

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.	19
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.	26
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. .	28
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 – Busca de habilidades da Alexa por categorias.	31
Figura 14 – Diagrama explicando o funcionamento de uma habilidade Alexa.	32
Figura 15 – Diagrama de integração do AVS ao serviço AWS IoT Core.	33
Figura 16 – Circuito eletrônico completo da maquete do projeto “PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ”.	34
Figura 17 – Imagem frontal da maquete do projeto “PROTÓTIPO DE AUTOMA- ÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ”. .	34
Figura 18 – Captura de tela do vídeo 5 da sequência de vídeos do projeto “ <i>Creating an Alexa voice controlled IoT using a Raspberry Pi</i> ”.	35
Figura 19 – Diagrama em alto nível do protótipo acionado pela Alexa e integrado ao serviço AWS IoT.	37
Figura 20 – Kit de desenvolvimento escolhido para ser protótipo do projeto (<i>B- L475E-IOT01A2</i>).	39
Figura 21 – Diagrama de estados da função <i>B-L475E-IOT01A2-Handler</i>	44
Figura 22 – Simulação de uma iteração do usuário com a Alexa através do <i>Amazon Alexa Console</i>	46
Figura 23 – Cliente de teste MQTT durante a execução da habilidade <i>B-L475E- IOT01A2-Skill</i>	46
Figura 24 – Mensagem de atualização os dados dos sensores do protótipo recebidos no Cliente de teste MQTT.	47

Figura 25 – Mensagem de requisição de alteração do estado do LED do protótipo recebidos no Cliente de teste MQTTC.	47
Figura 26 – Criando uma política no AWS IoT (A).	55
Figura 27 – Criando uma política no AWS IoT (B).	55
Figura 28 – Criando uma política no AWS IoT (C).	56
Figura 29 – Criando uma política no AWS IoT (D).	56
Figura 30 – Criando uma política no AWS IoT (E).	57
Figura 31 – Criando uma política no AWS IoT (F).	57
Figura 32 – Criando uma coisa no AWS IoT (A).	58
Figura 33 – Criando uma coisa no AWS IoT (B).	58
Figura 34 – Criando uma coisa no AWS IoT (C).	59
Figura 35 – Criando uma coisa no AWS IoT (D).	59
Figura 36 – Criando uma coisa no AWS IoT (E).	60
Figura 37 – Criando uma coisa no AWS IoT (F).	60
Figura 38 – Criando uma coisa no AWS IoT (G).	61

Lista de tabelas

Tabela 1 – Parâmetros mínimos recomendados pela AWS para o desenvolvimento de um dispositivo de IoT integrado à AVS.	38
Tabela 2 – kits de desenvolvimento recomendados pela Amazon para o desenvolvimento de aplicações IoT (A).	53
Tabela 3 – kits de desenvolvimento recomendados pela Amazon para o desenvolvimento de aplicações IoT (B).	54

Listas 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>	40
3.2	Formato da mensagem de alternância do estado do LED.	41
3.3	Trecho do arquivo JSON que descreve o front-end da habilidade <i>B-L475E-IOT01A2-Skill</i>	43

Lista de abreviaturas e siglas

ASK	Alexa Skills Kit
AVS	Alexa Voice Service
AWS	Amazon Web Services
CLI	AWS Command Line Interface
EC2	Amazon Elastic Compute Cloud
FOB	Livre A Bordo
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

1	INTRODUÇÃO	15
1.1	Estrutura	16
2	REFERENCIAL TEÓRICO	17
2.1	IoT	17
2.2	Computação em nuvem com a AWS	18
2.3	AWS IAM	19
2.4	Segurança	20
2.5	AWS IoT	21
2.5.1	AWS IoT Core	23
2.6	AWS Lambda	24
2.7	MQTT	24
2.7.1	MQTT Publish Message	27
2.7.2	MQTT Subscribe Message	27
2.7.3	MQTT Suback Message	28
2.7.4	MQTT Unsubscribe Message	29
2.7.5	MQTT Unsuback Message	29
2.8	Amazon Alexa	29
2.8.1	Alexa Skills Kit	30
2.8.2	Alexa Voice Service	32
2.9	Trabalhos correlatos	32
2.9.1	PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ	32
2.9.2	<i>Creating an Alexa voice controlled IoT using a Raspberry Pi</i>	33
3	METODOLOGIA	37
3.1	Objetivos	37
3.2	Projeto de Hardware	38
3.3	Criação de um nó IoT na rede AWS	39
3.4	Estabelecendo conexão entre o protótipo e o AWS IoT	40
3.5	Criação de Habilidades com o Alexa Skills Kit	41
3.5.1	Front-end da habilidade <i>B-L475E-IOT01A2-Skill</i>	42
3.5.2	Back-end da habilidade <i>B-L475E-IOT01A2-Skill</i>	42
4	RESULTADOS E DISCUSSÃO	45

5	CONCLUSÕES	49
5.1	Sugestão de Trabalhos Futuros	49
	REFERÊNCIAS	51
A	APÊNDICE	53
A.1	Kits de Desenvolvimento	53
A.2	Criação de uma política para o AWS IoT	54
A.3	Criação de uma coisa no AWS IoT	57
A.4	Configuração do dispositivo B-L475E-IOT01A2 via USB usando o aplicativo Tera Term	61

1 Introdução

Computação em nuvem é um termo que surgiu na década de 1960 com os sistemas de compartilhamento de tempo. Atualmente, define-se computação em nuvem como a entrega de recursos de TI através da Internet e sob demanda com definição de preço conforme o uso (??). Dessa forma, empresas e usuários de tecnologia conseguem expandir seus negócios de maneira rápida e estratégica sem ter que se preocupar com uma infraestrutura própria de TI. Algumas empresas que fornecem serviço de computação em nuvem são: AWS, Google Cloud Platform, Microsoft Azure e IBM Cloud.

Dessa forma, a nuvem oferece agilidade, elasticidade e economia de custos para organizações de todos os tipos, portes e setores. Esse tipo de tecnologia permite a geração rápida de recursos de acordo com a evolução das necessidades empresariais. Assim, empresas que fazem uso de computação em nuvem não correm o risco de provisionar recursos em excesso para absorver picos de atividades empresariais futuras. Por fim, a nuvem permite a troca de despesas fixas, como *datacenters*, segurança de dados e servidores públicos, por despesas variáveis que dependem somente do que foi consumido.

Já o termo Internet das Coisas, ou simplesmente IoT, surgiu em 1999 com um sistema de sensores omnipresentes conectados à internet. A tecnologia ganhou popularidade e se expande com o crescimento de tecnologias de conectividade, computação em nuvem e aprendizado de máquina, por exemplo. Uma definição simplificada de IoT é: uma rede de itens conectados à internet. A principal vantagem dessa tecnologia é a não intervenção humana, o que gera um aumento na produtividade de tarefas repetitivas e escalabilidade.

Outra tecnologia recente são as assistentes virtuais inteligentes. Uma assistente virtual inteligente é um agente de software que pode realizar tarefas ou serviços para um indivíduo (Wikipedia, 2022). Algumas informações que as assistentes são capazes de acessar e rapidamente leva-las ao usuário, em linguagem natural, são: notícias, condições meteorológicas e de trânsito, agenda do usuário, entre outras. Alguns exemplos disponíveis no mercado são a Alexa, Microsoft Cortana, Siri e Google Assistant.

Em 25 de setembro de 2019, o Alexa e o Google Assistant puderam ajudar seus usuários a se candidatarem a empregos no McDonald's usando serviços de reconhecimento de voz. É o primeiro serviço de emprego do mundo usando o serviço de comando de voz. A Amazon anunciou em 25 de setembro de 2019 que Alexa em breve poderá imitar vozes de celebridades, incluindo Samuel L. Jackson, custando US \$ 0,99 para cada voz (Wikipedia, 2022).

Dessa forma, uma tendência das tecnologias focadas na experiência do usuário, como a IoT, é o suporte às assistentes virtuais. Diante do que foi dito, este trabalho

propõe criar um protótipo de um nó IoT integrado à assistente de voz Alexa e serviços em nuvem.

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 [Apêndice A](#), 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;
- Aprendizado de máquina 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:

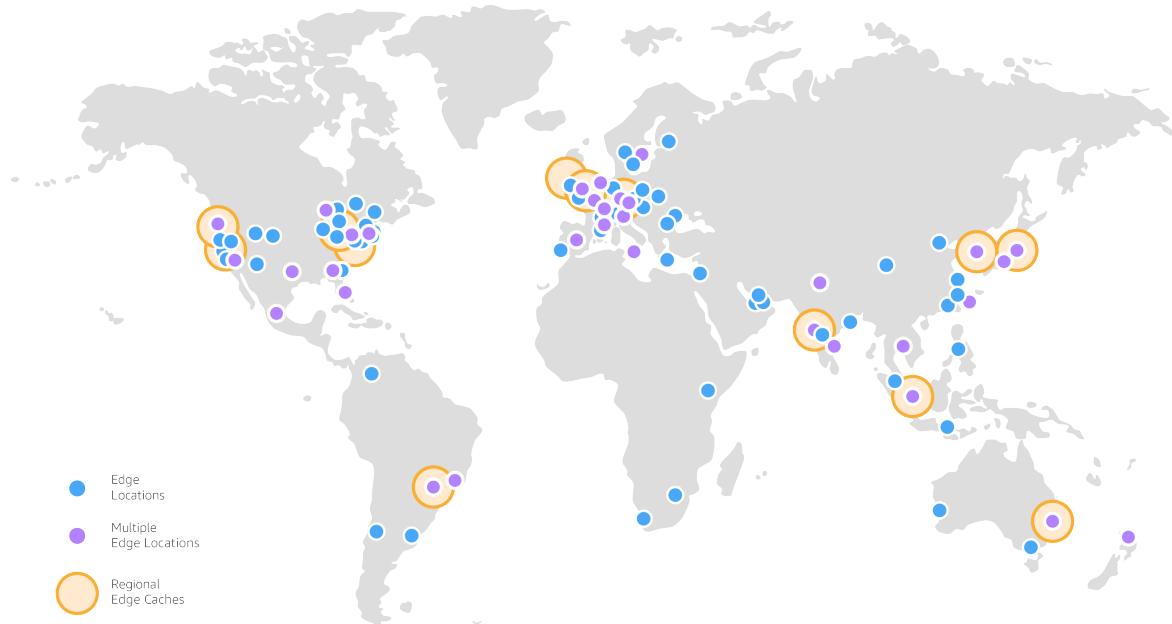
- Teclado, botões e microfones;
- Alarmes, visores e alto-falantes;
- Sensores;
- Atuadores;

2.2 Computação em nuvem 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.

2.3 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.

Figura 2 – Diagrama do serviço AWS IAM.



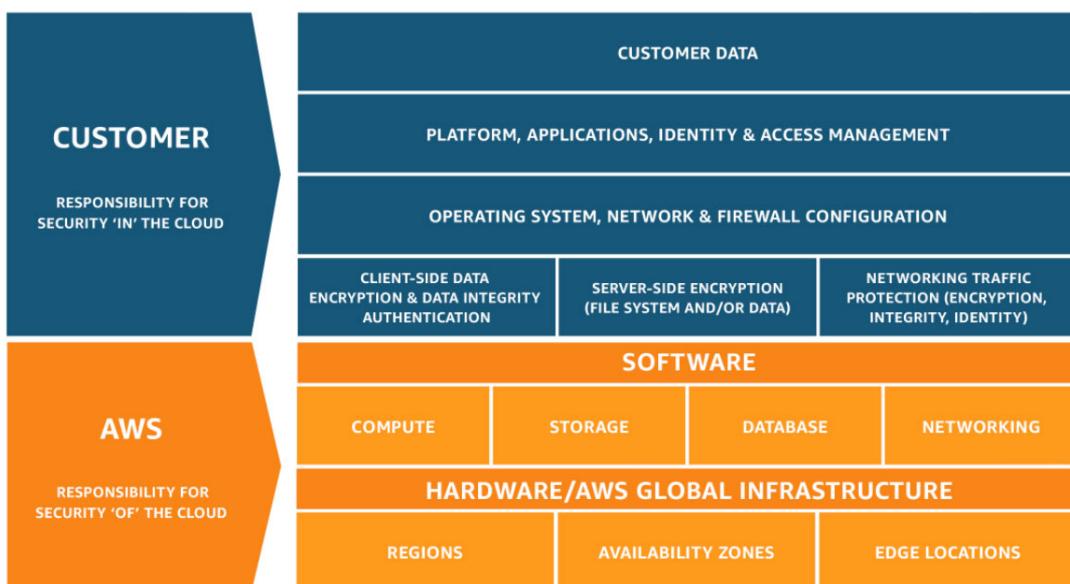
Fonte: Amazon AWS, com edições feitas pelo autor (2022).

2.4 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 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 2.5](#), 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.

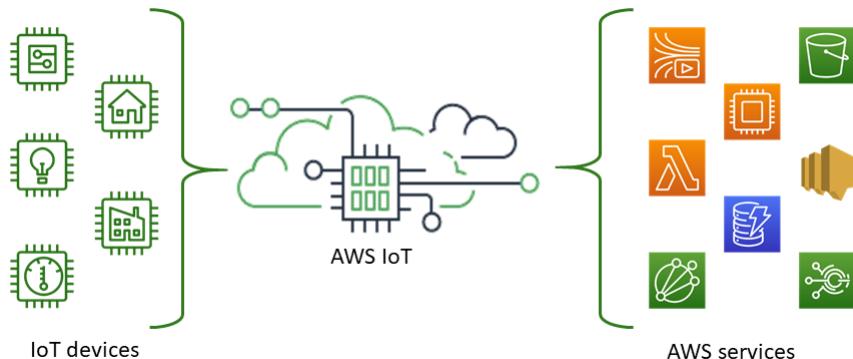
```

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 }
```

2.5 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.

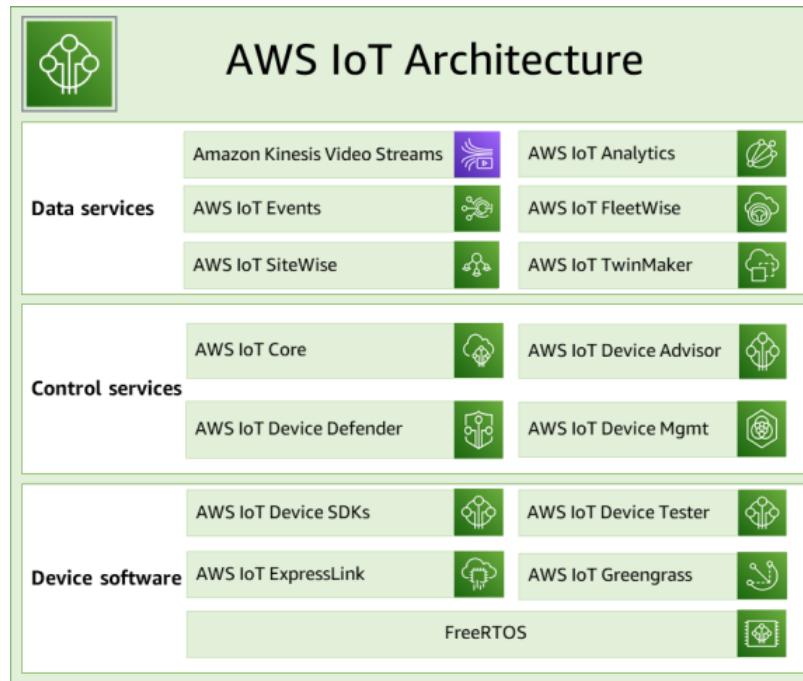
Para a completa integração desses dispositivos com serviços AWS e usuários, alguns componentes são essenciais. Aplicativos de celulares, por exemplo, são utilizados para que o usuário tenha acesso aos seus aparelhos e os configure conforme a sua preferência. Tem-se como outro exemplo os protocolos utilizados 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 2.7](#). Para o desenvolvimento de softwares a nível de dispositivo, o AWS IoT também fornece suporte para 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 2.5.1](#).

Figura 5 – Exemplos de serviços fornecidos pelo AWS IoT.



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

2.5.1 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).

2.6 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 back-end 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 precificação acontece sobre o número total de solicitações feitas por todas as funções. O preço final depende da quantidade de memória alocada por função.

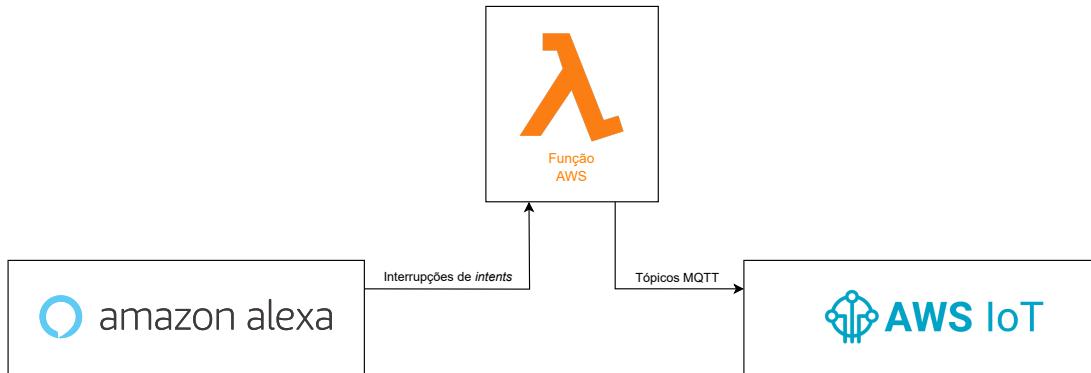
Para começar a usar o AWS Lambda, o usuário deve primeiro carregar o código na nuvem AWS. Isso pode ser feito via AWS CLI, submissão de um arquivo no formato zip ou até mesmo criando o código diretamente no console do Lambda. O Lambda oferece suporte nativo às linguagens Java, Go, PowerShell, Node.js, 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.

2.7 MQTT

MQTT é um protocolo de mensagens para dispositivos IoT. Ele foi desenvolvido para dispositivos com limitada disponibilidade de largura de banda (*network bandwidth*).

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



Fonte: Produzido pelo autor (2022).

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 2.7.1](#).
- *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 2.7.2](#);

- *Suback*: Para confirmar cada assinatura, o *broker* envia uma mensagem de confirmação *Suback* ao cliente. Mais informações sobre esse tipo de mensagem podem ser encontradas na [subseção 2.7.2](#);
- *Unsubscribe*: Esta mensagem exclui as assinaturas existentes de um cliente no *broker*. Mais informações sobre esse tipo de mensagem podem ser encontradas na [subseção 2.7.4](#);
- *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 encontradas na [subseção 2.7.5](#).

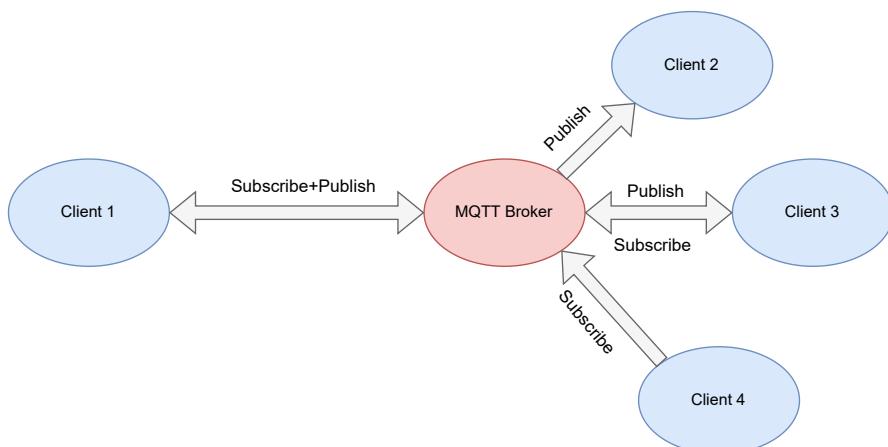
Depois que um cliente envia com êxito a mensagem *Subscribe* e recebe a mensagem *Suback*, ele obtém todas as mensagens publicadas de um tópico nas assinaturas contidas da mensagem *Subscribe*. Após receber o *Unsuback* do *broker*, o cliente pode assumir que as assinaturas na mensagem *Unsubscribe*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

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



Fonte: Produzido pelo autor (2022).

2.7.1 MQTT Publish Message

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

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



Fonte: HiveMQ (2015).

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 (“alexa/builtin/device”, por exemplo);
- **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 especificado. 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*.

2.7.2 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.

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

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



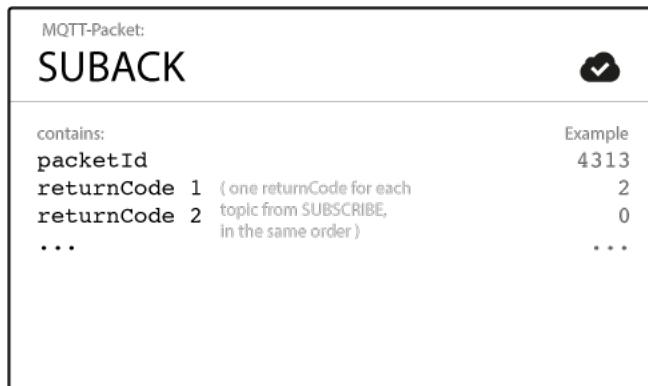
Fonte: HiveMQ (2015).

- **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.

2.7.3 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.

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



Fonte: HiveMQ (2015).

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.

2.7.4 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.

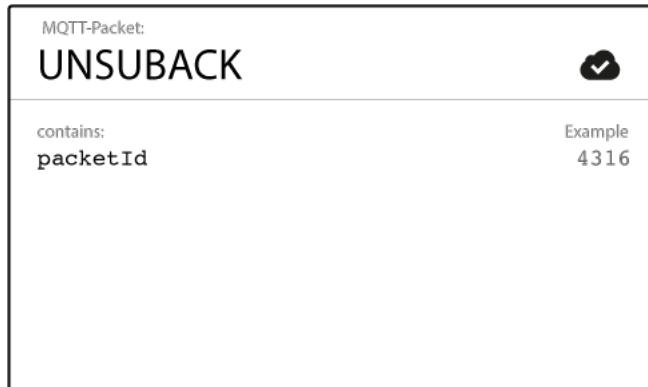
2.7.5 MQTT Unsuback Message

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.

2.8 Amazon Alexa

A Amazon Alexa, conhecida simplesmente como Alexa, é uma assistente virtual inteligente capaz de controlar dispositivos inteligentes e ter interações de voz com os usuários. As capacidades da Alexa podem ser expandidas por terceiros através da instalação de

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



Fonte: HiveMQ (2015).

habilidades (aplicativos). Algumas categorias de habilidades em destaque são: Notícias, Negócios e Finanças, Jogos e Curiosidades, Saúde e Boa Forma e Produtividade. A loja oficial de habilidades da Amazon permite que o usuário faça busca de habilidades por categorias, assim como pode ser visto na [Figura 13](#).

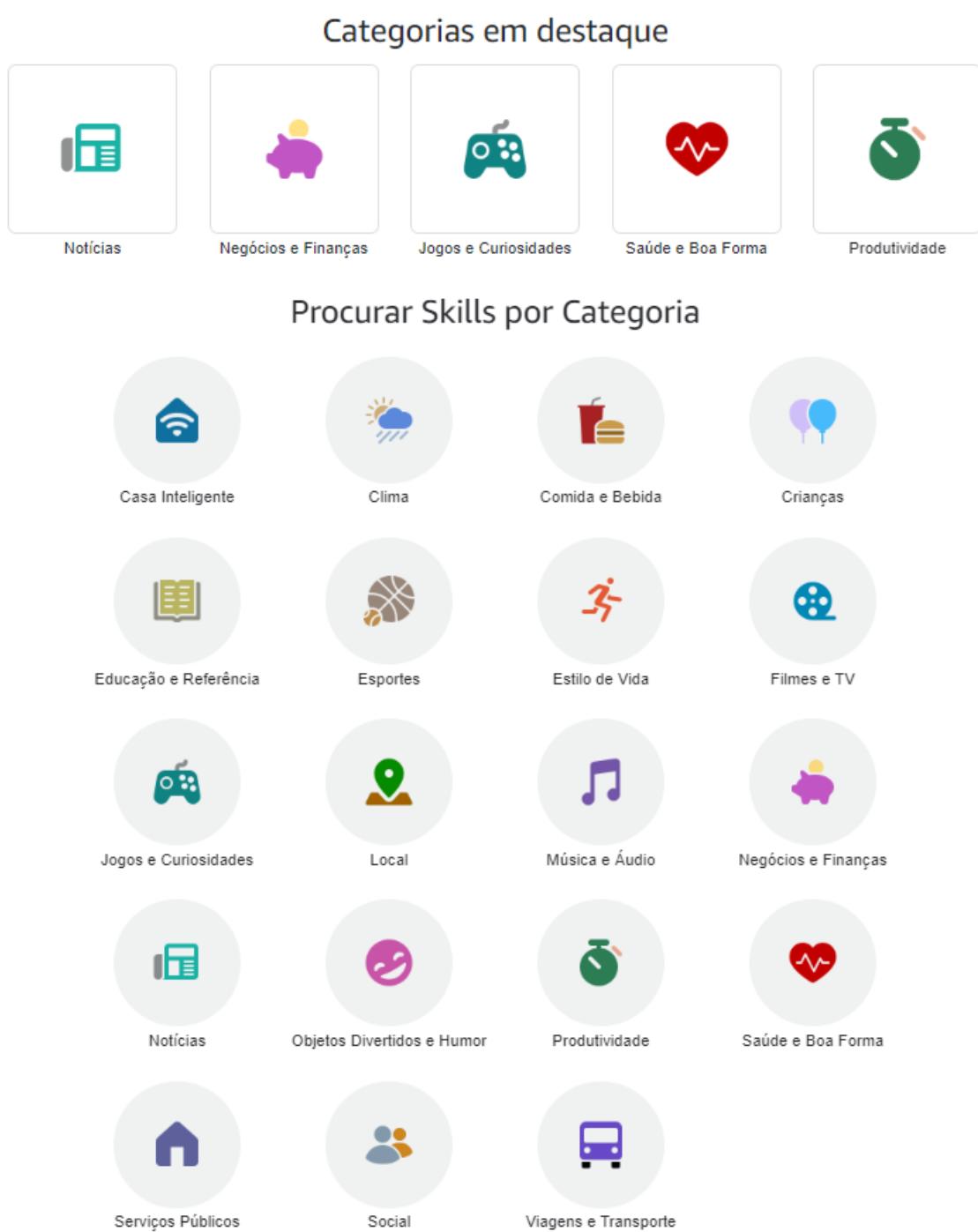
2.8.1 Alexa Skills Kit

O ASK fornece APIs e ferramentas para a criação de habilidades Alexa. Empresas podem desenvolver habilidades para seus produtos e serviços, ou contratarem uma agência especialista em Alexa. Uma habilidade Alexa tem tanto uma interface de voz com o usuário, como uma lógica de aplicativo (Desenvolvedor Amazon, 2022).

Quando um usuário interage com um dispositivo Alexa, ocorre um processamento da fala no contexto do seu modelo de interação para interpretar o que foi requisitado. A Alexa, então, envia a solicitação à sua lógica de aplicação, responsável por processar e computar essa informação. Essa computação acontece em nuvem AWS ou outro servidor. Dessa forma, caracteriza-se o processamento da fala e interpretação do que foi requisitado como front-end e a computação da lógica de aplicação como back-end. A [Figura 14](#) mostra um diagrama produzido pela Amazon explicando o funcionamento de uma habilidade Alexa.

Para o desenvolvimento de software, os ASK SDKs permitem uma redução da complexidade do projeto. Os SDKs para Java, Node.js e Python fornecem funções específicas da linguagem para tarefas comuns, permitindo a concentração na lógica das habilidades, não no código padrão (Desenvolvedor Amazon, 2022). A Amazon também fornece distintas de ferramentas de desenvolvimento. O desenvolvedor pode usar o console do portal ou CLI do ASK para criar, gerenciar, testar e publicar habilidades.

Figura 13 – Busca de habilidades da Alexa por categorias.



Fonte: Amazon (2022).

Figura 14 – Diagrama explicando o funcionamento de uma habilidade Alexa.



Fonte: Desenvolvedor Amazon (2022).

2.8.2 Alexa Voice Service

A Amazon permite que fabricantes construam dispositivos com a Alexa integrada usando o AVS, um serviço baseado na AWS com APIs que permitem a integração de dispositivos com a Alexa. Dessa forma, produtos usuários do AVS têm acesso a diversos recursos da Alexa, incluindo as habilidades Alexa. A principal vantagem do AVS é o fornecimento de APIs que permitem o reconhecimento automático de fala, com computação executada em nuvem, e compreensão de linguagem natural.

Um exemplo de uso do AVS é um dispositivo IoT com a Alexa embutida. Nesse exemplo, o desenvolvedor fará uso dos serviços AVS, AWS IoT e Login da Amazon. Um diagrama desenvolvido pela Amazon mostrando um exemplo de integração do AVS ao AWS IoT pode ser visto na [Figura 15](#).

2.9 Trabalhos correlatos

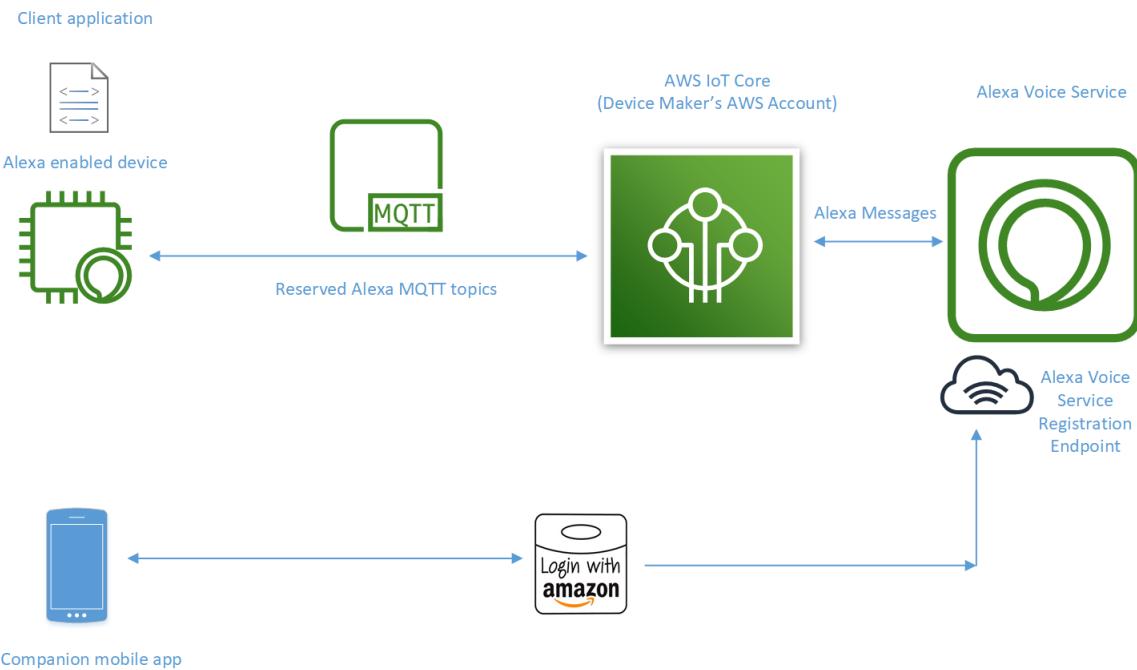
Essa sessão busca detalhar dois trabalhos correlatos ao protótipo desse trabalho. Os dois trabalhos escolhidos são:

- 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;

2.9.1 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

Figura 15 – Diagrama de integração do AVS ao serviço AWS IoT Core.



Fonte: Amazon AWS (2022).

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.

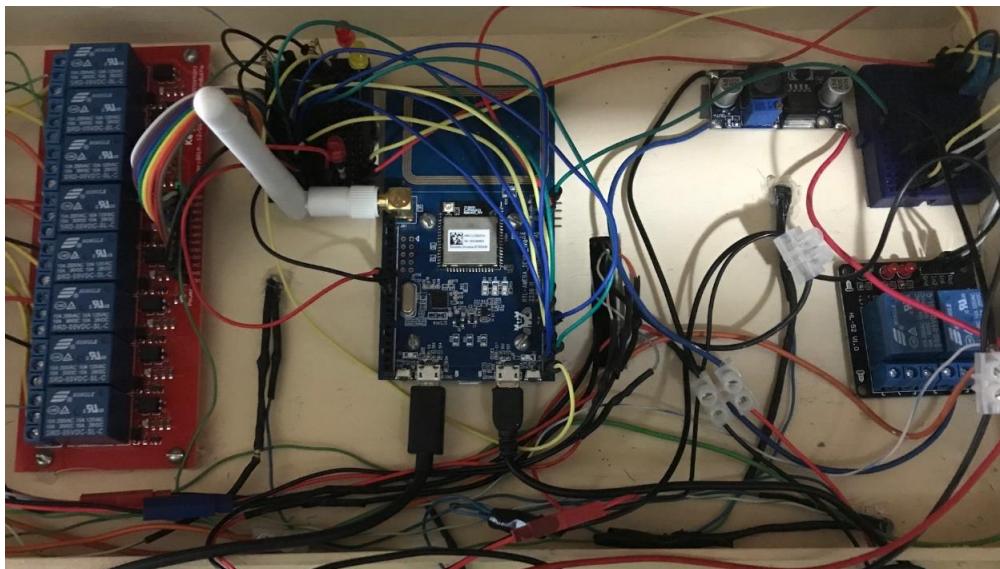
Para testes com o protótipo, uma maquete e um circuito foram montado. Esse circuito conta com o microcontrolador Arduino Ameba, dois módulos relé para Arduino, um regulador de tensão, um eletroímã de 12V e o sensor de temperatura DHT11. A Figura 16 mostra o circuito eletrônico completo da maquete, enquanto a Figura 17 mostra a vista frontal da maquete.

2.9.2 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 nó 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 18 mostra uma captura de tela do vídeo 5 da sequência de vídeos do projeto do YouTuber John Allwork.

Figura 16 – Circuito eletrônico completo da maquete do projeto “PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ”.



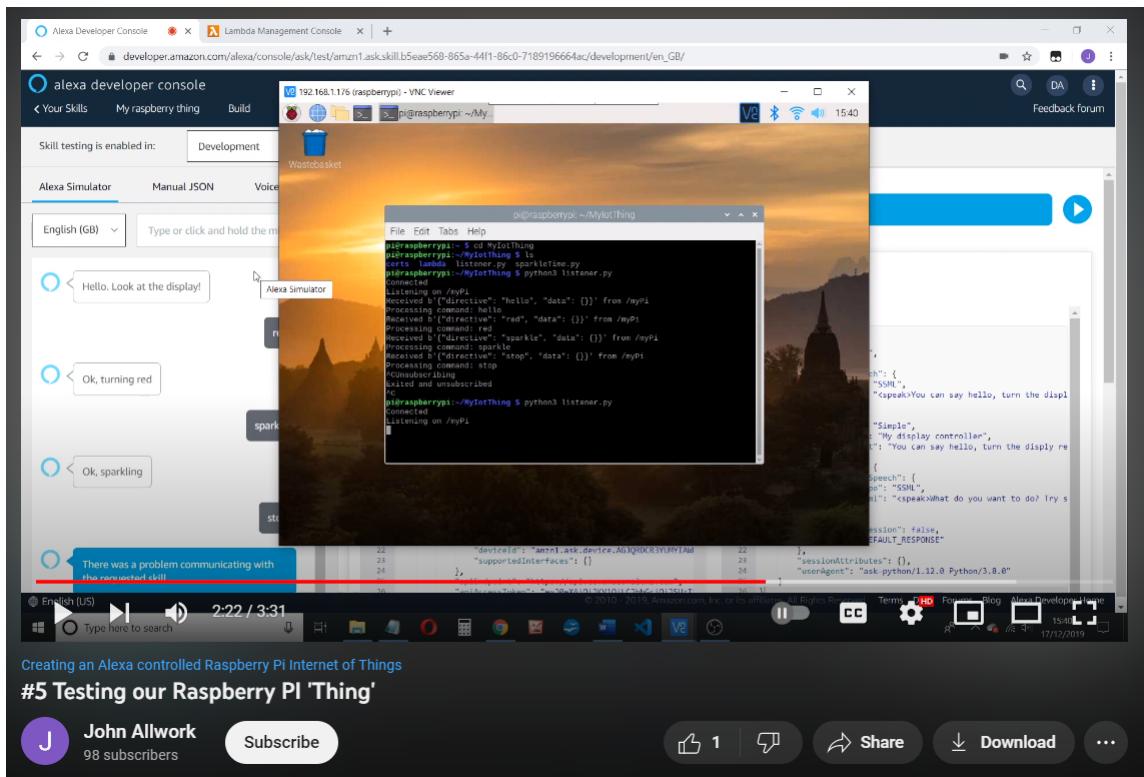
Fonte: Monografia de Leandro Dallarosa Neto (2018).

Figura 17 – Imagem frontal da maquete do projeto “PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL UTILIZANDO UMA ASSISTENTE DE VOZ”.



Fonte: Monografia de Leandro Dallarosa Neto (2018).

Figura 18 – Captura de tela do vídeo 5 da sequência de vídeos do projeto “*Creating an Alexa voice controlled IoT using a Raspberry Pi*”.



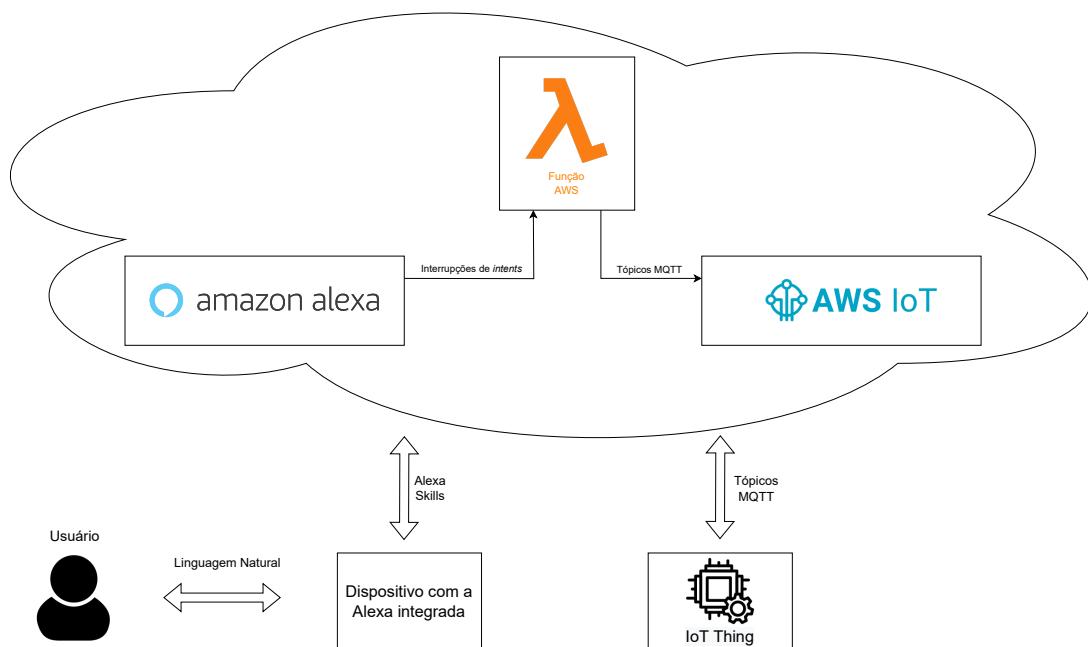
Fonte: Produzido pelo autor (2022).

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 19](#) mostra um diagrama em alto nível do projeto.

Figura 19 – 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.

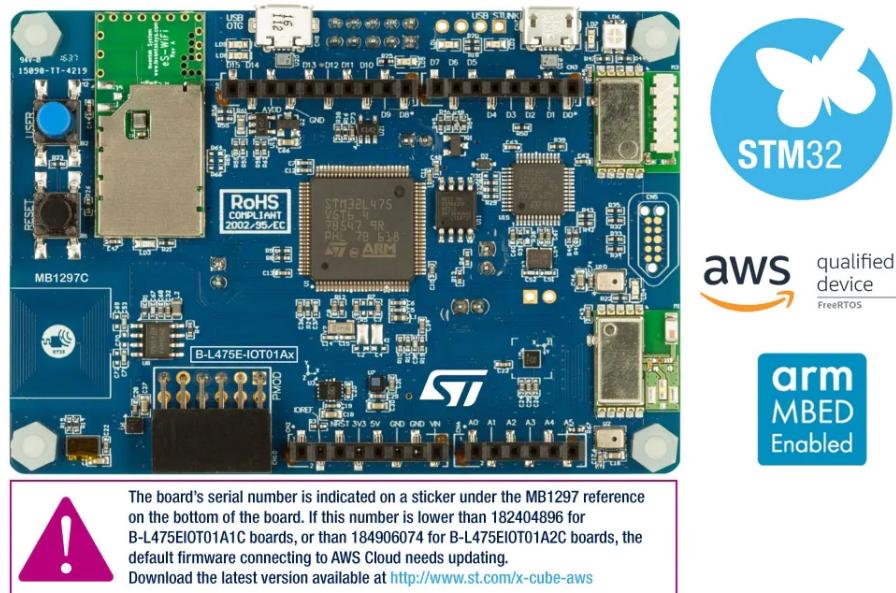
Para a prototipagem, alguns kits de desenvolvimento disponíveis no mercado foram estudados. As tabelas [Tabela 2](#) e [Tabela 3](#), presentes no [Apêndice A](#), mostram informações coletadas de cinco diferentes kits.

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 20](#).

Figura 20 – 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 A.3](#).

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:

- a) 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;
- b) O projeto conta com um arquivo binário que pode ser diretamente carregado no kit;
- c) 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 front-end e o back-end da habilidade, respectivamente.

3.5.1 Front-end da habilidade *B-L475E-IOT01A2-Skill*

O ASK permite que o desenvolvimento do front-end aconteça via interface web e/ou arquivo JSON. O código que define o front-end 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 Back-end da habilidade *B-L475E-IOT01A2-Skill*

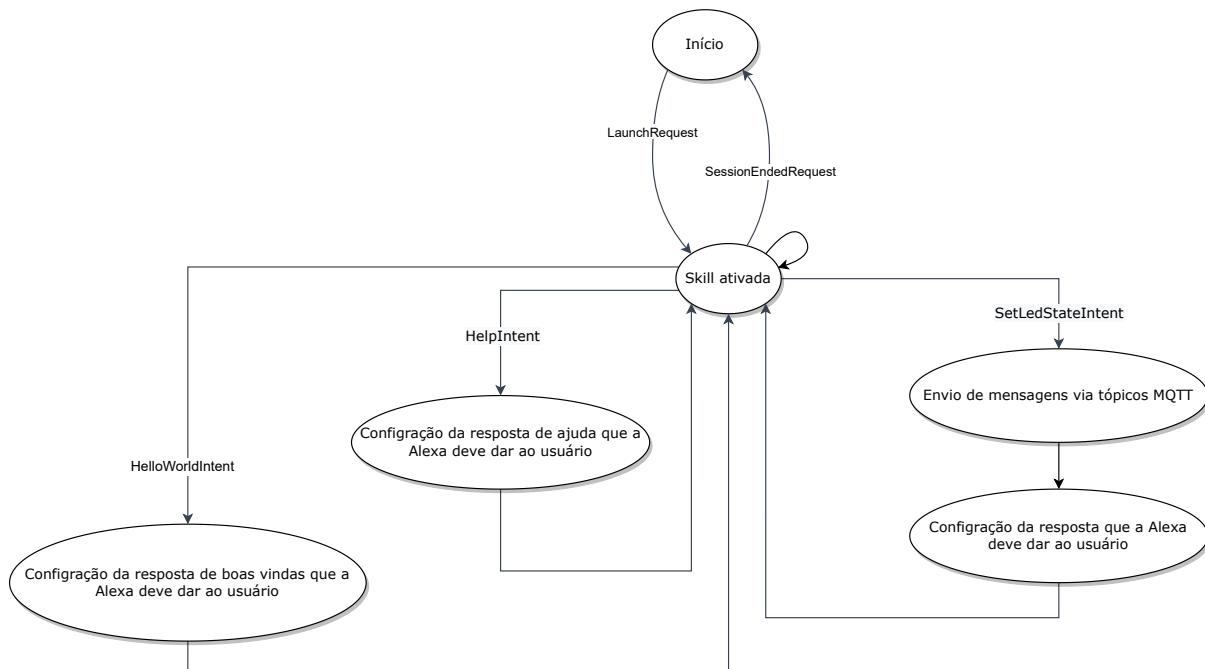
Para o desenvolvimento do back-end, optou-se por estender as funcionalidades do ASK através de funções no AWS Lambda. Dessa forma, o desenvolvimento pode acontecer em Node.js, Python etc, assim como foi mencionado na [seção 2.6](#). Para esse projeto, escolheu-se Python e *ask-sdk* como linguagem de 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 e segue o diagrama de estados apresentado na [Figura 21](#). 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 front-end 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 21 – Diagrama de estados da função *B-L475E-IOT01A2-Handler*.



Fonte: Produzido pelo autor (2022).

4 Resultados e Discussão

Para discussão dos resultados obtidos, deve-se retomar os objetivos propostos na [seção 3.1](#). Alcançar os objetivos funcionais do projeto está condicionado à escolha de um *hardware* que atenda todos os itens definidos como requisito. Dessa forma, o dispositivo escolhido para a prototipagem (B-L475E-IOT01A2) atendeu todos os requisitos funcionais do projeto, assim como os requisitos mínimos recomendados pela AWS. Esse dispositivo atende também os requisitos não-funcionais estipulados, uma vez que ele é qualificado pela AWS, apresenta versatilidade (contando com vários sensores, Bluetooth, Wi-Fi, sensor NFC, suporte à conexões em baixas frequências e microfones) e possui um preço de mercado inferior a seus concorrentes, uma vez que apresenta configurações mínimas para o desenvolvimento de aplicações AWS.

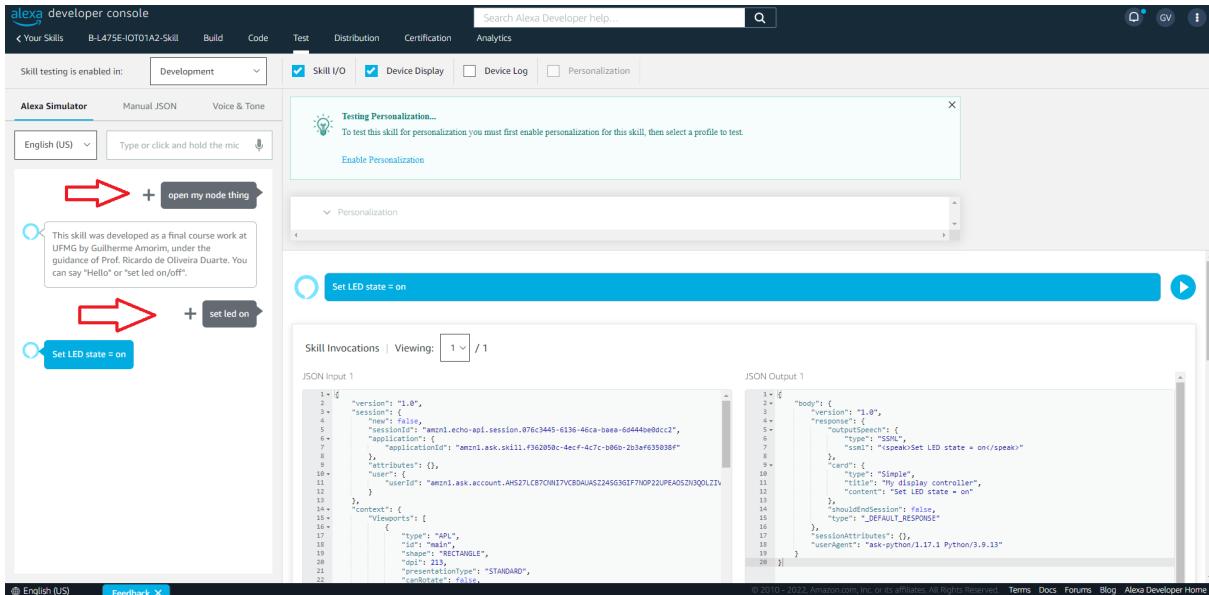
Durante os testes com o dispositivo, a empresa STMicroelectronics ofereceu suporte em seus canais de comunicação, o que acelerou o processo de prototipagem. Em contraponto, muitos códigos de exemplo fornecidos pela mesma empresa continham problemas de configuração de compilação, o que atrasou e dificultou o processo de desenvolvimento. Ademais, o processo de importação de produtos para prototipagem, assim como de outros produtos eletrônicos para o Brasil, possui alta alíquota de imposto, podendo chegar a 95% do FOB (valor da mercadoria e outras despesas anteriores ao embarque), o que encareceu o projeto.

Para a validação da habilidade *B-L475E-IOT01A2-Skill*, fez-se uso do *Amazon Alexa Console* e do *Cliente de teste MQTT*. Ao pronunciar a frase “set led on” para a Alexa, é possível visualizar que uma mensagem de mudança de estado do LED é recebida no tópico “\$aws/things/B-L475E-IOT01A2/shadow/update”. As figuras [Figura 22](#) e [Figura 23](#) mostram capturas de tela do processo de validação da habilidade *B-L475E-IOT01A2-Skill*.

Para os testes com o dispositivo *B-L475E-IOT01A2*, também utilizou-se o *Cliente de teste MQTT*. O primeiro passo foi a configuração do dispositivo via USB, processo detalhado na [seção A.4](#). Durante a execução da aplicação, dois tipos de mensagens são enviadas para o tópico MQTT “\$aws/things/B-L475E-IOT01A2/shadow/update”: uma com os dados atualizados dos sensores do kit de desenvolvimento e outra com uma requisição de alteração do estado do LED caso o botão da placa seja acionado. As figuras [Figura 24](#) e [Figura 25](#) mostram capturas de tela do *Cliente de teste MQTT* ao receber mensagens de atualização de dados dos sensores e de requisição de alteração do estado do LED, respectivamente.

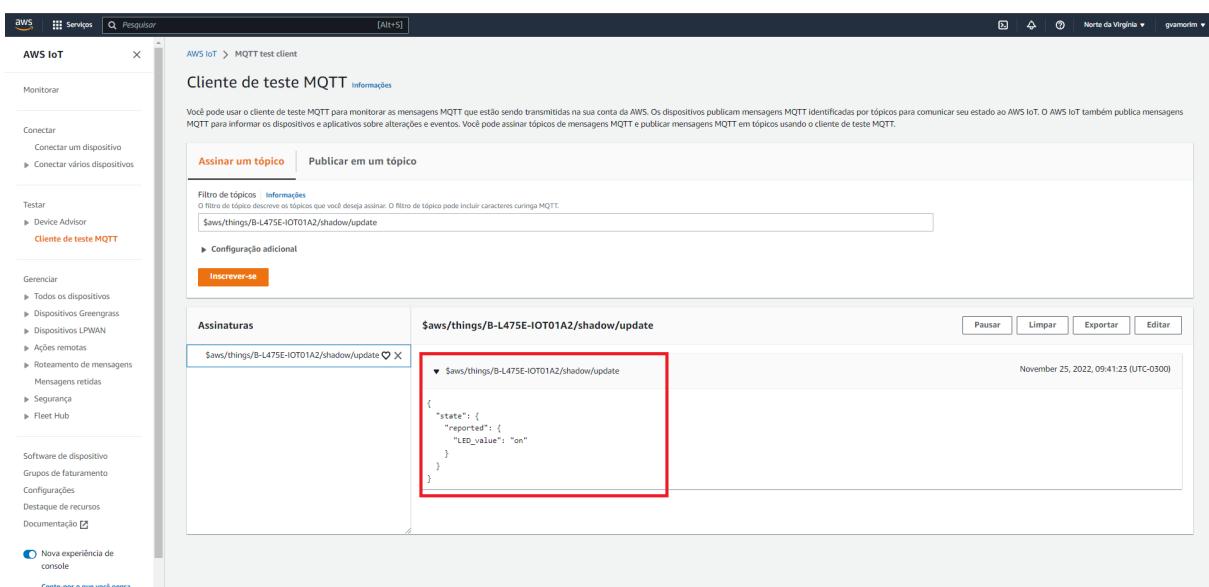
Após os testes individuais do dispositivo *B-L475E-IOT01A2* e da habilidade *B-*

Figura 22 – Simulação de uma iteração do usuário com a Alexa através do *Amazon Alexa Console*.



Fonte: Produzido pelo autor (2022).

Figura 23 – Cliente de teste MQTT durante a execução da habilidade *B-L475E-IOT01A2-Skill*.



Fonte: Produzido pelo autor (2022).

Figura 24 – Mensagem de atualização os dados dos sensores do protótipo recebidos no Cliente de teste MQTT.

The screenshot shows the AWS IoT Test Client interface. On the left, a sidebar lists various options like Monitorar, Conectar, Testar, and Cliente de teste MQTT. The main area is titled 'Assinar um tópico' (Subscribe to a topic) with a filter set to '\$aws/things/B-L475E-IOT01A2/shadow/update'. It displays two messages under the topic '\$aws/things/B-L475E-IOT01A2/shadow/update':

- Message 1 (November 25, 2022, 10:34:47 UTC-0300):


```
{
        "state": {
          "reported": {
            "temperature": 26.79,
            "humidity": 60.26,
            "pressure": 922.07,
            "proximity": 730,
            "acc_x": -16, "acc_y": -10, "acc_z": 1041,
            "gyr_x": 1010, "gyr_y": -98, "gyr_z": 70,
            "mag_x": -224, "mag_y": 187, "mag_z": 240
          }
        }
      }
```
- Message 2 (November 25, 2022, 10:34:41 UTC-0300):


```
{
        "state": {
          "desired": {
            "LED_value": "On"
          }
        }
      }
```

A red arrow points from the text above to the second message.

Fonte: Produzido pelo autor (2022).

Figura 25 – Mensagem de requisição de alteração do estado do LED do protótipo recebidos no Cliente de teste MQTT.

The screenshot shows the AWS IoT Test Client interface, similar to Figure 24. The sidebar includes 'Assinar um tópico' (Subscribe to a topic) with a filter set to '\$aws/things/B-L475E-IOT01A2/shadow/update'. It displays three messages under the topic '\$aws/things/B-L475E-IOT01A2/shadow/update':

- Message 1 (November 25, 2022, 10:36:14 UTC-0300):


```
{
        "state": {
          "desired": {
            "LED_value": "On"
          }
        }
      }
```
- Message 2 (November 25, 2022, 10:36:15 UTC-0300):


```
{
        "state": {
          "desired": {
            "LED_value": "On"
          }
        }
      }
```
- Message 3 (November 25, 2022, 10:36:12 UTC-0300):


```
{
        "state": {
          "reported": {
            "LED_value": "Off"
          }
        }
      }
```

Two red arrows point from the text above to the first two messages.

Fonte: Produzido pelo autor (2022).

L475E-IOT01A2-Skill, testou-se a alteração do estado do LED no dispositivo após o pronunciamento “set led on” para a Alexa. O vídeo PENDING mostra a execução completa do teste.

5 Conclusões

Conclui-se nesse trabalho que o kit de desenvolvimento *B-L475E-IOT01A2* da STMicroelectronics atende os requisitos mínimos para a prototipagem de um nó IoT integrado ao serviço AWS IoT. Ademais, a infraestrutura de serviços da Amazon satisfaz as necessidades de um desenvolvedor de aplicações IoT dado que a AWS oferece os seguintes recursos a seus clientes:

- Segurança com o modelo compartilhado de responsabilidade e o IAM.;
- Suporte para a criação de nós IoT com o serviço AWS IoT;
- Suporte de desenvolvimento e testes para o protocolo MQTT;
- Gerenciamento de dispositivos, ou conjunto de dispositivos, via AWS IoT Core;
- Pagamento somente do que for utilizado, reduzindo custos e permitindo crescimento em escala.

Conclui-se, também, que o desenvolvimento de habilidades para a Alexa através do ASK atende as necessidade dos desenvolvedores, uma vez a Amazon provem recursos para teste, SDKs para o desenvolvimento, suporte à CLI e possibilidade de implementação do back-end na própria AWS (através do AWS Lambda).

5.1 Sugestão de Trabalhos Futuros

Uma sugestão de trabalho futuro é a implementação de uma Alexa integrada no dispositivo *B-L475E-IOT01A2*. Esse kit de desenvolvimento atende todos os parâmetros mínimos definidos pela Amazon para um dispositivo de IoT integrado à AVS (definidos na [Tabela 1](#)). Um possível diagrama do projeto para esse trabalho futuro foi apresentado na [Figura 15](#).

Outra sugestão de trabalho futuro é a implementação do suporte a outras linguagens, como o português e o italiano, para a habilidade *B-L475E-IOT01A2-Skill*.

Por fim, outro trabalho importante é a produtização da habilidade *B-L475E-IOT01A2*. A produtização de habilidades permite que usuários tenham acesso aos recursos dessa tecnologia em seus dispositivos Alexa após o *download* na loja de habilidades da Amazon.

A série de vídeos [Zero to Hero: Alexa Skills](#), hospedada no canal [Alexa Developers](#), demonstra como adicionar funcionalidades à habilidades Alexa. Os vídeos [Part 2: Skill](#)

[Internationalization \(i18n\)](#), [Interceptors & Error Handling](#) e [Part 11: Publishing](#) mostram como adicionar novas linguagens à habilidades Alexa e o processo de publicação, respectivamente.

Referências

Nenhuma citação no texto.

Amazon AWS. O que é cloud computing (computação em nuvem)? - Amazon Web Services, 2022. Acesso em: 23 Jun. 2022. Citado na página [15](#).

A Apêndice

A.1 Kits de Desenvolvimento

As tabelas [Tabela 2](#) e [Tabela 3](#) mostram os parâmetros de cinco kits de desenvolvimento cogitados para serem o protótipo do projeto.

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).

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).

A.2 Criação de uma política para o AWS IoT

A sequência de figuras apresentam capturas de telas do processo de criação de uma política para o AWS IoT.

Acesse a [página principal do serviço AWS IoT](#), expanda a opção *Segurança* e selecione *Políticas*.

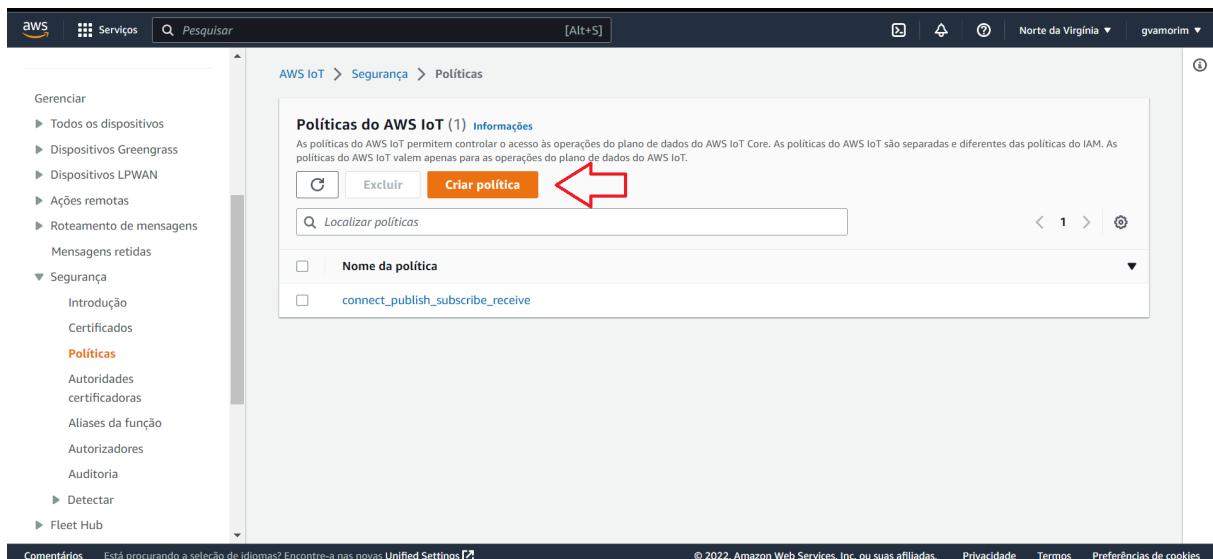
Figura 26 – Criando uma política no AWS IoT (A).



Fonte: Produzido pelo autor (2022).

Selecione a opção *Criar política*.

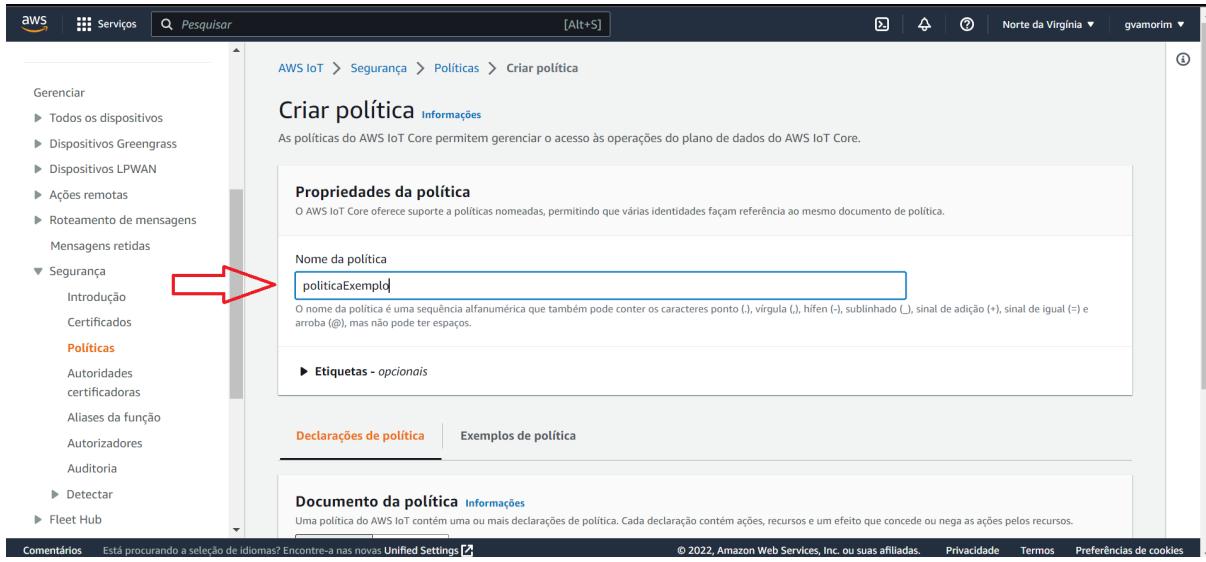
Figura 27 – Criando uma política no AWS IoT (B).



Fonte: Produzido pelo autor (2022).

Adicione um nome na opção *Nome da política*.

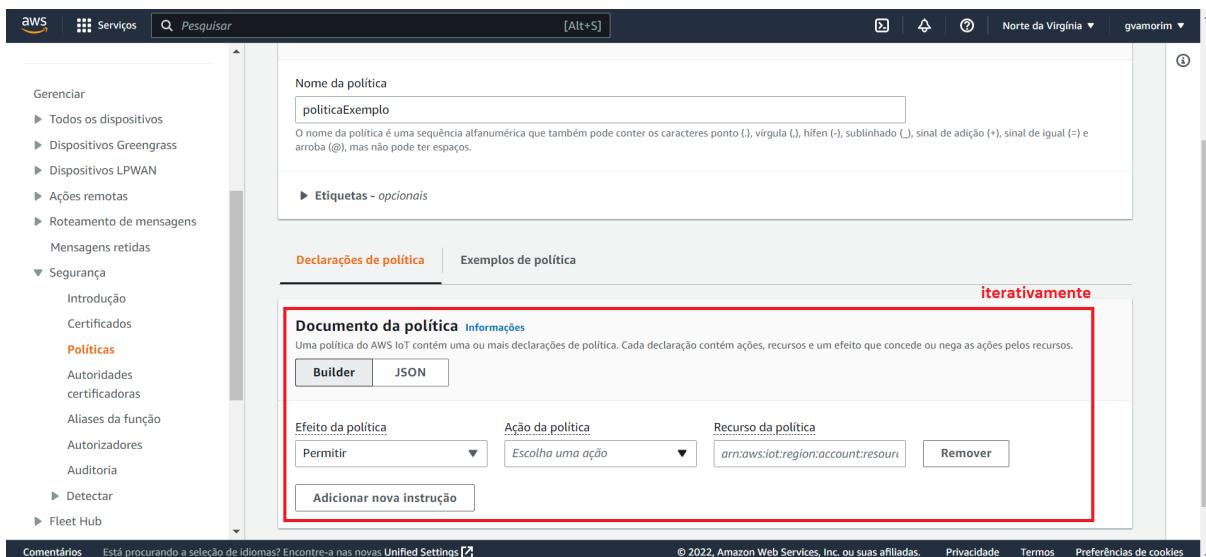
Figura 28 – Criando uma política no AWS IoT (C).



Fonte: Produzido pelo autor (2022).

Adicione declarações à política. As declarações podem ser adicionadas iterativamente (Figura 29) ou via documento JSON (Figura 30). O documento JSON da política utilizada no projeto foi apresentado no [Bloco de Código 3.1](#). Note que a opção *Resource*, para cada declaração, segue o seguinte formato: “arn:aws:iot: <regiao>:<id_do_usuario>:”.

Figura 29 – Criando uma política no AWS IoT (D).



Fonte: Produzido pelo autor (2022).

Figura 30 – Criando uma política no AWS IoT (E).

The screenshot shows the AWS IoT Policy Builder interface. On the left, there's a navigation sidebar with options like Gerenciar, Segurança, and Políticas. The main area is titled "Documento da política" and contains a JSON editor. The JSON code is as follows:

```

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 }
  
```

A red arrow points to the left side of the JSON code, highlighting the opening brace of the Statement array.

Fonte: Produzido pelo autor (2022).

Selecione a opção *Criar*.

Figura 31 – Criando uma política no AWS IoT (F).

The screenshot shows the AWS IoT Policy Builder interface with the same JSON policy document as in Figure 30. A red arrow points to the bottom right corner, where the "Criar" (Create) button is located.

Fonte: Produzido pelo autor (2022).

A.3 Criação de uma coisa no AWS IoT

A sequência de figuras Figura 32 até Figura 38 apresentam capturas de telas do processo de criação de uma coisa no AWS IoT.

Acesse a página principal do serviço AWS IoT, expanda a opção *Todos os Dispositivos* e selecione *coisas*.

Figura 32 – Criando uma coisa no AWS IoT (A).

The screenshot shows the AWS IoT console interface. On the left, there's a navigation sidebar with sections like 'Monitorar', 'Conectar', 'Testar', 'Gerenciar', 'Software de dispositivo', and 'Segurança'. Under 'Gerenciar', 'Todos os dispositivos' is expanded, and 'Coisas' is highlighted with a red arrow. The main content area has a dark header with the title 'AWS IoT' and a sub-header 'Conecte, teste e gerencie com segurança seus dispositivos de IoT'. Below this is a section titled 'Como funciona' with three cards: 'Conectar', 'Testar', and 'Gerenciar'. To the right, there are sections for 'Comece a usar o AWS IoT', 'Preço', 'Recursos de aprendizado', and 'Recursos de vídeo do AWS IoT'. A large orange button labeled 'Conectar dispositivo' is visible.

Fonte: Produzido pelo autor (2022).

Selecione a opção *Criar items*.

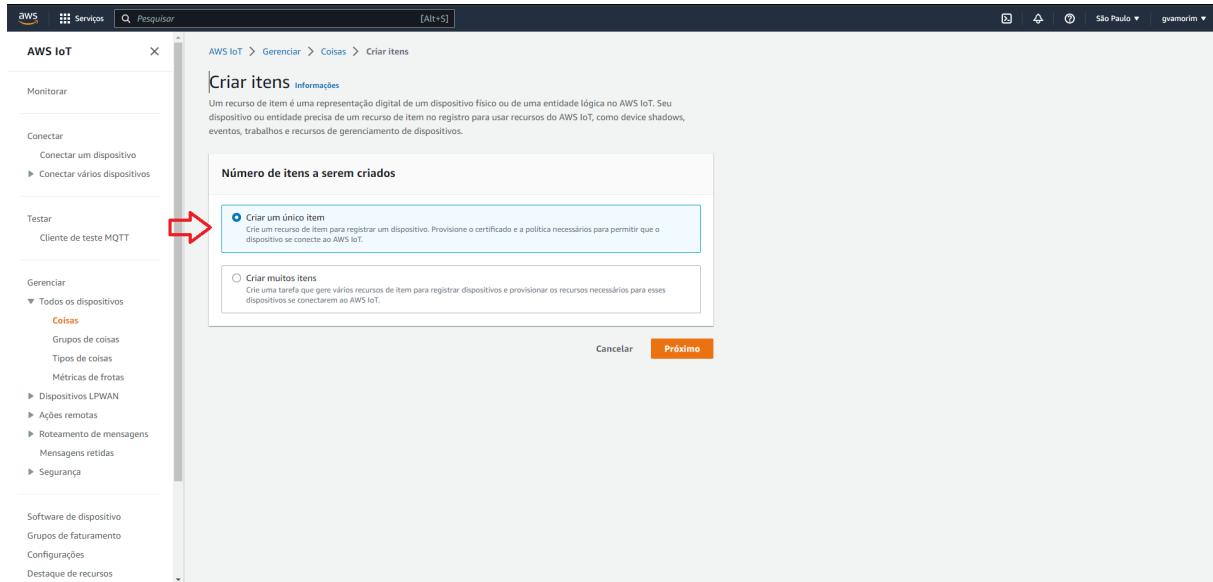
Figura 33 – Criando uma coisa no AWS IoT (B).

This screenshot shows the 'Coisas' (Things) list view. The left sidebar is identical to Figure 32. The main area displays a table with one row, 'Nenhuma coisa', under the 'Nome' column. At the top right of the table, there is a red arrow pointing to the 'Criar itens' (Create Items) button. Other buttons in the header include 'C', 'Pesquisa avançada', 'Executar agregações', 'Editar', and 'Excluir'.

Fonte: Produzido pelo autor (2022).

Selecione a opção *Criar um único item*.

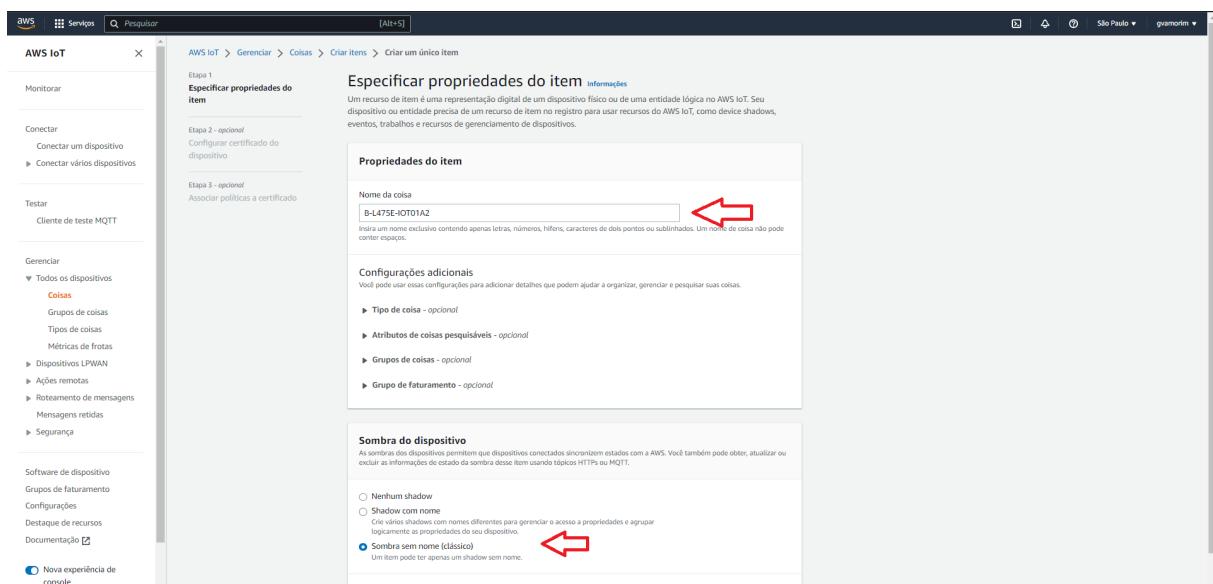
Figura 34 – Criando uma coisa no AWS IoT (C).



Fonte: Produzido pelo autor (2022).

Adicione um nome na opção *Nome da coisa* e selecione *Sombra sem nome (clássico)*.

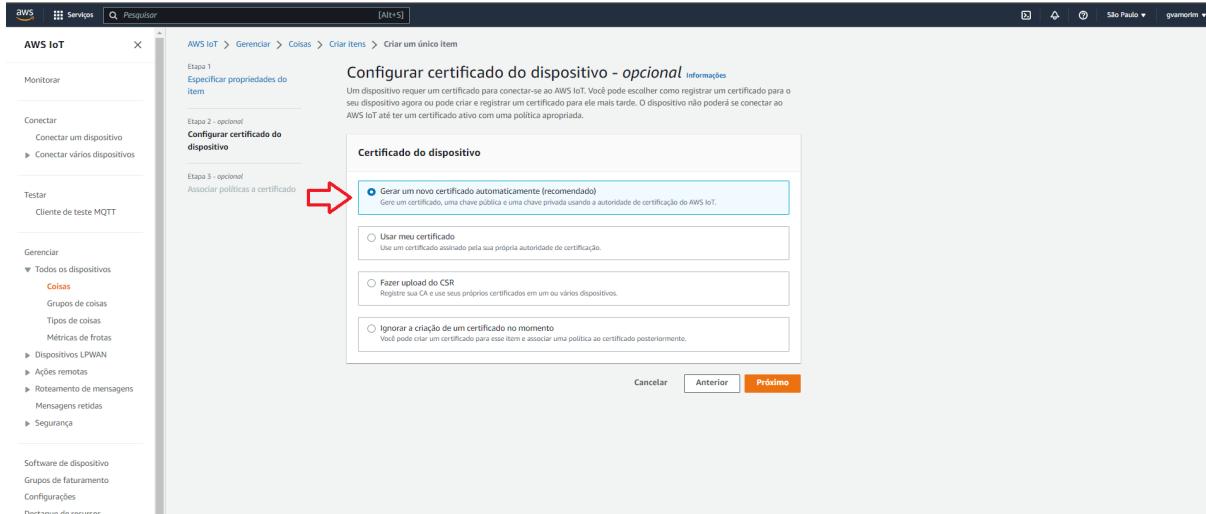
Figura 35 – Criando uma coisa no AWS IoT (D).



Fonte: Produzido pelo autor (2022).

Selecione a opção *Gerar um novo certificado automaticamente (recomendado)*. Um dispositivo requer um certificado para conectar-se ao AWS IoT.

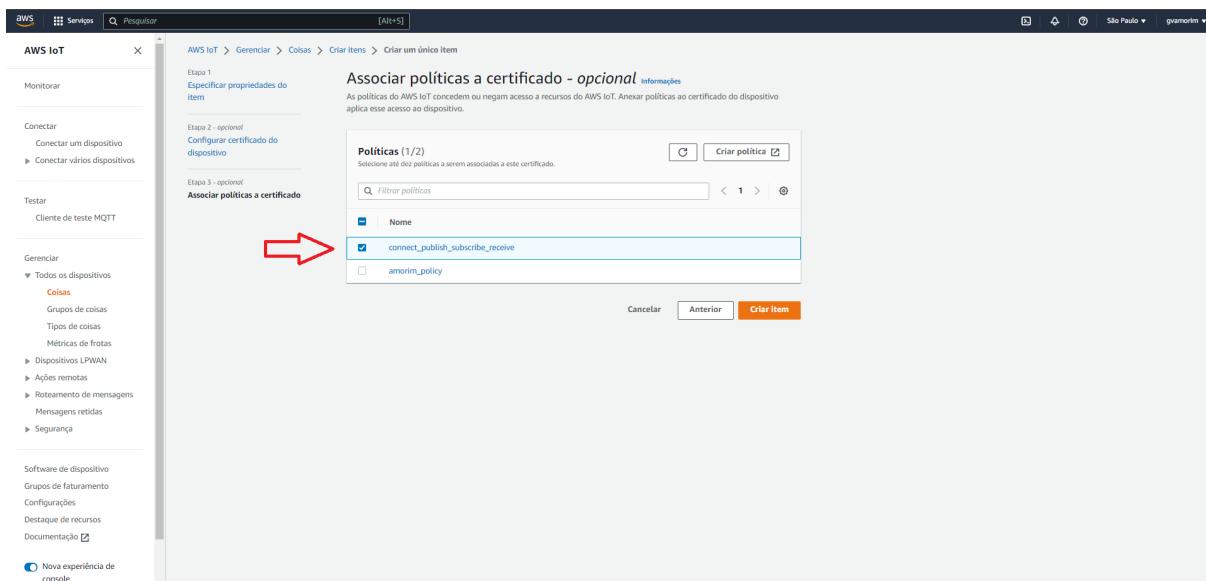
Figura 36 – Criando uma coisa no AWS IoT (E).



Fonte: Produzido pelo autor (2022).

Adicione uma política ao certificado. As políticas do AWS IoT concedem ou negam acesso a recursos do AWS IoT. O processo de criação de uma política para o AWS IoT foi descrito na [seção A.2](#).

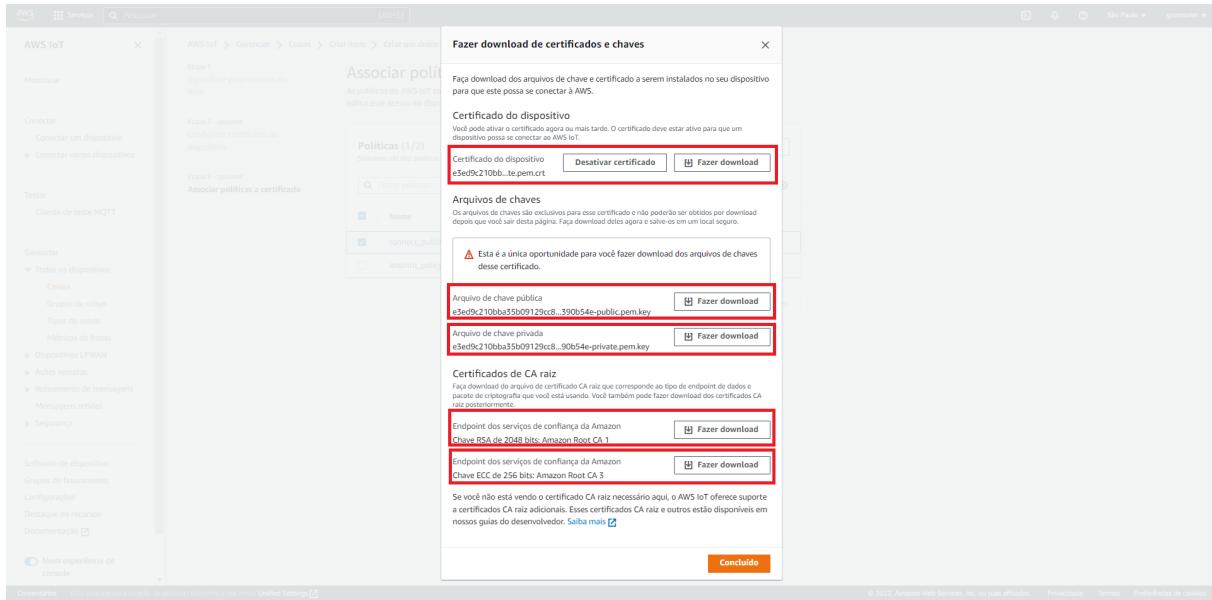
Figura 37 – Criando uma coisa no AWS IoT (F).



Fonte: Produzido pelo autor (2022).

Por fim, a coisa já está pronta no serviço AWS. Faça o *download* dos certificados para que sejam instalados no dispositivo para que este possa se conectar à AWS.

Figura 38 – Criando uma coisa no AWS IoT (G).



Fonte: Produzido pelo autor (2022).

A.4 Configuração do dispositivo B-L475E-IOT01A2 via USB usando o aplicativo Tera Term

SEQUÊNCIA DE CAPTURAS DE TELAS DO PROCESSO DE CONFIGURAÇÃO DO DISPOSITIVO B-L475E-IOT01A2 via USB...