

Guilherme Vinícius Barbosa Pereira Amorim

Protótipo de um nó IoT acionado pela Amazon Alexa e integrado ao serviço AWS IoT

Belo Horizonte

2022

Guilherme Vinícius Barbosa Pereira Amorim

Protótipo de um nó IoT acionado pela Amazon Alexa e integrado ao serviço AWS IoT

Monografia apresentada durante o Seminário dos Trabalhos de Conclusão do Curso de Graduação em Engenharia Elétrica da UFMG, como parte dos requisitos necessários à obtenção do título de Engenheiro Eletricista.

Universidade Federal de Minas Gerais – UFMG

Escola de Engenharia

Curso de Graduação em Engenharia Elétrica

Orientador: Prof. Ricardo de Oliveira Duarte

Belo Horizonte

2022

Agradecimentos

Listas de ilustrações

Figura 1 – Redes de presença da AWS.	18
Figura 2 – Diagrama do serviço AWS IAM.	20
Figura 3 – Modelo Compartilhado de Responsabilidade de Amazon.	20
Figura 4 – Integração de dispositivos IoT com serviços AWS por meio do AWS IoT. .	22
Figura 5 – Exemplos de serviços fornecidos pelo AWS IoT.	23
Figura 6 – Exemplos de uso do AWS Lambda com a Alexa gerando um evento acionador.	25
Figura 7 – Exemplo de uma rede MQTT com quatro clientes.	27
Figura 8 – Atributos de uma mensagem do tipo <i>Publish</i> para o protocolo MQTT. .	27
Figura 9 – Atributos de uma mensagem do tipo <i>Subscribe</i> para o protocolo MQTT. .	28
Figura 10 – Atributos de uma mensagem do tipo <i>Suback</i> para o protocolo MQTT. .	29
Figura 11 – Atributos de uma mensagem do tipo <i>Unsubscribe</i> para o protocolo MQTT.	29
Figura 12 – Atributos de uma mensagem do tipo <i>Unsuback</i> para o protocolo MQTT. .	30
Figura 13 – Circuito eletrônico completo da maquete do projeto "PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ".	31
Figura 14 – Imagem frontal da maquete do projeto "PROTÓTIPO DE AUTOMA- ÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ". .	32
Figura 15 – Captura de tela do vídeo 5 da sequência de vídeos do projeto " <i>Creating an Alexa voice controlled IoT using a Raspberry Pi</i> ".	32
Figura 16 – Diagrama em alto nível do protótipo acionado pela Alexa e integrado ao serviço AWS IoT.	33
Figura 17 – Kit de desenvolvimento escolhido para ser protótipo do projeto (<i>B- L475E-IOT01A2</i>).	35
Figura 18 – Diagrama de estados da função <i>B-L475E-IOT01A2-Handler</i>	40

Lista de tabelas

Tabela 1 – Parâmetros mínimos recomendados pela AWS para o desenvolvimento de um dispositivo de IoT integrado à AVS.	34
Tabela 2 – kits de desenvolvimento recomendados pela Amazon para o desenvolvimento de aplicações IoT (A).	45
Tabela 3 – kits de desenvolvimento recomendados pela Amazon para o desenvolvimento de aplicações IoT (B).	46

Lista de Blocos de Código

2.1	Exemplo de uma política dando permissões de uso do serviço AWS IoT para um dispositivo IoT.	21
3.1	Política dando permissões de conexão, publicação, inscrição, e recebimento de tópicos MQTT ao dispositivo <i>B-L475E-IOT01A2</i>	36
3.2	Formato da mensagem de alternância do estado do LED.	37
3.3	Trecho do arquivo JSON que descreve o <i>frontend</i> da habilidade <i>B-L475E-IOT01A2-Skill</i>	39

Lista de abreviaturas e siglas

ASK	Alexa Skills Kit
AWS	Amazon Web Services
CLI	AWS Command Line Interface
EC2	Amazon Elastic Compute Cloud
HTTPS	Hypertext Transfer Protocol Secure
IAM	AWS Identity and Access Management
IDE	Ambiente de desenvolvimento integrado
IoT	Internet das Coisas
LED	Diodo emissor de luz
MQTT	MQ Telemetry Transport
SDK	Kit de desenvolvimento de software
TI	Tecnologia da Informação
TLS	Transport Layer Security
USB	Universal Serial Bus

Sumário

	Lista de Blocos de Código	9
1	INTRODUÇÃO	15
1.1	Estrutura	15
2	REFERENCIAL TEÓRICO	17
2.1	IoT	17
2.2	Computação em Nuvem	18
2.3	Computação em nuvem com a AWS	18
2.4	AWS IAM	19
2.5	Segurança	19
2.6	AWS IoT	21
2.6.1	AWS IoT Core	23
2.7	AWS Lambda	24
2.8	MQTT	25
2.8.1	MQTT Publish Message	26
2.8.2	MQTT Subscribe Message	28
2.8.3	MQTT Suback Message	28
2.8.4	MQTT Unsubscribe Message	29
2.8.5	MQTT Unsuback Message	30
2.9	Amazon Alexa	30
2.10	Trabalhos correlatos	30
2.10.1	PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ	30
	<i>Creating an Alexa voice controlled IoT using a Raspberry Pi</i>	31
3	METODOLOGIA	33
3.1	Objetivos	33
3.2	Projeto de Hardware	34
3.3	Criação de um nó IoT na rede AWS	35
3.4	Estabelecendo conexão entre o protótipo e o AWS IoT	36
3.5	Criação de Habilidades com o Alexa Skills Kit	37
3.5.1	Frontend da habilidade <i>B-L475E-IOT01A2-Skill</i>	38
3.5.2	Backend da habilidade <i>B-L475E-IOT01A2-Skill</i>	38
4	RESULTADOS E DISCUSSÃO	41

5	CONCLUSÕES	43
6	APÊNDICE	45
6.1	Kits de Desenvolvimento	45
6.2	Criação de uma coisa no AWS IoT	45

1 Introdução

REFACTORING...

1.1 Estrutura

O trabalho será apresentado em cinco capítulos. O [Capítulo 1](#) apresenta a introdução, contexto histórico e a motivação do projeto. O [Capítulo 2](#) apresenta o referencial teórico e trabalhos correlatos. O [Capítulo 3](#) contém o desenvolvimento do trabalho, apresentando requisitos funcionais e não-funcionais, ferramentas utilizadas e o processo de desenvolvimento. O [Capítulo 4](#) faz a análise dos resultados. O [Capítulo 5](#), por fim, conclui o trabalho, listando dificuldades enfrentadas, vantagens, desvantagens e as sugestões para trabalhos futuros. O projeto conta também com o [Capítulo 6](#), que é um capítulo extra de apêndice.

2 Referencial teórico

2.1 IoT

Nos últimos anos, observou-se a ascensão de dispositivos inteligentes, dispositivos que se conectam à Internet. Esses dispositivos se comunicam entre si, possuem sensores e outras tecnologias. Em termos simples, define-se IoT como uma rede de itens - cada um com sensores integrados - que são conectados à Internet (IEEE, "Internet of Things", 2014).

Aprofundando, IoT também pode ser definido como uma rede que engloba um sistema de dispositivos de computação inter-relacionados, máquinas mecânicas e digitais, objetos, animais ou pessoa, e a capacidade de transferir dados através de uma rede sem a necessidade de interação entre humanos (Alexander S. Gillis, 2022). A presença de tecnologias de nuvem, *big data* e redes móveis, por exemplo, em redes de IoT permite o compartilhamento e a coleta de dados com o mínimo de intervenção humana (ORACLE, 2022). Pode-se listar as seguintes tecnologias como as protagonistas no crescimento do uso da tecnologia IoT:

- Acesso a tecnologia de sensores de baixo custo e baixa potência;
- Conectividade;
- Plataformas de computação em nuvem;
- Machine learning e análise avançada;
- Processamento de linguagem natural (Amazon Alexa, por exemplo).

Uma característica comum dos dispositivos IoT é a necessidade de componentes de interface para a interação com o mundo físico. Alguns exemplos de componentes de interface são:

- Teclado, botões e microfones;
- Alarmes, visores e alto-falantes;
- Sensores;
- Atuadores;

2.2 Computação em Nuvem

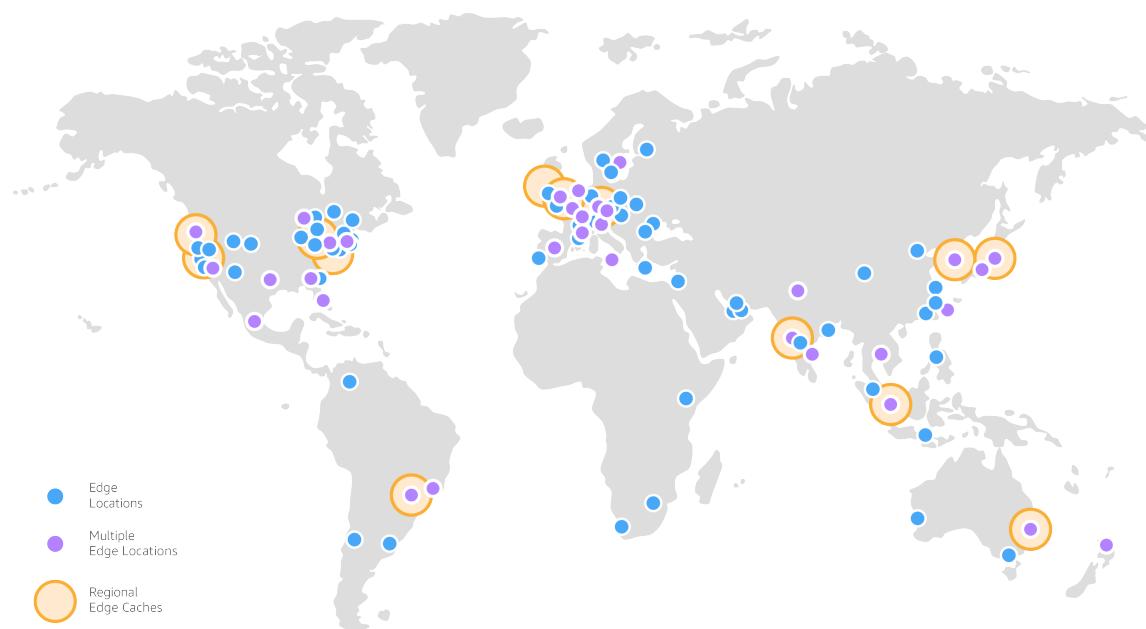
PENDING...

2.3 Computação em nuvem com a AWS

A Amazon Web Services, Inc. (AWS) é uma empresa subsidiária da Amazon responsável por fornecer a seus clientes plataformas para a computação em nuvem sob demanda via Internet. Empresas fazem uso desse serviço para a criação e execução de aplicações virtuais sem um custo inicial, uma vez que a AWS tem o *pay-as-you-go* como modelo de especificação. Ou seja, o cliente faz o pagamento conforme o uso (AWS, 2019). Em 2022, a AWS é capaz de oferecer a seus clientes mais de duzentos serviços em nuvem nas áreas de tecnologias de computação, banco de dados, *machine learning*, IoT, inteligência artificial etc.

Ademais, a AWS conta com mais de 410 pontos de presença em mais de 90 cidades e 48 países (Amazon AWS, 2022). Esse modelo de região e zona de disponibilidade da AWS foi reconhecido pelo Gartner, empresa de pesquisa e consultoria em TI, como o método recomendado para executar aplicativos corporativos que exigem alta disponibilidade (Amazon AWS, 2022). A [Figura 1](#) contém um mapa com as redes de presença da AWS, por região.

Figura 1 – Redes de presença da AWS.



Fonte: Amazon AWS (2022).

Para o desenvolvimento de aplicações na AWS, a Amazon oferece ao desenvolvedor

ferramentas e permite a escolha da linguagem de programação. Destaca-se as seguintes ferramentas:

- Console da web;
- Ferramenta de linha de comando;
- IDE;
- SDK;
- Infraestrutura como código.

2.4 AWS IAM

O AWS IAM é um serviço da Amazon que gerencia e controla o acesso aos recursos da AWS. Com o IAM, o usuário controla de maneira centralizada quem é autenticado (conectado) e autorizado (tem permissões) para usar recursos.

Ao criar uma conta AWS, uma identidade de login é iniciada. Essa identidade é chamada de usuário raiz da conta AWS e tem acesso completo a todos os serviços e recursos. O usuário raiz, contudo, não é recomendado em tarefas diárias. Recomenda-se que usuários tenham um privilégio mínimo e refinado para as suas tarefas.

Ademais, o IAM oferece um controle de acesso baseado em atributos, o que permite a criação de permissões baseadas em particularidades (como departamento, cidade e nome da equipe).

A [Figura 2](#) apresenta um diagrama detalhando o funcionamento do serviço AWS IAM.

2.5 Segurança

A AWS adota um modelo de responsabilidade compartilhada de segurança e conformidade entre a AWS e o cliente. Esse modelo tem como principais conceitos o que a Amazon chama de "segurança da nuvem" e "segurança na nuvem".

A "segurança da nuvem" define que a AWS é responsável por prover serviços seguros proteger a infraestrutura que executa todos os serviços oferecidos na Nuvem AWS. Já a "segurança na nuvem" é definida pelo serviço da AWS que o cliente estiver usando. Por exemplo, o serviço Amazon EC2 exige que o cliente execute todas as tarefas necessárias de configuração e gerenciamento da segurança. O cliente também é responsável por fatores como a sensibilidade dos dados, os requisitos da empresa e aplicáveis leis e regulações. A

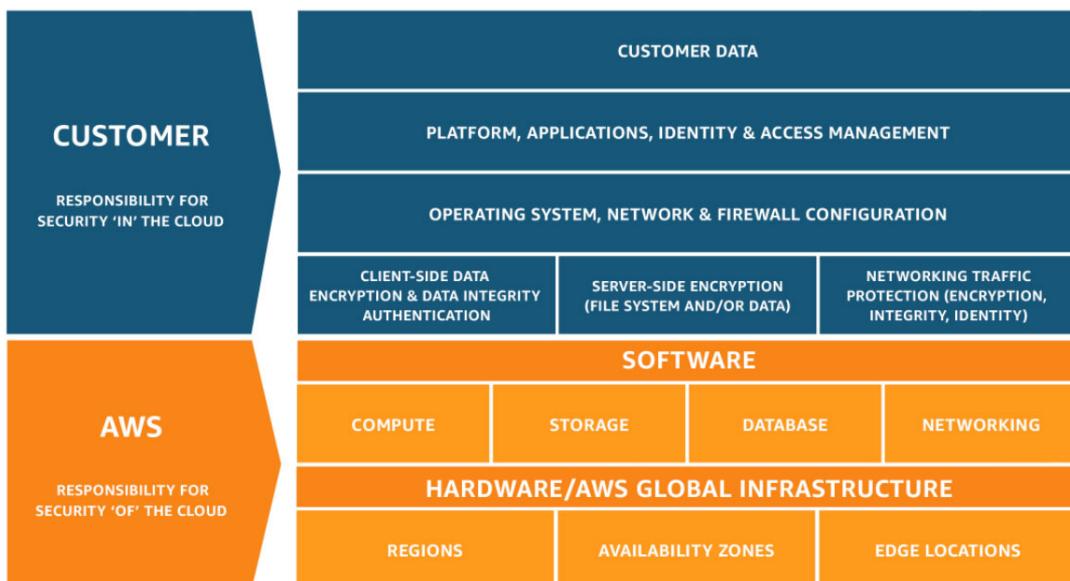
Figura 2 – Diagrama do serviço AWS IAM.



Fonte: Amazon AWS, com edições feitas pelo autor (2022).

Figura 3 mostra um diagrama oferecido pela AWS que descreve o modelo compartilhado de responsabilidade.

Figura 3 – Modelo Compartilhado de Responsabilidade de Amazon.



Fonte: Amazon AWS (2022).

Além disso, todo tráfego de dados acontece via TLS. TLS é um protocolo criptográfico projetado para fornecer segurança da comunicação em uma rede de computadores. O protocolo é amplamente usado em aplicativos como e-mail e mensagens instantâneas, mas seu uso na proteção de HTTPS continua sendo o mais visível publicamente (Wikipedia, 2022).

Para o serviço AWS IoT, que será mais bem detalhado na sessão [seção 2.6](#), cada

dispositivo e cliente precisa ter credenciais para a interação com a rede IoT. O cliente é responsável por gerenciar credenciais para cada dispositivo e políticas para o serviço AWS IoT. As credenciais têm o papel de criar identidade única para os dispositivos, enquanto as políticas administram as permissões de cada dispositivo ou grupo de dispositivos.

As credenciais geradas para cada dispositivo são uma forma de autenticação. Autenticação é um mecanismo de verificação de identidade de um cliente e/ou servidor. Um exemplo de certificado é o X.509. Os certificados X.509 são certificados digitais que usam o padrão de infraestrutura de chave pública X.509 para associar uma chave pública a uma identidade contida em um certificado (Amazon AWS, 2022). As cadeias de certificados X.509 são usadas para autenticação de servidor por clientes e autenticação de cliente pelo servidor.

As políticas, por sua vez, são mecanismos de autorização. Autorização é o processo de garantir permissões a uma identidade autenticada. De forma simples, as políticas determinam o que uma identidade autenticada pode fazer. Considere, por exemplo, um dispositivo conectado ao serviço AWS IoT com um certificado X.509. Com um documento de política, esse dispositivo pode ter acesso a todos os tópicos MQTT ou um número restrito de tópicos.

Para o gerenciamento de políticas, a AWS faz uso de documentos no formato JSON. O [Bloco de Código 2.1](#) mostra um exemplo de política que dá permissões para um dispositivo IoT acessar recursos do serviço AWS IoT de um usuário raiz com ID 332527922592 e localizado na região "sa-east-1" (São Paulo).

Bloco de Código 2.1 – Exemplo de uma política dando permissões de uso do serviço AWS IoT para um dispositivo IoT.

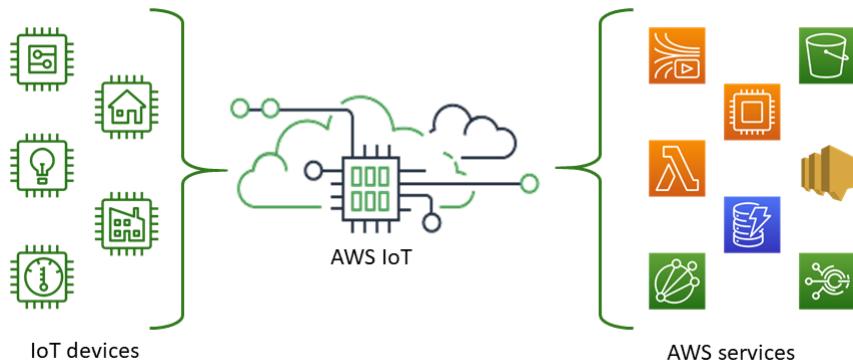
```
1 {
2     "Effect": "Allow",
3     "Action": "iot:Publish",
4     "Resource": "arn:aws:iot:sa-east-1:332527922592:/*"
5 },
6 {
7     "Effect": "Allow",
8     "Action": "iot:Receive",
9     "Resource": "arn:aws:iot:sa-east-1:332527922592:/*"
10 }
```

2.6 AWS IoT

O AWS IoT é um conjunto de serviços em nuvem oferecidos pela AWS que permitem a conexão de dispositivos IoT a outros dispositivos e a outros serviços oferecidos

pelo AWS. Em termos simples, o AWS IoT funciona como uma ponte entre dispositivos IoT e os serviços em nuvem que a AWS fornece, assim como pode ser visto na Figura 4.

Figura 4 – Integração de dispositivos IoT com serviços AWS por meio do AWS IoT.



Fonte: Amazon AWS (2022).

Os dispositivos IoT geralmente estão localizados próximos às interfaces do mundo real que monitoram e/ou controlam. Eles geralmente também incluem recursos de computação e armazenamento, como microcontroladores, CPUs e memórias. Alguns exemplos de dispositivos disponíveis no mercado são:

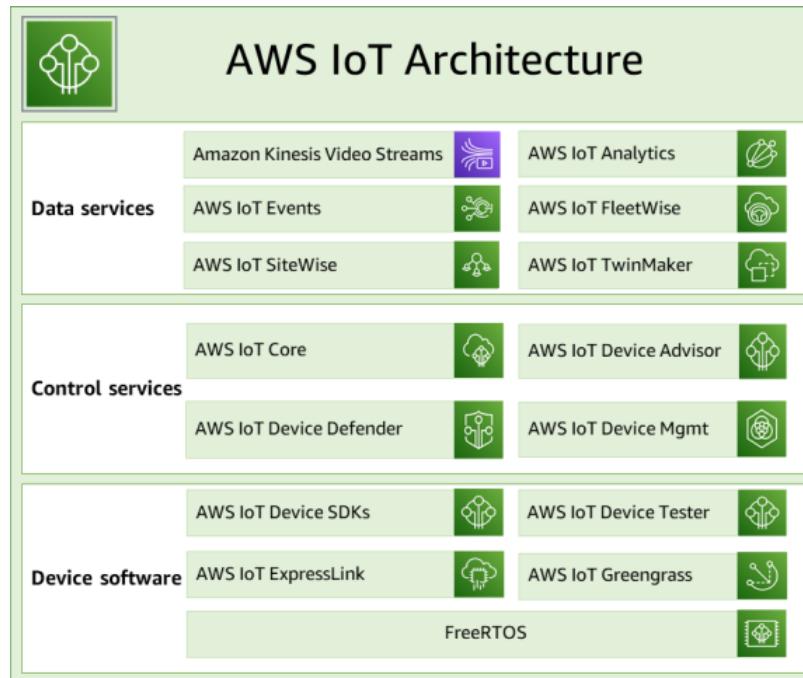
- LoRaWAN e dispositivos;
- Arduino;
- Raspberry PI;
- Dispositivos de IoT personalizados.

Para a completa integração desses dispositivos com serviços AWS e usuários, alguns componentes são essenciais. Aplicativos de celulares, por exemplo, são utilizados para que o usuário tenha acesso aos seus aparelhos e os configure conforme a sua preferência. Tem-se como outro exemplo os protocolos utilizados para a comunicação dos dispositivos com serviços em nuvem. A tecnologia AWS IoT possui suporte para os seguintes protocolos:

- MQTT;
- HTTPS;
- LoRaWAN.

O protocolo MQTT será mais bem detalhado na sessão [seção 2.8](#). Para o desenvolvimento de softwares a nível de dispositivo, o AWS IoT também fornece suporte para

Figura 5 – Exemplos de serviços fornecidos pelo AWS IoT.



Fonte: AWS IoT Core - Guia do desenvolvedor (2022).

um sistema operacional em tempo real para microcontroladores, o FreeRTOS. Alguns exemplos de outros serviços que o AWS IoT oferece suporte podem ser vistos na Figura 5.

O serviço IoT Core será mais bem detalhado na sessão [subseção 2.6.1](#).

2.6.1 AWS IoT Core

O AWS IoT Core é serviço de controle oferecido pela AWS que permite a conexão de bilhões de dispositivos de IoT e rotear trilhões de mensagens para serviços da AWS sem que o usuário tenha que gerenciar a infraestrutura. Esse serviço está disponível gratuitamente para clientes por doze meses a partir da data em que a conta da AWS é criada. Haverá taxas de uso do AWS IoT core após doze meses de uso ou quando a aplicação exceder os níveis de uso gratuito descritos abaixo:

- 2.250.000 minutos de conexão;
- 500.000 mensagens;
- 225.000 operações do Registry ou Device Shadow;
- 250.000 regras acionadas e 250.000 ações executadas.

Dessa forma, o AWS IoT Core é o serviço que permite a gerência de dispositivos inteligentes. A comunicação do serviço AWS IoT com os nós da rede IoT acontece via

protocolo MQTT. Os preços cobrados pela AWS após doze meses de uso ou excedendo os limites já citados pode ser visto abaixo:

- Preço da conectividade: 0,12 USD (por milhão de minutos de conexão);
- Até 1 bilhão de mensagens MQTT e HTTP: 1,50 USD (por milhão de mensagens);
- Próximos 4 bilhões de mensagens MQTT e HTTP: 1,20 USD (por milhão de mensagens);
- Mais de 5 bilhões de mensagens MQTT e HTTP: 1,05 USD (por milhão de mensagens).

2.7 AWS Lambda

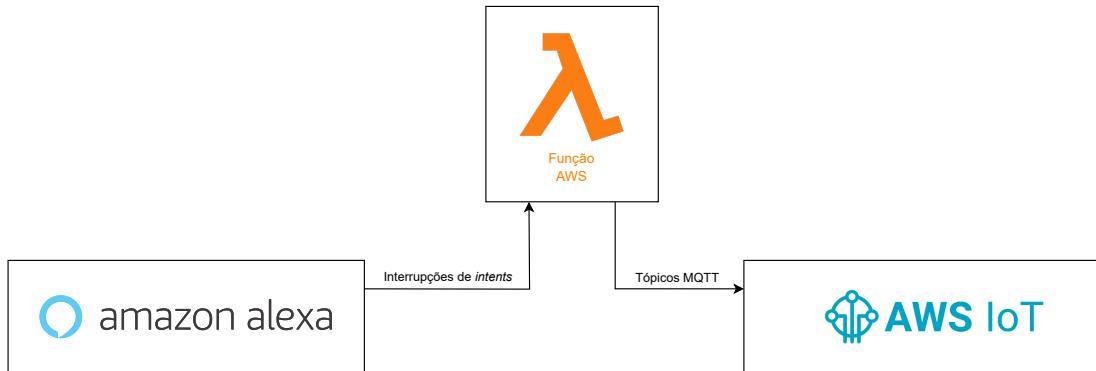
O AWS Lambda é um serviço que permite a execução de códigos sem que o usuário se preocupe com infraestrutura. A computação ocorre sem servidor e é orientado a eventos. Esses eventos podem ser mudanças de estado ou atualizações, como por exemplo o usuário acionando alguma tarefa através da Alexa. O AWS Lambda permite, também, que o usuário estenda a lógica de outros serviços da AWS. Sumariamente, o serviço AWS Lambda permite que o usuário execute serviços de *backend* sem provisionar ou gerenciar servidores.

O nível gratuito do AWS Lambda inclui um milhão de solicitações gratuitas por mês e 400.000 GB-segundos de tempo de computação por mês (AWS Lambda, 2022). Caso esses limites sejam excedidos, a especificação acontece sobre o número total de solicitações feitas por todas as funções. O preço final depende da quantidade de memória alocada por função.

Para começar a usar o AWS Lambda, o usuário deve primeiro carregar o código na nuvem AWS. Isso pode ser feito via AWS CLI, submissão de um arquivo no formato zip ou até mesmo criando o código diretamente no console do Lambda. O Lambda oferece suporte nativo às linguagens Java, Go, PowerShell, Node.js, C#, Python e Ruby. Após o carregamento do código, o usuário deve também configurar a memória e o tempo limites da função. Além disso, o usuário deve especificar o recurso que acionará a função. Alguns exemplos de recursos acionadores são uma transmissão do Amazon Kinesis e um evento gerado pela Alexa.

A [Figura 6](#) mostra um exemplo de uso do AWS Lambda. Nesse exemplo, a Alexa gera um evento acionador e a função transmite mensagens via tópicos MQTT para o AWS IoT.

Figura 6 – Exemplos de uso do AWS Lambda com a Alexa gerando um evento acionador.



Fonte: Produzido pelo autor (2022).

2.8 MQTT

MQTT é um protocolo de mensagens para dispositivos IoT. Ele foi desenvolvido para dispositivos com limitada disponibilidade de largura de banda (*network bandwidth*). Ele deve ser executado em um protocolo de transporte que forneça conexões bidirecionais ordenadas, sem perdas - normalmente, TCP/IP.

No protocolo MQTT, existem dois tipos de entidades: o *message broker* e os clientes. Um *message broker* é um servidor que tem o papel de receber diversas mensagens dos diversos clientes e, então, as rotear para os clientes de destino. Já um cliente pode ser qualquer dispositivo que se comunica via rede com o *message broker*, a partir de bibliotecas do protocolo MQTT.

As informações são organizadas em hierarquia de tópicos. Quando algum cliente possui um novo item de dados para distribuir, ele envia uma mensagem de controle com os dados para o *message broker*. O *broker*, então, distribui as informações para todos os clientes que se inscreveram nesse tópico. O cliente remetente (*publisher*) não precisa ter nenhuma informação sobre número de clientes que irão receber essa mensagem. Os clientes destinatários, por sua vez, também não precisam ser configurados com nenhuma informação dos destinatários.

Os sete tipos de mensagens existentes no protocolo MQTT são:

- *Connect*: Aguarda o estabelecimento de uma conexão com o *message broker* e cria um link entre os nós;
- *Disconnect*: Aguarda o fim da tarefa que está sendo executada pelo cliente e desconecta a sessão TCP/IP;
- *Publish*: Mensagem a ser distribuída para os clientes inscritos. Mais informações sobre esse tipo de mensagem podem ser encontrado na [Subseção 2.8.1](#).

- *Subscribe*: Para receber mensagens sobre tópicos de interesse, o cliente envia uma mensagem *Subscribe* para o *broker*. Mais informações sobre esse tipo de mensagem podem ser encontrado na [subseção 2.8.2](#);
- *Suback*: Para confirmar cada assinatura, o *broker* envia uma mensagem de confirmação *Suback* ao cliente. Mais informações sobre esse tipo de mensagem podem ser encontrado na [subseção 2.8.2](#);
- *Unsubscribe*: Esta mensagem exclui as assinaturas existentes de um cliente no *broker*. Mais informações sobre esse tipo de mensagem podem ser encontrado na [subseção 2.8.4](#);
- *Unsuback*: Para confirmar o cancelamento de assinatura, o *broker* envia uma mensagem de confirmação *Unsuback* ao cliente. Mais informações sobre esse tipo de mensagem podem ser encontrado na [subseção 2.8.5](#).

Depois que um cliente envia com êxito a mensagem *Subscribe* e recebe a mensagem *Suback*, ele obtém todas as mensagens publicadas de um tópico nas assinaturas contidas da mensagem *Subscribe*. Após receber o *Unsuback* do *broker*, o cliente pode assumir que as assinaturas na mensagem *Unsubscribe* foram excluídas.

Se o *message broker* receber uma mensagem em um tópico para o qual não há assinantes, a mensagem será descartada. Quando um cliente *publisher* se conecta pela primeira vez ao *broker*, ele pode configurar uma mensagem padrão para ser enviada aos assinantes se o *broker* detectar que o cliente *publisher* se desconectou inesperadamente.

Os clientes interagem apenas com um *broker*, mas um sistema pode conter vários *brokers* que trocam informações com base nos tópicos de seus assinantes atuais. Uma mensagem de controle MQTT deve ter entre dois e duzentos e cinquenta e seis bytes. O MQTT conta com o protocolo TCP para transmissão de dados. Uma variante, MQTT-SN, é usada com outros protocolos de transporte, como UDP ou Bluetooth.

A [Figura 7](#) mostra um exemplo de uma rede MQTT com quatro clientes.

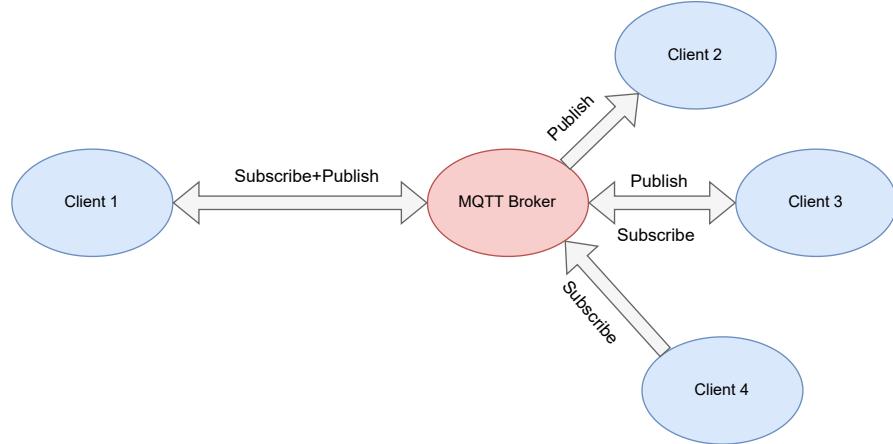
2.8.1 MQTT Publish Message

O formato de uma mensagem do tipo *Publish* está apresentado na [Figura 8](#).

Os atributos de uma mensagem do tipo *Publish* são caracterizados abaixo:

- *packetId*: O identificador de pacote identifica exclusivamente uma mensagem;
- *topicName*: O nome do tópico é uma *string* simples que é estruturada hierarquicamente com barras como delimitadores (“alex/builtin/device”, por exemplo);

Figura 7 – Exemplo de uma rede MQTT com quatro clientes.



Fonte: Produzido pelo autor (2022).

Figura 8 – Atributos de uma mensagem do tipo *Publish* para o protocolo MQTT.



Fonte: HiveMQ (2015).

- qos: Indica o nível de qualidade de serviço (QoS) da mensagem. Existem três níveis: 0, 1 e 2. O nível de serviço determina que tipo de garantia uma mensagem tem para chegar ao destinatário pretendido (cliente ou *broker*).
- retainFlag: Sinalizador indicando se a mensagem é salva pelo *broker* como o último valor válido conhecido para um tópico específico. Quando um novo cliente se inscreve em um tópico, ele recebe a última mensagem retida nesse tópico.
- payLoad: Conteúdo da mensagem. O MQTT é *data-agnostic*. Ou seja, é possível enviar qualquer tipo de informação que pode ser codificada em formato binário.
- dupFlag: Indica se a mensagem é uma duplicata e foi reenviada porque o destinatário pretendido (cliente ou *broker*) não enviou uma resposta de *acknowledge*.

2.8.2 MQTT Subscribe Message

O formato de uma mensagem do tipo *Subscribe* está apresentado na [Figura 9](#). Esta mensagem de assinatura é muito simples, contém um identificador de pacote exclusivo e uma lista de assinaturas.

Figura 9 – Atributos de uma mensagem do tipo *Subscribe* para o protocolo MQTT.



Fonte: HiveMQ (2015).

Os atributos de uma mensagem do tipo *Subscribe* são caracterizados abaixo:

- *packetId*: O identificador de pacote identifica exclusivamente uma mensagem;
- *List of Subscriptions*: Uma mensagem *Subscribe* pode conter várias assinaturas para um cliente. Cada assinatura é composta por um tópico (*topic_i*) e um nível de QoS (*qos_i*). O tópico pode conter *wildcards* que possibilitam a assinatura de um padrão de tópicos em vez de um tópico específico. Se houver assinaturas sobrepostas para um cliente, o *broker* entrega a mensagem que possui o nível de QoS mais alto para esse tópico.

2.8.3 MQTT Suback Message

O formato de uma mensagem do tipo *Suback* está apresentado na [Figura 10](#). Esta mensagem contém o identificador de pacote da mensagem *Subscribe* original e uma lista de códigos de retorno.

Os atributos de uma mensagem do tipo *Suback* são caracterizados abaixo:

- *packetId*: O identificador de pacote identifica exclusivamente uma mensagem;
- *Return Code*: O *broker* envia um código de retorno para cada par de tópico/QoS que recebe de mensagens *Subscribe*. O código de retorno reconhece cada tópico e mostra o nível de QoS concedido pelo *broker*. Se o *broker* recusar uma assinatura, a mensagem *Suback* conterá um código de retorno de falha para esse tópico específico.

Figura 10 – Atributos de uma mensagem do tipo *Suback* para o protocolo MQTT.

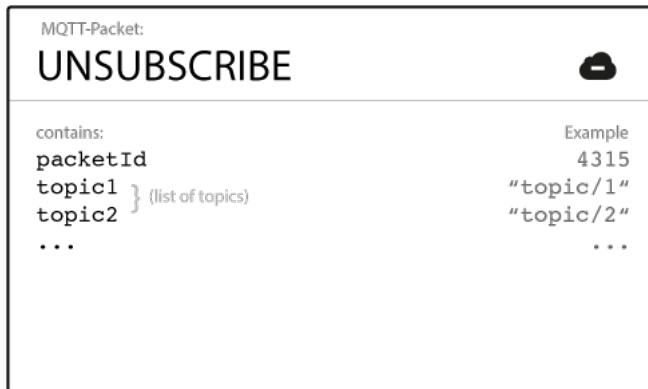


Fonte: HiveMQ (2015).

2.8.4 MQTT Unsubscribe Message

O formato de uma mensagem do tipo *Unsubscribe* está apresentado na [Figura 11](#). A mensagem *Unsubscribe* é semelhante à mensagem *Subscribe* e possui um identificador de pacote e uma lista de tópicos.

Figura 11 – Atributos de uma mensagem do tipo *Unsubscribe* para o protocolo MQTT.



Fonte: HiveMQ (2015).

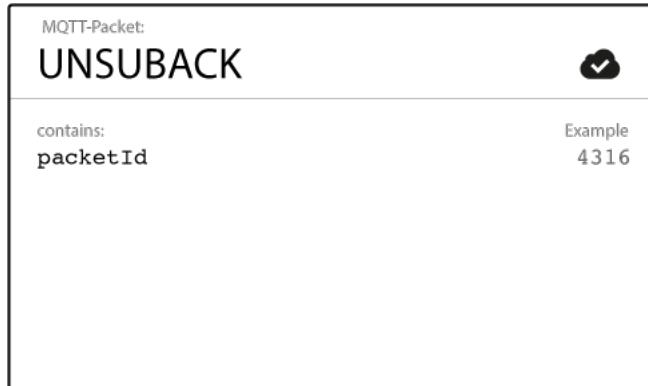
Os atributos de uma mensagem do tipo *Unsubscribe* são caracterizados abaixo:

- **packetId**: O identificador de pacote identifica exclusivamente uma mensagem;
- **List of Topic**: A lista de tópicos pode conter vários tópicos dos quais o cliente deseja cancelar a assinatura. Só é necessário enviar o tópico (sem QoS). O *broker* cancela a assinatura do tópico, independentemente do nível de QoS com o qual foi originalmente assinado.

2.8.5 MQTT Unsuback Message

O formato de uma mensagem do tipo *Unsuback* está apresentado na [Figura 12](#). Esta mensagem contém apenas o identificador de pacote da mensagem *Unsubscribe* original.

Figura 12 – Atributos de uma mensagem do tipo *Unsuback* para o protocolo MQTT.



Fonte: HiveMQ (2015).

2.9 Amazon Alexa

PENDING...

2.10 Trabalhos correlatos

Essa sessão busca detalhar dois trabalhos correlatos ao protótipo desse trabalho. Os dois trabalhos escolhidos são:

- PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ, de Leandro Dallarosa Neto;
- *Creating an Alexa voice controlled IoT using a Raspberry Pi*, de John Allwork;

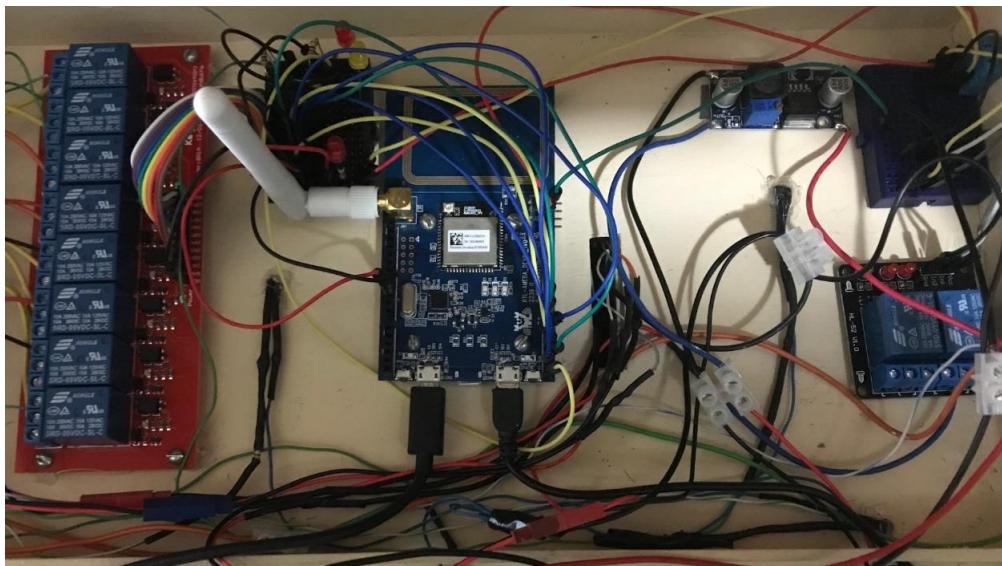
2.10.1 PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ

Esse projeto foi desenvolvido como um Trabalho de Conclusão de Curso do aluno Leandro Dallarosa Neto, pela Universidade Regional de Blumenau. Em resumo, o projeto teve como escopo a criação de um protótipo de automação residencial por comandos de voz utilizando a assistente pessoal Alexa. O protótipo desenvolvido utilizava o microcontrolador Arduino Ameba para controlar luzes, abrir uma porta magnética e enviar dados de temperatura para o servidor HTTP Thingspeak (2018). A computação do projeto foi feita

via serviços em nuvem da AWS: o AWS IoT e o AWS Lambda. A execução do protótipo acontece por comandos de voz recebidos em um aplicativo de dispositivos móveis.

Para testes com o protótipo, uma maquete e um circuito foram montado. Esse circuito conta com o microcontrolador Arduino Ameba, dois módulos relé para Arduino, um regulador de tensão, um eletroímã de 12V e o sensor de temperatura DHT11. A [Figura 13](#) mostra o circuito eletrônico completo da maquete, enquanto a [Figura 14](#) mostra a vista frontal da maquete.

Figura 13 – Circuito eletrônico completo da maquete do projeto "PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ".



Fonte: Monografia de Leandro Dallarosa Neto (2018).

2.10.2 *Creating an Alexa voice controlled IoT using a Raspberry Pi*

Esse projeto foi desenvolvido em 2019 pelo YouTuber John Allwork com o objetivo de demonstrar um Raspberry Pi sendo controlado por voz através da Alexa. O projeto conta com uma sequência de vídeos e documentação textual de como criar o seu IoT na rede AWS, carregar o código exemplo no AWS Lambda, criar habilidades na Alexa developer console e carregar o *firmware* em um Raspberry Pi 4.

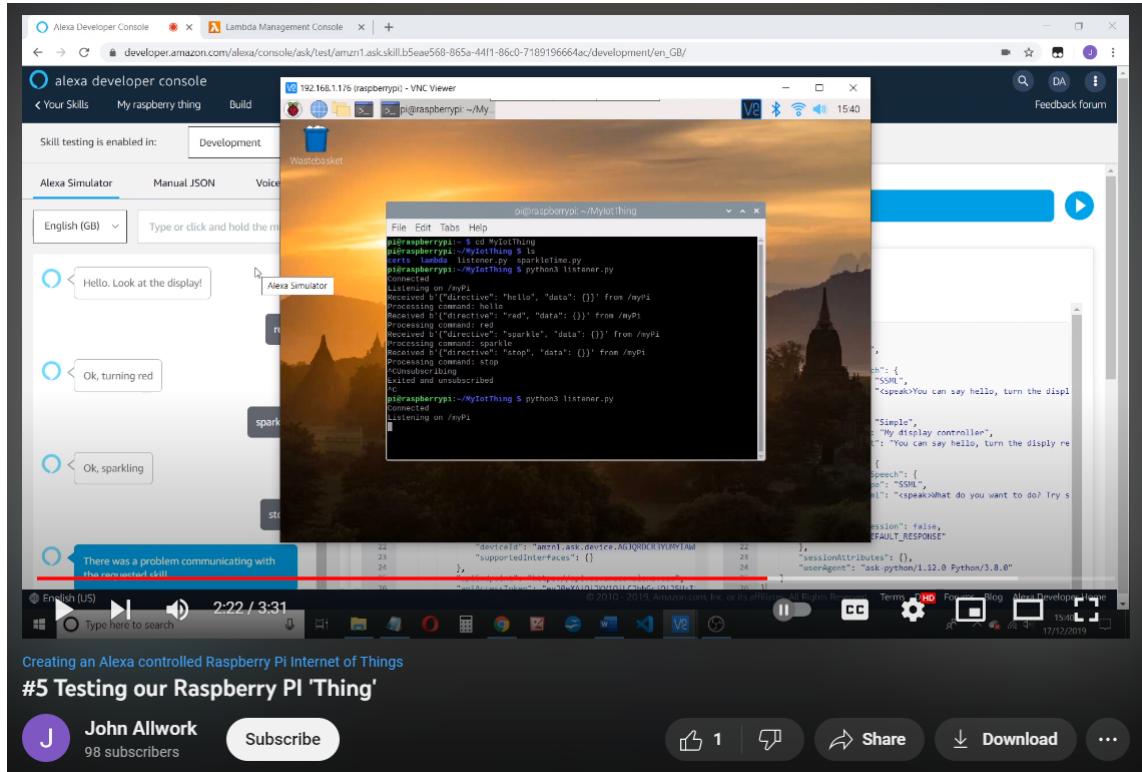
A [Figura 15](#) mostra uma captura de tela do vídeo 5 da sequência de vídeos do projeto do YouTuber John Allwork.

Figura 14 – Imagem frontal da maquete do projeto "PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ".



Fonte: Monografia de Leandro Dallarosa Neto (2018).

Figura 15 – Captura de tela do vídeo 5 da sequência de vídeos do projeto "Creating an Alexa voice controlled IoT using a Raspberry Pi".



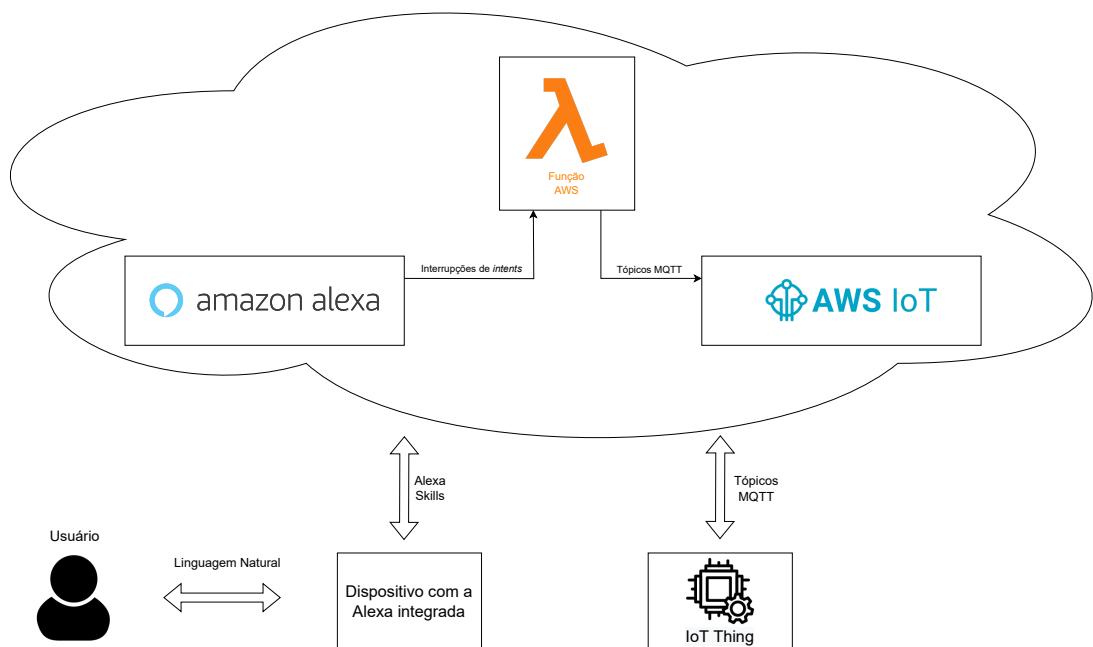
Fonte: Produzido pelo autor (2022).

3 Metodologia

3.1 Objetivos

Em resumo, o projeto propõe um protótipo de um nó IoT acionado pela Alexa. O protótipo comunicará com o serviço AWS IoT via tópicos MQTT - protocolo também utilizado na integração do AWS IoT com a Alexa. A [Figura 16](#) mostra um diagrama em alto nível do projeto.

Figura 16 – Diagrama em alto nível do protótipo acionado pela Alexa e integrado ao serviço AWS IoT.



Fonte: Produzido pelo autor (2022).

Para esse dispositivo, tem-se os seguintes requisitos funcionais:

- Supporte à rede Wifi IEEE 802.11 (2,4 GHz);
- Possibilidade de se comportar como um nó IoT;
- Supporte ao protocolo MQTT;
- Possibilidade de comunicação do usuário com o dispositivo em linguagem natural, a partir de microfones;
- Possibilidade de implementação de uma Alexa integrada;
- Possibilidade de acionamento da Alexa por voz ou toque;

g) Microfones capazes de entender o usuário em ambientes ruidosos;

A seguir, tem-se os requisitos não-funcionais:

- a) Ser um dispositivo qualificado pela AWS;
- b) Baixo custo de prototipagem;
- c) Versatilidade;

3.2 Projeto de Hardware

Uma vez escolhidos os requisitos funcionais e não-funcionais do projeto, inicia-se o projeto de hardware. Tomando o baixo custo de prototipagem como um importante requisito do projeto, optou-se pelo uso de sistemas embarcados no projeto. Computadores e notebooks pessoais, por exemplo, possuem um propósito geral e um alto valor agregado, enquanto um sistema embarcado realiza um conjunto de tarefas predefinidas e normalmente possuem um menor valor agregado (PrimeUP, 2022). Isso posto, inicia-se o estudo dos requisitos de um hardware a ser utilizado em aplicações IoT. Como um dos requisitos funcionais é a possibilidade de implementação de uma Alexa integrada, estuda-se os parâmetros mínimos recomendados pela AWS para um dispositivo com a Alexa integrada. Esses requisitos podem ser vistos na [Tabela 1](#).

Processador	ARM M7 or equivalent Arm M4 + AFE DSP
RAM	MB for ARM M7 500KB for M4 + AFE DSP
Target OS	FreeRTOS
Conectividade	MQTT/Wi-Fi
# de Microfones	2+
Alto-falante	Optimized for speech playback

Tabela 1 – Parâmetros mínimos recomendados pela AWS para o desenvolvimento de um dispositivo de IoT integrado à AVS.

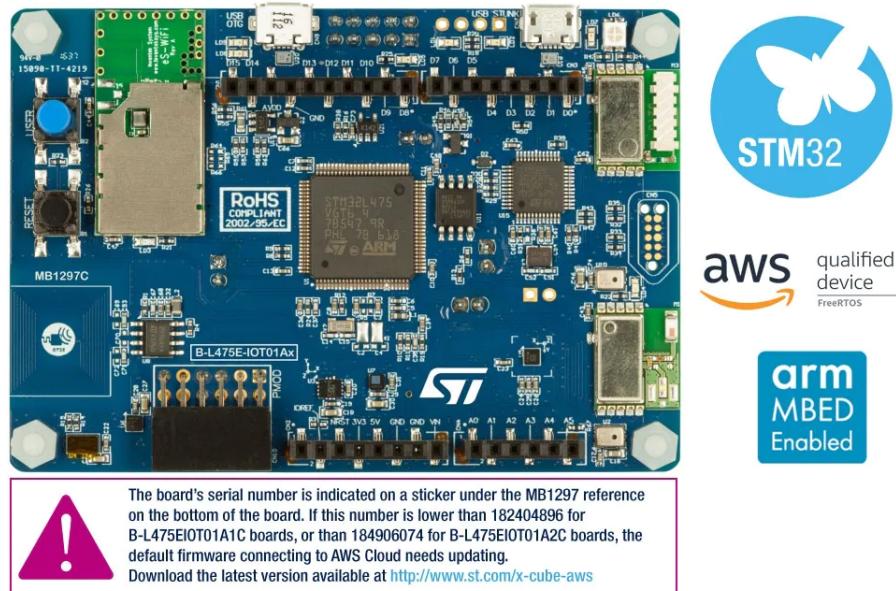
Para a prototipagem, alguns kits de desenvolvimento disponíveis no mercado foram estudados. As tabelas [Tabela 2](#) e [Tabela 3](#), presentes no [Capítulo 6](#), mostram informações coletadas de cinco diferentes kits.

Assim como pode ser visto na [Tabela 1](#), A AWS recomenda o Sistema Operacional FreeRTOS, descartando o *CORE-V MCU DevKit*. Ademais, definiu-se como requisito funcional a possibilidade de comunicação do usuário com o dispositivo através de linguagem natural, demandando no mínimo dois microfones e descartando o *Home Hub 100 Dev Kit for Amazon AVS*.

Por fim, restando somente os dispositivos *STEVAL-VOICE-UI*, *B-L475E-IOT01A2* e *SLN-ALEXA-IOT*, avaliou-se o preço nas lojas oficiais. Os preços, em Novembro de 2022, são respectivamente: \$248.75, \$51,94 e \$171,35. Essa discrepância de valores acontece devido à diferença nas especificações: o segundo dispositivo possui processador Arm® Cortex®-M4, enquanto o primeiro e o terceiro processadores Arm® Cortex®-M7 e Dual Arm® Cortex®-A7, respectivamente. Outrossim, os dispositivos *STEVAL-VOICE-UI* e *SLN-ALEXA-IOT* também contam com alto-falantes, já o *B-L475E-IOT01A2* não. Vale lembrar que a possibilidade de comunicação do dispositivo com o usuário via alto-falante não é um requisito funcional do projeto.

Dessa forma, conclui-se que o kit *B-L475E-IOT01A2* é o dispositivo, com menor custo, que mais se aproxima dos requisitos mínimos especificados pela AWS e atende os requisitos funcionais do projeto. O kit escolhido por ser visto na [Figura 17](#).

Figura 17 – Kit de desenvolvimento escolhido para ser protótipo do projeto (*B-L475E-IOT01A2*).



Fonte: STMicroelectronics (2022).

3.3 Criação de um nó IoT na rede AWS

Um nó IoT é representado na AWS como uma coisa no serviço AWS IoT. Cada coisa, ou grupo de coisas, pertence a alguma região da AWS. Assim, o dispositivo de protótipo foi autenticado na região *us-east-1* (N. Virginia) como *B-L475E-IOT01A2*. O processo completo de criação, autenticação e autorização de uma coisa no AWS IoT se encontra na [seção 6.2](#).

Para que o dispositivo tenha permissões de acesso a recursos da AWS, um certificado atrelado a uma política foi criado. Essa política da permissões de conexão, publicação,

inscrição, e recebimento de tópicos MQTT ao dispositivo e pode ser vista no Bloco de Código 3.1.

Bloco de Código 3.1 – Política dando permissões de conexão, publicação, inscrição, e recebimento de tópicos MQTT ao dispositivo *B-L475E-IOT01A2*.

```

1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Effect": "Allow",
6              "Action": "iot:Connect",
7              "Resource": "arn:aws:iot:us-east-1:332527922592:/*"
8          },
9          {
10             "Effect": "Allow",
11             "Action": "iot:Publish",
12             "Resource": "arn:aws:iot:us-east-1:332527922592:/*"
13         },
14         {
15             "Effect": "Allow",
16             "Action": "iot:Subscribe",
17             "Resource": "arn:aws:iot:us-east-1:332527922592:/*"
18         },
19         {
20             "Effect": "Allow",
21             "Action": "iot:Receive",
22             "Resource": "arn:aws:iot:us-east-1:332527922592:/*"
23         }
24     ]
25 }
```

Ao final do processo, o AWS IoT permite que o usuário faça *download* dos certificados de autenticação e autorização gerados. Esses certificados devem ser carregados no protótipo, dando permissões ao dispositivo de acesso aos serviços do AWS IoT e permitindo a sua identificação por parte da AWS.

3.4 Estabelecendo conexão entre o protótipo e o AWS IoT

Para estabelecer conexão entre o protótipo e o AWS IoT, através do protocolo MQTT, alguns projetos de exemplo disponibilizados pela STMicroelectronics e pelo Amazon FreeRTOS foram utilizados. Destaca-se os projetos *aws-demos* e o *B-L475E-IOT01-AWS*.

O projeto de exemplo *B-L475E-IOT01-AWS* é disponibilizado dentro do pacote expansão do software AWS IoT para STM32Cube, fornecido pela STMicroelectronics.

Esse pacote está disponível para *download* no link [AWS IoT software expansion for STM32Cube](#). Nesse exemplo, o dispositivo se conecta ao AWS IoT com as credenciais carregadas via USB. Quando o botão do usuário é pressionado, um comando de alternância de LED é enviado para o IoT *endpoint*, que retorna a mensagem para a placa e aciona a alternância do LED. Os dados de sensores da placa também são coletados e publicados na nuvem a cada 10 segundos.

A aplicação envia o comando de alternância do estado do LED para o tópico MQTT "\$aws/things/B-L475E-IOT01A2/shadow/update" e segue o formato JSON apresentado no [Bloco de Código 3.2](#).

Bloco de Código 3.2 – Formato da mensagem de alternância do estado do LED.

```

1 {
2     "state": {
3         "desired": {
4             "LED_value": "On"
5         }
6     }
7 }
```

O canal oficial da STMicroelectronics no YouTube disponibiliza um vídeo demonstrando o projeto sendo executado. O vídeo está disponível no link [Getting starting with STM32L4 Discovery kit IoT node](#).

Notas:

- O projeto de exemplo está disponível para o kit de desenvolvimento *B-L475E-IOT01* somente para as versões 1.2.1, 1.4.0 e 1.4.1;
- O projeto conta com um arquivo binário que pode ser diretamente carregado no kit;
- O projeto possui problemas de compilação quando executado no sistema operacional Windows. Para solucionar os erros de compilação, o usuário deve substituir alguns caminhos relativos no *script* de compilação por caminhos completos.

Para fins de prototipagem, o exemplo disponibilizado pela STMicroelectronics atendeu às necessidades do projeto e foi utilizado no processo de validação. Dessa forma, após a configuração do protótipo e autenticação no AWS IoT, a conexão via MQTT foi estabelecida.

3.5 Criação de Habilidades com o Alexa Skills Kit

Para a criação de uma habilidade na Amazon Alexa, fez-se uso do ASK e do Console do Desenvolvedor Alexa. A Amazon sugere que o desenvolvimento de uma habilidade

ocorra seguindo os seguintes passos:

1. Criação de um nome para a habilidade;
2. Criação de Intenções, amostras e Slots;
3. Compilação do modelo;
4. Definição de um *endpoint* para o gerenciamento de requisições feitas pelo usuário;
5. Produtização e monetização da habilidade.

O nome da habilidade foi definido como *B-L475E-IOT01A2-Skill*. Os itens [Item 2](#) e [Item 4](#) constituem o *frontend* e o *backend* da habilidade, respectivamente.

3.5.1 *Frontend* da habilidade *B-L475E-IOT01A2-Skill*

O ASK permite que o desenvolvimento do *frontend* aconteça via interface web e/ou arquivo JSON. O código que define o *frontend* da aplicação pode ser encontrada no seguinte repositório git: [guigultz/PENDING](#). Um trecho do código está disponibilizado no [Bloco de Código 3.3](#).

3.5.2 *Backend* da habilidade *B-L475E-IOT01A2-Skill*

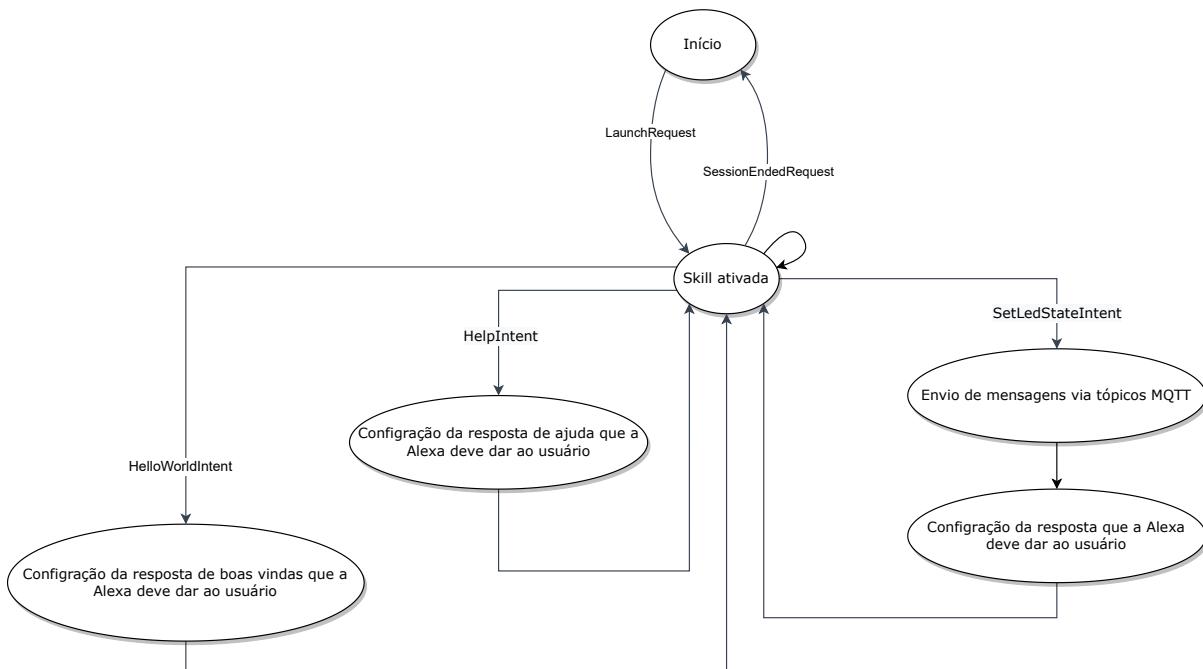
Para o desenvolvimento do *backend*, optou-se por estender as funcionalidades do ASK através de funções no AWS Lambda. Dessa forma, o desenvolvimento pode acontecer em Node.js, Python etc, assim como foi mencionado na [seção 2.7](#). Para esse projeto, escolheu-se Python e *ask-sdk* como linguagem de desenvolvimento e pacote SDK, respectivamente. *ask-sdk* é o pacote referência para o desenvolvimento com o ASK. A função hospedada no AWS Lambda foi nomeada *B-L475E-IOT01A2-Handler* e pode ser encontrada no seguinte repositório git: [guigultz/B-L475E-IOT01A2-Handler](#).

A função Lambda é orientada a eventos, assim como já foi explicado na [seção 2.7](#) e segue o diagrama de estados apresentado na [Figura 18](#). Em termos práticos, após o acionamento da intenção *SetLedStateIntent* através das frases "set led on", "set led off", "turn led on" e "set led off", função *B-L475E-IOT01A2-Handler* envia uma mensagem para o tópico MQTT "\$aws/things/B-L475E-IOT01A2/shadow/update" que segue o formato apresentado no [Bloco de Código 3.2](#).

Bloco de Código 3.3 – Trecho do arquivo JSON que descreve o *frontend* da habilidade *B-L475E-IOT01A2-Skill*.

```
1 {
2 // ...
3   "invocationName": "my node thing",
4   "intents": [
5     {
6       "name": "HelloWorldIntent",
7       "slots": [],
8       "samples": [
9         "good morning",
10        "hello"
11      ]
12    },
13    {
14      "name": "SetLedStateIntent",
15      "slots": [
16        {
17          "name": "ON_OFF_SLOT",
18          "type": "ON_OFF_SLOT"
19        }
20      ],
21      "samples": [
22        "set led {ON_OFF_SLOT}",
23        "turn led {ON_OFF_SLOT}",
24      ]
25    }
26  ],
27  "types": [
28    {
29      "name": "ON_OFF_SLOT",
30      "values": [
31        {
32          "name": {
33            "value": "OFF"
34          }
35        },
36        {
37          "name": {
38            "value": "ON"
39          }
40        }
41      ]
42    }
43  ],
44 // ...
45 }
```

Figura 18 – Diagrama de estados da função *B-L475E-IOT01A2-Handler*.



Fonte: Produzido pelo autor (2022).

4 Resultados e Discussão

PENDING...

5 Conclusões

PENDING...

6 Apêndice

6.1 Kits de Desenvolvimento

Número de produto	B-L475E-IOT01A2	CORE-V MCU DevKit	SLN-ALEXA-IOT
Descrição	STM32L4 Discovery kit Nô IoT, low-power wireless, BLE, NFC, SubGHz, Wi-Fi	European RISC-V chip for IoT development kit	EdgeReady MCU Based Solution for Alexa for IOT
Processador	Arm® Cortex®-M4	CV32E40P processor core	Arm® Cortex®-M7 Core
FreeRTOS	Sim	Não Disponível	Sim
Conectividade	Inventek ISM43362 Wi-Fi Module	Espressif AWS IoT ExpressLink Module for AWS IoT cloud interconnect	Bluetooth LE 4.2, 802.11 b/g/n Wi-Fi®
Microfones	2 digital omnidirectional microphones (MP34DT01)	Não Disponível	Digital MEMS microphones (x3)
Alto-falante	Não Disponível	Não Disponível	Não Disponível
Preço nas Lojas Oficiais	\$53,00	Preços Individuais	\$171,35

Tabela 2 – kits de desenvolvimento recomendados pela Amazon para o desenvolvimento de aplicações IoT (A).

6.2 Criação de uma coisa no AWS IoT

SEQUÊNCIA DE PRINTSCREENS DURANTE A CRIAÇÃO DE UMA COISA NO AWS IOT!

Número de produto	Home Hub 100 Dev Kit for Amazon AVS	STEVAL-VOICE-UI
Descrição	A hardware and software development kit with multi-core connectivity designed to support AVS for AWS IoT Core connected devices	Qualified hardware reference design enabling easy and cost effective addition of the Alexa Voice Service (AVS) Integration for AWS IoT core to your smart embedded devices
Processador	Arm Cortex-M4F CPU	Dual Arm® Cortex®-A7 and Cortex®-M4 Cores
FreeRTOS	Sim	Sim
Conectividade	Integrated Bluetooth 5, Qualcomm® Bluetooth Mesh connectivity, low power Wi-Fi 802.11n in 2.4GHz/5GHz bands, and 802.15.4, with support for ZigBee3.0 and Thread via OpenThread	WIFI subsystem including Murata 1DX module used in bypass mode and ISSI IS25LP016D 2Mbytes NOR flash hosting WIFI low level software
Microfones	Não Disponível	2 MP23DB01HP Microphone Mems
Alto-falante	1x	1x (8 ohm)
Preço nas Lojas Oficiais	£42.95	\$248.75

Tabela 3 – kits de desenvolvimento recomendados pela Amazon para o desenvolvimento de aplicações IoT (B).