MICROCHIP MASTERs 2019

IoT2019

Como controlar seu dispositivo IoT utilizando Amazon Alexa

SMART | CONNECTED | SECURE

MICROCHIP

Objetivo

Ao término desta apresentação você será capaz de:

- Entender como funciona e como conectar cada elemento necessário para controlar um dispositivo IoT com conexão WiFi utilizando comandos de voz através do Alexa Skill Kit (ASK)
- Aprender na prática como desenvolver uma habilidade Alexa e como enviar comandos para um dispositivo conectado

MICROCHIP MASTERs 2019

Assistentes de Voz Digital

SMART | CONNECTED | SECURE

MICROCHIP

Visão Geral

- Atualmente temos 4 soluções de assistentes de voz:
- Amazon “Alexa”
- Google “Google Assistant”
- Apple “Siri”
- Microsoft “Cortana”

- Open Source Mycroft

Tipos de Assistentes de Voz

- **Embarcados**
  - Reconhecem o comando localmente
  - O próprio equipamento toma a ação conforme o comando

- **Assistentes + dispositivos**
  - Reconhecimento de comandos feito na “nuvem”
  - Ação do comando operado por outro dispositivo

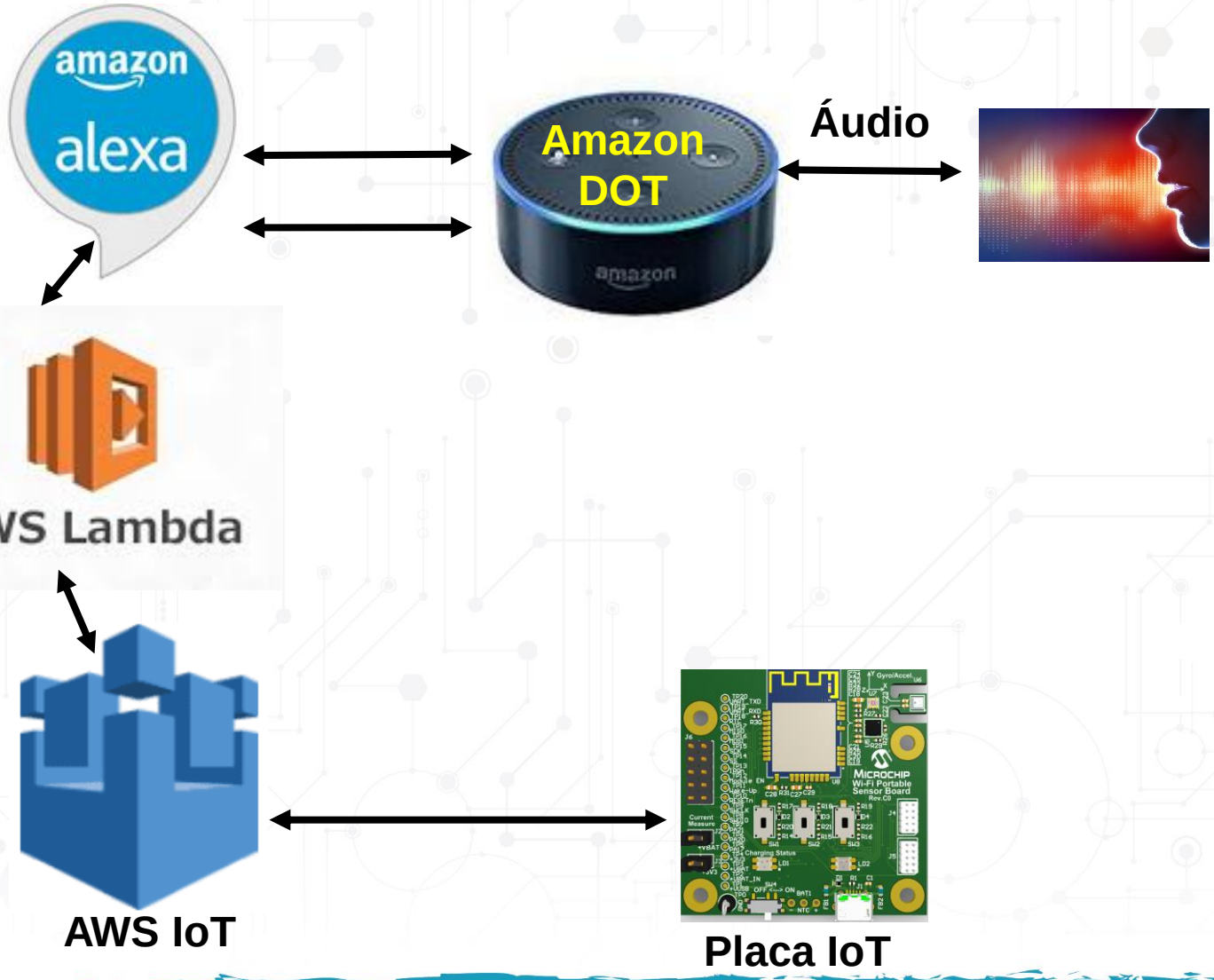
Áudio

MICROCHIP MASTERs 2019

Amazon Alexa

SMART | CONNECTED | SECURE

Alexa Skills
O que acontece



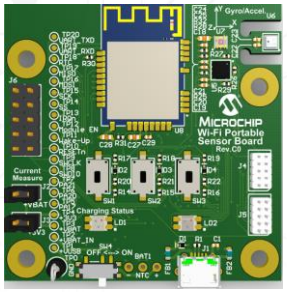
Alexa Skills
O que acontece

Áudio

Acorda o dispositivo "Alexa"

AWS Lambda

AWS IoT



Placa IoT

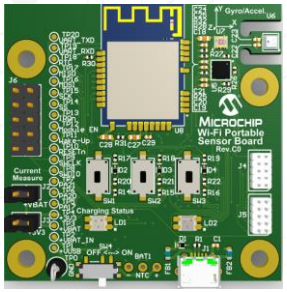
Alexa Skills
O que acontece

Busca os comandos ”Controle de dispositvos” SKILL

Áudio

“Abra Controle de dispositivos”

AWS IoT



Placa IoT

Alexa Skills
O que acontece

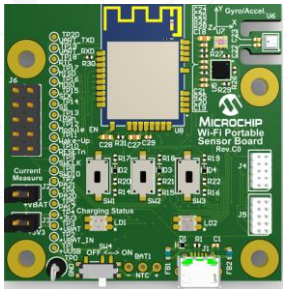
"Skill" envia commando para Lambda

Áudio

Lambda processa o comando, e toma a decisão da ação

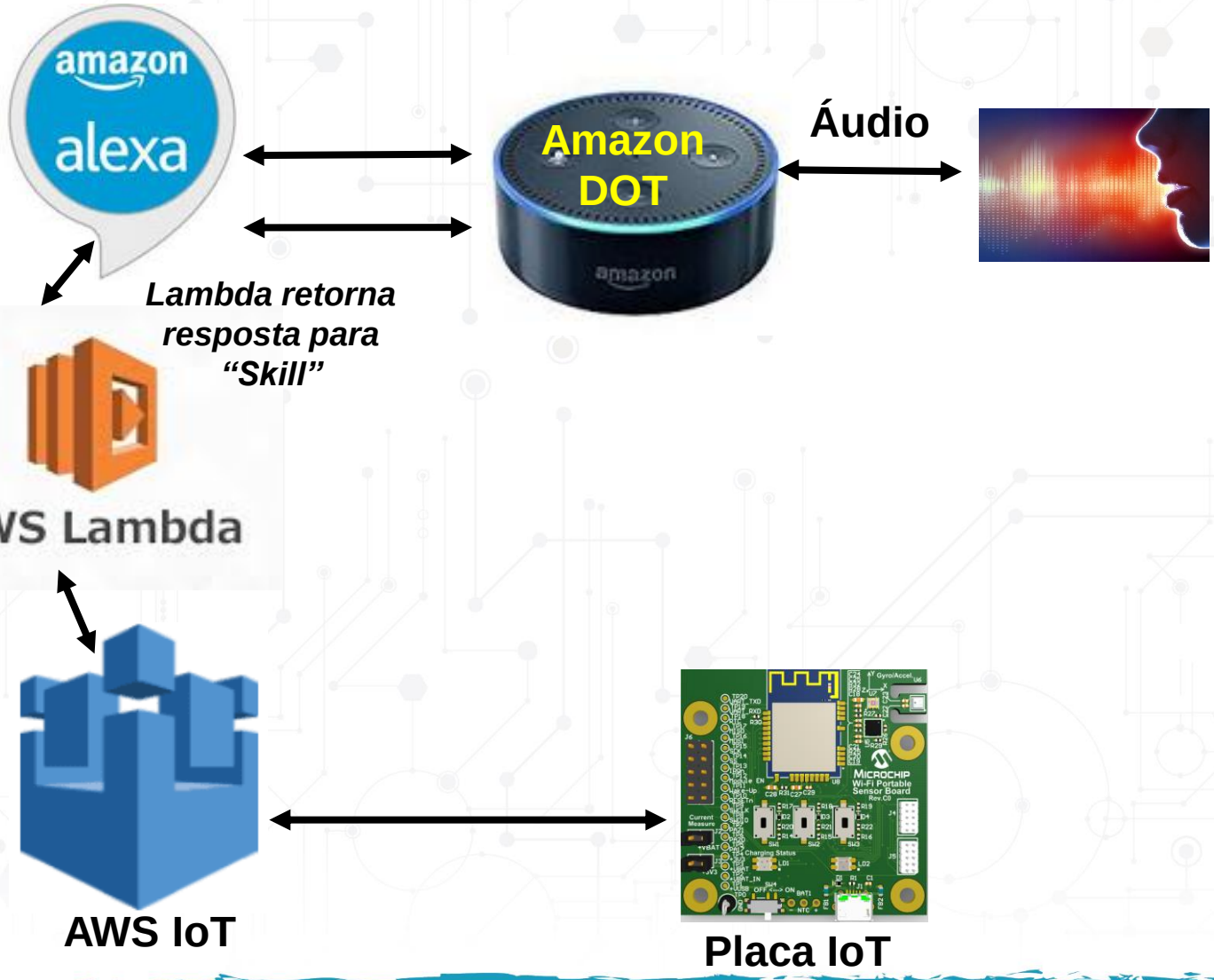
AWS Lambda

AWS IoT



Placa IoT

Alexa Skills
O que acontece



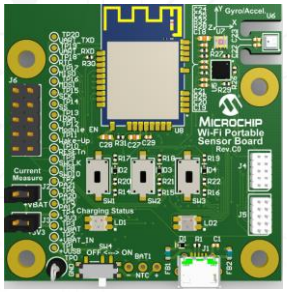
Alexa Skills
O que acontece

Áudio

Resposta: Bem vindos ao Masters Brasil

AWS Lambda

AWS IoT



Placa IoT

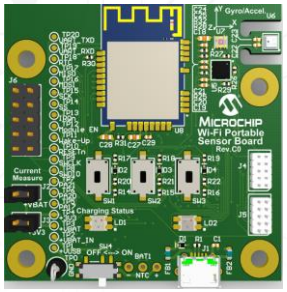
Alexa Skills
O que acontece

Áudio

AWS Lambda

"Liga o LED"

AWS IoT



Placa IoT

Alexa Skills
O que acontece

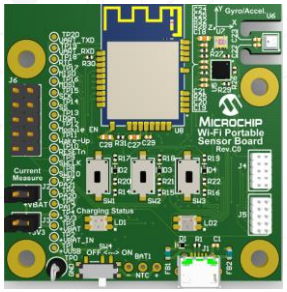
"Skill" envia commando para Lambda

Áudio

Lambda processa o comando, e toma a decisão da ação

AWS Lambda

AWS IoT



Placa IoT

Alexa Skills
O que acontece

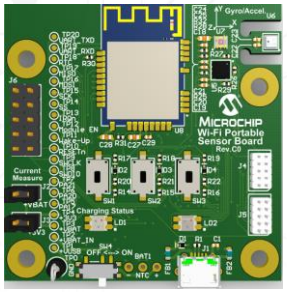
Áudio

Lambda retorna resposta para "Skill"

AWS Lambda

Lambda envia comando para AWS IoT

AWS IoT



Placa IoT

Alexa Skills
O que acontece

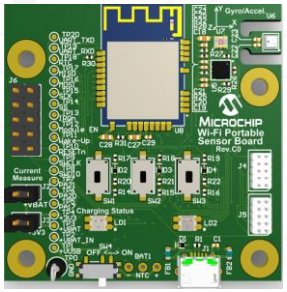
Áudio

Resposta: LED está Ligado

AWS Lambda

AWS IoT

MQTT Pub/Sub API



Placa IoT

Liga o LED

Alexa Skill Kit (ASK)

- **The Alexa Skills Kit (ASK)** é a coleção de APIs, ferramentas, documentação e códigos fontes para fácil e rápido desenvolvimento de comandos para o Alexa
- **5 tipos**
  - Custom
  - Smart Home (Pre-Built-model)
  - Flash Briefing (Pre-Built-model)
  - Video (Pre-Built-model)
  - Music (Pre-Built-model)

Exemplo (PT-BR)

“Alexa, abra placa de sensores”

Invocation Name

“Acenda a Luz verde”

Utterance para “**Acender a Luz**” Intent

Slot

“Luz Verde”

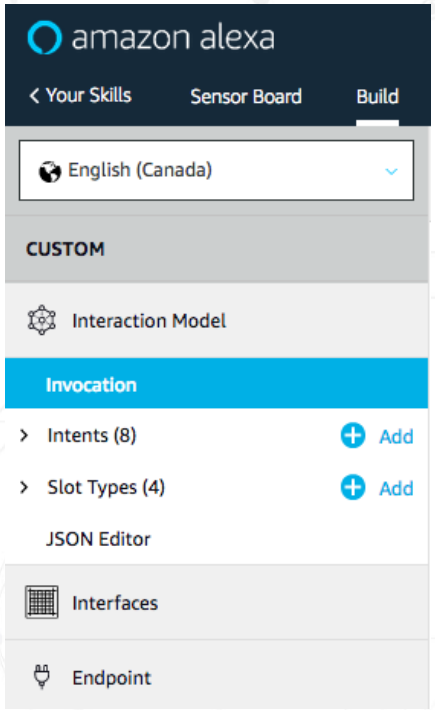
Utterance to “**Acender a Luz**” Intent

Slot

Como criar a sua habilidade

- **A plataforma de desenvolvimento de Alexa Skills tem todos os itens mencionados anteriormente [developer console](https://developer.amazon.com/pt-br/).**

- **https://developer.amazon.com/pt-br/**



Suporte ao seu Skill

- **Uma vez definida a interação com a Alexa através do Custom Skill**
- **Precisamos definir as respostas às perguntas.**
  - **Cloud Service**

Suporte ao seu Skill

- **O envio de requerimento e respostas para um skill Alexa segue um padrão. Esta interação é feita com um outro servidor baseado na nuvem. Este servidor precisa obrigatoriamente ser acessível a Internet. Na página de Endpoint você define o endereço que serão enviados as requisições.**
- **Endpoint:**
  - **AWS Lambda** (Serviço AWS): Serviço de execução de funções sob demanda da AWS que interpreta as requisições e envia as respostas conforme o padrão
  - **Web Service** (Seu próprio servidor): que interpreta as requisições e envia as respostas conforme o padrão

Suporte ao seu Skill

- Alexa se comunica com o seu suporte utilizando mecanismo de requisição e resposta utilizando protocolo **HTTPS** e método **POST**
- O formato da mensagem enviada é **JSON**
  - A mensagem de requisição contém todos os parâmetros necessários para que o servidor tome a decisão correta e preparar uma resposta em formato JSON

Alexa Skills

Gets "Skill" payload

Áudio

Wakes on Keyword

Alexa,
Ask Horoscope what is the horoscope for Gemini

JSON Encoded **Events**

```
{
    "version" : "1.0","Session":{},"Context":{},
    "request" : {
      "type" : "IntentRequest"
      "intent" : { "name" : "GetHoroscope" }
}
```

JSON Encoded **Response**

```
{
    "version" : "1.0",
    "response" : {
      "outputSpeech" : "Blah blah blah"
}
```

Skill Service
AWS Lambda

Skill Service (Lambda)

- **Função AWS Lambda**
  - O AWS Lambda permite que você execute códigos sem provisionar ou gerenciar servidores
  - Você paga apenas pelo tempo de computação que utilizar
  - Suporta códigos em Node.js, Java, Python, C#, Go e outras linguagens
- **Usaremos uma função AWS Lambda para tratar as resposta das requisições Alexa.**

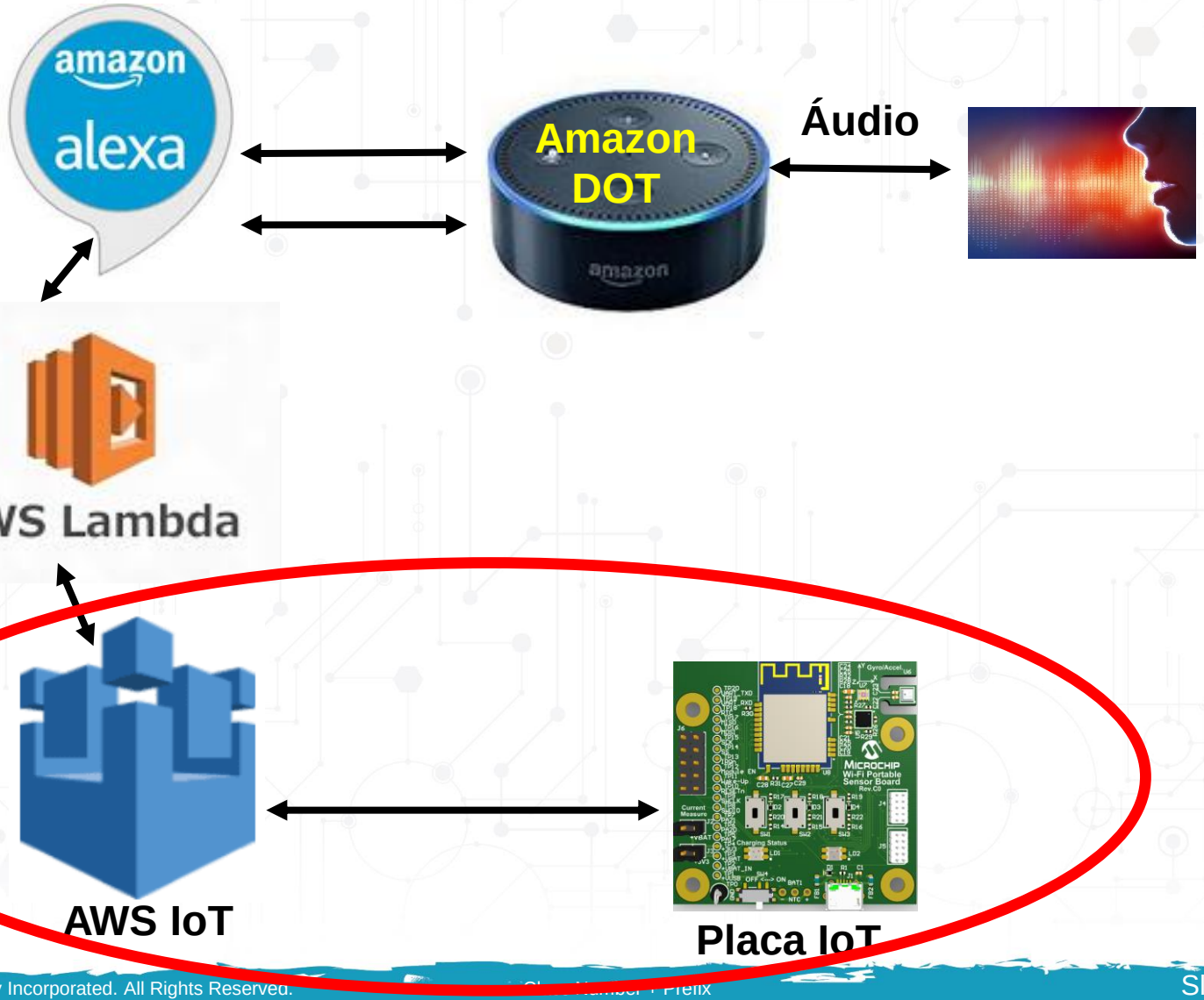
MICROCHIP MASTERs 2019

Amazon AWS IoT

SMART | CONNECTED | SECURE

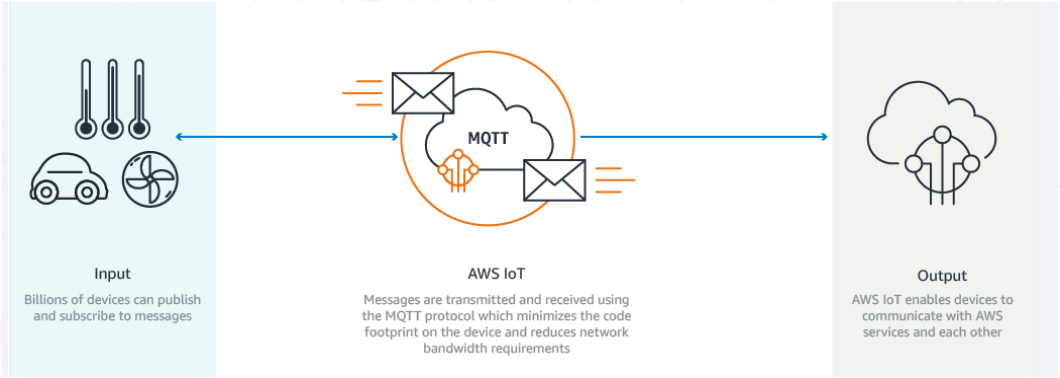
MICROCHIP

Conexão ao Dispositivo



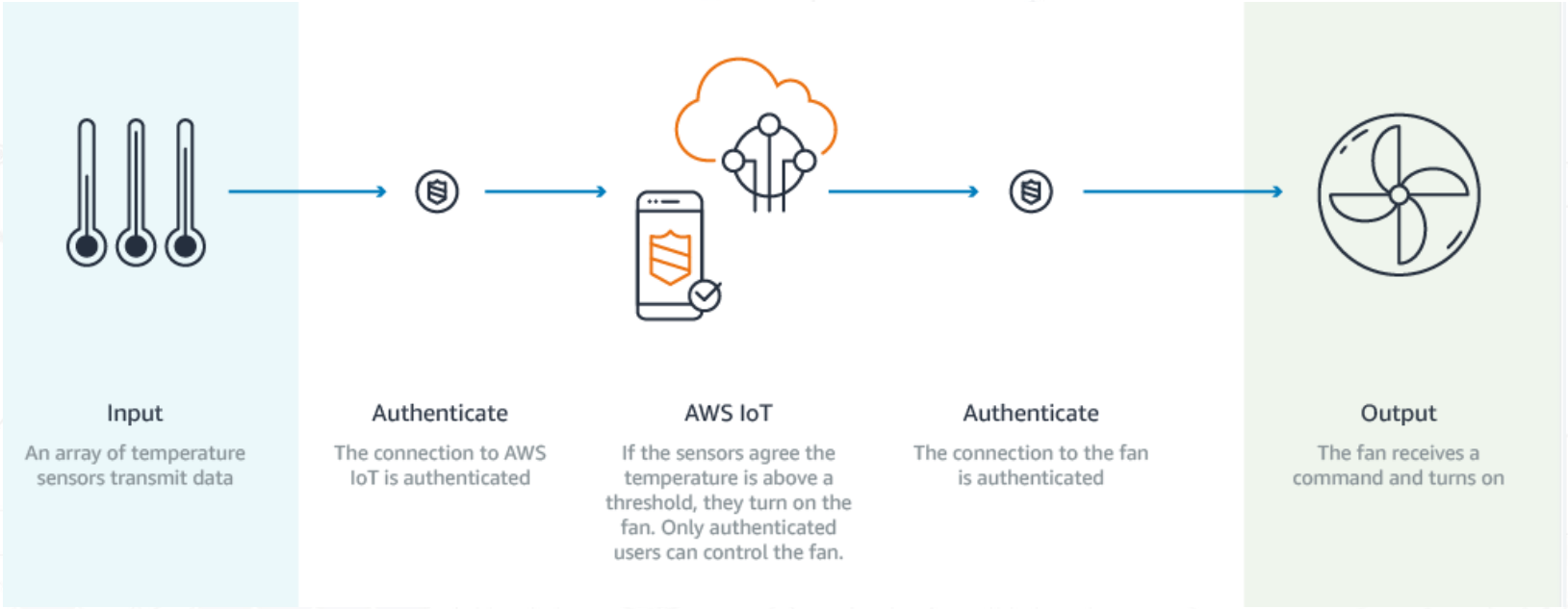
AWS IoT

- **AWS IoT Core proporciona comunicação bidirecional segura entre dispositivos conectados a Internet, como sensores e acionadores, e a nuvem da AWS.**
- **Conecte e gerencie os seus dispositivos**
  - AWS IoT Core permite que você conecte o seu dispositivo de forma simples a nuvem e a outros dispositivos
  - AWS IoT Core suporta protocolos HTTP, WebSockets, e **MQTT**

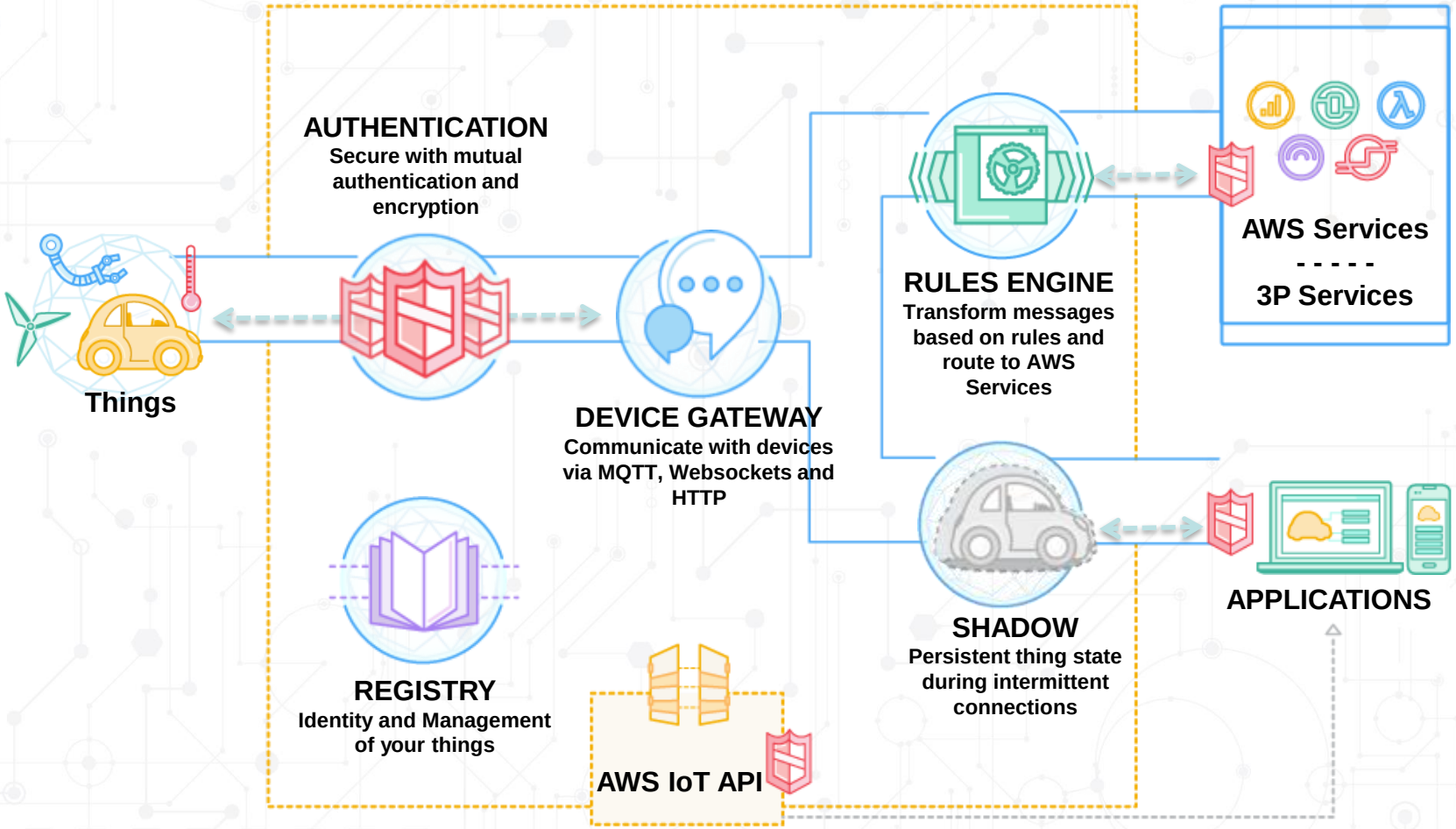


AWS IoT

- **Segurança na conexão e no envio de dados**
  - AWS IoT Core implementa autenticação mútua e encriptação ponto a ponto com todos os pontos de conexão
  - Os dados nunca são transferidos entre dispositivos e a nuvem se não houver o reconhecimento da autoria do dado.



AWS IoT Core Componentes



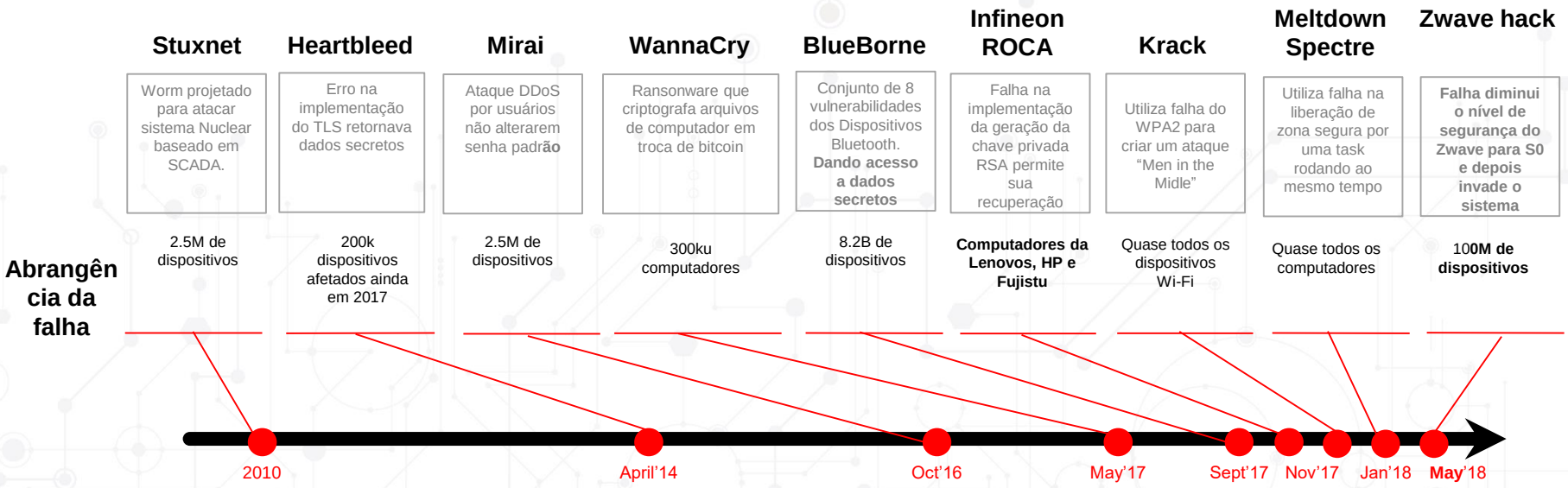
Segurança em IoT

- Existem os 3 pilares básicos que é preciso saber em segurança em aplicações de IoT (CIA)
  - Integridade - O dado não foi alterado
  - Confidencialidade - O dado é secreto
  - Autenticidade - O dado foi enviado pelo dispositivo correto

Segurança em IoT

Cyber ataques

Evolução dos ataques de software



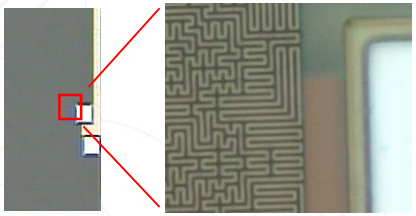
Segurança em IoT

- **POR QUÊ?**
- **Proteção de Marca**
  - Qualidade do Produto
  - Qualidade na usabilidade
- **Proteção ao faturamento**
  - Previne os clones e produção desviada
  - Previne usuários não autorizados
- **Proteção de propriedade Intelectual**

Introdução ao ATECC608A

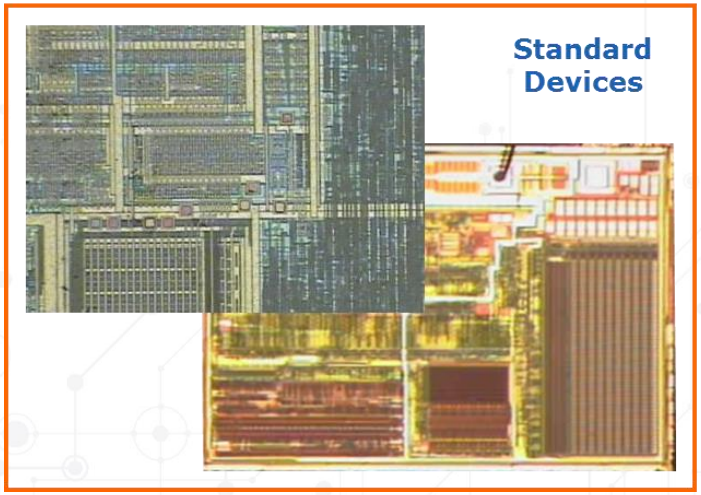
- Ambiente Seguro para geração, armazenamento e utilização de chaves criptográficas
  - Simétricas (SHA256, AES128)
  - Assimétrica (ECC-P256)
    - SECP256r1 e PRIME256v1
- 10Kbits de armazenamento divididos em 16 slots
- True Random Number Generator
  - NIST 800-90A/B/C
- Fácil Integração com Microcontrolador (I2C ou 1-wire)
- Proteção de hardware em Multi-níveis
  - Monitores de Temperatura, Tensão e Clock
  - Proteção de DIE

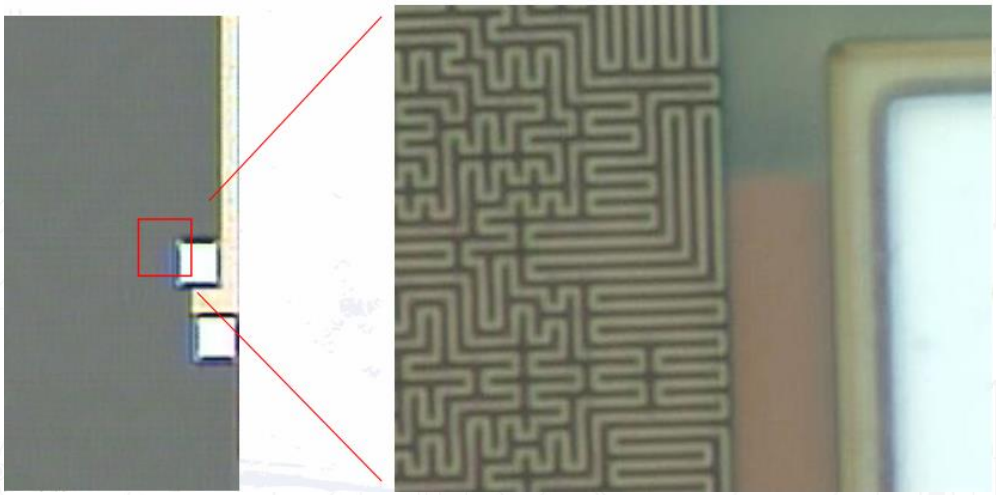
Microchip Active Shield



Armazenamento seguro

- Active shield
- Memória interna encriptada
- Operações matemáticas randomizadas
- Monitores de temperatura e tensão
- Contador incremental
- Sem comunicação JTAG

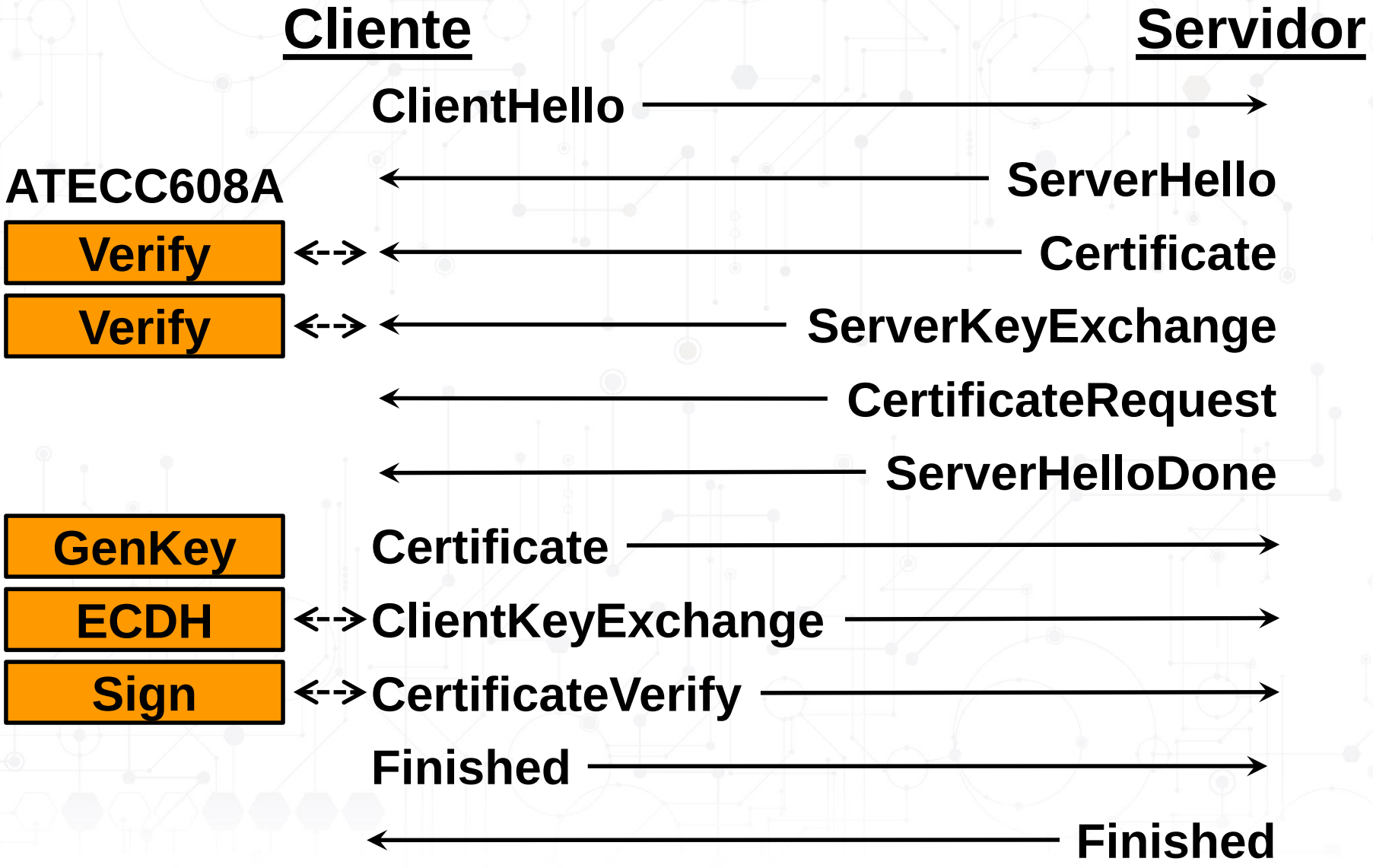




TLS – Transport Layer Security

- **Objetivos:**
  - Garantir privacidade e integridade dos dados em uma comunicação ponto a ponto.
  - Estabelecer conexão privada com criptografia simétrica baseada em uma chave negociada para cada conexão
  - Identificação de uma ou das duas partes através de criptografia assimétrica e certificados digitais utilizados na autenticação.
  - Interoperabilidade independente da implementação.

TLS com ATECC608A

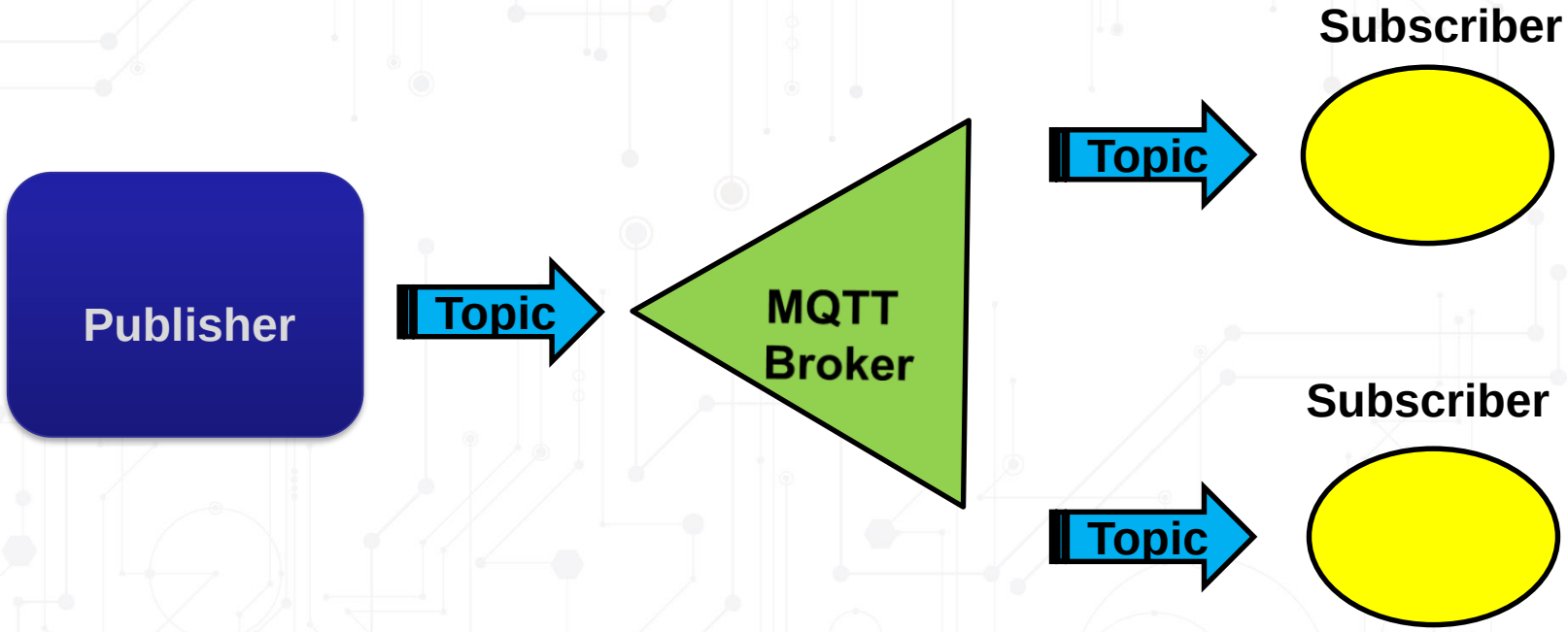


Gerenciamento dos certificados AWS IoT

- **Gerenciado pela AWS**
  - CA (Autoridade Certificadora) gerenciado pela AWS
  - AWS gera o par de chaves e o certificado para o dispositivo.
  - Ótimo para ambiente de teste
- **Gerenciado pelo Cliente**
  - CA gerenciado pelo Cliente e armazenado no AWS IoT Core
  - Chaves privadas criadas pelo cliente
  - Ótimo para produção em larga escala

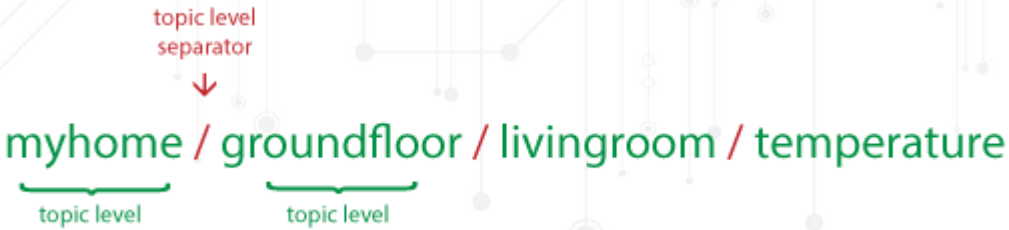
MQTT

- **Publisher/Subscriber**



Tópicos

- **Os tópicos no MQTT possuem a mesma estrutura de pastas**



- **Para se inscrever em um canal, é preciso especificar o endereço completo**
- **Ou usar marcadores especiais (+ e #)**

Device Shadow

- **O Device Shadow é um documento JSON criado automaticamente que contém os dados atuais do dispositivo.**
  - Representação virtual do último estado reportado
  - Shadow individual para cada dispositivo
  - Permite ler e enviar dados de/para o dispositivo utilizando MQTT mesmo quando o dispositivos **não** está conectado

JSON Shadow

- **Shadow showing a DELTA:**

```
{
    "desired":
        {   "LED_R": 1,    "LED_G": 0,    "LED_B": 1  },
    "reported":
        {   "macAddr": "f8f005e45f8c",
            "temp": 3081,
            "COUNT": 955,
            "BUTTON_1": 1,   "BUTTON_2": 1, "BUTTON_3": 1,
            "LED_R": 1, "LED_G": 1, "LED_B": 1,
            "uv": 8078000,
            "hum": 22,
            "pressure": 1011
        },
    "delta":
    {   "LED_G": 0  }
}
```

Agenda AWS IoT

- AWS IoT
- Importância da segurança em IoT
- Implementação do TLS com ATECC608A
- Requisitos do AWS IoT Authentication e JITR (Just in Time Registration)
- MQTT e AWS IoT Device SHADOW
- Kit de desenvolvimento
- Lab2
  - Sensor Board interagindo com Device SHADOW

Descrição do Hardware

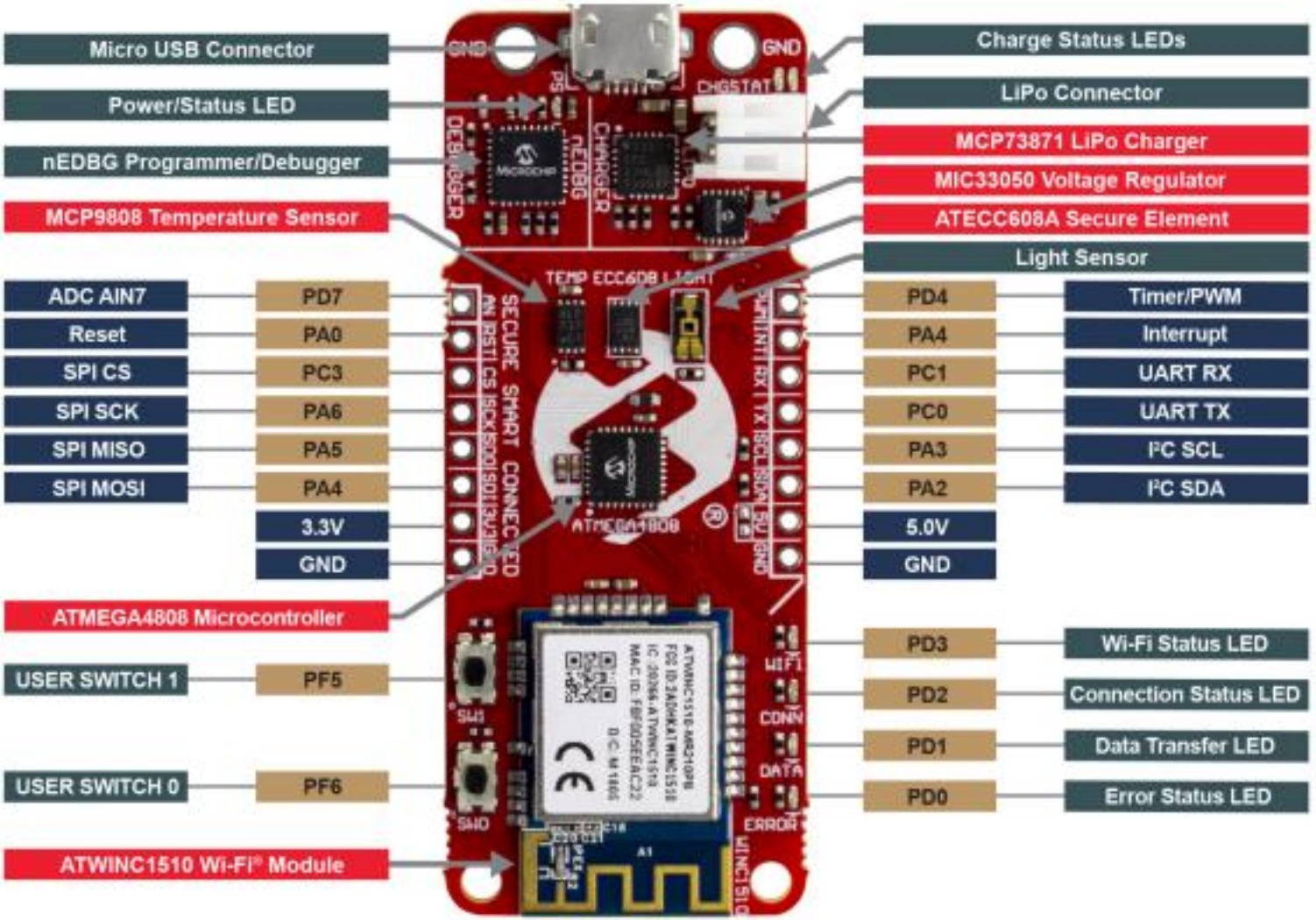
- Microcontrolador: ATmega4808 (AVR 8 bits, 6KB RAM, 48KB Flash)
- Modulo Wi-Fi®: ATWINC1510 com certificado ANATEL
- Power: Carregador de Bateriade Litio MCP73871
- Crypto IC: ECC608 (com certificado AWS)
- Interface: 2 Botões, 4 LEDs, Conector padrão Click

Mobile APP

Sensores do Hardware

- **Programador**
  - nEDBG embarcado
- **Sensores**
  - Sensor de Temperatura (MCP9808)
    - Saída Digital I2C
  - Sensor de Luminosidade(TEMT6000)
    - Saída Analógica

AC164160



Informações Adicionais

- **Kit de desenvolvimento AC164160:**
  - Informação
    https://www.microchip.com/DevelopmentTools/ProductDetails/AC164160
  - Compra
    https://www.microchipdirect.com/product/AC164160
  - Código fonte
    https://github.com/MicrochipTech/Repurpose_AVR-IoT_WG_to_Connect_to_AWS

Lab 2: Objetivos

- Programar AVR IoT com informações da rede e endpoint AWS
- Conectar a placa ao AWS IoT
- Enviar dados para o AWS IoT e verificá-los na estrutura de Shadow

MICROCHIP MASTERs 2019

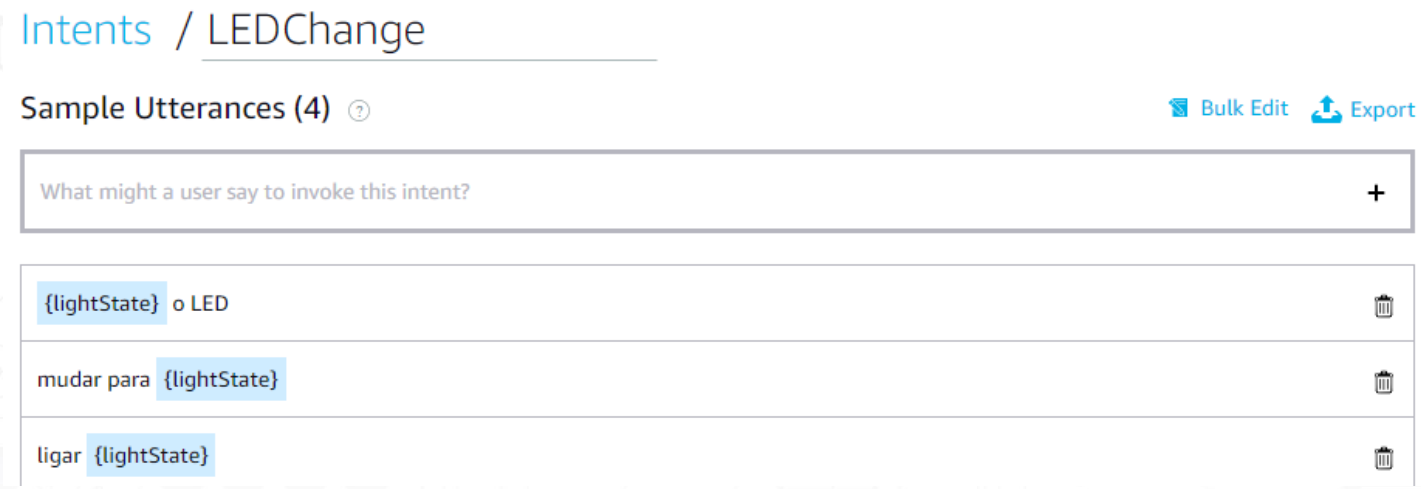
Integrando o Alexa ao AWS IoT para envio de dados para o dispositivo

SMART | CONNECTED | SECURE

MICROCHIP

Alexa *IntentRequest*

- **Uma requisição do tipo *IntentRequest* é enviado toda a vez que um comando reconhecido pela Alexa é enviado “falado” (ex: Liga o Led”)**



Alexa Skills

"Controle de dispositivos"
Custom Skill

Áudio

Liga o Led

JSON Encoded **Events**

```
{
    "version" : "1.0" , "session":"xxx","context":"yyy,
    "request" : {
      "type" : "IntentRequest"
      "intent" : { "name" : "LEDStateChangeIntent" }
      "slots" : { "Status" : {"Name":"Status","value":"on"}
    }
```

JSON Encoded **Response**

```
{
    "version" : "1.0"
    "response" : {
      "outputSpeech" : "O Led está Ligado"
}
```

Lambda Service

AWS IoT

AWS Lambda

- **Serviço de computação em nuvem que roda o código sem a necessidade de ter um servidor**
- **Executa o dado apenas quando é necessário, a partir de um gatilho / evento**
- **Facilmente escalável. AWS toma conta da escalabilidade de algumas requisições por dia para milhares por segundo**

- **Mais informações:**
  - https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

Função Handler (Node.js)

- O Alexa Service envia um comando para o Lambda via um HTTP POST contendo o JSON descrito anteriormente
- AWS Lambda chama a função via um objeto Handler

```
exports.myHandler = function(event, context) {
    ...
}
```

- myHandler é o nome do método que o AWS Lambda invoca ao iniciar a função.

Lambda Handler

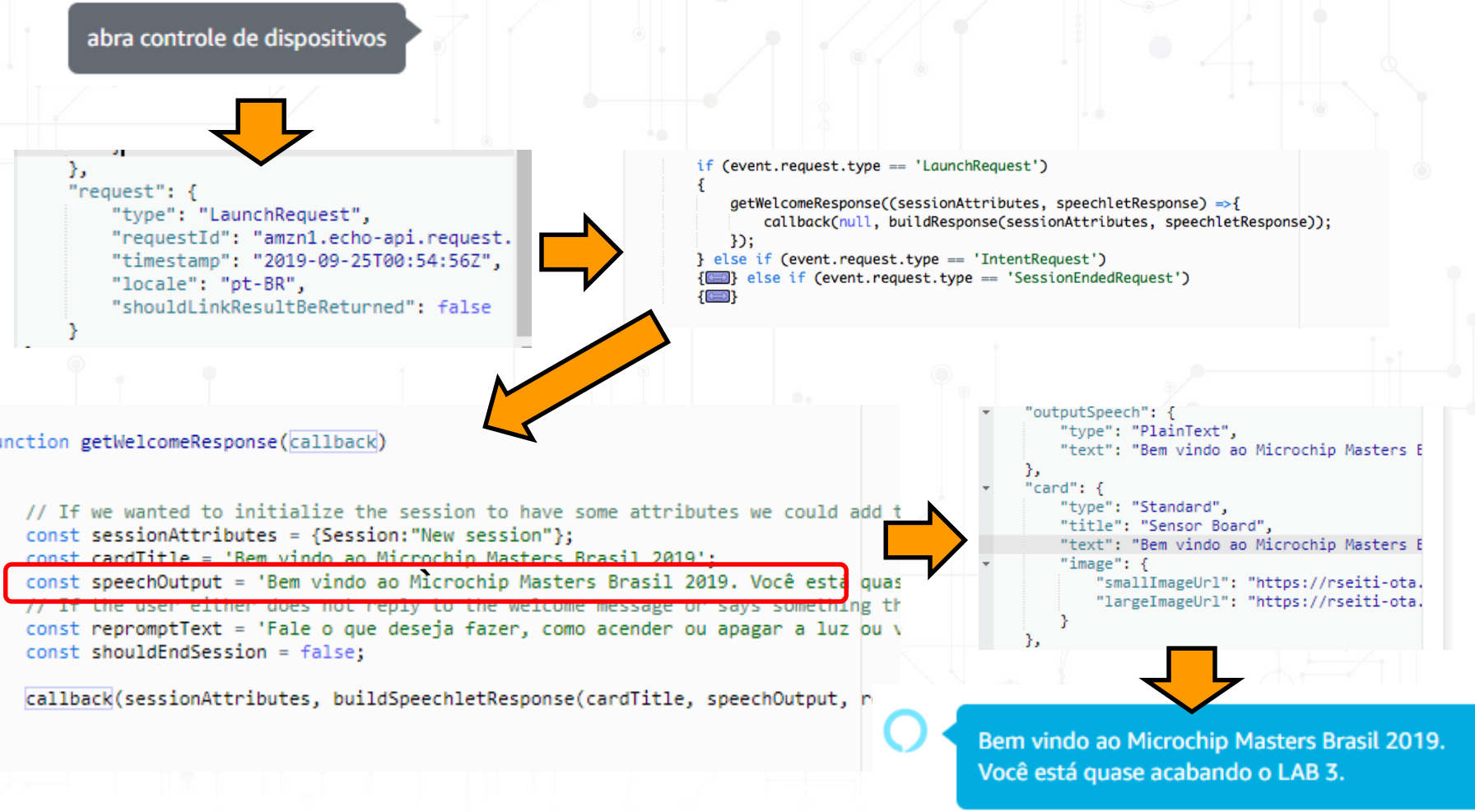
index.js

```
// Route the incoming request based on type (LaunchRequest, IntentR
exports.handler = (event, context, callback) =>
{

    try{

        /*...*/

        if (event.request.type == 'LaunchRequest')
        {...} else if (event.request.type == 'IntentRequest')
        {...} else if (event.request.type == 'SessionEndedRequest')
        {...}


    }

    catch(err)
    {
        callback(err);
    }

};

/* ---------------------- end: Main handler ----------------------
```

- **Entradas e Saídas:**
  - Entrada:
    - Event
  - Saídas:
    - Context
    - Callback
- **Eventos gerados pelo Alexa**
  - LaunchRequest
  - IntentRequest
  - SessionEndRequest

LaunchRequest

- **Chamado ao iniciar a habilidade**

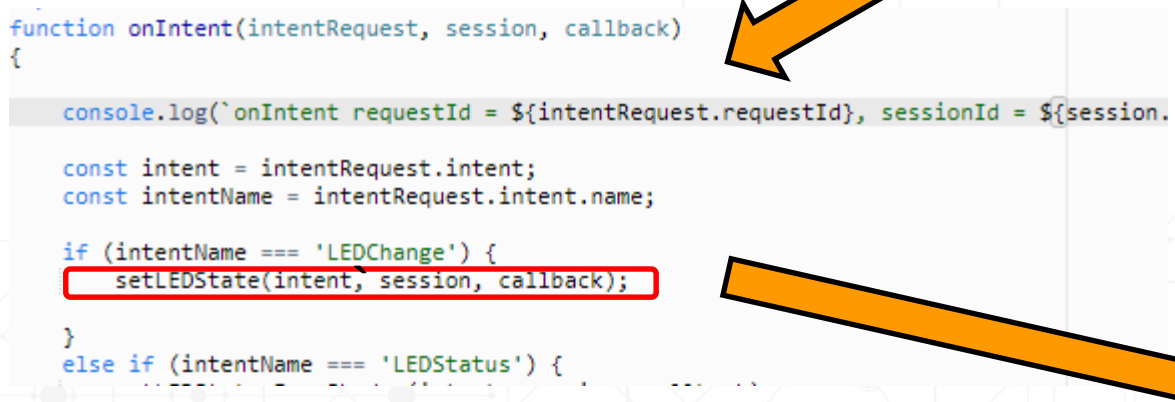


IntentRequest (Ligar o Led)

ligar o led

```
request": {
    "type": "IntentRequest",
    "requestId": "amzn1.echo-api.request.aa9ea7
    "timestamp": "2019-09-25T01:00:36Z",
    "locale": "pt-BR",
    "intent": {
        "name": "LEDChange",
        "confirmationStatus": "NONE",
        "slots": {
            "lightState": {
                "name": "lightState",
                "value": "ligar",
                "resolutions": {
```

```
if (event.request.type == 'LaunchRequest')
{} else if (event.request.type == 'IntentRequest')
{
    onIntent(event.request,
        event.session,
        (sessionAttributes, speechletResponse) =>{
            console.log("Returning from onIntent");
            console.log("buildResponse returns =>",buildResponse(sessionAttributes, speec

            callback(null, buildResponse(sessionAttributes, speechletResponse));

            console.log("Returning from callback");

});

} else if (event.request.type == 'SessionEndedRequest')
{}
```



```
function onIntent(intentRequest, session, callback)
{

    console.log(`onIntent requestId = ${intentRequest.requestId}, sessionId = ${session.

    const intent = intentRequest.intent;
    const intentName = intentRequest.intent.name;

    if (intentName === 'LEDChange') {
        setLEDState(intent, session, callback);

    }
    else if (intentName === 'LEDStatus') {
```

Próximo Slide

IntentRequest (Ligar o Led)

```
function setLEDState(intent, session, callback)
{

    const cardTitle = intent.name;
    const desiredLEDStateSlot = intent.slots.lightState;
    let shadowLED_R = 0;
    let shadowLED_G = 0;
    let shadowLED_B = 0;
    let repromptText = 'Diga outro comando';
    let sessionAttributes = {};

    const shouldEndSession = false;
    let speechOutput = '';

    if (desiredLEDStateSlot)
    {
        const desiredLEDState = desiredLEDStateSlot.value;
        sessionAttributes = createFavoriteLEDStatusAttributes(desiredLEDState);
        if ((desiredLEDState == 'branco')||(desiredLEDState == 'ligado')||(desiredLEDState == 'ligar')||(
        {
            shadowLED_R = 1;
            shadowLED_G = 1;
            shadowLED_B = 1;
            speechOutput = "O LED está " + desiredLEDState;
        }else
```

```
console.log("Alexa will say =>",speechOutput);
var payloadObj = {"state" :
    { "desired" :
        { "LED_R" : shadowLED_R,
          "LED_G" : shadowLED_G,
          "LED_B" : shadowLED_B

        }}};

//Prepare the parameters of the update call
var paramsUpdate = {

    "thingName" : config.IOT_THING_NAME,
    "payload" : JSON.stringify(payloadObj)

};
```

$aws/things/bfc7ea92a3f9add05b164f0a8b...

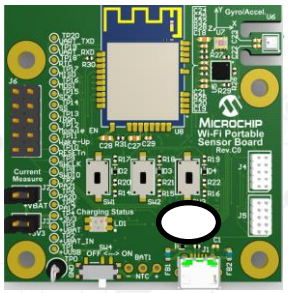
```
{
  "state": {
    "desired": {
      "LED_R": 1,
      "LED_G": 1,
      "LED_B": 1
    }
  }
}
```

```
"body": {
    "version": "1.0",
    "response": {
        "outputSpeech": {
            "type": "PlainText",
            "text": "O LED está ligar"
        },
        "card": {
            "type": "Standard",
            "title": "Sensor Board",
            "text": "O LED está ligar",
            "image": {
```

Fluxo dos comandos

AWS IoT

MQTT Broker



Fluxo dos comandos

AWS IoT

MQTT Broker

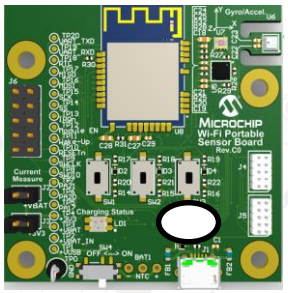
{“state”: {“reported”:{“LED_R”:1,“LED_G”:1,“LED_B”:1}}}

$aws/things/<THINGID>/shadow/update

Tópico pub cada 5seg

Tópico sub

$aws/things/<THINGID>/shadow/update/delta



SHADOW

```
{“state”:
 {“reported”:
   {“LED_R”:1,
    “LED_G”:1,
    “LED_B”:1
   }
  }
}
```

Fluxo dos comandos

Led Vermelho

IntentRequest
LEDChange
Vermelho

AWS Lambda

$aws/things/<THINGID>/shadow/update
{“state”:{“desired”:{“LED_R”:1,
“LED_G”:0,
“LED_B”:0}}}

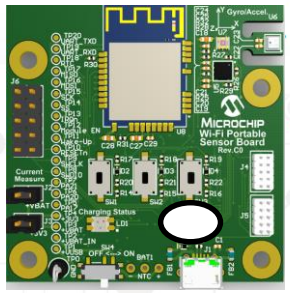
{“state”: {“reported”:{“LED_R”:1,“LED_G”:1,“LED_B”:1}}}

AWS IoT

MQTT
Broker

Tópico pub cada 5seg

Tópico sub



SHADOW

Anterior

```
{“state”:
 {“reported”:
   {“LED_R”:1,
    “LED_G”:1,
    “LED_B”:1
   }
  }
}
```

Atual

```
{“state”:
 {“reported”:
   {“LED_R”:1,
    “LED_G”:1,
    “LED_B”:1
   }
  }
 {“desired”:
   {“LED_R”:1,
    “LED_G”:0,
    “LED_B”:0
   }
  }
 {“delta”:
   {“LED_G”:0,
    “LED_B”:0
   }
  }
}
```

Fluxo dos comandos

Led Vermelho

AWS Lambda

AWS IoT

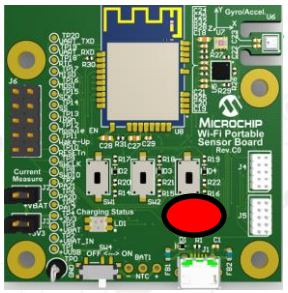
MQTT Broker

{“state”: {“reported”:{“LED_R”:1,“LED_G”:0,“LED_B”:0}}}

Tópico pub cada 5seg

Tópico sub

{“state”:{“delta”:{“LED_G:0,“LED_B”:0,}}}



SHADOW

Anterior

```
{“state”:
 {“reported”:
   {“LED_R”:1,
    “LED_G”:1,
    “LED_B”:1
   }
  }
 {“desired”:
   {“LED_R”:1,
    “LED_G”:0,
    “LED_B”:0
   }
  }
 {“delta”:
   {“LED_G”:0,
    “LED_B”:0
   }
  }
}
```

Atual

```
{“state”:
 {“reported”:
   {“LED_R”:1,
    “LED_G”:0,
    “LED_B”:0
   }
  }
}
```

IntentRequest (LED Status)

qual a cor?

```
"request": {
    "type": "IntentRequest",
    "requestId": "amzn1.echo-api.request.0
    "timestamp": "2019-09-25T13:48:25Z",
    "locale": "pt-BR",
    "intent": {
        "name": "LEDStatus",
        "confirmationStatus": "NONE",
        "slots": {
            "lightType": {
                "name": "lightType",
                "value": "cor",
                "resolutions": {
```

```
if (event.request.type === 'LaunchRequest')
{...} else if (event.request.type === 'IntentRequest')
{
    onIntent(event.request,
        event.session,
        (sessionAttributes, speechletResponse) =>{
            console.log("Returning from onIntent");
            console.log("buildResponse returns =>",buildResponse(sessionAttributes, speec

            callback(null, buildResponse(sessionAttributes, speechletResponse));

            console.log("Returning from callback");

});

} else if (event.request.type === 'SessionEndedRequest')
{...}
```

```
function onIntent(intentRequest, session, callback
{
    console.log("");
    const intent = intentRequest.intent;
    const intentName = intentRequest.intent.name;
    const intentSlot = intentRequest.intent.slots;
    console.log("inIntent =>",intentName);
    // Dispatch to your skill's intent handlers
    if (intentName === 'Information') {
        getDetailResponse(callback); }
    if (intentName === 'LEDStatus') {
        getLEDStatus(intent, session, callback); }
    if (intentName === 'LEDChange') {
```

```
function getSensorStatus(intent, session, callback) {

    let cardTitle = "Get SENSOR";
    let desiredLEDStatus;
    const repromptText = null;
    const sessionAttributes = {};
    let shouldEndSession = false;
    let speechOutput = '';

    //Prepare the parameters of the update call
    var paramsUpdate = {
        "thingName" : config.thingId,
        //"payload" : JSON.stringify(payloadObj)
    };
    iotData.getThingShadow(paramsUpdate, function(err, data)
        {
```

IntentRequest (LED Status)

Estado da sombra:

```
{
  "desired": {
    "LED_R": 1,
    "LED_G": 1,
    "LED_B": 0,
    "Light": 1
  },
  "reported": {
    "macAddr": "f8f00594d0bc",
    "uv": 43400000,
    "COUNT": 157,
    "hum": 35,
    "temp": 2582,
    "pressure": 940,
    "LED_R": 0,
    "LED_G": 1,
    "LED_B": 0,
    "Light": 1,
    "BUTTON_1": 0,
    "BUTTON_2": 0,
    "BUTTON_3": 0,
    "LED_INTENSITY": 100
  },
  "delta": {
    "LED_R": 1
  }
}
```

```
//console.log('LED State is => ',data.payload);
var redLED = shadow.state.reported.LED_R;
var greenLED = shadow.state.reported.LED_G;
var blueLED = shadow.state.reported.LED_B;
var ledState;

if (intent.slots.lightType.value == "state")
{
    if(redLED == 0 && greenLED == 0 && blueLED == 0)
        ledState ="desligado";
    else
        ledState ="ligado";

    speechOutput = `O led está  ${ledState}.`;

}else
if (intent.slots.lightType.value == "cor")
{
    if (redLED == 1 && greenLED == 0 && blueLED == 0)
        ledState = "vermenho";
    else
    if (redLED == 0 && greenLED == 1 && blueLED == 0)
        ledState = "verde";
    else
    if (redLED == 0 && greenLED == 0 && blueLED == 1)
        ledState = "azul";
    else
    if (redLED == 1 && greenLED == 1 && blueLED == 0)
        ledState = "amarelo";

    speechOutput = `O led está  ${ledState}.`;

}else
    speechOutput = "Me fale, Qual a cor ou como está o led";
```

Lab 3: Objetivos

- Criar uma nova Skill utilizando o Custom Skill que tenha interação com o kit de desenvolvimento
- Criar a função Lambda para responder aos comandos Alexa e também enviar e receber dados do Device Shadow
- Usar Alexa Test para alterar a cor do LED e verificar umidade e temperatura.

Lab 3 Resumo

- **Neste exercício vimos...**
  - Como criar uma Skill Alexa para comandar dispositivos usando o AWS IoT
  - Criar uma função Lambda para responder aos comandos Alexa e também enviar dados para o AWS IoT
  - Simular um Echo Dot utilizando o Alexa Simulator

MICROCHIP
MASTERs 2019

LEGAL NOTICE

SOFTWARE:

You may use Microchip software exclusively with Microchip products. Further, use of Microchip software is subject to the copyright notices, disclaimers, and any license terms accompanying such software, whether set forth at the install of each program or posted in a header or text file.

Notwithstanding the above, certain components of software offered by Microchip and 3rd parties may be covered by "open source" software licenses – which include licenses that require that the distributor make the software available in source code format. To the extent required by such open source software licenses, the terms of such license will govern.

NOTICE & DISCLAIMER:

These materials and accompanying information (including, for example, any software, and references to 3rd party companies and 3rd party websites) are for informational purposes only and provided "AS IS." Microchip assumes no responsibility for statements made by 3rd party companies, or materials or information that such 3rd parties may provide.

MICROCHIP DISCLAIMS ALL WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING ANY IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY DIRECT OR INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND RELATED TO THESE MATERIALS OR ACCOMPANYING INFORMATION PROVIDED TO YOU BY MICROCHIP OR OTHER THIRD PARTIES, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBLITY OF SUCH DAMAGES OR THE DAMAGES ARE FORESEEABLE. PLEASE BE AWARE THAT IMPLEMENTATION OF INTELLECTUAL PROPERTY PRESENTED HERE MAY REQUIRE A LICENSE FROM THIRD PARTIES.

TRADEMARKS:

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KEELOQ, KEELOQ logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, EtherSynch, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and Quiet-Wire are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

Silicon Storage Technology is a registered trademark of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.