Aula 3.5 - Bluetooth

inside.Docupedia Export

Author: Lundgren Daniel (CtP/ETS)
Date: 15-Dec-2021 15:50

Table of Contents

1 Visão geral **4**
2 Algumas aplicações do BLE **5**
3 Comparativo **6**
4 Código **7**
5 Conectando com Smartphone **11**

O **BLE** é um protocolo ligeiramente diferente do Bluetooth tradicional. Ele é mais adequado para aplicações **IoT** (Internet of Things) de **baixo consumo de energia**, já que a capacidade dos pacotes de dados é baixa.

Os dispositivos que trabalham com BLE podem ter duas funções diferentes em uma conexão: **Dispositivo Central ou Dispositivo Periférico** (Central Device or Peripheral Device). Geralmente, os dispositivos centrais são telefones celulares, tablets, computadores, etc. São dispositivos centrais que recebem dados. Já os dispositivos periféricos são sensores e dispositivos low power que se conectam ao dispositivo central. Podemos pensar também como uma estrutura cliente/servidor, onde um celular é o cliente e o sensor é o servidor que “serve” seus dados para o cliente.

GATT (Generic Attribute Profile) é a forma com que os dados são organizados para comunicação entre os dispositivos. GATT é composto por um ou mais serviços que, por sua vez, são compostos de características. Existem especificações padrão de GATT para os tipos de aplicação mais comuns encontradas no mercado. Várias dessas especificações podem ser encontradas no site oficial do Bluetooth. As características, por sua vez, são basicamente os valores em si.

Os serviços e as características são identificados por um **UUID** (universally unique identifier). Assim podemos customizar nossas próprias ou utilizar as já existentes. https://www.bluetooth.com/specifications/gatt/services

Como dito anteriormente, **podemos ter um ou mais serviços rodando no mesmo dispositivo** e, cada um deles, com suas próprias características.

Por exemplo, um ESP32 pode estar rodando um serviço de CLIMA, para enviar dados de temperatura e umidade, e um outro serviço de BATERIA, para enviar dados de medição de carga da bateria.

---

1 Visão geral

Podemos enviar e receber dados em ambos os lados.

Exemplo: ESP32 pode enviar dados de sensores.

Exemplo 2: Smartphone pode enviar comandos para o ESP32 ligar um relé.

2 Algumas aplicações do BLE

- Sensores de batimento cardíaco
- Comunicação M2M (máquina-a-máquina)
- Automatização de tarefas em uma residência
- Controle remoto
- Notificação de alerta de proximidade
- dentre outros

3 Comparativo

	Bluetooth clássico	Bluetooth Low Energy
Preço	Alto	Muito baixo
Consumo de energia	Alto	Muito baixo
Aplicação	Transferências com grandes quantidades de dados	Trasnferências de pequenas quantidades de dados

4 Código

Faremos um programa no qual o **ESP32** se tornará um **servidor BLE**. Quando um dispositivo estiver conectado a ele serão enviados em intervalos de tempo dados de um sensor de temperatura. O ESP32 também poderá receber dados para comandar o Liga/Desliga de um LED.

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>
#include "DHT.h"

BLECharacteristic *characteristicTX; //através desse objeto iremos enviar dados para o client

bool deviceConnected = false; //controle de dispositivo conectado

const int LED = 2; //LED interno do ESP32

// See the following for generating UUIDs:
// https://www.uuidgenerator.net/

#define SERVICE_UUID "ab8028b1-198e-4351-b779-901fa0e0371e" // UART service UUID
#define CHARACTERISTIC_UUID_RX "4aca9682-9736-4e5d-932b-e9b31405049c"
#define CHARACTERISTIC_UUID_TX "0972128C-7613-4075-AD52-756F33D4DA91"
#define DHTPIN 4
//our sensor is DHT11 type
#define DHTTYPE DHT11
//create an instance of DHT sensor
DHT dht(DHTPIN, DHTTYPE);
```

Após isso, devem ser criados os callbacks, que são as funções que rodam quando a ESP recebe alguma resposta. A lógica pode ser de difícil compreensão, pois contém ponteiros, classes e objetos em C++, não se preocupe em entender o código por completo.

Primeiro vamos criar os ServerCallbacks, no nosso caso, executará uma função sempre que um dispositivo for conectado, ou disconectado da ESP.

```
//callback para receber os eventos de conexão de dispositivos
class ServerCallbacks: public BLEServerCallbacks {
    void onConnect(BLEServer* pServer) {
      deviceConnected = true;
    };

    void onDisconnect(BLEServer* pServer) {
      deviceConnected = false;
    }
};
```

Agora vamos criar os callbacks das características, sempre que é recebido um valor, ou seja, o cliente enviou algum comando, a função onWrite é executada (Write pelo ponto de visão do cliente). Nela é feita a leitura da mensagem, se a mensagem for "L1" o LED é ligado, se for "L0" o LED é desligado.

```
//callback para envio das características
class CharacteristicCallbacks: public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *characteristic) {
      //retorna ponteiro para o registrador contendo o valor atual da caracteristica
      std::string rxValue = characteristic->getValue();
      //verifica se existe dados (tamanho maior que zero)
      if (rxValue.length() > 0) {
        Serial.println("**********");
        Serial.print("Received Value: ");

        for (int i = 0; i < rxValue.length(); i++) {
          Serial.print(rxValue[i]);
        }

        Serial.println();

        // Realiza um comando de acordo com a mensagem recebida
        if (rxValue.find("L1") != -1) {
          Serial.print("Turning LED ON!");
          digitalWrite(LED, HIGH);
        }
        else if (rxValue.find("L0") != -1) {
          Serial.print("Turning LED OFF!");
          digitalWrite(LED, LOW);
        }

        Serial.println();
        Serial.println("**********");
      }
    }
};
```

Agora vamos fazer as configurações gerais, nosso **setup**, aqui vamos inicializar a porta serial, o sensor e o BLE.

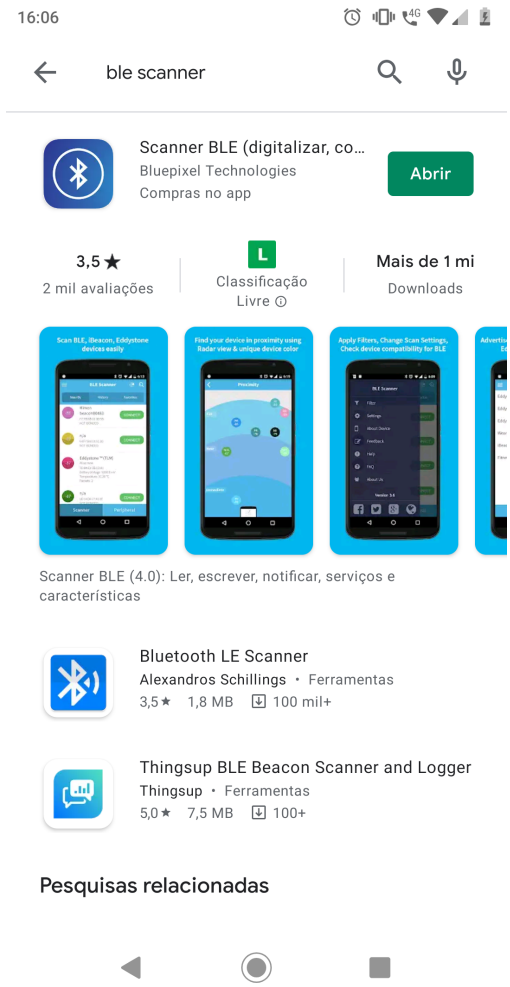
```
void setup() {
  Serial.begin(115200);

  dht.begin();

  pinMode(LED, OUTPUT);

  // Create the BLE Device
  BLEDevice::init("MINHA-ESP"); // nome do dispositivo bluetooth
  // Cria o BLE Server
  BLEServer *server = BLEDevice::createServer(); //cria um BLE server
  server->setCallbacks(new ServerCallbacks()); //seta o callback do server
  // Cria o BLE Service
  BLEService *service = server->createService(SERVICE_UUID);
  // Cria uma BLE Characteristic para envio de dados
  characteristicTX = service->createCharacteristic(
                      CHARACTERISTIC_UUID_TX,
                      BLECharacteristic::PROPERTY_NOTIFY
                    );

  characteristicTX->addDescriptor(new BLE2902());

  // Cria uma BLE Characteristic para recebimento de dados
  BLECharacteristic *characteristic = service->createCharacteristic(
                                         CHARACTERISTIC_UUID_RX,
                                         BLECharacteristic::PROPERTY_WRITE
                                       );

  characteristic->setCallbacks(new CharacteristicCallbacks());
  // Start the service
  service->start();
  // Start advertising (ESP fica visível para outros dispositivos)
  server->getAdvertising()->start();

  Serial.println("Waiting a client connection to notify...");

}
```

O **loop** é muito simples, se existir um dispositivo conectado, a ESP fará a leitura do valor do sensor, converterá de **float** para **char** e envia isso para o cliente. Na porta serial são enviadas mensagens informando o valor que está sendo enviado.

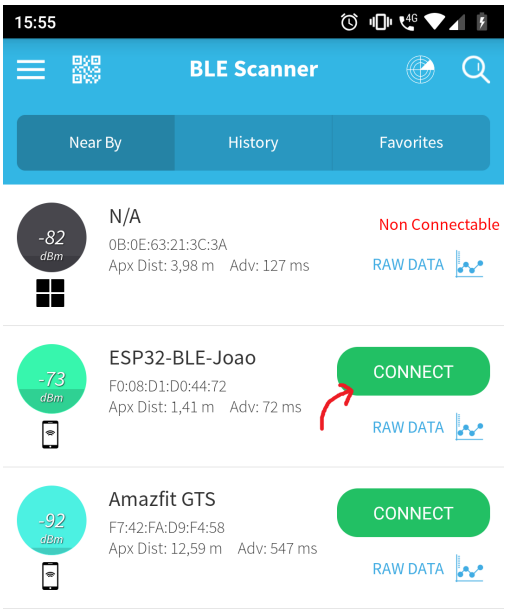
```
void loop() {
     //se existe algum dispositivo conectado
    if (deviceConnected) {
      float t = dht.readTemperature();

      // Convertendo para char para poder enviar pelo BLE:
      char txString[8]; // garantindo que é grande o suficiente
      dtostrf(t, 1, 2, txString); // float_val, min_width, digits_after_decimal, char_buffer

      characteristicTX->setValue(txString); //seta o valor que a caracteristica notificará (enviar)
      characteristicTX->notify(); // Envia o valor para o cliente (smartphone)

      Serial.print("*** Sent Value: ");
      Serial.print(txString);
      Serial.println(" ***");
  }
  delay(1000);

}
```

5 Conectando com Smartphone

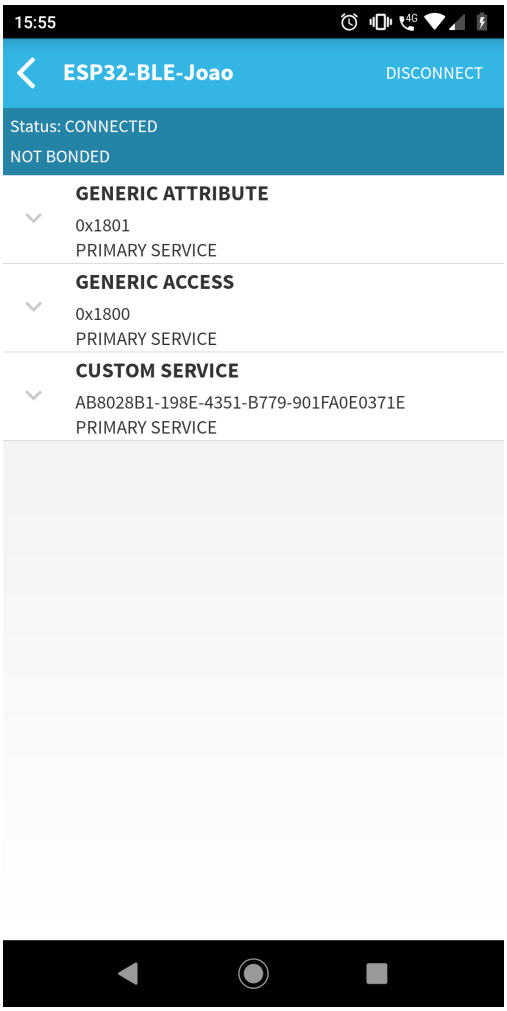
Para fazer a comunicação Bluetooth com a ESP, tanto leitura do sensor quanto envio de comandos, vamos utilizar um app no Smartphone:



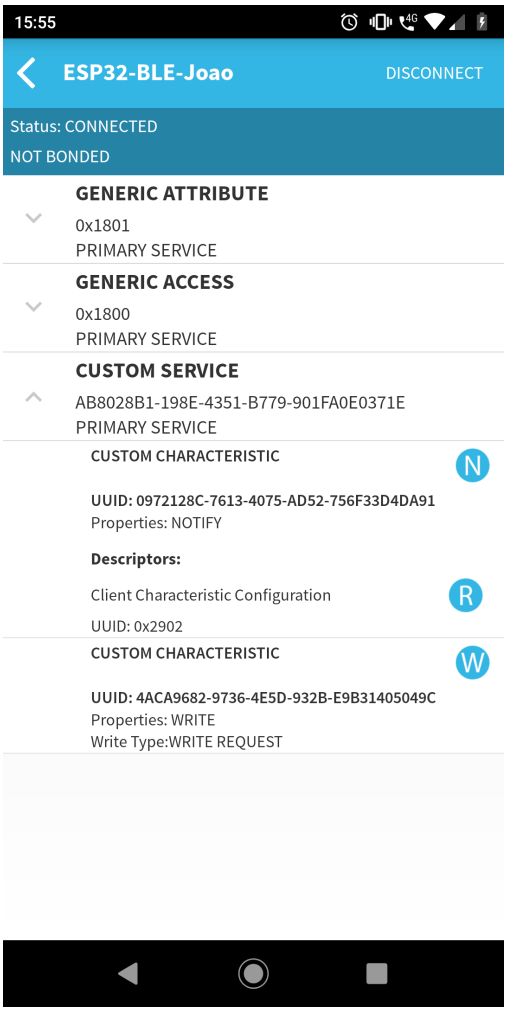
Ao abrir o aplicativo, você terá na tela todos os dispositivos bluetooth encontrados pelo celular. A sua ESP deve aparecer, clique em CONNECT:



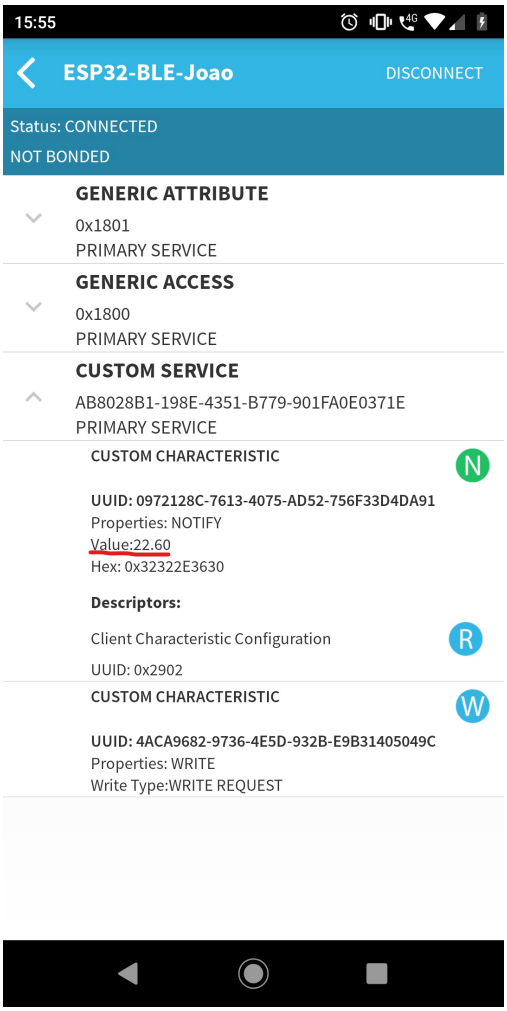
Ao conectar, a seguinte tela será mostrada, na qual aparecem os serviços desse dispositivo, CUSTOM SERVICE é o serviço que nós criamos, o UUID deve ser o mesmo que você configurou:



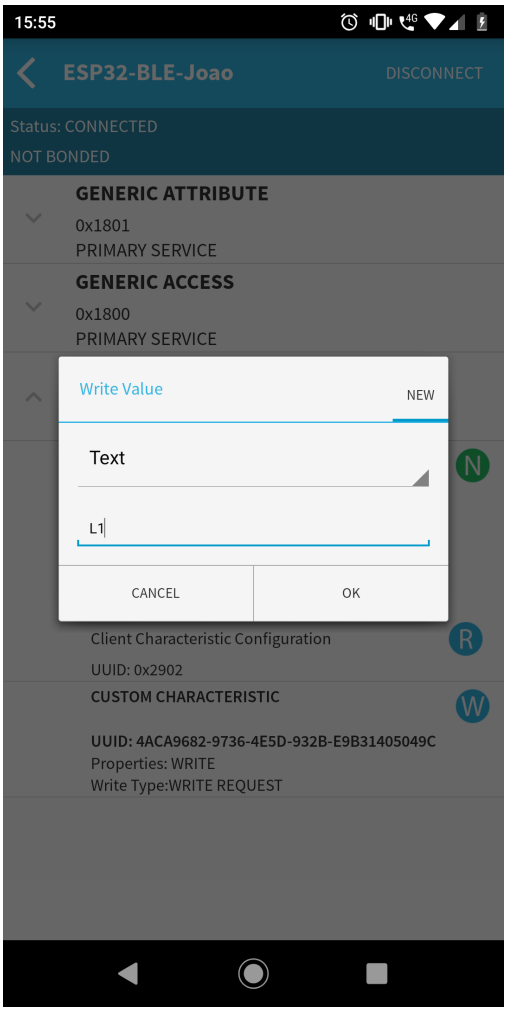
Ao clicar no serviço, as características serão apresentadas:



Ao clicar no círculo escrito **N** o valor que a ESP está enviando será mostrado na tela:



Ao clicar no círculo escrito **W** você poderá escrever um texto para enviar para a ESP. Lembrando do nosso código, ao enviar "L1" o LED deverá ser aceso:



Após seguir todos esses passos o seu sistema com comunicação Bluetooth Low Energy deve estar funcionando!