

UNIVERSIDADE FEDERAL DE SANTA CATARINA  
CURSO DE GRADUAÇÃO EM ENGENHARIA ELETRÔNICA  
ORIENTADOR: PROF. DR. DJONES VINICIUS LETTNIN

LUCIANO SILVA DO LAGO  
MARCOS MEYER HOLLERWEGER

# Bit-Bang BLE Broadcast

Florianópolis - SC  
2016

## Resumo

Este relatório descreve o trabalho de implementação parcial do protocolo BLE (*Bluetooth Low-Energy*) utilizando o módulo nRF24L01+, produzido pela NORDIC SEMICONDUCTOR. Os autores realizaram a implementação em linguagem de programação C++, com objetivo de compilar e executar o programa em um computador Raspberry Pi, em sistema operacional Linux.

## Abstract

*This report describes a partial implementation of the BLE (Bluetooth Low-Energy) protocol using the nRF24L01+ module from NORDIC SEMICONDUCTOR. The authors succeeded in this implementation using C++ programming language, aiming to compile and execute the software in a Raspberry Pi embedded computer, running Linux operating system.*

# SUMÁRIO

<b>SUMÁRIO</b>	<b>2</b>
<b>INTRODUÇÃO</b>	<b>3</b>
<b>REQUISITOS</b>	<b>4</b>
<b>DESENVOLVIMENTO</b>	<b>5</b>
IMPLEMENTAÇÃO BLE	5
MODELAGEM DO SISTEMA	7
<b>VERIFICAÇÃO</b>	<b>11</b>
WIRESHARK	11
NRF CONNECT	13
<b>CONCLUSÃO</b>	<b>15</b>

## 1. INTRODUÇÃO

Ao implementar uma rede de sensores, um engenheiro se depara com diversos tipos de rádios e protocolos de comunicação que pode escolher e que melhor atenda suas necessidades. Um caso muito ao se usar sensores é ser necessário os sensores se comunicar entre si e, ao mesmo tempo, transmitir esses dados para uma aplicação para o usuário final.

Redes de sensores necessitam de baixa transmissão de dados, baixa velocidade de transmissão e baixíssimo consumo de energia. Para tais requisitos existem diversos protocolos, tais como: SubGHz; ZigBee; IRDA; LoRa e muitos outros. Protocolos de dados para usuários (WiFi e Bluetooth principalmente) não foram feitos para atender essa parcela específica de usos pois consomem muita energia, WiFi por exemplo o dispositivo deve estar sempre ativo para que tenha um endereço IP.

A solução aqui apresentada tenta atenuar um pouco este problema, usando um emulador de Bluetooth Broadcast para transmitir dados tanto em Bluetooth para um usuário final quanto no protocolo ShockBurst™ para uma rede de sensores.

## 2. REQUISITOS

A ideia do projeto surgiu a partir da observação da necessidade de uma maneira fácil, rápida e eficiente de obter informações sobre sistemas embarcados. Informações estas que podem tratar-se da leitura de um sensor, parâmetros de funcionamento do sistema embarcado, ou até mesmo um nome ou conjunto de palavras-chave que tenha função de identificar aquele equipamento específico dentre um conjunto de vários equipamentos do mesmo tipo.

Observado esse problema, a utilização do protocolo BLE (*Bluetooth Low-Energy*), mais especificamente a classe de *beacons* se destaca, pois permite utilizar um único módulo transmissor para realizar um broadcast de informação que pode ser lido por qualquer outro equipamento que seja compatível com o padrão. Considerando que grande parte dos *smartphones* modernos possuem um módulo Bluetooth que inclui a especificação BLE, trata-se de uma grande vantagem, ao permitir que qualquer operador possa fazer a leitura dos dados sem necessidade de um equipamento exclusivo para tal.

Existem soluções comercialmente disponíveis (módulos) para tal implementação, porém estes costumam ser caros, principalmente para produção em baixo volume. Uma alternativa encontrada foi o módulo nRF24L01+, produzido pela NORDIC SEMICONDUCTOR, que apesar de não trazer a implementação do protocolo BLE em hardware, dispõe de meios que permitem tal implementação via software, através da correta manipulação dos parâmetros de transmissão e dos dados a serem transmitidos.

A solução proposta tem como requisitos:

- Implementação de um dispositivo transmissor Bluetooth Low-Energy beacon utilizando o módulo nRF24L01+. Deve ser utilizada a técnica de *bit-banging*, que consiste na implementação de um padrão de comunicação em software, e não em hardware.
- Deve ser utilizado o computador Raspberry Pi para execução do software e interface com o módulo nRF24L01+ ;
- O software deve ser escrito em C++;
- O software deve aproveitar-se do módulo SPI existente em hardware no Raspberry Pi para realizar a comunicação com o módulo nRF24L01+.

### 3. DESENVOLVIMENTO

#### 3.1. IMPLEMENTAÇÃO BLE

O início do desenvolvimento do projeto foi marcado pela pesquisa para encontrar o método mais adequado para possibilitar a implementação do protocolo BLE utilizando um módulo que não o suporta.

O nRF24L01+ possui suporte ao formato de pacote ShockBurst™, que é ilustrado na figura abaixo.

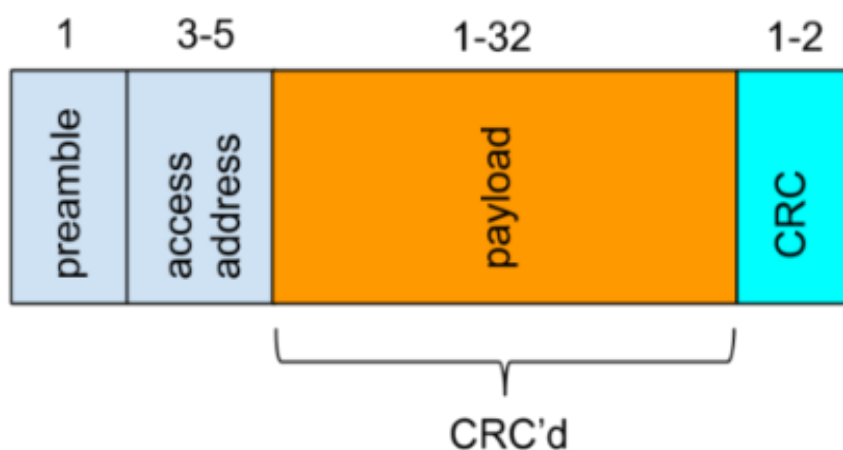


FIGURA 3.1 - Pacote do protocolo ShockBurst™.

Este protocolo possui um formato de pacote similar ao do BLE broadcast, ilustrado na figura abaixo.

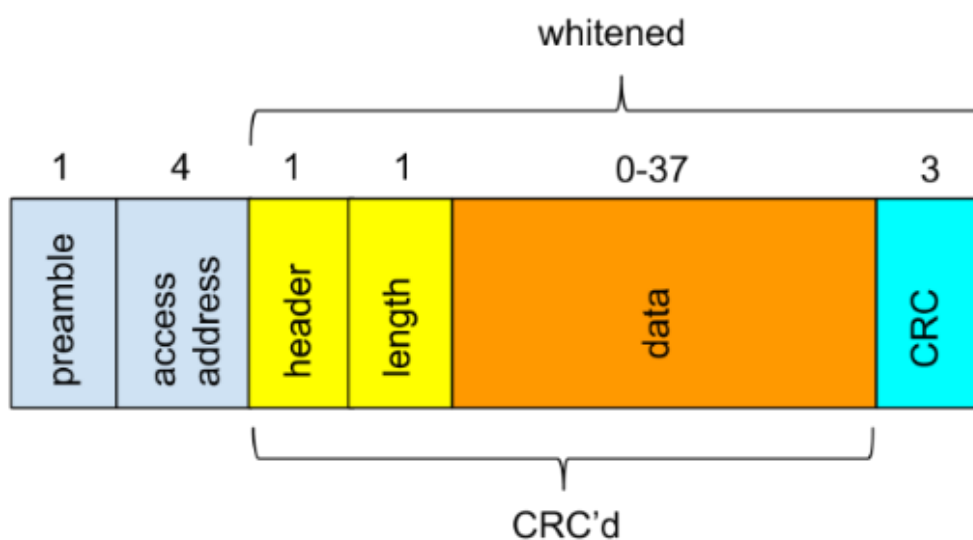


FIGURA 3.2 - Pacote do protocolo BLE Broadcast/Beacon.

Podemos perceber que o formato de pacote do protocolo do BLE Broadcast pode ser emulado utilizando o formato ShockBurst™. O CRC é calculado de maneira diferente,

porém o módulo de cálculo de CRC do nRF24L01+ pode ser desativado, permitindo o cálculo do CRC em software, incluindo-o no final dos dados.

Segundo a documentação oficial das especificações do Bluetooth 4.0 [1], o formato de um pacote é o seguinte.



FIGURA 3.3 - Formato padrão de um pacote Bluetooth 4.0.

Sendo o *Preamble* e *Access Address* os únicos octetos que não precisarão ser emulado via software, *Preamble* é um byte de 1 e 0 intercalado dependendo do primeiro byte do *Access Address*. O *Access Address* será sempre 0x8e89bed6 para canais de propaganda, que é o caso do Bluetooth Broadcast. O *PDU* varia conforme o tipo de dado que está sendo transmitido, para propaganda, o *PDU* tem o seguinte formato.

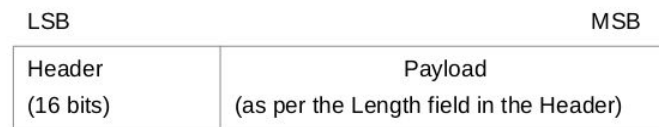


FIGURA 3.4 - Formato padrão de um PDU de Bluetooth Broadcast.

O *Header* tem a seguinte estrutura.

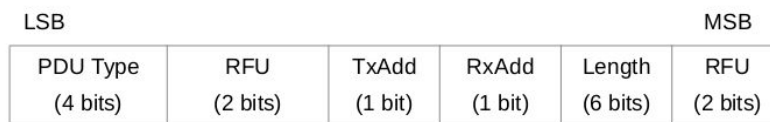


FIGURA 3.5 - Bluetooth Broadcast PDU header.

*PDU Type* identifica o tipo de pacote que está sendo transmitido, para o caso deste projeto será usado o ADV\_NONCONN\_IND (0b0010), que é um transmissor não conectável. *TxAdd* e *RxAdd* sinalizam se os endereços são aleatórios ou não, como o dispositivo não será conectável, esses parâmetros são irrelevantes. *RFU* são bits reservados para uso futuro e finalmente *Length* indica o tamanho do pacote transmitido (sem CRC).

O Payload de uma *PDU* ADV\_NONCONN\_IND deve ser da seguinte forma.

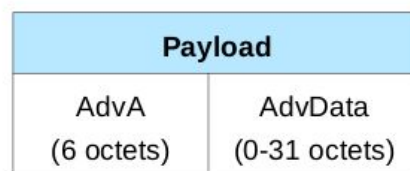


FIGURA 3.6 - Formato de um *PDU* ADV\_NONCONN\_IND.

Ainda dentro de *AdvData*, os dados serão divididos da seguinte forma. 1 octeto indicando o tamanho do próximo pacote, 1 octeto para *AD Type* que especifica o que será transmitido e N octetos que é o pacote em si. Neste caso será enviado 1 byte para setar

algumas FLAGS - resultando em 3 bytes - , 3 bytes sobre o fabricante - 5 bytes no total -, e, por fim, a mensagem em si. Contando todos os campos obrigatórios e que o NRF24L01+ pode transmitir um payload máximo de 32bytes, restam ainda 13 bytes para a mensagem que se deseja transmitir.

### 3.2. MODELAGEM DO SISTEMA

A modelagem do sistema foi feita com diagramas UML.

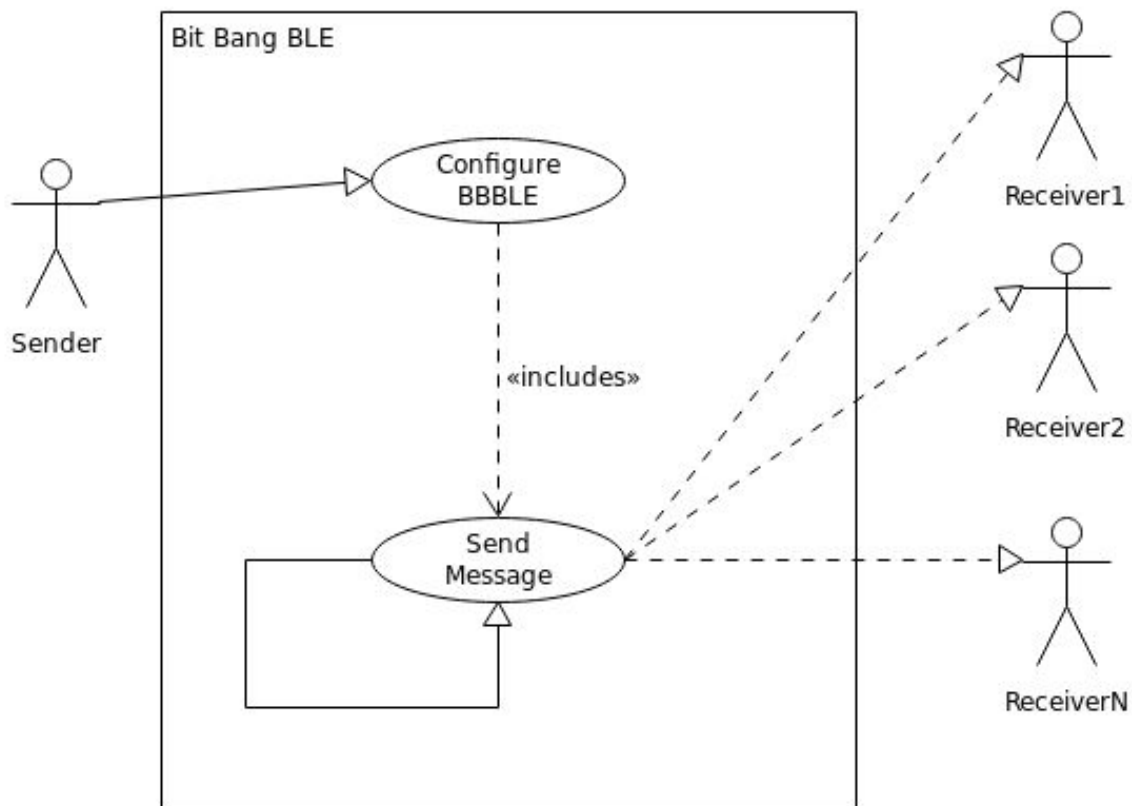


FIGURA 3.7 - Diagrama de Caso de Uso

O objetivo deste diagrama é evidenciar a característica do sistema de broadcast, em que os dados enviados por um único transmissor poderão ser recebidos e lidos por vários dispositivos.

Posteriormente foi desenvolvido um diagrama de classes, cujo objetivo é evidenciar os métodos das classes que comporão o software. Para este projeto, o software será composto por três classes, BBBLE, Radio e SPI.



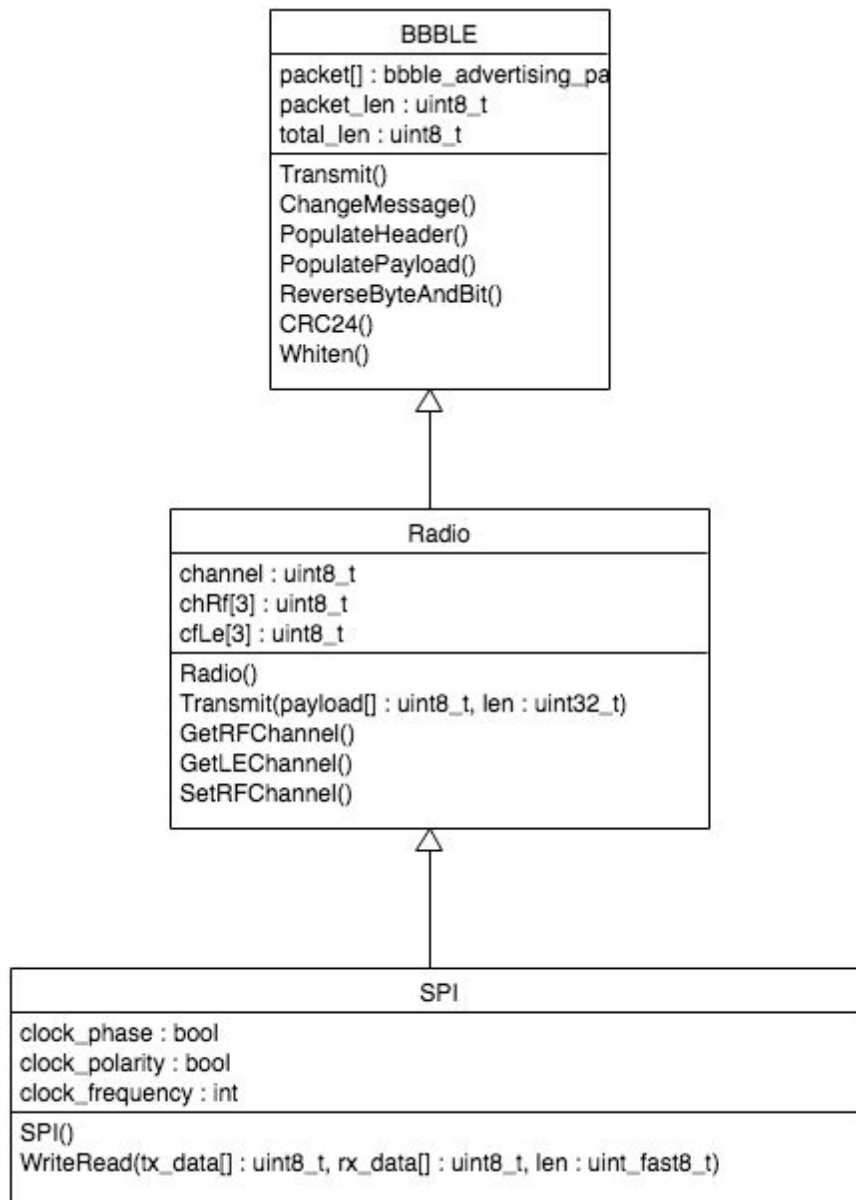


FIGURA 3.8 - Diagrama de Classes

Obs: o diagrama de classes apresentado acima foi atualizado de acordo com mudanças necessárias ao decorrer do projeto.

A classe SPI tem como função a interface com o módulo SPI disponível em hardware no Raspberry Pi através da biblioteca `bcm2835`.

- `SPI()` - construtor da classe, chama as funções apropriadas da biblioteca `bcm2835` com parâmetros especificados para inicialização do módulo SPI disponível em hardware;
- `WriteRead()` - método que possibilita o envio de um ou mais bytes pela interface SPI. Apesar do nome incluir a leitura, para este projeto apenas a função de escrita será utilizada.

A classe Radio tem como função o controle de alguns parâmetros de RF do módulo nRF24L01+, como sua configuração inicial de acordo com o necessário para a correta

implementação do projeto, e também o controle do canal de transmissão em uso, já que o protocolo exige que o transmissor *broadcast* faça *roaming* entre três diferentes canais para a correta implementação do *beacon*.

- Radio() - construtor da classe, inicializa o módulo nRF24L01+ para utilização com o sistema;
- Transmit() - envia um pacote de dados;

A classe BBBLE tem como função implementar a manipulação de dados necessária em alto nível para implementação do protocolo BLE. Seus métodos permitem transformar uma sequência de dados em um pacote pronto para envio para o módulo nRF24L01+.

- BBBLE() - construtor da classe;
- Transmit() - salva o pacote em formato .pcap para validação, e chama as funções de transmissão do pacote;
- ChangeMessage() - recebe uma mensagem a ser transmitida, e chama outras funções para formatar o pacote;
- PopulateHeader() - popula os bytes correspondentes ao *header* do pacote. Isso inclui o tipo de comunicação (ADV\_NONCONN\_IND - *advertising non-connectable*), tamanho total do pacote em bytes, endereço MAC, tipo de dados a ser transferido, entre outros;
- PopulatePayload() - popula os bytes correspondentes à mensagem a ser transmitida, e calcula seu tamanho em bytes;
- ReverseByteAndBit() - executa o procedimento de inversão de bits dentro dos bytes (necessário antes do envio do pacote para o nRF24L01+);
- CRC24() - calcula o CRC de 3 bytes e o adiciona ao final da mensagem;
- Whiten() - executa o procedimento de *data whitening*, requerido pelo protocolo BLE. O nome *whitening* é dado a esse procedimento pois ele transforma os dados de entrada em um vetor de ruído branco, ou seja, que possui igual amplitude em suas diferentes frequências.

Um diagrama de sequência também fez parte da modelagem. A partir deste é possível evidenciar a ordem na qual os métodos são chamados para formação do pacote a ser transmitido.

O primeiro método a ser chamado no diagrama, SetRFChannel(), está relacionado ao *roaming* entre canais que deve ser feito, portanto, antes de cada transmissão de pacote, o canal deve ser alterado. Posteriormente é chamado ChangeMessage(), que chama os métodos correspondentes à formação e manipulação do pacote. PopulatePayload() precisa ser chamado antes de PopulateHeader(), pois só saberemos certos dados do *header*, como o tamanho da mensagem em bytes, após popular a parte do pacote correspondente à mensagem.

O método Whiten() precisa ser chamado depois do método CRC24() pois o procedimento de whitening é aplicado ao pacote todo, incluindo o CRC.

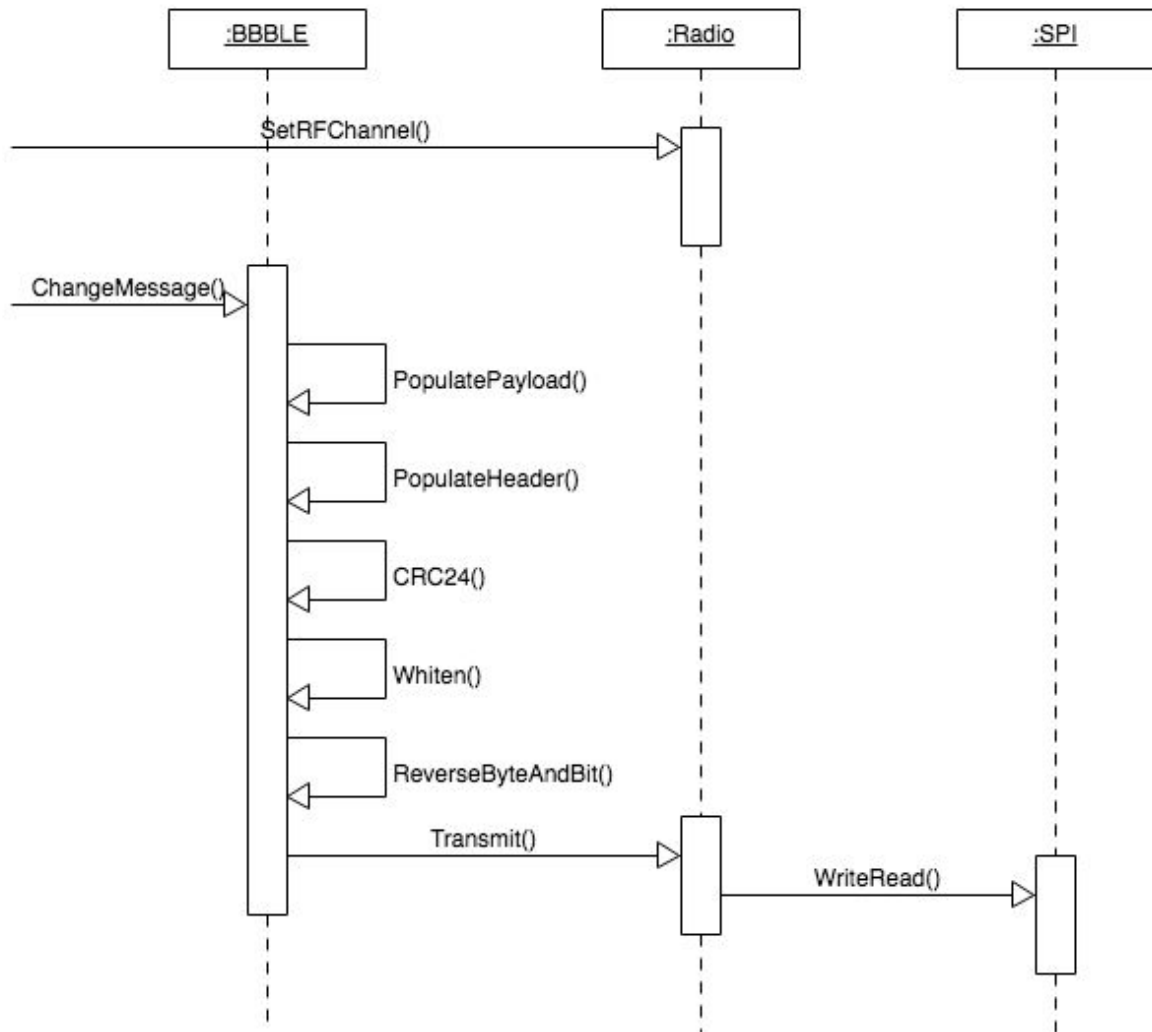


FIGURA 3.9 - Diagrama de sequência

## 4. VERIFICAÇÃO

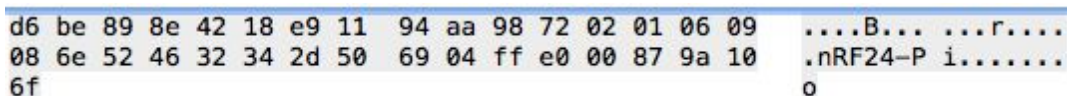
### 4.1. WIRESHARK

Wireshark é uma ferramenta *open-source* que realiza análise de pacotes de rede. O software oferece suporte a diversos tipos de protocolos de rede, entre eles, o protocolo *Bluetooth Low-Energy*.

Apesar de não ser seu formato de arquivo padrão, o Wireshark é capaz de abrir e analisar arquivos .pcap, um dos formatos mais comuns utilizados na captura de pacotes de rede. Este formato é suficientemente simples para que possamos implementar funções no software de nosso projeto para criar um arquivo .pcap simulando a captura de um pacote que estaria sendo transmitido pelo sistema.

Este arquivo foi, posteriormente, analisado pelo Wireshark, que decompõe o pacote em seus diversos campos, apresentando o significado de cada valor de header e flags, evidenciando os campos de mensagem, e verificando se o cálculo do CRC está correto.

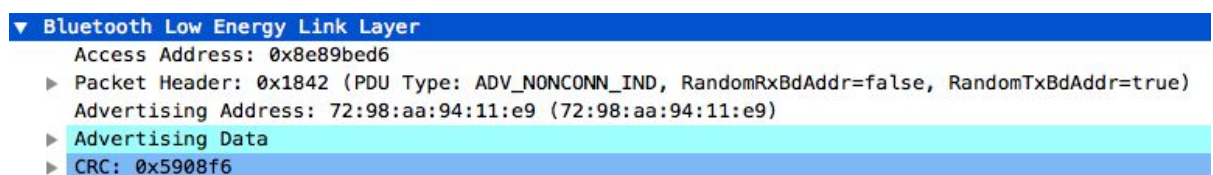
Foi uma ferramenta essencial na validação do sistema, principalmente por ter possibilitado a verificação do CRC, e por ter evidenciado eventuais erros na construção do pacote.



The image shows a Wireshark packet capture interface. The top row of data is highlighted in blue. The hexadecimal data is: d6 be 89 8e 42 18 e9 11 94 aa 98 72 02 01 06 09 08 6e 52 46 32 34 2d 50 69 04 ff e0 00 87 9a 10 6f. The corresponding ASCII data is: ....B... ..r.... .nRF24-P i..... o

FIGURA 4.1 - Um exemplo de pacote exibido na interface do Wireshark

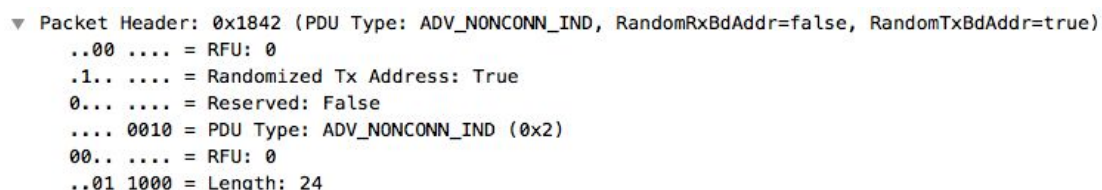
A decomposição do pacote em seus diversos campos realizada pelo Wireshark possibilita uma análise muito intuitiva do pacote.



The image shows the 'Packet Details' pane in Wireshark. The selected packet is of type 'Bluetooth Low Energy Link Layer'. The details are: Access Address: 0x8e89bed6, Packet Header: 0x1842 (PDU Type: ADV\_NONCONN\_IND, RandomRxBdAddr=false, RandomTxBdAddr=true), Advertising Address: 72:98:aa:94:11:e9 (72:98:aa:94:11:e9), Advertising Data, and CRC: 0x5908f6.

FIGURA 4.2 - Pacote decomposto em endereços, header, dados, e CRC

Na Figura 4.3 podem ser verificados os dados que compõem o *header* do pacote. Pode ser percebido que o tipo de comunicação anunciado é *ADV\_NONCONN\_IND*, e que 24 bytes de dados seguirão o *header*.



The image shows the 'Packet Details' pane in Wireshark, specifically the 'Packet Header' section. The details are: Packet Header: 0x1842 (PDU Type: ADV\_NONCONN\_IND, RandomRxBdAddr=false, RandomTxBdAddr=true), ..00 .... = RFU: 0, .1.. .... = Randomized Tx Address: True, 0... .... = Reserved: False, .... 0010 = PDU Type: ADV\_NONCONN\_IND (0x2), 00.. .... = RFU: 0, ..01 1000 = Length: 24.

FIGURA 4.3 - Header do pacote decomposto.

A FIGURA 4.4 ilustra o setor de dados e CRC do pacote. Dentro de Advertising Data, são evidenciados todos os dados que estão sendo transmitidos. No caso deste pacote de exemplo, podemos notar que:

- Existem 2 bytes de flags. Entre as diversas flags, é importante ressaltar que a transmissão está sendo feita em modo LE General Discoverable.
- Está sendo transmitido um campo de *Device Name*, contendo o nome do dispositivo "nRF24-Pi". O campo "*Length: 9*" observado para o *Device Name* corresponde aos 8 bytes do nome (um para cada caracter ASCII) e 1 byte que identifica o tipo de dado a ser transmitido (*Device Name*).
- Está sendo transmitido um campo de *Manufacturer Specific Data*. É nesse campo que podem ser transmitidos dados gerais de desejo. Em nossas simulações, era composto apenas de um byte que era randomizado a cada transmissão, para demonstrar as capacidades do sistema de transmitir dados que se alteram em tempo real.
- O campo de CRC é verificado pelo Wireshark como válido.

```
Advertising Data
▼ Flags
  Length: 2
  Type: Flags (0x01)
  000. .... = Reserved: 0x0
  ...0 .... = Simultaneous LE and BR/EDR to Same Device Capable (Host): false (0x0)
  .... 0... = Simultaneous LE and BR/EDR to Same Device Capable (Controller): false (0x0)
  .... .1.. = BR/EDR Not Supported: true (0x1)
  .... ..1. = LE General Discoverable Mode: true (0x1)
  .... ...0 = LE Limited Discoverable Mode: false (0x0)
▼ Device Name (shortened): nRF24-Pi
  Length: 9
  Type: Device Name (shortened) (0x08)
  Device Name: nRF24-Pi
▼ Manufacturer Specific
  Length: 4
  Type: Manufacturer Specific (0xff)
  Company ID: Google (0x00e0)
  ▶ Data: 87
CRC: 0x5908f6
▼ [Expert Info (Chat/Checksum): Correct CRC]
  [Correct CRC]
  [Severity level: Chat]
  [Group: Checksum]
```

FIGURA 4.4 - Setor de dados e CRC do pacote decompostos

Todas as informações do pacote aberto pelo Wireshark acima referem-se ao pacote gerado pelo nosso software e exportado em formato .pcap. Não trata-se do pacote real transmitido pelo módulo nRF24L01+.

Depois da implementação do sistema, foi possível também capturar o pacote que foi transmitido pelo sistema através do nRF24L01+ utilizando o Wireshark em um computador com módulo Bluetooth BLE. A figura 4.5 ilustra tal captura. Pode ser verificado que, nesse caso, consta também na imagem, a informação do RSSI.

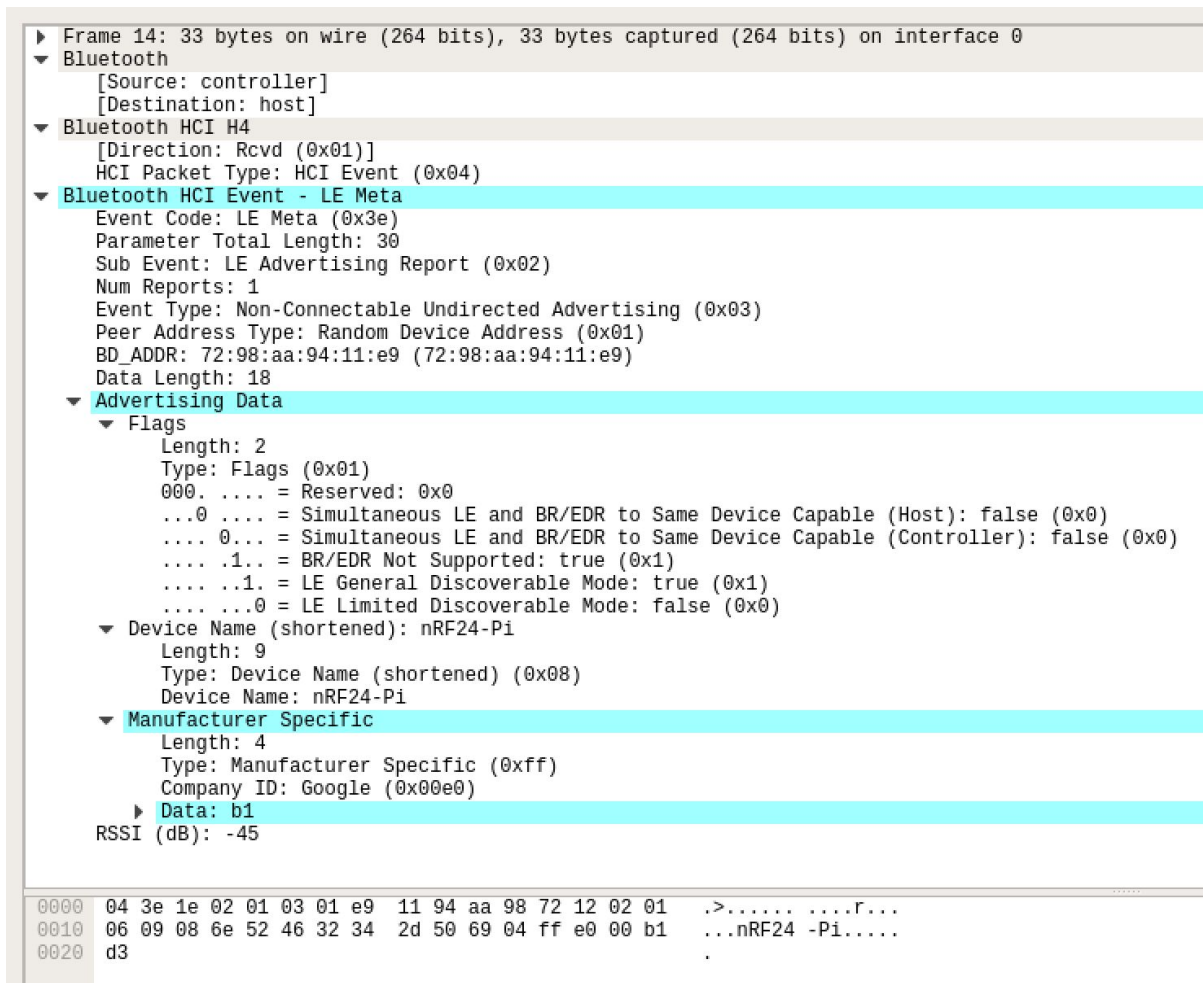





FIGURA 4.5 - Captura de um pacote real transmitido pelo sistema


## 4.2. NRF CONNECT

A NORDIC SEMICONDUCTOR disponibiliza um aplicativo para dispositivos móveis chamado nRF Connect. Este aplicativo é uma poderosa ferramenta para escanear dispositivos *Bluetooth Low-Energy* e verificar os dados sendo transmitidos. Trata-se de uma ferramenta genérica, que pode funcionar com qualquer tipo de *beacon* BLE.

O fato de termos utilizado um aplicativo desenvolvido e disponibilizado pela mesma empresa que fabrica o módulo nRF24L01+ também utilizado no projeto, não passa de uma coincidência. Como o módulo nRF24L01+ não suporta nativamente o protocolo BLE, o aplicativo não foi desenvolvido para receber pacotes enviados por ele, mas sim focado na linha de módulos BLE da NORDIC SEMICONDUCTOR.



**nRF24-Pi**  
72:98:AA:94:11:E9  
NOT BONDED  -45 dBm  N/A

CONNECT 

Type: BLE only  
Flags: GeneralDiscoverable,  
BrEdrNotSupported  
Shortened Local Name: nRF24-Pi  
Manufacturer data (Bluetooth Core 4.1):  
Company: Google <0x00E0> 0x2A

CLONE RAW MORE

FIGURA 4.5 - Beacon detectado no aplicativo nRF Connect

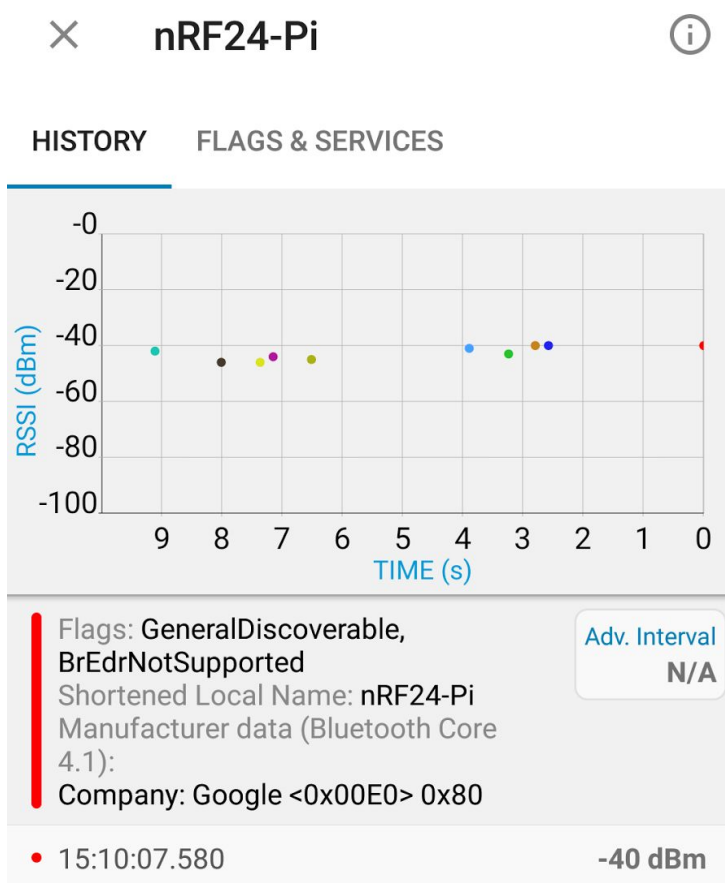


FIGURA 4.6 - Beacon detectado no aplicativo nRF Connect

## **5. CONCLUSÃO**

Neste projeto foi apresentado um passo a passo de como utilizar um rádio comum de 2.4GHz para realizar comunicação Bluetooth Broadcast. Seguindo todos os procedimentos de definição e especificações de projetos mostrados ao longo de todo o semestre, o projeto pôde ser executado de forma contínua e sem surpresas.

Usar as especificações oficiais de Bluetooth 4.0 foram essenciais, nelas estão todas as informações necessárias para o desenvolvimento de qualquer produto que contenha essa tecnologia.

O resultado final funcionou como esperado, porém não seria tão indicado para fins comerciais pois utiliza mais processamento que o necessário (neste caso processamento é sinônimo de energia), a quantidade de bytes que podem ser transmitidos como mensagens é bem reduzido e por questões de segurança, já que os dados estarão circulando de forma aberta.



## **6. REFERÊNCIAS**

[1] <https://www.bluetooth.com/specifications/adopted-specifications>, acessado em 23 de Novembro de 2016

[2] <http://dmitry.gr/index.php?r=05.Projects&proj=11.%20Bluetooth%20LE%20fakery>, acessado em 23 de Novembro de 2016

[3] <http://doc.lijun.li/misc-nrf24-ble.html>, acessado em 23 de Novembro de 2016