **transposition (opcionais)**	Estes parâmetros são os mesmos informados no comando TUNE_SOUND, porém afetam apenas essa reprodução de som específica, sem interferir nos demais sons reproduzidos pelo RoPE. Se estes parâmetros não forem informados, são usados os valores configurados globalmente pelo comando TUNE_SOUND	
Exemplos		

`{"command": "PLAY_SOUND", "parameters": {`
`"melody": "C#7s |DN7s |EN7q ",`
`"repetitions": 2`
`}}`
		Reproduz uma melodia 2 vezes usando as configurações de som que estão definidas globalmente

`{"command": "PLAY_SOUND", "parameters": {`
`"melody": "C#7s |DN7s |EN7q ",`
`"forever": true, "volume": 4}}`
		Reproduz a melodia indefinidamente com 50% do volume e usando as configurações de som globais para o andamento de transposição

Fonte: O autor

Quadro 15. Comando STOP_SOUND

Mensagem de controle		
Comando	**STOP_SOUND**	
Descrição	Interrompe um som que esteja sendo reproduzido pelo RoPE	
Exemplos		
`{"command": "STOP_ROPE"}`		

Fonte: O autor

Quadro 16. Comando TOGGLE_LED

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

Fonte: O autor

Quadro 17. Comando INSERT_INSTRUCTION

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 | ROTATE_CLOCKWISE | ROTATE_COUNTERCLOCKWISE]
index (opcional)	A posição da memória (começando em 0) na qual a instrução será inserida. Ao inserir, desloca para frente todas as instruções que estiverem na memória a partir do índice. Se nenhum valor for informado para o índice, insere no final da memória	Int: [0..45*]
Exemplos		
`{"command": "INSERT_INSTRUCTION", "parameters": { "instruction": "ROTATE_CLOCKWISE", "index": 2}}`		Insere a instrução “Rotacionar em sentido horário” na terceira posição da memória, movendo as demais instruções a partir desse índice para frente
`{"command": "INSERT_INSTRUCTION", "parameters": { "instruction": "MOVE_FORWARD"}}`		Insere a instrução “Mover para frente” no final da memória

Fonte: O autor

Quadro 18. Comando REMOVE_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. Ao remover, desloca todas as instruções a partir do índice para a esquerda. Se nenhum valor for informado remove a instrução que estiver no final da memória	Int: [0..45*]
Exemplos		
`{"command": "REMOVE_INSTRUCTION", "parameters": { "index": 4}}`		Remove a instrução na quinta posição da memória, deslocando as demais para a esquerda
`{"command": " REMOVE_INSTRUCTION"}`		Remove a instrução que está no final da memória

Fonte: O autor

Quadro 19. Comando REPLACE_INSTRUCTION

Mensagem de controle		
Comando	**REPLACE_INSTRUCTION**	
Descrição	Substitui uma instrução na memória do RoPE	
Parâmetros		
instruction	A instrução que será inserida na memória no lugar da instrução que está sendo removida	String: [MOVE_FORWARD | MOVE_BACKWARDS | ROTATE_CLOCKWISE | ROTATE_COUNTERCLOCKWISE]
index	A posição da memória (começando em 0) que terá a instrução substituída. Somente a instrução nesse índice da memória é afetado, as demais instruções permanecem inalteradas	Int: [0..45*]
Exemplos		
`{"command": "REPLACE_INSTRUCTION", "parameters": { "instruction": "MOVE_FORWARD", "index": 7}}`		Substitui a instrução na sétima posição da memória pela instrução “Mover para frente”

Fonte: O autor

Quadro 20. Comando RESIZE_MEMORY

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)

Fonte: O autor

Quadro 21. Comando CLEAR_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

Fonte: O autor

Quadro 22. Comando EXECUTE_PROGRAM

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

Fonte: O autor

Quadro 23. Comando PRESS_BUTTON

Mensagem de controle		
Comando	**PRESS_BUTTON**	
Descrição	Simula o pressionamento de um botão pelo usuário (**RF13**)	
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”

Fonte: O autor

Quadro 24. Comando SEND_ROBOT_STATE

Mensagem de controle	
Comando	**SEND_ROBOT_STATE**
Descrição	Requisita ao RoPE que publique seu estado atual (**RF31**)
Exemplos	
`{"command": "SEND_ROBOT_STATE"}`	Requisita ao RoPE que envie seu estado atual, contendo informações como: carga da bateria, tamanho e conteúdo da memória, configurações do som, etc.

Fonte: O autor

O processamento das mensagens de controle foi implementado da seguinte forma. Uma classe central chamada “IoTController” registra-se como *assinante* do *tema* “rope/{serial}/control”, através do método “subscribeDevice” na classe “H4P_AsyncMQTT” da biblioteca H4Plugins, e passa a receber todas as mensagens enviadas a esse *tema*. As mensagens recebidas são então validadas e convertidas para um objeto “JsonDocument” da biblioteca ArduinoJSON[19].

Uma vez nesse formato, é possível começar a extrair os dados presentes no JSON da mensagem. Primeiramente é extraído o campo “command”, o qual contém o nome do comando a ser executado. A classe “IoTController” busca então esse comando em uma lista interna que contém todos os comandos disponíveis no firmware do RoPE. Caso um comando com o nome em questão seja encontrado, um ponteiro para esse comando é obtido e ele pode então ser executado.

Todos os comandos do firmware do RoPE devem implementar a classe base “Command”, uma classe abstrata que possui um método chamado “executar”. Este método recebe como parâmetro o objeto “JsonDocument” e cada comando fica responsável por extrair e validar os campos necessários para sua própria execução. O diagrama da Figura 17, ilustra essa arquitetura, tomando como exemplo o comando “MoveCommand”, responsável por mover o RoPE para frente e para trás. Neste diagrama, os demais comandos do RoPE foram omitidos a fim de facilitar a compreensão.

[19] https://arduinojson.org/

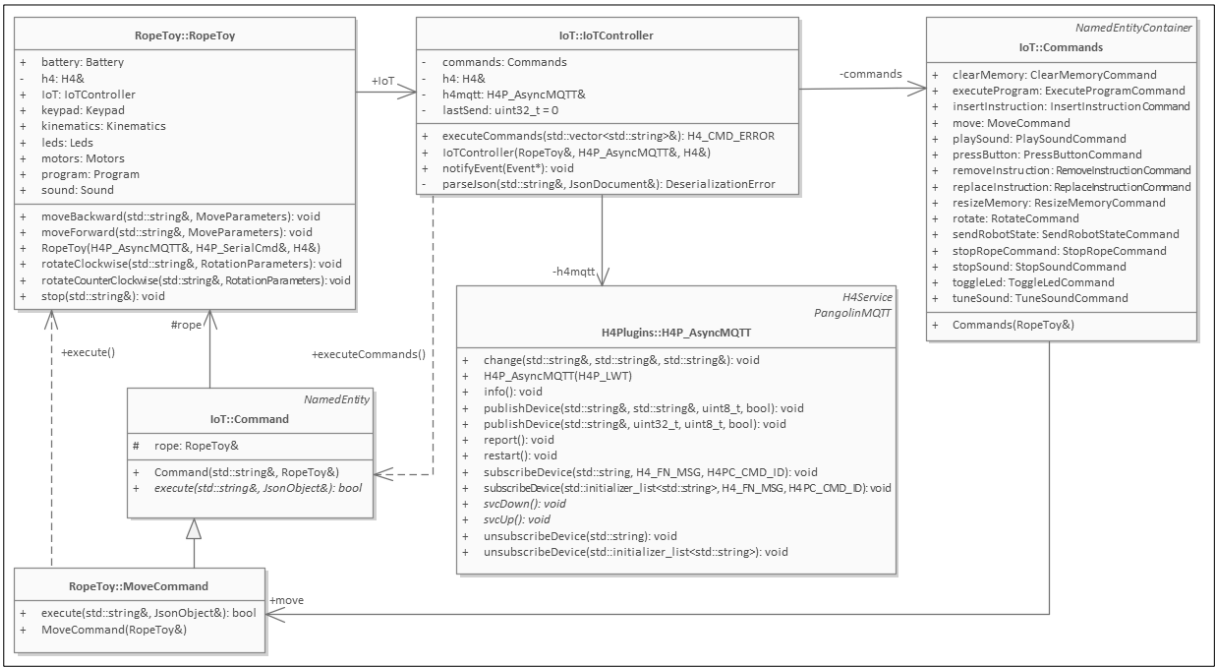


Figura 17. Arquitetura do mecanismo de envio de comandos ao RoPE

Fonte: O autor

3.3.4 Mensagens de evento

As mensagens de evento são as mensagens que o RoPE envia aos dispositivos remotos para notificá-los de que um determinado evento interno ocorreu. A seguir estão relacionadas todas as mensagens de evento definidas para o projeto (Quadro 25 até Quadro 44) e os requisitos funcionais que elas implementam.

Quadro 25. Evento MOVE_STARTED

Mensagem de evento		
Evento	**MOVE_STARTED**	
Descrição	Evento disparado quando o RoPE começa a se mover em alguma direção (**RF14**)	
Dados		
direction	A direção para a qual o RoPE se iniciou o movimento	String: [FORWARD | BACKWARDS]
distance	A distância que o RoPE deve se mover, em milímetros	Float: > 0.0
speed	A velocidade com que o RoPE iniciou o movimento, em milímetros por segundo	Float: > 0.0
duration	Quanto tempo, em segundos, que o RoPE deve levar para completar o movimento baseado na distância e velocidade	Float: > 0.0
forever	Indica se este é um movimento com distância e movimentação indeterminada, isto é, se o RoPE irá se mover até que seja interrompido	Boolean: [true | false]
Exemplos		
{"event": "MOVE_STARTED", "data": { "direction": "FORWARD", "distance": 120.0, "speed": 35.0, "duration": 3.42, "forever": false}}		Evento indicando que o RoPE iniciou um movimento para a frente, com uma velocidade de 35 mm/s. A distância a ser percorrida é de 120.0 milímetros e o tempo previsto é de 3.42 segundos
{"event": "MOVE_STARTED", "data": { "direction": "FORWARD", "speed": 50.0, "forever": true}}		Evento indicando que o RoPE iniciou um movimento para a frente, com uma velocidade de 50 mm/s. A distância e o tempo não podem ser determinados, pois o movimento foi configurado para ser executado até que seja interrompido

Fonte: O autor

Quadro 26. Evento MOVE_FINISHED

Mensagem de evento		
Evento	**MOVE_FINISHED**	
Descrição	Evento disparado quando o RoPE finalizou um movimento em alguma direção (**RF14**)	
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
speed	A velocidade com que o RoPE executou o movimento, em milímetros por segundo	Float: > 0.0
duration	Quanto tempo, em segundos, o RoPE levou para completar o movimento	Float: > 0.0
forever	Indica se foi um movimento com distância e movimentação indeterminada, até o RoPE ser interrompido	Boolean: [true | false]
finishMode	Indica se o movimento terminou de forma normal, pois foi completado, ou se foi interrompido	String: [NORMAL | ABORTED]
Exemplos		
`{"event": "MOVE_STARTED", "data": { "direction": "FORWARD", "distance": 120.0, "speed": 35.0, "duration": 3.42, "forever": false, "finishMode": "NORMAL"}}`		Evento indicando que o RoPE se moveu para a frente, com uma velocidade de 35 mm/s. A distância percorrida foi de 120.0 milímetros e o tempo gasto foi de 3.42 segundos
`{"event": "MOVE_STARTED", "data": { "direction": "FORWARD", "distance": 917.31, "speed": 50.0, "duration": 18.34, "forever": true, "finishMode": "ABORTED"}}`		Evento indicando que o RoPE iniciou um movimento para a frente, com uma velocidade de 50 mm/s. A distância percorrida foi de 917.31 e o tempo gasto foi 18.34 segundos. O movimento foi encerrado após ter sido interrompido

Fonte: O autor

Quadro 27. Evento ROTATION_STARTED

Mensagem de evento		
Evento	**ROTATION_STARTED**	
Descrição	Evento disparado quando o RoPE começa a rotacionar em algum sentido (**RF15**)	
Dados		
direction	O sentido no qual o RoPE iniciou a rotação	String: [CLOCKWISE | COUNTERCLOCKWISE]
degrees	A quantidade de graus que o RoPE deve rotacionar	Float: > 0.0
speed	A velocidade, em graus por segundo, com que o RoPE iniciou a rotação	Float: > 0.0
duration	A quantidade de tempo prevista para concluir a rotação, em segundos	Float: > 0.0
forever	Indica se a rotação vai ocorrer por tempo indeterminado, até ser interrompida	Boolean: [true | false]
Exemplos		
{"event": "ROTATION_STARTED", "data": { "direction": "CLOCKWISE", "degrees": 75.0, "speed": 17.0, "duration": 4.41, "forever": false}}		Evento indicando que o RoPE iniciou uma rotação de 75.0 graus em sentido horário. A velocidade é de 17 graus por segundo e o tempo previsto para completar a rotação é de 4.41 segundos
{"event": "ROTATION_STARTED", "data": { "direction": "CLOCKWISE", "speed": 38.0, "forever": true}}		Evento indicando que o RoPE iniciou uma rotação no sentido horário, na velocidade de 38 graus por segundo. A distância e o tempo são indeterminados, pois a rotação foi configurada para executar até que seja interrompida

Fonte: O autor

Quadro 28. Evento ROTATION_FINISHED

Mensagem de evento		
Evento	**ROTATION_FINISHED**	
Descrição	Evento disparado quando o RoPE termina de rotacionar em algum sentido (**RF15**)	
Dados		
direction	O sentido no qual o RoPE realizou a rotação	String: [CLOCKWISE | COUNTERCLOCKWISE]
degrees	A quantidade de graus que o RoPE rotacionou	Float: > 0.0
speed	A velocidade, em graus por segundo, com que o RoPE realizou a rotação	Float: > 0.0
duration	A quantidade de tempo que o RoPE levou para concluir a rotação, em segundos	Float: > 0.0
forever	Indica se a rotação ocorreu por tempo indeterminado, até ser interrompida	Boolean: [true | false]
finishMode	Indica se a rotação terminou de forma normal, pois foi completada, ou se foi interrompida	String: [NORMAL | ABORTED]
Exemplos		
`{"event": "ROTATION_FINISHED", "data": { "direction": "CLOCKWISE", "degrees": 75.0, "speed": 17.0, "duration": 4.41, "forever": false}}`		Evento indicando que o RoPE realizou uma rotação de 75.0 graus em sentido horário. A velocidade foi de 17 graus por segundo e o tempo gasto para completar a rotação foi de 4.41 segundos
`{"event": "ROTATION_FINISHED", "data": { "direction": "CLOCKWISE", "degrees": 101.49, "speed": 38.0, "duration": 2.67, "forever": true, "finishMode": "ABORTED"}}`		Evento indicando que o RoPE realizou uma rotação no sentido horário, na velocidade de 38 graus por segundo. Foram rotacionados 101.49 graus e o tempo gasto foi de 2.67 segundos. A rotação finalizou após ter sido interrompida

Fonte: O autor

Quadro 29. Evento BATTERY_STATUS

Mensagem de evento		
Evento	**BATTERY_STATUS**	
Descrição	Evento disparado quando a carga da bateria se altera (**RF16**)	
Dados		
level	O nível de carga da bateria, em percentual	Float: [0.0..1.0]
state	O estado da bateria com relação à sua carga	String: [FULL | EMPTY | CRITICAL | DISCHARGING]
critical_treshold	O limiar de carga antes da bateria entrar em estado crítico, em percentual	Float: [0.0..1.0]
voltage	O nível de tensão atual da bateria, tanto em volts quanto em milivolts. Em condições normais a bateria opera entre 3.1 e 3.7 volts	Float: [0.0..3.7] Float: [0.0..3700.00]
Exemplo		

{"event": "BATTERY_STATUS", "data": {
"state": "DISCHARGING", "level": 0.63,
"critical_treshold": 0.05, "voltage": {
"volts": 3.49, "millivolts": 3490.0}}}
	Evento indicando que o nível de carga da bateria é de 63.0%. O nível está acima do limiar de bateria crítica (5.0%), portanto a bateria não precisa ser recarregada no momento. A tensão da bateria é de 3.49 volts	

{"event": "BATTERY_STATUS", "data": {
"state": "CRITICAL", "level": 0.02,
"critical_treshold": 0.05, "voltage": {
"volts": 3.14, "millivolts": 3140.0}}}
	Evento indicando que o nível de carga da bateria é de apenas 2%. O nível está abaixo do limiar de bateria crítica (5.0%), portanto a bateria precisa ser recarregada imediatamente. A tensão da bateria é de 3.14 volts, muito próximo à tensão mínima de operação	

Fonte: O autor

Quadro 30. Evento SOUND_TUNED

Mensagem de evento		
Evento	**SOUND_TUNED**	
Descrição	Evento disparado quando os parâmetros de reprodução de som são alterados (**RF18**)	
Dados		
state **tempo** **volume** **transposition**	O evento pode conter qualquer um dos parâmetros definidos com o Comando TUNE_SOUND	
Exemplo		
`{"event": "SOUND_TUNED", "data": {` `  "volume": 2, "tempo": 240}}`		Evento indicando que o volume foi alterado para o nível 2 e a velocidade de reprodução para 240 bpm

Fonte: O autor

Quadro 31. Evento SOUND_PLAYBACK_STARTED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">SOUND_PLAYBACK_STARTED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando o RoPE começa a reproduzir um som (RF19)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>melody
volume
tempo
transposition
repetitions
forever</td><td>O evento pode conter qualquer um dos parâmetros definidos com o Comando PLAY_SOUND</td><td></td></tr>
<tr><td>index</td><td>Se a reprodução do som foi configurada para repetir mais de uma vez ou indefinidamente, este parâmetro indica qual é o índice desta reprodução, iniciando em 0</td><td>Int: >= 0</td></tr>
<tr><th colspan="3">Exemplos</th></tr>
<tr><td colspan="2"><pre>{"event": "SOUND_PLAYBACK_STARTED", "data": {
   "melody": "C#7s |DN7s |EN7q ",
   "repetitions": 5, "index": 1}}</pre></td><td>Evento indicando que a foi iniciada a segunda reprodução de uma melodia que foi programada para repetir cinco vezes. Foram usadas as configurações de som globais</td></tr>
<tr><td colspan="2"><pre>{"event": "SOUND_PLAYBACK_STARTED", "data": {
   "melody": "C#7s |DN7s |EN7q ",
   "repetitions": 3, "index": 2, "volume": 2}}</pre></td><td>Evento indicando que a foi iniciada a última reprodução de uma melodia que foi programada para repetir três vezes. Foram usadas as configurações de som globais, com exceção do volume, que foi definido com o nível 2 (25%)</td></tr>
</table>

Fonte: O autor

Quadro 32. Evento SOUND_PLAYBACK_FINISHED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">SOUND_PLAYBACK_FINISHED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando o RoPE termina de reproduzir um som (RF19)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>melody
volume
tempo
transposition
repetitions
forever</td><td>O evento pode conter qualquer um dos parâmetros definidos com o Comando PLAY_SOUND</td><td></td></tr>
<tr><td>index</td><td>Se a reprodução do som foi configurada para repetir mais de uma vez ou indefinidamente, este parâmetro indica qual é o índice desta reprodução, iniciando em 0</td><td>Int: >= 0</td></tr>
<tr><td>finishMode</td><td>Indice se a reprodução do som finalizou de forma normal, pois foi completada, ou se foi interrompida</td><td>String: [NORMAL | ABORTED]</td></tr>
<tr><th colspan="3">Exemplos</th></tr>
<tr><td colspan="2"><pre>{"event": "SOUND_PLAYBACK_FINISHED", "data": {
   "melody": "C#7s |DN7s |EN7q ",
   "repetitions": 5, "index": 1,
   "finishMode": "NORMAL"}}</pre></td><td>Evento indicando que a foi finalizada a segunda reprodução de uma melodia que foi programada para repetir cinco vezes. Foram usadas as configurações de som globais. A reprodução finalizou de forma normal</td></tr>
<tr><td colspan="2"><pre>{"event": "SOUND_PLAYBACK_FINISHED", "data": {
   "melody": "C#7s |DN7s |EN7q ",
   "repetitions": 3, "index": 0, "volume": 2,
   "finishMode": "ABORTED"}}</pre></td><td>Evento indicando que a foi finalizada a última reprodução de uma melodia que foi programada para repetir três vezes. Foram usadas as configurações de som globais, com exceção do volume, que foi definido com o nível 2 (25%). A reprodução finalizou porque foi interrompida, indicando que não as próximas duas reproduções não ocorrerão</td></tr>
</table>

Fonte: O autor

Quadro 33. Evento LED_TOGGLED

Mensagem de evento		
Evento	**LED_TOGGLED**	
Descrição	Evento disparado quando um LED é ligado ou desligado (**RF20**)	
Dados		
led	O LED que foi alternado	String: [FRONT | LEFT | RIGHT | BACK | CENTER]
state	O estado atual do LED	String: [ON | OFF]
Exemplos		
`{"event": "LED_TOGGLED ", "data": { "led": "FRONT", "state": "ON"}}`		Evento indicando que o LED frontal foi ligado
`{"event": "LED_TOGGLED", "data": { "led": "BACK", "state": "OFF"}}`		Evento indicando que o LED traseiro foi desligado

Fonte: O autor

Quadro 34. Evento INSTRUCTION_INSERTED

<table>
<tr><th colspan="3">Mensagem de Evento</th></tr>
<tr><td>Evento</td><td colspan="2">INSTRUCTION_ INSERTED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando uma instrução é inserida na memória do RoPE (RF21)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>instruction</td><td>A instrução que foi inserida na memória</td><td>String: [MOVE_FORWARD | MOVE_BACKWARDS | ROTATE_CLOCKWISE | ROTATE_COUNTERCLOCKWISE]</td></tr>
<tr><td>index</td><td>A posição da memória (começando em 0) na qual a instrução foi inserida</td><td>Int: [0..?]</td></tr>
<tr><td>memory</td><td>Objeto contendo informações úteis sobre a memória</td><td>Object</td></tr>
<tr><th colspan="3">Exemplo</th></tr>
<tr><td colspan="2"><pre>{"event": "INSTRUCTION_INSERTED", "data": {
    "instruction": "ROTATE_CLOCKWISE",
    "index": 0,
       "memory": {
          "previousState": {
             "size": 4, "used": 1, "free": 3,
             "insertions": 1, "removals": 0,
             "cleanings": 0, "resizings": 0,
             "content": [
                "MOVE_FORWARD"]},
          "currentState": {
             "size": 4, "used": 2, "free": 2,
             "insertions": 2, "removals": 0,
             "cleanings": 0, "resizings": 0,
             "content": [
                "ROTATE_CLOCKWISE",
                "MOVE_FORWARD"]}}}}</pre></td><td>Evento indicando que a instrução “Rotacionar em sentido horário” foi inserida na primeira posição da memória. A memória, que continha apenas 1 instrução “Mover para frente”, agora tem 2 instruções</td></tr>
</table>

Fonte: O autor

Quadro 35. Evento INSTRUCTION_REMOVED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">INSTRUCTION_REMOVED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando uma instrução é removida da memória do RoPE (RF22)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>instruction</td><td>A instrução que foi removida da memória</td><td>String: [MOVE_FORWARD | MOVE_BACKWARDS | ROTATE_CLOCKWISE | ROTATE_COUNTERCLOCKWISE]</td></tr>
<tr><td>index</td><td>A posição da memória (começando em 0) da qual a instrução foi removida</td><td>Int: [0..?]</td></tr>
<tr><td>memory</td><td>Objeto contendo informações úteis sobre a memória</td><td>Object</td></tr>
<tr><th colspan="3">Exemplo</th></tr>
<tr><td colspan="2"><pre>{"event": "INSTRUCTION_REMOVED", "data": {
    "instruction": "MOVE_FORWARD",
    "index": 0,
       "memory": {
          "previousState": {
             "size": 4, "used": 2, "free": 2,
             "insertions": 2, "removals": 0,
             "cleanings": 0, "resizings": 0,
             "content": [
                "MOVE_FORWARD",
                "ROTATE_CLOCKWISE"]},
          "currentState": {
             "size": 4, "used": 1, "free": 3,
             "insertions": 2, "removals": 1,
             "cleanings": 0, "resizings": 0,
             "content": [
                "ROTATE_CLOCKWISE"]}}}}</pre></td><td>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 “Rotacionar em sentido horário”</td></tr>
</table>

Fonte: O autor

Quadro 36. Evento INSTRUCTION_REPLACED

Mensagem de evento		
Evento	**INSTRUCTION_REPLACED**	
Descrição	Evento disparado quando uma instrução é substituída na memória do RoPE (**RF21 e RF22**)	
Dados		
instruction	A instrução que foi inserida na memória em substituição à instrução existente	String: [MOVE_FORWARD | MOVE_BACKWARDS | ROTATE_CLOCKWISE | ROTATE_COUNTERCLOCKWISE]
index	A posição da memória (começando em 0) que teve a instrução substituída	Int: [0..?]
memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		
{"event": "INSTRUCTION_REPLACED", "data": { "instruction": "MOVE_FORWARD", "index": 1, "memory": { "previousState": { "size": 4, "used": 2, "free": 2, "insertions": 2, "removals": 0, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "ROTATE_CLOCKWISE"]}, "currentState": { "size": 4, "used": 1, "free": 3, "insertions": 3, "removals": 1, "cleanings": 0, "resizings": 0, "content": ["MOVE_FORWARD", "MOVE_FORWARD"]}}}}	Evento indicando que a segunda posição da memória teve a instrução substituída para “Mover para frente”. A memória, que continha 2 instruções, continua tendo 2 instruções, porém seu conteúdo é diferente	

Fonte: O autor

Quadro 37. Evento MEMORY_RESIZED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">MEMORY_RESIZED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando a memória do RoPE é redimensionada (RF23)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>memory</td><td>Objeto contendo informações úteis sobre a memória</td><td>Object</td></tr>
<tr><th colspan="3">Exemplo</th></tr>
<tr><td colspan="2"><pre>{"event": "MEMORY_RESIZED", "data": {
    "memory": {
       "previousState": {
          "size": 2, "used": 1, "free": 1,
          "insertions": 1, "removals": 0,
          "cleanings": 0, "resizings": 0,
          "content": [
             "MOVE_FORWARD"]},
       "currentState": {
          "size": 4, "used": 1, "free": 3,
          "insertions": 1, "removals": 0,
          "cleanings": 0, "resizings": 1,
          "content": [
             "MOVE_FORWARD"]}}}}</pre></td><td>Evento indicando que a memória foi redimensionada de 2 para 4 instruções</td></tr>
</table>

Fonte: O autor

Quadro 38. Evento MEMORY_CLEARED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">MEMORY_CLEARED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando a memória do RoPE é apagada (RF24)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>memory</td><td>Objeto contendo informações úteis sobre a memória</td><td>Object</td></tr>
<tr><th colspan="3">Exemplo</th></tr>
<tr><td colspan="2"><pre>{"event": "MEMORY_CLEARED", "data": {
    "memory": {
       "previousState": {
          "size": 4, "used": 2, "free": 2,
          "insertions": 2, "removals": 0,
          "cleanings": 0, "resizings": 0,
          "content": [
             "TURN_RIGHT",
             "TURN_LEFT"]},
       "currentState": {
          "size": 4, "used": 0, "free": 4,
          "insertions": 2, "removals": 0,
          "cleanings": 1, "resizings": 0,
          "content": []
        }}}}</pre></td><td>Evento indicando que a memória foi apagada, liberando 2 posições que estavam ocupadas por instruções</td></tr>
</table>

Fonte: O autor

Quadro 39. Evento PROGRAM_STARTED

Mensagem de evento		
Evento	**PROGRAM_STARTED**	
Descrição	Evento disparado quando o RoPE inicia a execução de um programa (**RF25**)	
Dados		
execution	Objeto contendo informações sobre a execução atual	Object
memory	Objeto contendo informações úteis sobre a memória	Object
Exemplo		

{"event": "PROGRAM_STARTED ", "data": {
"execution": {
"progress": 0.0, "statistics": {
"started": 2,
"finished": 1,
"aborted": 0}},
"memory": {
"currentState": {
"size": 3, "used": 3, "free": 0,
"insertions": 3, "removals": 0,
"cleanings": 0, "resizings": 0,
"content": [
"MOVE_FORWARD",
"ROTATE_CLOCKWISE",
"MOVE_BACKWARDS"]}}}}
		Evento indicando que um programa foi iniciado

Fonte: O autor

Quadro 40. Evento INSTRUCTION_STARTED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">INSTRUCTION_STARTED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando uma instrução do programa é executada (RF26)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>execution</td><td>Objeto contendo informações sobre a execução atual</td><td>Object</td></tr>
<tr><td>memory</td><td>Objeto contendo informações úteis sobre a memória</td><td>Object</td></tr>
<tr><th colspan="3">Exemplo</th></tr>
<tr><td colspan="2"><pre>{"event": "INSTRUCTION_STARTED", "data": {
    "instruction": "ROTATE_CLOCKWISE",
    "index": 1,
    "execution": {
       "progress": 33.0,
       "statistics": {
          "started": 2,
          "finished": 1,
          "aborted": 0}},
    "memory": {
       "currentState": {
          "size": 3, "used": 3, "free": 0,
          "insertions": 3, "removals": 0,
          "cleanings": 0, "resizings": 0,
          "content": [
             "MOVE_FORWARD",
             "ROTATE_CLOCKWISE",
             "MOVE_BACKWARDS"]}}}}</pre></td><td>Evento indicando que a segunda instrução do programa começou a ser executada. Como a segunda instrução ainda está no início, o progresso é de 33%</td></tr>
</table>

Fonte: O autor

Quadro 41. Evento INSTRUCTION_FINISHED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">INSTRUCTION_FINISHED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando uma instrução do programa é finalizada (RF26)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>execution</td><td>Objeto contendo informações sobre a execução atual</td><td>Object</td></tr>
<tr><td>memory</td><td>Objeto contendo informações úteis sobre a memória</td><td>Object</td></tr>
<tr><td>finishMode</td><td>Indica se a instrução finalizou de forma normal, pois foi completada, ou se foi interrompida</td><td>String: [NORMAL | ABORTED]</td></tr>
<tr><th colspan="3">Exemplo</th></tr>
<tr><td colspan="2"><pre>{"event": "INSTRUCTION_STARTED", "data": {
    "instruction": "ROTATE_CLOCKWISE",
    "index": 1,
    "finishMode": "NORMAL",
    "execution": {
       "progress": 66.0,
       "statistics": {
          "started": 2,
          "finished": 2,
          "aborted": 0}},
    "memory": {
       "currentState": {
          "size": 3, "used": 3, "free": 0,
          "insertions": 3, "removals": 0,
          "cleanings": 0, "resizings": 0,
          "content": [
             "MOVE_FORWARD",
             "ROTATE_CLOCKWISE",
             "MOVE_BACKWARDS"]}}}}</pre></td><td>Evento indicando que a segunda instrução do programa terminou de ser executada, representando um progresso de 66.0% na execução do programa</td></tr>
</table>

Fonte: O autor

Quadro 42. Evento PROGRAM_FINISHED

<table>
<tr><th colspan="3">Mensagem de evento</th></tr>
<tr><td>Evento</td><td colspan="2">PROGRAM_FINISHED</td></tr>
<tr><td>Descrição</td><td colspan="2">Evento disparado quando um programa é finalizado (RF27)</td></tr>
<tr><th colspan="3">Dados</th></tr>
<tr><td>finishMode</td><td>Indica se o programa finalizou de forma normal, pois foi completado, ou se foi interrompido</td><td>String: [NORMAL | ABORTED]</td></tr>
<tr><td>execution</td><td>Objeto contendo informações sobre a execução atual</td><td>Object</td></tr>
<tr><td>memory</td><td>Objeto contendo informações úteis sobre a memória</td><td>Object</td></tr>
<tr><th colspan="3">Exemplos</th></tr>
<tr><td colspan="2"><pre>{"event": "PROGRAM_FINISHED", "data": {
    "finishMode": "ABORTED", "execution": {
       "progress": 33.0,
       "statistics": {
          "started": 1,
          "finished": 0,
          "aborted": 1}},
    "memory": {
       "currentState": {
          "size": 3, "used": 3, "free": 0,
          "insertions": 3, "removals": 0,
          "cleanings": 0, "resizings": 0,
          "content": [
             "MOVE_FORWARD",
             "TURN_RIGHT",
             "TURN_LEFT"]}}}}</pre></td><td>Evento indicando que o programa finalizou após ser abortado enquanto executava a primeira instrução. O programa completou apenas 33.0% da execução</td></tr>
<tr><td colspan="2"><pre>{"event": "PROGRAM_FINISHED", "data": {
    "finishMode": "NORMAL", "execution": {
       "progress": 100.0,
       "statistics": {
          "started": 3,
          "finished": 3,
          "aborted": 0}},
    "memory": {
       "currentState": {
          "size": 3, "used": 3, "free": 0,
          "insertions": 3, "removals": 0,
          "cleanings": 0, "resizings": 0,
          "content": [
             "MOVE_FORWARD",
             "TURN_RIGHT",
             "TURN_LEFT"]}}}}</pre></td><td>Evento indicando que o programa finalizou após ter completado a execução corretamente, ou seja, após ter executado 100.0% das instruções</td></tr>
</table>

Fonte: O autor

Quadro 43. Evento BUTTON_PRESSED

Mensagem de evento		
Evento	**BUTTON_PRESSED**	
Descrição	Evento disparado quando um botão do RoPE é pressionado (**RF28**)	
Dados		
button	O botão que foi pressionado	String: [FRONT | BACK | LEFT | RIGHT | CENTER]
Exemplos		
`{"event": " BUTTON_PRESSED", "data": { "button": "FRONT"}}`		Evento indicando que o botão frontal foi pressionado
`{"event": "BUTTON_PRESSED", "data": { "button": "CENTER"}}`		Evento indicando que o botão central foi pressionado

Fonte: O autor

Quadro 44. Evento ROBOT_STATE

Mensagem de evento		
Evento	**ROBOT_STATE**	
Descrição	Evento disparado quando o RoPE é ligado ou tem seu estado requisitado (**RF29**)	
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": "ROBOT_STATE", "data": {
"battery":{
"level": 0.75, "critical_treshold": 0.05},
"memory": {
"size": 3, "used": 0, "free": 3},
"sound":{
"state": "ON", "trasposition": 0}
}
	Evento disparado quando o RoPE é ligado ou tem seu estado requisitado. A mensagem contém as configurações atuais do brinquedo	

Fonte: O autor

O envio das mensagens de evento foi implementado da seguinte forma. Cada evento que ocorre dentro do RoPE e que precisa ser enviado via MQTT deve estender uma classe base abstrata chamada “Event”. Essa classe possui um método chamado “fillData” que recebe por parâmetro um objeto “JsonObject” da biblioteca ArduinoJSON[20]. As classes concretas devem sobrescrever esse método e preencher o objeto JSON com as informações referentes ao evento ocorrido. Por exemplo, a classe “RotationStartedEvent” irá preencher os campos “direction”, ”speed”, “degrees”, “duration” e “forever”, conforme estipulado no formato de mensagem desse evento (ver Quadro 27).

Quando um evento ocorre no RoPE, um objeto da classe concreta referente ao evento é instanciado e submetido à classe central “IoTController” através do método “notifyEvent”. Essa classe cria o objeto JSON citado anteriormente e chama então o método “fillData” do evento para

[20] https://arduinojson.org/

que os dados sejam preenchidos no JSON. Após o preenchimento, o envio da mensagem é realizado através do método “publilshDevice” na classe “H4P_AsyncMQTT” da biblioteca H4Plugins.

O diagrama da Figura 18, ilustra essa arquitetura, tomando como exemplo o movimento de rotação realizado pelo RoPE, o qual gera dois eventos que serão submetidos à classe “IoTController”: “RotationStartedEvent” e “RotationFinishedEvent”. Neste diagrama, os demais eventos do RoPE foram omitidos a fim de facilitar a compreensão. Os diagramas contendo os demais eventos e a arquitetura do restante do projeto podem ser visualizados nos apêndices A até G, no final do trabalho.

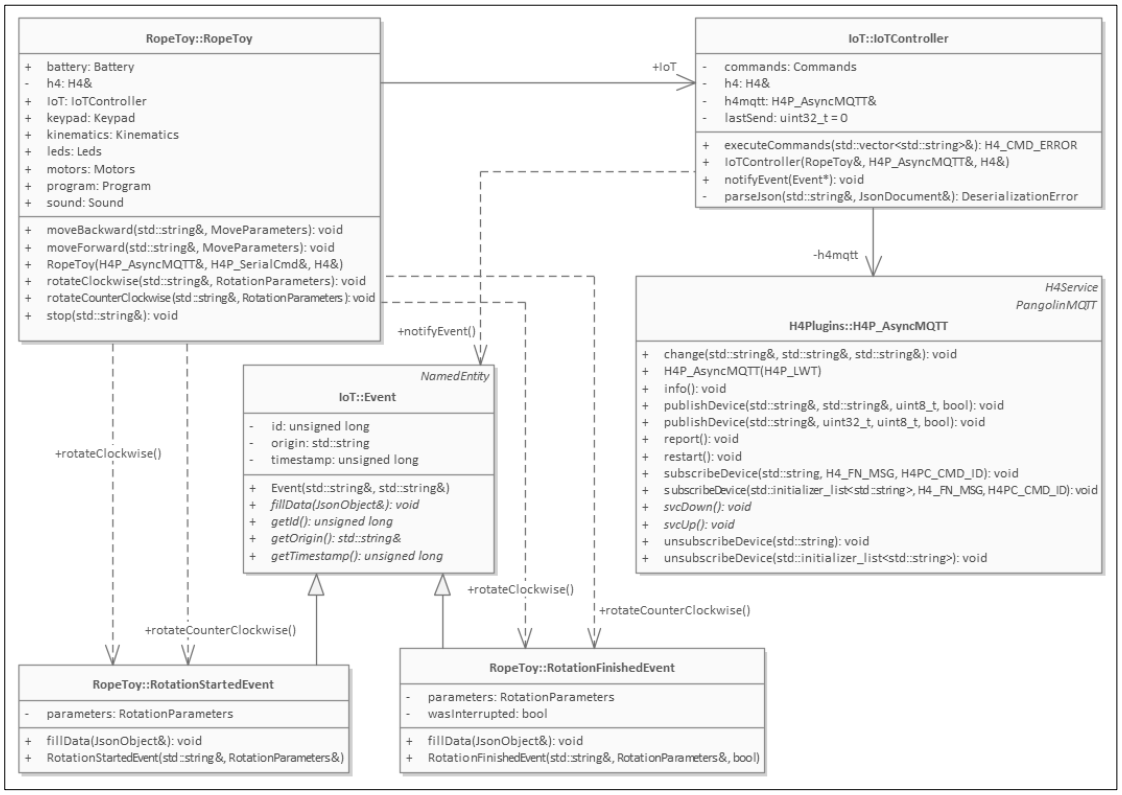


Figura 18. Arquitetura do mecanismo de envio de eventos do RoPE

Fonte: O autor

4 RESULTADOS

Como resultados deste trabalho foram gerados dois produtos. O primeiro produto é uma aplicação interativa denominada SmartRoPE, a qual permite interagir com o RoPE visualizando sua movimentação e enviando comandos para serem executados nele. O segundo produto, integrado à aplicação SmartRoPE, é uma ferramenta que permite usar o RoPE para auxiliar na coleta de dados para pesquisa e avaliação do pensamento computacional. Ambos os produtos foram desenvolvidos com o objetivo de testar e validar o funcionamento do novo firmware do RoPE. Neste capítulo são apresentadas essas ferramentas.

4.1 APLICAÇÃO SMARTROPE

Ao final da implementação do projeto foi desenvolvida uma aplicação para testar de forma prática o envio dos eventos e recebimento dos comandos pelo RoPE. A aplicação desenvolvida possui duas telas. A tela inicial (Figura 19), consiste em uma lista de todas as unidades do RoPE que estão ligadas e conectadas à Internet. As unidades estão identificadas pelo seu número de série e contém informações de contato do proprietário atual do brinquedo. Um campo de pesquisa, permite 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 19. Tela inicial da aplicação SmartRoPE
Fonte: O autor

Ao clicar ema unidade do RoPE na tela principal, a aplicação é redirecionada à tela de testes da unidade. A tela se divide em duas seções. A primeira seção contém abas com controles que

permitem enviar comandos ao RoPE e monitorar os eventos que estão sendo recebidos do Robô. A segunda seção consiste em um painel que permite acompanhar a movimentação do brinquedo em “tempo real”, ambos são detalhados a seguir.

A primeira aba disponível na seção de controles é a aba “Som”, ilustrada na Figura 20. Na parte superior encontram-se controles que permitem ajustar os parâmetros de reprodução de som como, volume, velocidade e transposição, conforme detalhado no capítulo **3.2.1 Plugin de som**. Em seguida, estão posicionados botões que permitem enviar ao brinquedo músicas para serem tocadas. As melodias não estão pré-programadas no brinquedo, mas sim na aplicação SmartRoPE. Ao clicar em um dos botões, as notas da música são enviadas ao RoPE usando o comando “PLAY_SOUND”, que as reproduz na sequência correta. Por fim, há um componente de teclado musical que faz uso do mesmo comando “PLAY_SOUND”, mas para enviar notas musicais isoladas ao brinquedo. Esse componente permite ao usuário reproduzir uma melodia de composição própria no RoPE.

Todos os controles dessa aba reagem aos eventos do RoPE para refletir o estado atual do brinquedo. Por exemplo, se o volume do som é diminuído diretamente no RoPE, o evento “SOUND_TUNED” é gerado pelo brinquedo e capturado na aplicação, fazendo com que o controle de volume na tela da aplicação se ajuste ao novo valor. Da mesma forma, se uma melodia é reproduzida no RoPE devido ao pressionamento de um botão, por exemplo, o evento “SOUND_PLAYBACK_STARTED” é gerado pelo brinquedo e capturado na aplicação, fazendo com que as notas musicais tocadas no RoPE sejam destacadas no componente de teclado.

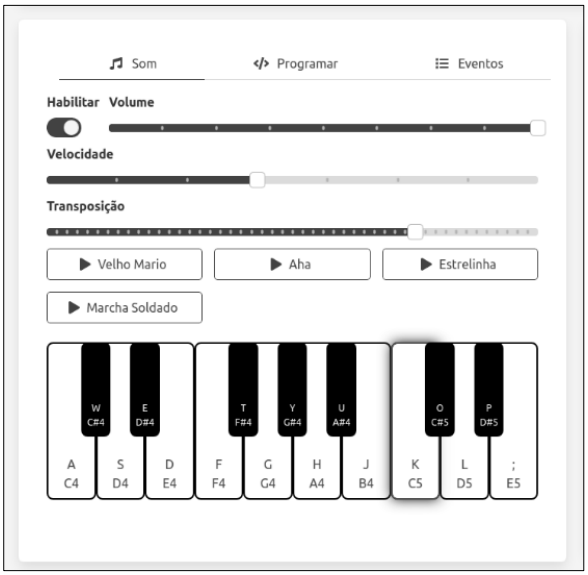


Figura 20. Controles de som da aplicação SmartRoPE

A segunda aba disponível na seção de controles é a aba “Programar”, ilustrada na Figura 21. Na parte superior encontram-se cinco botões que simulam um teclado virtual para o RoPE, seguindo a mesma disposição espacial e as mesmas cores do brinquedo. Ao clicar em um dos botões a aplicação gera e envia ao robô um comando “PRESS_BUTTON”. Ao receber o comando, o RoPE reage gerando um pressionamento de botão exatamente da mesma forma que ocorreria se um usuário o tivesse pressionado fisicamente. Da mesma forma, o teclado virtual reage aos pressionamentos físicos dos botões no RoPE, destacando na tela qual botão foi pressionado pelo usuário.

Na parte inferior da aba há um painel que exibe o conteúdo atual da memória do RoPE, isto é, quais instruções estão programadas para o RoPE realizar quando o botão central for pressionado. O painel também possui uma barra de progresso que permite visualizar a capacidade da memória e quantidade de instruções que foram programadas. Ao clicar em uma instrução no painel, um comando “REMOVE_INSTRUCTION” é enviado ao RoPE, removendo a instrução em questão da memória. Esta funcionalidade permite corrigir rapidamente instruções que tenham sido programadas de forma equivocada. Por fim, um botão permite realizar a limpeza completa da memória apagando todas as instruções de uma única vez, através do comando “CLEAR_MEMORY”.

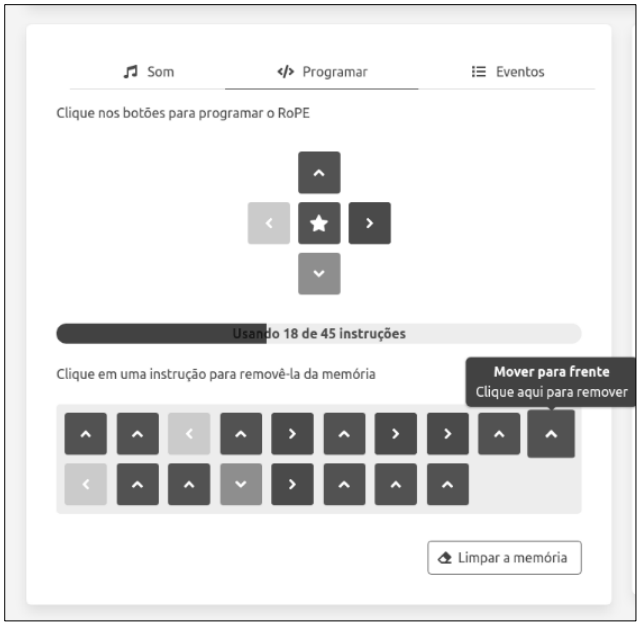


Figura 21. Controles de programação da aplicação SmartRoPE

Assim como anteriormente, a aplicação monitora todos os eventos do RoPE relacionados ao conteúdo da memória e atualiza a lista de instruções a cada alteração, independente se ela foi realizada

diretamente no RoPE pelo usuário, ao clicar em um botão, por exemplo, ou se foi realizada pelo próprio aplicativo através do teclado virtual.

A terceira aba disponível é a aba “Eventos”, ilustrada na Figura 22. Na parte inferior da aba há uma lista que registra todos os eventos que a aplicação recebe do RoPE via MQTT, com o nome do evento e a data e hora em que ocorreram. Os eventos mais recentes se encontram na parte superior da lista, enquanto os mais antigos se encontram na parte inferior e vão sendo descartados conforme o número de eventos atinge o limite estipulado. Ao clicar em um evento, é revelado o *payload* JSON que deu origem a ele.

Na parte superior da aba há um controle que permite ajustar a quantidade de eventos que serão registrados, variando de 5 até 20 eventos. Esses valores foram escolhidos arbitrariamente e podem ser ajustados posteriormente na aplicação conforme a necessidade. É importante notar que esta lista serve apenas para visualização dos eventos, pois todos os eventos gerados pelo RoPE continuam sendo capturados e processados pela aplicação em segundo plano, independentemente da quantidade definida nesta tela.

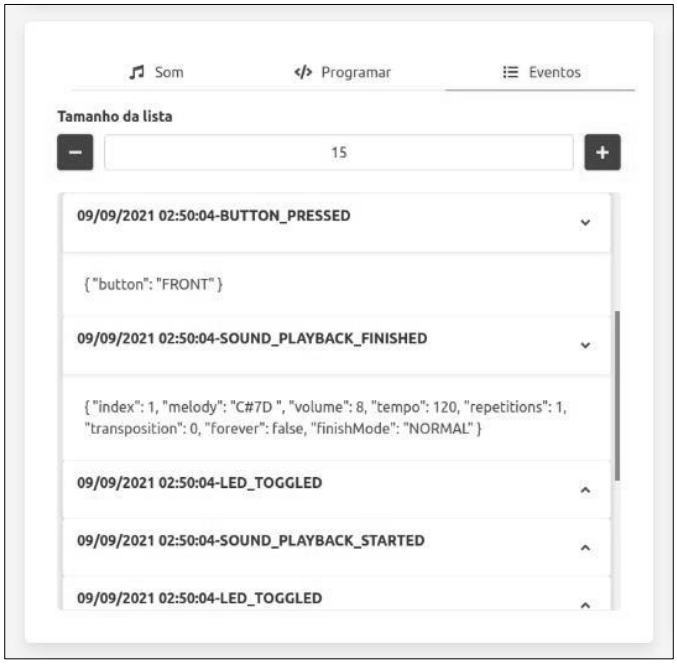


Figura 22. Lista de eventos da aplicação SmartRoPE

Na segunda seção da aplicação há um painel de visualização que permite acompanhar em “tempo real” a movimentação do RoPE em um modelo 3D. O modelo é animado de forma a representar de forma mais fiel possível os movimentos reais, respeitando a velocidade, distância e

ângulo que o robô percorre no mundo real. Para isso, a aplicação monitora os eventos do RoPE referentes à movimentação: “MOVE_STARTED” e “MOVE_FINISHED”, e os eventos referentes à rotação: “ROTATION_STARTED” e “ROTATION_FINISHED”.

A renderização 3D possui vários modos de câmera que podem ser usados para realizar a observação do brinquedo. O primeiro modo de câmera é o modo em 1ª pessoa, ilustrado na Figura 23. Nesse modo, a câmera é posicionada de forma a simular o ponto de vista do RoPE. O segundo modo de câmera disponível é o modo em 3ª pessoa, ilustrado na Figura 24. Nesse modo, a câmera é posicionada atrás do RoPE, permitindo visualizar os detalhes do modelo 3D e do cenário. Conforme o RoPE se movimenta pelo cenário a câmera acompanha sua movimentação. Por fim, há o modo de Visão Superior, ilustrado na Figura 25. Nesse modo, a câmera é posicionada de forma fixa sobre o cenário, permitindo visualizar de forma mais ampla a movimentação e a localização do RoPE.

Figura 23. Visualização do RoPE em 1ª pessoa na aplicação SmartRoPE

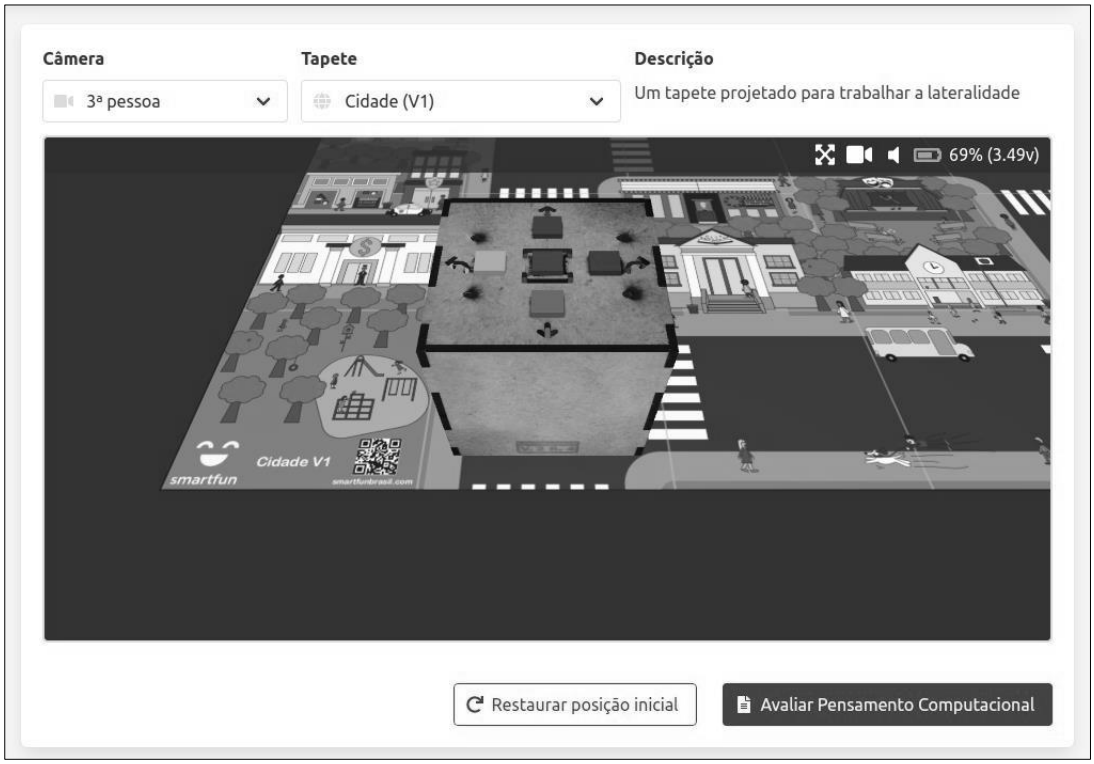


Figura 24. Visualização do RoPE em 3ª pessoa na aplicação SmartRoPE

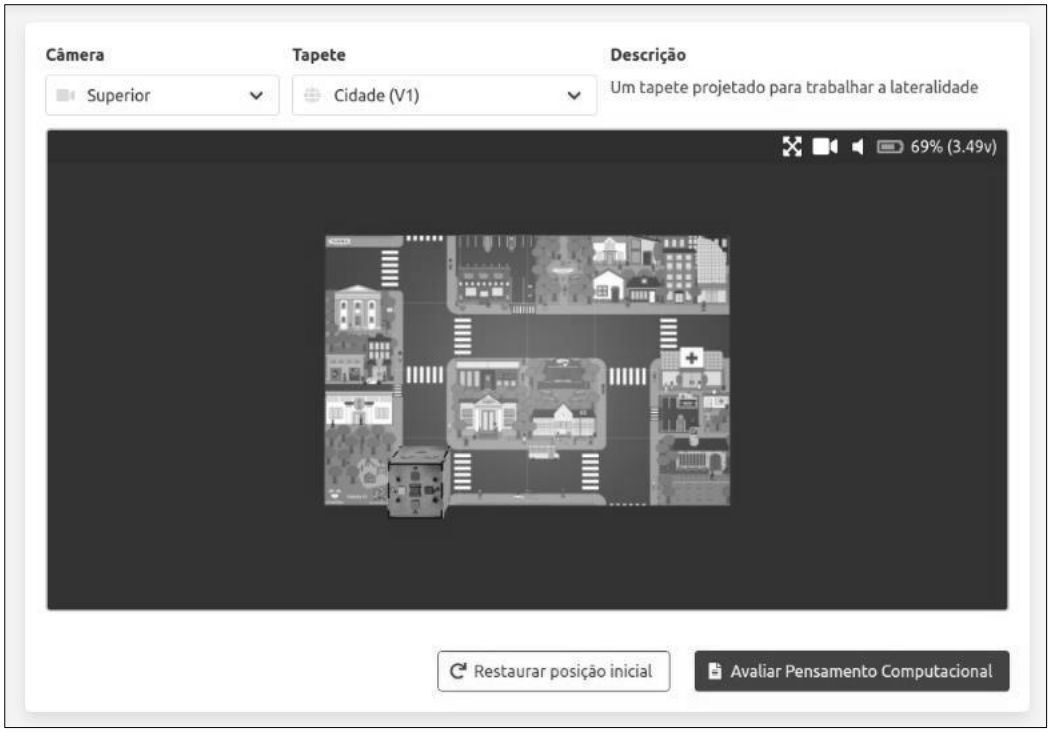


Figura 25. Visualização do RoPE em Visão Superior na aplicação SmartRoPE

Figura 26. Visualização combinada do RoPE na aplicação SmartRoPE

Além destes 3 modos de câmera principais, é possível também combinar os modos 1ª pessoa e 3ª pessoa com o modo de visão superior, conforme ilustrado na Figura 26. Nesse modo, a tela é dividida e a visão em 1ª pessoa/3ª pessoa fica no lado esquerda, enquanto a visão superior fica do lado direito. Por fim, a aplicação oferece um modo de visualização em tela cheia, na qual a renderização é sobreposta às demais telas da aplicação, ocupando toda a tela e oferecendo uma vista ampliada.

No topo do painel de renderização foi implementada uma barra de status que permite aferir algumas informações referentes ao estado do brinquedo como, por exemplo o nível e tensão da bateria e o estado do som: habilitado, desabilitado ou reproduzindo. Também foram colocados dois botões, um que permite alternar entre os modos de câmera citados anteriormente e outro que permite ativar/desativar o modo de tela cheia.

Como já foi mencionado no capítulo **2.2 O BRINQUEDO ROPE**, as atividades com o RoPE envolvem o uso de tapetes pedagógicos a fim de enriquecer a experiência do usuário. Levando isso em conta, foi implementada uma funcionalidade que permite alterar o tapete que é exibido na visualização 3D. É possível escolher qualquer um dos 7 tapetes disponíveis atualmente para o

brinquedo (Figura 27 até a Figura 33), além de um oitavo tapete genérico que não existe fisicamente (Figura 34), mas serve apenas visualizar a orientação espacial do RoPE na renderização 3D.

Figura 27. Tapete pedagógico “Animais V1”

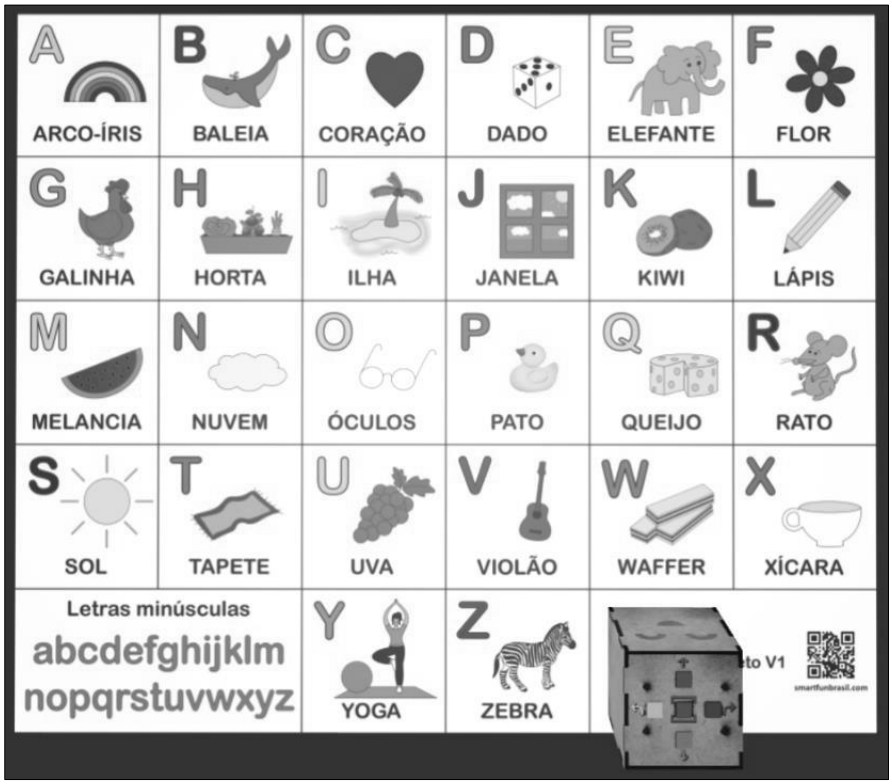


Figura 28. Tapete pedagógico “Alfabeto V1”

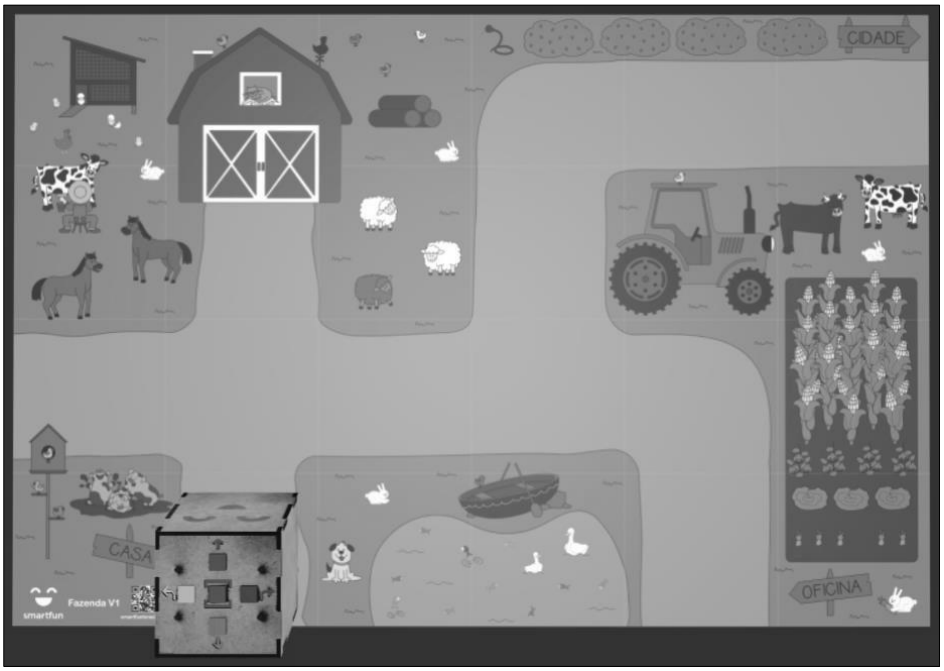


Figura 29. Tapete pedagógico “Fazenda V1”

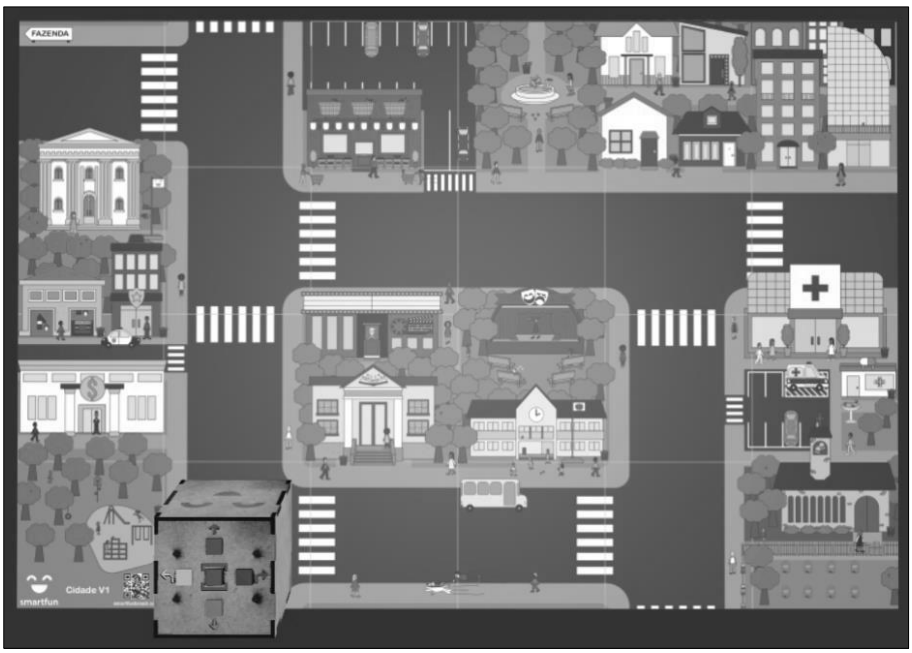


Figura 30. Tapete pedagógico “Cidade V1”

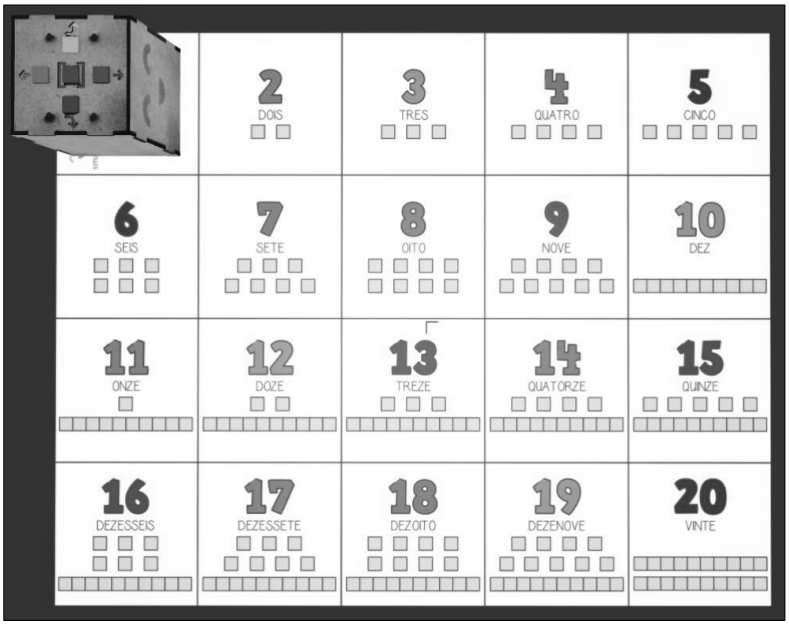


Figura 31. Tapete pedagógico “Números V1”

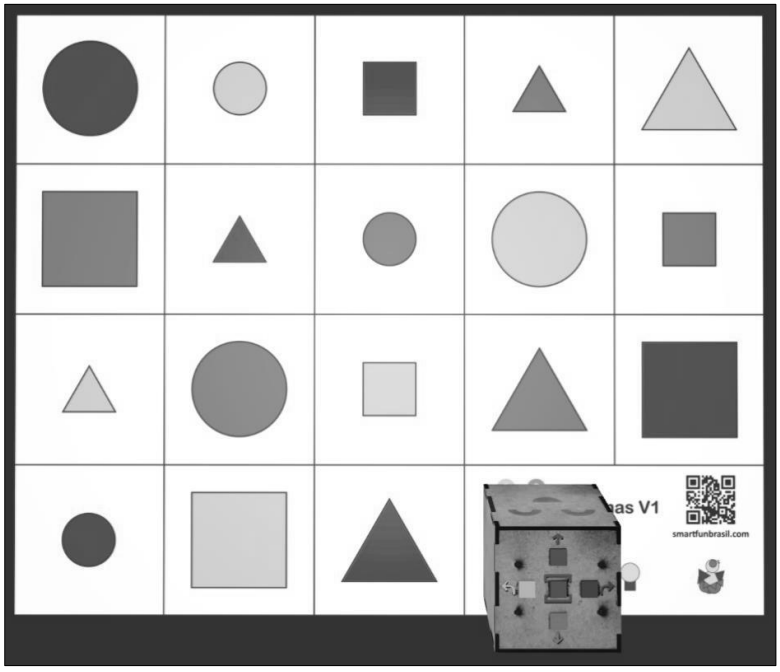


Figura 32. Tapete pedagógico “Formas V1”

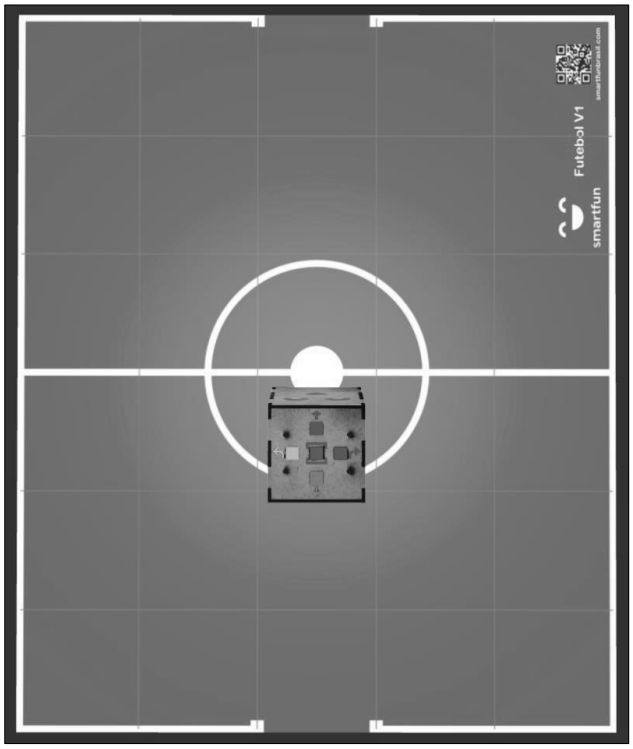


Figura 33. Tapete pedagógico “Futebol V1”

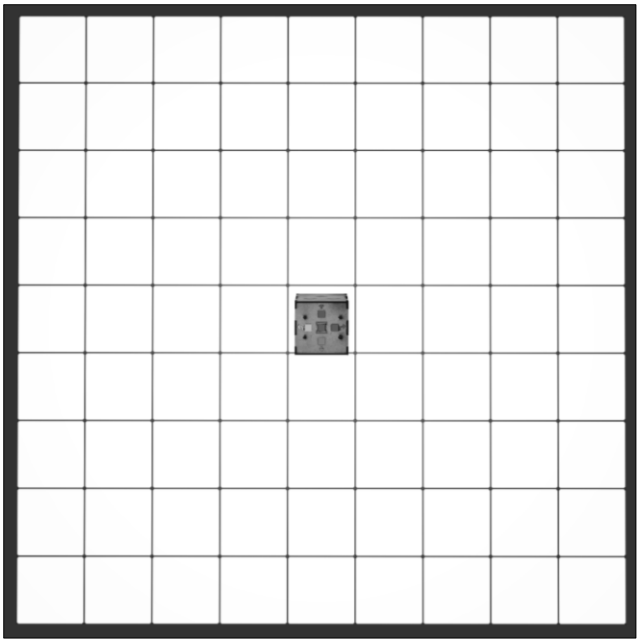


Figura 34. Tapete genérico do RoPE para orientação espacial

Na Figura 35 é possível observar a tela de testes da aplicação SmartRoPE em sua íntegra.

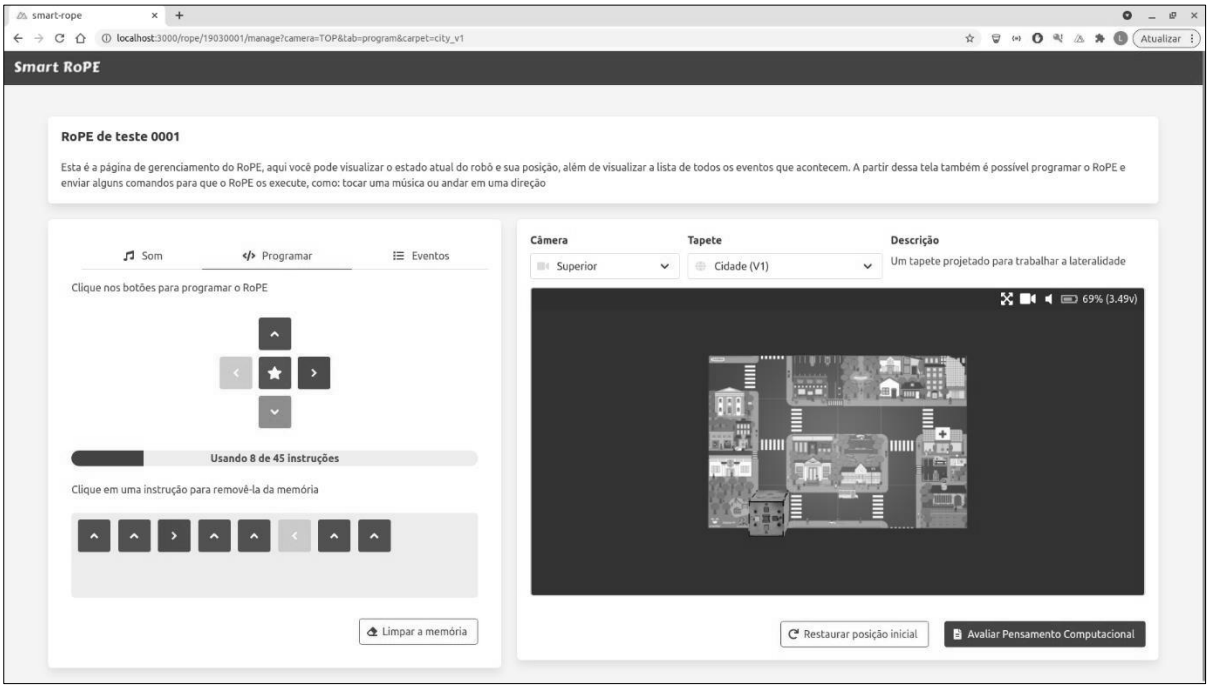


Figura 35. Tela de testes da aplicação SmartRoPE

Fonte: O autor

4.2 AVALIAÇÃO DO PENSAMENTO COMPUTACIONAL

Recentemente o grupo de pesquisa em informática na educação da UNIVALI desenvolveu um teste chamado RoPE Test, que consiste em um conjunto de desafios de programação elaborados com o intuito de avaliar o grau de desenvolvimento do pensamento computacional em crianças.

Figura 36. Tapete para avaliação do pensamento computacional

O teste utiliza como ferramenta o RoPE e seu tapete pedagógico “Animais V1” (Figura 36). 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.

O teste é dividido em várias etapas que deverão ser cumpridas em sequência. Cada etapa é composta por um ou mais desafios no qual é requisitado que a criança execute um algoritmo para cumprir um objetivo específico. O primeiro desafio do teste, por exemplo, consiste em solicitar à criança que programe o RoPE para sair da posição inicial e ir até o coelho.

Conforme a criança avança nos testes o nível de dificuldade e complexidade dos algoritmos é aumentada através da introdução de condições e restrições. Um dos desafios mais avançados, por exemplo, consiste em solicitar à criança que programe o RoPE da posição inicial até o cachorro passando pela galinha.

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 desafios 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 como, por exemplo, a quantidade de instruções que a criança usa para resolver o problema.

Dentro desse contexto foi desenvolvida uma ferramenta integrada à aplicação SmartRoPE para auxiliar na realização do Rope Test. Com base no documento descritivo do teste, disponível no **ANEXO A – ROPE TEST**, foi feito um levantamento das variáveis que poderiam ser coletadas de forma automática usando o sistema criado durante a etapa de desenvolvimento do trabalho. As variáveis levantadas estão relacionadas no Quadro 45.

Na primeira tela da ferramenta, ilustrada na Figura 37. Seleção do indivíduo para aplicação do teste do Pensamento Computacional, é exibida uma lista para seleção do indivíduo no qual o teste será aplicado. É possível adicionar rapidamente o indivíduo caso ele ainda não esteja cadastrado. Conforme o documento de referência, no cadastro devem ser preenchidos, o nome completo, a data de nascimento, o ano escolar e o gênero. Também é necessário informar se o indivíduo já teve algum contato com o RoPE anteriormente. A Figura 38 ilustra a tela de cadastro.

Quadro 45. Variáveis de avaliação do pensamento computacional coletadas pelo RoPE

Variável	**Variável**
Tempo total do desafio	Conseguiu concluir o desafio?
Tempo total programando	Encontrou uma solução ótima?
Tempo até a primeira instrução	Começou a programar imediatamente?
Tempo médio entre as instruções	Quantidade de pedidos de ajuda
Quantidade de instruções usadas	Quantidade de verbalizações realizadas
Instruções mais usadas	Quantidade de verbalizações positivas
Instruções menos usadas	Quantidade de verbalizações negativas
Quantidade de movimentos para a frente	Quantidade de giros para a esquerda
Quantidade de movimentos para trás	Quantidade de giros para a direita

Fonte: O autor

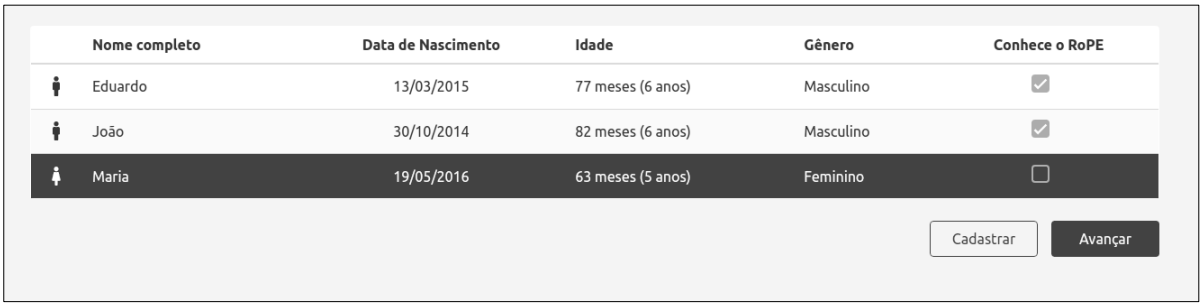


Figura 37. Seleção do indivíduo para aplicação do teste do Pensamento Computacional

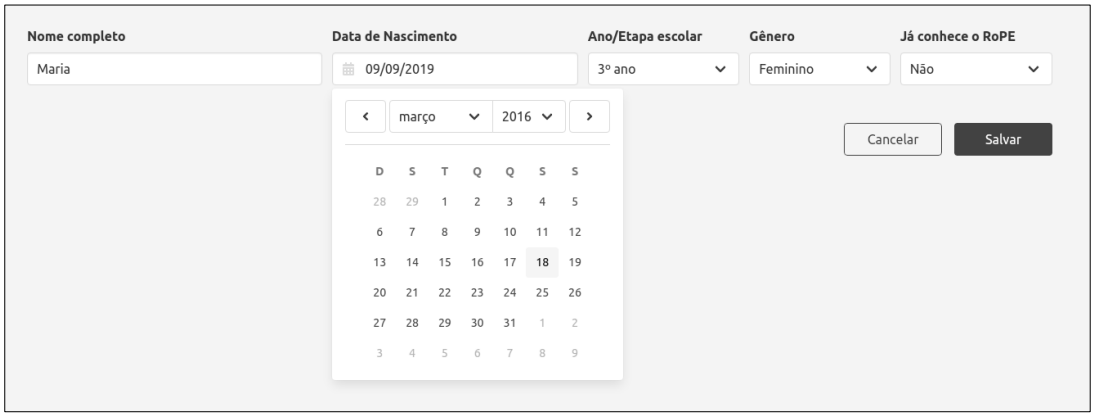


Figura 38. Cadastro de um novo indivíduo no teste do Pensamento Computacional

Após a seleção do indivíduo, é apresentada a tela para a seleção do teste a ser aplicado, conforme ilustra a Figura 39. Nesta tela é apresentado o título do teste e a sua descrição. Na data em que este trabalho está sendo escrito, somente o RoPE Test está disponível na ferramenta, no entanto, o sistema já foi projetado de forma a permitir a inclusão de novos conjuntos de testes no futuro.

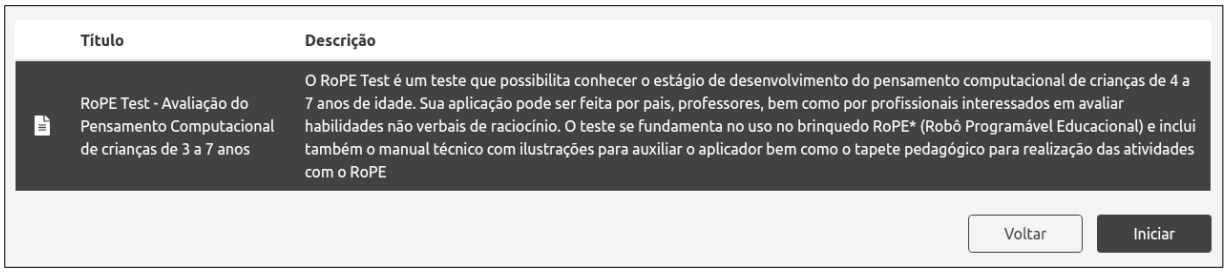


Figura 39. Seleção do teste do Pensamento Computacional a ser aplicado

Na sequência é exibida a tela de execução do teste, a qual está dividida em 3 seções. Na primeira seção, localizada ao lado esquerdo da tela, está o painel de navegação do teste, ilustrado na Figura 40. Este painel exibe, em forma de árvore, toda as etapas do teste, bem como os desafios existentes dentro de cada etapa.

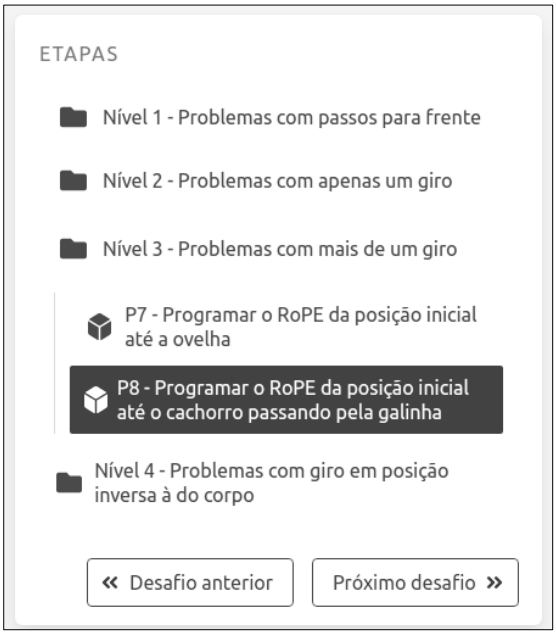


Figura 40. Painel de navegação do teste do Pensamento Computacional

Um desafio pode ser selecionado clicando diretamente no item correspondente ou usando os botões de navegação disponíveis na parte inferior. Quando o teste está sendo aplicado pela primeira vez com um indivíduo, somente o primeiro desafio do primeiro nível está disponível e os desafios

vão sendo liberados conforme são concluídos. Por outro lado, os desafios que já tiverem sido concluídos pelo indivíduo em uma sessão de testes anterior estarão automaticamente desbloqueados quando o teste for iniciado.

Na segunda seção, localizada ao lado direito da tela, está a lista de variáveis do desafio, conforme ilustrado na Figura 41. Inicialmente as variáveis são exibidas com um valor indefinido e, posteriormente os valores vão sendo computados e atualizados durante o andamento do teste.

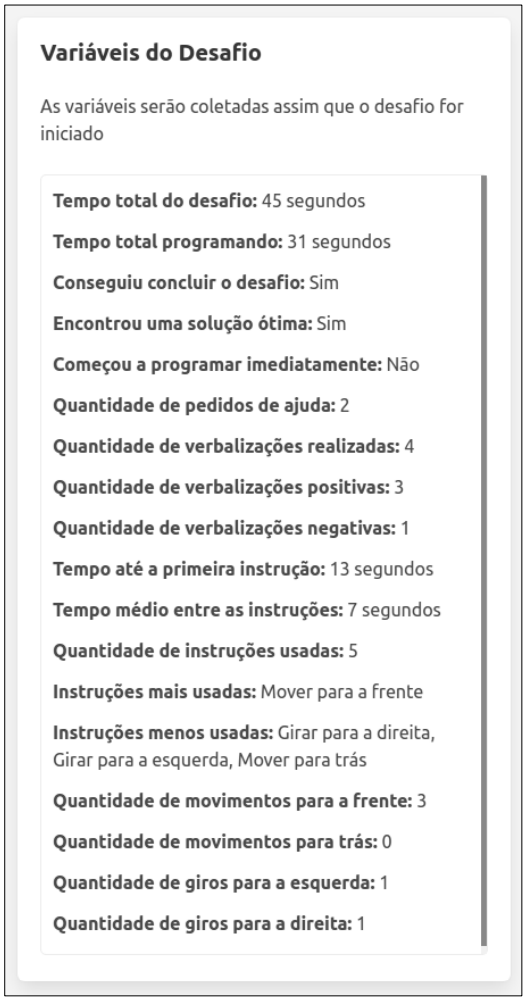


Figura 41. Variáveis do desafio do teste do Pensamento Computacional

A última seção, localizada no centro da tela, contém um painel com botões que permitem, iniciar, finalizar e abortar um desafio, conforme ilustrado na Figura 42. Neste painel, também é possível visualizar o título e a descrição do desafio selecionado no momento, bem como a quantidade de vezes em que ele já foi realizado. Na parte superior, estão o painel de renderização 3D e a lista com o conteúdo da memória, que já foram apresentados anteriormente no trabalho.

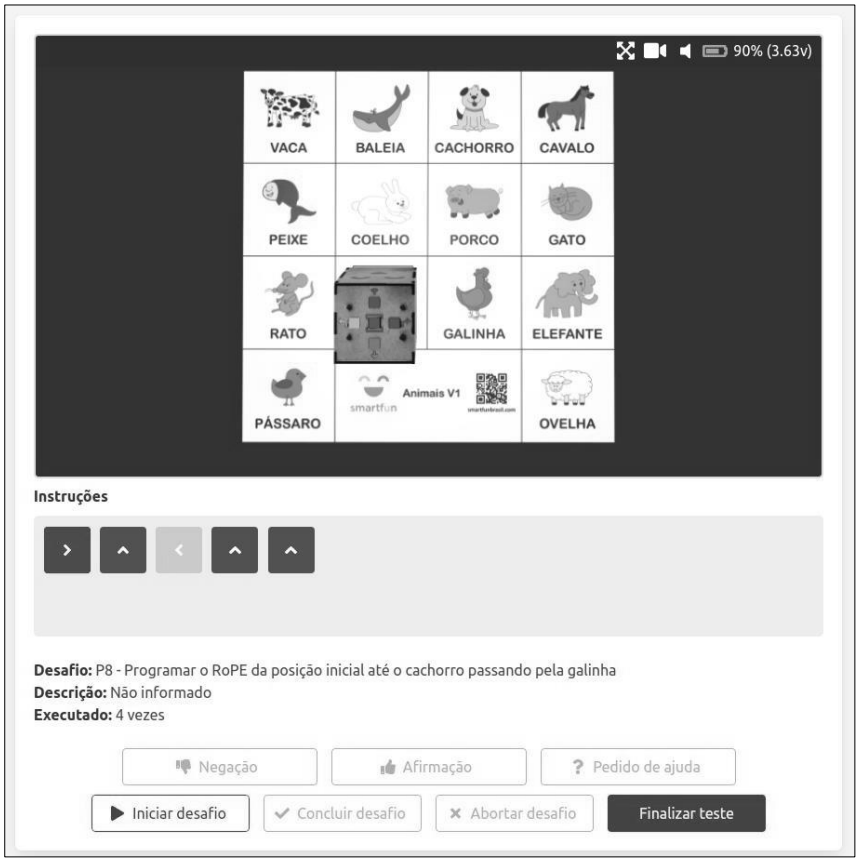


Figura 42. Controle de execução do teste do Pensamento Computacional

A Figura 43, mostra a tela de execução dos testes na sua íntegra.

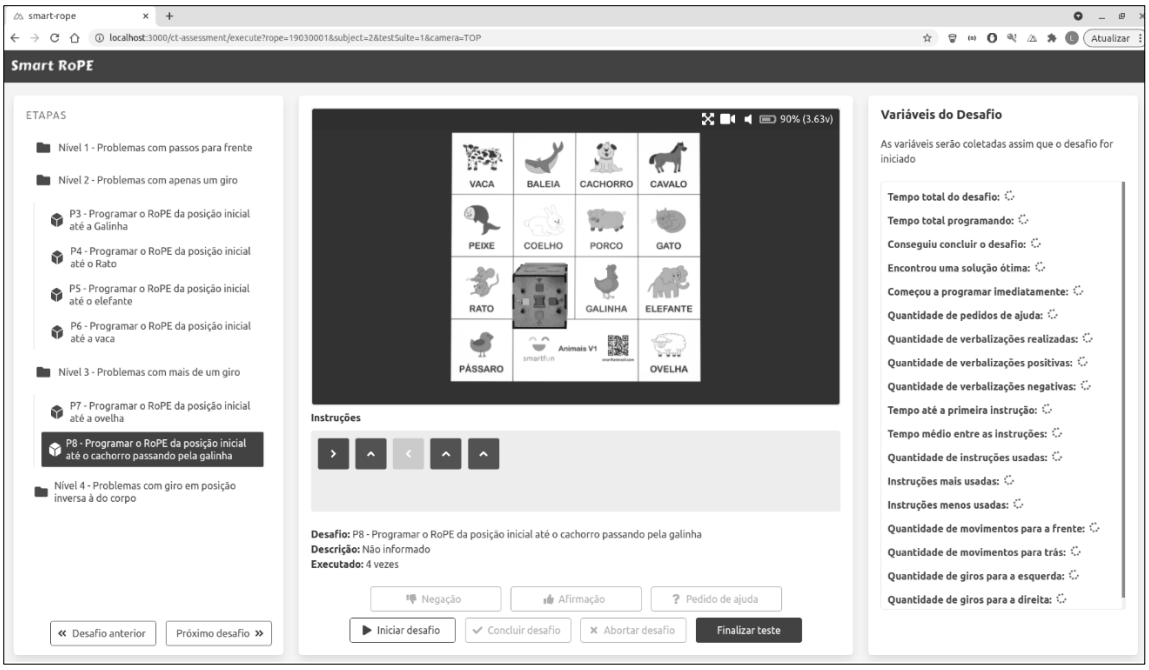
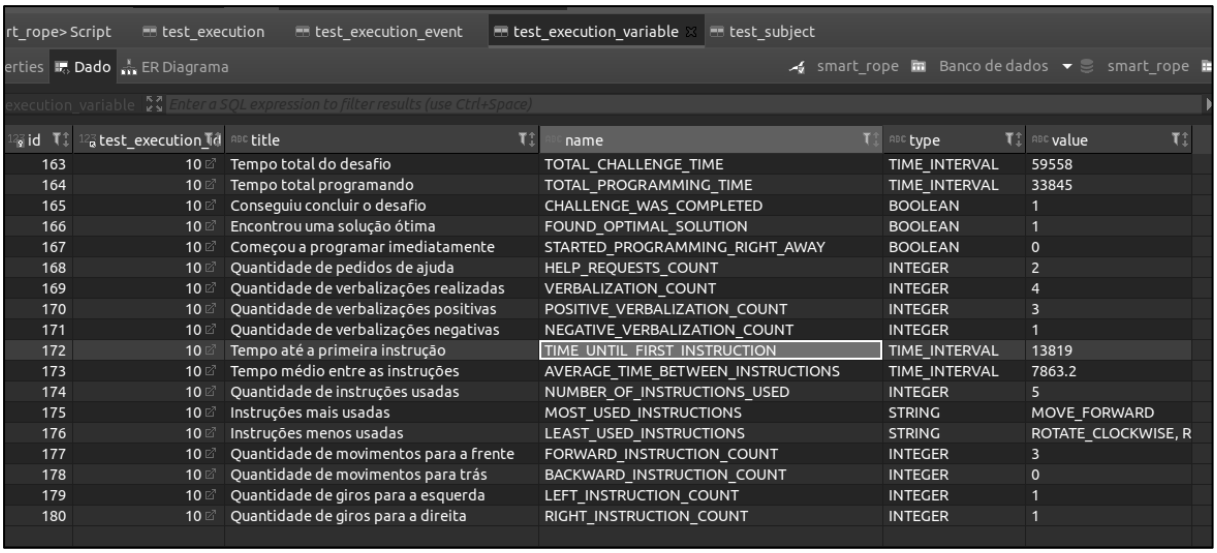


Figura 43. Tela de execução do teste do Pensamento Computacional

Devido às restrições impostas pela pandemia do COVID19, não foi possível realizar um teste real da ferramenta com as crianças. No entanto, foram executadas sessões de teste locais para validar se os dados estavam sendo coletados corretamente. Conforme pode ser visto na Figura 44 e na figura x, a aplicação conseguiu coletar as variáveis e os eventos e registrá-los em um banco de dados, tendo cumprido a sua proposta.



rt_rope> Script | test_execution | test_execution_event | test_execution_variable | test_subject

erties | Dado | ER Diagrama | smart_rope | Banco de dados | smart_rope

execution_variable | Enter a SQL expression to filter results (use Ctrl+Space)

id	test_execution_id	title	name	type	value
163	10	Tempo total do desafio	TOTAL_CHALLENGE_TIME	TIME_INTERVAL	59558
164	10	Tempo total programando	TOTAL_PROGRAMMING_TIME	TIME_INTERVAL	33845
165	10	Conseguiu concluir o desafio	CHALLENGE_WAS_COMPLETED	BOOLEAN	1
166	10	Encontrou uma solução ótima	FOUND_OPTIMAL_SOLUTION	BOOLEAN	1
167	10	Começou a programar imediatamente	STARTED_PROGRAMMING_RIGHT_AWAY	BOOLEAN	0
168	10	Quantidade de pedidos de ajuda	HELP_REQUESTS_COUNT	INTEGER	2
169	10	Quantidade de verbalizações realizadas	VERBALIZATION_COUNT	INTEGER	4
170	10	Quantidade de verbalizações positivas	POSITIVE_VERBALIZATION_COUNT	INTEGER	3
171	10	Quantidade de verbalizações negativas	NEGATIVE_VERBALIZATION_COUNT	INTEGER	1
172	10	Tempo até a primeira instrução	TIME_UNTIL_FIRST_INSTRUCTION	TIME_INTERVAL	13819
173	10	Tempo médio entre as instruções	AVERAGE_TIME_BETWEEN_INSTRUCTIONS	TIME_INTERVAL	7863.2
174	10	Quantidade de instruções usadas	NUMBER_OF_INSTRUCTIONS_USED	INTEGER	5
175	10	Instruções mais usadas	MOST_USED_INSTRUCTIONS	STRING	MOVE_FORWARD
176	10	Instruções menos usadas	LEAST_USED_INSTRUCTIONS	STRING	ROTATE_CLOCKWISE, R
177	10	Quantidade de movimentos para a frente	FORWARD_INSTRUCTION_COUNT	INTEGER	3
178	10	Quantidade de movimentos para trás	BACKWARD_INSTRUCTION_COUNT	INTEGER	0
179	10	Quantidade de giros para a esquerda	LEFT_INSTRUCTION_COUNT	INTEGER	1
180	10	Quantidade de giros para a direita	RIGHT_INSTRUCTION_COUNT	INTEGER	1

Figura 44. Variáveis do teste armazenadas em uma tabela do banco de dados

Vale ressaltar que, não está no escopo deste trabalho processar os dados a fim de avaliar o grau de desenvolvimento do pensamento computacional, mas sim, apenas validar a viabilidade de usar o RoPE para sua coleta. No entanto, a ferramenta ficará à disposição do grupo de pesquisa da UNIVALI para que sejam conduzidas novas coletas para que os dados sejam processados da forma adequada.

Para melhor compreensão de como os dados são armazenados pela ferramenta, foi modelado um diagrama de banco de dados (Figura 46). 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_execution”. Por fim, os eventos recebidos do RoPE bem como as variáveis computadas a partir desses eventos, são armazenados nas tabelas “test_execution_event” e “test_execuution_variable”, respectivamente.

rties | Dado | ER Diagrama — smart_rope | Banco de dados

xecution_event | Enter a SQL expression to filter results (use Ctrl+Space)

id	timestamp	rope_serial_nu	type	payload	test_execution_id
97	2021-09-09 15:10:56	19030001	MEMORY_CLEARED	{"id":1868601735,"timestamp":1868	10
98	2021-09-09 15:11:10	19030001	BUTTON_PRESSED	{"id":1882128310,"timestamp":1882	10
99	2021-09-09 15:11:10	19030001	INSTRUCTION_INSERTED	{"id":1882141684,"timestamp":1882	10
100	2021-09-09 15:11:10	19030001	LED_TOGGLED	{"id":1882145656,"timestamp":1882	10
101	2021-09-09 15:11:10	19030001	SOUND_PLAYBACK_STARTED	{"id":1882289260,"timestamp":1882	10
102	2021-09-09 15:11:10	19030001	LED_TOGGLED	{"id":1882301397,"timestamp":1882	10
103	2021-09-09 15:11:10	19030001	SOUND_PLAYBACK_FINISHED	{"id":1882525771,"timestamp":1882	10
104	2021-09-09 15:11:18	19030001	BUTTON_PRESSED	{"id":1890428303,"timestamp":1890	10
105	2021-09-09 15:11:18	19030001	INSTRUCTION_INSERTED	{"id":1890441827,"timestamp":1890	10
106	2021-09-09 15:11:18	19030001	LED_TOGGLED	{"id":1890445658,"timestamp":1890	10
107	2021-09-09 15:11:18	19030001	SOUND_PLAYBACK_STARTED	{"id":1890589264,"timestamp":1890	10
108	2021-09-09 15:11:18	19030001	LED_TOGGLED	{"id":1890600963,"timestamp":1890	10
109	2021-09-09 15:11:19	19030001	SOUND_PLAYBACK_FINISHED	{"id":1890825547,"timestamp":1890	10
110	2021-09-09 15:11:25	19030001	BUTTON_PRESSED	{"id":1897050305,"timestamp":1897	10
111	2021-09-09 15:11:25	19030001	INSTRUCTION_INSERTED	{"id":1897063518,"timestamp":1897	10
112	2021-09-09 15:11:25	19030001	LED_TOGGLED	{"id":1897067459,"timestamp":1897	10
113	2021-09-09 15:11:25	19030001	SOUND_PLAYBACK_STARTED	{"id":1897210923,"timestamp":1897	10
114	2021-09-09 15:11:25	19030001	LED_TOGGLED	{"id":1897223020,"timestamp":1897	10
115	2021-09-09 15:11:25	19030001	SOUND_PLAYBACK_FINISHED	{"id":1897447710,"timestamp":1897	10
116	2021-09-09 15:11:31	19030001	BUTTON_PRESSED	{"id":1903256328,"timestamp":1903	10
117	2021-09-09 15:11:31	19030001	INSTRUCTION_INSERTED	{"id":1903269831,"timestamp":1903	10
118	2021-09-09 15:11:31	19030001	LED_TOGGLED	{"id":1903273726,"timestamp":1903	10
119	2021-09-09 15:11:31	19030001	SOUND_PLAYBACK_STARTED	{"id":1903417289,"timestamp":1903	10
120	2021-09-09 15:11:31	19030001	LED_TOGGLED	{"id":1903429029,"timestamp":1903	10
121	2021-09-09 15:11:31	19030001	SOUND_PLAYBACK_FINISHED	{"id":1903653516,"timestamp":1903	10
122	2021-09-09 15:11:35	19030001	BUTTON_PRESSED	{"id":1907625304,"timestamp":1907	10
123	2021-09-09 15:11:35	19030001	INSTRUCTION_INSERTED	{"id":1907638449,"timestamp":1907	10
124	2021-09-09 15:11:35	19030001	LED_TOGGLED	{"id":1907642783,"timestamp":1907	10
125	2021-09-09 15:11:36	19030001	SOUND_PLAYBACK_STARTED	{"id":1907786254,"timestamp":1907	10
126	2021-09-09 15:11:36	19030001	LED_TOGGLED	{"id":1907797954,"timestamp":1907	10
127	2021-09-09 15:11:36	19030001	SOUND_PLAYBACK_FINISHED	{"id":1908022532,"timestamp":1908	10
128	2021-09-09 15:11:43	19030001	BUTTON_PRESSED	{"id":1915620314,"timestamp":1915	10
129	2021-09-09 15:11:43	19030001	LED_TOGGLED	{"id":1915633760,"timestamp":1915	10
130	2021-09-09 15:11:43	19030001	SOUND_PLAYBACK_STARTED	{"id":1915774857,"timestamp":1915	10
131	2021-09-09 15:11:44	19030001	LED_TOGGLED	{"id":1915787021,"timestamp":1915	10

Figura 45. Eventos do teste armazenadas em uma tabela do banco de dados

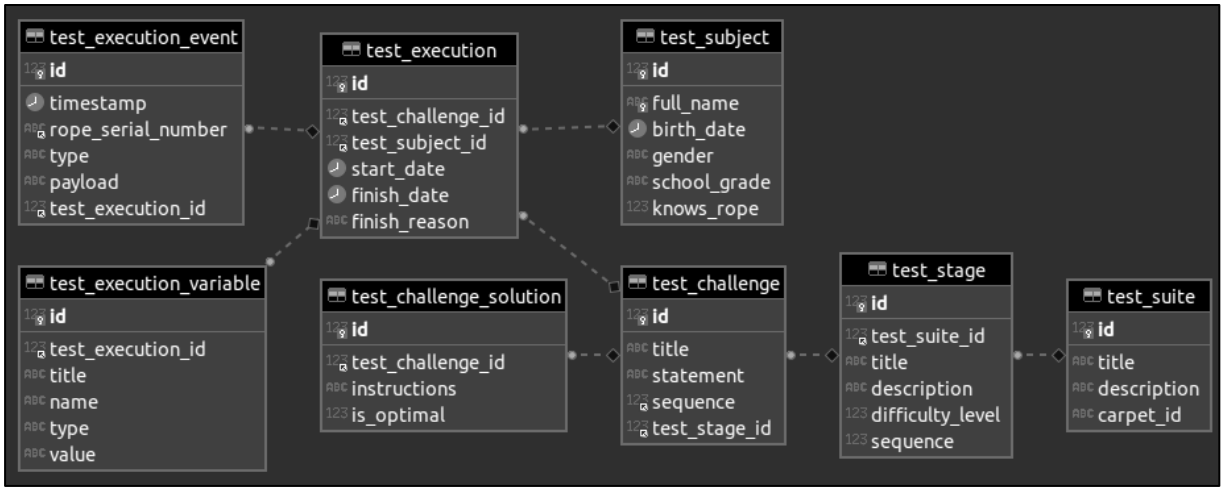


Figura 46. Diagrama ER do banco de dados da ferramenta de avaliação do Pensamento Computacional

Quanto às tecnologias empregadas, a aplicação SmartRoPE foi desenvolvida na linguagem Javascript usando os *frameworks* Vite[21], NuxtJS[22] e VueJS[23]. Para a composição das telas foi usado o framework Buefy[24], que possui um vasto conjunto de componentes de interface prontos para serem usados. Na implementação do painel de visualização 3D, foi usado o BabylonJS[25], um motor de renderização 3D escrito inteiramente na linguagem Javascript e especialmente otimizado para o carregamento dos modelos 3D via conexão HTTP. Tanto o modelo do RoPE quando os modelos dos tapetes foram criados do zero usando o Blender[26], um software de modelagem 3D Open Source. Por Fim, para a comunicação com o *broker* foi usado o Eclipse Paho[27], um cliente MQTT mantido pela *Eclipse Foundation* que possui uma implementação para a linguagem Javascript.

[21] https://vitejs.dev/
[22] https://nuxtjs.org/
[23] https://vuejs.org/
[24] https://buefy.org/
[25] https://www.babylonjs.com/
[26] https://www.blender.org/
[27] https://github.com/eclipse/paho.mqtt.javascript

5 CONCLUSÕES

A proposta desse trabalho foi incluir o suporte à Wi-Fi no brinquedo RoPE e a implementação de um sistema para monitoramento e controle do robô através da Internet, a fim de permitir sua evolução para um SmartToy no futuro. Para alcançar esses objetivos, tanto o *hardware* quanto o *software* do RoPE passaram por modificações.

Na parte de *hardware* o microcontrolador do RoPE, o ATMega328p, foi substituído pelo ESP32, um chip de baixo custo e com maior capacidade de memória e processamento que já vem com Wi-Fi e Bluetooth integrados. Na parte de *software*, o *firmware* do RoPE foi migrado para o novo microcontrolador com a ajuda da biblioteca H4Plugins. O sistema de monitoramento e controle foi implementado usando o protocolo MQTT.

Por fim, foram desenvolvidos dois produtos que serviram para validar o funcionamento do projeto. O primeiro produto, o SmartRoPE, consiste em uma aplicação Web que permite reproduzir sons, programar o RoPE e visualizar sua movimentação em "tempo real" através de uma animação 3D. O segundo produto, consiste em uma ferramenta de apoio à pesquisa que permite usar o RoPE para a coleta de dados usados na avaliação do pensamento computacional.

Os testes realizados com estes dois aplicativos demonstraram que o sistema de monitoramento e controle desenvolvido durante o trabalho está funcionando corretamente.

5.1 TRABALHOS FUTUROS

O mecanismo de controle remoto e monitoramento de eventos implementado no firmware do RoPE durante o desenvolvimento desse trabalho abre as portas para uma série de novas pesquisas e aplicações que podem ser desenvolvidas com o RoPE. Seguem algumas ideias.

- Programar o RoPE com a voz: isso poderia ser feito acoplando um microfone ao RoPE e submetendo a fala do usuário a um serviço de processamento de áudio na nuvem, que o converteria em texto. O texto poderia então ser processado para identificar quais comandos o usuário está dando ao RoPE e gerar as mensagens MQTT correspondentes

- Adaptar a aplicação SmartRoPE e transformá-la em um jogo para SmartPhones. A câmera do SmartPhone pode ser usada para detectar automaticamente qual tapete está sendo usado com base no seu QR-Code. Para tornar a atividade mais interessante, podem ser incluídos na renderização 3D obstáculos virtuais em cima do tapete. O usuário precisa então programar o RoPE no mundo real para desviar dos obstáculos virtuais no jogo.

- Criar um sistema multiagente que sincroniza vários RoPEs para realizar uma atividade em conjunto

- Melhorar a aplicação de avaliação do pensamento computacional para incluir novas variáveis no processamento. Incluir relatórios e gráficos com as estatísticas dos testes

Um dos pontos que ficou em aberto durante a execução deste trabalho e que pode ser melhorado no futuro é a parte de segurança do brinquedo envolvendo o protocolo MQTT. No desenvolvimento desse trabalho não foi implementada nenhuma medida de segurança, porém o protocolo MQTT dá suporte a criptografia TLS/SSL. Esse é um trabalho de extrema importância para viabilizar o SmartRoPE como um produto comercial no futuro.

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

BABIUCH, M.; FOLTYNEK, P.; SMUTNY, P. **Using the ESP32 Microcontroller for Data Processing**. 2019 20th International Carpathian Control Conference (ICCC). **Anais**... In: 2019 20TH INTERNATIONAL CARPATHIAN CONTROL CONFERENCE (ICCC). Krakow-Wieliczka, Poland: IEEE, maio 2019. Disponível em: <https://ieeexplore.ieee.org/document/8765944/>. Acesso em: 12 set. 2021

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

BOWLES, P. **H4 library repository**. . Disponível em: <https://github.com/philbowles/H4>. Acesso em: 12 set. 2021.

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

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. **ESP32 Series Datasheet**. . Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 12 set. 2021.

GATIAL, E.; BALOGH, Z.; HLUCHY, L. **Concept of Energy Efficient ESP32 Chip for Industrial Wireless Sensor Network**. 2020 IEEE 24th International Conference on Intelligent Engineering Systems (INES). **Anais**... In: 2020 IEEE 24TH INTERNATIONAL CONFERENCE ON INTELLIGENT ENGINEERING SYSTEMS (INES). Reykjavík, Iceland: IEEE, jul. 2020. Disponível em: <https://ieeexplore.ieee.org/document/9147189/>. Acesso em: 12 set. 2021

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

HIVEMQ. **Key Features of HiveMQ MQTT Broker**. . Disponível em: <https://www.hivemq.com/hivemq/mqtt-broker/>. Acesso em: 12 set. 2021.

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.

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.

MICROSHIP. **ATmega48A/PA/88A/PA/168A/PA/328/P megaAVR Data Sheet**Microship, , 2020. . 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.

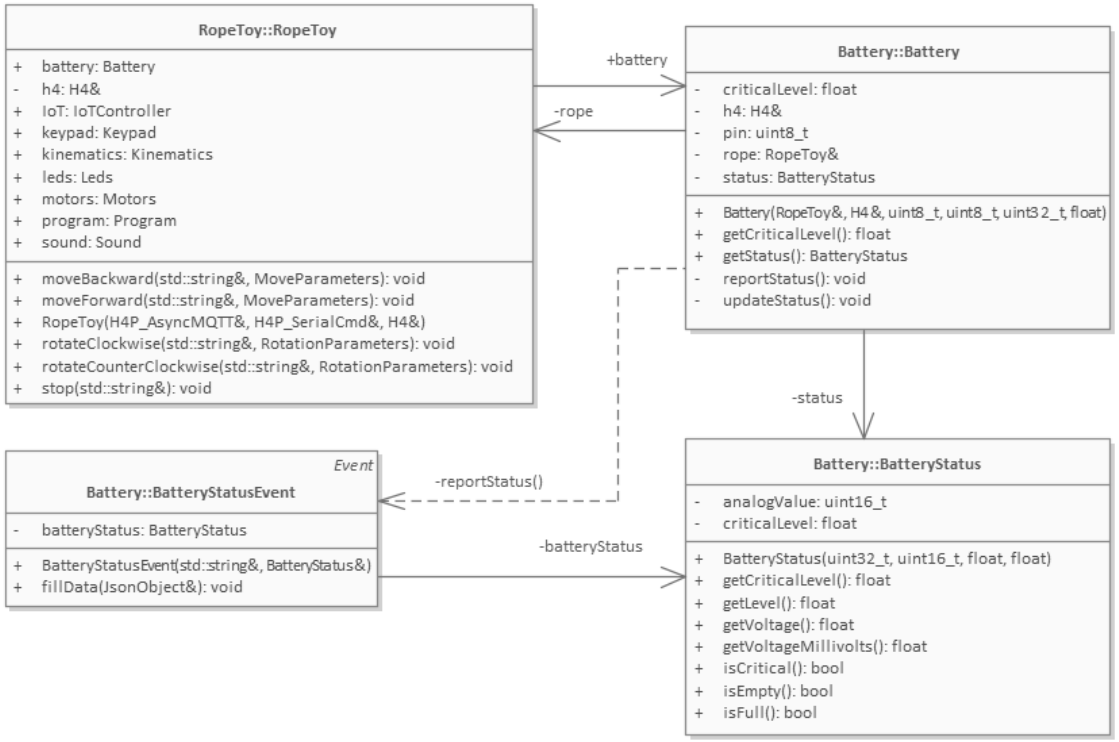
SULLIVAN, A. et al. KIBO Robot Demo: Engaging Young Children in Programming and Engineering. p. 4, 2015.

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

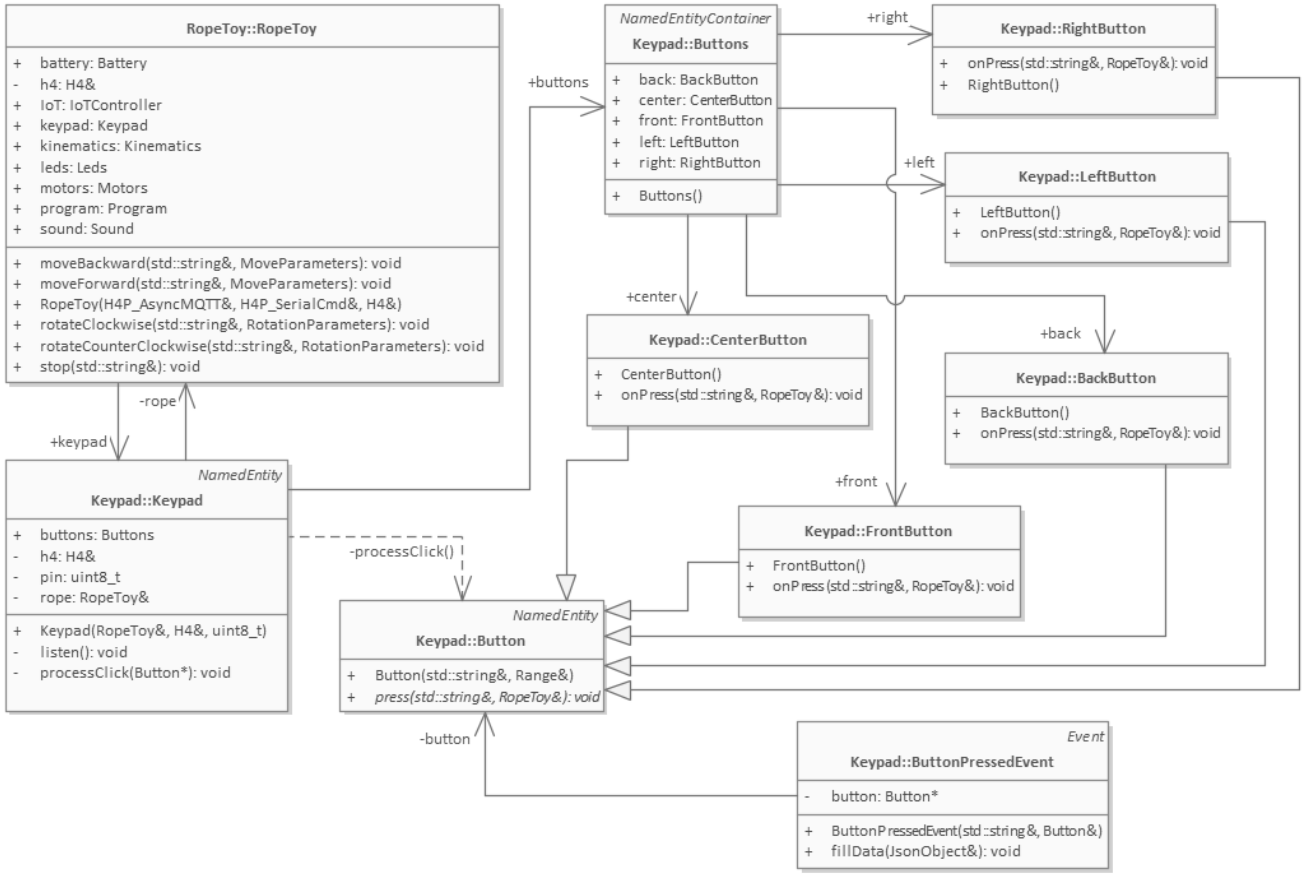
WEHNER, P.; PIBERGER, C.; GOHRINGER, D. **Using JSON to manage communication between services in the Internet of Things**. 2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC). **Anais**... In: 2014 9TH INTERNATIONAL SYMPOSIUM ON RECONFIGURABLE AND COMMUNICATION-CENTRIC 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.

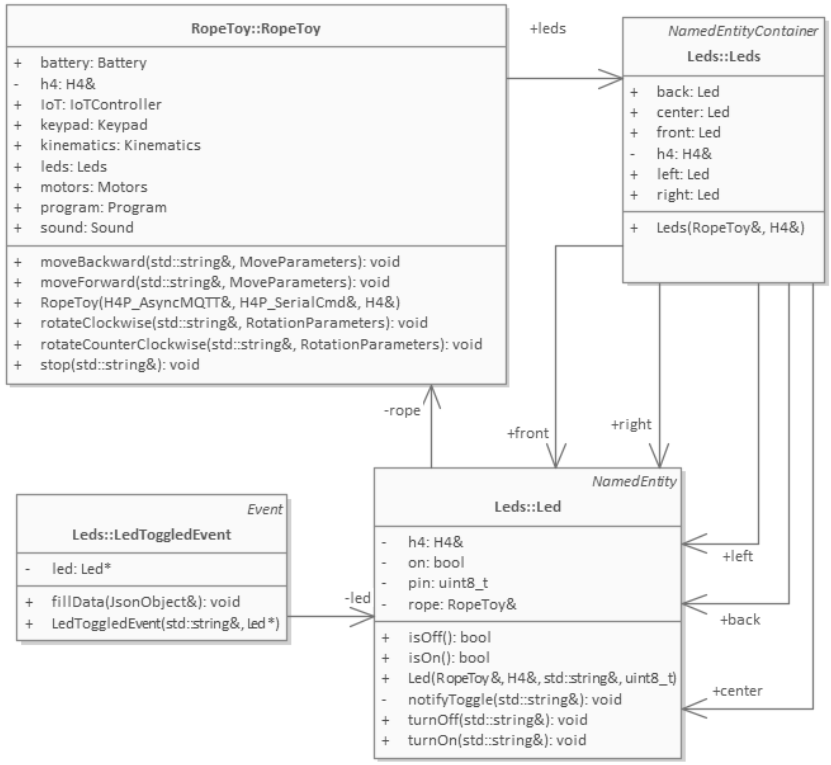
APÊNDICE A – MONITORAMENTO DA BATERIA



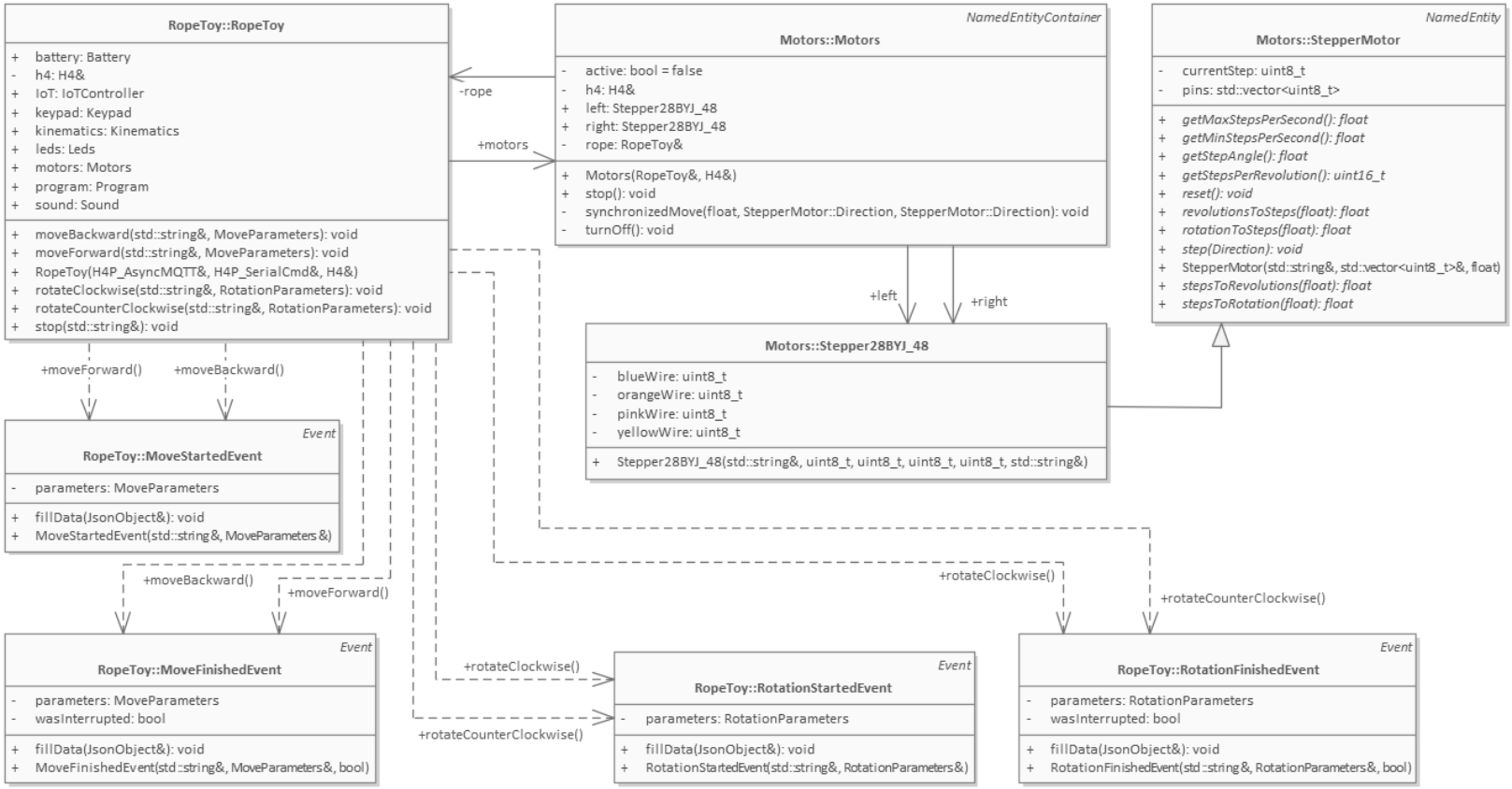
APÊNDICE B – LEITURA DO TECLADO



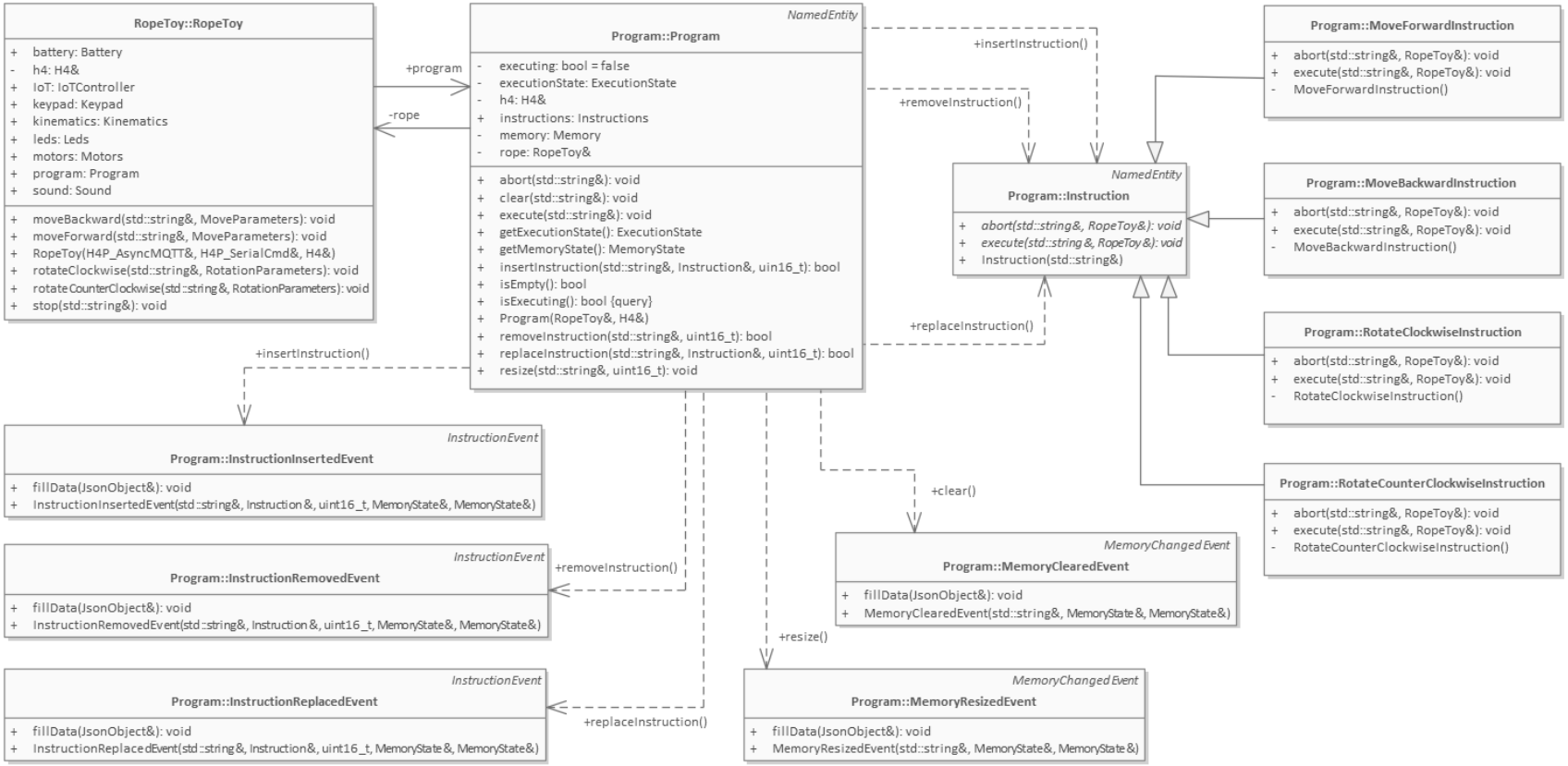
APÊNDICE C – ACIONAMENTO DOS LEDS



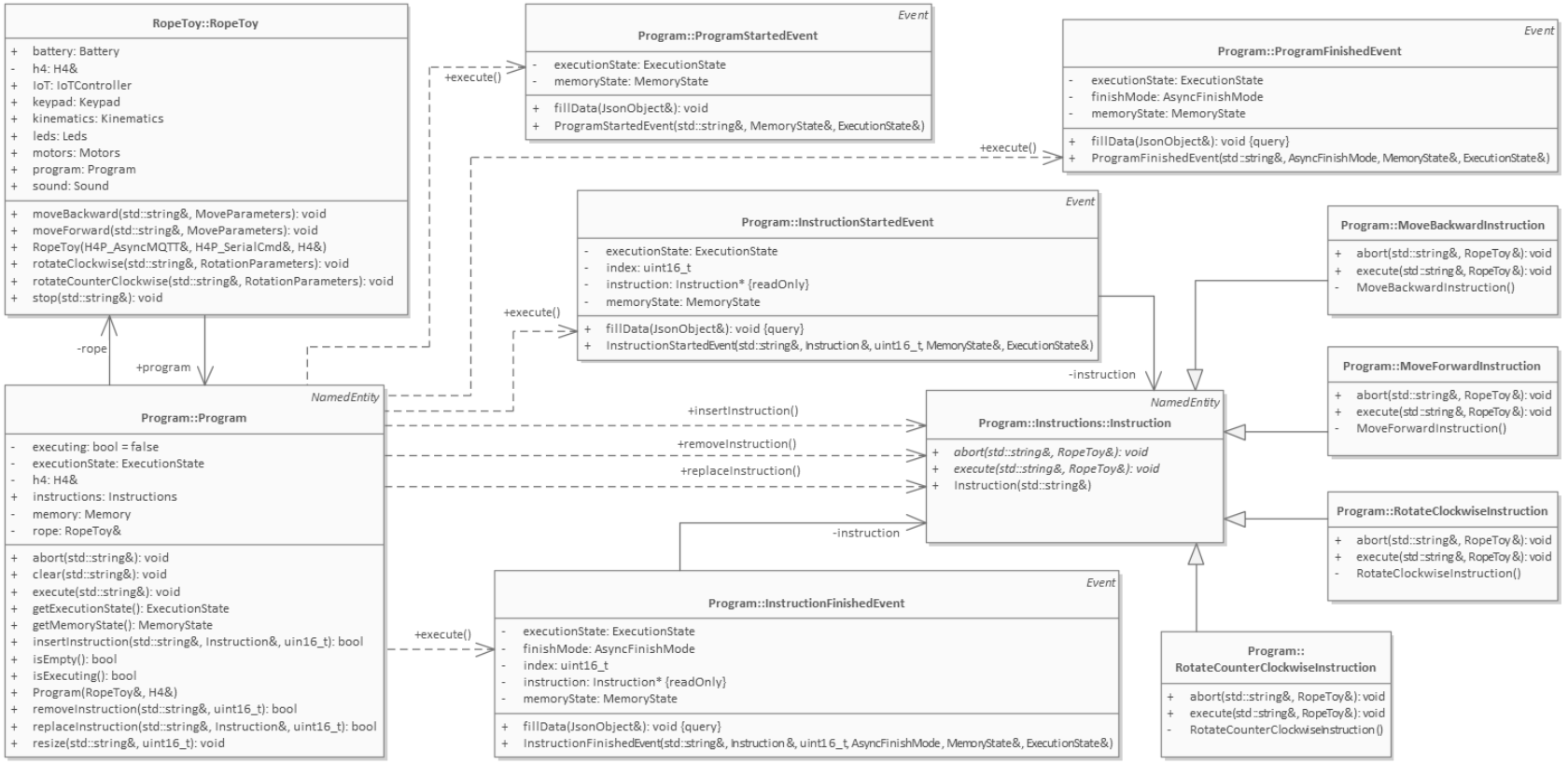
APÊNDICE D – ACIONAMENTO DOS MOTORES



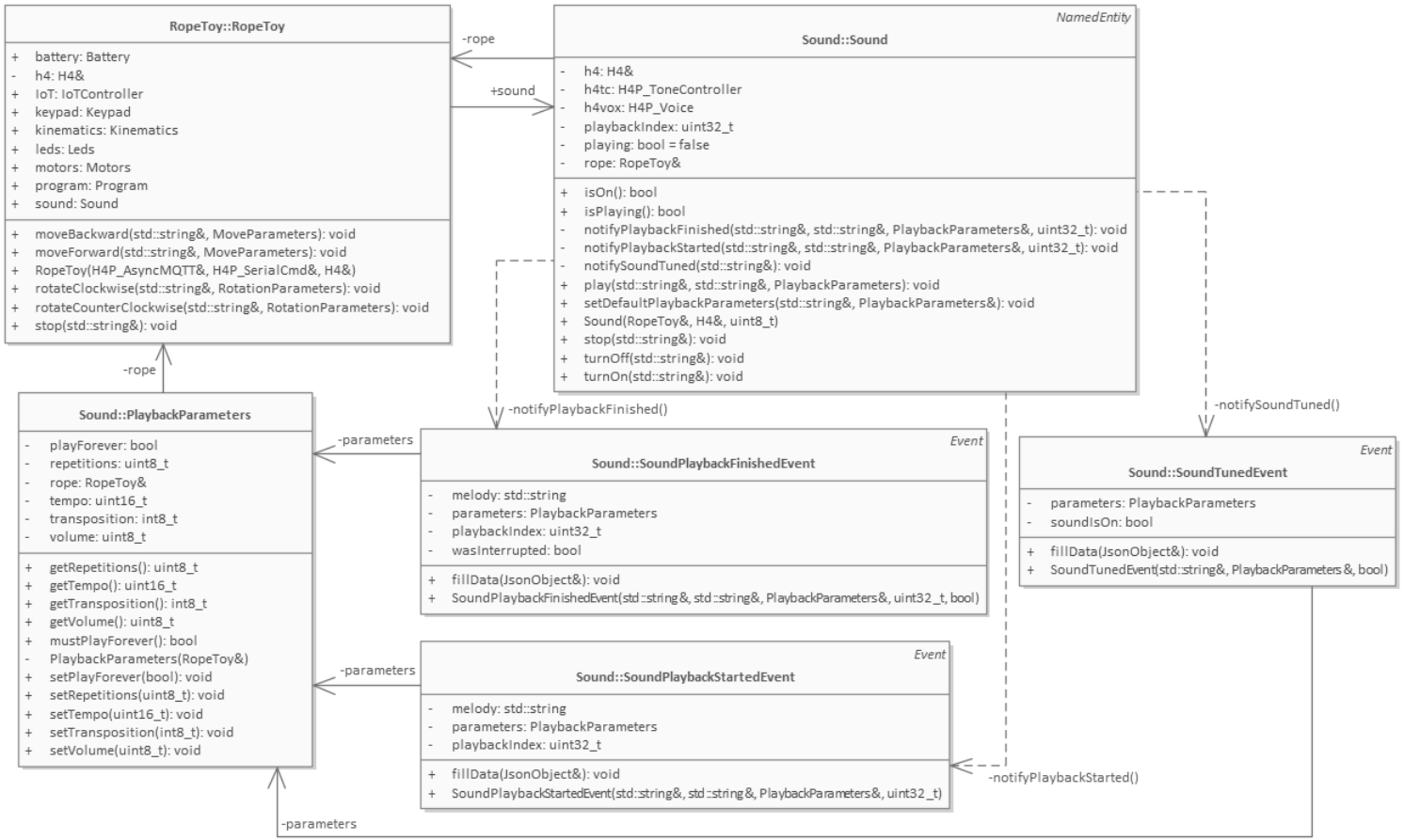
APÊNDICE E – PROGRAMAÇÃO DE INSTRUÇÕES



APÊNDICE F – EXECUÇÃO DE INSTRUÇÕES



APÊNDICE G – REPRODUÇÃO DE SONS



ANEXO A – ROPE TEST

Nome: RoPE Test - Avaliação do Pensamento Computacional de crianças de 3 a 7 anos

Autor: André Raabe
Idade: De 3 a 7 anos de idade
Escolaridade: Não há restrição
Aplicação: Individual

Tempo de Aplicação: Aproximadamente 40 minutos.

Descrição: O RoPE Test é um teste que possibilita conhecer o estágio de desenvolvimento do pensamento computacional de crianças de 4 a 7 anos de idade. Sua aplicação pode ser feita por pais, professores, bem como por profissionais interessados em avaliar habilidades não verbais de raciocínio. O teste se fundamenta no uso no brinquedo RoPE* (Robô Programável Educacional) e inclui também o manual técnico com ilustrações para auxiliar o aplicador bem como o tapete pedagógico para realização das atividades com o RoPE.
* O brinquedo RoPE não está incluso no teste e pode ser adquirido em smartfunbrasil.com

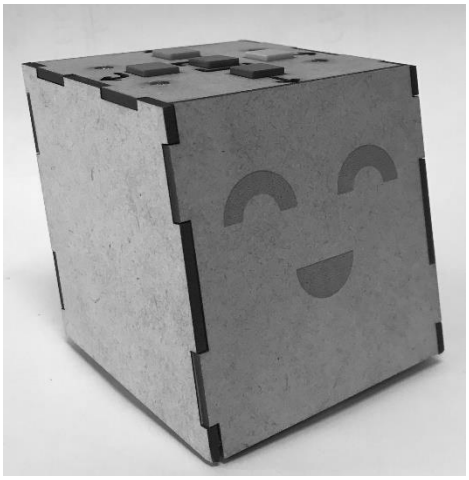
Indicação: O RoPE Test é indicado para conhecer o estágio de desenvolvimento do Pensamento Computacional de crianças de 3 a 7 anos. O Pensamento Computacional mobiliza o pensamento algoritmo, a identificação de padrões, a abstração e a modularização de problemas e soluções. O teste pode ser utilizado também como elemento complementar a avaliação do raciocínio lógico e da inteligência de lógico matemática de crianças com restrição de fala.

Restrito a psicólogos: Não.

Validade do Instrumento: 01/10/2021 a 01/10/2041

Instruções para Aplicação

Materiais Necessário:



Robô RoPE V.2.1

Tapete Animais V.1

Escolha do local
A aplicação do teste deve ser feita idealmente em um ambiente silencioso sem outras crianças ou fontes de distração. Deve haver espaço suficiente para a criança poder andar em volta do tapete. Recomenda-se realizar a aplicação no chão ou sob uma mesa infantil.

Etapas de Aplicação

O aplicador deve ser adulto e deve dominar o uso do brinquedo RoPe conhecendo seu funcionamento e como ele é programado.
Para cada criança que realiza o teste as seguintes etapas devem ser cumpridas

Etapa 1 - Ambientação (Tempo aproximado 5 a 10 minutos)

A etapa de ambientação só é necessária para crianças que nunca brincaram como o RoPe.

1.1 Apresentação: Apresente o brinquedo para a criança. Explique que ele é um robô que gosta de passear seguindo passos que nós ensinamos a ele. Mostre para a criança como fazer o RoPe andar UM PASSO para frente. Mostre a seguir como fazer o RoPE dar TRÊS passos para Frente. Explique que o Rope tem um som que ele toca ao terminar de executar um passeio.
1.2 Exploração livre: Após esta breve demonstração deixe a criança brincar livremente como Rope. Evite interferir e responda apenas o que a criança demandar. Nesta etapa é importante estabelecer que o Robô não deve ser empurrado ou pego com as mão. Ele deve se mover apenas por meio dos passos ensinados a ele com os botões em sua cabeça.
1.3 Descrição da criança: Registre na folha de teste os seguintes dados da criança: Sexo, idade em meses, ano/etapa escolar, se já teve contato prévio com o RoPE.

Etapa 2 - Teste, Observação e Registro

Utilize os problemas definidos na seção 2.1 Problemas na ordem em que eles estão apresentados. Para cada problema da lista a seguir realize os seguintes procedimentos:
1- Explique o problema/desafio para criança.
2- Posicione o RoPE no local e com a mesma orientação indicada pela figura do RoPe no tapete do teste.
3- Solicite à criança para realizar o desafio (lembre a criança de apenas mover o RoPE programando)
4- Registrar na folha do formulário de avaliação:
a. Atitude da criança (saiu programando, pensou antes, apontou, verbalizou, outro)
b. Conseguiu? Em quanto tempo. Quantos passos tem a solução.
c. Número de tentativas
d. Quantas vezes pediu ajuda
5- Sempre que a criança concluir com sucesso apresentar a próxima questão. Caso contrário, pergunte se ela quer parar ou avançar. Após dois problemas em sequência sem que a criança consiga solucionar o teste deve ser interrompido.

2.1 Problemas

Nível 1 - Problemas com passos para frente
P1 - Programar o RoPE da posição inicial até o Coelho Solução: (PF)

P2 - Programar o RoPE da posição inicial até a Baleia
Solução: (PF + PF)

Nível 2 - Problemas com apenas um giro

P3 - Programar o RoPE da posição inicial até a Galinha
Solução: (GD + PF)
P4 - Programar o RoPE da posição inicial até o Rato
Solução: (GE + PF)
P5 - Programar o RoPE da posição inicial até o elefante
Solução: (GD + PF + PF)
P6 - Programar o RoPE da posição inicial até a vaca
Solução: (PF + PF + GE + PF)

Nível 3 - Problemas com mais de um giro

P7 - Programar o RoPE da posição inicial até a ovelha
Solução: (GD + PF + PF + GD + PF)
P8 - Programar o RoPE da posição inicial até o cachorro passando pela galinha
Solução: (GD + PF + GE + PF + PF)

Nível 4 - Problemas com giro em posição inversa à do corpo

P9 - Programar o RoPE para passar sobre estes animais Cachorro, Gato e Galinha
Solução: (PF + PF + GD + PF + PF + GD + PF + PF + GD + PF)
P10 - Programar o RoPE para passar sobre estes animais nesta ordem: Cavalo, Cachorro, Porco e Coelho.
Solução: (GD + PF + PF + GE + PF + PF + GE + PF + GE + PF + GD + PF)

Folha do Teste
(uma por criança)

Nome da criança:
Idade (em meses):
Fase escolar:
Já teve contato anterior com o RoPE: ( ) Sim ( ) Não

Problema	Atitude	Solucionou (S/N)	Quantos programas	Número de passos da solução	Tempo aproximado. na questão	Pedidos de ajuda
P1	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P2	( ) Pressionou os botões impulsivamente					

	( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P3	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P4	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P5	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P6	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P7	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P8	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					
P9	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					

P10	( ) Pressionou os botões impulsivamente ( ) Pensou antes de programar ( ) Apontou com o dedo o caminho ( ) verbalizou o caminho ( ) outro:					