

UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL
UTILIZANDO UMA ASSISTENTE DE VOZ

LEANDRO DALLAROSA NETO

**BLUMENAU
2018**

LEANDRO DALLAROSA NETO

**PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL
UTILIZANDO UMA ASSISTENTE DE VOZ**

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Prof(a). Miguel Alexandre Wisintainer, Mestre - Orientador

**BLUMENAU
2018**

PROTÓTIPO DE AUTOMAÇÃO RESIDENCIAL

UTILIZANDO UMA ASSISTENTE DE VOZ

Por

LEANDRO DALLAROSA NETO

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente: Prof(a). Miguel Alexandre Wisintainer, Mestre – Orientador, FURB

Membro: Prof(a). Marcel Hugo, Mestre – FURB

Membro: Prof(a). Francisco Adell Péricas, Mestre – FURB

Blumenau, 12 de julho de 2018

Dedico este trabalho aos meus pais pelo amor,
carinho e dedicação em mim.

AGRADECIMENTOS

Ao meus pais por terem acreditado e apoiado em mim durante toda a minha vida.

Ao professor Miguel Alexandre Wisintainer pela confiança, apoio e orientação para poder concluir este trabalho.

RESUMO

Este trabalho tem como escopo criar um protótipo de automação residencial por comandos de voz utilizando a assistente pessoal da Amazon Alexa. O protótipo utiliza o microcontrolador Arduino Ameba para se conectar na Internet por meio de uma rede sem fio, controlar as luzes de uma casa, abrir uma porta por meio de um eletroímã e enviar dados de temperatura do ambiente para o servidor HTTP Thingspeak (2018) utilizando o sensor de temperatura e umidade DHT11. Para a execução na parte do servidor é utilizada a plataforma Amazon Web Services (AWS) com a linguagem de programação JavaScript na plataforma Node.js. Na programação do dispositivo embarcado Arduino é utilizada a linguagem de programação C++. A especificação da automação é feita utilizando diagramas da Unified Modeling Language (UML). A execução do protótipo é dada por comandos de voz por meio de um aplicativo de dispositivos móveis. Como resultado, obteve-se uma automação que traz uma maior praticidade para o usuário efetuar atividades rotineiras de uma residência.

Palavras-chave: Amazon Alexa. Arduino Ameba. Automação residencial. Internet das coisas. IOT.

ABSTRACT

This work has as scope to create a home automation prototype by voice commands using the personal assistant Amazon Alexa. The prototype uses the microcontroller Arduino Ameba to connect on Internet through a wireless network, controlling lights of a house, opening a door with an electromagnet and sends the ambient temperature data to the Thingspeak (2018) HTTP server using the sensor temperature and humidity DHT11. For the execution on the server side is used the Amazon Web Services (AWS) platform with JavaScript programming language on Node.js platform. On Arduino embedded device programming is used C++ programming language. The automation specification is made using Unified Modeling Language (UML) diagrams. The execution of prototype is given by voice commands with a mobile device application. As a result, an automation has been obtained more practical activities for user to perform in a residence.

Key-words: Amazon Alexa. Arduino Ameba. Home automation. Internet of things. IOT.

LISTA DE FIGURAS

Figura 1 – Imagem da placa Arduino Ameba.....	11
Figura 2 – Símbolo do AWS Lambda	12
Figura 3 – Imagem demonstrativa de processamento de um comando de voz	13
Figura 4 – Imagem da tela da aplicação de Botke (2014)	14
Figura 5 –Demonstrativo das luzes na aplicação de Beghini (2013)	15
Figura 6 – Estrutura de comunicação do sistema	16
Figura 7 – Página HTML do servidor web.....	16
Figura 8 – Diagrama de casos de uso do usuário	18
Figura 9 – Diagrama de atividades	19
Figura 10 – Diagrama esquemático	20
Figura 11 – Foto do espaço da maquete dedicado ao circuito eletrônico.....	21
Figura 12 – Eletroímã utilizado para trancar a porta	22
Figura 13 – Sensor de temperatura DHT11	23
Figura 14 – Circuito eletrônico completo da maquete	24
Figura 15 – Imagem frontal da maquete.....	25
Figura 16 – Gráfico do histórico da temperatura.....	27
Figura 17 – Tela de <i>intents</i> criados na Amazon Alexa.....	29
Figura 18 – Tela inicial do aplicativo Reverb	31
Figura 19 – Página inicial do site da Amazon Alexa	37
Figura 20 – Abrindo página Alexa Skills Kit.....	37
Figura 21 – Página Alexa Skill Kit.....	38
Figura 22 – Página com as <i>skills</i> já criadas	38
Figura 23 – Dando o nome à <i>skill</i>	39
Figura 24 – Selecionar o modelo Custom	39
Figura 25 – Tela de configuração da <i>skill</i> criada.....	40
Figura 26 – Inserindo um nome de invocação da <i>skill</i>	40
Figura 27 – Adicionando novo <i>intent</i>	41
Figura 28 – Criando um <i>intent</i> personalizável	41
Figura 29 – Inserindo um comando	42
Figura 30 – Criando um tipo de <i>slot</i>	42
Figura 31 – Adicionando valores ao novo <i>slot</i>	43

Figura 32 – Tela com o <i>slot</i> e <i>intent</i> criados	43
Figura 33 – Mensagem depois de clicar em “Build Model”	44
Figura 34 – Página com as funções Lambda	44
Figura 35 – Dando um nome para uma nova função.....	44
Figura 36 – Criando a função	45
Figura 37 – Adicionando o gatilho de <i>skill</i> da Alexa	45
Figura 38 – ID da Skill na Amazon Alexa	46
Figura 39 – Inserir o ID da habilidade.....	46
Figura 40 – Salvar e copiar código ARN gerado.....	47
Figura 41 – Colar código ARN gerado no AWS Lambda (2018)	47
Figura 42 – Definindo as permissões da função.....	48
Figura 43 – Inserindo as permissões.....	48
Figura 44 – Após a inclusão da permissão do AWS IoT (2018).....	49
Figura 45 – Página inicial de AWS IoT (2018).....	50
Figura 46 – Incluindo novo dispositivo.....	50
Figura 47 – Conectando o dispositivo	51
Figura 48 – Como se conectar ao AWS IoT	52
Figura 49 – Dar o nome à nova “coisa”.....	52
Figura 50 – Baixar kit de conexão.....	53
Figura 51 – Clicar para concluir	54
Figura 52 – Concluindo nova “coisa”	54
Figura 53 – Nome ARN	55
Figura 54 – Clicar em “Criar certificado”	55
Figura 55 – Certificado criado.....	56
Figura 56 – Clicando na função Lambda	56
Figura 57 – Projeto exemplo do Arduino com o AWS	58
Figura 58 – Variáveis para conectar-se na rede sem fio.....	59
Figura 59 – Certificado CA	60
Figura 60 – Variáveis que armazenam o certificado e a chave privada	61

LISTA DE QUADROS

Quadro 1 – Código do setup do Arduino.....	25
Quadro 2 – Rotina <code>callback</code>	26
Quadro 3 – Método <code>updatePinState</code>	27
Quadro 4 – Código do método <code>loop</code>	28
Quadro 5 – Código do método de envio da temperatura	28
Quadro 6– Função <code>onIntent</code> no servidor	30
Quadro 7 – Ações e os respectivos comandos.....	31
Quadro 8 - Tabela de tempo de resposta de execução.....	32
Quadro 9 - Comparativo das funcionalidades dos trabalhos correlatos	32
Quadro 10 – Descrição do UC-01 Controlar luzes.....	36
Quadro 11 – Descrição do UC-02 Abrir uma porta	36
Quadro 12 – Descrição do UC-03 Consultar temperatura ambiente	36
Quadro 13 – Código exemplo para modificar permissão da função	49
Quadro 14 – Alterar <i>endpoint</i> no AWS Lambda.....	56
Quadro 15 – Alterar nome do <i>intent</i>	57
Quadro 16 – Substituir “LightState” e “ameba”	57

LISTA DE ABREVIATURAS E SIGLAS

AWS – Amazon Web Services

GPIO – General Purpose Input/Output

HTTP – Hypertext Transfer Protocol

ID – Identity

IOT – Internet of Things

LED – Light Emitting Diode

MDF – Medium Density Fiberboard

MQTT – Message Queuing Telemetry Transport

SD – Secure Digital Card

SSL – Secure Sockets Layer

UML – Unified Modeling Language

USB – Universal Serial Bus

SUMÁRIO

1 INTRODUÇÃO.....	9
1.1 OBJETIVOS	9
1.2 ESTRUTURA.....	9
2 FUNDAMENTAÇÃO TEÓRICA.....	11
2.1 ARDUINO AMEBA	11
2.2 AMAZON WEB SERVICES.....	12
2.2.1 AWS IoT	12
2.3 AMAZON ALEXA	13
2.4 TRABALHOS CORRELATOS	13
2.4.1 AUTOMAÇÃO DE RESIDÊNCIAS ATRAVÉS DE APLICAÇÃO INTEGRADA COM ARDUINO	14
2.4.2 AUTOMAÇÃO RESIDENCIAL DE BAIXO CUSTO POR MEIO DE DISPOSITIVOS MÓVEIS COM SISTEMA OPERACIONAL ANDROID	15
2.4.3 WEBHOME – AUTOMAÇÃO RESIDENCIAL UTILIZANDO RASPBERRY PI	15
3 DESENVOLVIMENTO DA AUTOMAÇÃO	17
3.1 REQUISITOS	17
3.2 ESPECIFICAÇÃO	17
3.2.1 Diagrama de Casos de Uso	17
3.2.2 Diagrama de atividades	18
3.2.3 Diagrama esquemático	19
3.3 IMPLEMENTAÇÃO	20
3.3.1 Técnicas e ferramentas utilizadas.....	20
3.3.2 Operacionalidade da implementação	30
3.4 ANÁLISE DOS RESULTADOS	32
4 CONCLUSÕES.....	33
4.1 EXTENSÕES	33
REFERÊNCIAS	34
APÊNDICE A – DESCRIÇÃO DOS CASOS DE USO	36
APÊNDICE B – CRIANDO UMA SKILL PARA A AMAZON ALEXA	37

1 INTRODUÇÃO

Atualmente as assistentes pessoais estão conosco todo o tempo. Os principais smartphones já vêm de fábrica com suas assistentes pessoais. Como exemplo, para o Google Assistente podem ser feitas perguntas que ele irá procurar na Internet e irá responder. Ou basta mandar enviar uma mensagem de texto ou efetuar uma chamada telefônica que o assistente tratará de executar o pedido. Tudo isso sem as mãos, apenas por comandos de voz (NEVES, 2018). As tecnologias presentes nesses assistentes de voz são capazes de fazerem algo a mais além de responder a previsão do tempo e agendar um horário para o despertador.

A Amazon Alexa, por exemplo, é um serviço de assistente pessoal inteligente na nuvem onde pode-se solicitar tarefas como realizar pesquisas, mandar executar uma lista de músicas ou questionar o horário atual. (VIGLIAROLO, 2017). Segundo o site da Amazon Alexa (2018), o serviço permite conectar-se com dispositivos, e por meio do Web Service da Amazon (AWS), efetuar comandos de voz, interpretá-los e tomar uma ação correspondente.

O Arduino Ameba é uma placa eletrônica para o desenvolvimento de Internet das coisas. Com ele é possível se conectar à Internet via rede sem fio, controlar LEDs, diversos sensores ou se conectar com outros dispositivos. Além disso, com o Ameba é possível conectar com o serviço da Amazon chamado Alexa. O preço elevado de empresas de automação deve-se à ausência de um sistema onde não dependem de uma plataforma única e o Arduino junto à compatibilidade com a Amazon visa suprir essa necessidade.

Diante do que foi dito, este trabalho se propõe a estudar a assistente de voz Alexa e o Arduino Ameba para poder controlar acender luzes, abrir uma porta e consultar a temperatura de uma maquete.

1.1 OBJETIVOS

Este trabalho tem como objetivo controlar periféricos de uma casa automatizada por meio de comando de voz.

Os seus objetivos específicos são:

- a) controlar o estado (ligado/desligado) de uma luz através de comandos de voz;
- b) consultar a temperatura do ambiente por comandos de voz;
- c) abrir uma porta eletrônica utilizando comandos de voz.

1.2 ESTRUTURA

O trabalho está organizado em quatro capítulos. O capítulo 1 apresenta a introdução e os objetivos do trabalho. O capítulo 2 apresenta a fundamentação teórica para contextualizar

este trabalho e termina com os trabalhos correlatos. O capítulo 3 descreve o desenvolvimento do trabalho com os requisitos, as ferramentas utilizadas, a especificação, a implementação, a operacionalidade do protótipo e a análise dos resultados. O capítulo 4 finaliza com a conclusão do trabalho junto das vantagens e limitações do protótipo, e as sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

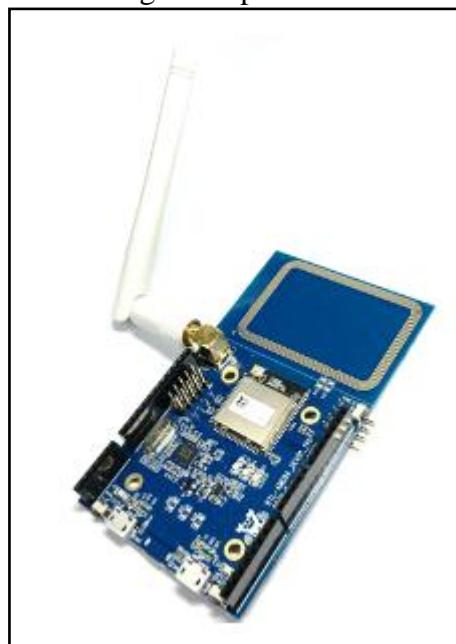
O atual capítulo apresenta os principais conceitos para o desenvolvimento deste trabalho. Este capítulo está subdividido em 5 seções, onde a seção 2.1 apresentará o microcontrolador Arduino Ameba. A seção 2.2 apresentará o a Amazon Web Services (AWS). A seção 2.3 apresentará a Amazon Alexa. E por fim, a seção 2.4 tratará sobre os trabalhos correlatos.

2.1 ARDUINO AMEBA

O Arduino é uma plataforma de hardware e software livre com o propósito de criar dispositivos para interagir com o ambiente de modo fácil e rápido. O diferencial da plataforma Arduino é a vasta documentação, as entradas analógicas ou digitais, e a grande variedade de extensões, as chamadas Shields. Estas Shields fornecem novas utilidades caso o modelo já não tenha embutido, como por exemplo: módulo de rede sem fio, Ethernet, Bluetooth, entrada USB ou SD. (BEGHINI, 2013).

O modelo Arduino Ameba já conta com módulo de rede sem fio embutido, como pode ser visto na Figura 1 (REALTEK IOT/ARDUINO SOLUTION, 2018). Ele é principalmente conhecido pela compatibilidade de projetos com o serviço Amazon Alexa. Com ele, o Ameba, pode-se facilmente se conectar à Internet para acessar o Alexa.

Figura 1 – Imagem da placa Arduino Ameba



Fonte: Realtek Iot/Arduino Solution (2018).

O Ameba conta com uma CPU de arquitetura Arm de 32 bits e implementa em hardware o protocolo de segurança SSL. Além de possuir conexão à rede sem fio, possui um módulo de

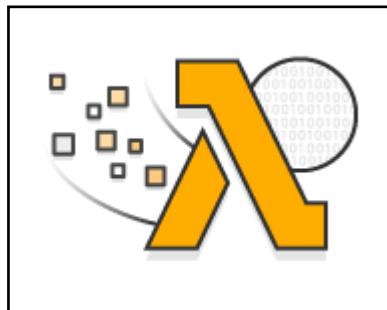
Tag NFC (REALTEK IOT/ARDUINO SOLUTION, 2018d). Este microcontrolador pode ser comprado pelo preço de US\$58,34, incluindo taxas e frete.

2.2 AMAZON WEB SERVICES

A Amazon Web Services (AWS) é uma plataforma de serviços de computação em nuvem prestando soluções para empresas, como processamento e armazenamento de dados (AWS, 2018). Através de uma forma simples oferece serviços via Internet, onde ela faz gerenciamento e a manutenção do hardware que está conectado à rede, enquanto o cliente apenas se dedica em gerenciar seu negócio por meio de uma aplicação web (AWS, 2018b).

O AWS Lambda da Amazon é uma plataforma que permite que o usuário, ou cliente, pague apenas o que consumir. Disponibiliza processamento sempre que preciso, ou até quando agendado que seja executado automaticamente. (AWS LAMBDA, 2018b). A Figura 2 mostra o símbolo do serviço Lambda.

Figura 2 – Símbolo do AWS Lambda



Fonte: AWS Lambda (2018).

Com a alta disponibilidade, o usuário apenas fornece seu código, ou cria a sua Função do Lambda, como é chamado. Cada função é gerenciada pela Amazon, desde a disponibilidade de execução e segurança. Com o AWS Lambda pode-se conectar com outros serviços da Amazon, como o AWS IoT (2018) e a Amazon Alexa (AWS LAMBDA, 2018b).

2.2.1 AWS IoT

A função da Amazon voltada para a Internet das Coisas (AWS IOT) permite conectar dispositivos à Internet para que se possa transmitir dados para serem armazenados e analisados. Muitos tipos de objetos são usados nas aplicações para Internet das Coisas, desde câmeras de segurança a refrigeradores. Seja qual for o dispositivo desde que possa ser ligado, pode fazer parte do IoT. Essas aplicações tornaram dados antigamente inúteis em dados importantes para os usuários (AWS IOT ,2018b).

A Amazon oferece uma plataforma de Internet of Things (Internet das Coisas) onde gerencia-se dispositivos na nuvem facilmente. Pode-se criar aplicações que processam ou

analism informações de dispositivos conectados sem necessitar que o cliente tenha uma infraestrutura. O AWS IoT dá suporte ao protocolo HTTP e MQTT, que visa reduzir consumo de dados na rede (IOT CORE, 2018).

2.3 AMAZON ALEXA

A Amazon Alexa é um serviço de voz na nuvem de uma assistente pessoal inteligente. Este serviço recebe o áudio de um dispositivo, passa pelo reconhecimento de voz e interpreta os comandos de voz efetuados (HAACK, 2017). Por exemplo, pode-se solicitar alguma informação da Internet como a previsão do tempo ou até mesmo dar um comando para efetuar uma habilidade.

Essas habilidades são chamadas de skills. Uma skill é uma tarefa personalizada para cada aplicação. Após efetuado um comando de voz, ele é interpretado e é executado por servidores da própria Amazon, onde são executados conforme programados. Esses servidores são chamados de AWS.

A Figura 3 mostra a sequência de passos do processamento de um comando de voz de uma skill. Ao realizar um comando para a Alexa, é enviado o comando de voz interpretado em forma de JSON para o AWS Lambda. De acordo com o comando em JSON, uma mensagem é enviada para o servidor AWS IoT que realiza de fato o comando. Por fim o IoT retorna para o servidor Lambda, que em seguida retorna com a resposta da execução para a Alexa. (REALTEK IOT/ARDUINO SOLUTION, 2018b).

Figura 3 – Imagem demonstrativa de processamento de um comando de voz



Fonte: Realtek IoT/Arduino Solution (2018b).

2.4 TRABALHOS CORRELATOS

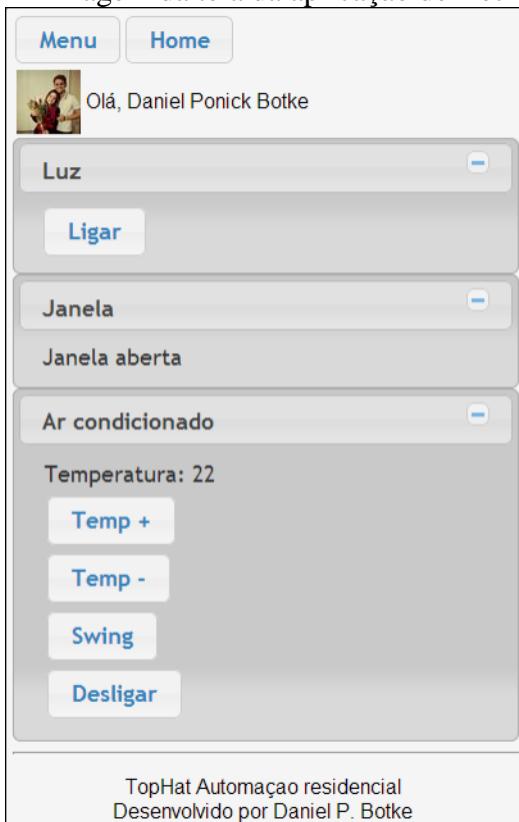
Neste tópico é feito o comparativo de três trabalhos relacionados ao tema proposto. No trabalho de Botke (2014) foi desenvolvida uma aplicação web com autenticação de usuário para monitorar a janela, ligar e desligar luzes e controlar o condicionador de ar. No trabalho de Beghini (2013) foi desenvolvida uma aplicação Android que controla luzes, a refeição de animal doméstico e o alarme da casa. No projeto de Lisboa et al (2014) desenvolveu-se sistemas

Android e web para controlar luzes, consultar temperatura do ambiente e verificar se a porta está aberta ou fechada.

2.4.1 AUTOMAÇÃO DE RESIDÊNCIAS ATRAVÉS DE APLICAÇÃO INTEGRADA COM ARDUINO

No trabalho de Botke (2014) utilizou-se o microcontrolador Arduino Mega e uma interface Ethernet. Foi feita uma aplicação web onde o usuário passa por uma autenticação de usuário. Esta aplicação permite consultar o estado da luz (ligada ou desligada), acender ou desligar a luz, verificar se a janela está aberta e até mesmo controlar o condicionador de ar, conforme a Figura 4.

Figura 4 – Imagem da tela da aplicação de Botke (2014)



Fonte: Botke (2014, p. 54).

Para o controle da residência, o projeto tem um componente emissor infravermelho, para controlar o condicionador de ar, dois relê para ligar ou desligar as luzes e um sensor de contato para verificar o estado da janela (aberta ou fechada).

Por ser uma aplicação web, ela tem como vantagem a diversidade de plataforma, podendo ser utilizada em sistema Android ou IOS. A partir da autenticação de usuário, as informações de uso da aplicação são salvas em um banco de dados.

2.4.2 AUTOMAÇÃO RESIDENCIAL DE BAIXO CUSTO POR MEIO DE DISPOSITIVOS MÓVEIS COM SISTEMA OPERACIONAL ANDROID

Já no trabalho de Beghini (2013) foi utilizado Arduino Uno e módulo Ethernet. Com este equipamento, a aplicação controla luzes, aciona o alarme e controle da alimentação dos animais. Ao contrário do anterior, este possui um aplicativo desenvolvido unicamente para Android.

Conforme a Figura 5, cada botão inserido controla o estado de um objeto na casa.

Figura 5 –Demonstrativo das luzes na aplicação de Beghini (2013)



Fonte: Beghini (2013, p. 55).

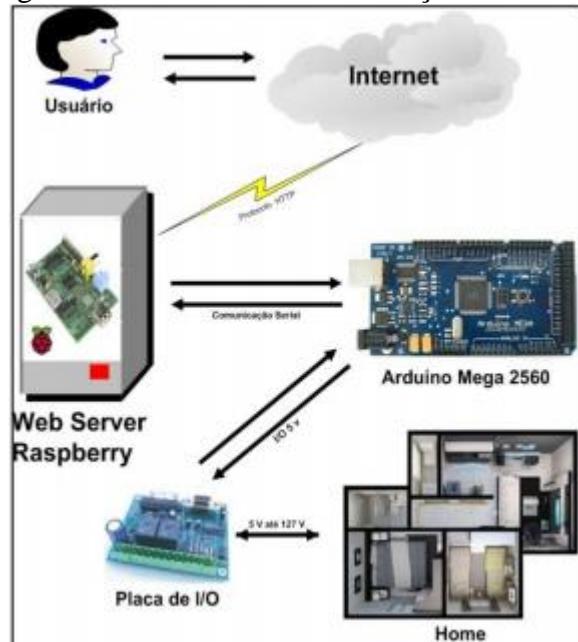
Antes de ter acesso às funcionalidades da aplicação, é necessário inserir uma senha. Visto que foi desenvolvida para dispositivos móveis e há uma possibilidade da perda ou furto do dispositivo, é importante passar por uma autenticação, pois pessoas não autorizadas poderiam ter acesso à residência sem autorização.

2.4.3 WEBHOME – AUTOMAÇÃO RESIDENCIAL UTILIZANDO RASPBERRY PI

O trabalho de Lisboa et al (2014) consiste em um sistema que controle uma residência por meio de um dispositivo móvel que utilize o sistema operacional Android e possua acesso à Internet ou à rede sem fio. A aplicação é capaz de monitorar a temperatura, estado das portas e alterar o estado das lâmpadas.

O sistema constitui-se de um dispositivo móvel com Android, um computador Raspberry PI, um microcontrolador Arduino Mega, uma placa de I/O, sensores e LEDs. Como pode ser visto na Figura 6, o usuário toma a ação através da aplicação Android que por sua vez envia a requisição para o servidor web no Raspberry via Internet. O servidor se encarrega de solicitar ao Arduino acionar a placa de I/O fazer a leitura dos sensores ou acionar os LEDs.

Figura 6 – Estrutura de comunicação do sistema



Fonte: Lisboa et al (2014).

Os resultados dessa aplicação podem ser vistos na Figura 7 que mostra a página HTML de monitoramento no servidor web. Notam-se os estados das lâmpadas, o estado da porta (aberta ou fechada) e a temperatura, onde é acionada uma ventoinha caso o sensor de temperatura atinja o valor estipulado previamente.

Figura 7 – Página HTML do servidor web

AMBIENTE	STATUS
O QUARTO - 01	DESLIGAR
O COZINHA	DESLIGAR
O SALA	LIGAR
O CORREDOR	LIGAR
O BANHEIRO	LIGAR
O QUARTO - 02	LIGAR

STATUS PORTA: FECHADA!! Temperatura: 24.8 oC

Fonte: Lisboa et al (2014).

3 DESENVOLVIMENTO DA AUTOMAÇÃO

No presente capítulo são descritas as etapas para o desenvolvimento deste projeto e é abordado resumidamente as ferramentas utilizadas. Na seção 3.1 contém os requisitos funcionais e não funcionais e na seção 3.2 é apresentada a especificação detalhada do funcionamento do protótipo. A seção 3.3 tem detalhes da implementação e a montagem da maquete de demonstração e na seção 3.4 é feita uma análise dos resultados obtidos com o protótipo.

3.1 REQUISITOS

Os requisitos do protótipo desenvolvido são:

- a) a aplicação deverá permitir que o usuário acenda e apague uma luz da residência por comando de voz (Requisito Funcional – RF);
- b) a aplicação deverá permitir que o usuário consulte o estado atual da luz (ligada ou apagada) por comando de voz (RF);
- c) a aplicação deverá permitir consultar a temperatura do ambiente por comando de voz (RF);
- d) a aplicação deverá permitir abrir uma porta eletrônica por comando de voz (RF);
- e) a aplicação deverá ser desenvolvida na plataforma do Arduino Ameba (Requisito Não-Funcional – RNF);
- f) a aplicação deverá utilizar a plataforma da Amazon Web Services (AWS) (RNF).

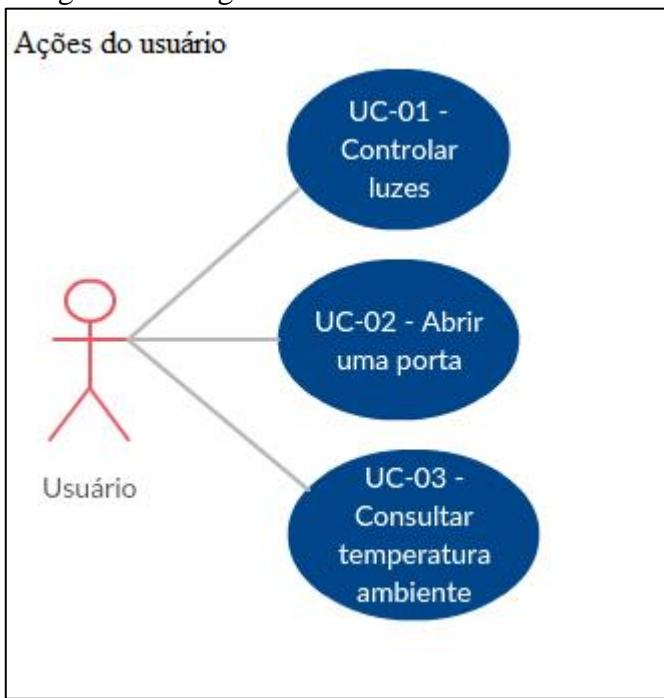
3.2 ESPECIFICAÇÃO

Para especificação deste projeto foram desenvolvidos: os diagramas de casos de usos, o diagrama de atividades e o diagrama esquemático. Para elaboração do diagrama de casos de usos e de atividades foi utilizada a ferramenta Creately e para o diagrama esquemático a ferramenta Fritzing.

3.2.1 Diagrama de Casos de Uso

Nesta subseção é apresentado o diagrama de casos de uso que mostra as funcionalidades da aplicação feita pelo usuário. A explicação do diagrama de casos de uso está no Apêndice A. A Figura 8 mostra o diagrama de casos de uso das funcionalidades do protótipo.

Figura 8 – Diagrama de casos de uso do usuário



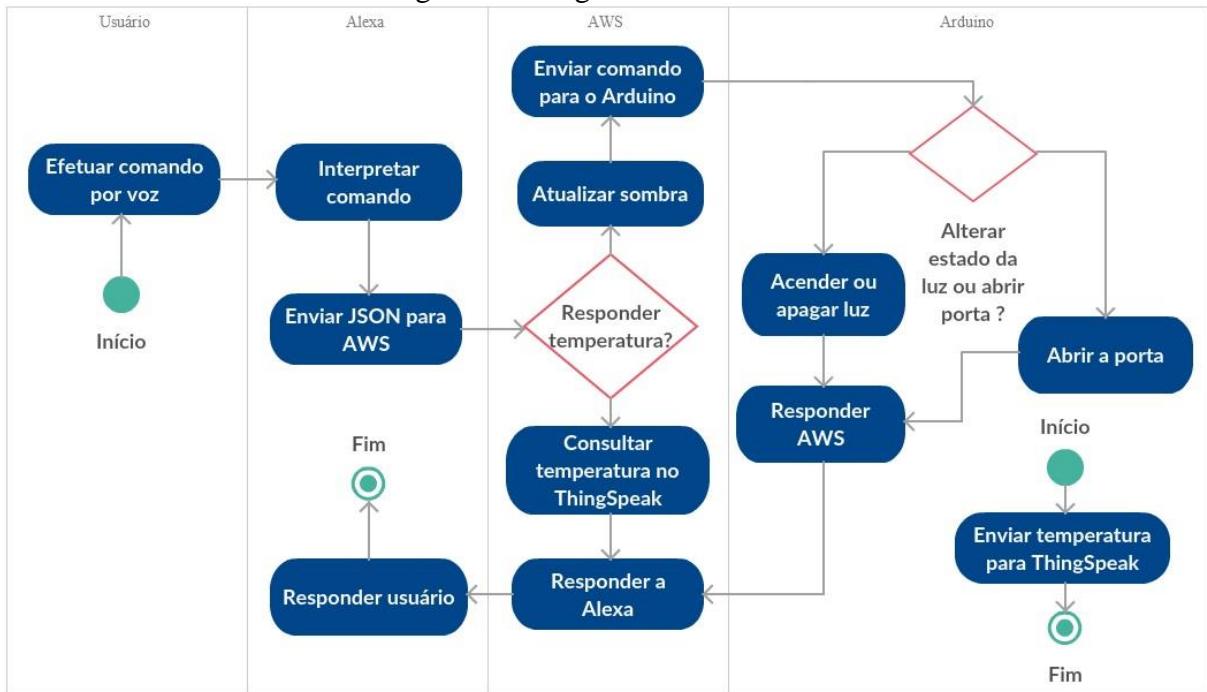
Fonte: elaborado pelo autor.

No caso de uso UC-01 o ator efetua um comando de voz para acender ou apagar as luzes da maquete. O UC-02 representa o comando abrir uma porta e o UC-03 o de consultar a temperatura ambiente.

3.2.2 Diagrama de atividades

O diagrama de atividades da Figura 9 apresenta os fluxos de execução do protótipo de acordo com as ações apresentadas anteriormente. O processo inicia-se pelo comando de voz efetuado pelo usuário. A Alexa sintetiza o comando em formato JSON para o servidor AWS tomar a ação de acordo com o comando. Se a ação solicitada for a temperatura, o AWS consulta a temperatura no servidor ThingSpeak e retorna para a Alexa com o valor da temperatura. Se for um comando para alterar o estado da luz ou abrir a porta, o AWS envia a requisição para o Arduino efetuar a ação correspondente.

Figura 9 – Diagrama de atividades



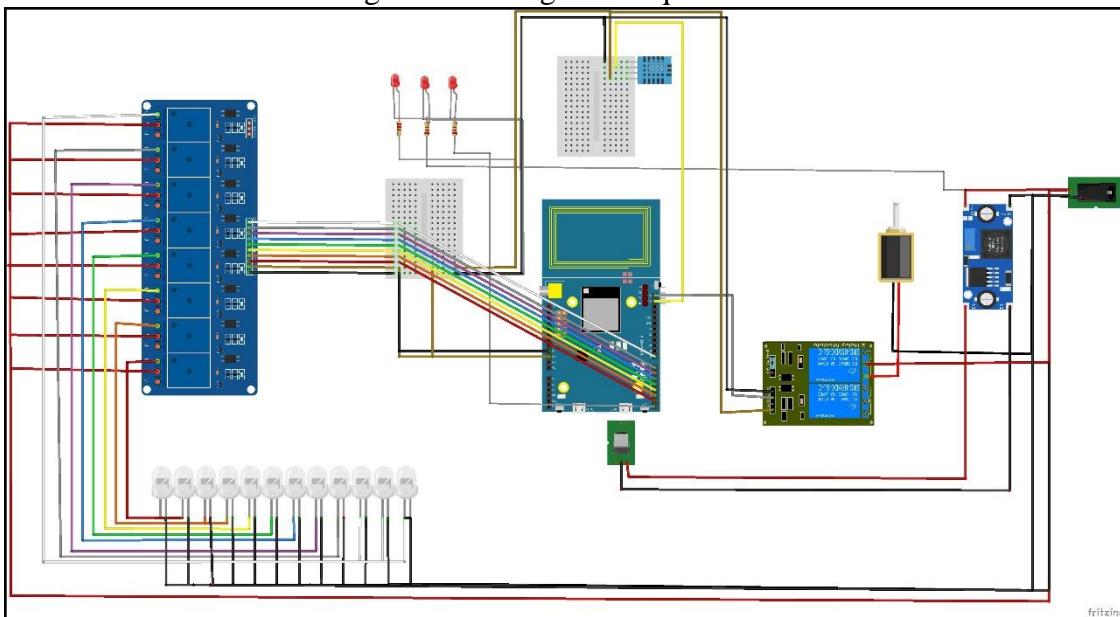
Fonte: elaborado pelo autor.

Além disso, constantemente o Arduino efetua a ação de enviar ao servidor ThingSpeak a temperatura, sem necessidade de uma solicitação do AWS.

3.2.3 Diagrama esquemático

Foi desenhado todo o circuito para o protótipo na Figura 10 utilizando a ferramenta Fritzing. O diagrama mostra o conector DC fêmea como a fonte de energia para o circuito. Ligado diretamente a ele tem um LED indicativo de corrente elétrica. Esta mesma corrente elétrica é associada ao regulador de tensão, ao eletroímã e aos módulos de relés. O LED ligado na GPIO 0 do Arduino indica, quando aceso, que está conectado em uma rede sem fio, e quando apagado, que não está conectado em uma rede. Já o LED ligado nos 5V do Arduino indica se o microcontrolador está recebendo energia. Demais descrições do circuito são indicados na seção 3.3.

Figura 10 – Diagrama esquemático



Fonte: elaborado pelo autor.

3.3 IMPLEMENTAÇÃO

Em seguida são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

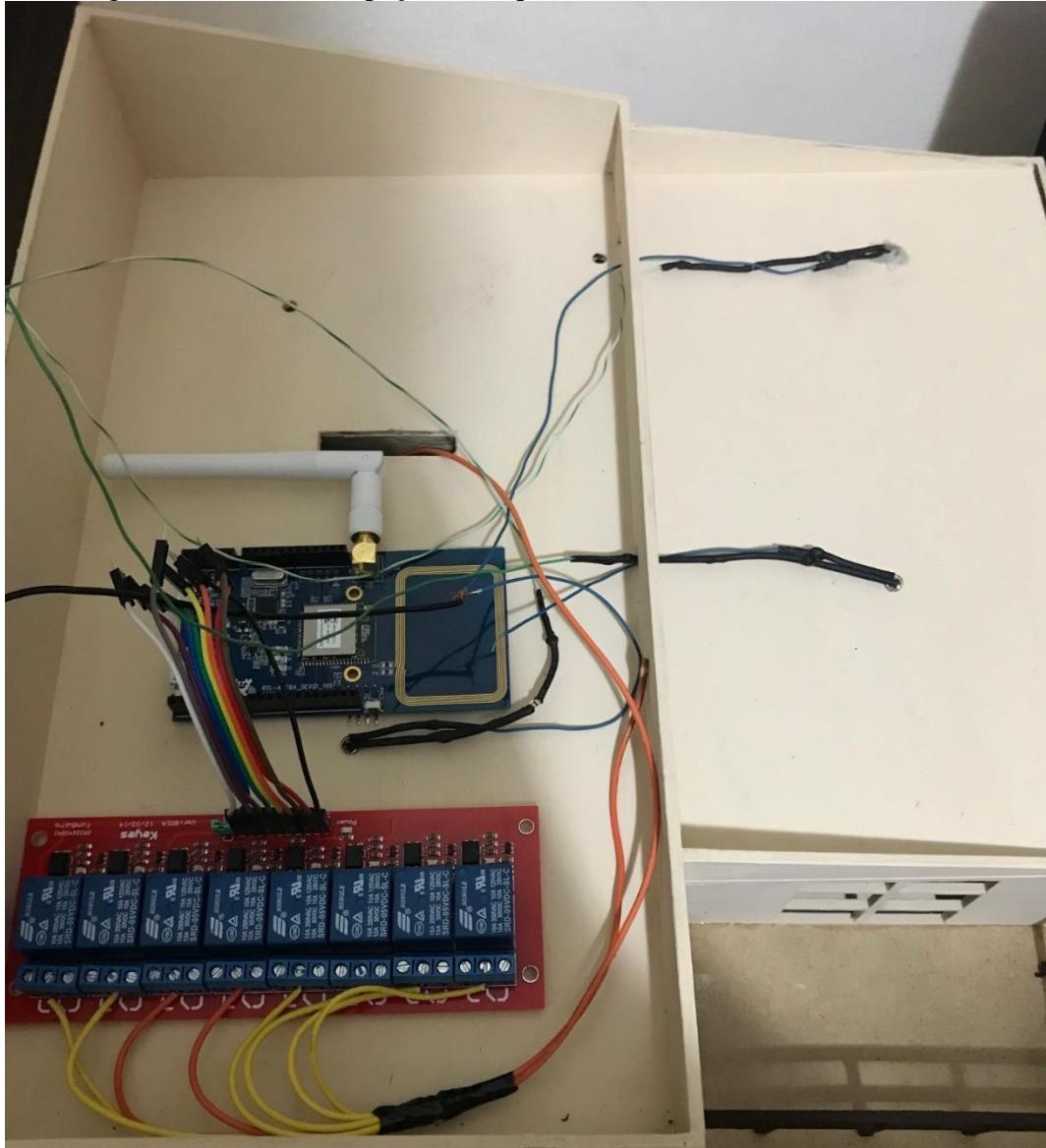
A aplicação embarcada foi desenvolvida utilizando a linguagem de programação C++ utilizando o ambiente de desenvolvimento Arduino Genuino. Para o desenvolvimento da aplicação no Amazon Web Services (AWS) foi utilizada a linguagem JavaScript no ambiente de desenvolvimento no próprio site do AWS que conta com a plataforma Node.js. O desenho do circuito eletrônico para a maquete foi elaborado na ferramenta Fritzing. Para os códigos de programação foi utilizado o editor de texto Notepad++.

No Apêndice B foi criado um passo-a-passo de exemplo de como criar e configurar uma *skill* na Amazon e configurar o Arduino Ameba.

3.3.1.1 Preparação do circuito eletrônico para a maquete

Para a demonstração do trabalho foi utilizada uma maquete feita em MDF de dimensões de 35 centímetros de comprimento por 33 centímetros de largura por 28 centímetros de altura. A maquete conta com 12 LEDs que simulam lâmpadas residenciais em oito diferentes ambientes. Na parte superior tem um espaço de até 6,5 centímetros de altura numa área de 1.193 centímetros quadrados suficientes para alojar todo o circuito eletrônico. Na Figura 11 pode-se visualizar o Arduino e o módulo relé no início do projeto.

Figura 11 – Foto do espaço da maquete dedicado ao circuito eletrônico



Fonte: elaborado pelo autor.

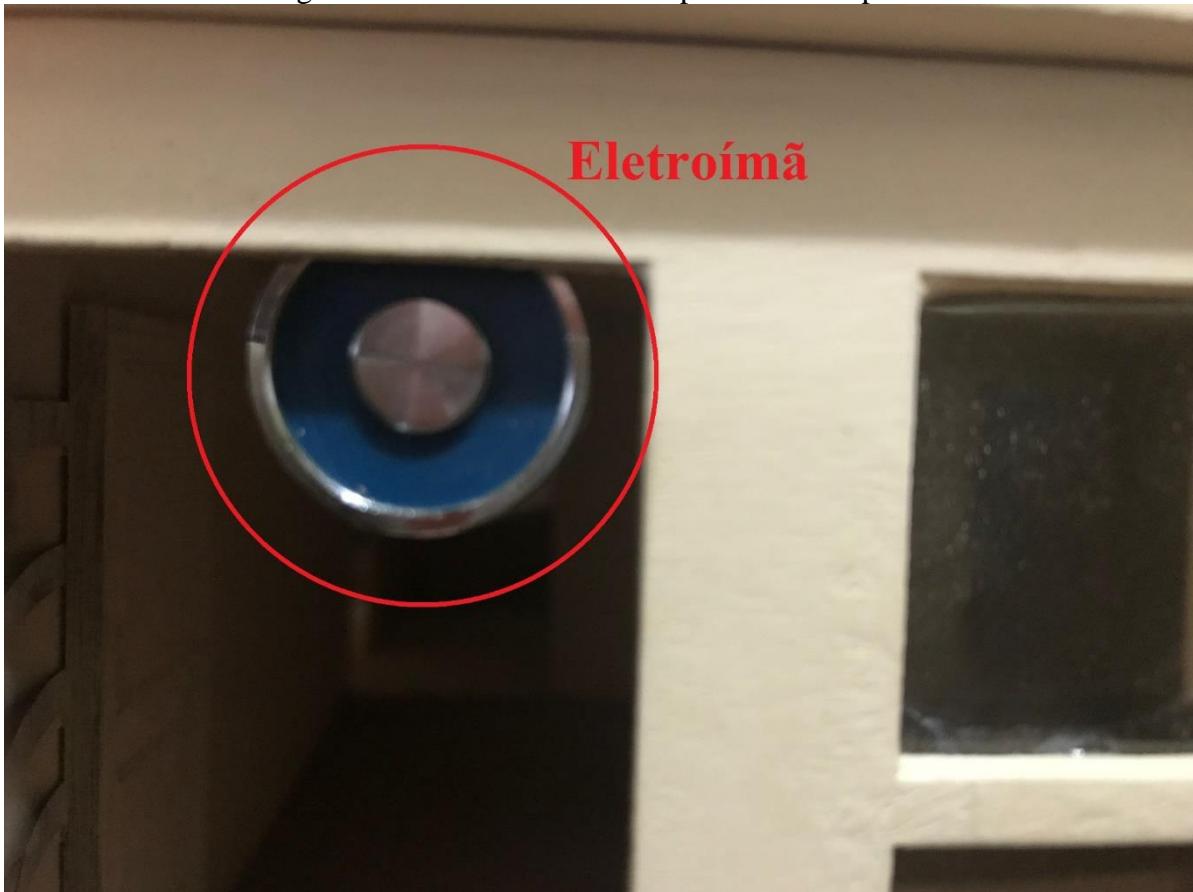
Para simular a casa automatizada foi utilizado o microcontrolador Arduino Ameba, dois módulos relé para Arduino, um regulador de tensão e um eletroímã de 12V. Novamente na Figura 11, pode ser visto o conector DC P4 fêmea por onde é alimentado todo o circuito.

Para os acionamentos das lâmpadas foi utilizado um módulo de relés de oito canais. Com este módulo é possível acionar até 250V de tensão elétrica, suficientes para ligar uma lâmpada comum de uma residência. Os oito canais deste módulo foram ligados nas GPIO 1 a 8 do Arduino. Os LEDs estão ligados numa rede de 12V mantidos por uma fonte elétrica. Os relés são acionados pelas saídas de 5V do Arduino que ligam e desligam as lâmpadas pelo circuito de 12V.

Para a simulação da abertura de uma porta foi usado o eletroímã de 12V. Este eletroímã fica ligado no contato NA (normalmente aberto) do módulo relé de 2 canais. Portanto, ao ligar

a fonte de 12V, o eletroímã já é acionado, onde mantém a porta da casa fechada. Para abrir a porta, o relé fica conectado na GPIO 15 do Arduino que, ao ser acionada, o relé fecha o contato NF (normalmente fechado), cortando energia do eletroímã, abrindo, assim, a porta. O eletroímã instalado na maquete pode ser visto na Figura 12.

Figura 12 – Eletroímã utilizado para trancar a porta

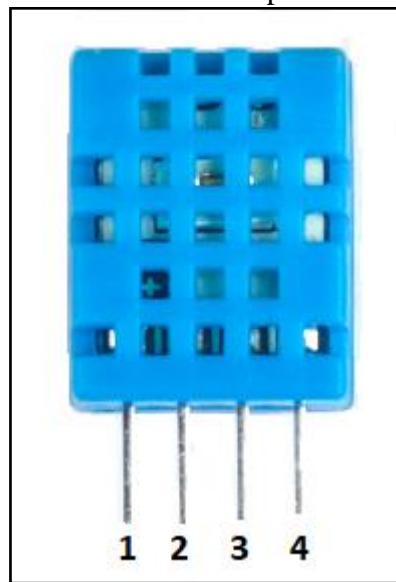


Fonte: elaborado pelo autor.

Uma dificuldade para efetuar a abertura da porta via eletroímã foi a disponibilidade de um componente desse. Não foi encontrado em lojas de componentes eletrônicos na região de Blumenau, sendo necessário encomendar em anúncio na Internet. Portanto, foi investido em frete de Minas Gerais mais o que o próprio valor do componente, somando R\$59,60.

Atendendo outro objetivo do trabalho, o sensor de temperatura DHT11 foi ligado no circuito. Na Figura 13 pode-se ver o sensor e a sequência de pinos. O pino 1 do sensor é ligado em 5V, o pino 2 que é utilizado para dados, é ligado na GPIO 14 do Arduino e o pino 4 é ligado em GND.

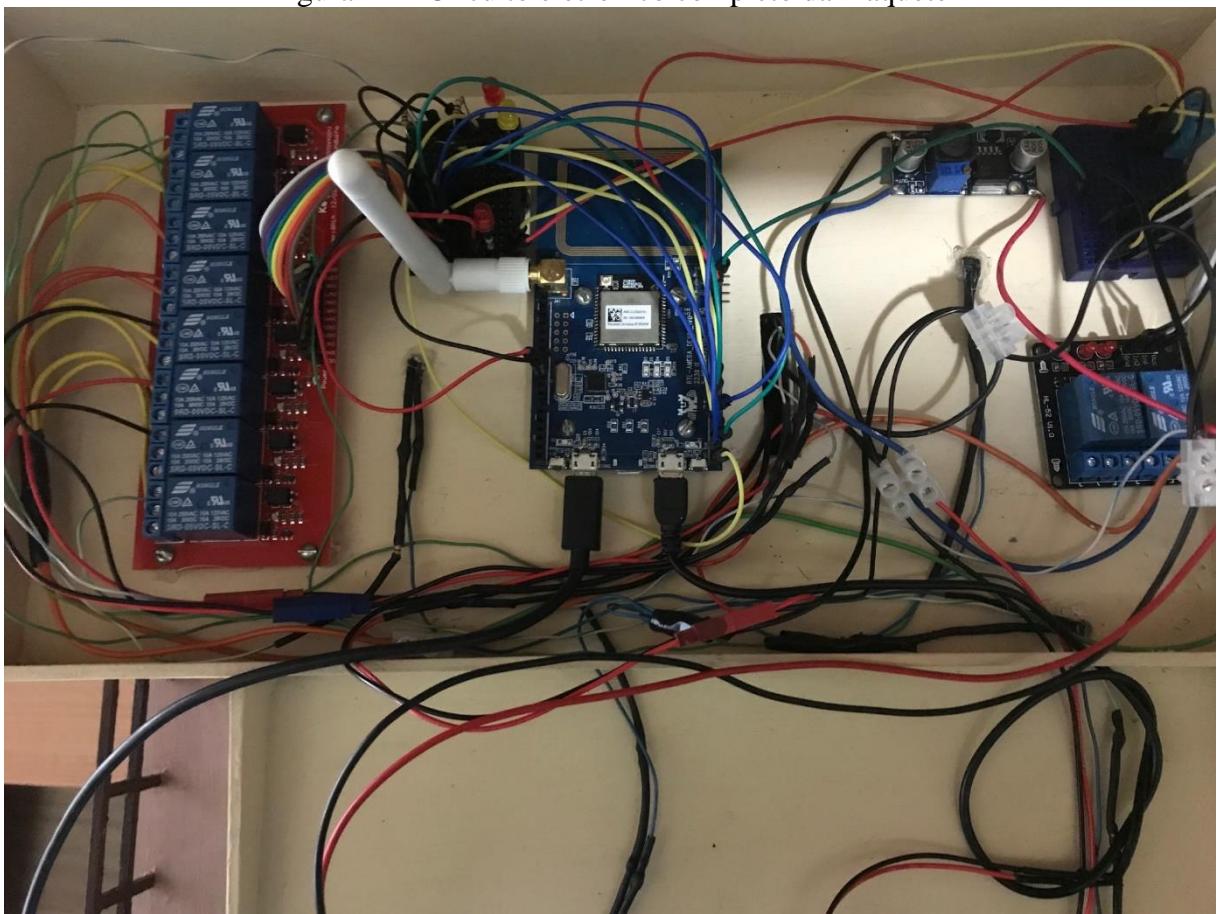
Figura 13 – Sensor de temperatura DHT11



Fonte: Elaborado pelo autor.

Para alimentar o Arduino foi utilizado um regulador de tensão. O papel dele neste projeto foi converter os 12V vindo da fonte de alimentação de energia para 5V para alimentar o microcontrolador. Logo, ao invés de utilizar duas fontes de alimentação, é possível utilizar uma só, e deixar o USB do Arduino como opção. Na Figura 14 pode-se ver como ficou todo o circuito pronto dentro da maquete de demonstração.

Figura 14 – Circuito eletrônico completo da maquete



Fonte: elaborado pelo autor.

Na Figura 15 vê-se o resultado da maquete pintada e pronta. Os LEDs estão acesos e a porta fechada. Com o eletroímã ligado, tem-se que fechar a porta, pois ela possui um conjunto de molas que fazem abrir a porta assim que liberar a corrente elétrica do eletroíma.

Figura 15 – Imagem frontal da maquete



Fonte: elaborado pelo autor.

3.3.1.2 Programando o Arduino Ameba

O ponto de entrada no Arduino resume-se nos métodos `setup` e `loop`. O `setup` é executado sempre que o Arduino é iniciado. O `loop` é um laço que é constantemente executado após ele mesmo terminar. No Quadro 1 mostra parte do código do método `setup` adaptado de Realtek IoT (2018) onde o Arduino tenta se conectar na rede sem fio.

Quadro 1 – Código do setup do Arduino

```

363 while (statusWifi != WL_CONNECTED) {
364     Serial.print("Tentando se conectar na rede ");
365     Serial.println(ssid);
366
367     statusWifi = WiFi.begin(ssid, pass);
368
369     if (statusWifi == WL_CONNECTED) {
370         break;
371     }
372
373     delay(1000); // Tentar novamente em um segundo.
374 }
375
376 wifiClient.setRootCA((unsigned char*)rootCABuff);
377 wifiClient.setClientCertificate((unsigned char*)certificateBuff, (unsigned char*)privateKeyBuff);
378
379 client.setServer(mqttServer, 8883);
380 client.setCallback(callback);
381 delay(1500);
382
383 dht.begin();
384 int temperature = (int) dht.readTemperature();
385 sendTemperature(temperature);
386 printf("Temperatura publicada: %d.\r\n", temperature);

```

Fonte: Elaborado pelo autor.

A biblioteca DHT para leitura do sensor de temperatura é muito simples. Pode-se ver nas linha 383 e 384 do quadro anterior que o sensor de temperatura é inicializado com o método `dht.begin` e na linha 384 é obtida a temperatura com `dht.readTemperature`. Assim, nessas duas linhas já se tem a temperatura para ser enviada para o servidor HTTP.

No Quadro 2 apresenta a rotina de callback. Esta rotina é chamada pelo servidor e é responsável por toda ação que o servidor solicita que o Arduino Ameba execute. Na linha 271 vemos o método `updatePinState`. É para este método que é passado o status e o *intent*, ou seja, a ação criada para ser interpretada pelo comando de voz, e seja executado.

Quadro 2 – Rotina callback

```

248 void callback(char* topic, byte* payload, unsigned int length) {
249     char buf[MQTT_MAX_PACKET_SIZE];
250     char *pch;
251     int desired_pin_state;
252     char desired_pin_output[20];
253
254     strncpy(buf, (const char *)payload, length);
255     buf[length] = '\0';
256     printf("Message arrived [%s] %s\r\n", topic, buf);
257
258     if ((strstr(topic, "/shadow/get/accepted") != NULL) || (strstr(topic, "/shadow/update/accepted") != NULL)) {
259         pch = strstr(buf, "\"desired\":{\"status\":\"");
260         if (pch != NULL) {
261             pch += strlen("\"desired\":{\"status\":\"");
262             desired_pin_state = *pch - '0';
263             pch += strlen("\"output\":\"") + 2;
264
265             strncpy(desired_pin_output, pch, length);
266             desired_pin_output[19] = '\0';
267
268             printf("ponteiro: [%s] \r\n", desired_pin_output);
269             printf("Pino acionado: " + desired_pin_state);
270
271             updatePinState(desired_pin_output, desired_pin_state);
272
273         }
274     }
275 }
```

Fonte: elaborado pelo autor.

Por exemplo, se o servidor solicitar que seja ligada ou desligada uma luz do “*IntentOne*”, é feita chamada do método `updatePinState` passando o nome do *intent* e o seu status, que será alterado pelo método `digitalWrite` da linha 165 do Quadro 3.

Quadro 3 – Método updatePinState

```

160 void updatePinState(char* desired_pin_output, int desired_pin_state) {
161   printf("change %s to %d\r\n", desired_pin_output, desired_pin_state);
162
163   if(strstr(desired_pin_output, "IntentOne") != NULL) {
164     pin_1_state = desired_pin_state;
165     digitalWrite(pin_1, pin_1_state);
166     printf("Saída um = %s\r\n", desired_pin_output);
167   } else if(strstr(desired_pin_output, "IntentTwo") != NULL) {
168     pin_2_state = desired_pin_state;
169     digitalWrite(pin_2, pin_2_state);
170     printf("Saída dois = %s\r\n", desired_pin_output);
171   } else if(strstr(desired_pin_output, "IntentThree") != NULL) {
172     pin_3_state = desired_pin_state;
173     digitalWrite(pin_3, pin_3_state);
174     printf("Saída três = %s\r\n", desired_pin_output);
175   } else if(strstr(desired_pin_output, "IntentFour") != NULL) {
176     pin_4_state = desired_pin_state;
177     digitalWrite(pin_4, pin_4_state);
178     printf("Saída quatro = %s\r\n", desired_pin_output);
179   } else if(strstr(desired_pin_output, "IntentFive") != NULL) {
180     pin_5_state = desired_pin_state;
181     digitalWrite(pin_5, pin_5_state);
182     printf("Saída cinco = %s\r\n", desired_pin_output);

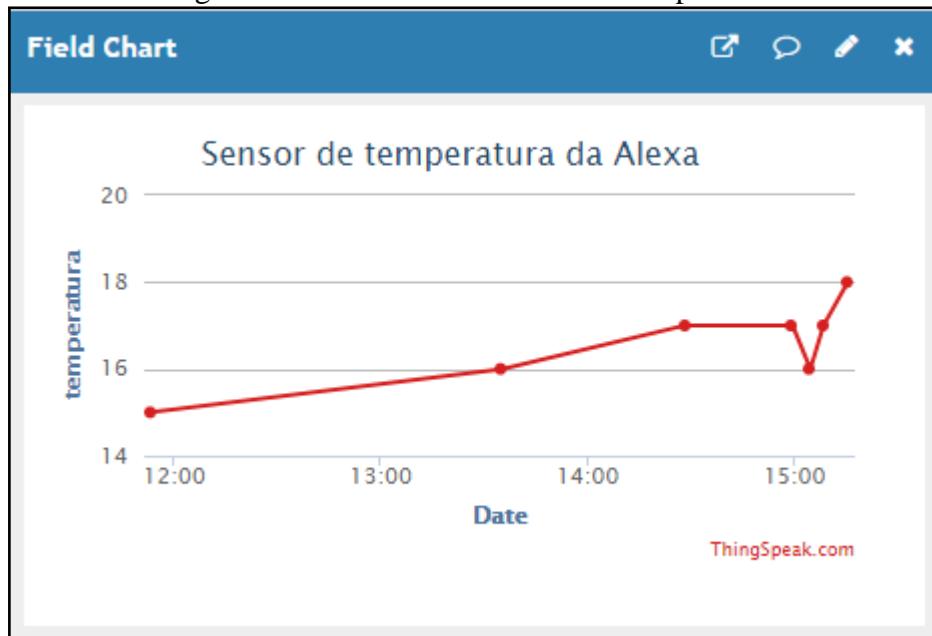
```

Fonte: elaborado pelo autor.

3.3.1.3 Utilizando um servidor HTTP para armazenar a temperatura

Após repetidas tentativas em enviar um dado do Arduino para o Web Service da Amazon sem sucesso, foi optado por utilizar o servidor HTTP da Thingspeak (2018). A utilização desta plataforma permite guardar o histórico de temperaturas e poder analisa-las num gráfico com data, hora e os respectivos valores, como demonstrado na Figura 16.

Figura 16 – Gráfico do histórico da temperatura



Fonte: Thingspeak (2018) e adaptado pelo autor.

Constantemente a aplicação embarcada verifica se está conectado na rede dentro do método `loop`. Neste mesmo método tem a chamada para o envio da temperatura para o servidor do ThingSpeak, como pode-se ver no Quadro 4. Na linha 396 vê-se que tem um contador dentro de uma condição para não ser enviada desnecessariamente inúmeras vezes a temperatura para o servidor.

Quadro 4 – Código do método `loop`

```

391 void loop()
392 {
393     if (!client.connected()) {
394         reconnect();
395     }
396     if (tempCounter > 1000) {
397         tempCounter = 0;
398         dht.begin();
399         int temperature = (int) dht.readTemperature();
400         sendTemperature(temperature);
401         printf("Temperatura publicada: %d.\r\n", temperature);
402     }
403     delay(100);
404     tempCounter = tempCounter + 1;
405     client.loop();
406 }
```

Fonte: elaborado pelo autor.

No Quadro 5 nota-se que é simples conectar no servidor do ThingSpeak. Através do `thingspeakWifiClient.connect` conecta-se no servidor utilizando a porta 80 enviando a temperatura.

Quadro 5 – Código do método de envio da temperatura

```

309 void sendTemperature(int temperature) {
310     if (thingspeakWifiClient.connect(thingspeakServer,80)) {
311         String postStr = thingspeakApiKey;
312         postStr += "&field1=";
313         postStr += String(temperature);
314
315         thingspeakWifiClient.print("POST /update HTTP/1.1\n");
316         thingspeakWifiClient.print("Host: api.thingspeak.com\n");
317         thingspeakWifiClient.print("Connection: close\n");
318         thingspeakWifiClient.print("X-THINGSPEAKAPIKEY: "+thingspeakApiKey+"\n");
319         thingspeakWifiClient.print("Content-Type: application/x-www-form-urlencoded\n");
320         thingspeakWifiClient.print("Content-Length: ");
321         thingspeakWifiClient.print(postStr.length());
322         thingspeakWifiClient.print("\n\n");
323         thingspeakWifiClient.print(postStr);
324     }
325     thingspeakWifiClient.stop();
326 }
```

Fonte: elaborado pelo autor.

3.3.1.4 Entendendo a estrutura da Amazon Web Services

O maior desafio deste trabalho foi entender como funciona todo o processo de configuração da Amazon Alexa e do AWS Lambda (2018). Para o desenvolvimento deste

trabalho foi usado o exemplo de Realtek Iot/Arduino Solution (2018) para repetir os passo e entender como funciona.

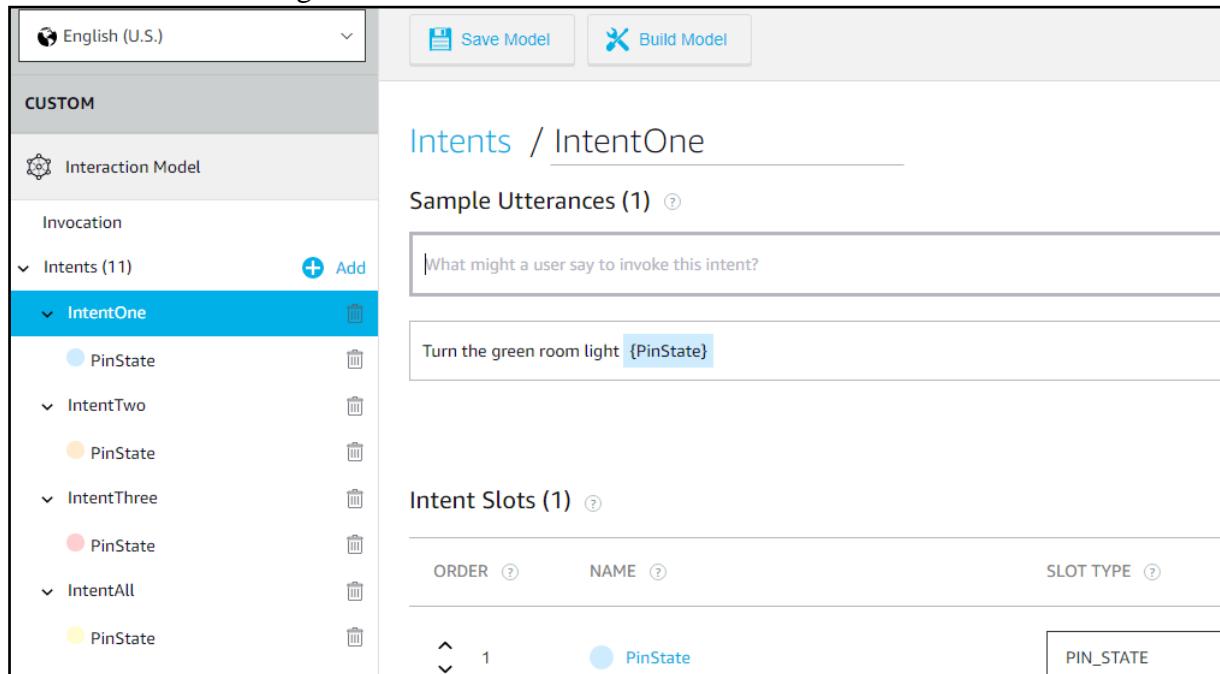
Contudo, a página da Amazon Alexa, tanto quanto do AWS Lambda e AWS IoT tiveram diversas modificações, aumentando a dificuldade em entender todo o processo dos mesmos. Além do mais, o passo-a-passo da Realtek Iot/Arduino Solution é em inglês.

Pensando nestas dificuldades, foi criado um passo-a-passo apresentado o apêndice B de como configurar o Amazon Alexa, criar a função no Amazon Lambda e registrar um objeto e gerar as chaves de segurança da “coisa” em AWS IoT.

Primeiramente cria-se uma *skill* na Amazon Alexa. Para isso, primeiro cadastra-se o nome de invocação. Esse nome de invocação é o nome que a Alexa irá identificar que se trata da *skill* criada. Em seguida, são criadas as ações ou *intents*, que a Alexa irá solicitar ao Amazon Lambda que seja executado cada comando.

Os comandos são criados na tela de *intents*. Para ativar a execução de um *intent*, podem ser criadas diversos comandos. Ou seja, quando se quer que através de diversas maneiras de falar apenas uma específica ação seja executada, cria-se uma lista de comando para apenas um *intent*. Na Figura 17 pode-se observar que o comando contém uma palavra entre chaves. Esta palavra chama-se *slot*. Um *slot* é uma espécie de variável, onde os seus valores podem ser previamente definidos. Neste exemplo o *slot* “PinState” possui dois estados, *on* ou *off*.

Figura 17 – Tela de *intents* criados na Amazon Alexa



Fonte: elaborado pelo autor.

Cada *intent* é uma ação. Cada ação pode ter um ou mais comandos. Cada comando pode ou não ter um *slot*. No comando de acender ou apagar o quarto verde vemos que tem apenas

um comando com um *slot*. Já na ação (*intent*) de acender ou apagar todas as luzes vê-se três comandos, ou três formas de chamar esta ação. Porém, no exemplo de abrir a porta, lê-se no Quadro 7 três comandos, contudo, sem um *slot*.

Cada comando do *intent* exposto no quadro acima faz uma chamada para o servidor no AWS. Essa execução do *intent* passa pela função `onIntent` apresentada no Quadro 6. De acordo com o *intent* executado, a execução faz uma chamada para o Arduino, que acende uma luz ou abre a porta, ou faz uma chama para o servidor ThingSpeak (2018) para obter o último dado informado pelo Arduino sobre a temperatura.

Quadro 6– Função `onIntent` no servidor

```

56 function onIntent(intentRequest, session, callback) {
57   console.log("onIntent requestId=" + intentRequest.requestId + ", sessionId=" + session.sessionId +
58
59   var intent = intentRequest.intent,
60     intentName = intentRequest.intent.name;
61
62   // Dispatch to your skill's intent handlers
63   if ("IntentOne" === intentName) {
64     setStateInSession(intent, session, callback);
65   } else if ("IntentTwo" === intentName) {
66     setStateInSession(intent, session, callback);
67   } else if ("IntentThree" === intentName) {
68     setStateInSession(intent, session, callback);
69   } else if ("IntentFour" === intentName) {
70     setStateInSession(intent, session, callback);
71   } else if ("IntentFive" === intentName) {
72     setStateInSession(intent, session, callback);
73   } else if ("IntentSix" === intentName) {
74     setStateInSession(intent, session, callback);
75   } else if ("IntentSeven" === intentName) {
76     setStateInSession(intent, session, callback);
77   } else if ("IntentEight" === intentName) {
78     setStateInSession(intent, session, callback);

```

Fonte: elaborado pelo autor.

3.3.2 Operacionalidade da implementação

Para preparar o protótipo para execução, é necessário acesso à Internet. Basta ligar uma fonte elétrica de 12V na maquete que automaticamente o Arduino se conectar na rede sem fio e estabelece uma conexão com o servidor da Amazon e uma conexão com o servidor do ThingSpeak (2018) enviando a temperatura.

Para efetivamente efetuar os comandos para a Alexa isso foi utilizado o aplicativo para IOS chamado Reverb, como pode ser visto sua tela inicial na Figura 18. Basta entrar com o usuário e senha da conta da Amazon que o aplicativo executa comandos criados na Amazon Alexa.

Figura 18 – Tela inicial do aplicativo Reverb



Fonte: elaborado pelo autor.

Houve uma certa dificuldade em entender o aplicativo ou o que a Amazon Alexa espera que seja dito para entender o nome de invocação da *skill*. O Quadro 7 mostra as ações criadas na Alexa e seus respectivos comandos.

Quadro 7 – Ações e os respectivos comandos

AÇÃO	COMANDO
acender/apagar luz do quarto verde	turn the green room light <i>on/off</i>
acender/apagar luz do quarto azul	turn the blue room light <i>on/off</i>
acender/apagar luz do banheiro de baixo	turn the downstairs bathroom light <i>on/off</i>
acender/apagar luz do banheiro de cima	turn the upstairs bathroom light <i>on/off</i>
acender/apagar luz da sala de baixo	turn the downstairs livingroom light <i>on/off</i>
acender/apagar luz da sala de cima	turn the upstairs livingroom light <i>on/off</i>
acender/apagar luz de fora	turn the outside light <i>on/off</i>
acender/apagar luz da cozinha	turn the kitchen light <i>on/off</i>
acender/apagar todas as luzes	turn all the lights <i>on/off</i> turn <i>on/off</i> all the lights turn <i>on/off</i> all lights
abrir a porta	to open the principal door to open the door open the door
consultar a temperatura	to get the temperature to get temperature get the temperature get temperature

Fonte: elaborado pelo autor.

A respeito da dificuldade dita acima, dar o comando de invocação “house” seguido do comando da ação não era entendido pelo aplicativo Reverb. Após pesquisas, viu-se que ele espera um antes da invocação. Por exemplo, ao falar “tell house turn the green room light on”,

foi possível finalmente entender o que era esperado. Também, ao falar “ask house to get the temperature” a Alexa interpreta o código e efetua o comando programado.

3.4 ANÁLISE DOS RESULTADOS

O tempo de resposta da Alexa superou as expectativas. Tanto para efetuar uma ação no Arduino quanto para acessar a requisição no ThingSpeak (2018). O Quadro 8 apresenta as três funcionalidades e seus respectivos tempos. Nota-se que não há alteração no tempo entre cada tipo de comando. Porém, o tempo pode ser relativo visto que depende da velocidade de conexão da Internet.

Quadro 8 - Tabela de tempo de resposta de execução

FUNCIONALIDADE	TEMPO EM SEGUNDOS
acender ou apagar uma luz	4
responder a temperatura	4
abrir a porta	4

Fonte: elaborado pelo autor.

Os três trabalhos correlatos utilizaram aplicações em smartphones. Contudo, apenas a de Beghini (2013) não é multiplataforma, ou seja, é necessário um dispositivo móvel na plataforma Android para ser executada. Pode-se ver este comparativo no Quadro 9. É importante ressaltar que todos dependem do acesso à Internet.

Quadro 9 - Comparativo das funcionalidades dos trabalhos correlatos

TRABALHO/ CARACTERÍSTICA	MULTIPLATAFORMA	NECESSIDADE DE INTERNET	ACIONAMENTO POR VOZ
Botke (2014)	sim	sim	não
Beghini (2013)	não	sim	não
Lisboa et al (2014)	sim	sim	não
Este trabalho	sim	sim	sim

Fonte: Elaborado pelo autor.

Diferentemente dos trabalhos correlatos, este é o único com acionamento por voz. O comando efetuado por voz traz uma comodidade bem maior ao usuário, pois não se faz necessário ter em mãos algum dispositivo para a aplicação tomar uma ação. Este tipo de funcionalidade pode abrir margem para novas inovações além da domótica. Podemos tomar como exemplo o ambiente industrial, onde o contato físico com algum dispositivo pode trazer algum perigo ao usuário. Logo, este pode tornar o trabalho mais prático e seguro.

4 CONCLUSÕES

Este trabalho descreveu o desenvolvimento de um protótipo de uma automação residencial utilizando a assistente de voz conforme os objetivos propostos. Os serviços da Amazon, Amazon Alexa, o AWS Lambda (2018) e o AWS IoT (2018) são ferramentas poderosas e muito útil para atender a proposta deste trabalho.

A Amazon Alexa (2018) é clara ao falar e muito assertiva em identificar o comando do usuário. Além do mais, o tempo de resposta ao efetuar o comando para o aplicativo por voz é em torno de quatro segundos, o que significa um excelente tempo de resposta.

As ferramentas utilizadas para a especificação do protótipo atenderam as necessidades. O Arduino Ameba tem um preço relativamente acessível. Com R\$210,44 já é possível ter um microcontrolador com diversas entradas e saídas e rede sem fio embutida, não necessitando a compra de um adaptador para o dispositivo para ter essa conexão.

Dentre vantagens do protótipo, destacam-se:

- a) dar os comandos de qualquer lugar do mundo a partir de um dispositivo móvel;
- b) grande praticidade para dar os comandos;
- c) em caso de queda de energia o dispositivo se reconecta automaticamente na rede sem fio.

Das limitações constatadas, sobressaem as seguintes:

- a) a Alexa entender apenas os idiomas inglês e francês;
- b) a necessidade de conexão com a Internet.

4.1 EXTENSÕES

Para trabalhos futuros, sugere-se alguns pontos que podem ser aprimorados a partir deste trabalho.

Recomenda-se a criação de um meio de que novos dispositivos sejam adicionados ao protótipo de forma automática para que este projeto seja customizado de maneira prática.

Para este trabalho foi criada uma *shadow* (sombra) representando um objeto dentro da residência. Sugere-se representação para cada dispositivo dentro da casa, pois cada dispositivo pode ter um estado diferente. Por exemplo, se cair a energia da residência, o microcontrolador Arduino pode buscar o estado anterior de cada item da casa.

REFERÊNCIAS

- AMAZON ALEXA.** **Amazon Alexa.** 2018. Disponível em:
<https://developer.amazon.com/alexa>. Acesso em: 08 mar. 2018.
- AMEBA8195.** [S.I.], 2016. Disponível em:
https://github.com/Ameba8195/Arduino/blob/master/hardware_v2/libraries/MQTTClient/examples/amazon_awsiot_with_ack/amazon_awsiot_with_ack.ino. Acesso em: 08 mar. 2018.
- AWS.** **Computação em nuvem com a Amazon Web Services.** 2018. Disponível :
<https://aws.amazon.com/pt/what-is-aws/>. Acesso em: 30 jun. 2018.
- _____. **O que é a computação em nuvem?** 2018b. Disponível em:
https://aws.amazon.com/pt/what-is-cloud-computing/?nc1=f_cc. Acesso em: 30 jun. 2018.
- AWS IOT.** **AWS.** 2018. Disponível em: <https://console.aws.amazon.com/iot>. Acesso em: 30 jun. 2018.
- _____. **O que é a Internet das Coisas?** 2018b. Disponível em:
<https://aws.amazon.com/pt/iot/>. Acesso em: 30 jun. 2018.
- AWS LAMBDA.** **AWS Lambda,** 2018. Disponível em:
<https://console.aws.amazon.com/lambda>. Acesso em: 08 mar. 2018.
- _____. **AWS Lambda,** 2018b. Disponível em:
https://aws.amazon.com/pt/lambda/?nc2=h_m1. Acesso em: 08 mar. 2018.
- BEGHINI, Lucas B.** **Automação Residencial de Baixo Custo por Meio de Dispositivos Móveis com Sistema Operacional Android.** 2013. 76 f. Trabalho de Conclusão de Curso (Engenharia Elétrica) – Departamento de Engenharia Elétrica, Universidade de São Paulo, São Carlos. Disponível em: http://www.tcc.sc.usp.br/tce/disponiveis/18/180450/tce-04022014-152853/publico/Beghini_Lucas_Bragazza.pdf. Acesso em: 03 mar. 2018.
- BOTKE, Daniel P.** **Automação de Residências Através de Aplicação Integrada com Arduino.** 2014. 77 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau. Disponível em: http://www.bc.furb.br/docs/MO/2014/363428_1_1.pdf. Acesso em: 03 mar. 2018.
- HAACK, William et al.** **Security Analysis of the Amazon Echo.** [S.I.], 2017. Disponível em:
<https://courses.csail.mit.edu/6.857/2017/project/8.pdf>. Acesso em: 08 mar. 2018.
- IOT CORE.** **AWS IoT Core.** 2018. Disponível em: <https://aws.amazon.com/pt/iot-core/>. Acesso em: 30 jun. 2018.
- LISBOA, Emerson F.** et al. WebHome – Automação residencial utilizando Raspberry PI. **Revista Ciência e Tecnologia**, Campinas, v.17, n.31, p. 35-43, jul. 2014. Disponível em: <http://www.revista.unisal.br/sj/index.php/123/article/view/365/249>. Acesso em: 08 mar. 2018.
- NEVES, Andressa.** **10 funções e dicas para usar o assistente virtual da Google.** São Bernardo do Campo, 2018. Disponível em: <https://canaltech.com.br/apps/10-funcoes-e-dicas-para-usar-o-assistente-virtual-da-google/>. Acesso em: 18 jul. 2018.
- REALTEK IOT/ARDUINO SOLUTION.** **Ameba Arduino:** Getting Started With RTL8195, 2018. Disponível em: <https://www.amebaito.com/en/ameba-arduino-getting-started>. Acesso em: 08 mar. 2018.

- _____. **Ameba Arduino: [RTL8195]** Amazon Alexa, 2018b. Disponível em: <<https://www.amebait.com/en/ameba-arduino-amazon-alexa/>>. Acesso em: 08 mar. 2018.
- _____. [S.I.], 2018c. Disponível em: <<https://www.amebait.com/wp-content/uploads/2016/08/code.txt>>. Acesso em: 08 mar. 2018.
- _____. **Ameba Arduino: Boards**, 2018d. Disponível em: <<https://www.amebait.com/en/boards/>>. Acesso em: 14 jul. 2018.
- SYMANTEC. [S.I.], 2018. Disponível em : <<https://www.symantec.com/content/en/us/enterprise/verisign/roots/VeriSign-Class%203-Public-Primary-Certification-Authority-G5.pem>>. Acesso em: 03 mar. 2018.
- THINGSPEAK. **Understand Your Things**. 2018. Disponível em: <<https://thingspeak.com>>. Acesso em: 08 mar. 2018.
- VIGLIAROLO, Brandon. **Amazon Alexa: The smart person's guide**. [S.I.], 2017. Disponível em: <<https://www.techrepublic.com/article/amazon-alexa-the-smart-persons-guide>>. Acesso em: 08 mar. 2018.

APÊNDICE A – Descrição dos Casos de Uso

Este apêndice descreve os casos de uso abordados nos diagramas expostos na seção 3.2.1 deste trabalho. No Quadro 10 apresenta-se a descrição do caso de uso Controlar Luzes do Usuário.

Quadro 10 – Descrição do UC-01 Controlar luzes

UC-01 – Controlar luzes

Acende ou apaga uma luz da maquete após solicitação da Amazon Alexa.

Ator: Arduino

Pré-condição: O Arduino deve estar conectado à Internet.

Pré-condição: A Alexa deve solicitar a alteração do estado da luz.

Pós-condição: O Arduino acende ou apaga a luz desejada.

Cenário Principal

1. A Alexa requisita a alteração de estado de uma luz;
2. O AWS Lambda altera o estado da sombra no AWS IoT;
3. O Arduino altera o estado da luz desejada.

A descrição do caso de uso UC-02 Abrir uma porta é abordado no Quadro 11.

Quadro 11 – Descrição do UC-02 Abrir uma porta

UC-02 – Abrir a porta

Abre a porta da maquete após a solicitação da Amazon Alexa.

Ator: Arduino

Pré-condição: O Arduino deve estar conectado à Internet.

Pré-condição: A Alexa deve solicitar abrir a porta da casa.

Pós-condição: O Arduino abre a porta.

Cenário Principal

1. A Alexa requisita que a porta seja aberta;
2. O AWS Lambda altera o estado da sombra no AWS IoT;
3. O Arduino abre a porta.

No Quadro 12 é descrito o caso de uso do usuário que consulta a temperatura ambiente.

Quadro 12 – Descrição do UC-03 Consultar temperatura ambiente

UC-03 – Acender ou apagar luzes

Consulta a temperatura ambiente da maquete.

Ator: Arduino

Pré-condição: O Arduino deve ter enviado a temperatura no servidor HTTP.

Pós-condição: A Alexa responde a temperatura ambiente da maquete.

Cenário Principal

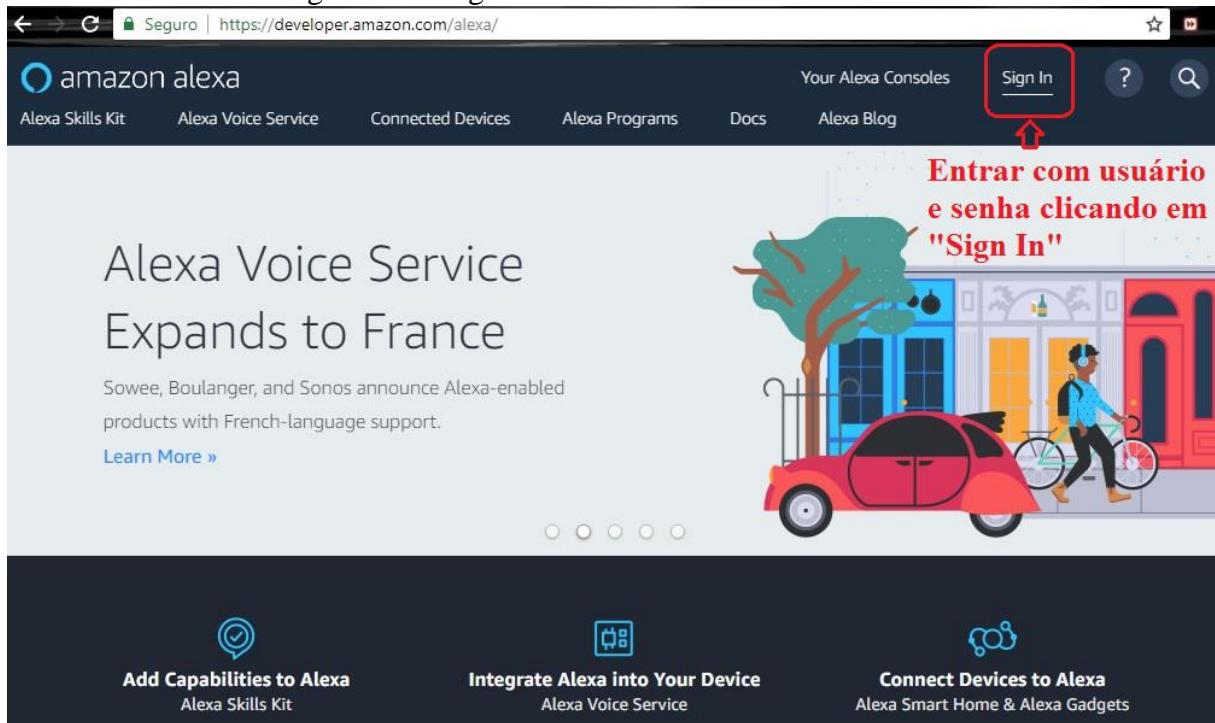
1. A Alexa requisita a temperatura da maquete;
2. O AWS Lambda consulta o dado no servidor HTTP e envia para a Alexa;
3. A Alexa responde ao usuário a temperatura ambiente da maquete.

APÊNDICE B – Criando uma Skill para a Amazon Alexa

A Amazon Alexa fornece dois tipos de serviços: O Smart Home Skills, que são modelos já existentes para o uso, e Custom Skills, que são habilidades customizáveis para o desenvolvedor criar conforme sua necessidade. Neste apêndice é apresentado o passo-a-passo de exemplo para a criação de uma *skill* customizável.

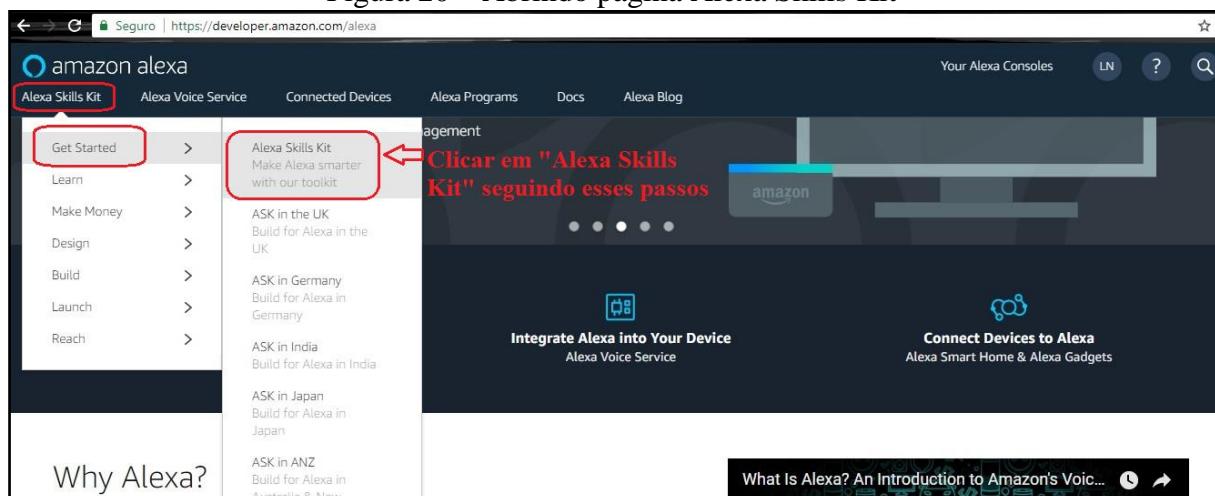
A partir da Figura 19 até a Figura 33 é mostrado como configurar uma nova função na Amazon Alexa (2018).

Figura 19 – Página inicial do site da Amazon Alexa



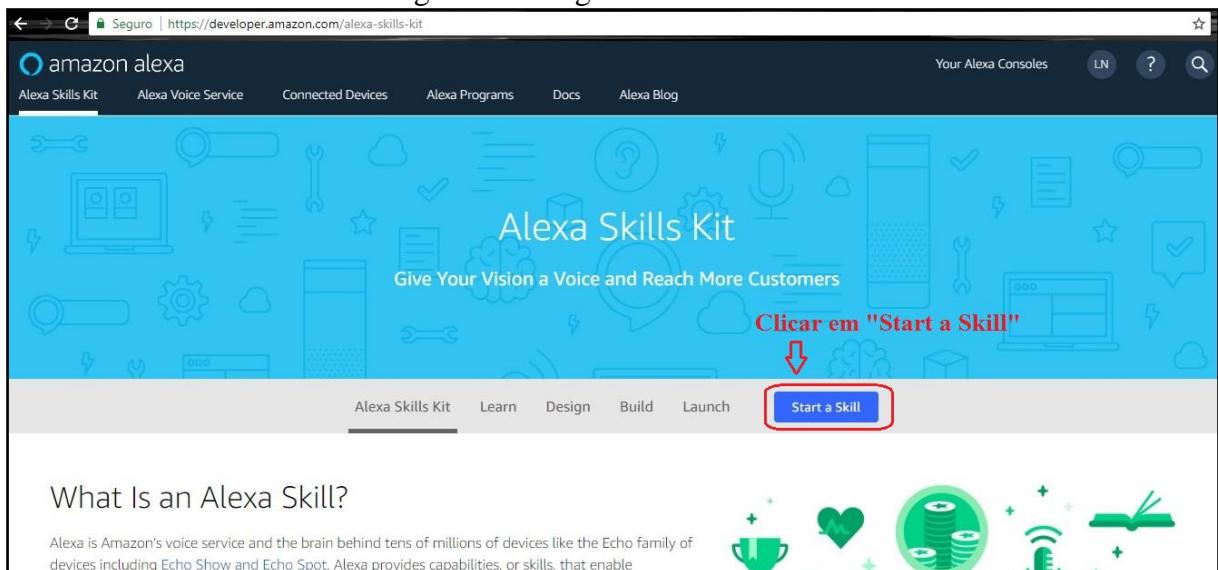
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 20 – Abrindo página Alexa Skills Kit



Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 21 – Página Alexa Skill Kit



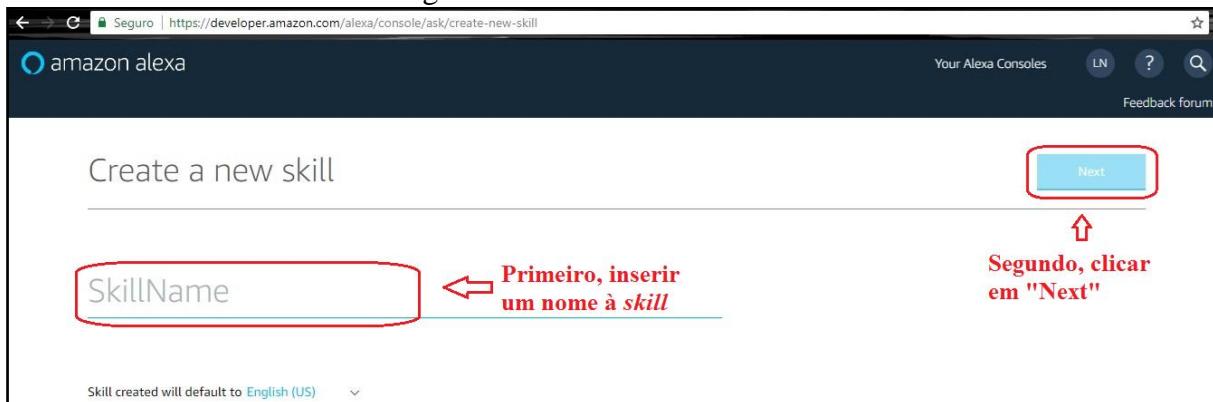
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 22 – Página com as *skills* já criadas

Skills						Earnings	Payments	Feedback forum			
Alexa Skills											
Welcome to the new Alexa Skills Kit Developer Console											
Curious about what's new? Watch the video overview or read about what's changed.											
SKILL NAME	LANGUAGE	TYPE	MODIFIED	STATUS	ACTIONS						
 Home Automation With A...	English (US)	Custom	2018-05-20	● In Development	Measure Edit Delete	Clicar em "Create Skill"					

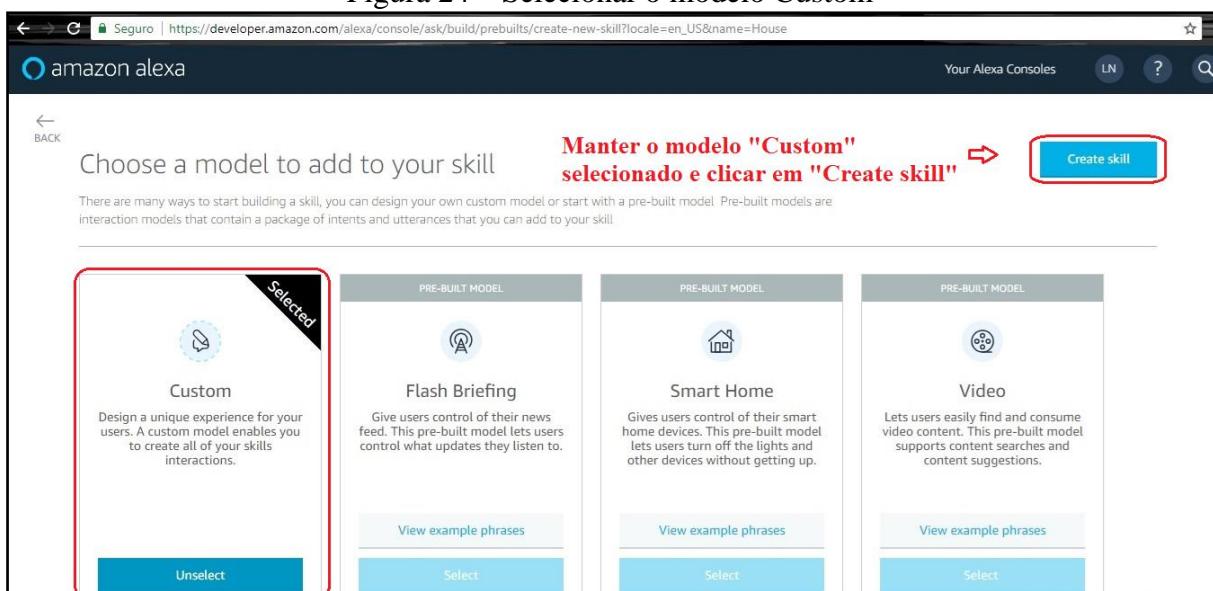
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 23 – Dando o nome à skill



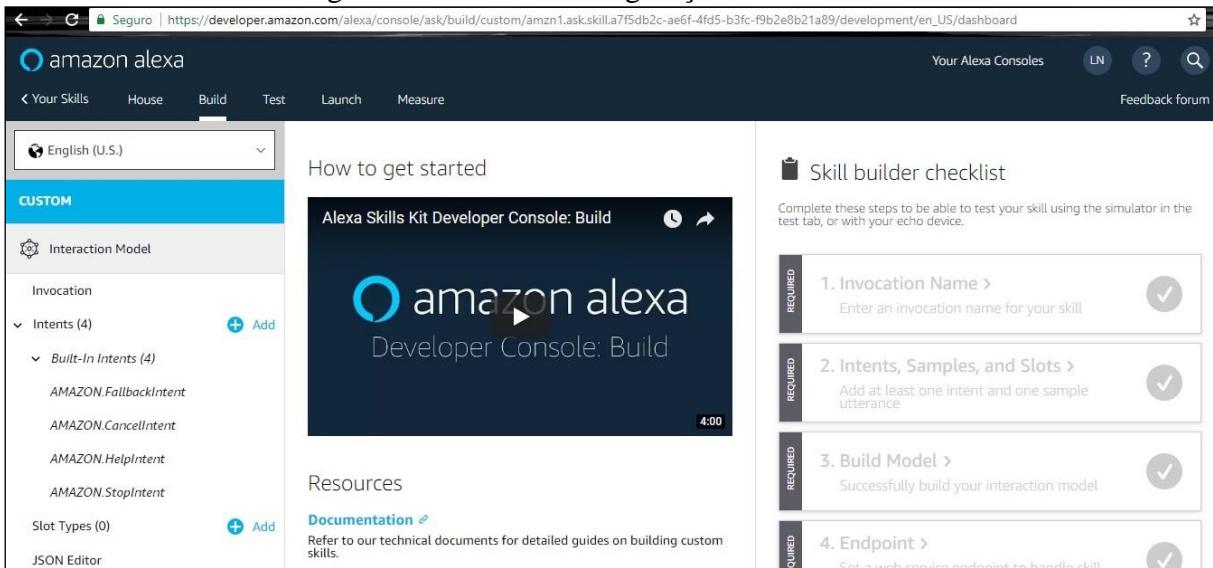
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 24 – Selecionar o modelo Custom



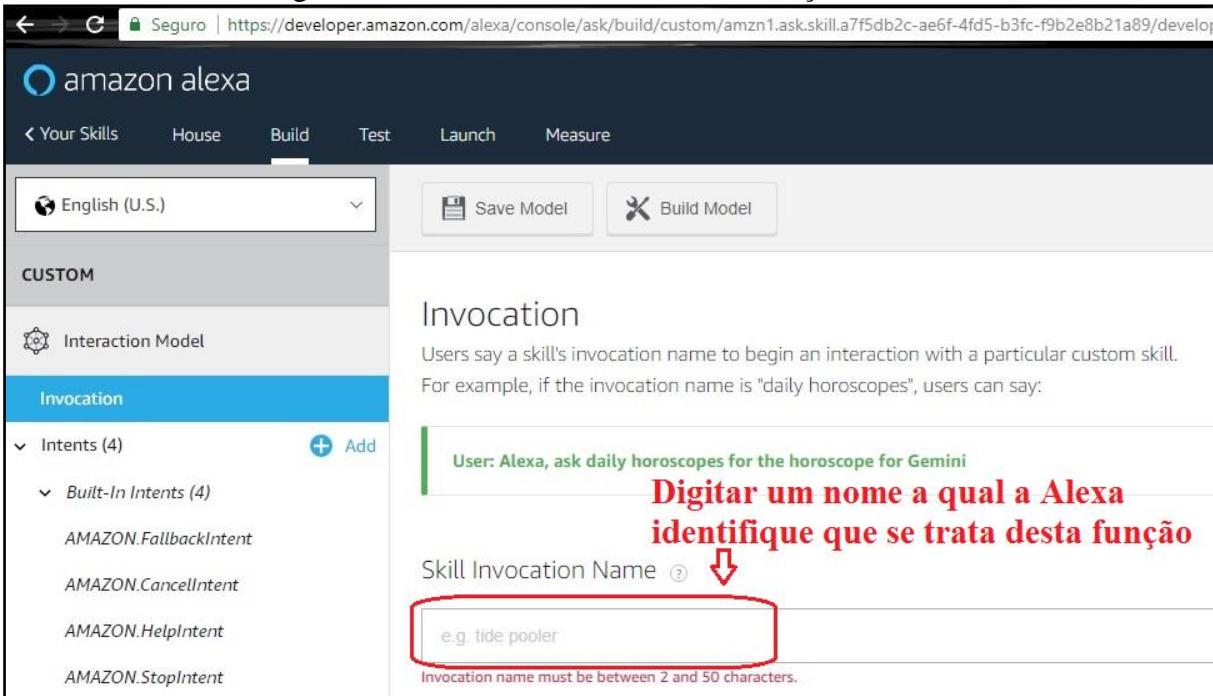
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 25 – Tela de configuração da *skill* criada



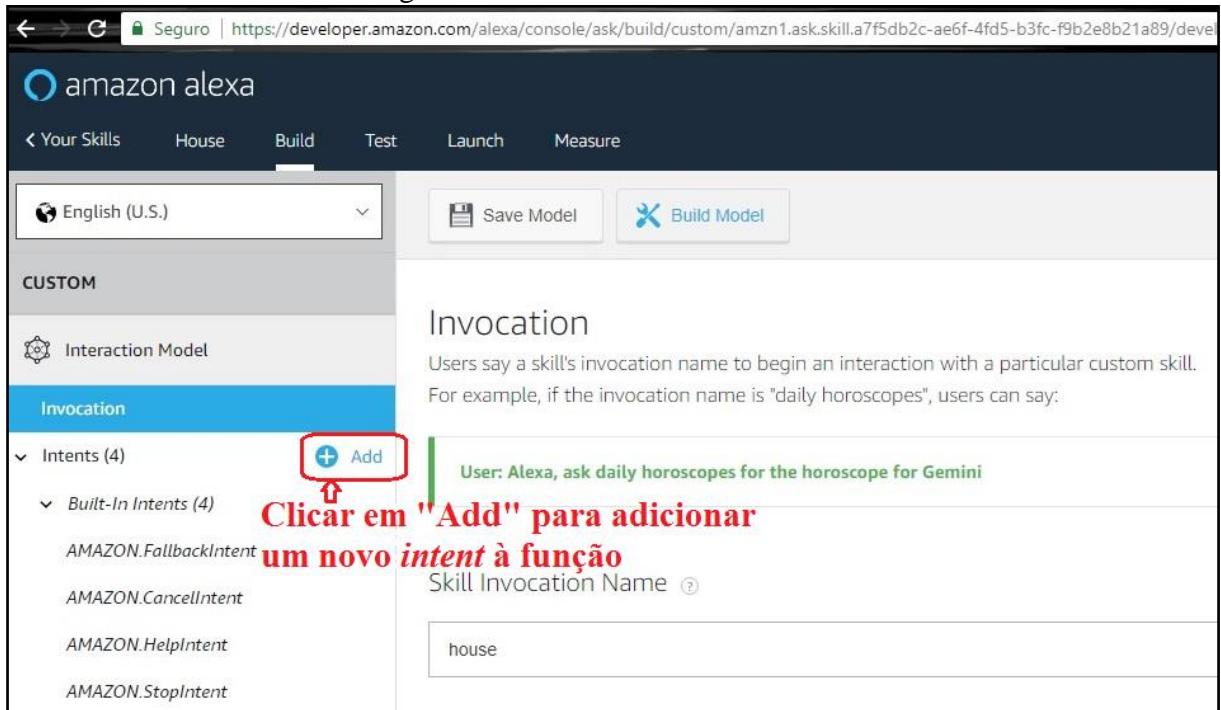
Fonte: Amazon Alexa (2018).

Figura 26 – Inserindo um nome de invocação da *skill*



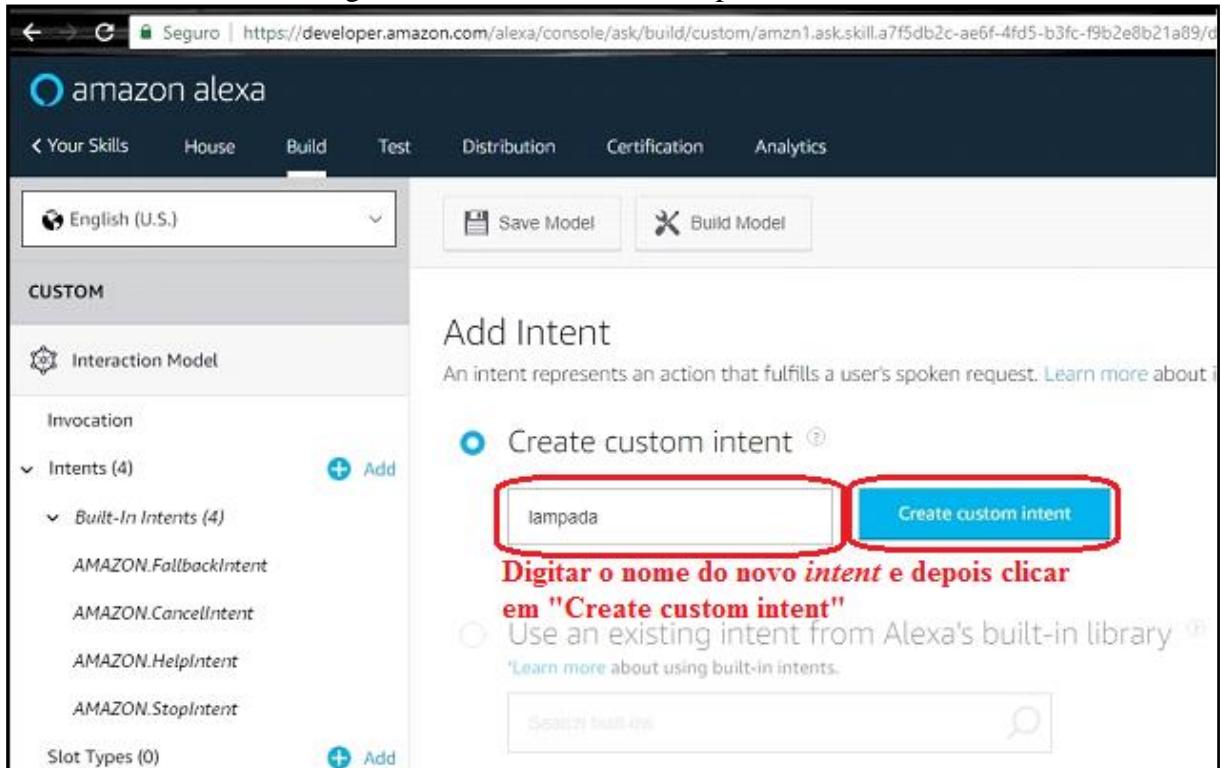
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 27 – Adicionando novo *intent*



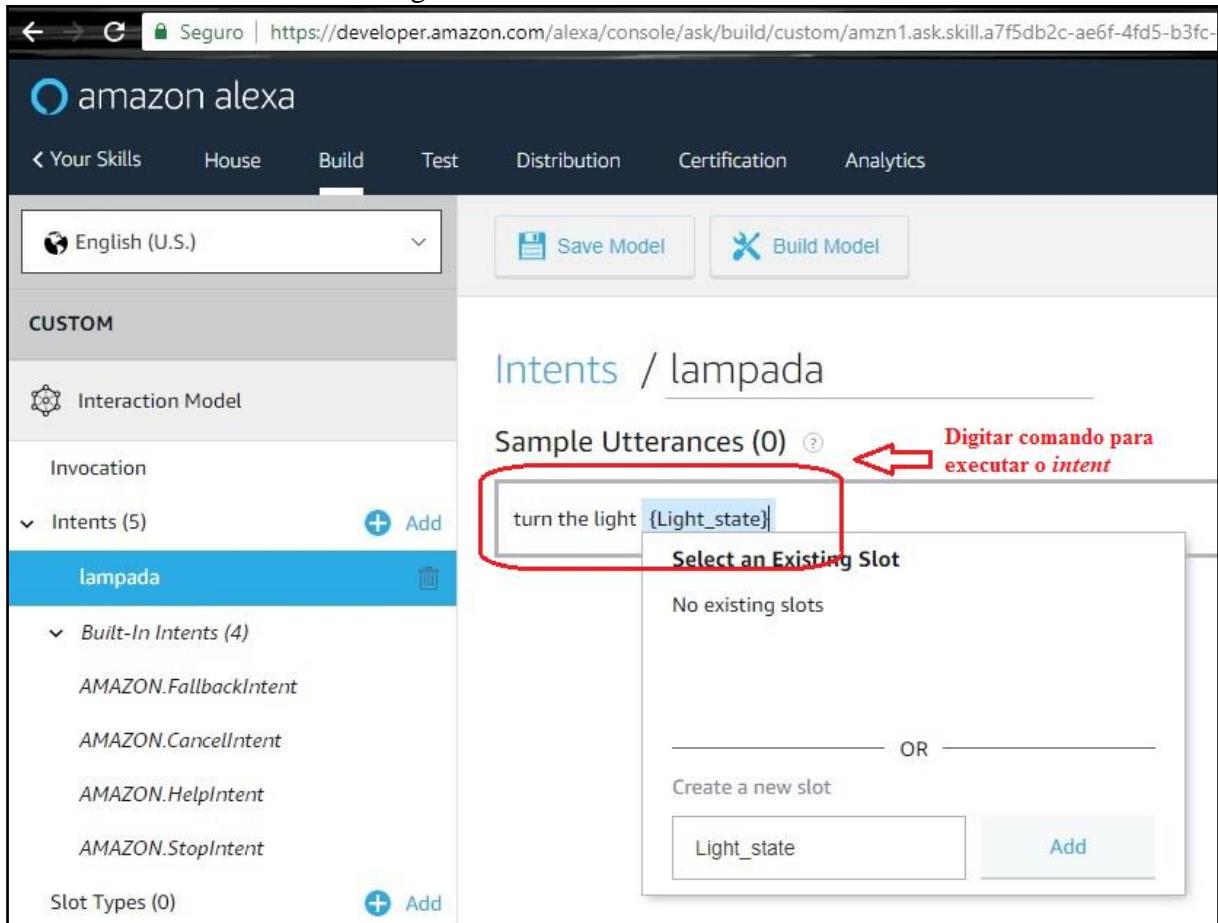
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 28 – Criando um *intent* personalizável



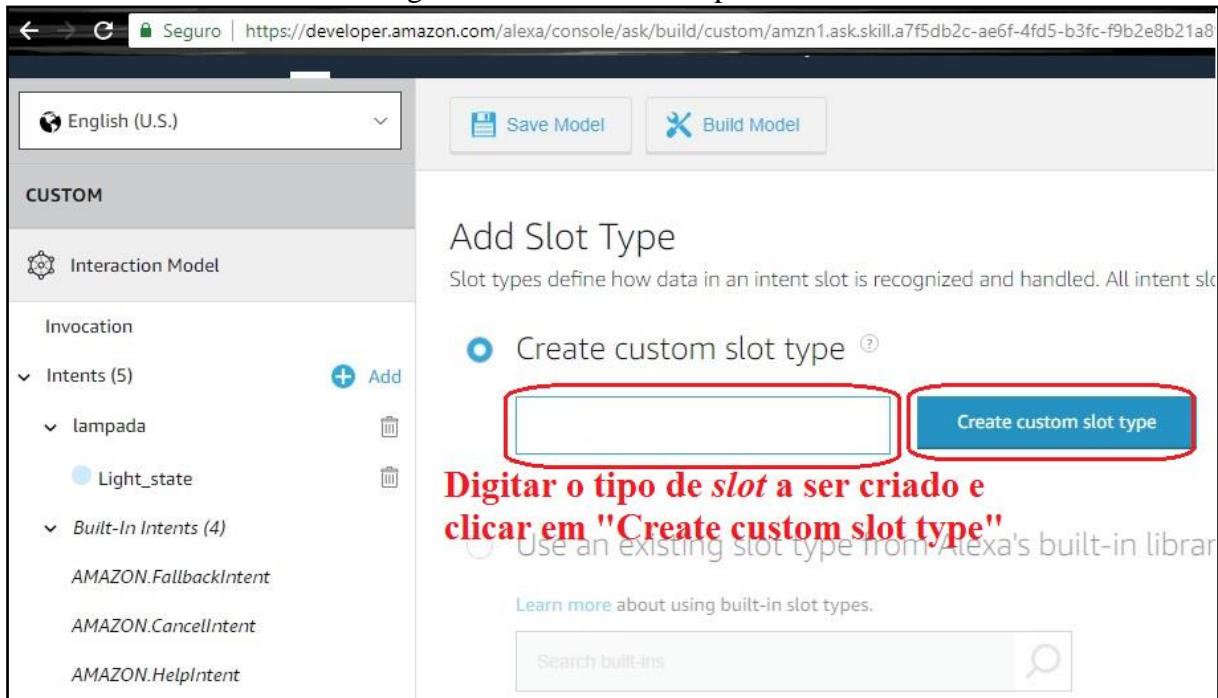
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 29 – Inserindo um comando



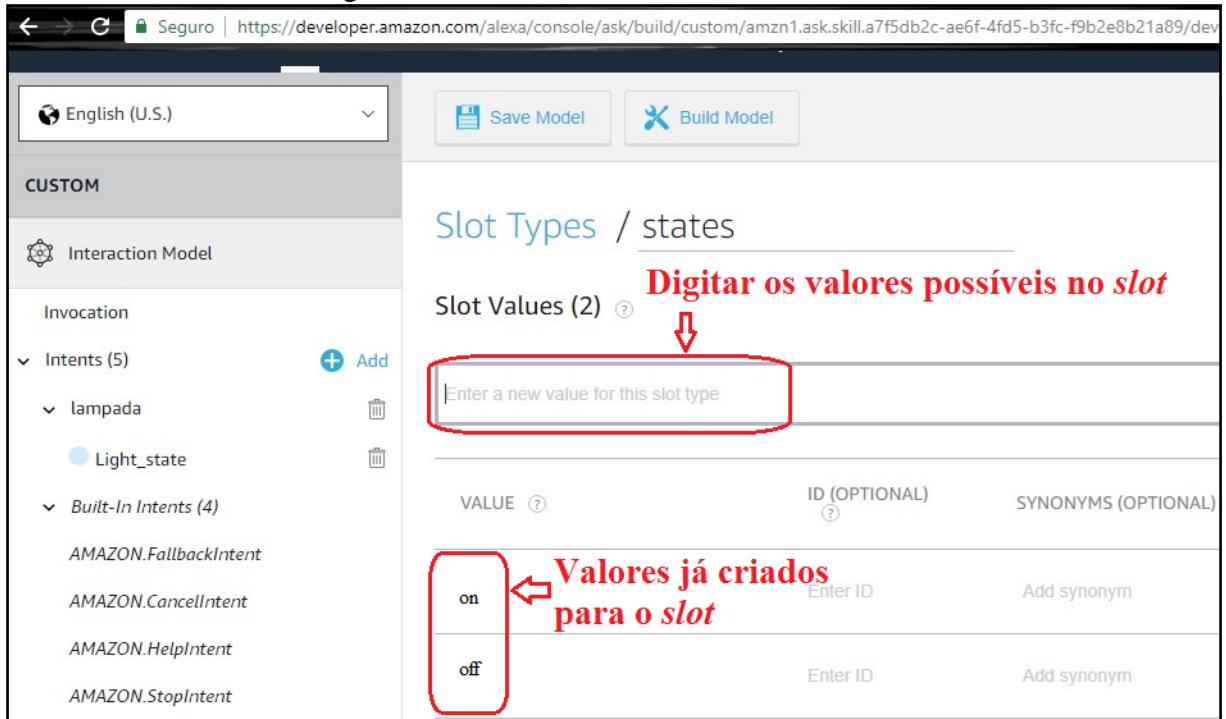
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 30 – Criando um tipo de slot



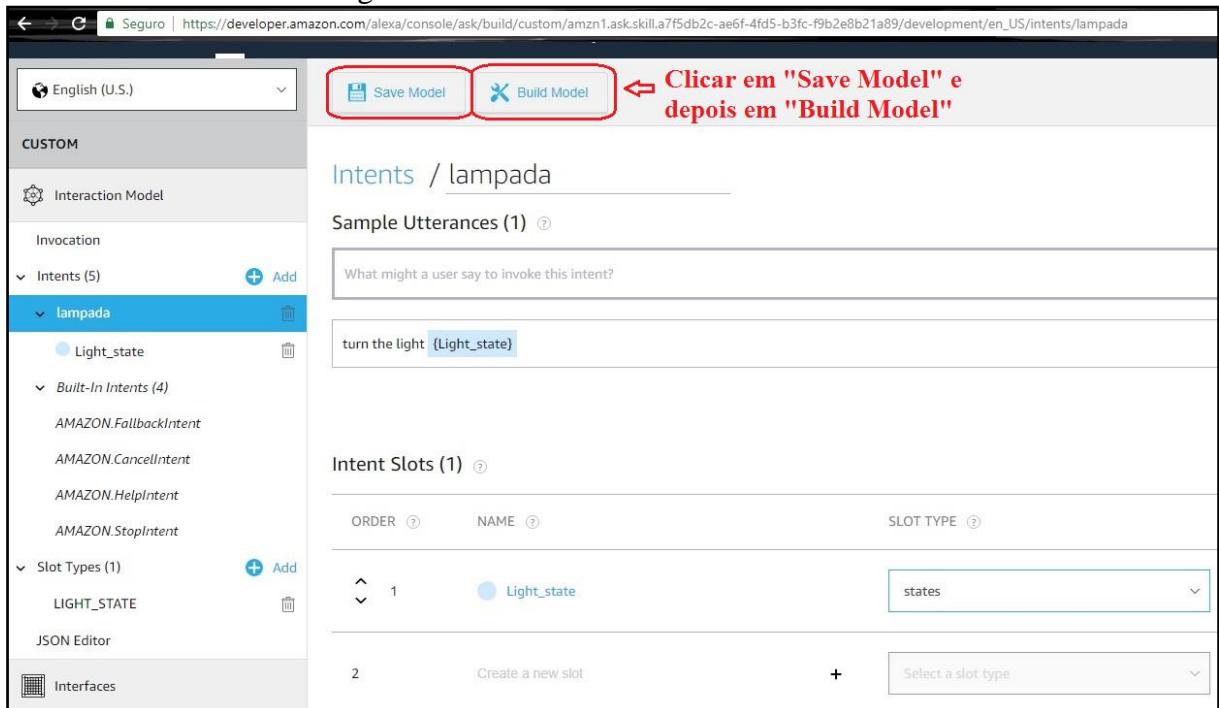
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 31 – Adicionando valores ao novo slot



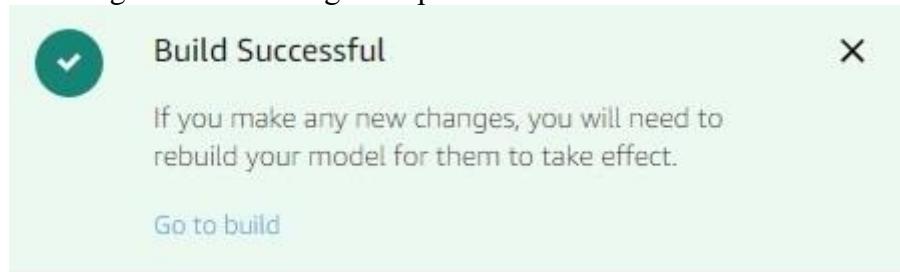
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 32 – Tela com o slot e intent criados



Fonte Amazon Alexa (2018) e alterado pelo autor.

Figura 33 – Mensagem depois de clicar em “Build Model”



Fonte: Amazon Alexa (2018).

Nesta etapa, deverá ser criada uma função no AWS Lambda (2018). Os passos são descritos a partir da Figura 34 até a Figura 37.

Figura 34 – Página com as funções Lambda

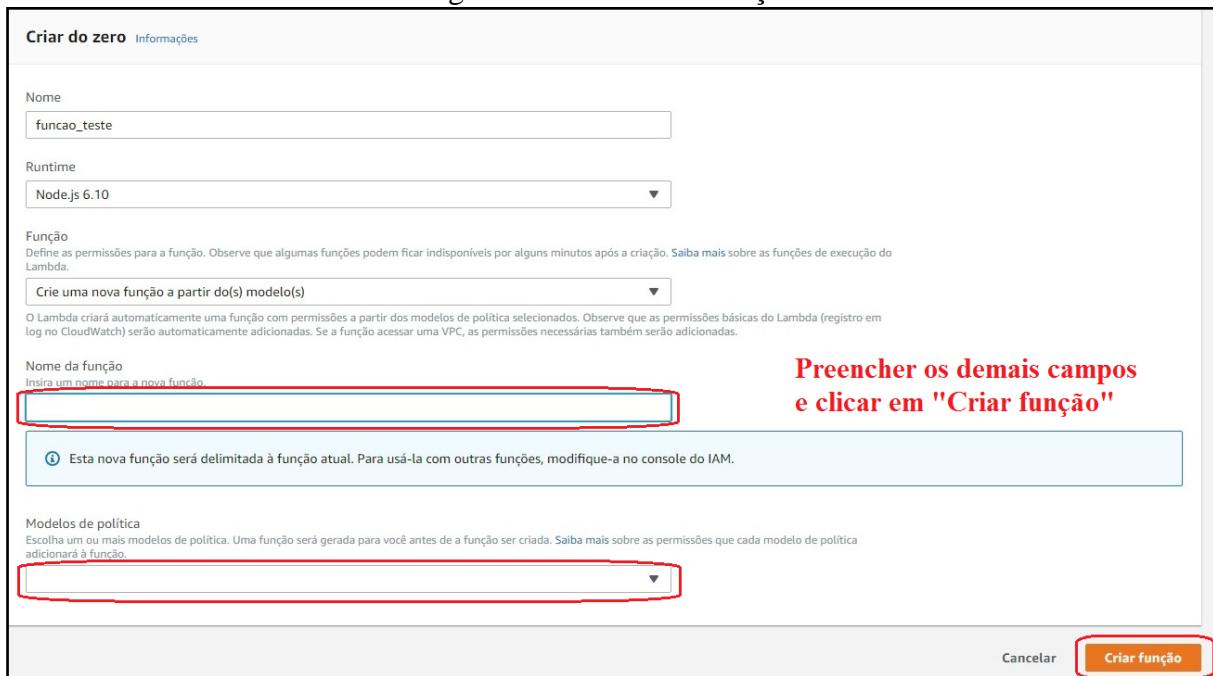
Nome da função	Descrição	Runtime	Tamanho do código	Última modificação
HelloWorld		Node.js 6.10	1.3 kB	há 4 meses
ControlLight	Demonstrates a basic skill built with the Amazon Alexa Skills Kit.	Node.js 6.10	2.3 kB	há 10 dias

Fonte: AWS Lambda (2018) e alterado pelo autor.

Figura 35 – Dando um nome para uma nova função

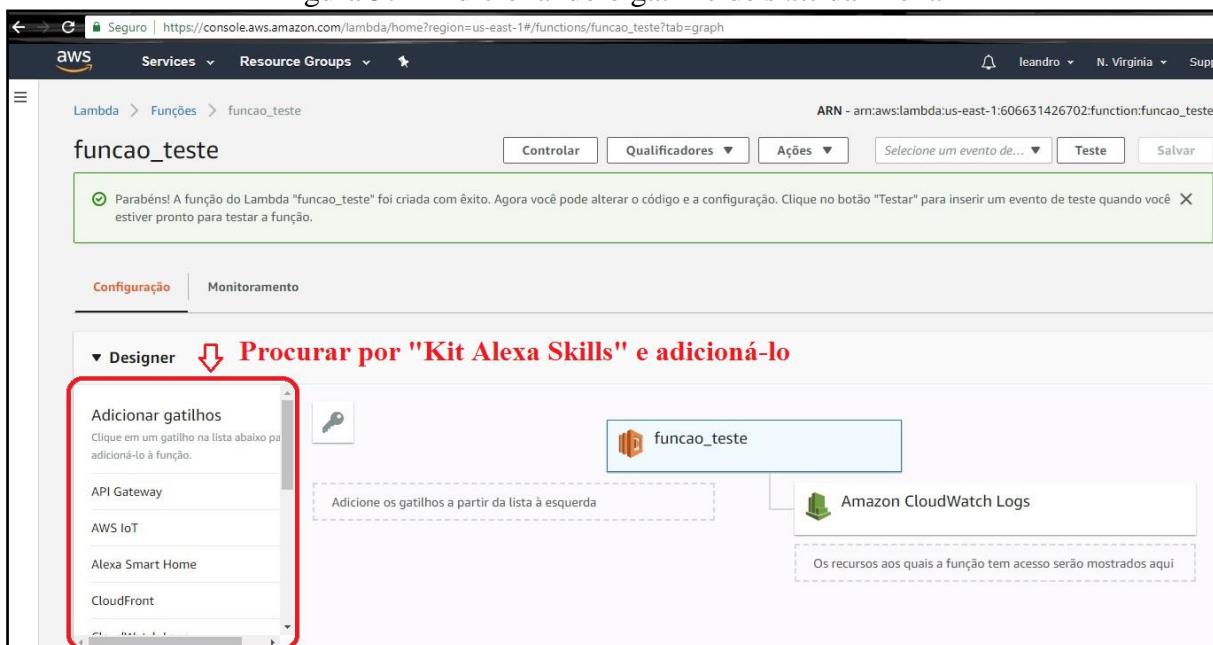
Fonte: AWS Lambda (2018) e alterado pelo autor.

Figura 36 – Criando a função



Fonte: AWS Lambda (2018) e alterado pelo autor.

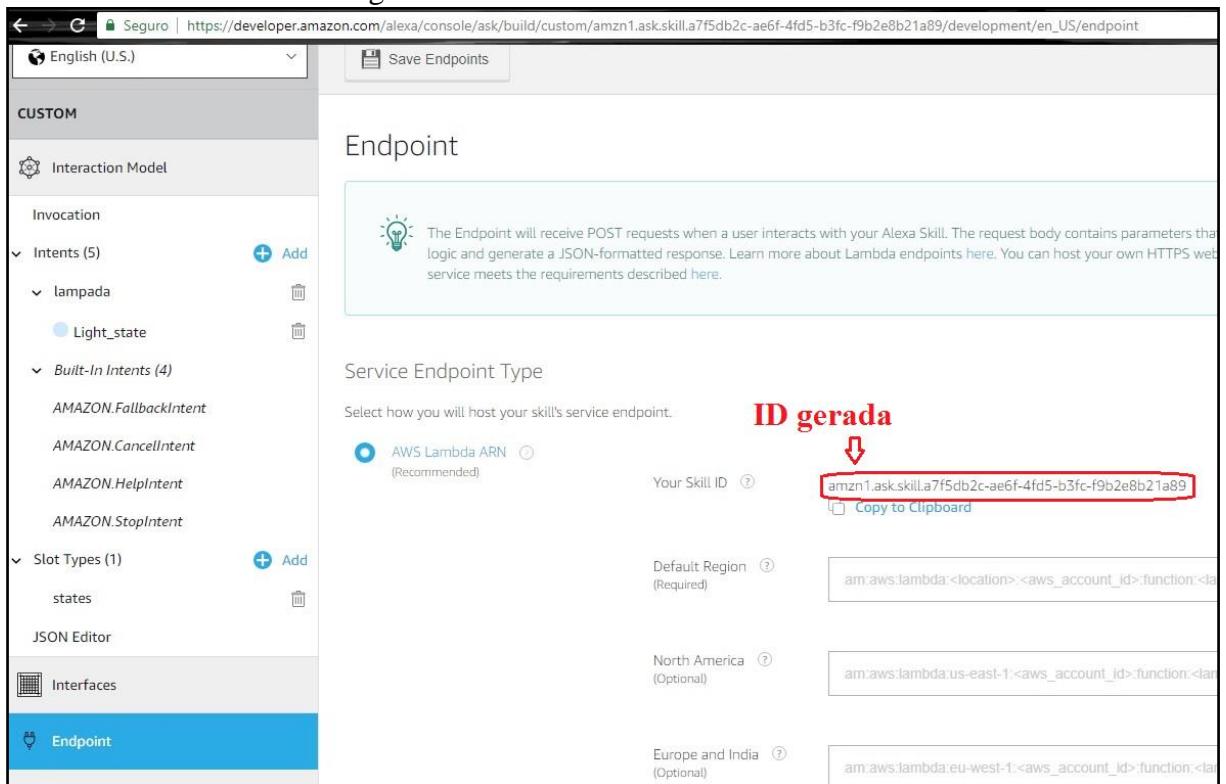
Figura 37 – Adicionando o gatilho de skill da Alexa



Fonte: AWS Lambda (2018) e alterado pelo autor.

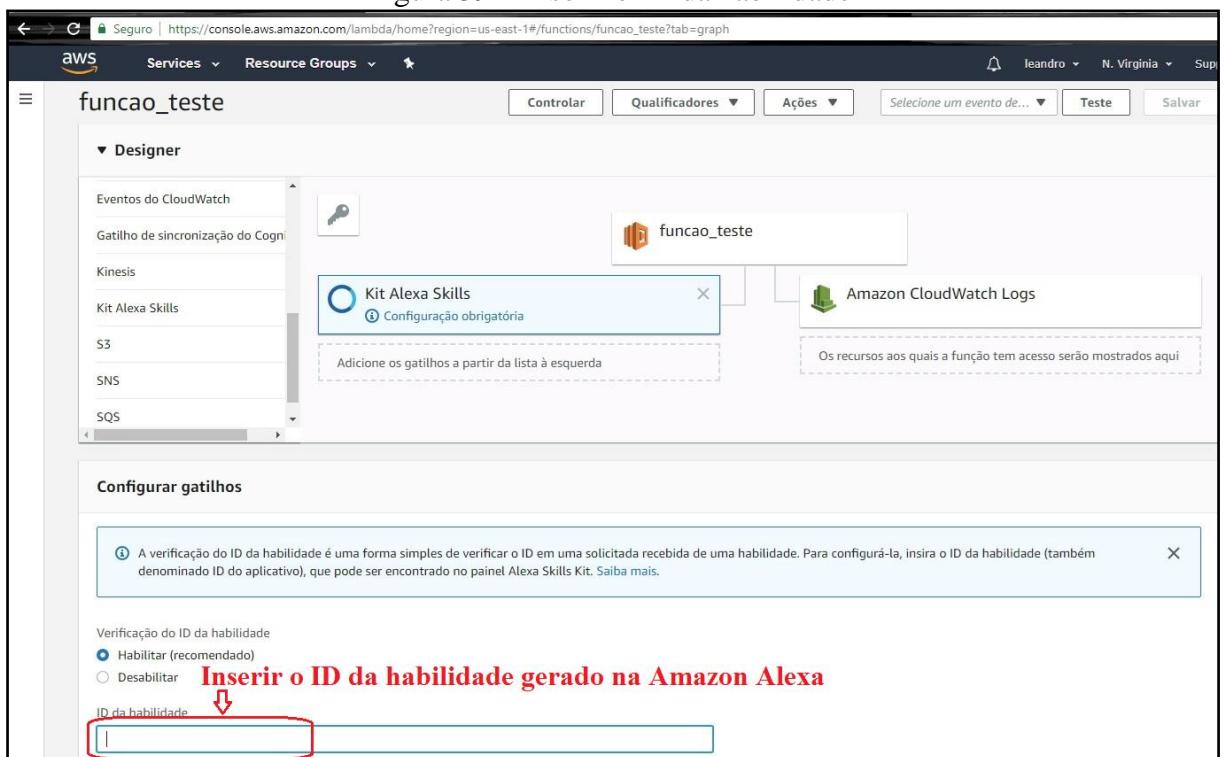
Nesta etapa será necessário o ID gerado na Amazon Alexa (2018). A Figura 38 aponta onde se obtém na Amazon Alexa (2018). Na Figura 39 mostra onde deve ser inserido o ID da Alexa. Após inserir o ID, clicar em salvar. A Figura 40 e a Figura 41 apresentam os passos seguintes.

Figura 38 – ID da Skill na Amazon Alexa



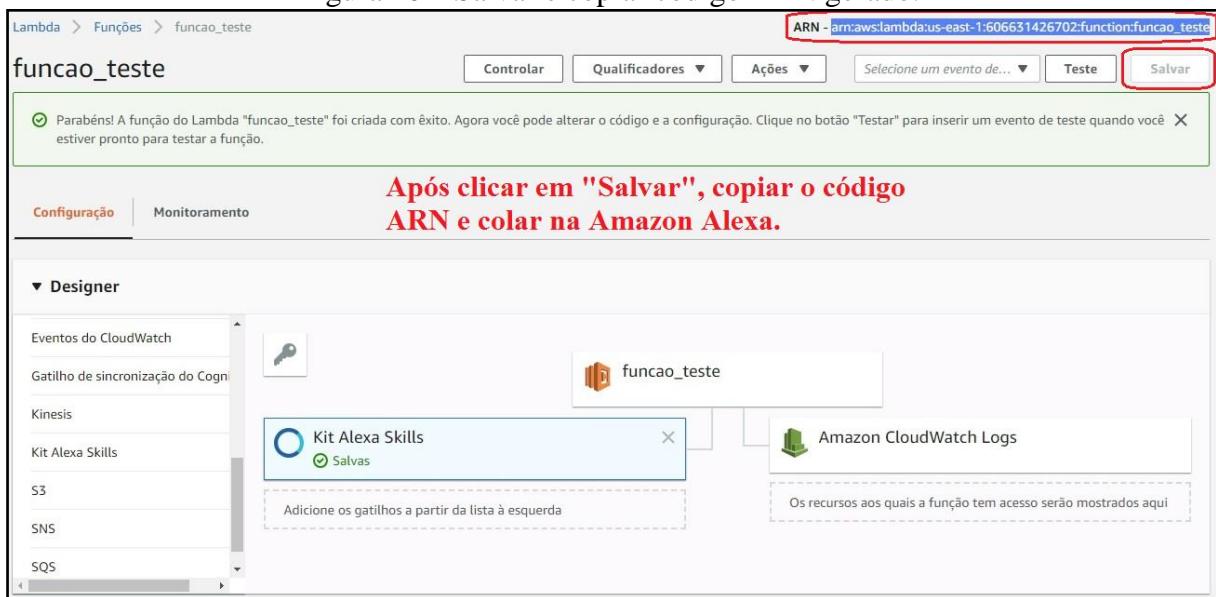
Fonte: Amazon Alexa (2018) e alterado pelo autor.

Figura 39 – Inserir o ID da habilidade



Fonte: AWS Lambda (2018) e alterado pelo autor.

Figura 40 – Salvar e copiar código ARN gerado.



Fonte: AWS Lambda (2018) e alterado pelo autor.

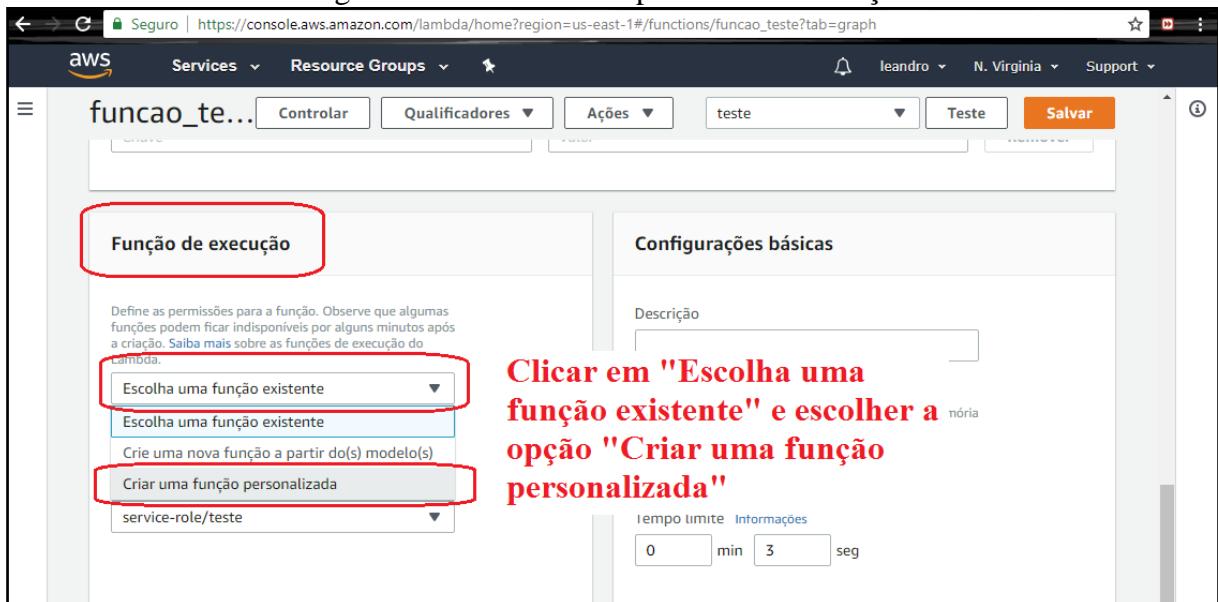
Figura 41 – Colar código ARN gerado no AWS Lambda (2018)



Fonte: Amazon Alexa (2018) e alterado pelo autor.

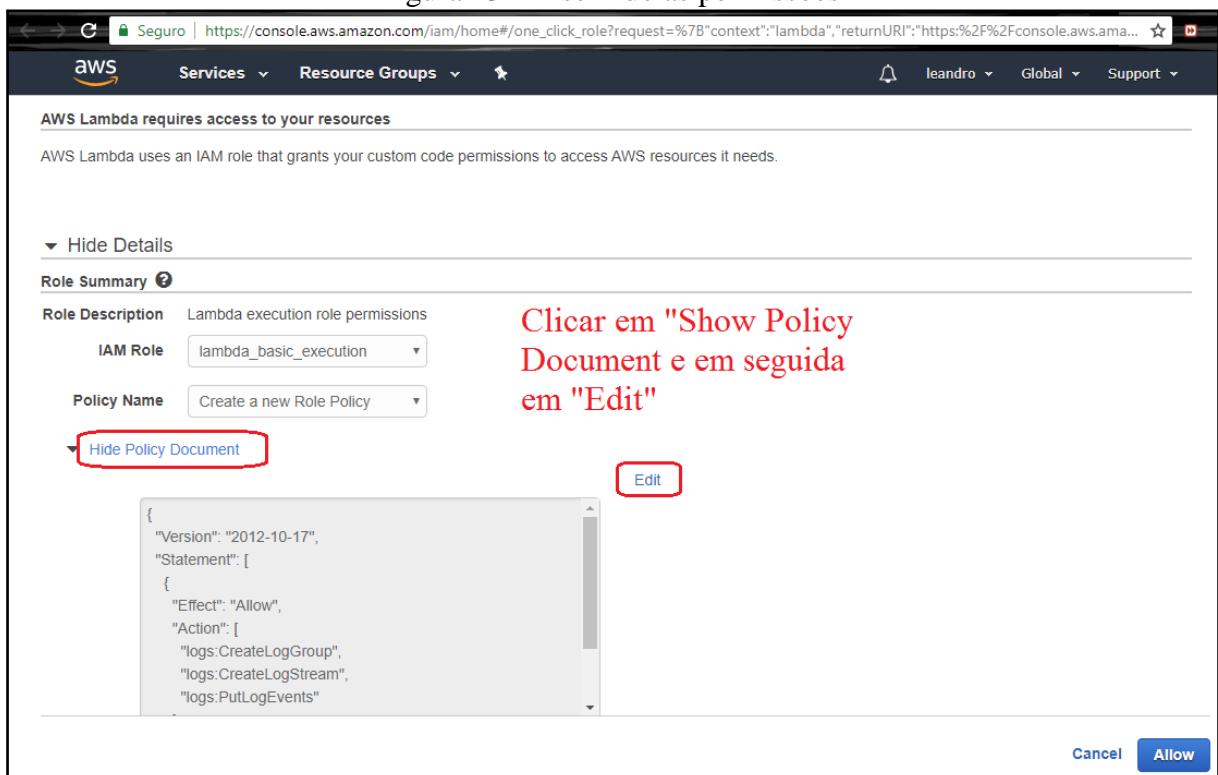
Agora será feita a configuração de permissão para a função criada. Para isso, seguir o passo-a-passo da Figura 42 e da Figura 43.

Figura 42 – Definindo as permissões da função



Fonte: AWS Lambda (2018) e alterado pelo autor.

Figura 43 – Inserindo as permissões



Fonte: AWS Lambda (2018) e alterado pelo autor.

Na tela da Figura 43, incluir o código de permissão do AWS IoT (2018) seguindo o exemplo do Quadro 13 abaixo.

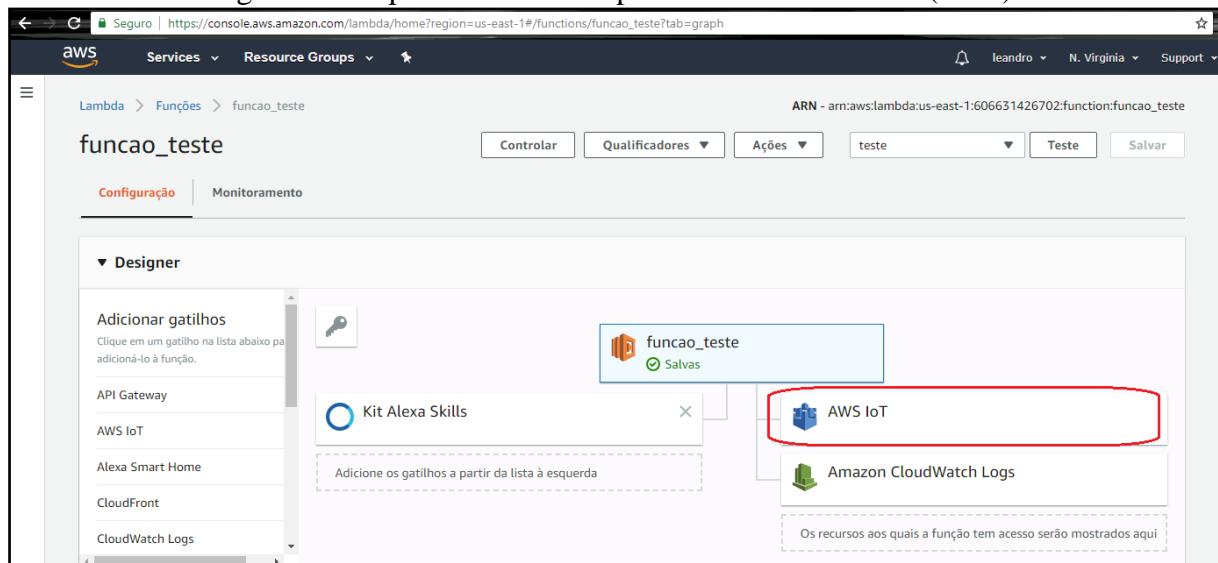
Quadro 13 – Código exemplo para modificar permissão da função

Código anterior	Código novo
<pre>{ "Effect": "Allow", "Action": ["logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents"], "Resource": "arn:aws:logs:*:*:*" }</pre>	<pre>{ "Effect": "Allow", "Action": ["logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents"], "Resource": "arn:aws:logs:*:*:*" }, { "Effect": "Allow", "Action": ["iot:/*"], "Resource": "arn:aws:iot:*:*:*" }</pre>

Fonte: elaborado pelo autor.

Ao clicar no botão “Allow” a página deverá ficar como a Figura 44.

Figura 44 – Após a inclusão da permissão do AWS IoT (2018)



Fonte: AWS Lambda (2018) e alterado pelo autor.

A partir da Figura 45 é apresentado o passo-a-passo de como criar uma “coisa”, ou um novo dispositivo, à função em AWS IoT (2018), terminando na Figura 55.

Figura 45 – Página inicial de AWS IoT (2018)



Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 46 – Incluindo novo dispositivo



Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 47 – Conectando o dispositivo

Conectar ao AWS IoT

A conexão de um dispositivo (como um kit de desenvolvimento ou seu computador) com o AWS IoT requer a execução das etapas a seguir. Neste processo, você vai:

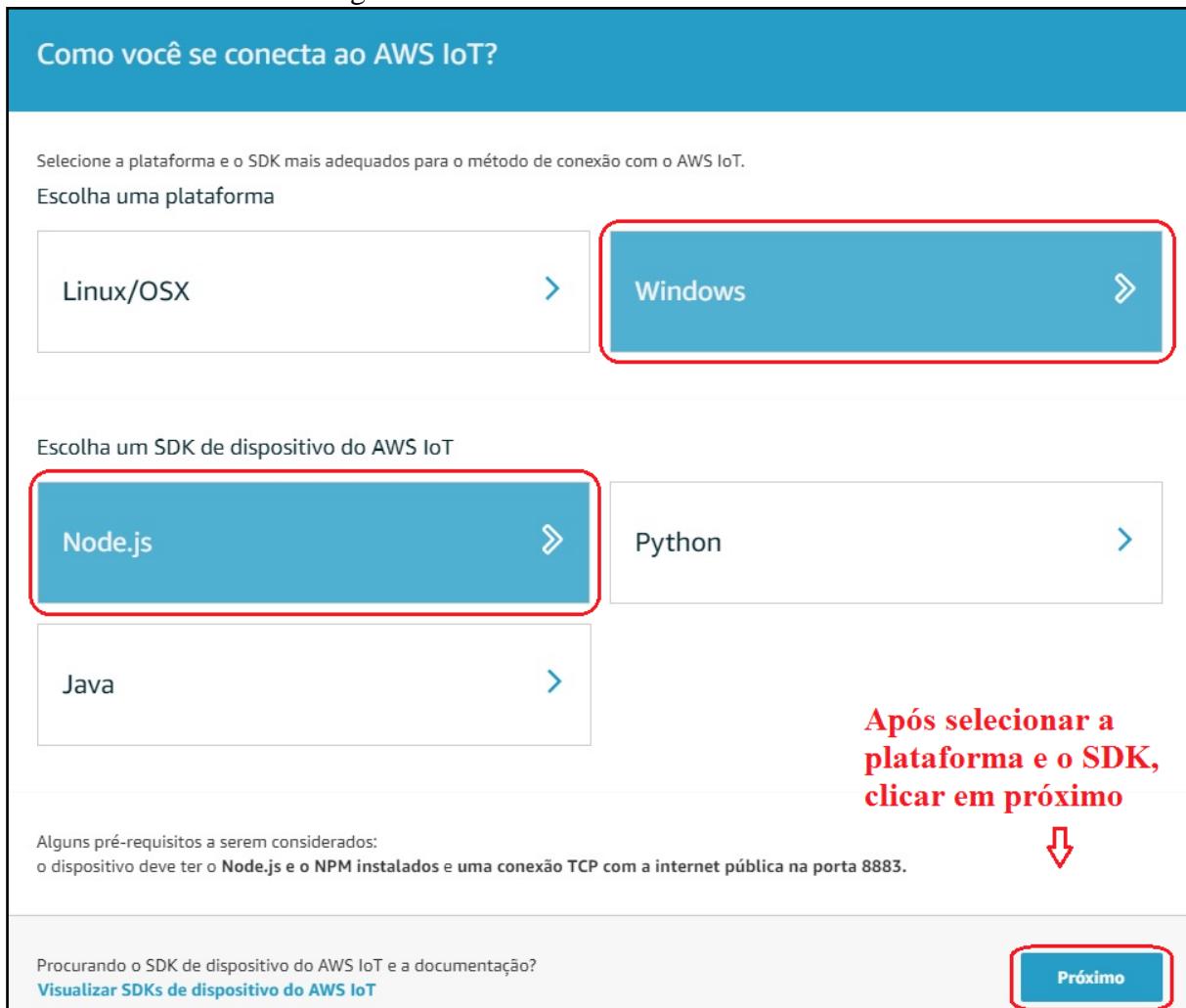
-  **Registrar um dispositivo**
Coisa é a **representação e o registro** de seu dispositivo físico na nuvem. Qualquer dispositivo físico precisa de um registro de coisa para funcionar com o AWS IoT.
-  **Fazer download de um kit de conexão**
O kit de conexão inclui alguns componentes importantes: **credenciais de segurança, o SDK de sua escolha e um projeto de exemplo**.
-  **Configurar e testar o dispositivo**
Com o kit de conexão, você vai configurar seu dispositivo **transferindo arquivos e executando um script, e testar para verificar se ele está conectado ao AWS IoT corretamente**.

Quer saber mais sobre os componentes do AWS IoT?
[Experimente a visão geral interativa](#)

**Clicar em
"Comece a usar"** ➔ [Comece a usar](#)

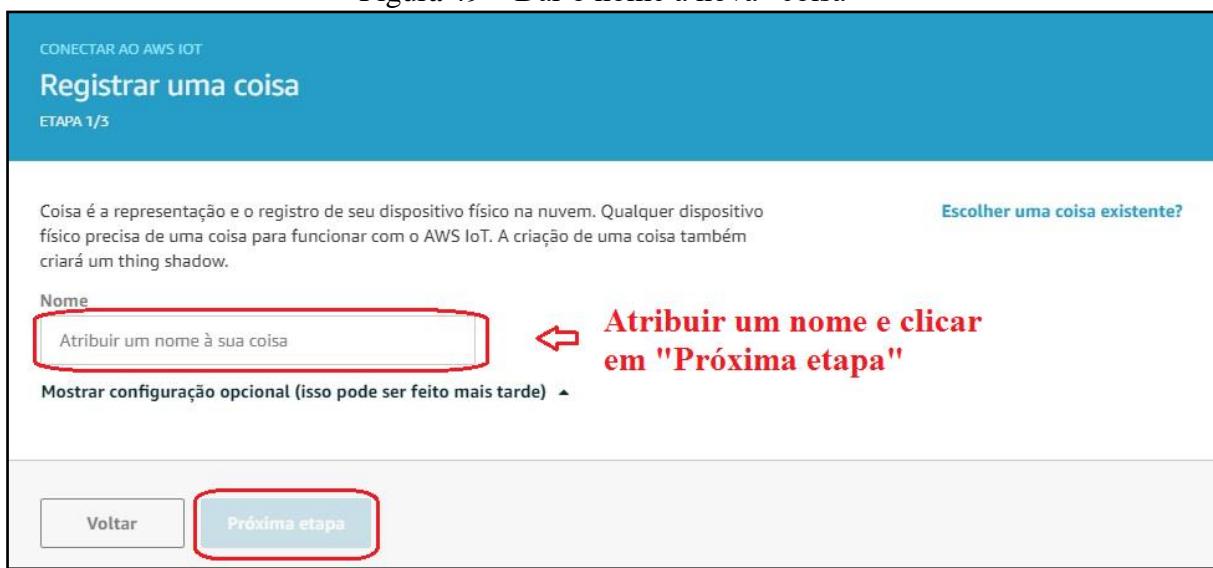
Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 48 – Como se conectar ao AWS IoT



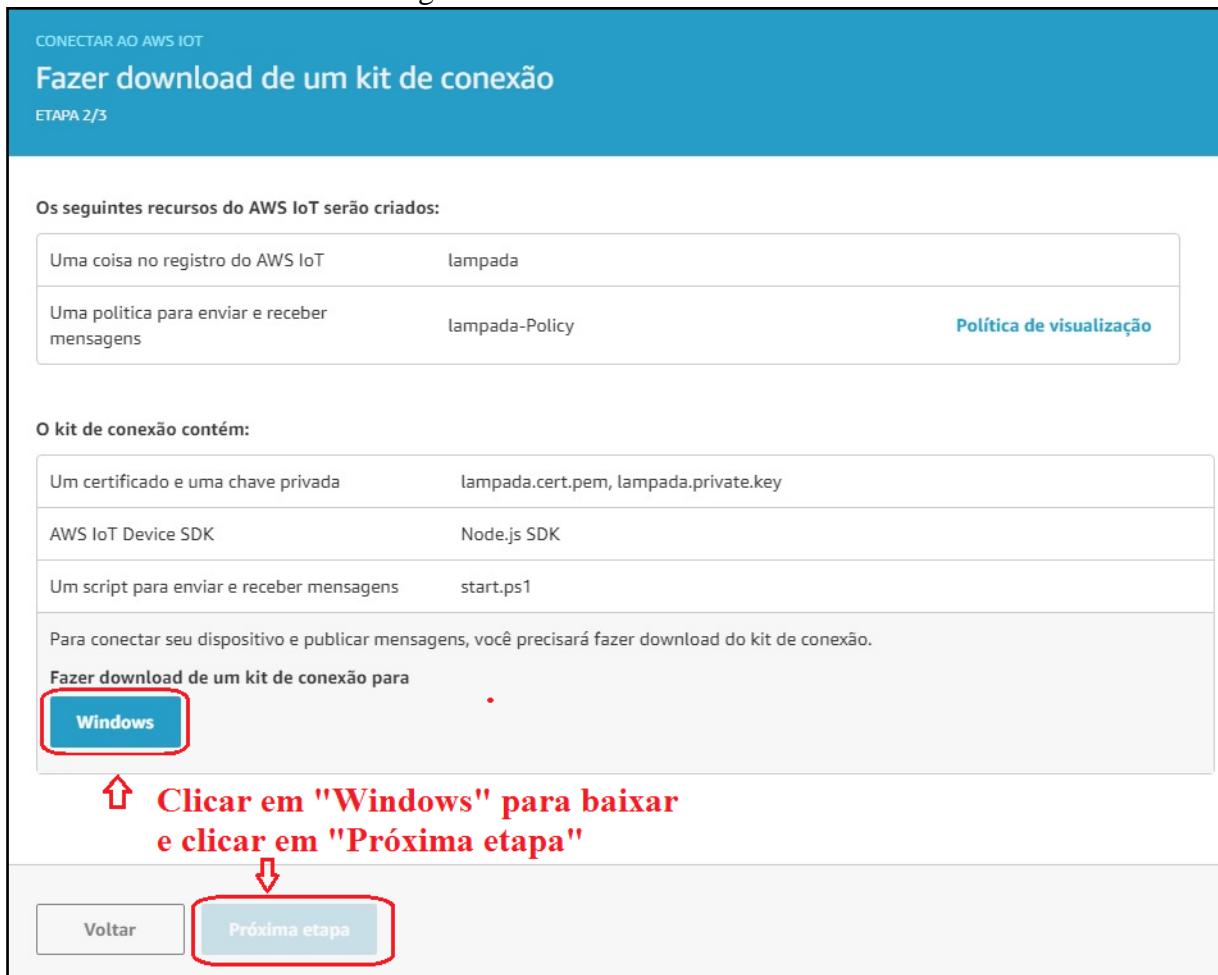
Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 49 – Dar o nome à nova “coisa”



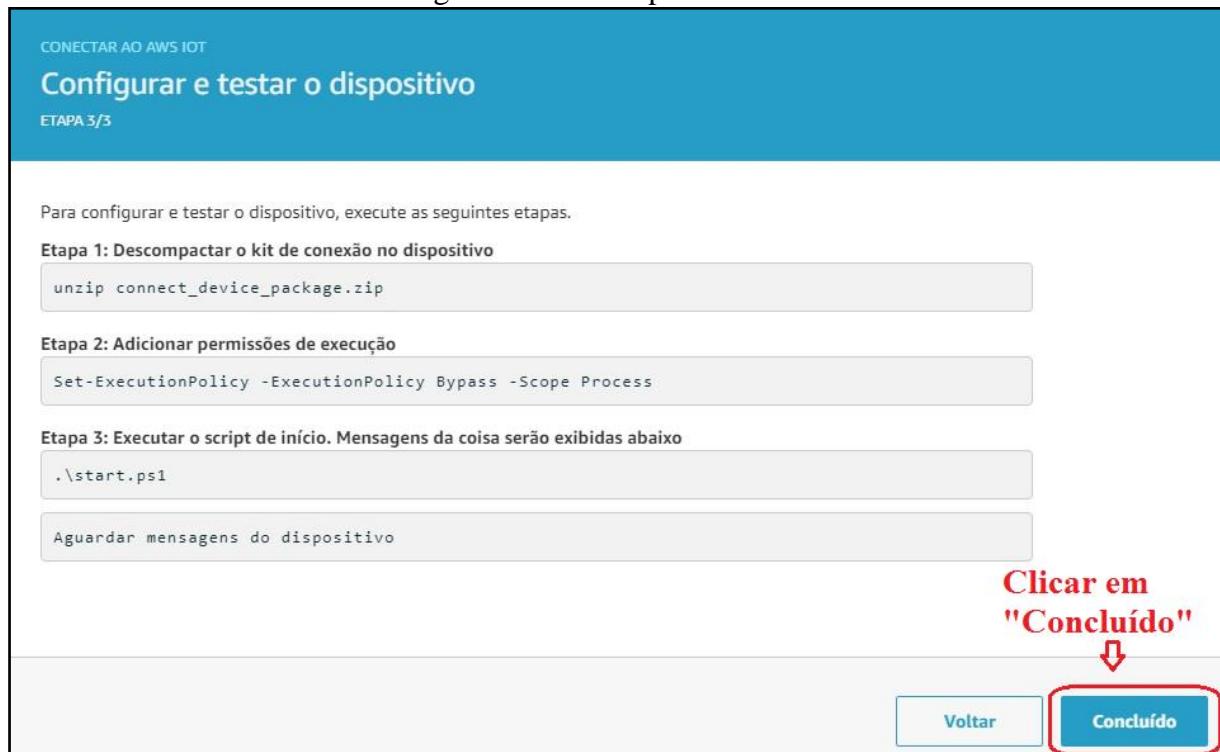
Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 50 – Baixar kit de conexão



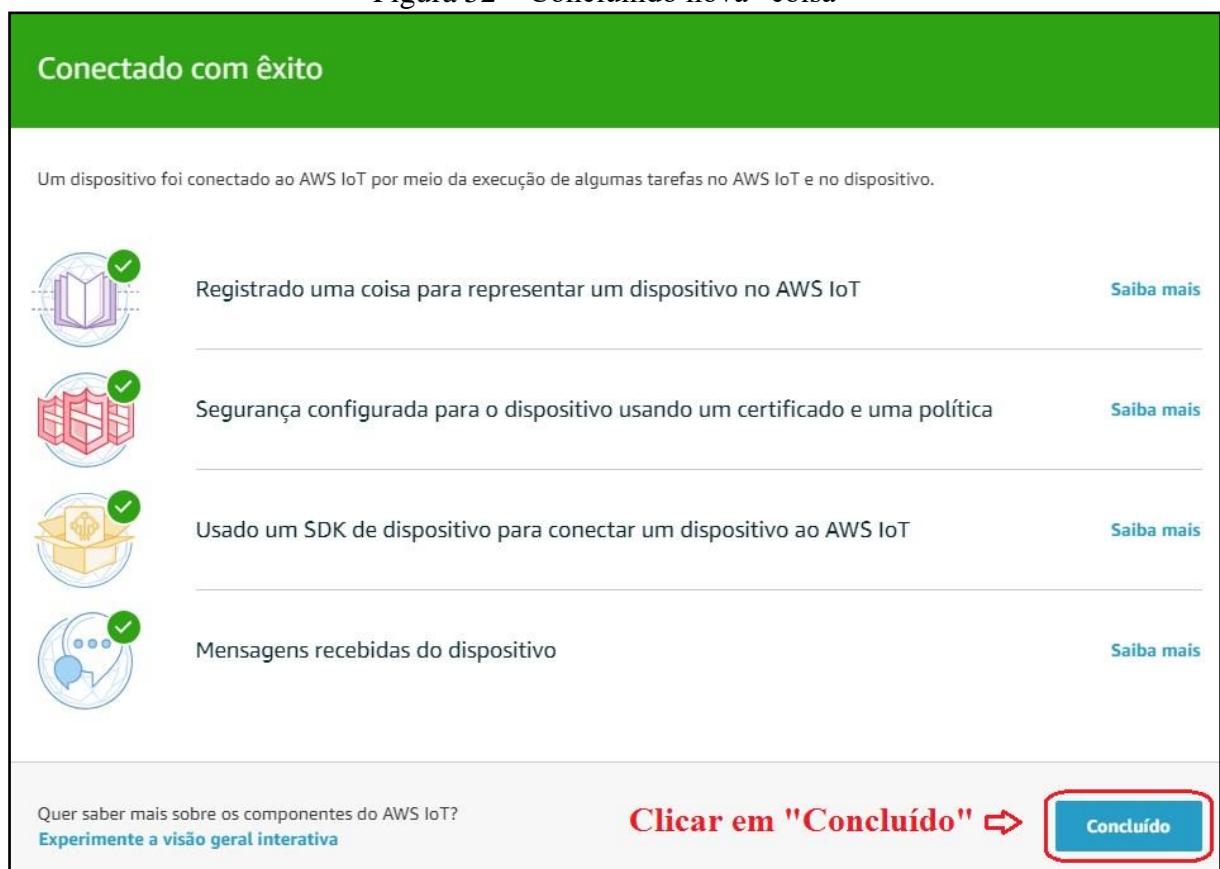
Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 51 – Clicar para concluir



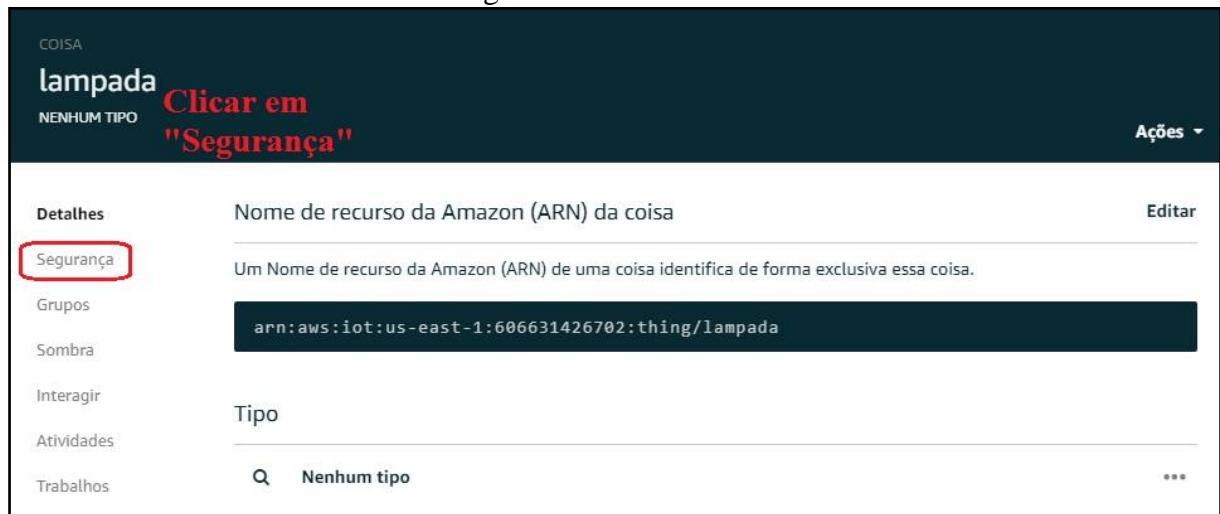
Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 52 – Concluindo nova “coisa”



Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 53 – Nome ARN



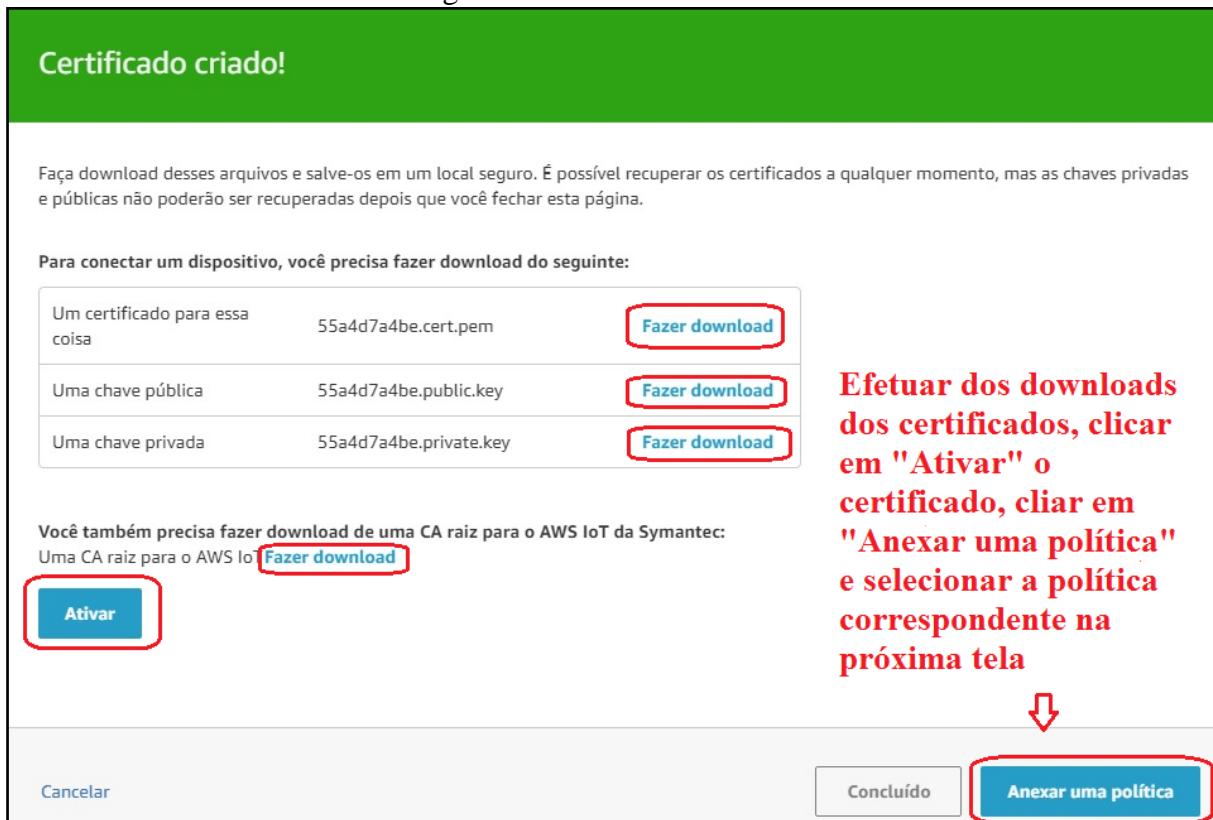
Fonte: AWS IoT (2018) e alterado pelo autor.

Figura 54 – Clicar em “Criar certificado”



Fonte: AWS IoT (2018) e alterado pelo autor.

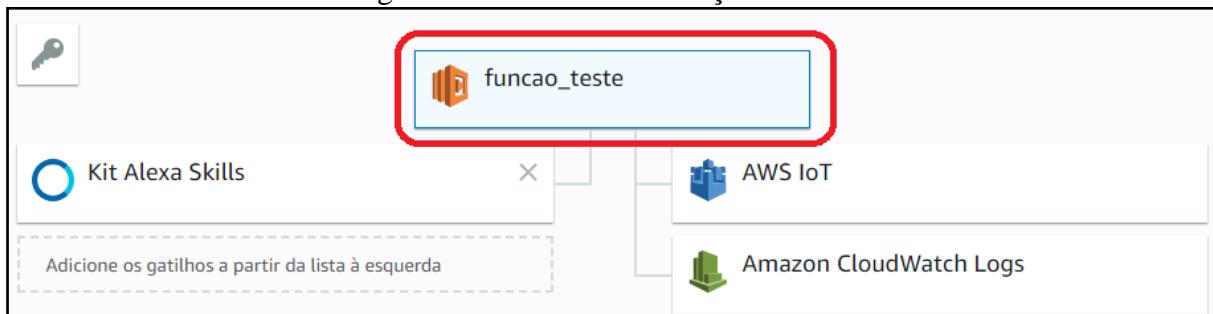
Figura 55 – Certificado criado



Fonte: AWS IoT (2018) e alterado pelo autor.

Voltando na página AWS Lambda (2018), clicar na função, conforme indicado na Figura 56. Descendo na página, substituir o código pelo indicado em Realtek IoT/Arduino Solution (2018c).

Figura 56 – Clicando na função Lambda



Fonte: AWS Lambda (2018) e alterado pelo autor.

Deverá substituir partes do código conforme os Quadro 14 a Quadro 16.

Quadro 14 – Alterar *endpoint* no AWS Lambda

```

10 var AWS = require('aws-sdk');           Substituir pelo endpoint informado em AWS IoT
11 AWS.config.region = "us-east-1";
12 var iotData = new AWS.IotData({endpoint: "a2zweh2b7yb784.iot.us-east-1.amazonaws.com"});
13

```

Fonte: AWS IoT (2018c) e alterado pelo autor.

Quadro 15 – Alterar nome do *intent*

```
84 // Dispatch to your skill's intent handlers
85 if ("ControlLight" === intentName) {
86     setLightInSession(intent, session, callback);
87 } else if ("AMAZON.HelpIntent" === intentName) {
88     getWelcomeResponse(callback);
89 } else if ("AMAZON.StopIntent" === intentName || "AMAZON.CancelIntent" === intentName) {
90     handleSessionEndRequest(callback);
91 } else {
92     throw "Invalid intent";
93 }
94 }
```

Substituir "ControlLight"
por "Lampada"

Fonte: AWS IoT (2018c) e alterado pelo autor.

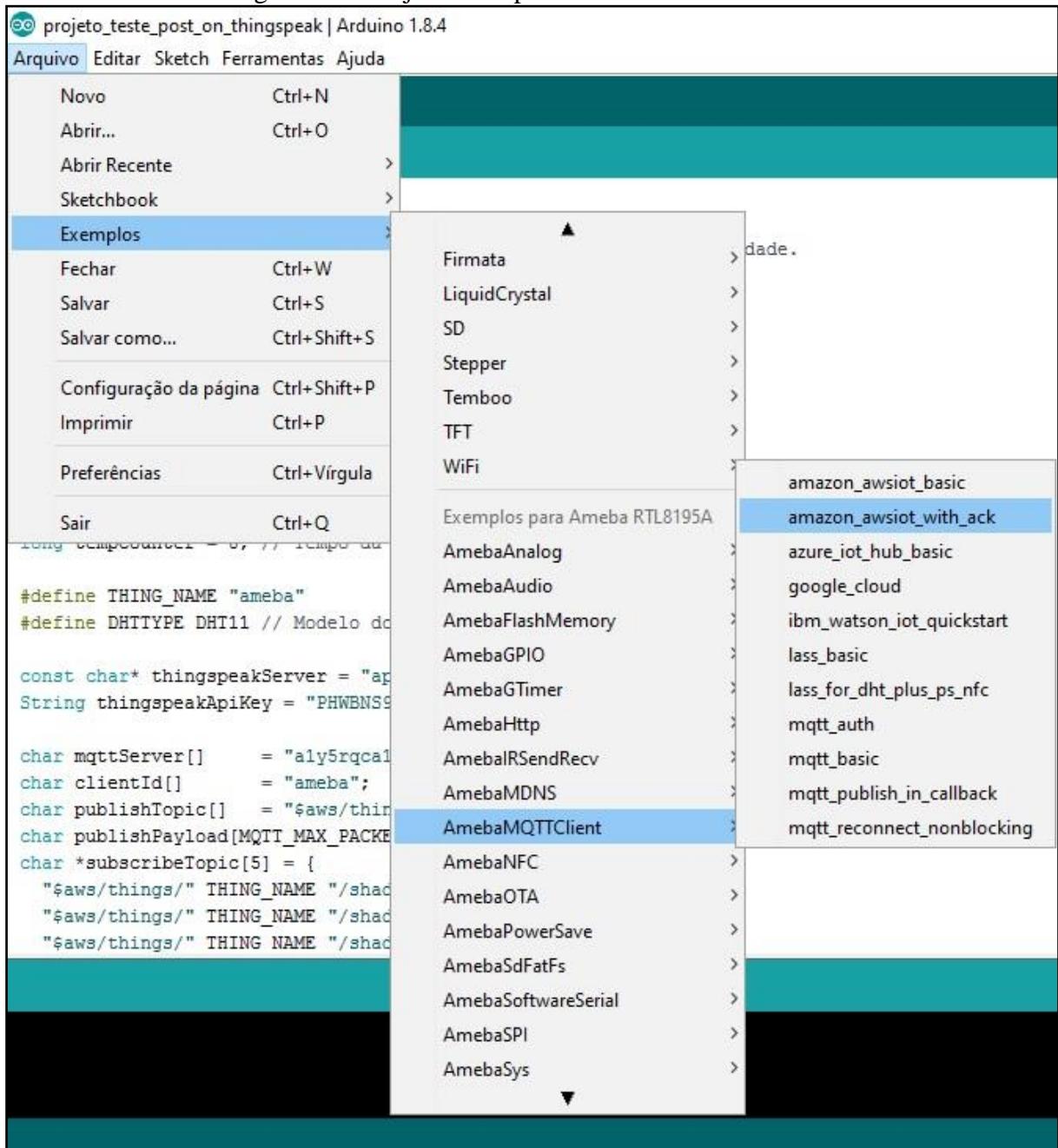
Quadro 16 – Substituir “LightState” e “ameba”

```
136 function setLightInSession(intent, session, callback) {  
137     var cardTitle = intent.name;  
138     var lightStateRequest = intent.slots.LightState; Substituir "LightState" por "states"  
139     var repromptText = "";  
140     var sessionAttributes = {};  
141     var shouldEndSession = true;  
142     var speechOutput = "";  
143  
144     if (lightStateRequest) {  
145         var lightState = lightStateRequest.value;  
146         var paramsUpdate;  
147  
148         if (lightState === "on") {  
149             paramsUpdate = {  
150                 "thingName" : "ameba", Substituir "ameba" por "lampada"  
151                 "payload" : '{"state": {"desired": {"led":1}}}'  
152             };  
153         } else {  
154             paramsUpdate = {  
155                 "thingName" : "ameba", "ameba"  
156                 "payload" : '{"state": {"desired": {"led":0}}}'  
157             };  
158         }  
159     }  
160     callback({  
161         title: cardTitle,  
162         speech: speechOutput,  
163         reprompt: repromptText,  
164         sessionAttributes: sessionAttributes,  
165         shouldEndSession: shouldEndSession  
166     });  
167 }  
168
```

Fonte: AWS IoT (2018c) e alterado pelo autor.

Partindo para o Arduino, será utilizado o exemplo “amazon_awsiot_with_ack”, disponível em Ameba8195 (2016). Abrir o projeto conforme a Figura 57.

Figura 57 – Projeto exemplo do Arduino com o AWS



Fonte: elaborado pelo autor.

A partir da Figura 58 até a Figura 60 é apresentado o que deve ser alterado no projeto exemplo para funcionar a conexão com a Amazon Alexa (2018).

Figura 58 – Variáveis para conectar-se na rede sem fio

```

amazon_awsiot_with_ack | Arduino 1.8.4
Arquivo Editar Sketch Ferramentas Ajuda

amazon_awsiot_with_ack

#include <PubSubClient.h>
|
// Update these with values suitable for your network.
char ssid[] = "yourNetwork"; // your network SSID (name)
char pass[] = "secretPassword"; // your network password
int status = WL_IDLE_STATUS; // the Wifi radio's status

WiFiSSLClient wifiClient;
PubSubClient client(wifiClient);

#define THING_NAME "ameba" // Alterar para o nome dado "lampada"

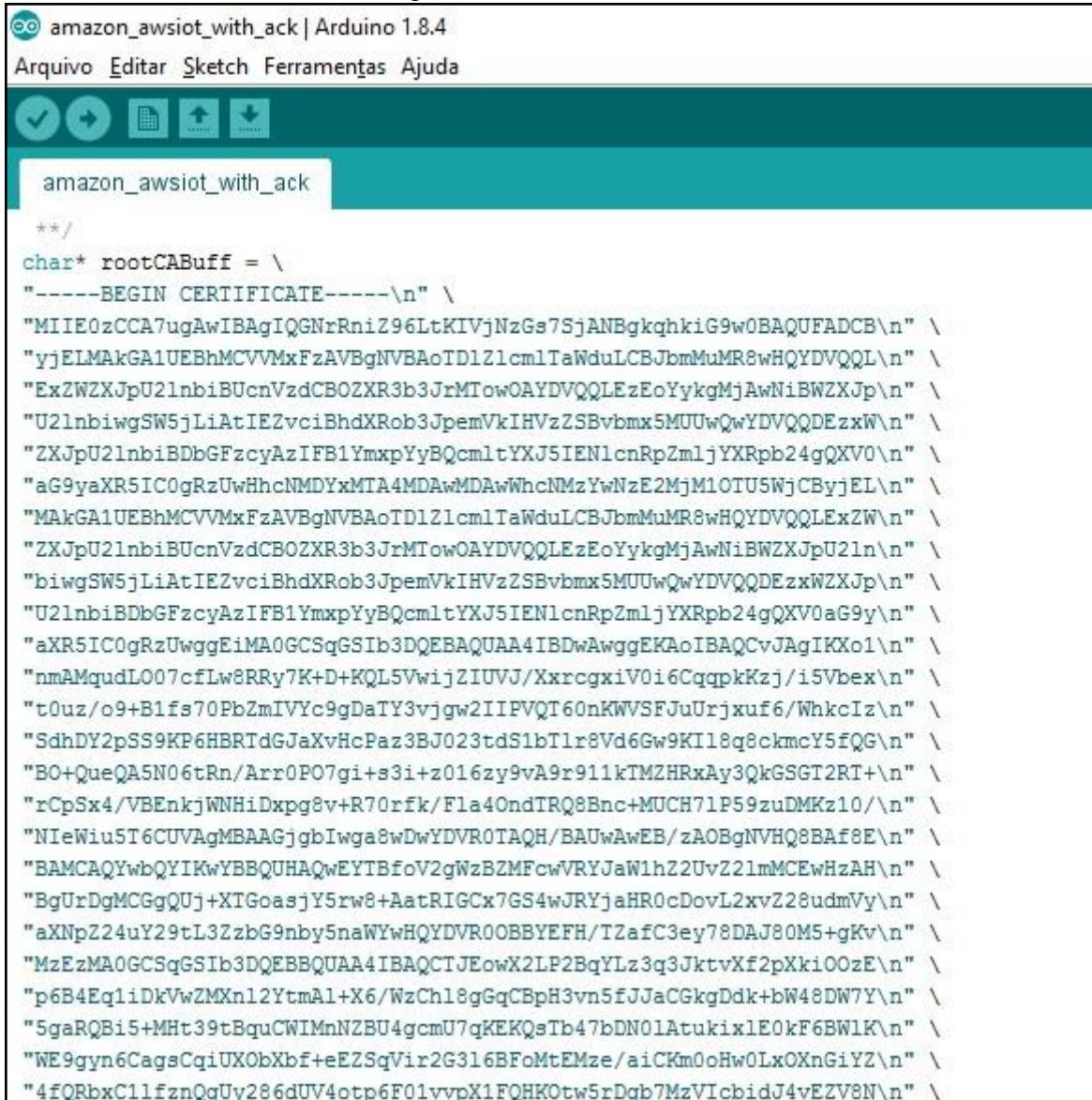
char mqttServer[] = "az2weh2b7yb784.iot.ap-southeast-1.amazonaws.com"; // Alterar para endpoint informado em AWS IoT
char clientid[] = "amebaClient";
char publishUpdateTopic[] = "$aws/things/" THING_NAME "/shadow/update";
char publishGetTopic[] = "$aws/things/" THING_NAME "/shadow/get";
char publishPayload[MQTT_MAX_PACKET_SIZE];
char *subscribeTopic[5] = {
    "$aws/things/" THING_NAME "/shadow/update/accepted",
    "$aws/things/" THING_NAME "/shadow/update/rejected",
    "$aws/things/" THING_NAME "/shadow/update/delta",
    "$aws/things/" THING_NAME "/shadow/get/accepted",
    "$aws/things/" THING_NAME "/shadow/get/rejected"
};

```

Fonte: Ameba8195 (2016) e adaptado pelo autor.

Alterar a variável `rootCABuff` da Figura 59 para o certificado disponível em Symantec (2018).

Figura 59 – Certificado CA



```

  File Edit Sketch Ferramentas Ajuda
  amazon_awsiot_with_ack | Arduino 1.8.4
  Arquivo Editar Sketch Ferramentas Ajuda
  amazon_awsiot_with_ack

  /**
  char* rootCABuff = \
  "-----BEGIN CERTIFICATE-----\n" \
  "MIIE0zCCA7ugAwIBAgIQGNrRniZ96LtkIVjNzG87SjANBgkqhkiG9w0BAQUFADC\n" \
  "yjELMAkGA1UEBhMCVVMxFzAVBgNVBAoTD1Z1cm1TaWduLCBjbmMuMR8wHQYDVQQL\n" \
  "ExZWZXJpU21nbiBUcnVzdCBOZXR3b3JrMTowOAYDVQQLEzEoYykMjAwNiBWZXJp\n" \
  "U21nbiwgSW5jLiAtIEZvciBhdXRb3JpemVkJHvzZSBvbmx5MUUwQwYDVQQDEzxW\n" \
  "ZXJpU21nbiBDbGFzcyAzIFB1YmwpYyBQcm1tYXJ5IEN1cnRpZmljYXRpb24gQXV0\n" \
  "aG9yaXR5IC0gRzUwHhcNMDYxMTA4MDAwMDAwWhcNMzYwNzE2MjM1OTU5WjCBjEL\n" \
  "MAkGA1UEBhMCVVMxFzAVBgNVBAoTD1Z1cm1TaWduLCBjbmMuMR8wHQYDVQQLEzW\n" \
  "ZXJpU21nbiBUcnVzdCBOZXR3b3JrMTowOAYDVQQLEzEoYykMjAwNiBWZXJpU21n\n" \
  "biwgSW5jLiAtIEZvciBhdXRb3JpemVkJHvzZSBvbmx5MUUwQwYDVQQDEzxWZXJp\n" \
  "U21nbiBDbGFzcyAzIFB1YmwpYyBQcm1tYXJ5IEN1cnRpZmljYXRpb24gQXV0aG9y\n" \
  "aXR5IC0gRzUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQCVJAgIKXo1\n" \
  "nmAMqudL007cfLw8RRy7K+D+KQL5VwijZIUVJ/XxrcgxiV0i6CqqpkKzj/i5Vbex\n" \
  "t0uz/o9+B1fs70PbZmIVYc9gDaTY3vJgw2IIPVQT60nKWVSFJuUrjxuf6/WhkcIz\n" \
  "SdhDY2pSS9KP6HBRTdGJaXvHcPaz3BJ023tdS1bT1r8Vd6Gw9KI18q8ckmcY5fQG\n" \
  "BO+QueQA5N06tRn/Arr0PO7gi+s3i+z016zy9vA9r911kTMZHRxAy3QkGSGT2RT+\n" \
  "rCpSx4/VBEnkjWNHiDxpg8v+R70rfk/Fla40ndTRQ8Bnc+MUCH71P59zuDMKz10/\n" \
  "NIeWiu5T6CUVAgMBAAGjgbIwga8wDwYDVR0TAQH/BAUwAwEB/zAOBgNVHQ8BAf8E\n" \
  "BAMCAQYwbQYIKwYBBQUHAQwEYTBFoV2gWzBZMFcwVRYJaW1hZ2UvZ21mMCEwHzAH\n" \
  "BgUrDgMCGgQUj+XTGoasjY5rw8+AatRIGCx7GS4wJRYjaHR0cDovL2xvZ28udmVy\n" \
  "aXNpZ24uY29tL3ZzbG9nby5naWYwHQYDVR0OBByEFH/TZafC3ey78DAJ80M5+gKv\n" \
  "MzEzMA0GCSqGSIb3DQEBBQUAA4IBAQCTJEowX2LP2BqYLz3q3JktvXf2pXkiOOzE\n" \
  "p6B4Eq1iDkVw2MXn12YtmA1+X6/WzCh18gGqCBpH3vn5fJJaCGkgDdk+bW48DW7Y\n" \
  "5gaRQBi5+Mht39tBquCWIMnNZBU4gcmU7qKEKQsTb47bDN01Atukix1E0kF6BW1K\n" \
  "WE9gyn6CagsCqiUXObXbf+eEZSqVir2G316BFoMtEMze/aiCKm0oHw0LxOXnGiYZ\n" \
  "4fQRbxC11fznQgUy286dUV4otp6F01vvpx1FQHK0tw5rDgb7MzViCbidJ4vEZV8N\n"

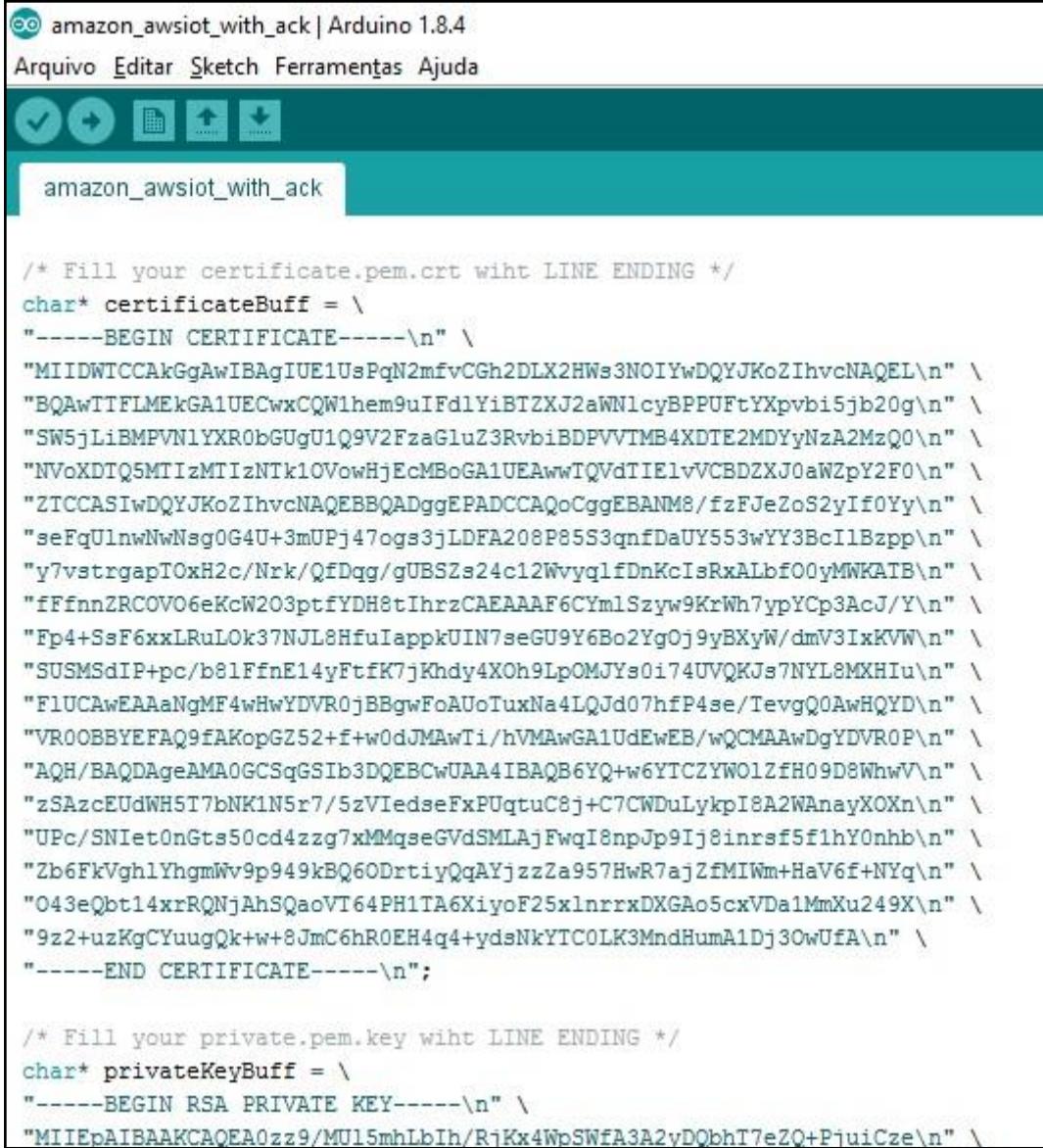
```

Fonte: Ameba8195 (2016) e adaptado pelo autor.

Ao configurar a “coisa” em AWS IoT (2018), foi efetuado o download de arquivos.

Nestes arquivos, procurar o `certificate.pem.crt` e o `private.pem.key` e abrir como texto, copiar o conteúdo deles e colar nas variáveis `certificateBuff` e `privateKeyBuff`, conforme a Figura 60.

Figura 60 – Variáveis que armazenam o certificado e a chave privada



```

  File  Edit  Sketch  Ferramentas  Ajuda

amazon_awsiot_with_ack

/*
 * Fill your certificate.pem.crt wiht LINE ENDING *
char* certificateBuff = \
"-----BEGIN CERTIFICATE-----\n" \
"MIIDWTCCAkGgAwIBAgIUE1UsPqN2mfvCGh2DLX2HWs3NOIYwDQYJKoZIhvcNAQEL\n" \
"BQAwTTFLMEkGA1UECwxCQWlhem9uIFd1YiBTZXJ2aNlcYBPUFtYXpvbi5jb20g\n" \
"SW5jLiBMPVN1YXR0bGUgU1Q9V2FzaGlz3Rvb1BDPVVTMB4XDTE2MDYyNzA2MzQ0\n" \
"NVoXDTQ5MTIzMzNTk10VowHjEcMBoGA1UEAwwTQVdTIE1vvCBDZXJ0aWZpY2F0\n" \
"ZTCCASIWdQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANM8/fzFJeZoS2yIf0Yy\n" \
"seFqUlnwNwNsg0G4U+3mUPj47ogs3jLDF208P85S3qnfDaUY553wYY3BcI1Bzpp\n" \
"y7vstrgapTOxH2c/Nrk/QfDqg/gUBSZs24c12WvyqlfDnKcIsRxALbfO0yMWKATB\n" \
"fFFnnZRCOV06eKcW2O3ptfYDH8tIhrzCAEAAAF6CYmlSzyw9KrWh7ypYCp3AcJ/Y\n" \
"Tp4+SsF6xxLRuLOk37Njl8HfuIappkUIN7seGU9Y6Bo2YgOj9yBXyW/dmV3IxKVW\n" \
"SUSMSdIP+pc/b81FfnE14yFtfK7jKhdy4XOh9LpOMJYs0i74UVQKJs7NYL8MXHIu\n" \
"FlUCAwEAAsNgMF4whYDVR0jBBgwFoAUoTuxNa4LQJd07hfP4se/TevgQ0AwHQYD\n" \
"VR0OBByEFAQ9fAKopGZ52+f+w0dJMAwTi/hVMAwGA1UdEwEB/wQCMAAwDgYDVR0P\n" \
"AQH/BAQDAgeAMA0GCSqGSIb3DQEBCwUA4IBAQB6YQ+w6YTCZYWO1zfH09D8WhwV\n" \
"zSAzcEUdWH5T7bNK1N5r7/5zVIedseFxPUqtuC8j+C7CWDuLykpI8A2WAnayXOXn\n" \
"UPc/SNIet0nGts50cd4zzg7xMMqseGVdSMLAjFwqI8npJp9Ij8inrsf5f1hY0nhb\n" \
"Zb6FkVgh1YhgmWv9p949kBQ60DrtyQqAYjzz2a957HwR7ajZfMIWm+HaV6f+NYq\n" \
"043eQbt14xrRNjAhSQaoVT64PH1TA6XiyoF25xlnrrxDXGAo5cxVDa1MmXu249X\n" \
"9z2+uzKgCYuugQk+w+8JmC6hR0EH4q4+ydsNkYTC0LK3MndHumA1Dj30wUfA\n" \
"-----END CERTIFICATE-----\n";

/*
 * Fill your private.pem.key wiht LINE ENDING *
char* privateKeyBuff = \
"-----BEGIN RSA PRIVATE KEY-----\n" \
"MIIEpAIBAAKCAQEA0zz9/MU15mhLbIh/RjKx4WpSWfA3A2vDQbhT7eZQ+PiuiCze\n" \

```

Fonte: Ameba8195 (2016) e adaptado pelo autor.

Assim a Amazon Alexa (2018) está pronta para acionar a GPIO 10 do Arduino. Basta conectar um LED a esta saída.