Guilherme Vinícius Barbosa Pereira Amorim

Protótipo de sistema embutido de baixo custo integrado ao Amazon Voice Service

Belo Horizonte

2022

Guilherme Vinícius Barbosa Pereira Amorim

Protótipo de sistema embutido de baixo custo integrado ao Amazon Voice Service

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

Sumário

1 Introdução

Atualmente o termo ‘sistemas embarcados’ não está muito presente no cotidiano brasileiro, mas essa tecnologia é responsável por fornecer equipamentos inteligentes como semáforos de trânsito, relógios, respiradores mecânicos, roteadores e aparelhos de ar-condicionado, por exemplo. Em termos simples, pode-se definir um sistema embarcado como um dispositivo controlado por um computador encapsulado. Ou seja, um sistema embarcado é na verdade um sistema microprocessado. Dentre as principais vantagens de sistemas embarcados, destaca-se o baixo custo, a eficiência e a facilidade de programação (NORLETO, 2020).

O primeiro sistema embarcado é o AGC (Apollo Guidance Computer). Ele foi desenvolvido nos EUA por Charles Stark Draper do MIT em 1966 (EMBARCADOS, 2014). Desde então, os sistemas embarcados ficaram mais acessíveis, mais velozes e mais compactos.

Com o passar dos anos, novas tecnologias foram adicionadas à sistemas embarcados, como por exemplo o Wifi, o Bluetooth etc. Na última década foi observado a ascensão de assistentes virtuais como a Alexa e o Google Assistente. A Alexa foi criada em 2014 e apareceu pela primeira vez como parte das caixas de som. Hoje a Alexa, a partir de comandos de voz, pode realizar pesquisas, mandar executar uma lista de músicas, disparar um alarme etc (VIGLIAROLO, 2017). Segundo o site oficial da Amazon Alexa (2022), o serviço permite a conexão com dispositivos, efetuar comandos por voz, interpretá-los e tomar uma ação correspondente. Isso tudo acontece por meio do Web Service da Amazon (AWS).

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.

Visto que as tecnologias Amazon AWS e Amazon Alexa são recente, ainda não se observa uma variedade de dispositivos, não comercializados pela Amazon e pela Google, que fazem o uso de linguagem natural. Com a baixa oferta de produtos com essa tecnologia, produtos simples - como interruptores - estão sendo comercializados por preços não acessíveis ao mercado global.

Diante do que foi dito, este trabalho se propõe criar um dispositivo de baixo custo integrado à tecnologia AVS e serviços em nuvem.

1.1 Estrutura

O trabalho será apresentado em cinco capítulos. O capítulo 1 apresenta a introdução e os objetivos. O capítulo 2 apresenta o referencial teórico, contextualizando o projeto e citando trabalhos correlatos. O capítulo 3 contém o desenvolvimento do trabalho, apresentando requisitos funcionais e não-funcionais, ferramentas utilizadas e a implementação. O capítulo 4 expõe e faz a análise dos resultados. O capítulo 5, por fim, conclui o trabalho listando vantagens e desvantagens do protótipo, e as sugestões para trabalhos futuros.

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

- Acesso a tecnologia de sensores de baixo custo e baixa potência;
- Conectividade;
- Plataformas de computação em nuvem;
- Machine learning e análise avançada;
- Processamento de linguagem natural (AVS, 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:

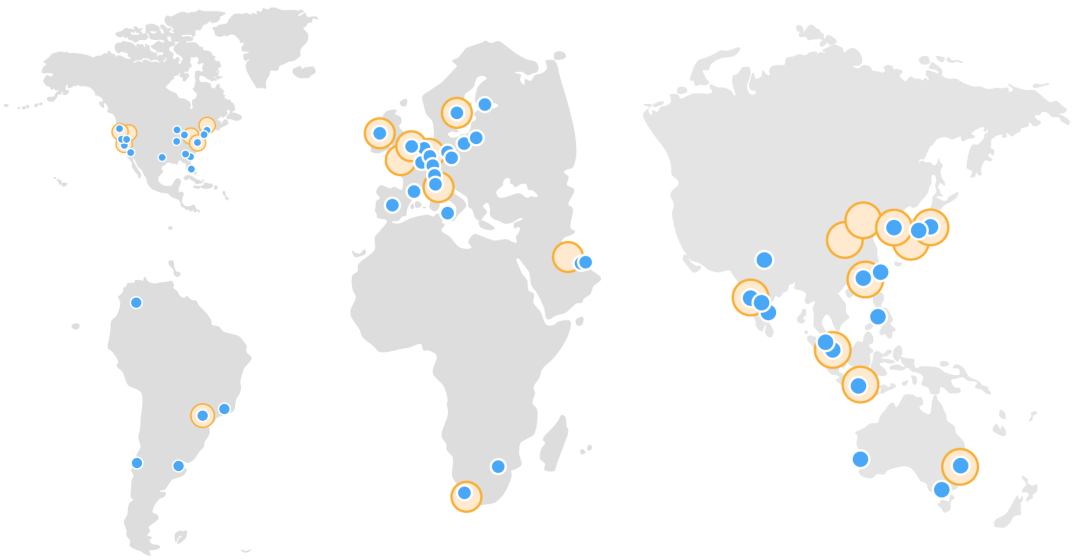
- Teclado, botões e microfones;
- Alarmes, visores e alto-falantes;
- Sensores;
- Atuadores;

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 precificação. 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 84 zonas de disponibilidade em 26 regiões geográficas. Esse modelo de região e zona de disponibilidade da AWS foi reconhecido pelo Gartner, empresa de pesquisa e consultoria em tecnologia da informação (TI), como o método recomendado para executar aplicativos corporativos que exigem alta disponibilidade (AWS, 2022). As Figura 1 contém mapas com as redes de presença da AWS, por região.

Figura 1 – AWS Presence Networks.



Fonte: Author.

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:

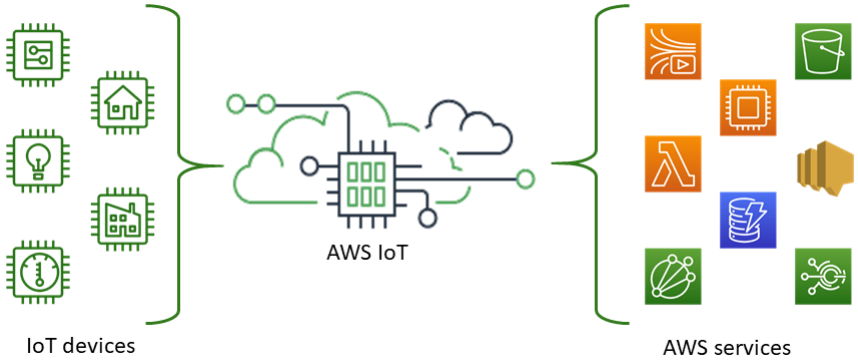
- Console da web;
- Ferramenta de linha de comando;
- Ambiente de desenvolvimento integrado (IDE);

- Kit de desenvolvimento de software (SDK);
- Infraestrutura como código.

2.3 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 2.

Figura 2 – 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:

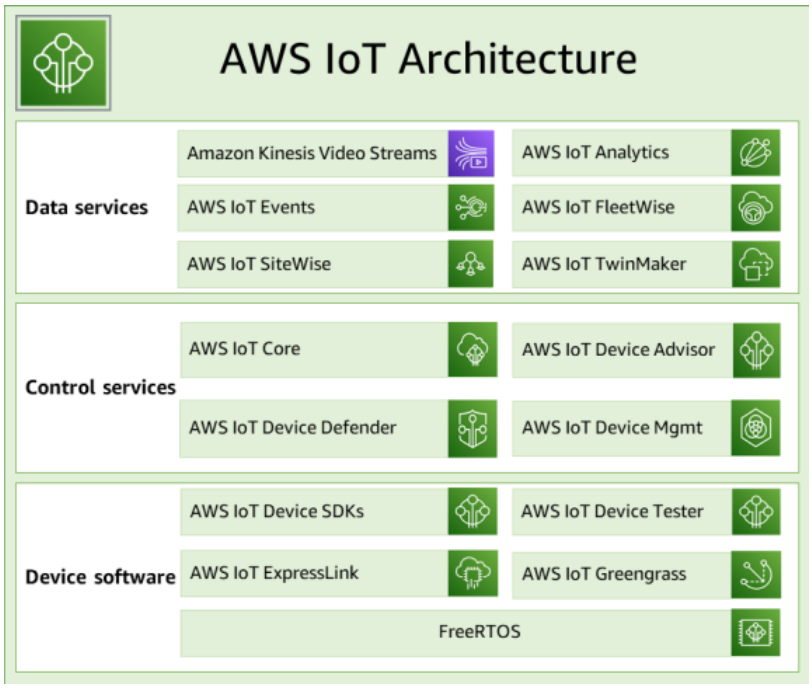
- LoRaWAN e dispositivos;
- Arduino;
- Raspberry PI;
- Dispositivos de IoT personalizados.

Para a completa integração desses dispositivos com serviços AWS e usuários, alguns componentes são esseciais. 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 melhor detalhado na sessão seção 2.6. 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, que será melhor detalhado na sessão seção 2.4. Alguns exemplos de outros serviços fornecidos pelo AWS IoT podem ser vistos na Figura 3.

Figura 3 – AWS IoT architecture.



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

O serviço IoT Core será melhor detalhado na sessão seção 2.5. a

2.4 Amazon FreeRTOS

O FreeRTOS é um kernel de sistema operacional em tempo real de código aberto para microcontroladores e distribuído sob a licença do MIT. O seu logo pode ser visto na Figura 4. Originalmente, o FreeRTOS foi desenvolvido em 2003 por Richard Barry e posteriormente mantido pela empresa Real Time Engineers Ltd. Em 2017, a AWS assumiu a administração do kernel. Além de ser um kernel de código aberto, a distribuição do

FreeRTOS conta com demonstrações e documentação para diferentes compiladores. Essas características permitem um rápido desenvolvimento dispositivos.

Figura 4 – Logo FreeRTOS.

Fonte: Wikipedia (2022).

O FreeRTOS surgiu com o propósito de ser simples, compacto, ágil e com baixo consumo de energia. Essa característica o torna o kernel escolhido por diversos dispositivos de IoT. Sua estrutura é baseada em uma arquitetura de rotinas programáveis (*scheduler routines*). Ele também é capaz de fazer alocações de memória estaticamente e/ou dinamicamente. Atualmente, o FreeRTOS conta com métodos que possibilitam a implementação dos seguintes componentes e *features*:

a) *Threads*;

b) *tasks* e suporte para *Coroutines*;

c) Priorização de *Threads*;

d) Mutexes;

e) Semáforos;

f) *timers*;

g) O *scheduler* pode ser configurado para *preemptive* ou *cooperative multitasking*;

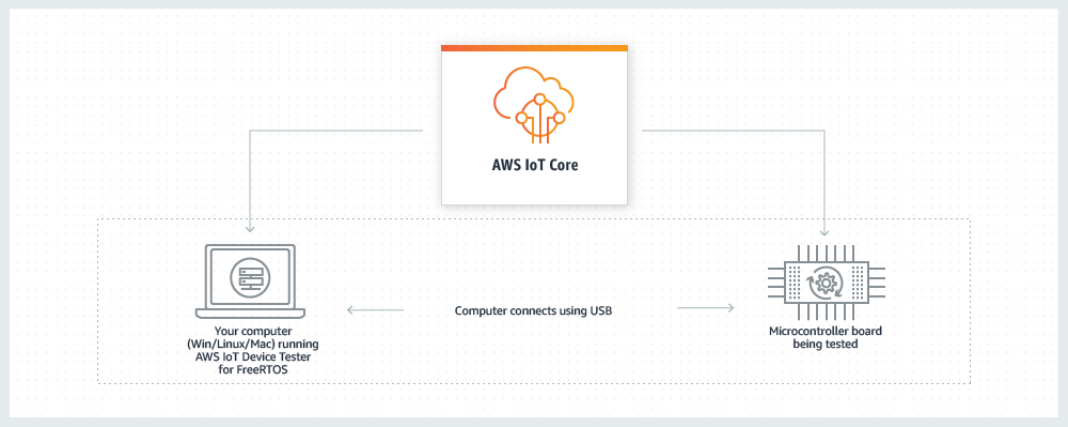
h) modo *tickless*, para aplicações de baixo consumo;

i) Visualizador em tempo de tempo de execução para depuração e verificação.

Em 2022, mais de 90 microcontroladores estão disponíveis no mercado para o desenvolvimento de aplicações baseadas no FreeRTOS. A AWS conta hoje com verificador que permite que o usuário confirme se algum dispositivo é compatível com o seu kernel, o *AWS IoT Device Tester*. Isso possibilita que usuários e empresas testem, inclusive, microcontroladores não disponíveis no mercado. Esse verificador faz uso do do AWS IoT Core, serviço que será melhor explicado na seção 2.5. A Figura 5 mostra o diagrama de conexões para o *AWS IoT Device Tester*. Ademais, a AWS também disponibiliza um simulador do FreeRTOS para Windows que possibilita o teste de software sem hardware.

Durante o processo de desenvolvimento, as bibliotecas necessárias para o projeto podem ser obtidas no próprio site da AWS, GitHub ou FreeRTOS.org. Também é possível acessar o kernel FreeRTOS autônomo por meio de qualquer um desses canais. Vale

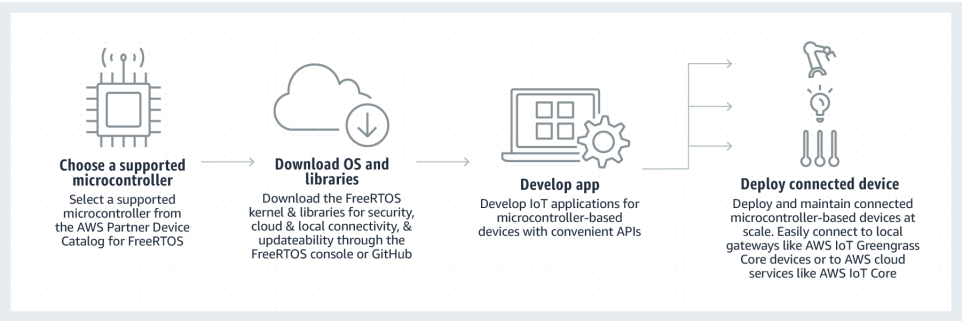
Figura 5 – Diagrama de conexões para o *AWS IoT Device Tester*.



Fonte: Amazon AWS (2022).

lembrar que o FreeRTOS tem código aberto, portanto, o desenvolvedor pode ampliar, modificar ou excluir qualquer uma das bibliotecas de código-fonte. O processo de download personalizado das bibliotecas é orientado por console e interface gráfica na AWS. Um fluxo recomendado pela AWS para o *deploy* de dispositivos usando o FreeRTOS pode ser visto na Figura 6.

Figura 6 – Fluxo recomendado pela AWS para o *deploy* de dispositivos usando o FreeRTOS.



Fonte: Amazon AWS (2022).

2.5 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:

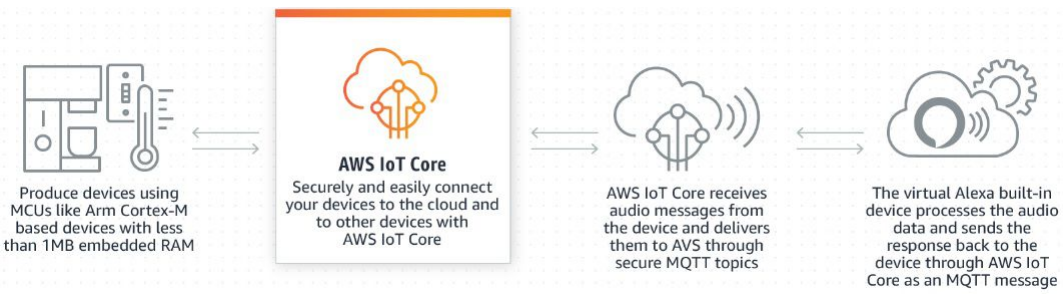
- 2.250.000 minutos de conexão;
- 500.000 mensagens;
- 225.000 operações do Registry ou Device Shadow;
- 250.000 regras acionadas e 250.000 ações executadas.

Dessa forma, o AWS IoT Core é o serviço que permite a integração da AVS à sistemas embarcados. Essa integração 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).

A Figura 7 mostra um diagrama disponibilizado pela AWS que melhor apresenta como o AWS IoT Core permite a integração de sistemas embutidos à AVS:

Figura 7 – Diagrama de integração de sistemas embutidos à AVS por meio do AWS IoT Core.



Fonte: Amazon AWS (2022).

2.6 MQTT

MQTT é um protocolo de mensagens para dispositivos IoT. Ele foi desenvolvido para dispositivos com limitada disponibilidade de largura de banda (*network bandwidth*). Ele deve ser executado em um protocolo de transporte que forneça conexões bidirecionais ordenadas, sem perdas — normalmente, TCP/IP. O logo do MQTT pode ser visto na Figura 8.

Figura 8 – Logo MQTT.

Fonte: Wikimedia (2022).

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 pode ser encontrado na subseção 2.6.1.

- *Subscribe*: Para receber mensagens sobre tópicos de interesse, o cliente envia uma mensagem *Subscribe* para o *broker*. Mais informações sobre esse tipo de mensagem pode ser encontrado na subseção 2.6.2;

- *Suback*: Para confirmar cada assinatura, o *broker* envia uma mensagem de confirmação *Suback* ao cliente. Mais informações sobre esse tipo de mensagem pode ser encontrado na subseção 2.6.2;

- *Unsubscribe*: Esta mensagem exclui as assinaturas existentes de um cliente no *broker*. Mais informações sobre esse tipo de mensagem pode ser encontrado na subseção 2.6.4;

- *Unsuback*: Para confirmar o cancelamento de assinatura, o *broker* envia uma mensagem de confirmação *Unsuback* ao cliente. Mais informações sobre esse tipo de mensagem pode ser encontrado na subseção 2.6.5.

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.

2.6.1 MQTT Publish Message

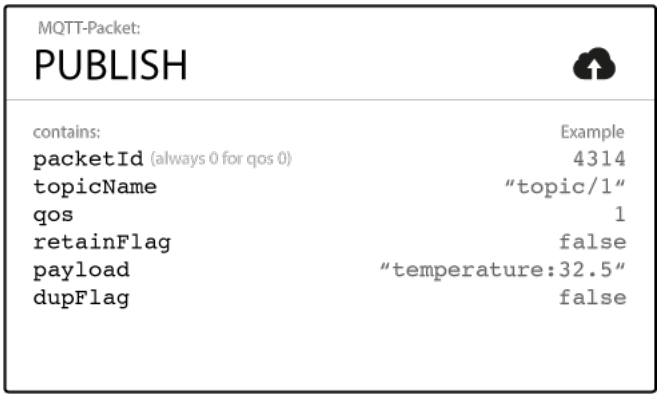
O formato de uma mensagem do tipo *Publish* está apresentado na Figura 9.

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);

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



Fonte: HiveMQ (2015).

- qos: Indica o nível de qualidade de serviço (QoS) da mensagem. Existem três níveis: 0, 1 e 2. O nível de serviço determina que tipo de garantia uma mensagem tem para chegar ao destinatário pretendido (cliente ou *broker*).
- retainFlag: Sinalizador indicando se a mensagem é salva pelo *broker* como o último valor válido conhecido para um tópico 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*.

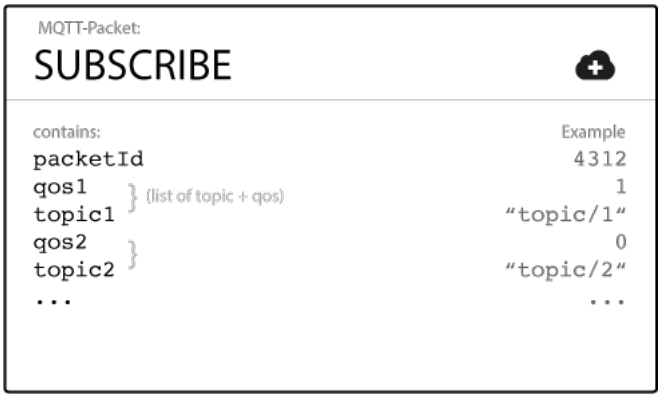
2.6.2 MQTT Subscribe Message

O formato de uma mensagem do tipo *Subscribe* está apresentado na Figura 10. 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:

- 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 (*topici*) e um nível de QoS (*qosi*). 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.

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

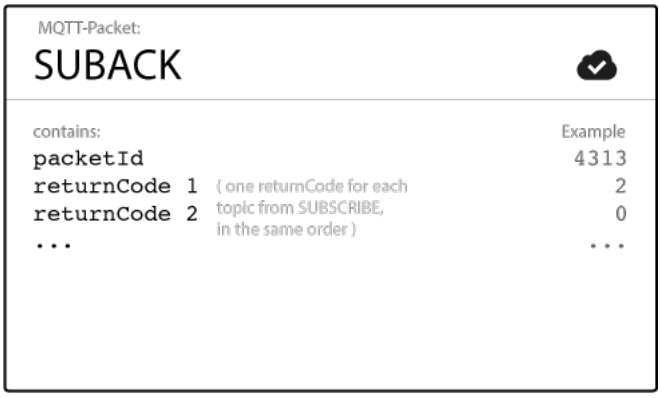


Fonte: HiveMQ (2015).

2.6.3 MQTT Suback Message

O formato de uma mensagem do tipo *Suback* está apresentado na Figura 11. Esta mensagem contém o identificador de pacote da mensagem *Subscribe* original e uma lista de códigos de retorno.

Figura 11 – 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.

2.6.4 MQTT Unsubscribe Message

O formato de uma mensagem do tipo *Unsubscribe* está apresentado na Figura 12. A mensagem *Unsubscribe* é semelhante à mensagem *Subscribe* e possui um identificador de pacote e uma lista de tópicos.

Figura 12 – 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.

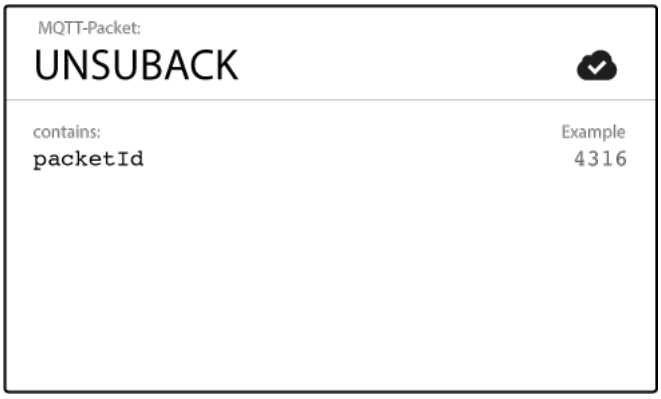
2.6.5 MQTT Unsuback Message

O formato de uma mensagem do tipo *Unsuback* está apresentado na Figura 13. Esta mensagem contém apenas o identificador de pacote da mensagem *Unsubscribe* original.

2.7 Amazon Alexa Voice Service (AVS)

A Amazon Alexa Voice Service é um serviço hospedado em nuvem capaz de integrar diferentes funcionalidades e funções da assistente de voz Alexa em produtos capazes de se conectar à Internet. Esse serviço oferece aos desenvolvedores de dispositivos inteligentes um conjunto de APIs que permitem a integração desses dispositivos com a Alexa. Ao se conectar à Alexa, dispositivos passam a ter acesso a todas as as habilidades da Alexa, também chamadas de *Alexa Skills*. Ademais, o AVR provê aos seus usuários ferramentas

Figura 13 – Atributos de uma mensagem do tipo *Unsuback* para o protocolo MQTT.



Fonte: HiveMQ (2015).

de reconhecimento automático de fala (*automatic speech recognition*, ASR) e compreensão de linguagem natural (*natural language understanding*, NLU). A seção 2.7 mostra o logo da Amazon Alexa.

Figura 14 – Logo da Amazon Alexa.

Fonte: Amazon AWS (2022).

Alguns serviços ofertados pela AVS aos desenvolvedores são:

- *Alexa Built-in Product (ABI)*: Uma categoria de dispositivos AVS com um conjunto de microfones habilitados para *wake word*, alto-falante e um conjunto principal de recursos e funções do Alexa;
- *AVS Device SDK*: Bibliotecas baseadas em C++ que aproveitam as APIs AVS. O AVS Device SDK é uma poderosa ferramenta que permite a rápida integração fr quase todos os recursos e funções do Alexa. Essa SDK É modular e abstrato, fornecendo componentes e APIs personalizáveis.;

- *The Alexa APIs*: As interfaces AVS que correspondem à funcionalidade do cliente. Cada interface contém mensagens agrupadas logicamente chamadas diretivas (da nuvem) e eventos (do seu cliente). O AVS também fornece APIs para etapas de implementação como autorização e HTTP/2.

A seguir tem-se os casos mais comuns de uso da Alexa e que, a partir de recursos da AVS, agora serão acessíveis em dispositivos com a Alexa integrada:

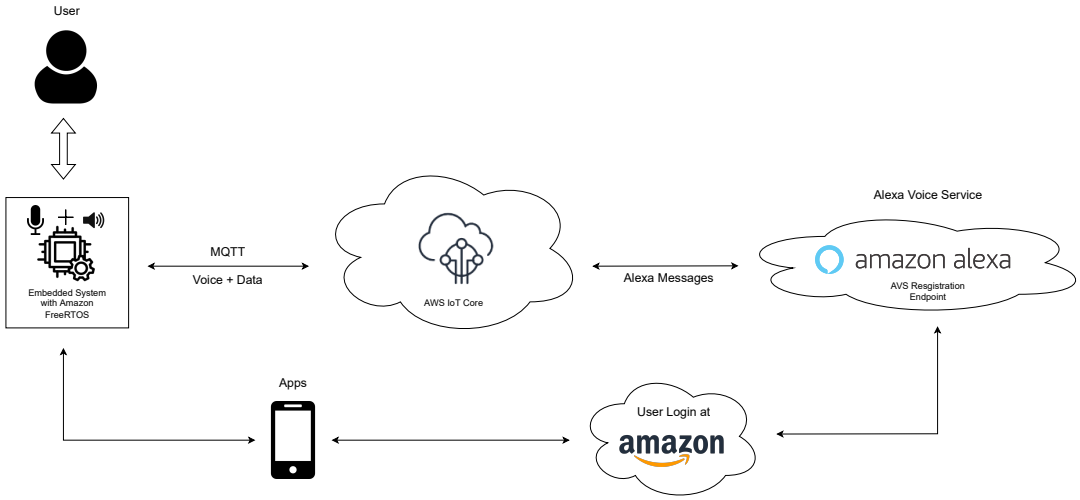
- Configuração e autenticação da Alexa;
- Acionamento da Alexa por voz ou toque;
- Interrompa o Alexa por voz ou toque;
- Transmissão e controle de músicas de vários provedores de serviços de música, como o Spotify;
- Sincronização de dispositivos para a audição sincronizada de músicas;
- Criação e cancelamento de *timers*, alarmes e lembretes;
- Receber, excluir e desativar notificações;
- Emparelhar com outros dispositivos através de Bluetooth;
- Mantenha contato com chamadas, mensagens e anúncios;
- Alternar quaisquer configurações disponíveis;
- Saiba quando o Alexa está ouvindo, pensando e respondendo por meio de um sistema de atenção;

2.8 Trabalhos correlatos

3 Metodologia

Dentre os diversos serviços fornecidos pela AWS, o projeto fará uso do AWS IoT Core, serviço de integração da AVS com sistemas IoT. A Figura 15 mostra o diagrama de integração do sistema embarcado (dispositivo IoT) à AVS, por meio da tecnologia AWS IoT Core.

Figura 15 – Diagrama de uso dos serviços AWS e da AVS no projeto.



Fonte: Autor (2022).

3.1 Objetivos

Este trabalho tem o objetivo de criar um sistema embutido de baixo custo integrado aos serviços em nuvem da AWS à assistente de voz Alexa. Para esse dispositivo, tem-se os seguintes requisitos funcionais:

a) Suporte à rede Wifi IEEE 802.11 (2,4 GHz);

b) Acionamento da Alexa por voz ou toque;

c) Comunicação com o usuário em linguagem natural, a partir de caixas de som e microfones;

d) Ser configurado pelo aplicativo Amazon Alexa;

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

a) Velocidade a definir;

b) Preço a definir;

Processor	ARM M7 or equivalent Arm M4 + AFE DSP
RAM	MB for ARM M7 500KB for M4 + AFE DSP
Target OS	FreeRTOS
Connectivity	MQTT over Wi-Fi
# of Microphones	2+
Speaker	Optimized for speech playback

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

c) Confiabilidade;

d) Capaz de entender o usuário em ambientes ruidosos (Como medir isso?);

3.2 Projeto de Hardware

Para a escolha do hardware do projeto, primeiro estudou-se os requisitos mínimos recomendados pela Amazon. Os parâmetros mínimos recomendados pela AWS podem ser vistos na Tabela 1.

Para a prototipagem, alguns kits de desenvolvimento disponíveis no mercado foram estudados. A **??** mostra uma tabela comparando 4 dispositivos.

4 Resultados e Discussão

5 Conclusões