# 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: 31

## RESUMO

O pensamento computacional 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. Por esse motivo, pesquisadores da área têm feito um grande esforço em promover o desenvolvimento do pensamento computacional o mais cedo possível, já nos primeiros anos da educação infantil. No entanto, desenvolver o pensamento computacional em crianças requer a utilização de instrumentos que levem em consideração as particularidades desse público, entre elas, a realização de atividades lúdicas, o trabalho em conjunto, o engajamento 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

# 1 PROJETO

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), 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) 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), "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) 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

## 1.1 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).

### 1.1.1 Solução Proposta

Falar sobre a nossa solução. Citar o uso do RoPE e dos tapetes e o módulo WiFi

### 1.1.2 Delimitação de Escopo

Delimitar o escopo a crianças de 4 a 7 anos apenas

### 1.1.3 Justificativa

## 1.2 OBJETIVOS

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

### 1.2.1 Objetivo Geral

Avaliar o grau de impacto do idioma de uma linguagem de programação na aprendizagem da lógica de programação

### 1.2.2 Objetivos Específicos

1. Avaliar trabalhos similares que relatem sobre a influência do idioma na aprendizagem de lógica;
2. Projetar o experimento;
3. Desenvolver um ambiente para teste que tenha a mesma IDE para ambos idiomas;
4. Realizar um experimento com turmas de graduação;

## 1.3 METODOLOGIA

Nas seções seguintes a metodologia a ser utilizada nesta pesquisa é classificada e uma síntese dos procedimentos metodológicos utilizados é apresentada.

### 1.3.1 Metodologia da Pesquisa

Neste trabalho será aplicado o método indutivo, o qual consiste em se estabelecer uma verdade universal ou uma referência geral com base em dados e fatos previamente conhecidos e comprovados. Nesta pesquisa parte-se do conhecimento prévio de que existe uma dificuldade no aprendizado de programação relacionada ao idioma no Brasil e busca-se verificar o grau de impacto do idioma nesse aprendizado. Para verificar este grau de impacto será usada uma técnica estatística denominada Propensity score matching, desta forma, a pesquisa utilizada a abordagem quantitativa.

Sob o ponto de vista da natureza da pesquisa, esta é uma pesquisa aplicada. A pesquisa aplicada objetiva gerar conhecimentos para aplicações práticas dirigidos à solução de problemas. Neste trabalho o conhecimento gerado será o grau de impacto que o idioma exerce no aprendizado de programação e poderá ser aplicado em trabalhos futuros para solucionar esse problema.

Ainda, sob o ponto de vista do objetivo da pesquisa, esta é uma pesquisa exploratória. As pesquisas exploratórias têm como objetivo proporcionar maior familiaridade com o problema, com vistas a torná-lo mais explícito ou a constituir hipóteses, com isso, buscam criar fundamento para pesquisas mais aprofundadas. Nesse sentido, esta pesquisa visa aplicar conhecimentos e técnicas existentes através de estudos de caso, onde serão confirmadas ou refutadas as hipóteses formuladas.

### 1.3.2 Procedimentos Metodológicos

Esta seção apresenta os procedimentos metodológicos adotados nesta pesquisa.

**Revisão bibliográfica:** Esta etapa tem como objetivo proporcionar a fundamentação teórica necessária ao desenvolvimento da pesquisa.

**Revisão sistemática da literatura:** Esta etapa tem como objetivo realizar uma revisão sistemática da literatura analisando os trabalhos similares que tratam do impacto do idioma no aprendizado de programação.

**Implementação do ambiente de testes:** Esta etapa tem como objetivo implementar o ambiente de testes que será usado no experimento.

**Realização do experimento:** Esta etapa tem como objetivo realizar o experimento com as turmas de graduação a fim de coletar os dados para análise.

**Análise dos dados coletados:** Esta etapa tem como objetivo aplicar a técnica de Propensity Score Matching nos dados coletados e analisá-los.

**Conclusão:** Esta etapa tem como objetivo analisar as contribuições da pesquisa e apresentar sugestões de trabalhos futuros relevantes.

## 1.4 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, se faz 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 que integra um chip principal aos seus periféricos como, por exemplo: cristal, memória flash, regulador de tensão e outros componentes. Tendo esclarecido isso, o módulo escolhido para o desenvolvimento do Smart RoPE foi o ESP-01, o qual integra o chip ESP8266[1] da Espressif (MICROSHIP, 2020a) e é amplamente utilizado em projetos IoT em conjunto

[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

com o Arduino (baseadas em microcontroladores da família AVR da Atmel). A Figura 1 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.

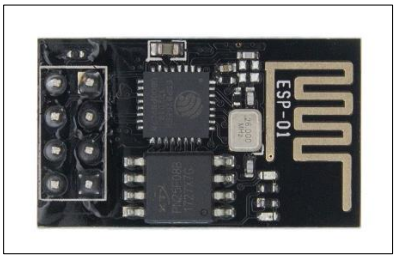


Figura 1. Módulo ESP-01

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]. Um exemplo de código para atualização OTA pode ser visto na Figura 2. Outra facilidade disponibilizada pela atualização OTA é a gravação simultânea de firmware em múltiplas unidades, apropriado para o processo de manufatura em grande escala.

[2] https://github.com/esp8266/Arduino

```
ESP8266WiFiMulti WiFiMulti;

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

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

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

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

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

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

Figura 2. Exemplo de código para atualização OTA

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

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

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

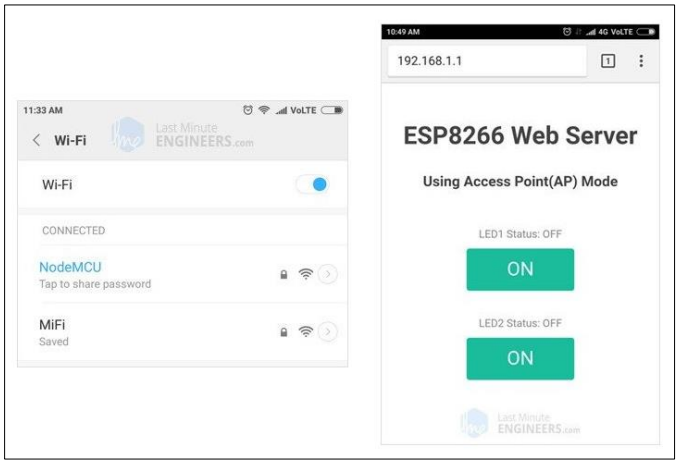


Figura 3. Rede SofAP e página web hospedada no ESP8266

A Figura 3 demonstra em exemplo de página de configuração hospedada no ESP8266 e uma rede denominada “NodeMCU” usada para a conexão. Já na Figura 4 é possível visualizar um trecho de código simplificado que pode ser usado para criar uma página semelhante no RoPE e gravar os parâmetros na *EEPROM*.

```
ESP8266WebServer server(80);

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

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

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

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

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

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

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

Figura 4. 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 1 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]<br>Máximo: 20 MHz | Padrão: 80 MHz<br>Máximo: 160 MHz |
| **RAM** | 2 kB | 36 kB |
| **Flash** | 32 kB | Padrão: 1 MB[4]<br>Máximo: 16 MB |

Tabela 1. Comparação de hardware entre o ATmega328p e o ESP8266

## 1.5 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.

[3] No RoPE, a frequência foi reduzida para 8 MHz a fim de economizar energia
[4] O módulo fabricado pela AIThinker vem com 1 MB, módulos genéricos costumam variar entre 512 kB e 4 MB

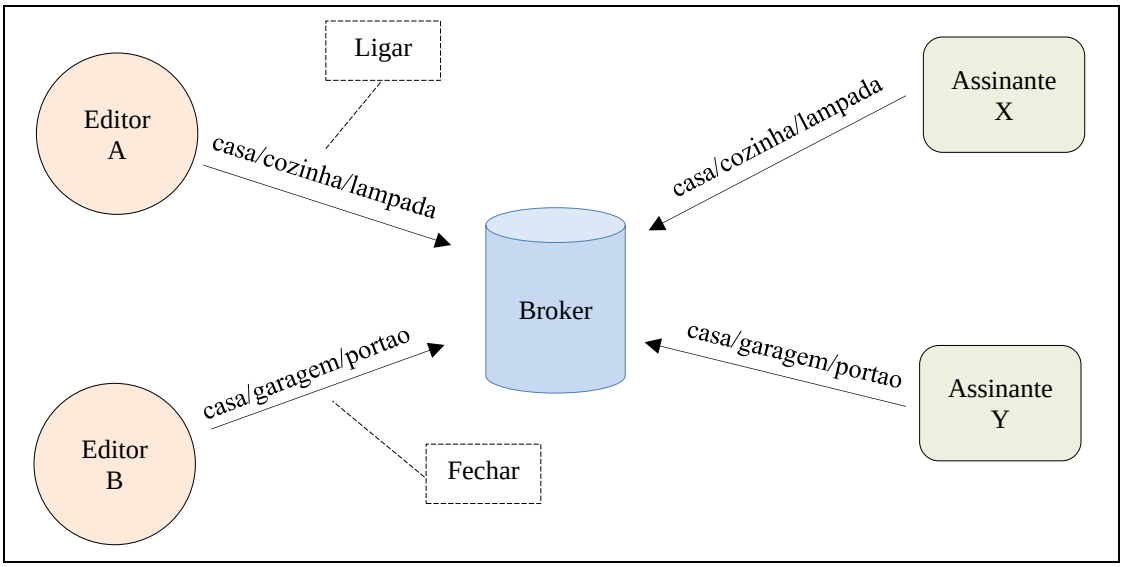


Figura 5. Arquitetura do MQTT

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

A complexidade é outro ponto relevante para o projeto, pois interfere no tempo necessário para o desenvolvimento, além de impactar na manutenção do *firmware* conforme o RoPE for evoluindo. Neste quesito, o MQTT também se mostra uma ótima opção, pois segundo (ELHADI et al., 2018) é um protocolo fácil de ser implementado e independe de 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* 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*.

## 1.6 SMART ROPE

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. Para alcançar esses objetivos foi feito 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 2 e servirão como norte para o desenvolvimento.

| **Requisitos funcionais do Smart RoPE** | |
|---|---|
| **RF1** | Os dispositivos externos devem conseguir mover o RoPE para frente e para trás |
| **RF2** | Os dispositivos externos devem conseguir girar o RoPE em sentido horário e anti-horário |
| **RF3** | Os dispositivos externos devem conseguir ligar e desligar o som do RoPE |
| **RF4** | Os dispositivos externos devem conseguir alterar a tonalidade do som do RoPE |
| **RF5** | Os dispositivos externos devem conseguir reproduzir notas musicais no RoPE |
| **RF6** | Os dispositivos externos devem conseguir ligar e desligar os LEDs do RoPE |
| **RF7** | Os dispositivos externos devem conseguir inserir instruções na memória do RoPE |
| **RF8** | Os dispositivos externos devem conseguir remover instruções da memória do RoPE |
| **RF9** | Os dispositivos externos devem conseguir alterar o tamanho da memória de instruções do RoPE |
| **RF10** | Os dispositivos externos devem conseguir limpar completamente a memória de instruções do RoPE |
| **RF11** | Os dispositivos externos devem conseguir executar um programa armazenado na memória do RoPE |
| **RF12** | Os dispositivos externos devem conseguir interromper um programa que esteja sendo executado |
| **RF13** | Os dispositivos externos devem conseguir instruir o RoPE a aguardar um intervalo de tempo |
| **RF14** | Os dispositivos externos devem conseguir simular o pressionamento dos botões do RoPE |
| **RF15** | Os dispositivos externos devem ser notificados quando o RoPE se mover |
| **RF16** | Os dispositivos externos devem ser notificados quando o RoPE girar |
| **RF17** | Os dispositivos externos devem ser notificados quando o nível de bateria do RoPE mudar |
| **RF18** | Os dispositivos externos devem ser notificados quando o som do RoPE for ligado ou desligado |
| **RF19** | Os dispositivos externos devem ser notificados quando a tonalidade dos sons do RoPE for alterada |
| **RF20** | Os dispositivos externos devem ser notificados quando uma nota musical for reproduzida no RoPE |
| **RF21** | Os dispositivos externos devem ser notificados quando um LED do RoPE for ligado ou desligado |
| **RF22** | Os dispositivos externos devem ser notificados quando uma instrução for inserida na memória do RoPE |
| **RF23** | Os dispositivos externos devem ser notificados quando uma instrução for removida da memória do RoPE |
| **RF24** | Os dispositivos externos devem ser notificados quando a memória do RoPE for apagada |
| **RF25** | Os dispositivos externos devem ser notificados quando o RoPE iniciar a execução de um programa |
| **RF26** | Os dispositivos externos devem ser notificados quando o RoPE finalizar a execução de um programa |
| **RF27** | Os dispositivos externos devem ser notificados quando um botão do RoPE for pressionado |
| **RF28** | Os dispositivos externos devem ser notificados do estado inicial do RoPE quando ele for ligado |
| **RF29** | Os dispositivos externos devem conseguir obter uma lista com o número de série de todas as unidades do RoPE conectadas ao *broker* |

Tabela 2. Requisitos funcionais do Smart Rope

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 6, pois a PCB do

RoPE já expõe um conjunto de pinos para a conexão com módulos externos usando a interface serial (*UART*).

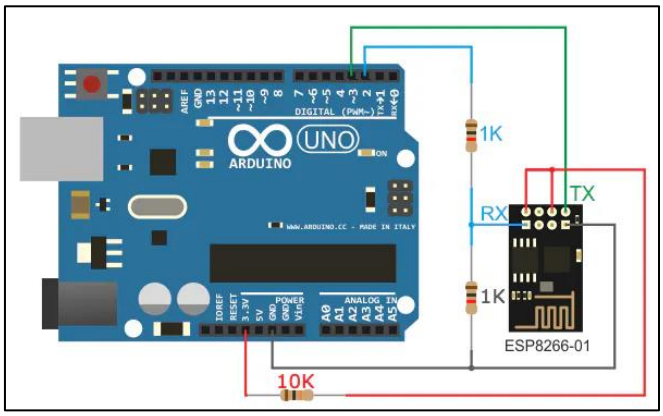


Figura 6. Conexão do ESP-01 com o RoPE

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 7. Uma vez conectado à rede Wi-Fi o ESP-01 já estará pronto para transmitir e receber dados através da Internet.

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

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

Figura 7. Trecho de código para configuração da rede Wi-Fi

O Smart Rope funcionará usando o protocolo MQTT, abordado no capítulo 1.5. 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 3 temas, os quais estão relacionados na Tabela 3 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 |
| 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 |

| Tema | Descrição |
|---|---|
| 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 3. Temas do Smart RoPE

Para o recebimento de comandos remotos a comunicação ocorre em duas etapas. Na primeira etapa 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 para o dispositivo, que pode armazená-lo para uso futuro. Na segunda etapa, o dispositivo publica uma mensagem contendo o comando que deve ser executado 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 8 apresenta um diagrama de sequência exemplificando esse processo.

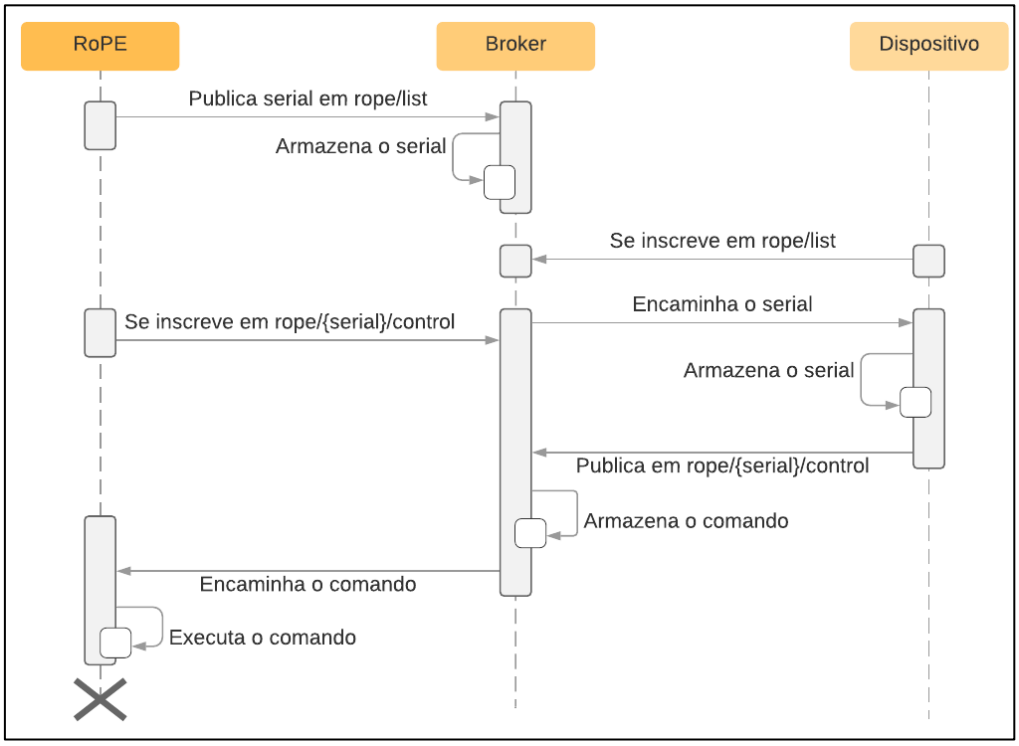


Figura 8. Envio de comandos para o RoPE

Para o monitoramento e coleta de dados do RoPE, foi definido 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 mecanismo permite que um dispositivo monitore as ações que estão sendo executados no RoPE por intermédio de outro dispositivo.

Assim como no caso anterior, a notificação dos eventos também ocorre em duas etapas. Na primeira etapa 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. Na segunda etapa, 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 9 apresenta um diagrama de sequência exemplificando esse processo.

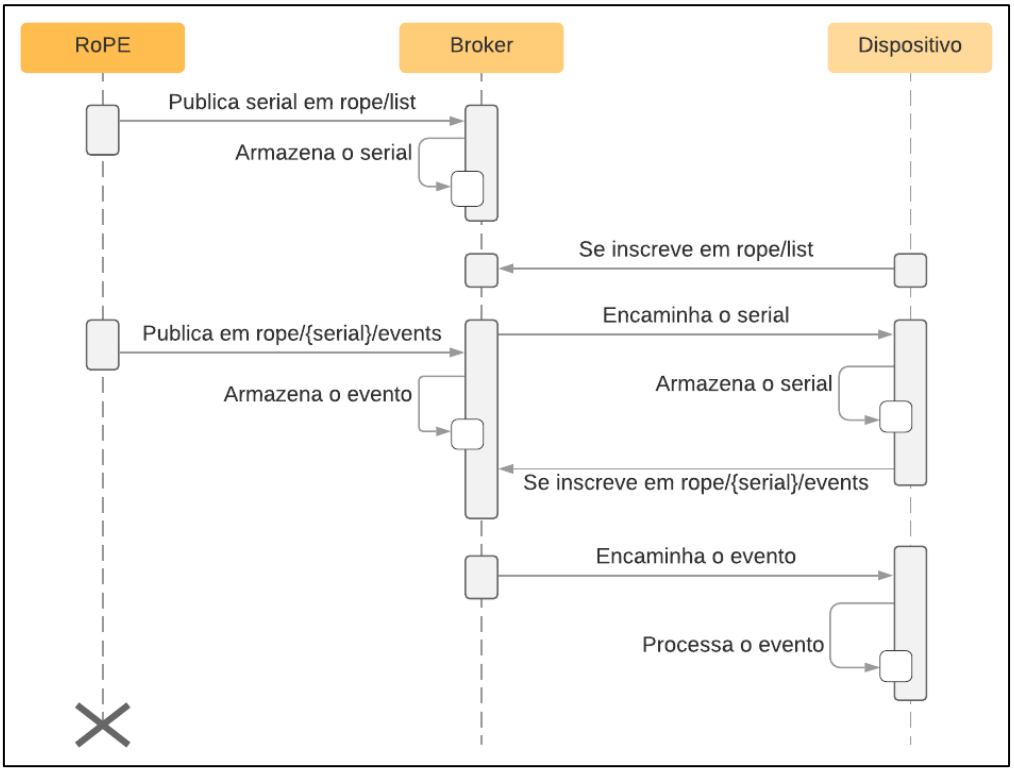


Figura 9. Envio de eventos para os dispositivos

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 do RoPE 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). Todas as mensagens construídas para o Smart RoPE estão relacionadas da tabela X até a tabela Y.

| **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 |
| **Exemplos** | | |
| {"command": "MOVE",<br>"parameters": {"direction": "FORWARD", "distance": 10.0}<br>} | | Move 10.0 milímetros para a frente |
| {"command": "MOVE",<br>"parameters": {"direction": "BACKWARDS", "distance": 30.0}<br>} | | Move 30.0 milímetros para trás |

Tabela 4. 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 |
| **Exemplos** | | |
| {"command": "TURN",<br>"parameters": {"direction": "CLOCKWISE", "degrees": -75.0}<br>} | | Gira 75 graus em sentido horário |
| {<br>"command": "TURN",<br>"parameters": {"direction": "COUNTERCLOCKWISE", "degrees": 90.0}<br>} | | Gira 90.0 graus em sentido anti-horário |

Tabela 5. Comando TURN

| Mensagem de controle | | |
|---|---|---|
| **Comando** | **TOGGLE_SOUND** | |
| **Descrição** | Alterna o status do som (**RF3**) | |
| **Parâmetros** | | |
| **State** | O novo estado do som | String: [ON \| OFF] |
| **Exemplos** | | |
| {<br>"command": "TOGGLE_SOUND",<br>"parameters": {"state": "ON"}<br>} | | Liga o som |
| {<br>"command": "TOGGLE_SOUND",<br>"parameters": {"state": "OFF"}<br>} | | Desliga o som |

Tabela 6. Comando TOGGLE_SOUND

| Mensagem de controle | | |
|---|---|---|
| **Comando** | **MOVE** | |
| **Descrição** | Aplica uma variação de semitons nas notas emitidas pelo *buzzer* (**RF4**) | |
| **Parâmetros** | | |
| **Direction** | A direção para qual o RoPE vai se mover | String: [FORWARD \| BACKWARDS] |
| **Distance** | A distância a ser movida (em milímetros) | Float |
| **Exemplos** | | |
| {"command": "MOVE",<br>"parameters": {"direction": "FORWARD", "distance": 10.0}<br>} | | Move 10.0 milímetros para a frente |
| {"command": "MOVE",<br>"parameters": {"direction": "BACKWARDS", "distance": 30.0}<br>} | | Move 30.0 milímetros para trás |

Tabela 7. Comando MOVE

| **SNDT** - Aplica uma variação de semitons nas notas emitidas pelo *buzzer* (**RF4**) | | |
|---|---|---|
| **VARIATION** | A quantidade de semitons que será variada | int |
| **Exemplos** | **SNDT 2** | Aumenta 2 semitons, mais agudo: A4 → B4 |
| | **SNDT -3** | Diminui 3 semitons, mais grave: A4 → Gb4 |

Tabela 8. Instrução SNDT

Tanto as mensagens de controle remoto quanto as mensagens de eventos serão enviadas e recebidas pelo módulo ESP-01 no formato JSON. Em ambos os casos o JSON deverá conter No entanto,

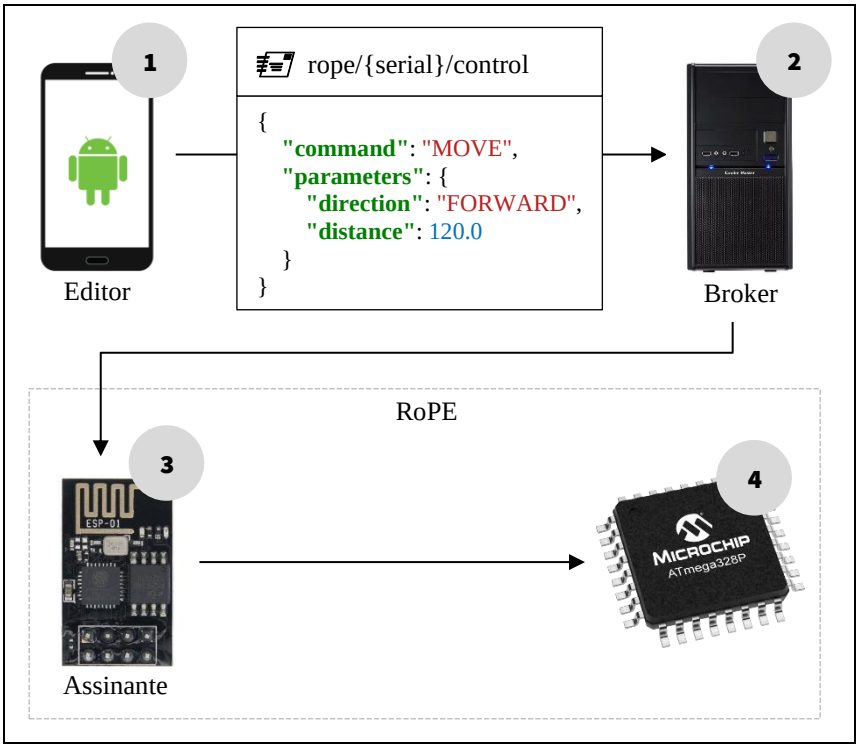

Figura 10. Conversão realizada ao receber um comando recebido

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. Para este trabalho, optou-se por usar o formato JSON, pois é amplamente usado na Web, é um formato leve, independe de linguagem de programação e permite estruturar as informações de forma legível para seres humanos (WEHNER; PIBERGER; GOHRINGER, 2014).

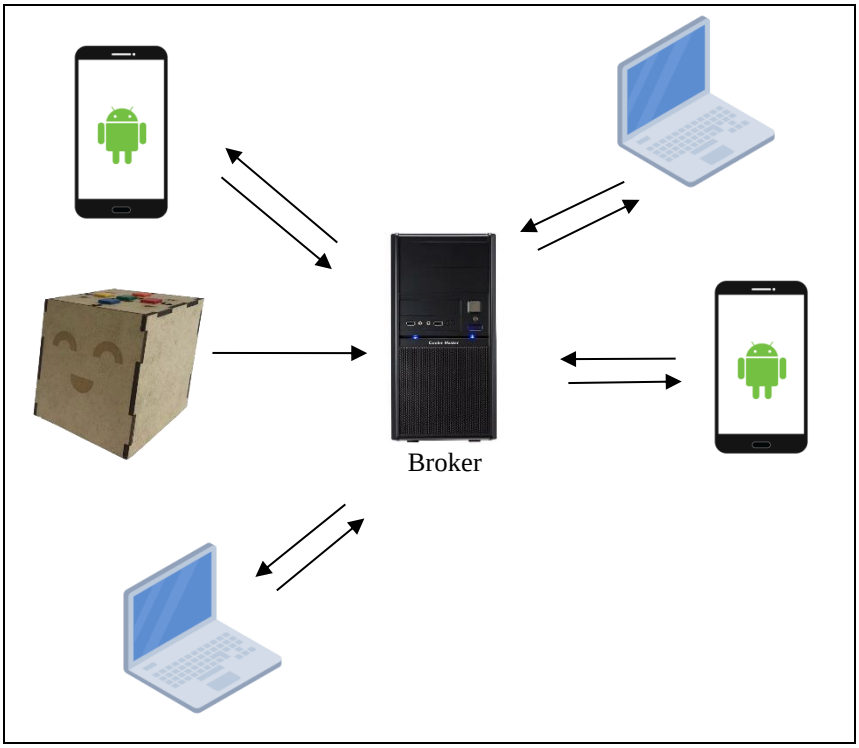


Figura 11. Arquitetura do Smart RoPE usando MQTT

### 1.6.1 Recebimento de comandos

Para cumprir com o segundo objetivo, que é

Conforme já foi mencionado no capítulo “1.4 MÓDULO WIFI ESP-01”, neste trabalho o módulo ESP-01 será acoplado à placa já existente do RoPE comunicando-se com ela através da interface serial.

Nulla facilisi. Etiam commodo rutrum turpis ut varius. Ut quis accumsan tortor, ac imperdiet risus. Nam vel pulvinar enim. Phasellus nec hendrerit nisl, vitae facilisis urna. Aliquam neque libero, dictum sodales ultricies nec, congue vel nulla. Cras justo dui, lacinia vitae purus eget, sagittis tempus purus. Nam ornare non nulla eu suscipit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum enim nisi, volutpat vitae placerat eget, efficitur ut ex. Curabitur sit amet eros leo. Pellentesque commodo porta erat, vel laoreet lacus pellentesque vitae. Vivamus placerat nibh vitae metus mollis ornare eu at mauris. Morbi nec auctor tortor. Aliquam eget neque molestie, laoreet nulla a, gravida arcu. Aliquam erat volutpat. Praesent nisl velit, egestas in elementum vel, posuere non leo. Aenean et accumsan elit. Duis varius auctor augue, in tincidunt est iaculis non. Orci varius natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Praesent et nulla mauris. Morbi scelerisque orci ac erat faucibus ultrices.

| **SNDP** - Reproduz uma nota no *buzzer* (**RF5**) | | |
|---|---|---|
| **NOTE** | A nota musical a ser reproduzida | string: [C \| D \| E \| F \| G \| A \| B] |
| **OCTAVE** | A oitava da nota a ser reproduzida | int |
| **DURATION** | A duração da nota em segundos | float |
| **PAUSE** | Uma pausa em segundos que será acrescentada após reproduzir a nota | float |
| **Exemplos** | **SNDP C, 5, 1.5, 0.5** | Reproduz a nota C5 (523.25 Hz) por 1 segundo e meio, logo após, dá uma pausa de meio segundo |
| | **SNDP E, 3, 2.0, 0.0** | Reproduz a nota E3 (164.81 Hz) por 2 segundos, sem dar nenhuma pausa após a reprodução |

Tabela 9. Instrução SNDP

| **LEDS** - Liga ou desliga um determinado LED (**RF6**) | | |
|---|---|---|
| **LED** | O identificador do LED que deverá ser alternado | string:<br>[FRONT \| LEFT \| RIGHT \| BACK \| CENTER] |

| **LEDS** - Liga ou desliga um determinado LED (**RF6**) | | |
|---|---|---|
| **STATE** | O novo estado do LED | string: [ON \| OFF] |
| **Exemplos** | **LEDS FRONT, ON** | Liga o LED frontal |
| | **LEDS BACK, OFF** | Desliga o LED traseiro |

Tabela 10. Instrução LEDS

| **PRGI** - Insere uma instrução na memória do RoPE (**RF7**) | | |
|---|---|---|
| **INSTRUCTION** | A instrução a ser inserida na memória | string:<br><br>[MOVE_FORWARD \| MOVE_BACKWARDS \| TURN_LEFT \| TURN_RIGHT] |
| **INDEX** | A posição da memória onde a instrução será inserida. Se nenhum valor for informado insere no final da memória | int |
| **Exemplos** | **PRGI TURN_LEFT** | Programa o RoPE para girar à esquerda. A instrução será inserida no final do programa |
| | **PRGI MOVE_FORWARD, 2** | Programa o RoPE para se mover para frente. A instrução será inserida na terceira posição da memória |

Tabela 11. Instrução PRGI

| **PRGR** - Remove uma instrução da memória do RoPE (**RF8**) | | |
|---|---|---|
| **INDEX** | A posição da memória da qual a instrução será removida. Se nenhum valor for informado remove a instrução no final da memória | int |
| **Exemplos** | **PRGR** | Remove a instrução no final da memória |
| | **PRGR 7** | Remove a instrução localizada na oitava posição da memória |

Tabela 12. Instrução PRGR

| **STPS** - Altera o tamanho da memória de instruções (**RF9**) | | |
|---|---|---|
| **SIZE** | O novo tamanho da memória de instruções | int |
| **Exemplos** | **STPS 20** | Ajusta a memória para 20 instruções |
| | **STPS 45** | Ajusta a memória para 45 instruções (valor padrão) |

Tabela 13. Instrução STPS

| Instruções de programação sem parâmetros | |
|---|---|
| **CLRP** | Limpa a memória de programa do RoPE (**RF10**) |
| **EXEC** | Executa o programa armazenado na memória (**RF11**) |
| **STOP** | Interrompe o programa que está sendo executado (**RF12**) |

Tabela 14. Instruções CLRP, EXEC e STOP

| **WAIT** - Pausa o RoPE por um determinado intervalo (**RF13**) | | |
|---|---|---|
| **INTERVAL** | O intervalo de tempo (em segundos) que o RoPE deve aguardar | float |
| **Exemplos** | **WAIT 2.5** | Pausa o RoPE por 2 segundos e meio |
| | **WAIT 0.1** | Pausa o RoPE por 100 milissegundos |

Tabela 15. Instrução WAIT

| **KEYP** - Simula o pressionamento de um botão pelo usuário (**RF14**) | | |
|---|---|---|
| **KEY** | O botão que será pressionado | string:<br><br>[MOVE_FORWARD \| MOVE_BACKWARDS \| TURN_LEFT \| TURN_RIGHT] |
| **Exemplos** | **KEYP TURN_RIGHT** | Pressiona o botão de girar à direita |
| | **KEYP EXECUTE** | Pressiona o botão de executar |

Tabela 16. Instrução KEYP

Nullam mollis, tortor sit amet elementum vehicula, tellus neque hendrerit erat, in luctus risus lacus vitae justo. Suspendisse vestibulum at quam sed finibus. Donec vitae auctor eros. Quisque et mauris non quam tempus porta. Curabitur molestie justo vitae porttitor aliquam. Vestibulum ac ipsum eu libero mollis vulputate eu at ex. Suspendisse fringilla risus enim, nec molestie ligula eleifend sit amet. Donec bibendum nisi non orci auctor, id malesuada lorem fringilla. Proin vel feugiat nulla. Nunc mattis libero ut felis porta tincidunt. Integer quis felis id felis scelerisque scelerisque a at velit. In hac habitasse platea dictumst. Vestibulum ultricies turpis vitae nibh accumsan suscipit. Mauris gravida placerat varius.

### 1.6.2 Comunicação externa

A primeira etapa para a execução do projeto é definir os requisitos funcionais que devem ser atendidos

A comunicação do RoPE com os dispositivos externos através do protocolo MQTT foi separada em duas categorias

A Figura 10 ilustra um exemplo de um aplicativo de celular que deseja controlar uma unidade específica do RoPE. Neste cenário, o ESP-01 atua como um *assinante* inscrito,

o aplicativo de celular atua como um *editor*, a fim de publicar mensagens de controle para o RoPE. Ao mesmo

Pri para , é possível observar o funcionamento do mecanismo de controle remoto do RoPE. Neste exemplo, um

Vivamus quis lorem ut magna suscipit venenatis. Suspendisse et turpis a ante faucibus ullamcorper convallis ac enim. Praesent congue hendrerit odio vel vestibulum. Cras eget varius dolor. Suspendisse congue turpis lectus, ut bibendum tortor cursus nec. Sed in aliquet nunc. Maecenas ullamcorper arcu vitae iaculis euismod. Donec placerat dapibus felis, et sagittis diam dignissim ac. Aliquam pulvinar placerat dui convallis aliquet.

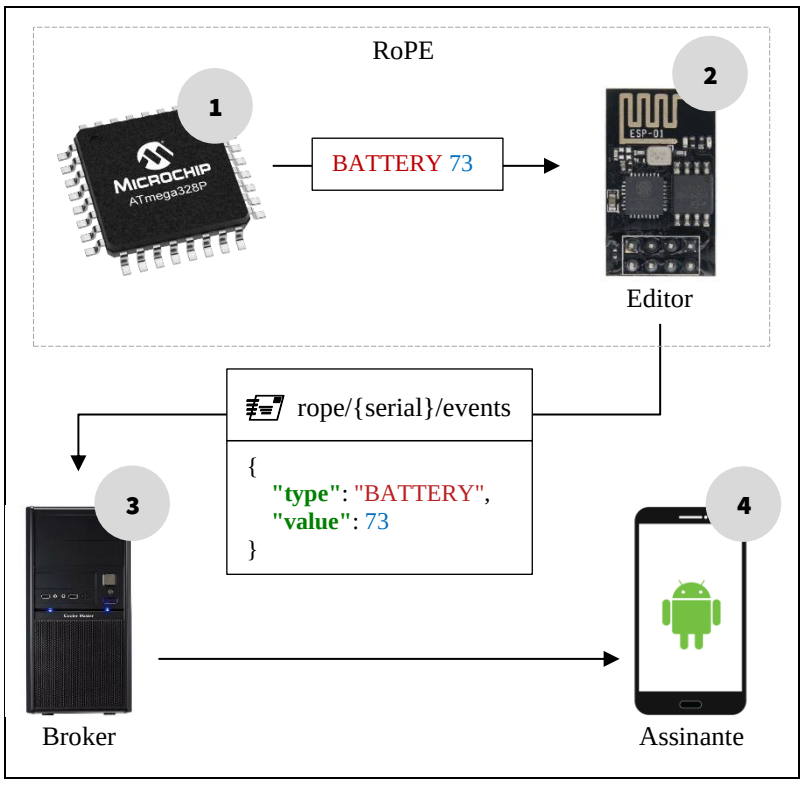


Figura 12. Fluxo do RoPE enviando dados

Nam et augue eget enim blandit ornare. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia curae; Donec commodo, arcu id dapibus auctor, lorem est convallis quam, sed sodales urna est id tortor. Integer aliquet condimentum mauris vel ullamcorper.

| Mensagem de controle | Instrução serial | Eventos gerados |
|---|---|---|
| rope/{serial}/control<br>{<br>"action": "MOVE",<br>"parameters": {<br>"direction": "FORWARD",<br>"distance": 120.0<br>}<br>} | **MOVE 120.0** | rope/{serial}/events<br>rope/{serial}/events/move<br>{<br>"type": "MOVE",<br>"parameters": {<br>"direction": "FORWARD",<br>"distance": 120.0<br>}<br>} |

Phasellus et ligula hendrerit, condimentum metus sit amet, lacinia metus. Maecenas in augue non mauris porta faucibus. Nullam vel suscipit urna. Duis volutpat posuere ligula malesuada ultrices. Integer porta, risus vitae ultricies condimentum, mauris sapien molestie erat, et malesuada ligula lacus at metus. Duis et ante ut mi elementum iaculis.

### 1.6.3 Aplicativo de testes

Vestibulum eu consequat tellus, et eleifend sem. In mauris ligula, fermentum sit amet odio sed, finibus egestas felis. Etiam commodo id arcu vel pellentesque. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Cras imperdiet sapien ipsum, ac consequat orci fermentum a.

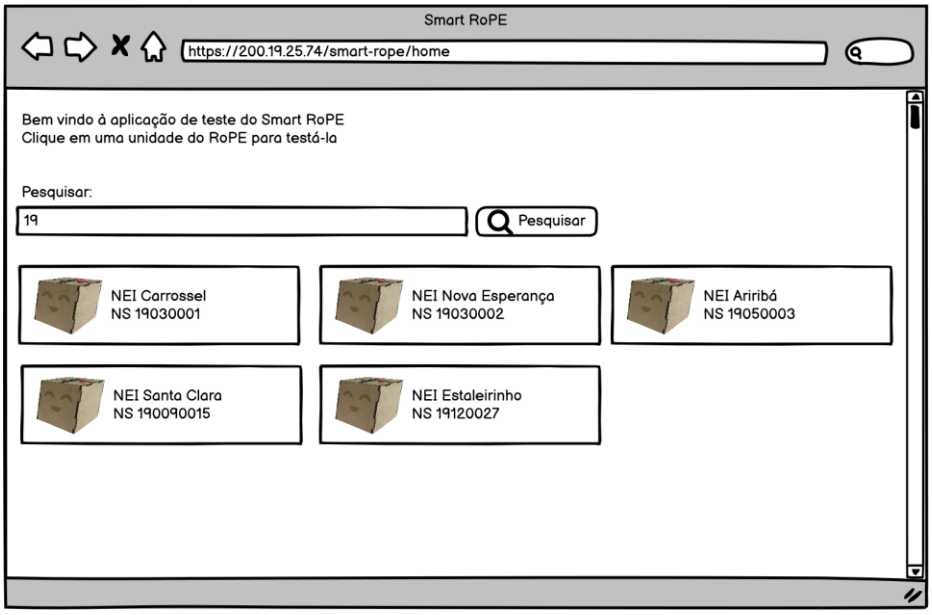


Figura 13. Protótipo da tela inicial do aplicativo Smart RoPE

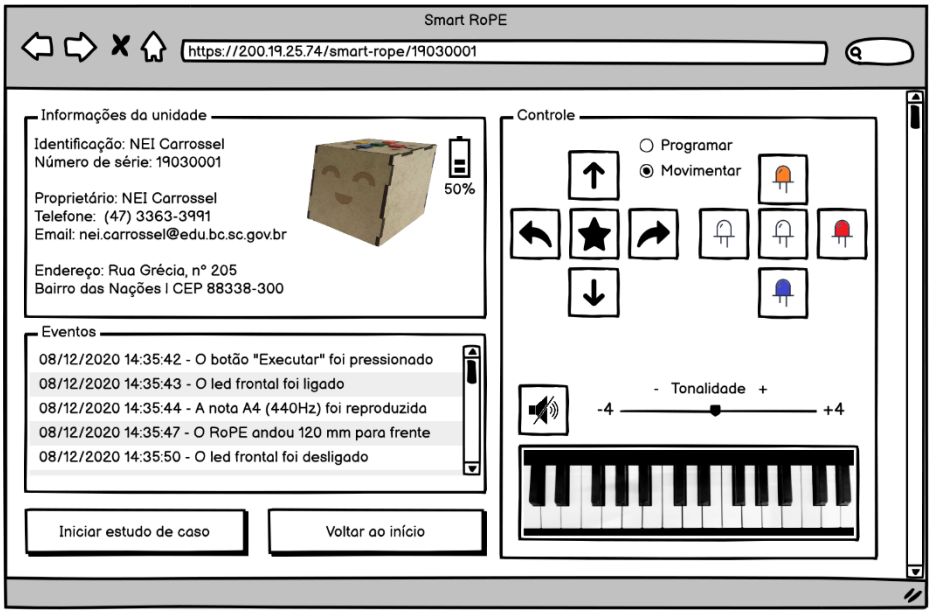

Figura 14. Protótipo da tela de gerenciamento de unidade do aplicativo Smart RoPE

Donec a dui risus. Sed felis mauris, viverra ac dignissim non, volutpat at massa. Donec tortor diam, posuere sit amet congue ut, venenatis rhoncus enim. Morbi quis volutpat mauris, et porta est. Curabitur dignissim ante urna, ac tempus ex sodales quis. Proin ullamcorper a augue at sagittis.

## 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

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.

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

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. **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

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

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