





# **Microcontroladores Aplicados a IoT**

Dilson Liukiti Ito



# 11 – Bluetooth Low Energy (BLE)

**Microcontroladores Aplicados a IoT**

# Aplicativos BLE



- Android:
  - nRF Connect: [https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=pt\\_BR&gl=US](https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp&hl=pt_BR&gl=US)
  - Beacon Scanner: [https://play.google.com/store/apps/details?id=com.bridou\\_n.beacon\\_scanner](https://play.google.com/store/apps/details?id=com.bridou_n.beacon_scanner)
- iOS
  - nRF Connect: <https://apps.apple.com/br/app/nrf-connect-for-mobile/id1054362403>
  - BLE Hero: <https://apps.apple.com/us/app/ble-hero/id1013013325>

# Bluetooth



- Tecnologia sem fio PAN (Personal Area Network)
  - é formada pela comunicação de curta distância entre equipamentos em um ambiente reduzido, como uma mesa de trabalho
- Opera na banda 2,4GHz ISM (Industrial, Scientific and Medical)
- Especificações são definidas pelo grupo Bluetooth SIG (Special Interest Group)

# ISM Band



- São porções do espectro reservados internacionalmente para fins industriais, científicos e médicos;
- Mas é mais utilizado em sistemas de comunicação sem fio de curto alcance
  - Telefones sem fio, dispositivos Bluetooth, dispositivos de comunicação de campo próximo (NFC), abridores de portas de garagem, babás eletrônicas e redes de computadores sem fio (Wi-Fi)

# Bluetooth Clássico vs BLE

Bluetooth Classic (BR/EDR)	BLE
Utilizado para aplicações de streaming como áudio, transferência de arquivos e headsets	Utilizado para dados de sensor, controle de dispositivos e aplicações de baixa largura de banda
Não otimizado para baixa potência, mas possui maior taxa de transmissão (3Mbps máximo, comparado com 2Mbps do BLE)	Destinado a baixa potência e baixo ciclo de trabalho
Opera sobre 79 canais de RF (Radio frequência)	Opera sobre 40 canais RF
Discovery (descoberta) ocorre em 32 canais	A descoberta ocorre em 3 canais, levando a uma descoberta e conexões mais rápidas do que o Bluetooth Classic

- Os dois tipos são incompatíveis entre si. Porém existem dispositivos chamados de Dual Mode Bluetooth

# Versões pré BLE



Bluetooth Version	Name	Data Rate (Mbps)	Notes
1.0	Basic rate (BR)	1	Released 1999
1.2			
2.0	Enhanced Data Rate (EDR)	2 & 3	Released 2004
2.1			Released 2007 Simplified pairing
3.0	High speed (HS)	Up to 24 (when using Wi-Fi)	Released 2009

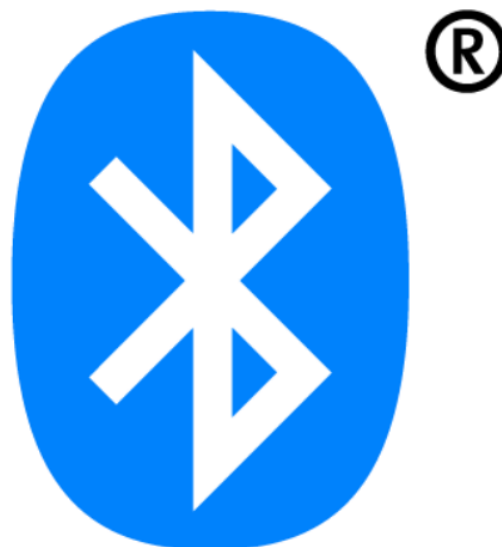


# Versões pós BLE



## The evolution of Bluetooth Low Energy

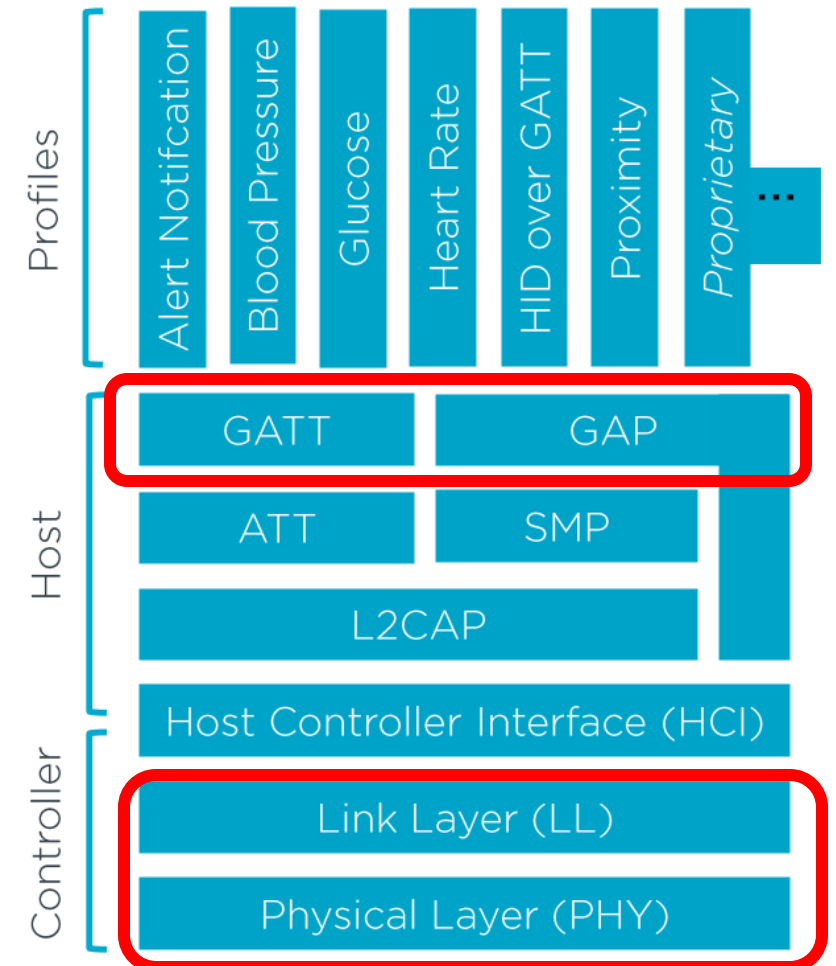
- 2010 - Bluetooth 4.0
- 2013 - Bluetooth 4.1
  - Concurrent Peripheral/Central
- 2014 - Bluetooth 4.2
  - LE Secure Connections
  - Data Length Extension
- 2016 - Bluetooth 5
  - 2 Mbps
  - Long Range
  - Advertising Extensions
  - 10 -> 20 dBm max TX power



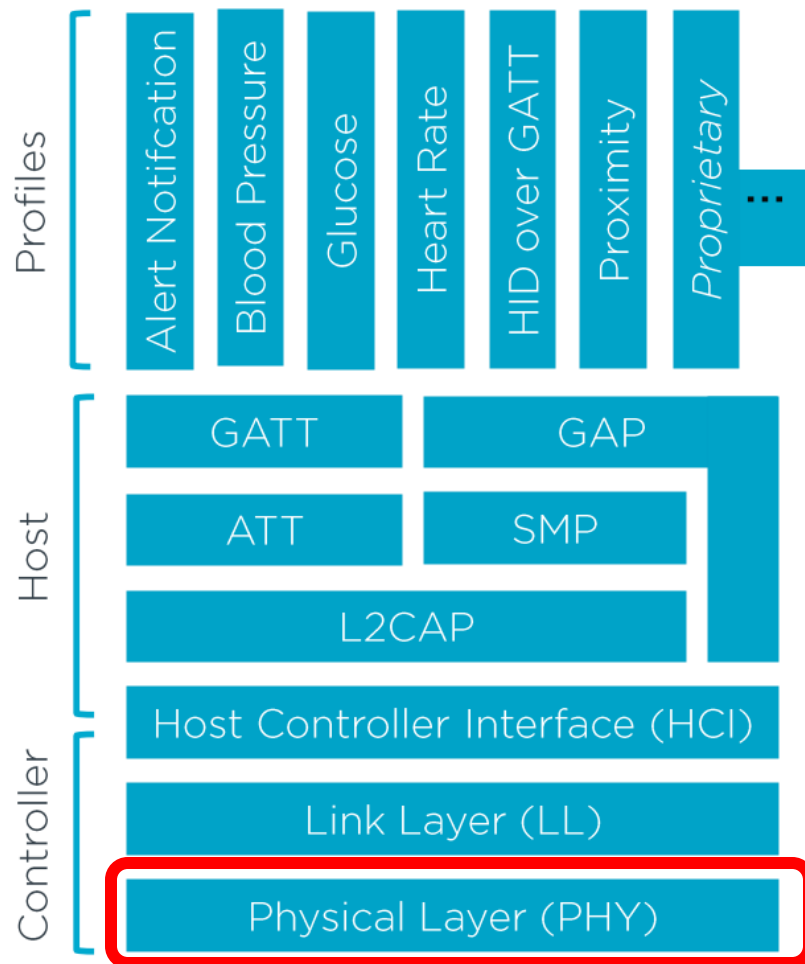
- 2017 - Bluetooth mesh Profile
- 2019 - Bluetooth 5.1
  - Direction Finding
- 2020 - Bluetooth 5.2
  - Isochronous channels
  - LE Power Control
  - Enhanced Attribute Protocol
- Soon - LE Audio

# Arquitetura Bluetooth LE

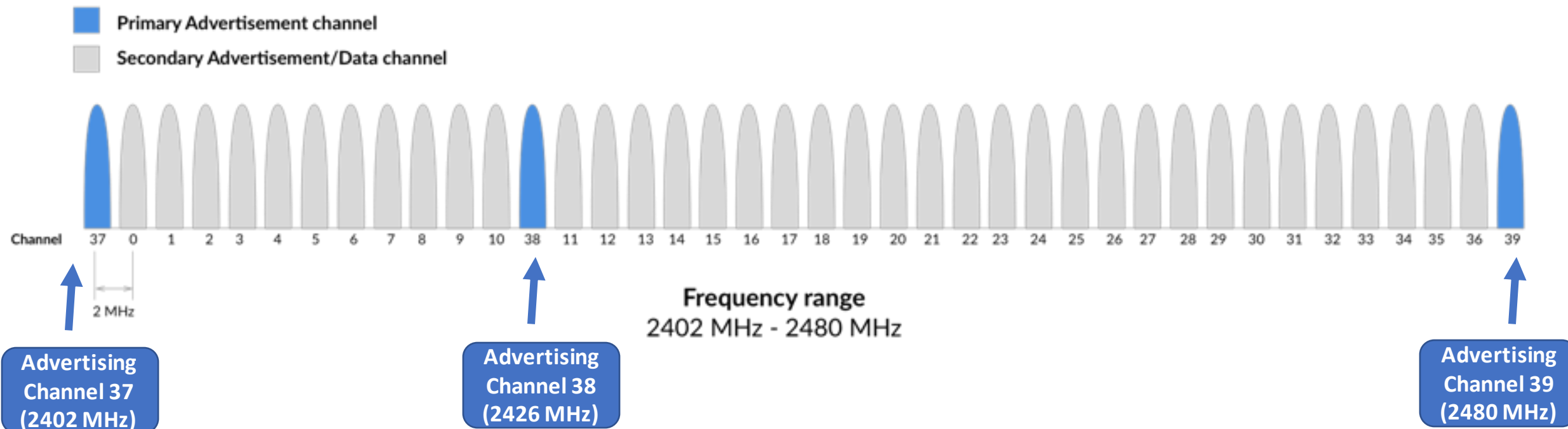
- Três principais blocos:
  - Profile: aplicação do usuário que faz interface com a pilha de protocolos Bluetooth
  - Host: camadas superiores da pilha de protocolos bluetooth
  - Controller: camadas baixas da pilha de protocolos Bluetooth, incluindo o rádio



# Controller: Physical Layer



# PHY Layer: BLE

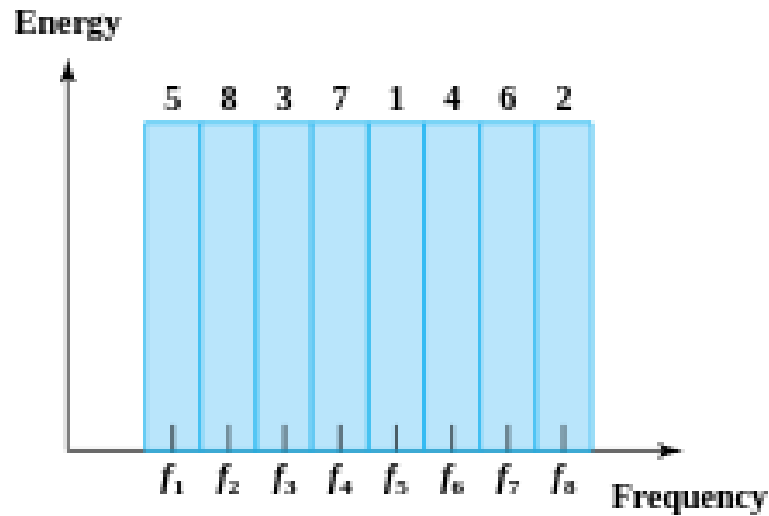


- Dividido em 40 canais de 2,402GHz até 2,480GHz | Max 20 dBm

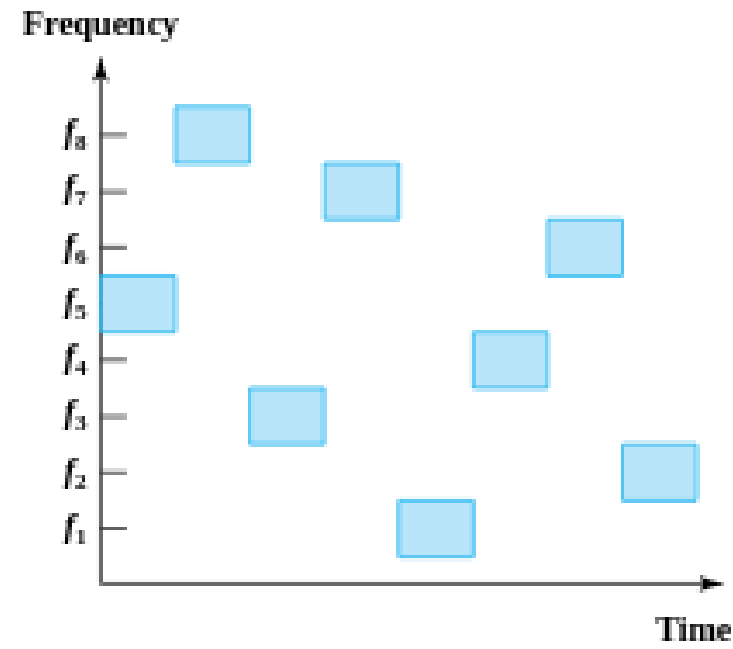
# FHSS - Frequency Hopping Spread Spectrum



- Escalonamento de espectro por salto de frequência
- O sinal transmitido é espalhado em múltiplos canais
- Exemplo:

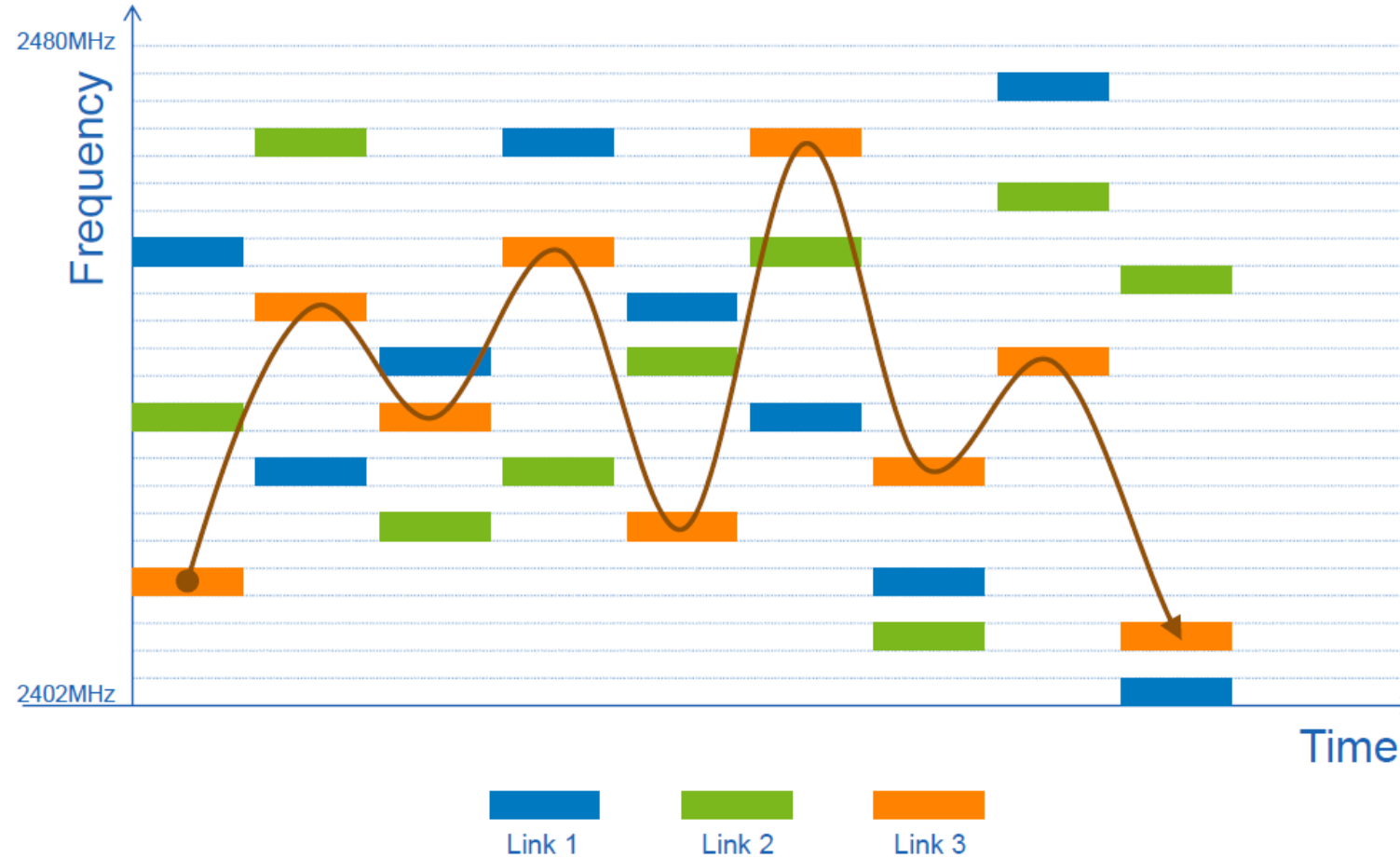


(a) Channel assignment



(b) Channel use

# Múltiplas comunicações

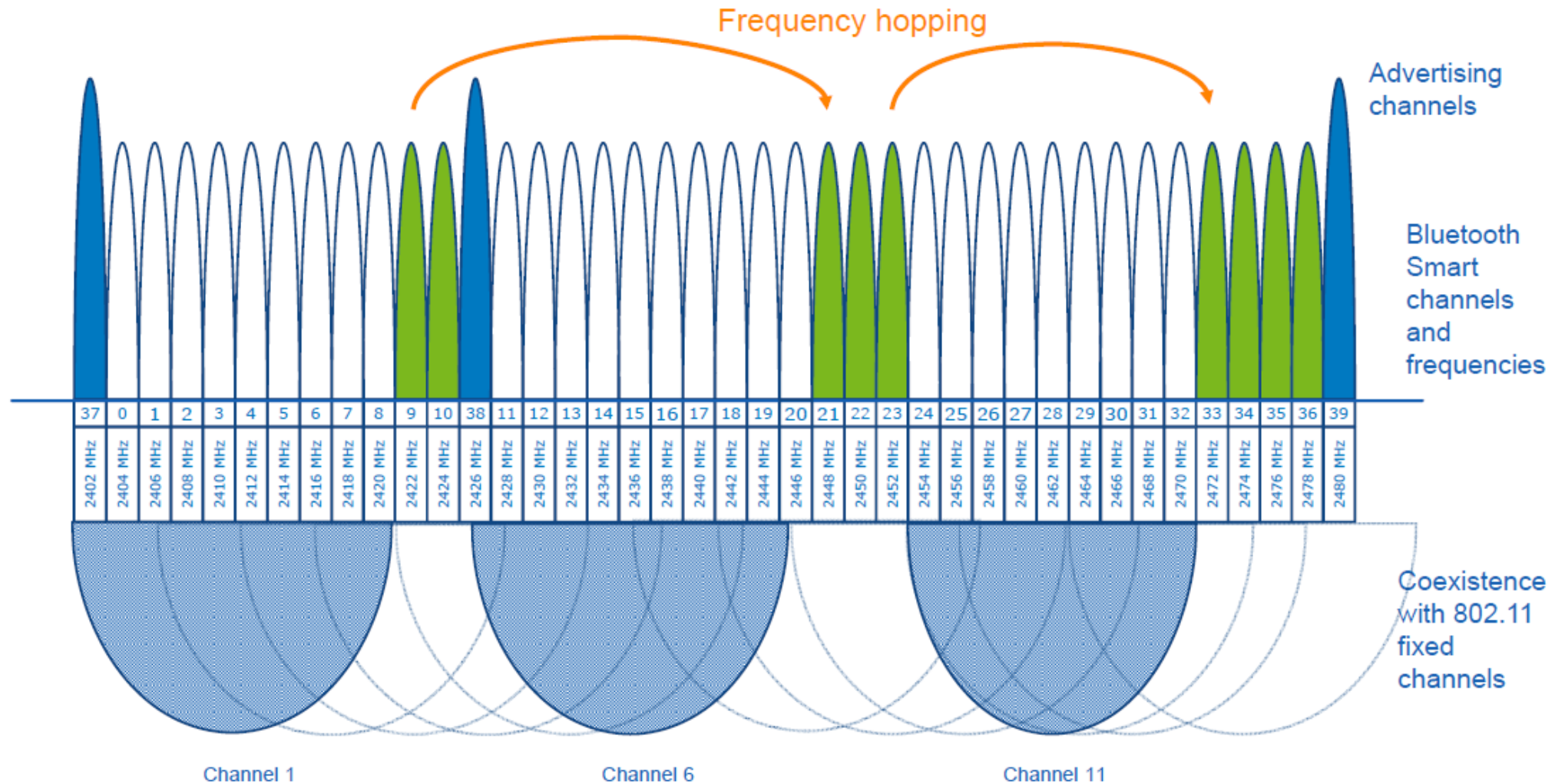


# Salto de Frequência Adaptativo



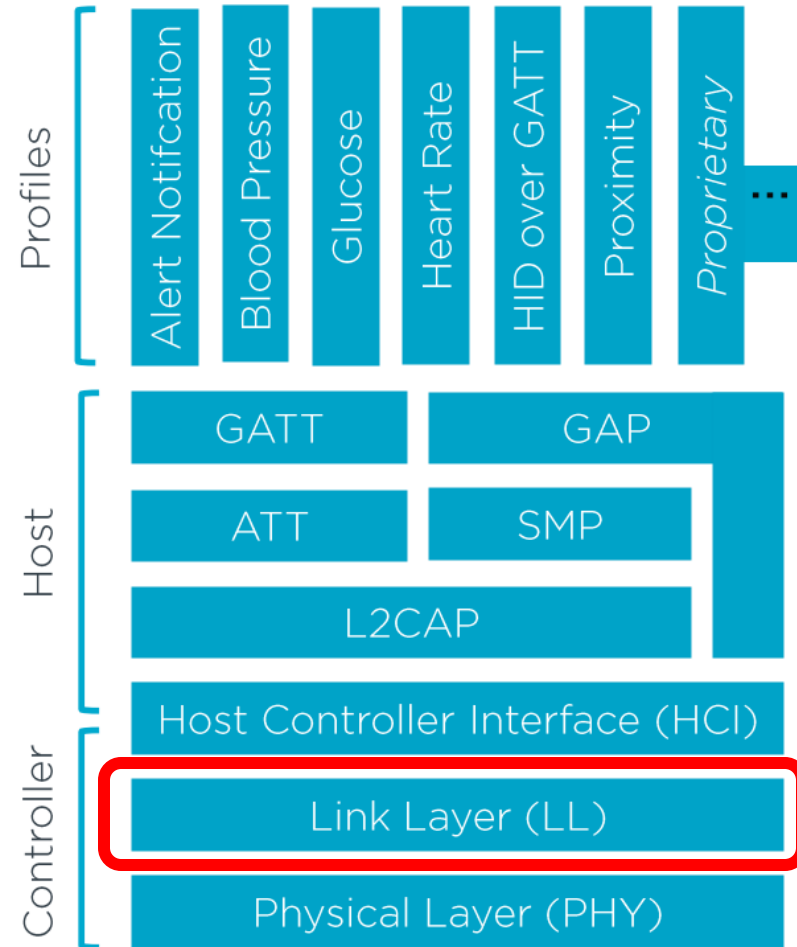
- Esse mecanismo é usado para que a interferência de outros dispositivos (ou seja, Wi-Fi) seja reduzida.
- O dispositivo BLE em coexistência com Wi-Fi nos canais 1, 6 e 11, marcaria os canais BLE 0-8, 11-20 e 24-32 como canais ruins (canais que se sobrepõem). Remapearão os canais restantes para um conjunto de bons canais, conforme mostrado a seguir, destacados em verde

# Coexistência BLE e WiFi





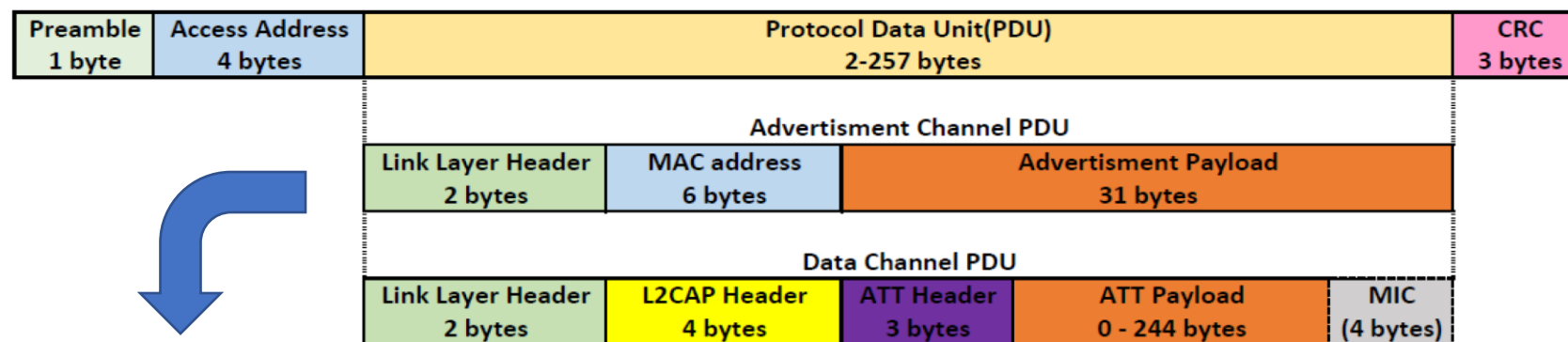
# Controller: Link Layer



# Link Layer (LL)

- Responsável, entre outros, por:
  - Formatar os dados em um pacote link layer (Pacote BLE);
  - Atender aos requisitos de tempo da especificação BLE

# Pacotes BLE



## • Advertisement

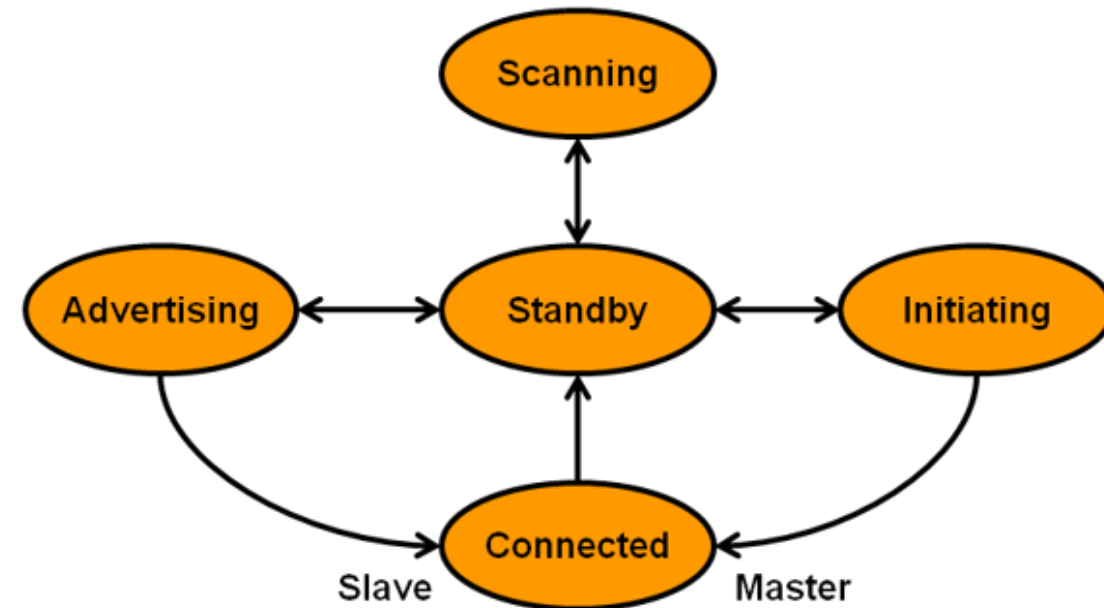
- BT 4.0: (31 bytes payload + 31 opcionais com o pacote de scan\_response);
- BT 5.0: (255 bytes payload com advertising extensions)

## • Data

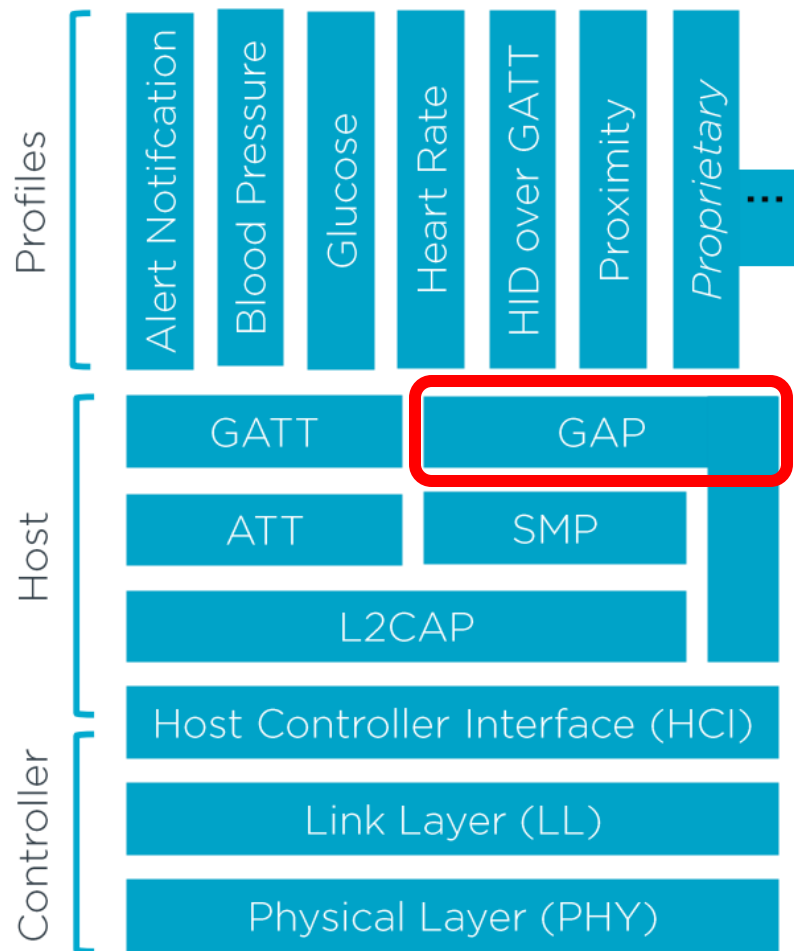
- BT 4.0: (27 byte payload, somente 20 bytes disponíveis para dados de usuário)
- BT 4.2: (251 byte payload, 244 bytes disponíveis para dados de usuário)

# Estados Link Layer

- Define os seguintes estados:
  - Advertising: Um dispositivo que envia pacotes de advertisement
  - Scanning: Um dispositivo que escaneia por pacotes de advertisement
  - Master: um dispositivo que inicia e gerencia uma conexão
  - Slave: dispositivo que aceita requisições de conexão e segue os tempos do master



# Host: GAP



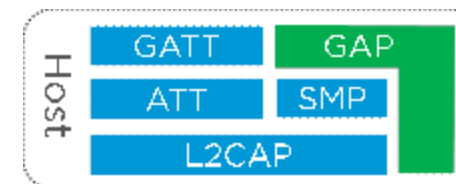
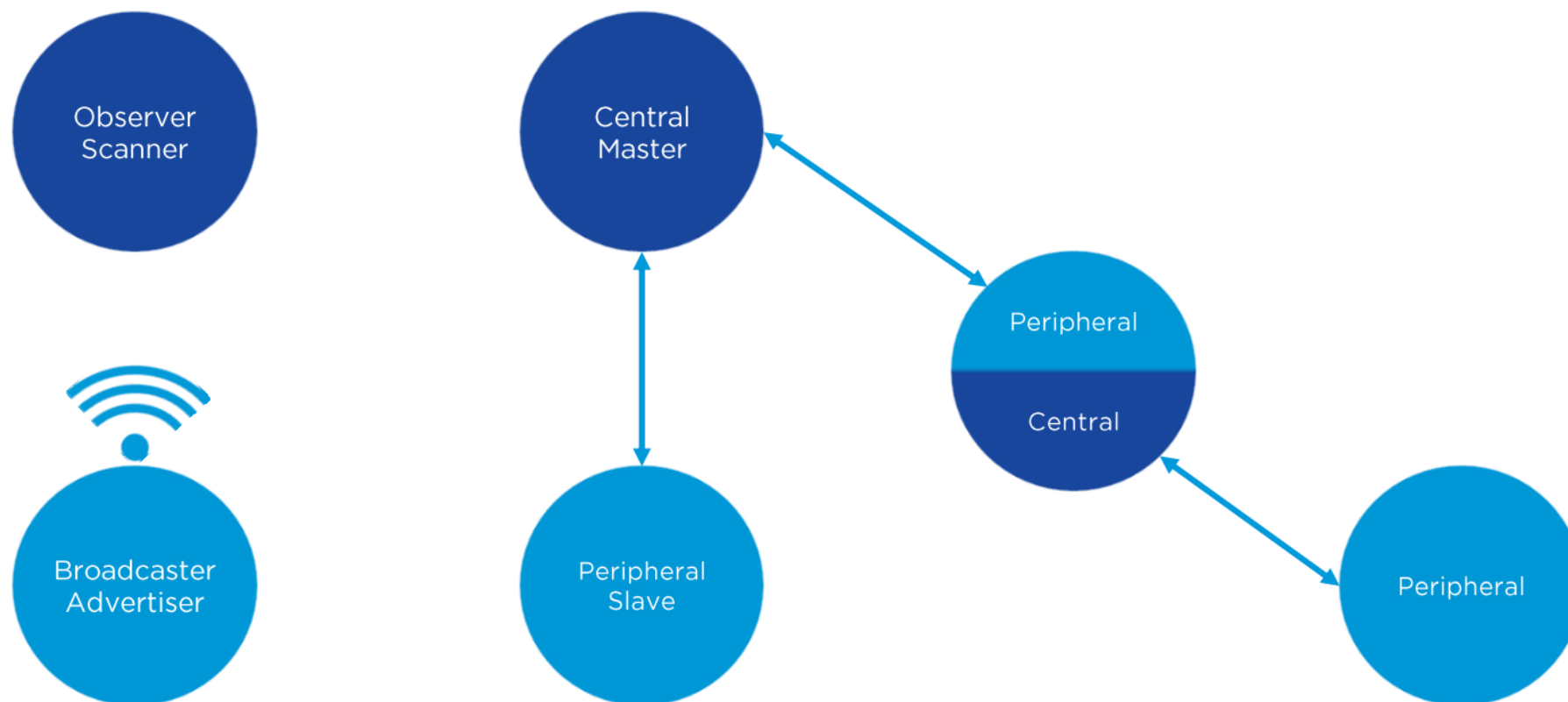
# GAP



- Generic Access Profile
- Define procedimentos de como os dispositivos irão se descobrir, se conectar e apresentar informações entre si
- Define papéis na rede:

GAP role	Link Layer state
Broadcaster	Advertising
Observer	Scanning
Peripheral	Advertising Connection (Slave)
Central	Scanning Initiating Connection (Master)

# Topologia de rede



# Operações GAP

- Setar os dados de advertisement
- Começar/Parar advertising
- Começar/Parar scanning
- Conectar/Desconectar um dispositivo
- Atualizar os parâmetros de conexão
- Encriptar o link



# Conexões BLE

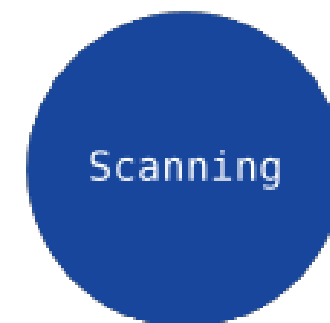


- Os estados se alternam durante as fases de conexões
  - Broadcast (multi ponto) ou
  - Unicast (ponto a ponto)

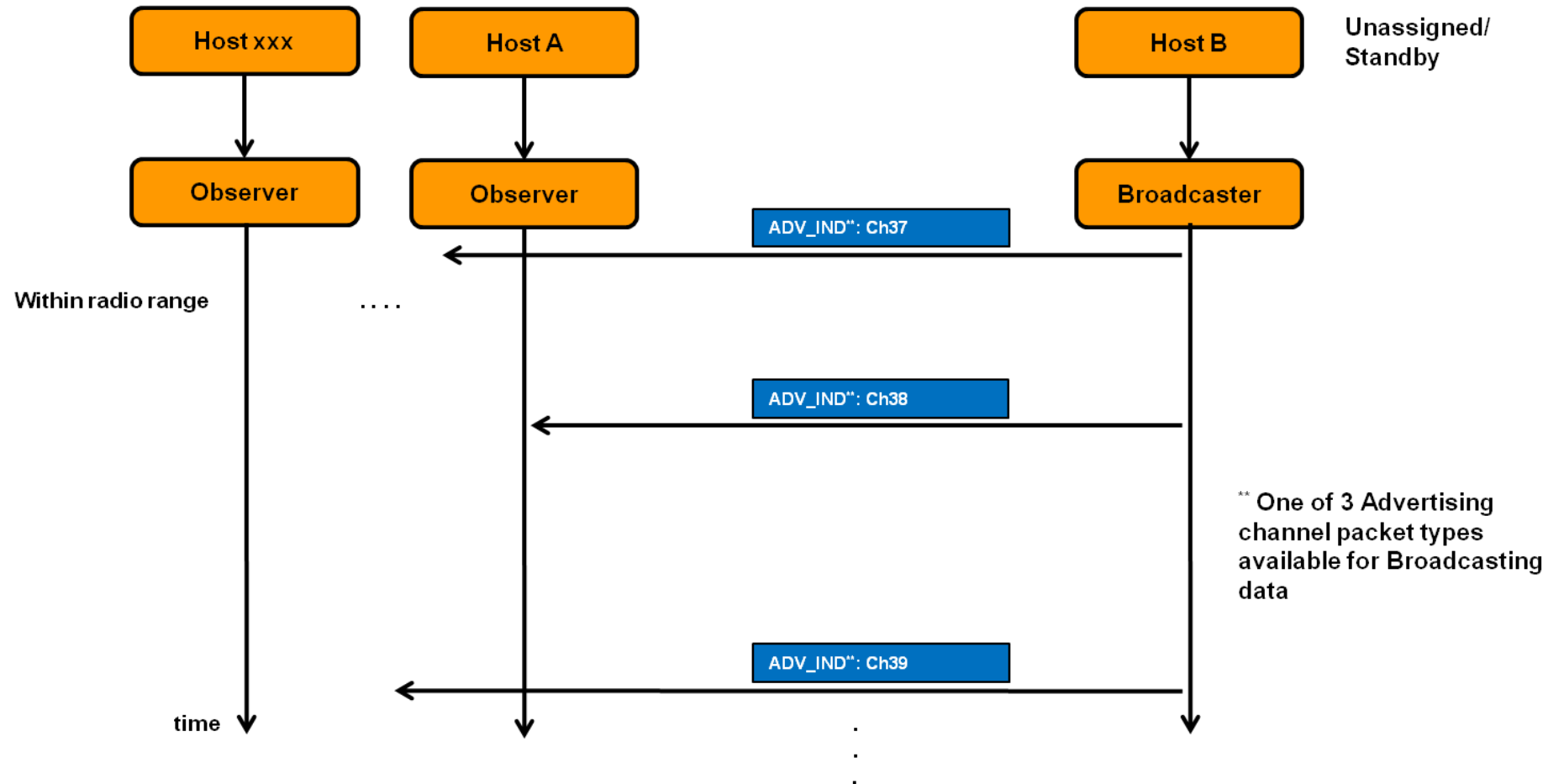
# Broadcast



- Os estados do Link Layer não se alteram
- Os estados definidos são
  - Broadcaster (Advertising): que envia os pacotes
  - Observer (Scanning)
- Note que:
  - As mensagens são unidirecionais
  - As mensagens são de um para muitos



# Broadcast (cont.)



# Broadcast

- Broadcast é adequado para transmissão de dados publicamente
- Utilizado em Beacons para serviços de localização de baixa precisão (margem de erro de 1,5m)
- Baseia-se na comparação do RSSI
- Para maior exatidão (margem de erro de poucos centímetros) é recomendado utilizar o BLE 5.1 que utiliza tecnologia baseada em Ângulo de Chegada e Ângulo de Partida.

# Exemplo beacon

- Baixe os arquivos da [Aula11.zip](#)
- Descompacte o arquivo dentro da pasta myCodes
- Abra o arquivo beacon.c e altere o nome do dispositivo:

```
static const char *TAG = "ibeacon";  
const char *name = "ESP_DILSON";
```

- Execute o arquivo beacon.c (altere o arquivo CMakeLists.txt)

# Exemplo beacon (cont.)

## ESP32: monitor

```
I (994) ibeacon: started
I (994) TAG: Device BLE MAC: 24:0a:c4:61:1c:7a
I (994) TAG: major=c461 = 50273
I (994) TAG: minor=1c7a = 7290
W (1004) ibeacon: ESP_GAP_BLE_ADV_DATA_RAW_SET_COMPLETE_EVT
W (1024) ibeacon: ESP_GAP_BLE_ADV_START_COMPLETE_EVT
```

## App: Beacon Scanner

iBeacon


• 0x004C

0,24 m  
Há 0 seg.

UUID  
fda50693-a4e2-4fb1-afcf-c6eb07647825

Major	Minor	RSSI	TX
50273	7290	-66 dBm	-69 dBm

## App: nRF Connect




N/A (iBeacon)  
24:0A:C4:61:1C:7A  
NOT BONDED ▲ -72 dBm ↔ 43 ms

Device type: LE only  
Advertising type: Legacy  
Flags: GeneralDiscoverable, BrEdrNotSupported

Beacon:  
Company: Apple, Inc. <0x004C>  
Type: Beacon <0x02>  
Length: 21 bytes  
UUID: fda50693-a4e2-4fb1-afcf-c6eb07647825  
Major: 50273  
Minor: 7290  
RSSI at 1m: -69 dBm

# Raw Advertising



N/A (iBeacon)  
24:0A:C4:61:1C:7A  
NOT BONDED ▲ -72 dBm ↔ 43 ms

Device type: LE only  
Advertising type: Legacy  
Flags: GeneralDiscoverable, BrEdrNotSupported  
**Beacon:**  
Company: Apple, Inc. <0x004C>  
Type: Beacon <0x02>  
Length: 21 bytes  
UUID: fda50693-a4e2-4fb1-afcf-c6eb07647825  
Major: 50273  
Minor: 7290  
RSSI at 1m: -69 dBm

CLONE **RAW** MORE

## Raw data:

0x0201061AFF4C000215FDA50693A4E24F  
B1AFCFC6EB07647825C4611C7ABB



## Details:

LEN.	TYPE	VALUE
2	0x01	0x06
26	0xFF	0x4C000215FDA50693A4E24FB1AFC FC6EB07647825C4611C7ABB

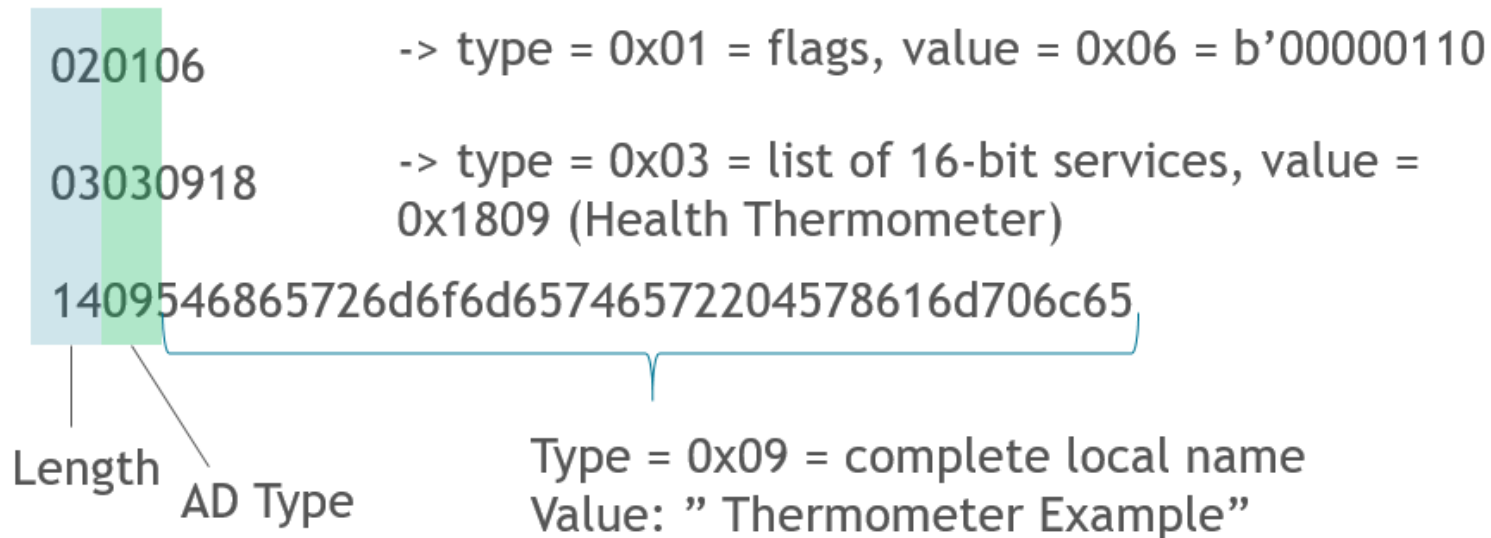
LEN. - length of EIR packet (Type + Data) in bytes,  
TYPE - the data type as in [Generic Access Profile.pdf](#)

# Raw Advertising

Raw advertising data (28 bytes)

020106030309181409546865726d6f6d65746572204578616d706c65

Same data split into AD elements:



Tipo de Dado	Nome do Tipo de Dado
0x01	Flags
0x02	Incomplete List of 16-bit Service Class UUIDs
0x03	Complete List of 16-bit Service Class UUIDs
...	...
0x06	Incomplete List of 128-bit Service Class UUIDs
0x07	Complete List of 128-bit Service Class UUIDs
0x08	Shortened Local Name
0x09	Complete Local Name
...	...
0xFF	Manufacturer Specific Data

0xxx (Tipo de Dado)



# iBeacon advertisement packet

Byte 0: Length : 0x02  
Byte 1: Type: [0x01 \(Flags\)](#)  
Byte 2: Value: 0x06 (Typical Flags 0b00000110) (LE General Discoverable Mode, BR/EDR Not Supported)

Major: 50273  
Minor: 7290  
RSSI at 1m: -69 dBm

CLONE

RAW

MORE

## nRF Connect: RAW

Raw data:

0x0201061AFF4C000215FDA50693A4E24F  
B1AFCFC6EB07647825C4611C7ABB



Details:

LEN.	TYPE	VALUE
2	0x01	0x06
26	0xFF	0x4C000215FDA50693A4E24FB1AFC FC6EB07647825C4611C7ABB

LEN. - length of EIR packet (Type + Data) in bytes,  
TYPE - the data type as in [Generic Access Profile.pdf](#)

## nRF Connect: MORE

Flags:

00000110 = 0x06

LE Limited Discoverable Mode  
LE General Discoverable Mode  
BR/EDR Not Supported  
LE and BR/ERD Capable (Controller)  
LE and BR/ERD Capable (Host)  
Reserved

# iBeacon advertisement packet (cont.)

- Byte 3-29: Apple Defined iBeacon Data

Byte 3: Length: 0x1a (Of the following section)  
Byte 4: Type: 0xff (Custom Manufacturer Data)  
Byte 5-6: Manufacturer ID : 0x4c00 (Apple's Bluetooth SIG registered company code, 16-bit Little Endian)  
Byte 7: SubType: 0x02 (Apple's iBeacon type of Custom Manufacturer Data)  
Byte 8: SubType Length: 0x15 (Of the rest of the iBeacon data; UUID + Major + Minor + TXPower)  
Byte 9-24: Proximity UUID (Random or Public/Registered UUID of the specific beacon)  
Byte 25-26: Major (User-Defined value)  
Byte 27-28: Minor (User-Defined value)  
Byte 29: Measured Power (8 bit Signed value, ranges from -128 to 127, use Two's Complement to "convert" if necessary, Units: Measured Transmission Power in dBm @ 1 meters from beacon) (Set by user, not dynamic, can be used in conjunction with the received RSSI at a receiver to calculate rough distance to beacon)

## [0xFF \(Manufacturer Specific Data\)](#)

0xFF	«Manufacturer Specific Data»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.4 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.4 and 18.11 (v4.0)Core Specification Supplement, Part A, section 1.4
------	------------------------------	---

# iBeacon advertisement packet (cont.)



Raw data:

```
0x0201061AFF4C000215FDA50693A4E24F
B1AFCFC6EB07647825C4611C7ABB
```



Details:

LEN.	TYPE	VALUE
2	0x01	0x06
26	0xFF	0x4C000215FDA50693A4E24FB1AFC FC6EB07647825C4611C7ABB

LEN. - length of EIR packet (Type + Data) in bytes,  
TYPE - the data type as in [Generic Access Profile.pdf](#)

OK

Byte 3: Length: 0x1A ou 26

Byte 4: Type: 0xFF

Byte 5-6: Manufacturer ID 0x4C00

Byte 7: SubType: 0x02

Byte 8: SubType Length: 0x15 ou 21

Byte 9-24: Proximity UUID

0xFDA50693A4E24FB1AFCFC6EB07647825

Byte 25-26: Major 0xC461 ou 50273

Byte 27-28: Minor 0x1C7A ou 7290

Byte 29: Measured Power 0xBB ou -69

[https://www.hobbyprojects.com/calculators/binary\\_decimal\\_hex\\_converter.html](https://www.hobbyprojects.com/calculators/binary_decimal_hex_converter.html)

# Advertising e Scan Response Data



- Existem duas maneiras de enviar advertising com o GAP:
  - advertising data
  - scan response.
- Ambos os payloads podem conter até 31 bytes de dados
- o scan response é uma payload opcional e permite enviar mais dados, como strings para um nome de dispositivo, etc.

# Habilitando scan response

- Descomente a linha


```
static const char *TAG = "ibeacon";  
const char *name = "ESP_DILSON";
```

```
//#define SCAN_RSP
```

```
#ifdef SCAN_RSP  
char scan_rsp[31] = {'0'};  
#endif
```



# Scan Response: Exemplo


**ESP\_DILSON (iBeacon)**  
 Z4:0A:C4:61:1C:7A  
 NOT BONDED -68 dBm ↔ 43 ms  
  
 Device type: LE only  
 Advertising type: Legacy  
 Flags: GeneralDiscoverable, BrEdrNotSupported  
 Beacon:  
 Company: Apple, Inc. <0x004C>  
 Type: Beacon <0x02>  
 Length: 21 bytes  
 UUID: fda50693-a4e2-4fb1-afcf-c6eb07647825  
 Major: 50273  
 Minor: 7290  
 RSSI at 1m: -69 dBm  
 Complete Local Name: ESP\_DILSON  
  
 CLONE RAW MORE

Raw data:  
 0x0201061AFF4C000215FDA50693A4E24FB1AFCFC6EB07647825C4611C7ABB0B094553505F44494C534F4E  
  
 Details:  

LEN.	TYPE	VALUE
2	0x01	0x06
26	0xFF	0x4C000215FDA50693A4E24FB1AFCFC6EB07647825C4611C7ABB0B094
11	0x09	0x4553505F44494C534F4E

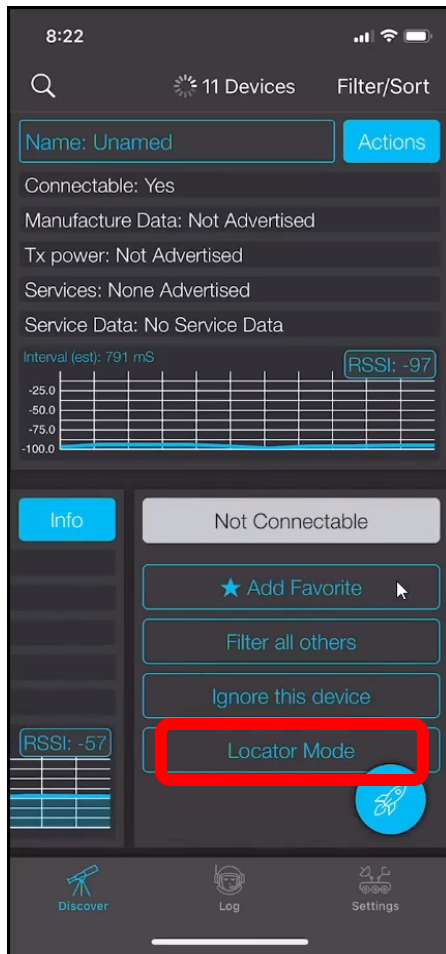
 LEN - length of EIR packet (Type + Data) in bytes,  
 TYPE - the data type as in [Generic Access Profile.pdf](#)  
  
 OK

## 0x09 (Complete Local Name)

0x09	«Complete Local Name»	Bluetooth Core Specification: Vol. 3, Part C, section 8.1.2 (v2.1 + EDR, 3.0 + HS and 4.0) Vol. 3, Part C, sections 11.1.2 and 18.4 (v4.0) Core Specification Supplement, Part A, section 1.2
------	-----------------------	---

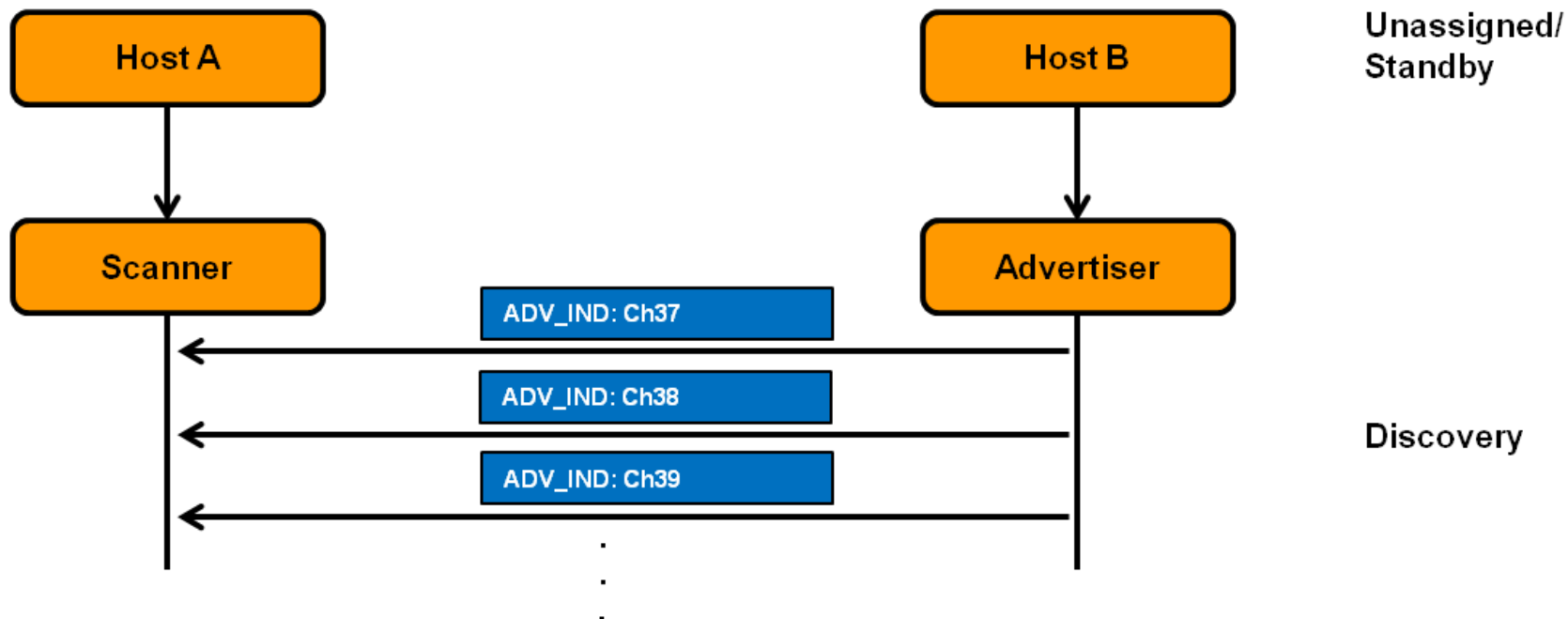


# BLE Hero (iOS)



# Unicast: discovery

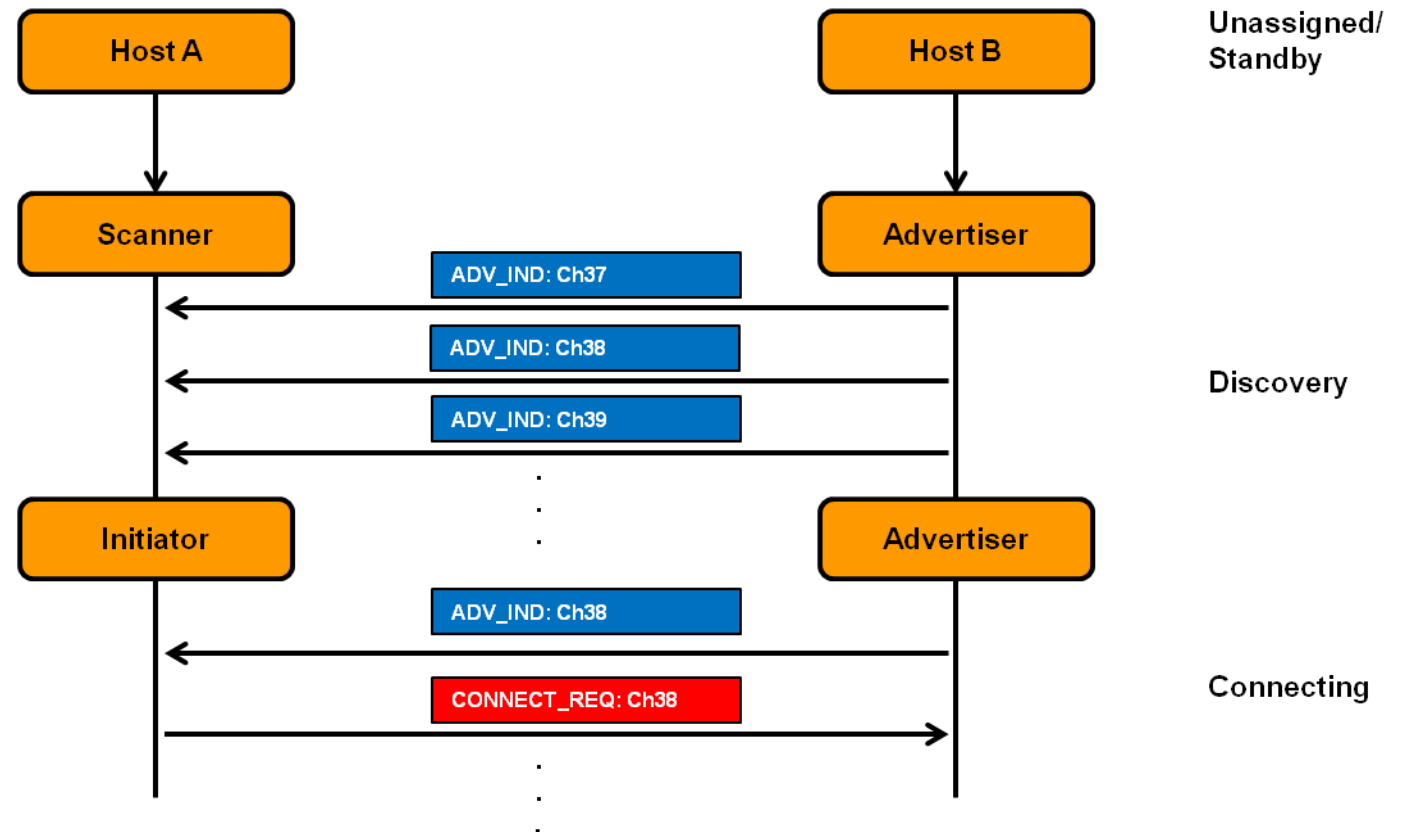
- Os dispositivos entram em um estado de Descoberta onde o dispositivo que deseja ser descoberto se torna o **Advertiser** e o dispositivo que deseja se conectar se torna o **Scanner**





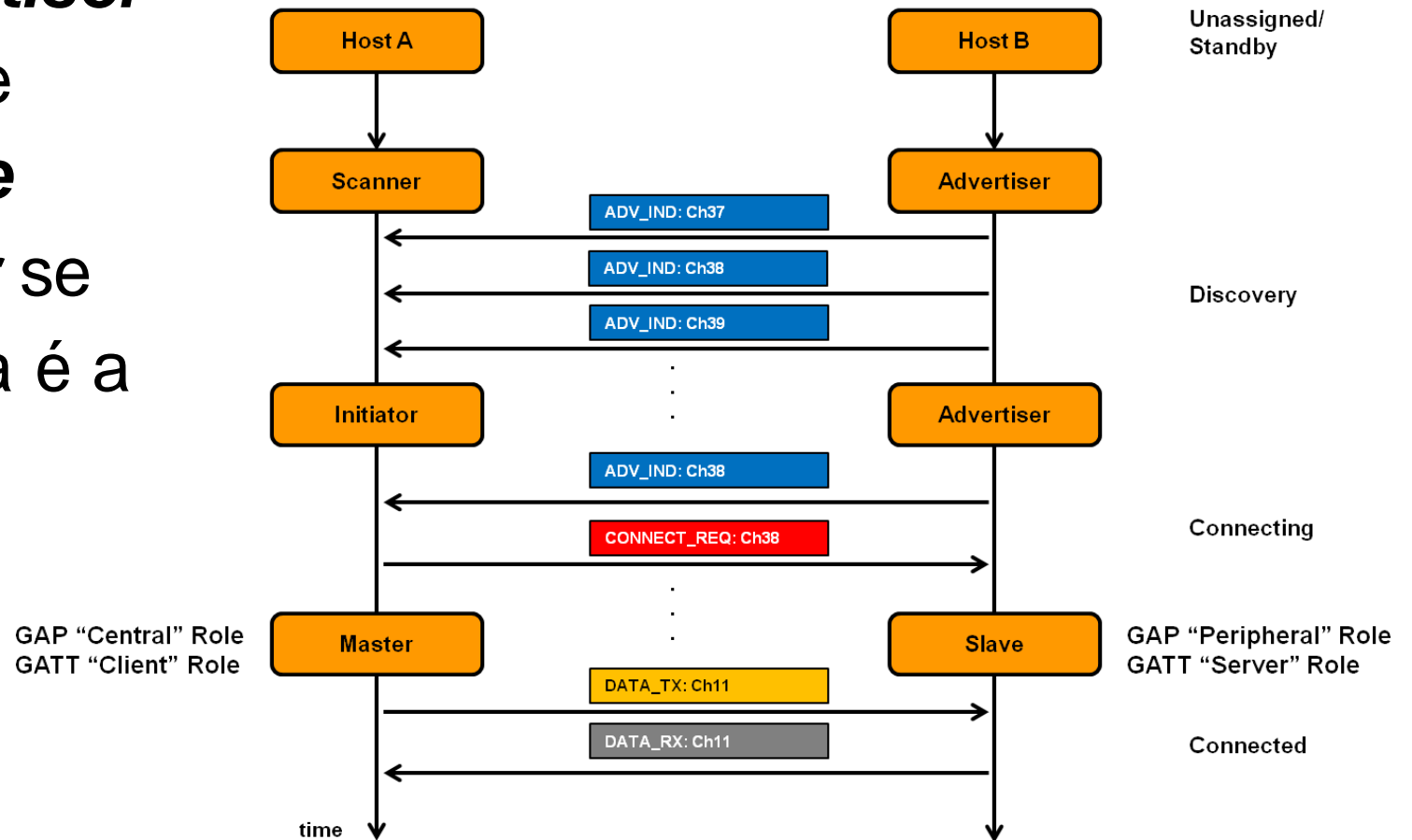
# Unicast: Connecting

- O **Scanner** se torna um **Initiator** e inicia uma conexão com um **Advertiser** específico. Isso é conhecido como a fase de Conexão, e é destacado com o envio de um pacote de **CONNECT\_REQ**

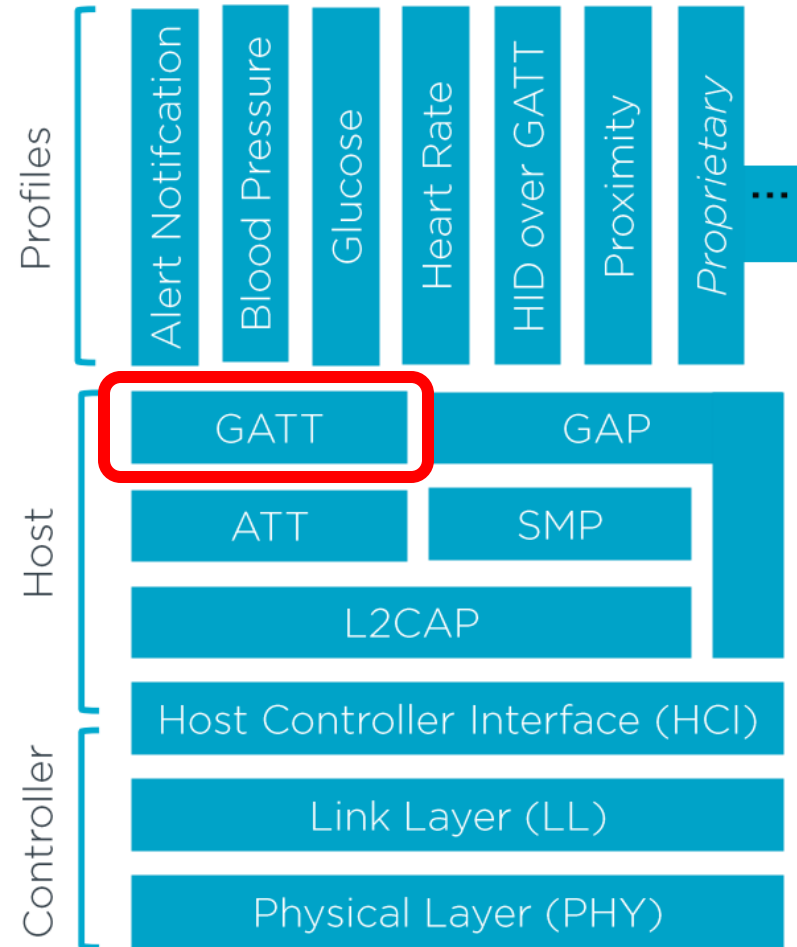


# Unicast: connected

- Finalmente o **Advertiser** aceita a conexão, se tornando o **Slave** enquanto o **Initiator** se torna o **Master**. Esta é a fase Conectada



# Host: GATT



# GAP e GATT



- O GAP fornece uma estrutura padrão para controlar um dispositivo BLE,
- enquanto o GATT fornece uma estrutura padrão para gerenciar dados em um dispositivo BLE.

# GATT



- Define procedimentos de como os dados são trocados em uma conexão BLE
- Utiliza uma arquitetura cliente-servidor

# GATT Servidor x Cliente

- Servidor (dispositivo embarcado)
  - Contém os recursos (dados) a serem monitorados
  - Recebe requisições de um cliente e envia respostas
  - Associado com o Link Layer Slave e GAP Peripheral
- Cliente (smartphone, tablet, computador)
  - Averigua sobre a presença e a natureza dos atributos em um servidor
  - Envia requisições ao servidor e recebe respostas
  - Associado com o Link Layer Master e GAP Central

# GATT: Atributos

- Os atributos são organizados hierarquicamente:
  - Serviços (services)
  - Características (characteristics)
  - Descritores (descriptors)

# UUID



- Universal Unique Identifier
- Serviços, características e descritores todos tem um UUID
- É um número de 16 bits (Atributos Adotados pelo SIG Bluetooth)
  - <https://btprodspecificationrefs.blob.core.windows.net/assigned-values/16-bit%20UUID%20Numbers%20Document.pdf>
  - Os 16 bits são na verdade parte de um UUID de 128 bits padrão
  - 0000XXXX-0000-1000-8000–00805f9b34fb
- Ou 128 bits (tipos de atributos customizáveis): vendor specific UUID

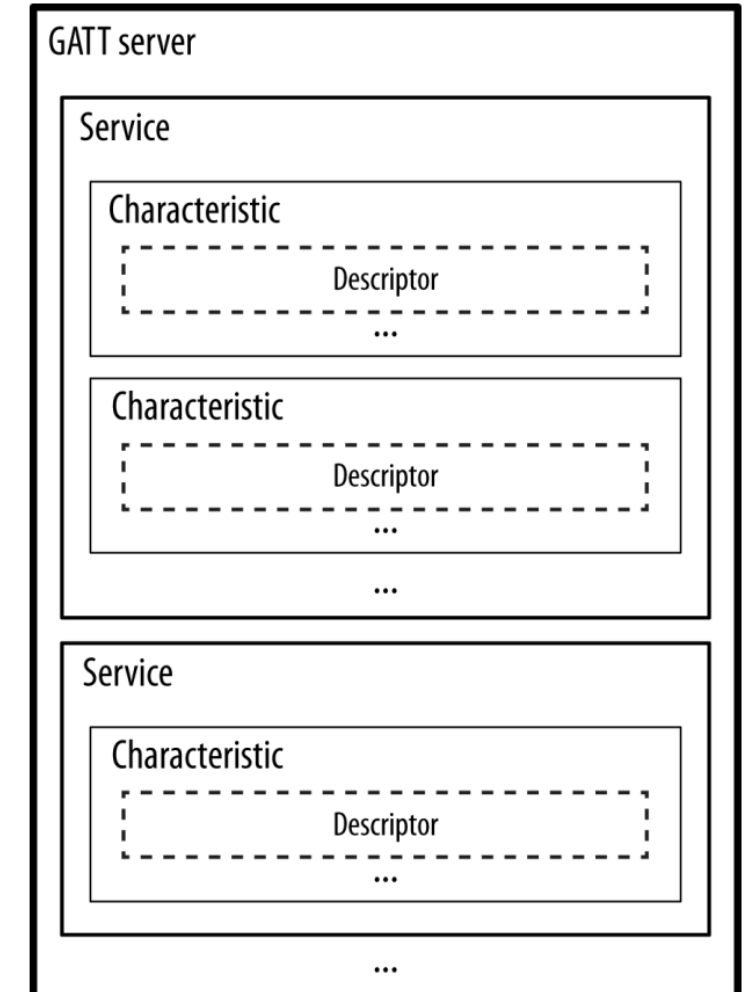


# GATT: serviços

- Um servidor GATT contém um ou mais **serviços** GATT.
- Um serviço contém uma ou mais **características**.
- Ex.: Serviço de Bateria definido pela SIG: [0x180F](#)

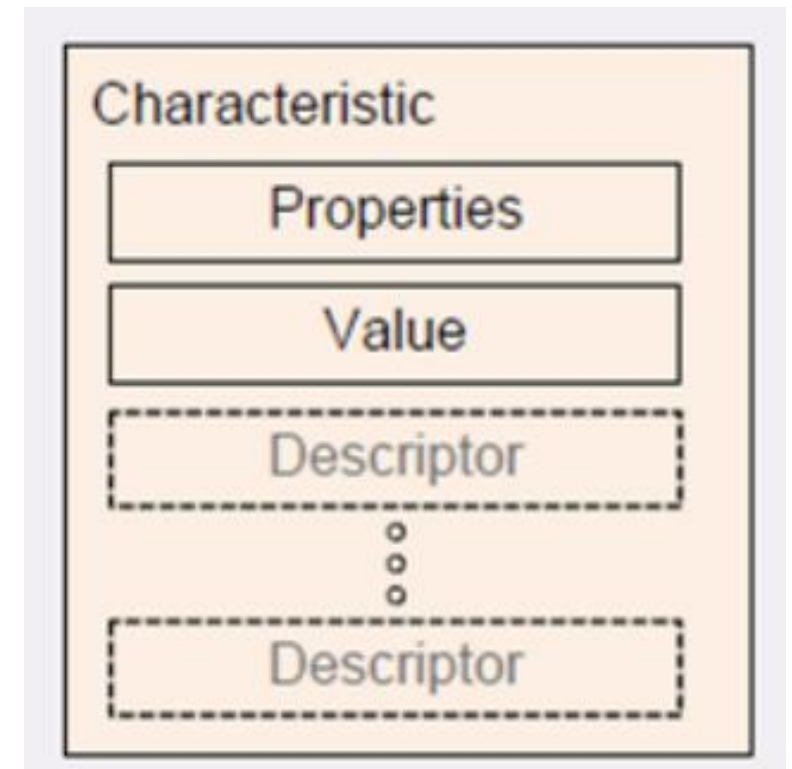
- 0000180F-0000-1000-8000-00805F9B34FB

GATT Service	0x180D	Heart Rate
GATT Service	0x180E	Phone Alert Status
GATT Service	0x180F	Battery
GATT Service	0x1810	Blood Pressure
GATT Service	0x1811	Alert Notification



# GATT: característica

- Uma característica encapsula pelo menos dois atributos:
  - o atributo de declaração da característica: contém metadados sobre a característica
  - atributo de valor: contém os dados de usuário em seu campo de valor



# Metadados da Característica



- UUID da característica: o UUID pode ser de 16 bits ou vendor specific 128 bits
- Handle do Valor da Característica: É o valor necessário para o cliente referenciar o valor. Pense sobre isso como um endereço do dado
- Propriedades da característica: Lista as operações permitidas no valor da característica

# Propriedades da Característica



- Read: permite que o cliente leia o valor da característica;
- Write: o cliente pode escrever no valor da característica e espera receber uma resposta do servidor;
- Notify: habilita o servidor a enviar dados para o cliente a qualquer momento. É também conhecido como transferência de dados iniciado pelo servidor. Requer um descriptor;
- Indicate: o mesmo que o notify, exceto que o cliente tem que enviar uma resposta na chegada do dado (confirmação de recebimento)

# Do profile de tensão de bateria



- Ex.: Característica de tensão de bateria:
  - [0x2A19](#)

GATT Characteristic and Object Type	0x2A17	Time Update State
GATT Characteristic and Object Type	0x2A18	Glucose Measurement
GATT Characteristic and Object Type	0x2A19	Battery Level
GATT Characteristic and Object Type	0x2A1C	Temperature Measurement
GATT Characteristic and Object Type	0x2A1D	Temperature Type

- Com [read e notify implementados](#)

```
esp_ble_gatts_add_char(service_def.service_handle,  
    &service_def.char_uuid,  
    ESP_GATT_PERM_READ | ESP_GATT_PERM_WRITE,  
    ESP_GATT_CHAR_PROP_BIT_READ | ESP_GATT_CHAR_PROP_BIT_NOTIFY,  
    &sensor_data, NULL);
```

# GATT: Descriptor



- Em adição à declaração e ao valor, a característica pode incluir um descriptor;
- Em geral um descriptor é utilizado para:
  - Incluir mais metadados sobre a característica, por exemplo, especificar a unidade utilizada no valor
  - Habilitar ou desabilitar indicate e notify. Este tipo de descriptor é conhecido como CCCD (Client Characteristic Configuration Descriptor)

# Descriptor e Notify

- Adicionando descriptor [0x2902](#)

GATT Descriptor	0x2900	Characteristic Extended Properties
GATT Descriptor	0x2901	Characteristic User Description
GATT Descriptor	0x2902	Client Characteristic Configuration
GATT Descriptor	0x2903	Server Characteristic Configuration

```
esp_ble_gatts_add_char_descr(service_def.service_handle,  
    &service_def.descr_uuid,  
    ESP_GATT_PERM_READ | ESP_GATT_PERM_WRITE,  
    &notify_data, NULL);
```

# Exemplo Perfil Bateria

- Abra o arquivo batt.c
- Altere o nome do dispositivo

```
#include "sdkconfig.h"
#include "battery_service.h"


#define TAG "batt"
#define SENSOR_NAME "ESP32-DILSON"

EventGroupHandle_t events;
#define NOTIFY_ENABLED_BIT BIT0
#define CONNECTED_BIT BIT1
```

- Execute o arquivo batt.c (altere o arquivo CMakeLists.txt)



# Exemplo Perfil Bateria


**ESP32-DILSON**  
 24:0A:C4:61:1C:7A  
 NOT BONDED ▲ -59 dBm ↔ 43 ms

CONNECT

Device type: LE only  
 Advertising type: Legacy  
 Flags: GeneralDiscoverable, BrEdrNotSupported  
 Complete Local Name: ESP32-DILSON  
 Complete list of 16-bit Service UUIDs: 0x180F  
 Slave Connection Interval Range: 7.50ms - 20.00ms

CLONE
 

RAW

 MORE

Raw data:

0x0201060D0945535033322D44494C534F  
 4E03030F18051206001000


Details:

LEN.	TYPE	VALUE
2	0x01	0x06
13	0x09	0x45535033322D44494C534F4E
3	0x03	0x0F18
5	0x12	0x06001000

LEN. - length of EIR packet (Type + Data) in bytes,  
 TYPE - the data type as in [Generic Access Profile.pdf](#)

0x03 <a href="#">0x03</a>	«Complete List of 16-bit Service Class UUIDs»	Bluetooth Core Specification:Vol. 3, Part C, section 8.1.1 (v2.1 + EDR, 3.0 + HS and 4.0)Vol. 3, Part C, sections 11.1.1 and 18.2 (v4.0)Core Specification Supplement, Part A, section 1.1
0x12 <a href="#">0x12</a>	«Slave Connection Interval Range»	Bluetooth Core Specification:Vol. 3, Part C, sections 11.1.8 and 18.8 (v4.0)Core Specification Supplement, Part A, section 1.9

# Flags



**ESP32-DILSON**  
24:0A:C4:61:1C:7A  
NOT BONDED    ▲ -59 dBm    ↔ 43 ms

CONNECT

⋮

Device type: LE only  
Advertising type: Legacy  
Flags: GeneralDiscoverable, BrEdrNotSupported  
**Complete Local Name:** ESP32-DILSON  
Complete list of 16-bit Service UUIDs: 0x180F  
Slave Connection Interval Range: 7.50ms - 20.00ms

CLONE

RAW

**MORE**

×

ESP32-DILSON

HISTORY

**FLAGS & SERVICES**

Flags:

00000110 = 0x06

LE Limited Discoverable Mode

LE General Discoverable Mode

BR/EDR Not Supported

LE and BR/ERD Capable (Controller)


LE and BR/ERD Capable (Host)

Reserved

Complete List of 16-bit Service Class UUIDs:

0000180f-0000-1000-8000-00805f9b34fb (Battery Service)

# Conectar



**ESP32-DILSON**  
24:0A:C4:61:1C:7A  
NOT BONDED    ▲ -59 dBm    ↔ 43 ms

**CONNECT** ⋮

Device type: LE only  
Advertising type: Legacy  
Flags: GeneralDiscoverable, BrEdrNotSupported  
**Complete Local Name:** ESP32-DILSON  
Complete list of 16-bit Service UUIDs: 0x180F  
Slave Connection Interval Range: 7.50ms - 20.00ms

CLONE    RAW    MORE

BONDED    ADVERTISER    **ESP32-DILSON** X  
24:0A:C4:61:1C:7A

**CONNECTED**  
NOT BONDED    **CLIENT**    SERVER    ⋮

**Generic Attribute**  
UUID: 0x1801  
PRIMARY SERVICE

**Generic Access**  
UUID: 0x1800  
PRIMARY SERVICE

**Battery Service**  
UUID: 0x180F  
PRIMARY SERVICE

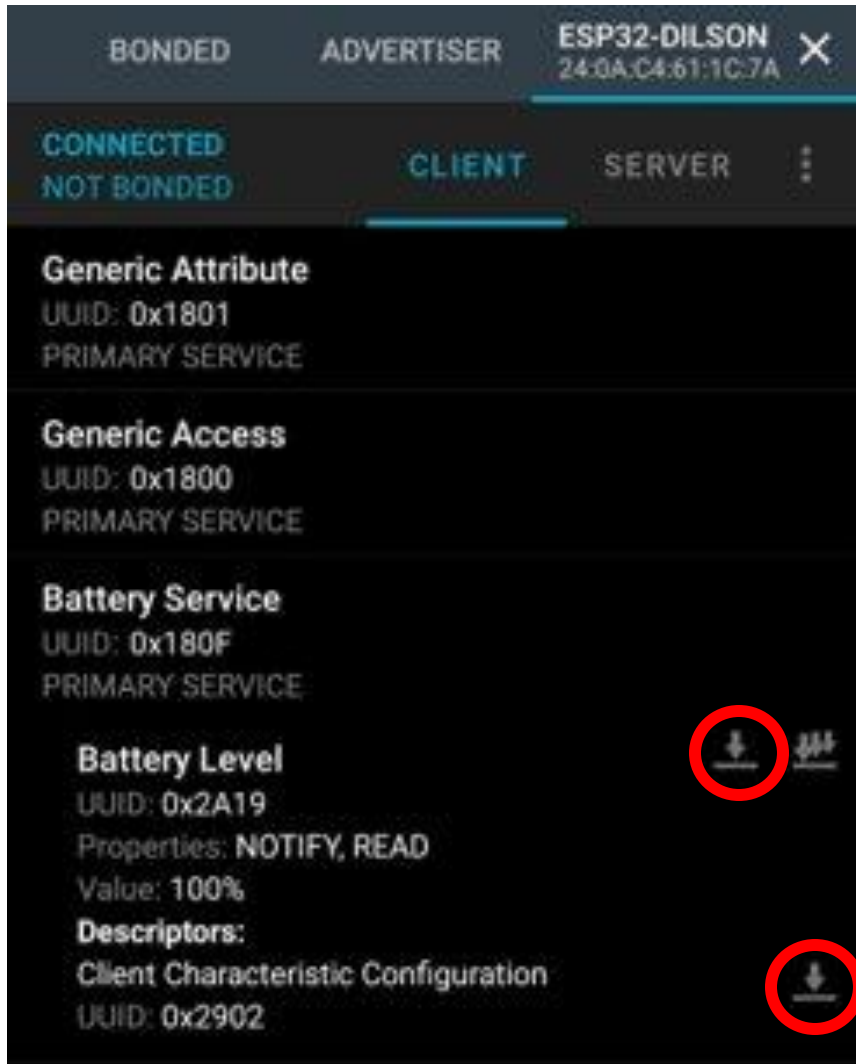
**Battery Level**    ↓    ⇅  
UUID: 0x2A19  
Properties: NOTIFY, READ

**Descriptors:**  
Client Characteristic Configuration  
UUID: 0x2902

**Serviço**

**Característica**

**Descriptor**

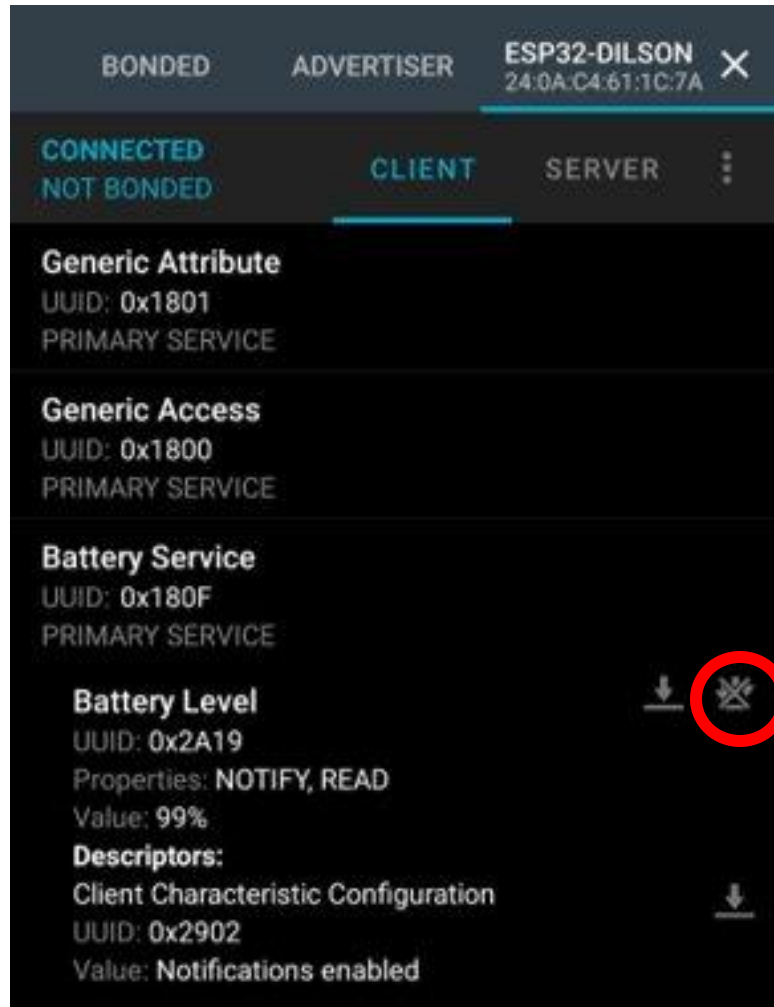


Read  
Operation

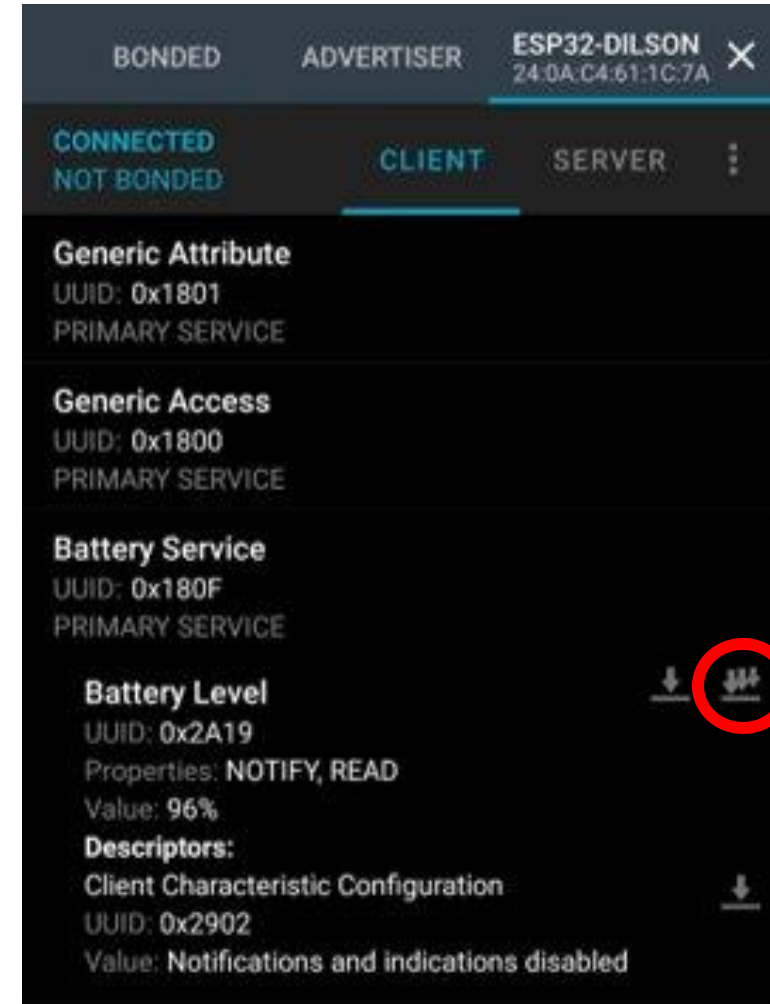
Characteristic

Descriptor CCCD

# Habilitar/Desabilitar Notification



Notification  
Habilitado



Notification  
Desabilitado

# Exercício



- Altere o exemplo anterior e envie informações reais de tensão lidas como no exercício 5 da Aula 02;
- Utilize 0% para 2,8V e 100% para 4,2V;
- As informações deverão ser enviadas a cada 10 segundos pelo notification.



# Referências

- [https://github.com/NordicPlayground/nRF52-Bluetooth-Course/blob/master/pdf/03\\_Bluetooth%20Overview.pdf](https://github.com/NordicPlayground/nRF52-Bluetooth-Course/blob/master/pdf/03_Bluetooth%20Overview.pdf)
- <https://hardwear.io/document/ble-security%20essentials.pdf>
- <https://embeddedcentric.com/introduction-to-bluetooth-low-energy-bluetooth-5/>
- <https://embeddedcentric.com/lesson-2-ble-profiles-services-characteristics-device-roles-and-network-topology/>
- [https://smartlockpicking.com/slides/HiP19\\_A\\_45min\\_introduction\\_to\\_Bluetooth\\_Low\\_Energy\\_workshop.pdf](https://smartlockpicking.com/slides/HiP19_A_45min_introduction_to_Bluetooth_Low_Energy_workshop.pdf)

# Referências (cont.)

- [https://devzone.nordicsemi.com/cfs-file/\\_\\_key/communityserver-discussions-components-files/4/Introduction-to-Bluetooth-Low-Energy.pdf](https://devzone.nordicsemi.com/cfs-file/__key/communityserver-discussions-components-files/4/Introduction-to-Bluetooth-Low-Energy.pdf)
- <https://www.youtube.com/watch?v=5TxUnbsHsR8&t=4s>
- <https://microchipdeveloper.com/wireless:ble-introduction>
- <https://emanuelepagliari.it/2020/10/15/how-bluetooth-low-energy-work-internet-of-things-wireless-sensor-networks/#Generic-Attribute-Profile-GATT-Services-and-Characteristics-for-IoT-communications>



# Referências (cont.)

- <https://novelbits.s3.us-east-2.amazonaws.com/Website/Lead+Magnets/Intro+to+Bluetooth+Low+Energy+v1.1.pdf>
- [https://community.silabs.com/s/article/kba-bt-0201-bluetooth-advertising-data-basics?language=en\\_US](https://community.silabs.com/s/article/kba-bt-0201-bluetooth-advertising-data-basics?language=en_US)



PADO  
**Labs**