EVOLUINDO O BRINQUEDO ROPE PARA SE TORNAR UM SMART TOY

Luiz Fernando Noschang

Novembro/2020

Orientador: André Luís Alice Raabe, Dr.

Área de Concentração: Computação Aplicada

Linha de Pesquisa: Informática na Educação

Palavras-chave: pensamento computacional, aprendizagem, educação infantil, RoPE, IoT

Número de páginas: 50

**RESUMO**

O com o material didático, o feedback imediato das ações e a adoção de recursos visuais e auditivos. Nesse sentido, brinquedos de programar como a Bee-Bot, Kibo, Pro-Bot e o RoPE têm se mostrado excelentes ferramentas, pois conseguem atender a boa parte desses requisitos. Porém, tão importante quanto desenvolver o pensamento computacional é a capacidade de avaliar o grau desse desenvolvimento, uma vez que isso permite determinar o quão eficazes são as metodologias utilizadas e, assim, aperfeiçoá-las.

Sendo assim, existe a necessidade de se criar uma ferramenta que possa auxiliar no

Levando em conta esses fatores, escolheu-se para o desenvolvimento deste trabalho o

o RoPE, um robô programável educacional de baixo custo produzido no Brasil. O RoPE foi escolhido por ser um brinquedo que atende aos requisitos necessários ao se trabalhar com educação infantil: é um objeto tangível (palpável), é lúdico e promove o trabalho em conjunto

Nesse contexto, o objetivo deste trabalho é modificar o brinquedo RoPE

desenvolver uma métrica que permita fazer essa mensuração do pensamento computacional em crianças de 4 a 7 anos e que possa ser usada como um referencial pelos pesquisadores da área. Por se tratar da educação infantil, optou-se por utilizar como ferramenta para o desenvolvimento do trabalho o RoPE, um robô programável educacional de baixo custo produzido no Brasil. O RoPE foi escolhido por ser um brinquedo que atende aos requisitos necessários ao se trabalhar com educação infantil: é um objeto tangível (palpável), é lúdico e promove o trabalho em conjunto. Para viabilizar o desenvolvimento do trabalho, por consequência da pandemia do COVID-19, o RoPE será adaptado para incluir um módulo WiFI e permitir que os testes sejam executados na residência dos indivíduos objetos do teste, e que os dados coletados sejam enviados através da Internet para posterior análise usando técnicas de estatística aplicada à validação de instrumentos. Ao final do trabalho espera-se ter um conjunto de testes que possa ser reproduzido/aplicado por demais pesquisadores e que possa dizer o nível de desenvolvimento do pensamento computacional dos indivíduos em cada uma das habilidades citadas: decomposição, abstração, reconhecimento de padrões e pensamento algorítmico

# INTRODUÇÃO

O pensamento computacional é um assunto que vem sendo discutido há tempos pela comunidade científica, tendo sido abordado já na década de 80 por Seymour Papert. Em seu trabalho, [(PAPERT, 1980)](https://www.zotero.org/google-docs/?12sTHD), já defendia a importância do pensamento computacional ao perceber seus efeitos positivos no aprendizado. Segundo ele, ao programar um computador o indivíduo está ensinando a máquina a pensar e, ao fazer isso, passa a compreender a sua própria forma de pensar.

Após o trabalho de Papert outros pesquisadores abordaram o pensamento computacional de uma forma mais aprofundada gerando um maior interesse pelo tema. Em seu artigo, [(WING, 2006)](https://www.zotero.org/google-docs/?y3F90B) demonstra como o pensamento computacional pode ser aplicado em diversas áreas além da Ciência da Computação, citando, por exemplo, o uso do aprendizado de máquina na estatística e o uso de abstração na biologia para representar a estrutura de proteínas. Wing resume o pensamento computacional como a habilidade de reformular um problema complexo em um problema mais simples e possível de resolver, usando: redução, incorporação, transformação ou simulação.

Definir o conceito exato de pensamento computacional é uma tarefa desafiadora, pois o assunto é bastante abrangente e cada autor aborda diferentes aspectos do mesmo. No entanto, existe um consenso, e este é o principal ponto que precisa ser compreendido, apesar de o pensamento computacional estar relacionado aos computadores ele não se trata dos computadores em si, mas sim de solucionar problemas. De acordo com (CURZON; BLACK; et al, 2009) citado no trabalho de [(MOHAGHEGH; MCCAULEY, 2016)](https://www.zotero.org/google-docs/?rkiX98), "o pensamento computacional é uma coleção de múltiplas habilidades de resolução de problemas com base nos princípios fundamentais da Ciência da Computação".

Uma vez que o pensamento computacional se trata em resolver problemas

demonstra claros como ele pode ser aplicado em problemas em diversas disciplinas e do dia a dia em diversas áreas de conh

é de extrema importância para a sociedade, pois capacita as pessoas a resolverem problemas complexos através do desenvolvimento de um conjunto útil de habilidades: decomposição, abstração, reconhecimento de padrões e pensamento algorítmico

De acordo com [(MOHAGHEGH; MCCAULEY, 2016)](https://www.zotero.org/google-docs/?UN56lE) o pensamento computacional

Falar sobre o pensamento computacional e porque ele é importante, principalmente na educação infantil. Falar sobre os esforços de pesquisa para introduzir o pensamento computacional na educação e onde esse trabalho se encaixa

Dentro deste contexto, este trabalho procura fazer uma contribuição na área de

## PROBLEMA DE PESQUISA

Verificar e validar se existem ou não métricas para avaliação do PC na educação infantil. Se não houver, salientar que esta é a proposta do trabalho. Se houver, discutir as soluções existentes e argumentar porque a nossa solução será diferente

Nesta seção, você deve descrever qual é o problema a ser resolvido. É necessário evidenciar que existem questões em aberto, que o tema é complexo e que há interesse na comunidade em resolver o problema. O texto deve responder às seguintes perguntas:

Qual a relevância e complexidade do problema apresentado?

Existe alguma solução consolidada ou o problema ainda está em aberto?

Nesta seção, você deve ainda indicar quais as perguntas de pesquisa que você buscará responder por meio do seu trabalho. Usualmente, as perguntas permitem a formulação de uma ou mais hipóteses que serão apresentadas na seção a seguir (Solução Proposta).

É possível

### Solução Proposta

Smart toys são brinquedos inteligentes que utilizam recursos tecnológicos de diferentes formas e enriquecem a experiência de brincar, pois a possibilidade de acesso à internet provida pela tecnologia IOT (internet of things) aliados a disponibilidade de controladoras de baixo custo, permitem que sejam concebidos brinquedos que explorem novas formas de interagir com a criança. Conforme Shah e Haradi (2016), a mistura de controladores e nuvem levou à era de uma tendência emergente.

Este trabalho propõe a implementação de um sistema de comunicação entre o RoPE e outros dispositivos através da Internet, permitindo que o RoPE seja controlado e monitorado remotamente. O sistema 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.

### Delimitação de Escopo

Este trabalho limita-se ao desenvolvimento do 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.

### Justificativa

Qual é a relevância da solução da proposta?

Ao conectar o RoPE

Transformar o RoPE em um *Smart Toy* abre as portas para que novas funcionalidades e recursos sejam acrescentados, melhorando o brinquedo.

Qual é a complexidade da solução proposta?

Qual é a aplicabilidade da solução?

A solução é viável?

Qual é o seu diferencial a outros similares?

Qual é a motivação para ele?

## OBJETIVOS

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

### 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

### Objetivos Específicos

1. Analisar trabalhos similares;
2. Escolher um módulo Wi-Fi e conectá-lo ao RoPE para realizar o acesso à Internet;
3. Escolher um protocolo de comunicação para a troca de mensagens entre o RoPE e os dispositivos remotos;
4. Definir o conjunto de comandos do RoPE que poderão ser executados remotamente;
5. Definir o conjunto de dados do RoPE que poderão ser monitorados remotamente;
6. Desenvolver um aplicativo para validar a implementação;
7. Realizar um estudo de caso usando a versão evoluída do RoPE para auxiliar na avaliação do pensamento computacional;

## METODOLOGIA

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

A escolha do módulo Wi-Fi e do protocolo de comunicação usado na implementação do sistema foi feita levando em conta as necessidades e limitações do RoPE, as quais estão descritas e justificadas no texto dos capítulos 4.1 e 4.2.

Na etapa de projeto do sistema foram definidos os requisitos funcionais, o conjunto de comandos que poderão ser executados remotamente no RoPE e o conjunto de dados que poderão ser monitorados. Todos esses itens foram relacionados e documentados no texto do capítulo 4.3.

Na etapa de projeto do aplicativo de testes foram criados protótipos de telas demonstrando o funcionamento desejado. Foi também criado o projeto do banco de dados usado para dar suporte ao estudo de caso envolvendo a avaliação do pensamento computacional. O protótipo do banco de dados está documentado em um diagrama ER no final do capítulo 4.4.2 .

## ESTRUTURA DA DISSERTAÇÃO

Este documento foi dividido em quatro capítulos: (i) Introdução, (ii) Fundamentação teórica, (iii) Trabalhos relacionados e (iv) Desenvolvimento.

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

No Capítulo 2, Fundamentação Teórica, apresentou-se a revisão bibliográfica sobre: brinquedos inteligentes (*Smart Toys*) e o uso de internet das coisas (IOT) em brinquedos. Também foi feita uma breve introdução ao brinquedo RoPE, descrevendo seus recursos e como ele tem sido aplicado na educação infantil.

No Capítulo 3, Trabalhos relacionados, foi feito um levantamento de trabalhos similares envolvendo brinquedos inteligentes e IOT. Ao final foi feita uma análise comparativa mostrando as suas diferenças e seus pontos positivos e negativos.

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

# FUNDAMENTAÇÃO TEÓRICA

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum cursus sem vel quam fermentum, id dapibus nulla vulputate. Mauris hendrerit elementum nisl nec semper. Maecenas nec mi mollis, sollicitudin sapien commodo, tristique dui. Ut rhoncus scelerisque libero, non euismod metus consectetur at. Morbi ligula nisi, tempus sit amet venenatis ut, hendrerit nec magna.

## SMART TOYS

Novo brinquedos tecnológicos tem surgido a cada dia

orem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum cursus sem vel quam fermentum, id dapibus nulla vulputate. Mauris hendrerit elementum nisl nec semper. Maecenas nec mi mollis, sollicitudin sapien commodo, tristique dui. Ut rhoncus scelerisque libero, non euismod metus consectetur at. Morbi ligula nisi, tempus sit amet venenatis ut, hendrerit nec magna.

## USO DE IOT EM BRINQUEDOS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum cursus sem vel quam fermentum, id dapibus nulla vulputate. Mauris hendrerit elementum nisl nec semper. Maecenas nec mi mollis, sollicitudin sapien commodo, tristique dui. Ut rhoncus scelerisque libero, non euismod metus consectetur at. Morbi ligula nisi, tempus sit amet venenatis ut, hendrerit nec magna.

## 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 3 anos de pesquisas envolvendo aproximadamente 20 pesquisadores de várias disciplinas: Educação, Ciência da Computação, Design e Engenharias Mecânica, Computação e Arquitetura (RAABE et al., 2017). Suas principais fontes de inspiração foram, a linguagem Logo (PAPERT, 1980) e o Bee-Bot, um brinquedo de programar usado pelo grupo de pesquisa em um experimento para identificar o impacto deste tipo de brinquedo na aprendizagem infantil (RAABE; VIEIRA; ROSÁRIO, 2015).

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

Seguindo essa mesma linha de raciocínio, o RoPE foi projetado para ensinar programação de forma lúdica e tangível para crianças na faixa dos cinco anos, empregando um visual amigável e colorido. Ele permite que as crianças programem seus movimentos através de botões localizados em sua parte superior, os quais emitem um *feedback* imediato através de luzes e sons.

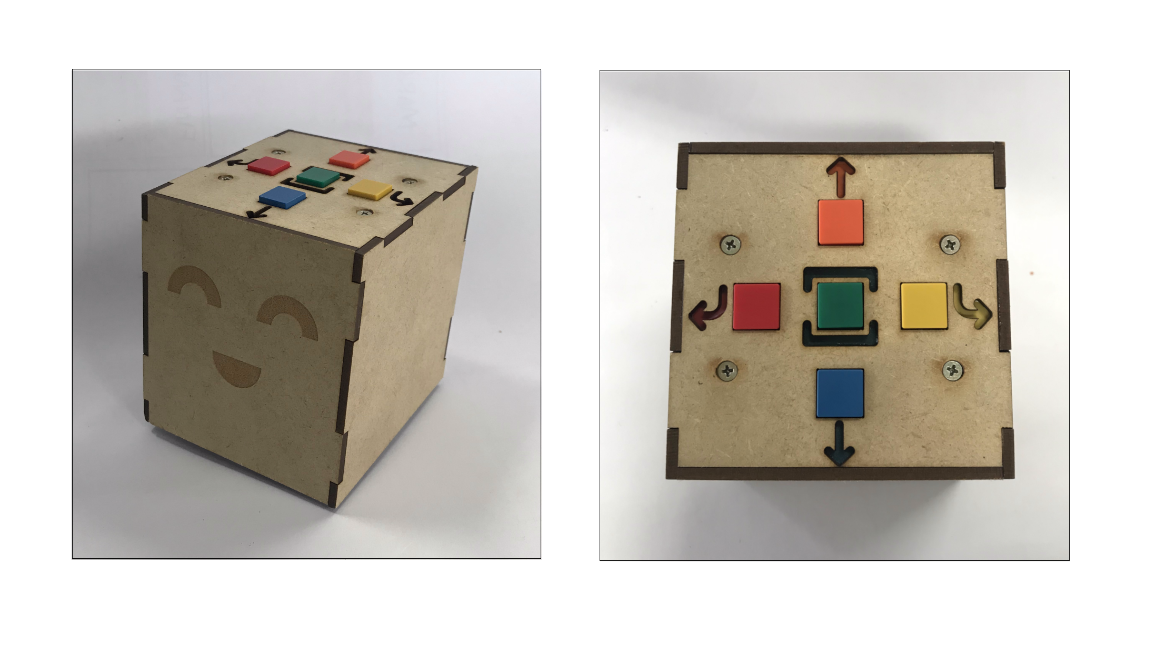


Figura 1. O brinquedo de programar RoPE e sua interface de programação

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

A experiência de brincar com o RoPE é enriquecida através de um conjunto de tapetes pedagógicos (Figura 2) 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”.

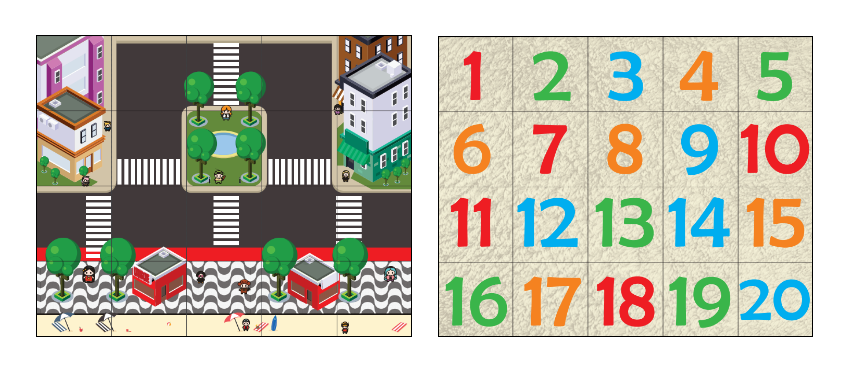


Figura 2. Tapetes pedagógicos do RoPE

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

O microcontrolador ATmega328P presente no RoPE é o mesmo chip embarcado no Arduino Uno, placa essa muito 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.

| **Imagem** | **Componente** |  | **Imagem** | **Componente** |
| --- | --- | --- | --- | --- |
|  | 2x Motor de passo  Modelo 28byj-48 |  | 1x Microcontrolador ATmega328P |
|  | 1x Driver para motor de passo  Modelo ULN2803AG |  | 1x Buzzer  Modelo KC-1206 |

Tabela 1. Principais componentes do RoPE

De (LIN, 2015)

Embora o RoPE seja um brinquedo tecnológico, orem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum cursus sem vel quam fermentum, id dapibus nulla vulputate. Mauris hendrerit elementum nisl nec semper. Maecenas nec mi mollis, sollicitudin sapien commodo, tristique dui. Ut rhoncus scelerisque libero, non euismod metus consectetur at. Morbi ligula nisi, tempus sit amet venenatis ut, hendrerit nec magn

# TRABALHOS RELACIONADOS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum cursus sem vel quam fermentum, id dapibus nulla vulputate. Mauris hendrerit elementum nisl nec semper. Maecenas nec mi mollis, sollicitudin sapien commodo, tristique dui. Ut rhoncus scelerisque libero, non euismod metus consectetur at. Morbi ligula nisi, tempus sit amet venenatis ut, hendrerit nec magna.

## Woobo 2019

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum cursus sem vel quam fermentum, id dapibus nulla vulputate. Mauris hendrerit elementum nisl nec semper. Maecenas nec mi mollis, sollicitudin sapien commodo, tristique dui. Ut rhoncus scelerisque libero, non euismod metus consectetur at. Morbi ligula nisi, tempus sit amet venenatis ut, hendrerit nec magna.

## TACTIC-KIBO

O K.

## ANÁLISE COMPARATIVA

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum cursus sem vel quam fermentum, id dapibus nulla vulputate. Mauris hendrerit elementum nisl nec semper. Maecenas nec mi mollis, sollicitudin sapien commodo, tristique dui. Ut rhoncus scelerisque libero, non euismod metus consectetur at. Morbi ligula nisi, tempus sit amet venenatis ut, hendrerit nec magna.

# DESENVOLVIMENTO

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

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

## MÓDULO WIFI ESP-01

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

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

Para melhor compreensão, é importante salientar que o termo módulo é utilizado para descrever uma placa de circuito impresso (PCI) que integra um chip principal aos seus periféricos como, por exemplo: cristal, memória flash, antena e outros componentes demandados. O módulo escolhido para o desenvolvimento do Smart RoPE foi o ESP-01, o qual integra o chip ESP8266[[1]](#footnote-1) da Espressif (MICROSHIP, 2020a) e é amplamente utilizado em projetos IoT em conjunto com o Arduino (baseadas em microcontroladores da família AVR da Atmel). A Figura 3 ilustra um exemplar do ESP-01 fabricado pela AIThinker.

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

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



Figura 3. Módulo ESP-01

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

De forma similar, o ESP8266 também oferece funções de economia de energia, que conforme apresentado no trabalho de (MEHTA, 2015) podem reduzir o consumo de corrente para valores abaixo de 0.012mA. Este consumo é muito mais baixo do que do próprio ATMega328P, que segundo o datasheet da (MICROSHIP, 2020b) consome entre 1.2mA a 2.7mA em modo de espera, quando conectado a uma fonte de energia de 5V e ativado o cristal interno de 8MHz, configuração usada pelo RoPE. As capacidades de economia de energia do ESP8266 já foram testadas por (MESQUITA et al., 2018) e os experimentos demonstraram que sob certas condições ele pode ser capaz de funcionar por até 4 dias consecutivos usando uma bateria de 1000mAh. Sendo assim, no quesito de consumo energia o ESP8266 se mostra uma ótima opção e deve atender à necessidade do projeto.

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

A partir da implementação do ESP8266 estes problemas poderiam ser solucionados com a inclusão do recurso de atualização *Over The Air* (OTA), o qual permite baixar e gravar uma nova versão do *firmware* a partir da nuvem. O uso desse recurso no ESP8266 é relativamente simples, pois a comunidade desenvolve e mantém um conjunto de bibliotecas no Github que implementam essa funcionalidade[[2]](#footnote-2). Um exemplo de código para atualização OTA pode ser visto na Figura 4. Outra facilidade disponibilizada pela atualização OTA é a gravação simultânea de firmware em múltiplas unidades, apropriado para o processo de manufatura em grande escala.

|  |
| --- |
|  |

Figura 4. Exemplo de código para atualização OTA

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

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

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Figura 5. Rede SofAP e página web hospedada no ESP8266

A Figura 5 demonstra em exemplo de página de configuração hospedada no ESP8266 e uma rede denominada “NodeMCU” usada para a conexão. Já na Figura 6 é possível visualizar um trecho de código simplificado que pode ser usado para criar uma página semelhante no RoPE e gravar os parâmetros na *EEPROM*.

|  |
| --- |
|  |

Figura 6. Código para configuração do RoPE

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

|  |  |  |
| --- | --- | --- |
|  | **RoPE (ATmega328p)** | **ESP-01 (ESP8266)** |
| **CPU** | Padrão: 16 MHz[[3]](#footnote-3)  Máximo: 20 MHz | Padrão: 80 MHz  Máximo: 160 MHz |
| **RAM** | 2 kB | 36 kB |
| **Flash** | 32 kB | Padrão: 1 MB[[4]](#footnote-4)  Máximo: 16 MB |

Tabela 2. Comparação de hardware entre o ATmega328p e o ESP8266

## 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.

|  |
| --- |
| casa/garagem/portao  Assinante  Y  Fechar  Ligar  casa/garagem/portao  casa/cozinha/lampada  casa/cozinha/lampada  Broker  Assinante  X |

Figura 7. Arquitetura do MQTT

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

Para uma melhor compreensão, o diagrama da Figura 7 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)](https://www.zotero.org/google-docs/?dYH4o2) é um protocolo fácil de ser implementado e independentemente da linguagem de programação. É possível encontrar bibliotecas de MQTT em diversas linguagens, incluindo C e PHP que serão usadas nesse trabalho. A implementação em PHP será útil para desenvolver uma aplicação de estudo de caso ao final do trabalho. Já a implementação em C será usada no ESP8266 e conta com mais de 10 opções de bibliotecas, conforme apontado por (OLIVEIRA et al., 2018).

Um dos objetivos do trabalho é permitir que o RoPE tenha seus dados coletados e seja controlado remotamente através da Internet por outros dispositivos, porém é preciso garantir que somente dispositivos autorizados sejam capazes de obter esse acesso. Para auxiliar nessa tarefa o MQTT possui um mecanismo de segurança baseado em usuário e senha (SONI; MAKWANA, 2017) com suporte a criptografia TLS dependendo do *broker* utilizado. Nesse modelo de segurança, os usuários precisam estar registrados no *broker* e os dispositivos devem enviar as suas credenciais na fase de estabelecimento da conexão.

O mecanismo de autenticação via usuário senha é o suficiente para o desenvolvimento deste projeto, no entanto, apresenta alguns problemas. Segundo (BHAWIYUGA; DATA; WARDA, 2017), a necessidade de envio das credenciais a cada conexão facilita a sua captura por possíveis *sniffers[[5]](#footnote-5)* conectados à rede. Ainda de acordo com o autor, pelo fato de as credenciais nunca expirarem, uma vez que tenham sido capturadas elas podem ser usadas indefinidamente para ataques até que sejam alteradas no *broker*. A literatura apresenta alguns trabalhos que podem ser usados como referência para melhorar a camada de segurança do RoPE em trabalhos futuros: (SINGH et al., 2015), (SHIN et al., 2016), (PERRONE et al., 2017) e (PATEL; DOSHI, 2020).

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

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

## ARQUITETURA DO PROJETO

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

### Requisitos funcionais

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

| **Requisitos funcionais do Smart RoPE** | |
| --- | --- |
|  | Os dispositivos externos devem conseguir mover o RoPE para frente e para trás |
|  | Os dispositivos externos devem conseguir girar o RoPE em sentido horário e anti-horário |
|  | Os dispositivos externos devem conseguir ligar e desligar o som do RoPE |
|  | Os dispositivos externos devem conseguir alterar a tonalidade do som do RoPE |
|  | Os dispositivos externos devem conseguir reproduzir sons no RoPE |
|  | Os dispositivos externos devem conseguir ligar e desligar os LEDs do RoPE |
|  | Os dispositivos externos devem conseguir inserir instruções na memória do RoPE |
|  | Os dispositivos externos devem conseguir remover instruções da memória do RoPE |
|  | Os dispositivos externos devem conseguir alterar o tamanho da memória de instruções do RoPE |
|  | Os dispositivos externos devem conseguir limpar completamente a memória de instruções do RoPE |
|  | Os dispositivos externos devem conseguir executar um programa armazenado na memória do RoPE |
|  | Os dispositivos externos devem conseguir interromper um programa que esteja sendo executado |
|  | Os dispositivos externos devem conseguir instruir o RoPE a aguardar um intervalo de tempo |
|  | Os dispositivos externos devem conseguir simular o pressionamento dos botões do RoPE |
|  | Os dispositivos externos devem ser notificados quando o RoPE se mover |
|  | Os dispositivos externos devem ser notificados quando o RoPE girar |
|  | Os dispositivos externos devem ser notificados quando o nível de bateria do RoPE mudar |
|  | Os dispositivos externos devem ser notificados quando o som do RoPE for ligado ou desligado |
|  | Os dispositivos externos devem ser notificados quando a tonalidade dos sons do RoPE for alterada |
|  | Os dispositivos externos devem ser notificados quando um som for reproduzido no RoPE |
|  | Os dispositivos externos devem ser notificados quando um LED do RoPE for ligado ou desligado |
|  | Os dispositivos externos devem ser notificados quando uma instrução for inserida na memória do RoPE |
|  | Os dispositivos externos devem ser notificados quando uma instrução for removida da memória do RoPE |
|  | Os dispositivos externos devem ser notificados quando a memória do RoPE for redimensionada |
|  | Os dispositivos externos devem ser notificados quando a memória do RoPE for apagada |
|  | Os dispositivos externos devem ser notificados quando o RoPE iniciar a execução de um programa |
|  | Os dispositivos externos devem ser notificados do progresso da execução de um programa |
|  | Os dispositivos externos devem ser notificados quando o RoPE finalizar a execução de um programa |
|  | Os dispositivos externos devem ser notificados quando um botão do RoPE for pressionado |
|  | Os dispositivos externos devem ser notificados do estado inicial do RoPE quando ele for ligado |
|  | Os dispositivos externos devem conseguir obter uma lista com o número de série de todas as unidades do RoPE conectadas ao *broker* |
|  | Os dispositivos externos devem conseguir registrar um *tema* personalizado e requisitar que o RoPE publique seus eventos nesse *tema* |
|  | Os dispositivos externos devem conseguir remover um *tema* personalizado registrado anteriormente para que o RoPE pare de publicar os eventos nesse *tema* |

Tabela 3. Requisitos funcionais do Smart Rope

### Conexão Wi-Fi

Para cumprir o objetivo de conectar o RoPE à Internet é necessário acoplar o módulo ESP-01 à placa do brinquedo. A conexão é bem simples, como pode ser visto na Figura 8, pois a PCI do RoPE já expõe um conjunto de pinos para a conexão com módulos externos usando a interface serial (*UART*).

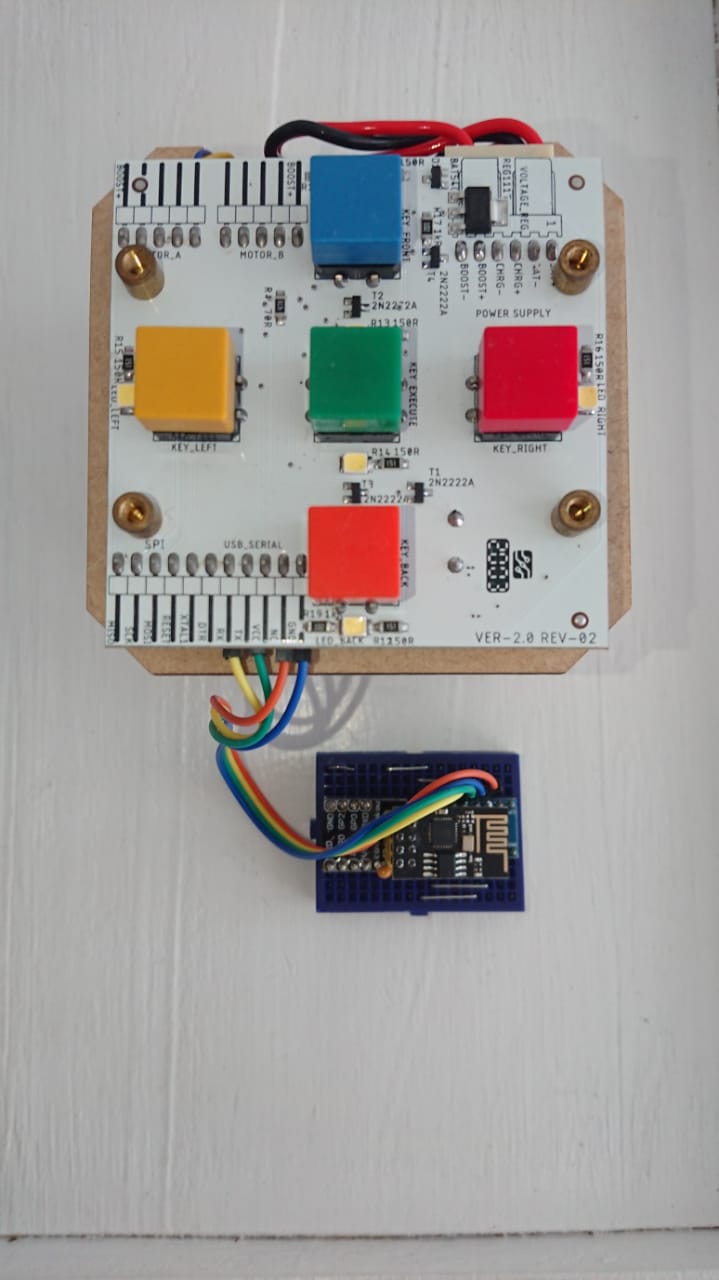


Figura 8. Conexão do ESP-01 com o RoPE

Após a ligação do módulo, a segunda etapa é realizar a configuração da rede Wi-Fi. Esse processo é realizado programaticamente fornecendo o SSID da rede e a senha da rede que se deseja conectar, conforme ilustrado na Figura 9. Uma vez conectado à rede Wi-Fi o ESP-01 já estará pronto para transmitir e receber dados através da Internet.

|  |
| --- |
|  |

Figura 9. Trecho de código para configuração da rede Wi-Fi

### Roteamento das mensagens

O Smart Rope funcionará usando o protocolo MQTT, abordado no capítulo 4.2. Conforme é estabelecido nesse protocolo, para que ocorra a comunicação entre o RoPE e os demais dispositivos é necessário que sejam definidos os *temas* nos quais eles deverão publicar e se inscrever. Para esse projeto foram definidos três temas, os quais estão relacionados na Tabela 4 juntamente com seu propósito.

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

Tabela 4. Temas do Smart RoPE

No recebimento de comandos para controle remoto do RoPE as mensagens são originadas no dispositivo externo e enviadas ao *broker*, que as encaminha para o RoPE. Estas mensagens são recebidas pelo módulo ESP-01 que as retransmite ao microcontrolador ATmega328P através da interface serial (*UART*) para serem então processadas. O caminho percorrido pelas mensagens pode ser visto de maneira simplificada a seguir na Figura 10.

|  |
| --- |
| Forma, Polígono  Descrição gerada automaticamenteForma, Polígono  Descrição gerada automaticamenteForma, Polígono  Descrição gerada automaticamenteForma, Polígono  Descrição gerada automaticamente  Editor  RoPE  Broker  Assinante |

Figura 10. Roteamento das mensagens de controle remoto

Para que as mensagens percorram esse caminho ocorrer a seguinte sequência de eventos. Primeiro o RoPE se conecta ao *broker* e publica uma mensagem contendo o seu número de série no *tema* “rope/list”. Ao mesmo tempo, o dispositivo que deseja enviar comandos ao RoPE se conecta ao *broker* e se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o número de série do RoPE para o dispositivo, que pode armazená-lo para uso futuro. Em seguida, o dispositivo publica uma mensagem contendo o comando que deve ser executado pelo RoPE no *tema* “rope/{serial}/control”, onde “{serial}” é o número de série do RoPE. Ao mesmo tempo, o RoPE se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o comando ao RoPE, que pode então executá-lo. Para uma melhor compreensão, a Figura 11 apresenta um diagrama de sequência exemplificando esse processo.

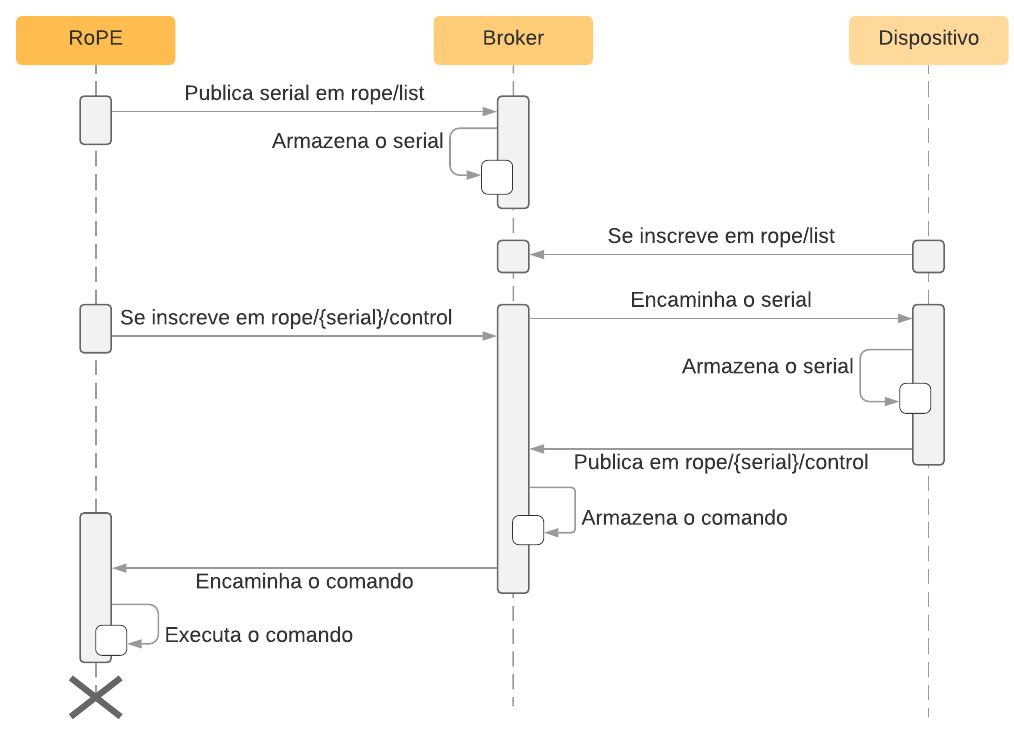


Figura 11. Sequência para envio de comandos para o RoPE

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

No envio de eventos aos dispositivos, o caminho é o inverso do anterior. As mensagens são originadas dentro do RoPE pelo microcontrolador ATmega328P, que as transmite ao módulo ESP-01 através da interface serial (*UART*). No módulo ESP-01 as mensagens são enviadas para o *broker*, que por sua vez as encaminha para o dispositivo a fim de serem então processadas. O caminho percorrido pelas mensagens pode ser visto de maneira simplificada a seguir na Figura 12.

|  |
| --- |
| Assinante  Broker  Editor  RoPE |

Figura 12. Encaminhamento das mensagens de evento

Para que as mensagens percorram esse caminho ocorrer a seguinte sequência de eventos. Primeiro o RoPE se conecta ao *broker* e publica uma mensagem contendo o seu número de série no *tema* “rope/list”. Ao mesmo tempo, o dispositivo que deseja observar os eventos do RoPE se conecta ao *broker* e se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o número de série para o dispositivo, que pode armazená-lo para uso futuro. Em seguida, o RoPE publica uma mensagem contendo o evento ocorrido no *tema* “rope/{serial}/events”, onde “{serial}” é o número de série do RoPE. Ao mesmo tempo, o dispositivo externo se inscreve no mesmo *tema*. Após a inscrição, o *broker* encaminha a mensagem contendo o evento ao dispositivo, que pode então processá-lo. Para uma melhor compreensão, a Figura 13 apresenta um diagrama de sequência exemplificando esse processo.

Diagrama

Descrição gerada automaticamente

Figura 13. Sequência para o envio de eventos para os dispositivos

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

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

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

### Mensagens de controle

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

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

Tabela 5. Comando MOVE

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

Tabela 6. Comando TURN

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

Tabela 7. Comando TOGGLE\_SOUND

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

Tabela 8. Comando TUNE\_SOUND

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

Tabela 9. Comando PLAY\_SOUND

|  |  |  |
| --- | --- | --- |
| **Mensagem de controle** | | |
| **Comando** | **TOGGLE\_LED** | |
| **Descrição** | Liga ou desliga um determinado LED (**RF6**) | |
| **Parâmetros** | | |
| **Led** | O LED que deverá ser alternado | String: [FRONT | LEFT | RIGHT | BACK | CENTER] |
| **State** | O novo estado do LED | String: [ON | OFF] |
| **Exemplos** | | |
| {"command": "TOGGLE\_LED ", "parameters": {"led": "FRONT", "state": "ON"}} | | Liga o LED frontal |
| {"command": "TOGGLE\_LED", "parameters": {"led": "BACK", "state": "OFF"}} | | Desliga o LED traseiro |

Tabela 10. Comando TOGGLE\_LED

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

Tabela 11. Comando INSERT\_INSTRUCTION

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

Tabela 12. Comando REMOVE\_INSTRUCTION

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

Tabela 13. Comando RESIZE\_MEMORY

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

Tabela 14. Comando CLEAR\_MEMORY

|  |  |
| --- | --- |
| **Mensagem de controle** | |
| **Comando** | **EXECUTE\_PROGRAM** |
| **Descrição** | Executa o programa armazenado na memória (**RF11**) |
| **Exemplo** | |
| {"command": "EXECUTE\_PROGRAM "} | |

Tabela 15. Comando EXECUTE\_PROGRAM

|  |  |
| --- | --- |
| **Mensagem de controle** | |
| **Comando** | **ABORT\_PROGRAM** |
| **Descrição** | Interrompe o programa que está sendo executado (**RF12**) |
| **Exemplo** | |
| {"command": "ABORT\_PROGRAM "} | |

Tabela 16. Comando ABORT\_PROGRAM

|  |  |  |
| --- | --- | --- |
| **Mensagem de controle** | | |
| **Comando** | **WAIT** | |
| **Descrição** | Pausa o RoPE por um determinado intervalo de tempo (**RF13**) | |
| **Parâmetros** | | |
| **Interval** | O intervalo de tempo (em segundos) que o RoPE deve aguardar | Float: > 0.0 |
| **Exemplos** | | |
| {"command": "WAIT", "parameters": {"interval": 2.5}} | | Pausa o RoPE por 2 segundos e meio |
| {"command": "WAIT", "parameters": {"interval": 0.1}} | | Pausa o RoPE por 100 milissegundos |

Tabela 17. Comando WAIT

|  |  |  |
| --- | --- | --- |
| **Mensagem de controle** | | |
| **Comando** | **PRESS\_BUTTON** | |
| **Descrição** | Simula o pressionamento de um botão pelo usuário (**RF14**) | |
| **Parâmetros** | | |
| **Button** | O botão que será pressionado | String: [MOVE\_FORWARD | MOVE\_BACKWARDS | TURN\_LEFT | TURN\_RIGHT | EXECUTE] |
| **Exemplos** | | |
| {"command": "PRESS\_BUTTON", "parameters": {"button": "TURN\_LEFT"}} | | Pressiona o botão “Girar para a esquerda” |
| {"command": "PRESS\_BUTTON", "parameters": {"button": "TURN\_RIGHT"}} | | Pressiona o botão “Girar para a direita” |

Tabela 18. Comando PRESS\_BUTTON

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

Tabela 19. Comando ADD\_EVENT\_LISTENER

|  |  |  |
| --- | --- | --- |
| **Mensagem de controle** | | |
| **Comando** | **REMOVE\_EVENT\_LISTENER** | |
| **Descrição** | Remove um *tema* previamente registrado para receber eventos do RoPE (**RF33**) | |
| **Parâmetros** | | |
| **Topic** | O *tema* no qual o RoPE deve parar de publicar os eventos | String |
| **Exemplos** | | |
| {"command": "REMOVE\_EVENT\_LISTENER", "parameters": {  "topic": "ct-assessment/session-1/test-1"}} | | Remove o *tema* “ct-assessment/session-1/test-1” previamente registrado na lista de *temas* que terão os eventos publicados. Quando um evento ocorrer no RoPE ele não será mais publicado no *tema* padrão *tema* adicional “rope/{serial}/events/ct-assessment/session-1/test-1”, somente no *tema* padrão “rope/{serial}/events” |

Tabela 20. Comando REMOVE\_EVENT\_LISTENER

### Mensagens de evento

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

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

Tabela 21. Evento MOVED

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

Tabela 22. Evento TURNED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **BATTERY\_CHARGE\_CHANGED** | |
| **Descrição** | Evento disparado quando a carga da bateria se altera (**RF17**) | |
| **Dados** | | |
| **Previous charge** | O nível de carga anterior da bateria (em percentual) | Float: [0.0..100.0] |
| **Current charge** | O nível de carga atual da bateria (em percentual) | Float: [0.0..100.0] |
| **Critical Treshold** | O limiar de carga antes da bateria entrar em estado crítico (em percentual) | Float: [0.0..100.0] |
| **Exemplo** | | |
| {"event": "BATTERY\_CHARGE\_CHANGED", "data": {  "previous\_charge": 63.0, "current\_charge": 59.0, "critical\_treshold": 5.0}} | | Evento indicando que o nível de carga da bateria diminuiu de 63.0% para 59.0%. O nível está acima do limiar de bateria crítica (5.0%), portanto a bateria não precisa ser recarregada no momento |

Tabela 23. Evento BATTERY\_CHARGE\_CHANGED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **SOUND\_TOGGLED** | |
| **Descrição** | Evento disparado quando o som é ligado ou desligado (**RF18**) | |
| **Dados** | | |
| **Previous State** | O estado anterior do som | String: [ON | OFF] |
| **Current State** | O estado atual do som | String: [ON | OFF] |
| **Exemplo** | | |
| {"event": "SOUND\_TOGGLED", "data": {  "previous\_state": "OFF", "current\_state": "ON"}} | | Evento indicando que o som estava desligado e agora foi ligado |

Tabela 24. Evento SOUND\_TOGGLED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **SOUND\_TUNED** | |
| **Descrição** | Evento disparado quando a tonalidade do *buzzer* é alterada (**RF19**) | |
| **Dados** | | |
| **Previous Variation** | A variação de semitons anterior | Int: [-36..36] |
| **Current Variation** | A variação de semitons atual | Int: [-36..36] |
| **Exemplo** | | |
| {"event": "SOUND\_TUNED", "data": {  "previous\_variation": 1, "current\_variation": 7}} | | Evento indicando que a variação de tonalidade foi alterada de 1 semitom para 7 semitons |

Tabela 25. Evento SOUND\_TUNED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **SOUND\_PLAYED** | |
| **Descrição** | Evento disparado quando um som é reproduzido no *buzzer* (**RF20**) | |
| **Dados** | | |
| **Mode** | O modo de usado para reproduzir o som | String: [NOTE | FREQUENCY] |
| **Note** | A nota musical que foi reproduzida | String: [C | Cs | Db | D | Ds | Eb | E | F | Fs | Gb | G | Gs | Ab | A | As | Bb | B] |
| **Octave** | A oitava da nota que foi reproduzida | Int: [0..8] |
| **Duration** | A duração da nota em segundos | Float: [0.0..60.0] |
| **Pause (Opcional)** | A pausa (em segundos) que foi gerada após reproduzir a nota (se houver) | Float: [0.0..60.0] |
| **Exemplos** | | |
| {"event": "SOUND\_PLAYED", "data": {  "mode": "NOTE", "note": "C", "octave": 5, "frequency": 523.25,  "duration": 1.5, "pause": 0.5}} | | Evento indicando que a nota C5 foi reproduzida por 1 segundo e meio, com uma pausa de meio segundo após a reprodução. A frequência, 523.25 Hz, foi inferida a partir da nota |
| {"event": "SOUND\_PLAYED", "data": {  "mode": "FREQUENCY", "frequency": 523.25, "note": "C", "octave": 5,  "duration": 1.5, "pause": 0.5}} | | Evento indicando que a frequência 523.25 Hz foi reproduzida por 1 segundo e meio, com uma pausa de meio segundo após a reprodução. A nota, C5, foi inferida a partir da frequência |
| {"event": "SOUND\_PLAYED", "data": {  "mode": "FREQUENCY", "frequency": 510.17,  "duration": 1.5, "pause": 0.5}} | | Evento indicando que a frequência 510.17 Hz foi reproduzida por 1 segundo e meio, com uma pausa de meio segundo após a reprodução. A nota não pôde ser inferida pois a frequência não representa uma nota válida |

Tabela 26. Evento SOUND\_PLAYED

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

Tabela 27. Evento LED\_TOGGLED

|  |  |  |
| --- | --- | --- |
| **Mensagem de Evento** | | |
| **Evento** | **INSTRUCTION\_ INSERTED** | |
| **Descrição** | Evento disparado quando uma instrução é inserida na memória do RoPE (**RF22**) | |
| **Dados** | | |
| **Instruction** | A instrução que foi inserida na memória | String: [MOVE\_FORWARD | MOVE\_BACKWARDS | TURN\_LEFT | TURN\_RIGHT] |
| **Index** | A posição da memória (começando em 0) na qual a instrução foi inserida | Int: [0..?] |
| **Memory** | Objeto contendo informações úteis sobre a memória | Object |
| **Exemplo** | | |
| {"event": "INSTRUCTION\_INSERTED", "data": {  "instruction": "TURN\_RIGHT", "index": 2,  "memory": {  "previous\_state": {  "size": 4, "used": 1, "free": 3,  "insertions": 1, "removals": 0, "cleanings": 0, "resizings": 0,  "content": ["MOVE\_FORWARD", "", "", ""]},  "current\_state": {  "size": 4, "used": 2, "free": 2,  "insertions": 2, "removals": 0, "cleanings": 0, "resizings": 0,  "content": ["MOVE\_FORWARD", "", "TURN\_RIGHT", ""]}  }}} | | Evento indicando que a instrução “Girar para a esquerda” foi inserida na terceira posição da memória. A memória, que continha apenas 1 instrução “Mover para frente”, agora tem 2 instruções |

Tabela 28. Evento INSTRUCTION\_INSERTED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **INSTRUCTION\_REMOVED** | |
| **Descrição** | Evento disparado quando uma instrução é removida da memória do RoPE (**RF23**) | |
| **Dados** | | |
| **Instruction** | A instrução que foi removida da memória | String: [MOVE\_FORWARD | MOVE\_BACKWARDS | TURN\_LEFT | TURN\_RIGHT] |
| **Index** | A posição da memória (começando em 0) da qual a instrução foi removida | Int: [0..?] |
| **Memory** | Objeto contendo informações úteis sobre a memória | Object |
| **Exemplo** | | |
| {"event": "INSTRUCTION\_REMOVED", "data": {  "instruction": "MOVE\_FORWARD", "index": 0,  "memory": {  "previous\_state": {"size": 4, "used": 2, "free": 2,  "insertions": 2, "removals": 0, "cleanings": 0, "resizings": 0,  "content": ["MOVE\_FORWARD", "", "TURN\_RIGHT", ""]},  "current\_state": {"size": 4, "used": 1, "free": 3,  "insertions": 2, "removals": 1, "cleanings": 0, "resizings": 0,  "content": ["", "", "TURN\_RIGHT", ""]}  }}} | | Evento indicando que uma instrução “Mover para frente” foi removida da primeira posição da memória. A memória, que continha 2 instruções, agora contém apenas 1 instrução “Girar para a direita” |

Tabela 29. Evento INSTRUCTION\_REMOVED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **MEMORY\_RESIZED** | |
| **Descrição** | Evento disparado quando a memória do RoPE é redimensionada (**RF24**) | |
| **Dados** | | |
| **Memory** | Objeto contendo informações úteis sobre a memória | Object |
| **Exemplo** | | |
| {"event": "MEMORY\_RESIZED", "data": {  "memory": {  "previous\_state": {  "size": 2, "used": 1, "free": 1,  "insertions": 1, "removals": 0, "cleanings": 0, "resizings": 0,  "content": ["", "TURN\_RIGHT"]},  "current\_state": {  "size": 4, "used": 1, "free": 3,  "insertions": 1, "removals": 0, "cleanings": 0, "resizings": 1,  "content": ["", "TURN\_RIGHT", "", ""]}  }}} | | Evento indicando que a memória foi redimensionada de 2 para 4 instruções. A instrução “Girar para a direita”, que se encontrava na posição 1, não foi afetada |

Tabela 30. Evento MEMORY\_RESIZED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **MEMORY\_CLEARED** | |
| **Descrição** | Evento disparado quando a memória do RoPE é apagada (**RF25**) | |
| **Dados** | | |
| **Memory** | Objeto contendo informações úteis sobre a memória | Object |
| **Exemplo** | | |
| {"event": "MEMORY\_CLEARED", "data": {  "memory": {  "previous\_state": {  "size": 4, "used": 2, "free": 2,  "insertions": 2, "removals": 0, "cleanings": 0, "resizings": 0,  "content": ["", "TURN\_RIGHT", "", "TURN\_LEFT"]},  "current\_state": {  "size": 4, "used": 0, "free": 4,  "insertions": 2, "removals": 0, "cleanings": 1, "resizings": 0,  "content": ["", "", "", ""]}  }}} | | Evento indicando que a memória foi apagada, liberando 2 posições que estavam ocupadas por instruções |

Tabela 31. Evento MEMORY\_CLEARED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **PROGRAM\_STARTED** | |
| **Descrição** | Evento disparado quando o RoPE inicia a execução de um programa (**RF26**) | |
| **Dados** | | |
| **Execution** | Objeto contendo informações sobre a execução atual | Object |
| **Memory** | Objeto contendo informações úteis sobre a memória | Object |
| **Exemplo** | | |
| {"event": "PROGRAM\_STARTED ", "data": {  "execution": {  "progress": 0.0,  "statistics": {"started": 2, "completed": 1, "aborted": 0}},  "memory": {  "current\_state": {  "size": 3, "used": 3, "free": 0,  "insertions": 3, "removals": 0, "cleanings": 0, "resizings": 0,  "content": ["MOVE\_FORWARD", "TURN\_RIGHT", "TURN\_LEFT"]}  }}} | | Evento indicando que um programa foi iniciado. O programa que está sendo executado é o que está armazenado na memória |

Tabela 32. Evento PROGRAM\_STARTED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **INSTRUCTION\_EXECUTED** | |
| **Descrição** | Evento disparado quando a próxima instrução do programa é executada (**RF27**) | |
| **Dados** | | |
| **Execution** | Objeto contendo informações sobre a execução atual | Object |
| **Memory** | Objeto contendo informações úteis sobre a memória | Object |
| **Exemplo** | | |
| {"event": "INSTRUCTION\_EXECUTED", "data": {  "execution": {  "instruction": 1, "progress": 66.0,  "statistics": {"started": 2, "completed": 1, "aborted": 0}},  "memory": {  "current\_state": {  "size": 3, "used": 3, "free": 0,  "insertions": 3, "removals": 0, "cleanings": 0, "resizings": 0,  "content": ["MOVE\_FORWARD", "TURN\_RIGHT", "TURN\_LEFT"]}  }}} | | Evento indicando que a segunda instrução do programa foi executada, representando um progresso de 66.0% na execução do programa. O programa que está sendo executado é o que está armazenado na memória |

Tabela 33. Evento INSTRUCTION\_EXECUTED

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

Tabela 34. Evento PROGRAM\_FINISHED

|  |  |  |
| --- | --- | --- |
| **Mensagem de evento** | | |
| **Evento** | **BUTTON\_PRESSED** | |
| **Descrição** | Evento disparado quando um botão do RoPE é pressionado (**RF29**) | |
| **Dados** | | |
| **Button** | O botão que foi pressionado | String: [MOVE\_FORWARD | MOVE\_BACKWARDS | TURN\_LEFT | TURN\_RIGHT | EXECUTE] |
| **Exemplos** | | |
| {"event": " BUTTON\_PRESSED", "data": {"button": "TURN\_LEFT"}} | | Evento indicando que o botão “Girar para a esquerda” foi pressionado |
| {"event": "BUTTON\_PRESSED", "data": {"button": "TURN\_RIGHT"}} | | Evento indicando que o botão “Girar para a direita” foi pressionado |

Tabela 35. Evento BUTTON\_PRESSED

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

Tabela 36. Evento SWITCHED\_ON

## VALIDAÇÃO

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

### Aplicativo de testes

O aplicativo de testes a ser desenvolvido vai possuir 3 telas. A tela inicial (Figura 14), consiste em uma lista de todas as unidades do RoPE que estão ligadas e conectadas à Internet. As unidades estarão identificadas pelo seu número de série e informações de propriedade (previamente gravadas no firmware). A tela deve possuir um campo para pesquisa, permitindo buscar unidades específicas do RoPE a partir dessas informações. O objetivo desta tela é permitir que seja feita a seleção da unidade do RoPE com a qual os testes serão realizados.

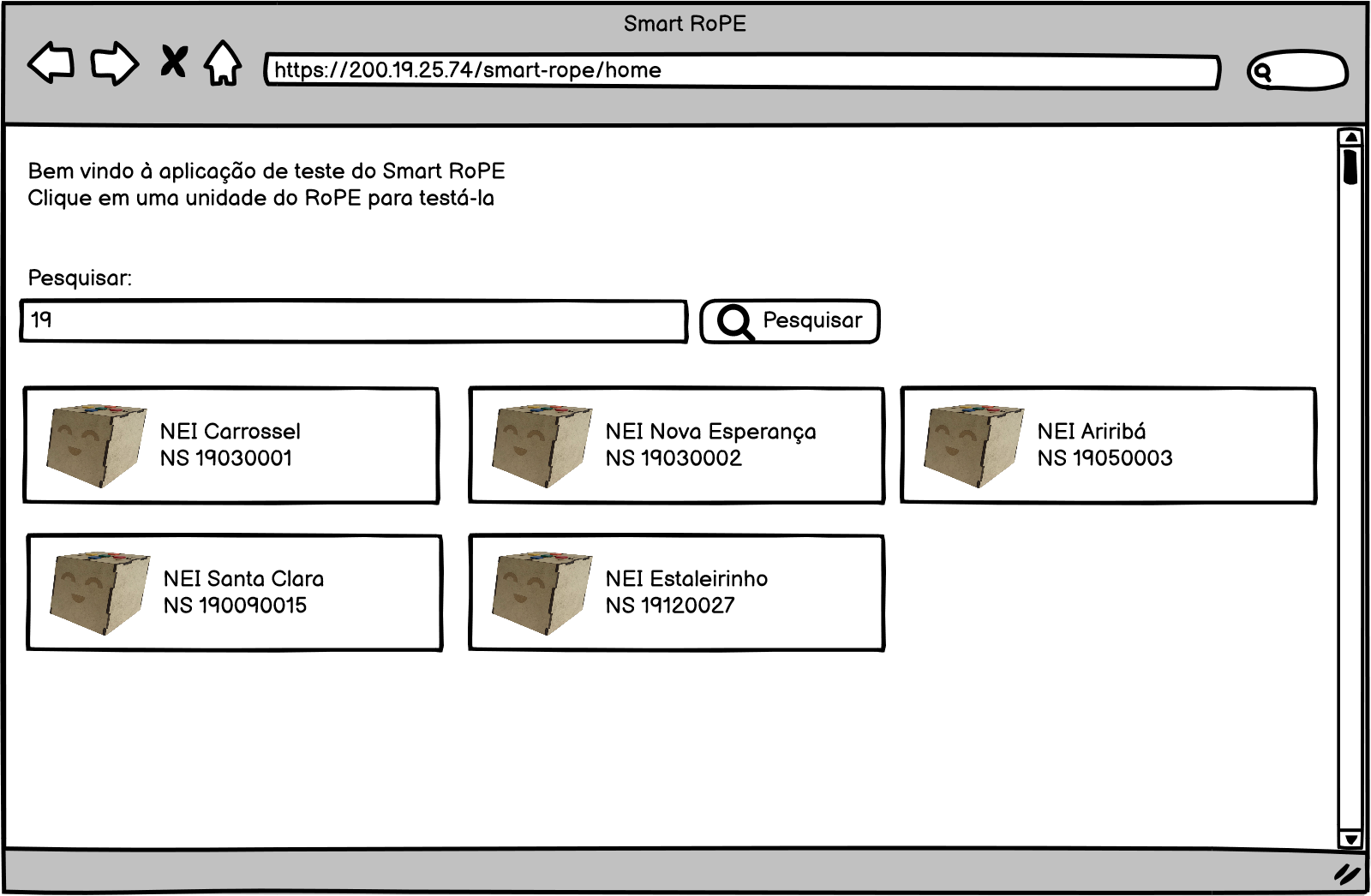


Figura 14. Protótipo da tela inicial do aplicativo de testes

Ao clicar ema unidade do RoPE na tela principal o aplicativo deve ser redirecionado à tela de testes da unidade (Figura 15). Nesta tela, são apresentadas novamente as informações básicas da unidade, a fim de identificá-la, bem como dois painéis de gerenciamento que serão usados para conduzir os testes.

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

* **Teclado direcional:** permite programar o RoPE simulando o pressionamento dos botões físicos e/ou enviar comandos para movimentá-lo instantaneamente na direção escolhida
* **Painel de LEDs:** permite enviar comandos para ligar e desligar os LEDs correspondentes do RoPE. Os LEDs são da mesma cor dos LEDs do RoPE e arranjados no mesmo layout do LEDs físicos do brinquedo
* **Controles de som:** permite ligar e desligar o som do RoPE e alterar a tonalidade dos sons reproduzidos pelo *buzzer*
* **Teclado musical:** um teclado que imita um piano com pelo menos duas oitavas musicais. Permite reproduzir no *buzzer* do RoPE a nota musical correspondente à tecla

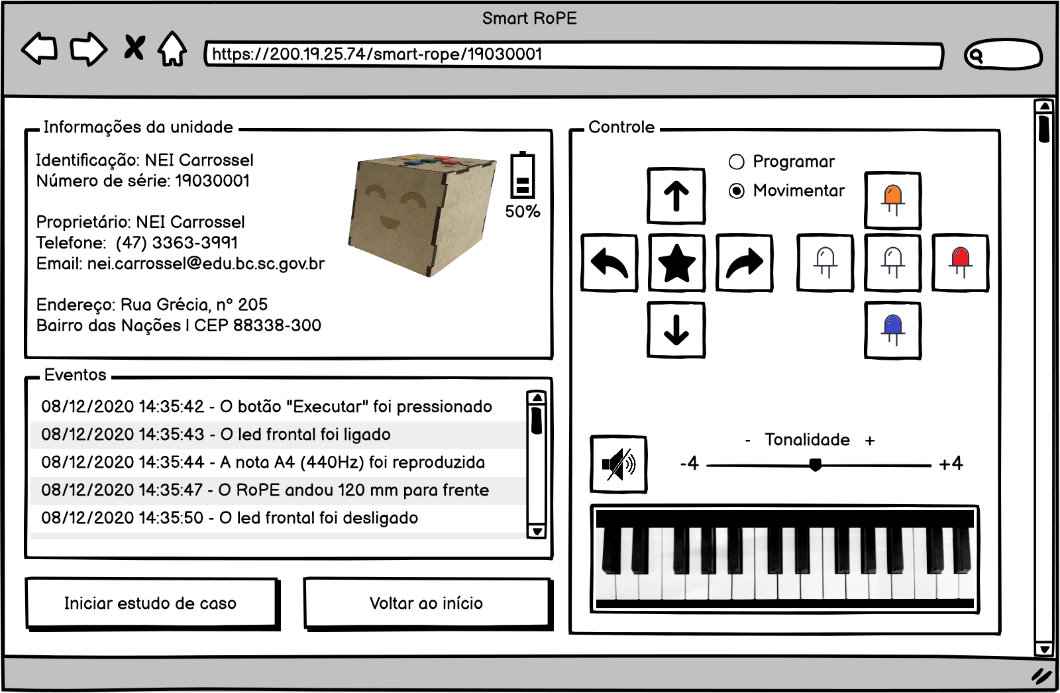


Figura 15. Protótipo da tela de teste da unidade do aplicativo

A validação do controle remoto é feita da seguinte forma:

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

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

1. O usuário escolhe um controle do painel, por exemplo, um LED
2. O usuário interage com o controle, por exemplo, clicando no LED
3. O usuário observa a lista de eventos
   1. Um novo item é acrescentado na lista, correspondente à ação realizada no RoPE. O evento é validado
   2. Nenhum item é acrescentado na lista ou o item acrescentado não corresponde à ação realizada no RoPE. O evento é invalidado
4. O processo é repetido escolhendo outro controle

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

### Estudo de caso

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

O teste em questão faz uso de um tapete do RoPE que foi elaborado especificamente com o propósito de permitir a avaliação do pensamento computacional (Figura 16). 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.

Logotipo, nome da empresa

Descrição gerada automaticamente

Figura 16. Tapete para avaliação do pensamento computacional

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

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

Tabela 37. Enunciados para a avalição do pensamento computacional

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

| **Variáveis do pensamento computacional** |
| --- |
| Número de instruções usadas para resolver o problema |
| Tempo total necessário para resolver o problema |
| Quantidade de erros cometidos |
| Quantidade de vezes que o teste precisou ser reiniciado |
| Quantidade de vezes que a criança interrompeu o programa no meio da execução |
| Se a criança concluiu o desafio usando uma solução válida diferente da esperada |
| Número de instruções a mais que foram usadas |
| Se a criança chegou ao objetivo, mas usando instruções proibidas pelo enunciado |
| Média de tempo entre cada instrução programada |
| Intervalo de tempo desde o início do teste até a programação da primeira instrução |
| Média de tempo para a solução dos enunciados |

Tabela 38. Variáveis do pensamento computacional

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

Dentro desse contexto, a proposta do trabalho é desenvolver uma ferramenta de apoio à realização dos testes, usando o mecanismo de propagação de eventos do *Smart RoPE* para facilitar e automatizar a mensuração dessas variáveis. A ferramenta será implementada dentro do aplicativo de testes proposto na seção 4.4.1 .

A tela principal da ferramenta (Figura 17) exibirá uma árvore com todas as etapas de teste e seus respectivos enunciados. O componente permitirá iniciar qualquer teste em qualquer ordem, ficando a cargo do operador selecionar os testes na ordem correta.

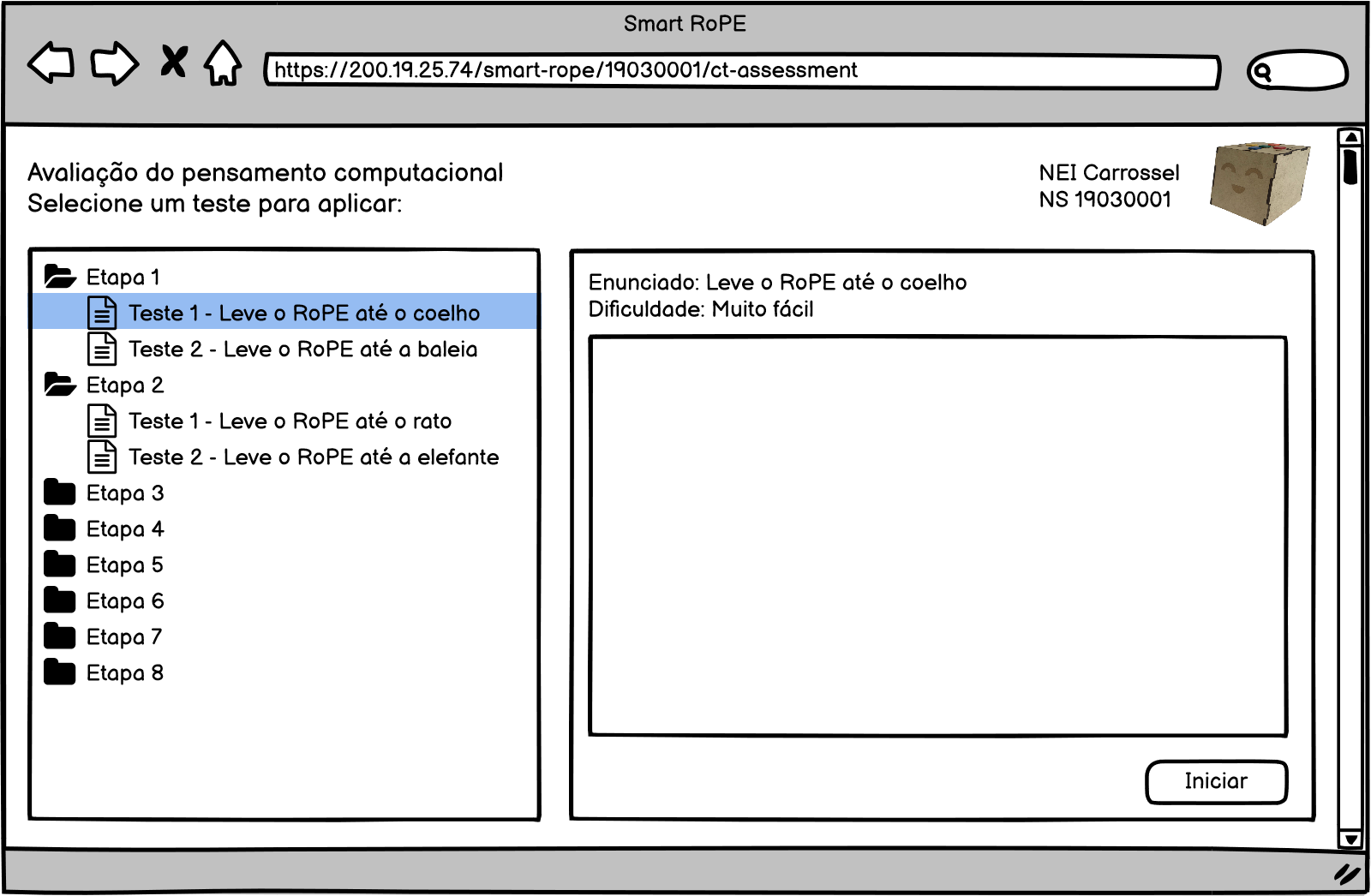


Figura 17. Tela principal do gerenciador de testes

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

1. O professor ligará o RoPE e aguardará até que ele esteja conectado na rede WiFi
2. O avaliador acessará o aplicativo, selecionará o RoPE correspondente e abrirá a tela de gerenciamento dos testes
3. O avaliador selecionará um teste na árvore e sinalizará ao professor quando estiver pronto para iniciar o experimento, informando ao professor qual teste foi escolhido
4. Ao ser sinalizado, o professor fará a preparação da criança da forma que achar necessário como, por exemplo, explicando o funcionamento do RoPE
5. Quando a criança estiver preparada, o professor (que terá uma cópia impressa dos testes a serem realizados) fará a leitura do enunciado para a criança e posicionará o brinquedo na posição inicial
6. Neste momento, o avaliador clicará no botão “Iniciar” para dar início à coleta das informações. O aplicativo começará a fazer o registro dos eventos enviados pelo RoPE e exibirá na tela essas informações para que o avaliador possa acompanhar (Figura 18). As informações que aparecerem na tela serão também registradas em um banco de dados para processamento posterior

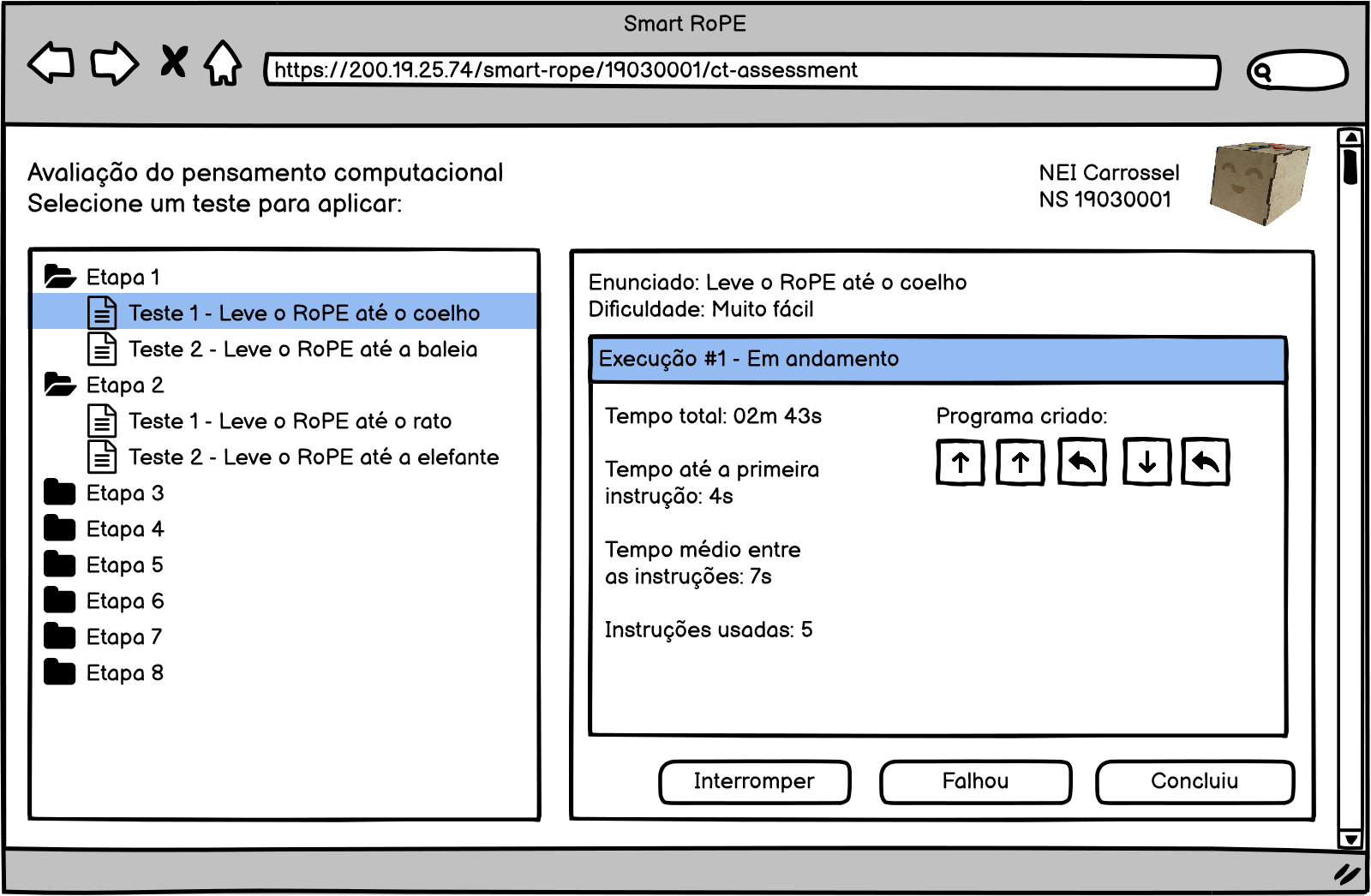


Figura 18. Execução de um teste em andamento

1. Ambos, o professor e o avaliador deixarão a criança livre para experimentar o brinquedo enquanto observam, devendo intervir somente nas seguintes situações:
   1. quando a criança concluir o desafio com sucesso. Neste cenário, o avaliador clicará no botão “Concluir”, sinalizando o aplicativo de que o desafio foi concluído com sucesso pela criança
   2. quando a criança desistir do desafio, quando a criança pedir para reiniciar o desafio ou quando se esgotar o tempo estipulado para a conclusão do desafio. Neste cenário, o avaliador clicará no botão “Falhou”, sinalizando o aplicativo de que a criança falhou em concluir o desafio
   3. quando houver qualquer outro acontecimento que impeça a conclusão do desafio como, por exemplo: (i) a criança necessitou ir ao banheiro, (ii) os pais da criança vieram levá-la para casa, (iii) outros eventos não previstos. Neste cenário, o avaliador clicará no botão “Interromper”, sinalizando o aplicativo de que o teste precisou ser interrompido, mas que não houve falha nem sucesso
2. Em todos os cenários descritos no item anterior, o aplicativo fará o registro da execução e voltará ao estado inicial (Figura 19). Neste instante, o avaliador pode:
   1. Clicar no registro da execução para visualizar os dados coletados
   2. Clicar no botão “Iniciar” para começar uma nova execução do mesmo teste (Figura 20). Retorna ao item 6
   3. Avança para o item 9

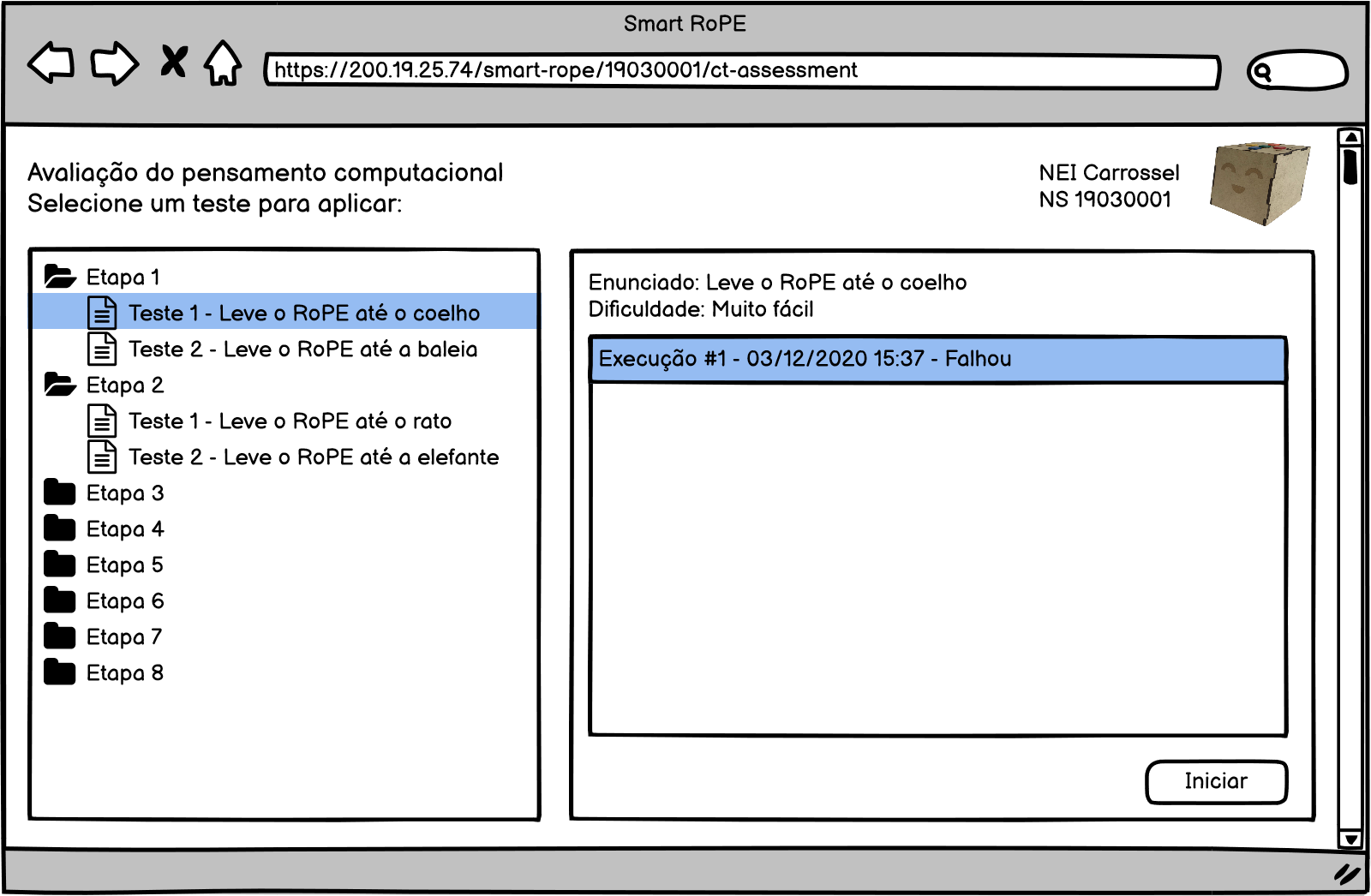


Figura 19. Os testes já executados ficam registrados

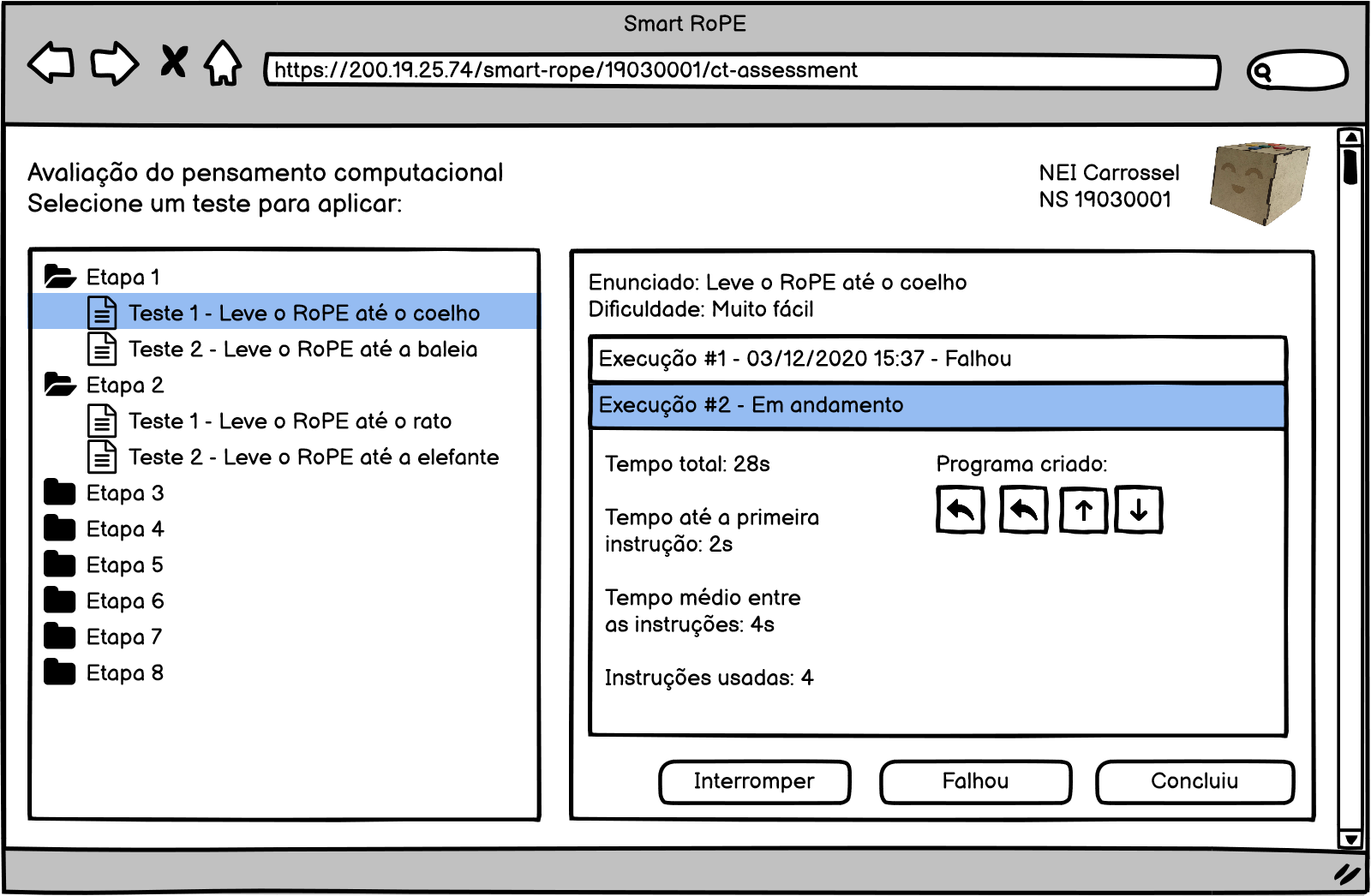


Figura 20. Um mesmo teste sendo executado novamente

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

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

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

Para melhor compreensão de como os dados serão armazenados foi modelado um diagrama de banco de dados (Figura 21) . As tabelas “test\_suite”, “test\_stage” e “test\_challenge” armazenam o teste do pensamento computacional, as etapas do teste e os enunciados de cada etapa, respectivamente. O registro das execuções dos testes, ilustrado anteriormente no protótipo da aplicação, são armazenadas na tabela “test\_executions”. Por fim, os eventos recebidos do RoPE bem como as variáveis computadas a partir desses eventos, são armazenados nas tabelas “rope\_event” e “test\_variable”, respectivamente.

Diagrama

Descrição gerada automaticamente

Figura 21. Diagrama ER do banco de dados do aplicativo Smart RoPE

**Plano de trabalho**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Quadro 1. Cronograma de execução para o segundo semestre do primeiro ano (2019/2)   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Atividade** | **Jul** | **Ago** | **Set** | **Out** | **Nov** | **Dez** | | Revisão bibliográfica | x | x |  |  |  |  | | Revisão sistemática |  |  | x | x |  |  | | Implementação do ambiente de testes |  |  |  |  | x | x | |
| Quadro 2. Cronograma de execução para o primeiro semestre do segundo ano (2020/1)   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Atividade** | **Jan** | **Fev** | **Mar** | **Abr** | **Mai** | **Jun** | | Implementação do ambiente de testes | x | x |  |  |  |  | | Realização do experimento |  |  | x | x |  |  | | Análise dos dados coletados |  |  |  |  | x | x |   Quadro 3. Cronograma de execução para o segundo semestre do segundo ano (2020/2)   |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | **Atividade** | **Jul** | **Ago** | **Set** | **Out** | **Nov** | **Dez** | | Escrita da dissertação | x | x | x | x | x | x | |

**Análise de riscos**

Não existem riscos que impeçam o desenvolvimento deste trabalho

**REFERÊNCIAS**

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

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

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

ESPRESSIF. **ESP8266EX Datasheet**Espressif, , 2020. . Disponível em: <https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex\_datasheet\_en.pdf>. Acesso em: 5 dez. 2020

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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.

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

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

1. Após uma revisão de arquitetura o ESP8266 passou a se chamar ESP8266EX. No entanto, ambos os nomes se referem ao mesmo chip e possuem os mesmos recursos, sendo mais comum encontrar o nome ESP8266 na literatura [↑](#footnote-ref-1)
2. https://github.com/esp8266/Arduino [↑](#footnote-ref-2)
3. No RoPE, a frequência foi reduzida para 8 MHz a fim de economizar energia [↑](#footnote-ref-3)
4. O módulo fabricado pela AIThinker vem com 1 MB, módulos genéricos costumam variar entre 512 kB e 4 MB [↑](#footnote-ref-4)
5. *Sniffer* é um software ou hardware que permite ao usuário “farejar” ou monitorar o tráfego da rede [↑](#footnote-ref-5)