



**TÉCNICO**  
LISBOA

Mestrado em Engenharia Electrónica

# **Bluetooth Low-Energy**

—

**Redes BLE e aplicações em  
Redes de Sensores**

Diogo Guerra 68247

Docente: Professor Rui Rocha

Junho 2015

## Índice

Índice de Figuras .....	3
Índice de Tabelas .....	3
Introdução.....	4
A tecnologia .....	5
Bluetooth.....	5
Bluetooth Low-Energy .....	5
Organização .....	5
Comunicação.....	8
Módulo BLE112.....	9
Protótipo da Placa de Expansão .....	10
Update de <i>Firmware</i> .....	11
Funcionamento do Software no MotelST .....	13
Resultados.....	14
Demosntração com o MotelST como Servidor GATT.....	15
Demosntração com o MotelST como Cliente GATT.....	17
Conclusão.....	18
Sobre as Redes de Sensores.....	19

## Índice de Figuras

Figura 1 - Máquina de estados da comunicação um módulo BLE.....	6
Figura 2 - Difusão de um pacote pelo periférico e interacção entre o Sink.....	6
Figura 3 - Ligação entre dispositivos numa rede BLE. ....	7
Figura 4 - Organização da base de dados do servidor BLE de acordo com o Protocolo de Atributos. ....	8
Figura 5 - Modos de interacção com a API da Bluegiga presente no módulo BLE112..	9
Figura 6 - Placa de expansão BLE112v1.....	10
Figura 7 - Exemplo de compilação bem sucedida efectuada no terminal. ....	12
Figura 8 - Ferramenta de Update de Software da Bluegiga. ....	12
Figura 9 - Mensagem apresentada na janela de debug no início de sessão. ....	14
Figura 10 - Janela da interface visual com a descrição dos respectivos campos. ....	15
Figura 11 - Resultado final da leitura (a pedido do cliente) da característica IO Port Status do servidor GATT. ....	16
Figura 12 - Resultado da interacção realizada pelo cliente no Servidor GATT MotelST. ....	16
Figura 13 - Janela de debug com a interacção entre o MotelST como Cliente GATT e o Dongle BLED112. ....	17
Figura 14 - Configurações principais da interface UART do módulo para comunicação com o MotelST. ....	18

## Índice de Tabelas

Tabela 1 - Portos e configurações disponíveis para interacção com o módulo BLE....	10
--	----

## Introdução

Redes de sensores (RS) são sistemas constituídos por nós distribuídos espacialmente que medem de forma automática grandezas físicas do meio ambiente. Estas grandezas podem ser temperatura, luz, pressão, humidade, aceleração, diferenças magnéticas e eléctricas. Esta rede pode ser constituída por poucos nós ou muitos nós, dependendo da aplicação e do meio onde se inserem.

RS podem ser encontradas em diferentes ambientes como na domótica, devido ao crescente aparecimento dos electrodomésticos inteligentes, em ambientes industriais para controlo e monitorização dos processos de fabrico e na saúde para controlo e alerta de sinais vitais ou medicação do paciente.

Às redes que recolhem e transmitem dados, através de diferentes dispositivos, num espaço relativamente pequeno são chamadas de Rede de Área Pessoal do inglês "PAN - Personal Area Network". Estas redes têm tipicamente capacidade de débito de dados reduzida, relativamente a outras redes como o 802.11 (ou wireless), e alcance reduzido, limitado a alguns metros.

O principal objectivo do trabalho é o de criar uma pequena rede construída por nós que comuniquem entre si através de BLE - Bluetooth Low Energy. Nesta rede existe um nó responsável por efectuar a sensorização dos mais variados parâmetros ambientais e um nó que assuma ser o coordenador ou SINK da rede.

## A tecnologia

### Bluetooth

A tecnologia Bluetooth foi concebida como um substituto para a comunicação por cabos como os auscultadores, teclado e rato de computadores, entre outros. Hoje em dia, a tecnologia expandiu-se para conectar diferentes dispositivos que vão desde áreas como a indústria, multimédia, saúde. Esta última onde estes dispositivos são usados como sensores de monitorização de actividade e bem estar de pacientes através de medições do nível de glucose, tensão arterial, e outros.

Tendo sido adaptada por muitos dispositivos móveis, incluindo computadores, telemóveis e notebooks foi criada a necessidade de conceber um standard com a mesma intuito que o Bluetooth, mas que fosse mais economizador de energia, um dos principais problemas de mobilidade nos gadgets de hoje em dia.

### Bluetooth Low-Energy

O Bluetooth Low-Energy (BLE) foi concebido com o intuito de melhorar o desempenho de consumo energético dos dispositivos móveis. Introduzido em 2010 o standard foi adoptado com uma velocidade impressionante (relativamente a outras tecnologias).

Em qualquer protocolo ou tecnologia, existem trocas que se realizam entre velocidade de transferência, alcance, tempo de vida e outros. O standard BLE foi desenvolvido com o compromisso de ter consumo reduzido e por isso, a taxa de transferência do protocolo é muito reduzida.

### Organização

A interligação física entre dispositivos BLE está dividida entre Mestres (Sink) e Escravos (Periféricos). A qualquer altura um dispositivo BLE encontra-se em um determinado modo bem definido que pode ser visualizado na máquina de estados presente na Figura 1.

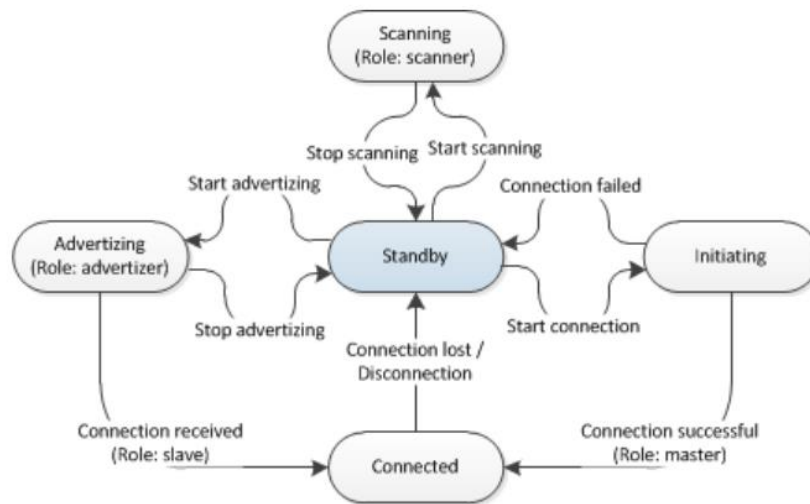


Figura 1 - Máquina de estados da comunicação um módulo BLE.

Este processo é controlado pelo Perfil de Acesso Genérico, do Inglês GAP (*Generic Access Profile*) que é responsável em estabelecer ligações e difusão de pacotes de informação.

Nesta tecnologia,

**Mestres** – São os nós que estão em constante varrimento à procura de pacotes de informação proveniente de nós Escravos. Nesta tecnologia, este nó é aquele que consome mais energia e por esse motivo é considerado que não se encontra restrito em questões energéticas.

**Escravos** – São os nós que têm mais problemas com recursos, nomeadamente acesso a energia. Os pacotes de informação difundidos pelos periféricos podem ter períodos mínimos de 20 ms. Períodos de difusão maiores permitem um maior prolongamento do tempo de vida do nó, mas diminuem a responsividade.

Na Figura 2 encontra-se uma ilustração deste tipo de comunicação.

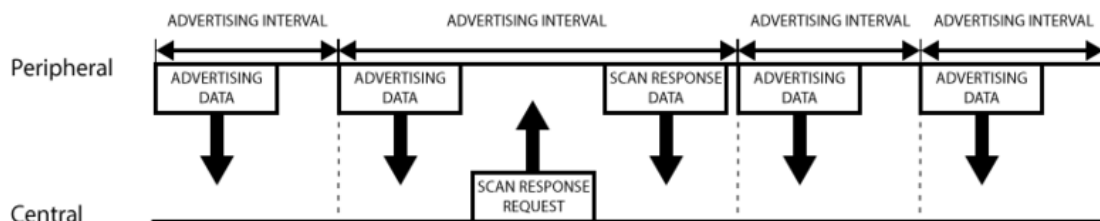


Figura 2 - Difusão de um pacote pelo periférico e interação entre o Sink.

Apesar de poder ser transmitida informação em pacotes de difusão é preferencial, quando se pretende transmitir dados, a utilização de uma ligação Mestre-Escravo para este tipo de transacções. O facto de este tipo de comunicação ser realizada em canais de rádio com frequências diferentes e permitir que a frequência de transmissão seja acordada entre Mestre e Escravo reduzem o consumo de energia nos dois nós.

Neste tipo de comunicação, o intervalo de tempo mínimo entre pacotes (por ligação) é de 7,5 ms e o intervalo máximo é de 4 s. Assumindo que não existe perda e retransmissão de pacotes, a velocidade de transmissão máxima é de 2933 B/s e a mínima é de 5,5 B/s. Estes valores correspondem à velocidade máxima total da ligação (emissão+recepção). A velocidade máxima de leitura de dados de um servidor é então, assumindo situações ideais com transmissão de pacotes de ACK, metade destes valores, ou seja, 1467 B/s para o intervalo de tempo mais pequeno.

No entanto, a velocidade de transferência total de um nó Mestre pode ser maior, já que este pode realizar comunicações com múltiplos nós simultaneamente. A interligação entre diferentes dispositivos numa rede BLE é realizada através de uma topologia em estrela, já que neste protocolo os dispositivos Escravos apenas podem estar ligados a um dispositivo Mestre a um dado momento. Na Figura 3 pode-se visualizar o tipo de ligações possíveis de existir em uma rede BLE.

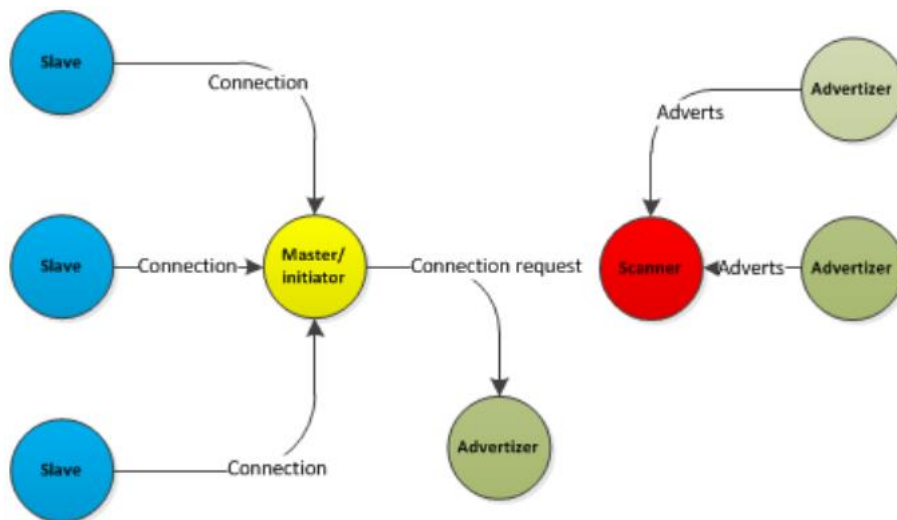


Figura 3 - Ligação entre dispositivos numa rede BLE.

### Comunicação

Na tecnologia BLE existe uma diferença lógica entre a camada física e o acesso aos dados. Ao contrário do que é natural, na tecnologia BLE os nós que são considerados como Escravos, são os Servidores dos dados. Da mesma forma, os nós Mestre são os Clientes. Isto permite que os Servidores (ou escravos) apenas transmitam informação quando esta se encontra disponível. Este é o principal factor que torna o BLE um protocolo de baixo consumo.

No BLE os dados são guardados no dispositivo e encontram-se organizados em um conceito de Serviços. Um módulo BLE pode ser constituído por vários Serviços em que cada um tem várias características.

Estes dados são geridos através de um Protocolo de Atributos, do inglês ATT – *Attribute Protocol*. A organização dos dados dentro do módulo pode ser encontrada na Figura 4 onde a característica Perfil corresponde a uma configuração específica do módulo (a uma dada altura o módulo BLE apenas pode ter um único Perfil).

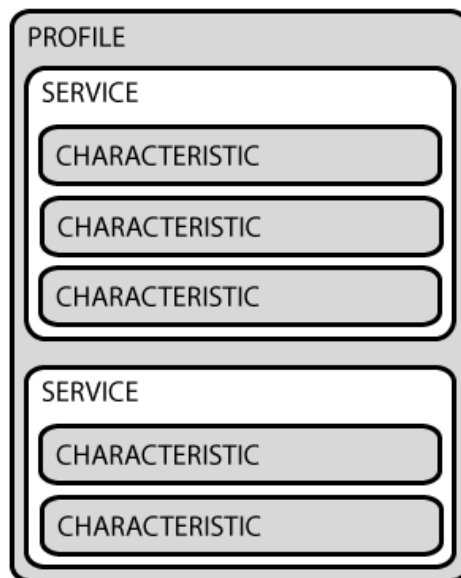


Figura 4 - Organização da base de dados do servidor BLE de acordo com o Protocolo de Atributos.



## Módulo BLE112

Sendo o módulo BLE112 um Soc com diferentes portas, como SPI, UART e USB, é possível realizar um módulo *stand-alone* que contenha um dado sensor, (p.ex temperatura) e através de um programa na linguagem proprietária da Bluegiga (BGScript) interagir directamente com este. No entanto, estando planeada a interligação deste com a plataforma MotelST foi optada utilizar uma abordagem por comunicação externa.

A comunicação com o módulo Bluetooth pode ser realizada com uma API fornecida pela Bluegiga. Esta API actua como um analisador de input da porta série e interage directamente com a correspondente API existente no módulo. Na Figura 5 pode-se visualizar um diagrama com a representação dos diferentes modos de interacção com a API presente no módulo.

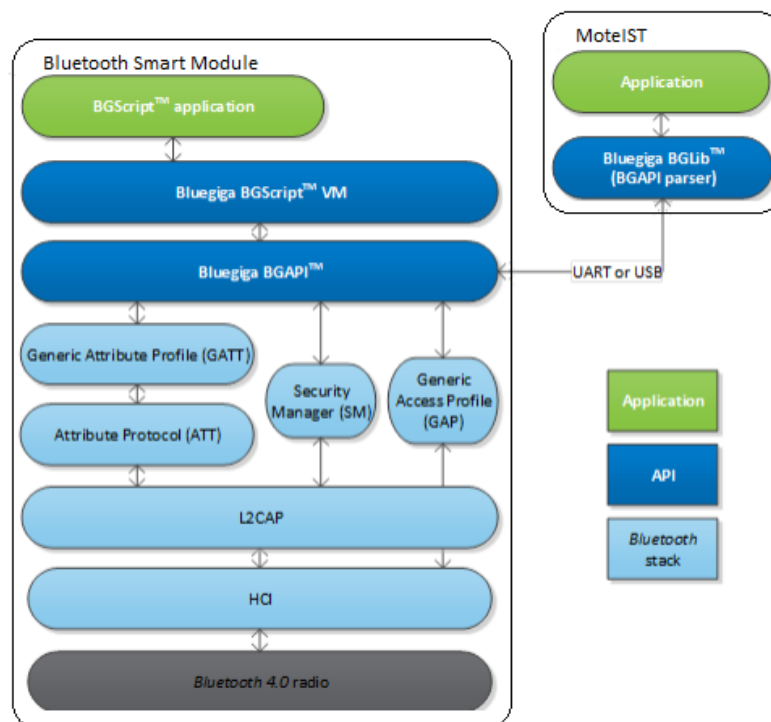


Figura 5 - Modos de interacção com a API da Bluegiga presente no módulo BLE112.

### Protótipo da Placa de Expansão

Primeiramente procede-se a uma prototipagem rápida do módulo BLE para que este possa ser integrado com o MotelST. Desta forma é realizada uma placa de expansão simples onde apenas são extraídas a alimentação do módulo (VCC e GND) o pino de Reset e a porta de comunicação série. Na Figura 6 pode visualizar-se a Placa desenvolvida.



Figura 6 - Placa de expansão BLE112v1.

Para a interacção com o módulo foi escolhido utilizar o periférico UART. Este permite uma comunicação assíncrona com o módulo, simplificando a integração e controlo dos drivers dos periféricos com a nossa aplicação, ao contrário do que acontece com recurso a SPI.

Tabela 1 - Portos e configurações disponíveis para interacção com o módulo BLE.

PERIPHERAL / FUNCTION		P0							
		7	6	5	4	3	2	1	0
ADC		A7	A6	A5	A4	A3	A2	A1	A0
USART 0 SPI	Alt. 1			C	SS	MO	MI		
	Alt.2								
USART 0 UART	Alt. 1			RT	CT	TX	RX		
	Alt.2								
USART 1 SPI	Alt. 1			MI	MO	C	SS		
	Alt.2								
USART 1 UART	Alt. 1			RX	TX	RT	CT		
	Alt.2								

OS módulos BLE vêm pré-carregados com o firmware exemplo 'UARTDemo' que utilizam a UART(1) controlada pelos pinos de CTS e RTS e P0.4 e P0.5 (TX e RX), respectivamente

Como os pinos e o tipo de comunicação série utilizados dependem do firmware que vem pré-programado com o módulo bluetooth, existem configurações que devem ser tidas em conta. Os pinos escolhidos disponíveis na placa de expansão do módulo correspondem a P0.4 e P0.5 para RX e TX do módulo BLE, respectivamente. A comunicação série por UART pode ser utilizada de dois modos:

- **Modo Controlado** - A comunicação entre o módulo BLE e o MCU é regulada através de sinal CTS - *Clear To Send* e RTS - *Ready To Send*.
- **Modo por Pacote** - À priori do envio de um pacote para o rádio, o primeiro byte enviado corresponde sempre ao comprimento total do pacote a receber (Cabeçalho + Corpo).

Para não utilizar os pinos de CTS, e RTS e para proceder a outras modificações aos serviços presentes no módulo é necessário modificar o *firmware* presente no módulo.

### Update de *Firmware*

Para proceder à comunicação externa foi necessário fazer uma actualização do *firmware* do módulo. Depois de proceder às alterações necessárias nos ficheiros correspondentes, nomeadamente o ficheiro gatt.xml e hardware.xml, é necessário recompilar o código para posterior carregamento no módulo com a placa de desenvolvimento BLE112. Para isto faz-se:

- 1 - Com um terminal no directorio da instalação 'Bluegiga' fazer,

```
cd Bluegiga\ble-1.3.2-122\example\dkble112\uartdemo
```

- 2 - Depois disso, para compilar as alterações insere-se o comando,

```
..\..\bin\bgbuild.exe project.bgproj
```

- 3 - O novo ficheiro para carregar encontra-se agora em

```
Bluegiga\ble-1.3.2-122\example\dkble112\uartdemo\out-ble112.hex
```

Na Figura 7 é apresentada a janela do terminal com a compilação bem sucedida.

```
Qt: Untested Windows version 6.2 detected!
baudm:216 baudr:11 rate:115234
UART channel:1
baudrate :115200
actual :115234
error% :0.0295139
alternate f:1
Qt: Untested Windows version 6.2 detected!

RAM Memory
-----
Core RAM end          0 0x00afa 2810
Top of RAM            0 0x01f00 7936
RAM left for data      = 0x01406 5126
Attribute RAM          - 0x00005 5
Connections            1 - 0x00196 406
Script Variables       - 0x00000 0
RAM for packet buffers 120 - 0x01248 4680

Flash Memory
-----
Core flash reserved    0 0x18000 98304
Top of flash           0 0x1f800 129024
Flash left for data     = 0x07800 30720
Common configuration   - 0x00070 112
16 bit UUIDs           - 0x00014 20
128 bit UUIDs          - 0x00030 48
Attribute database     - 0x0006c 108
Constant attributes data - 0x00088 136
USB descriptor         - 0x000c6 198
Flash for PS Store     14 - 0x07000 28672
Qt: Untested Windows version 6.2 detected!
```

Figura 7 - Exemplo de compilação bem sucedida efectuada no terminal.

Podemos proceder ao carregamento da nova aplicação para o módulo utilizando a Ferramenta de Update de Software da Bluegiga carregando primeiramente em 'Info' para obter a chave de licença e depois de inserida no sítio correspondente premir 'Update' e esperar carregamento bem sucedido (janela aparece verde com 'Update Completed'). Na Figura 8 apresenta-se a janela de update de *firmware* da bluegiga.

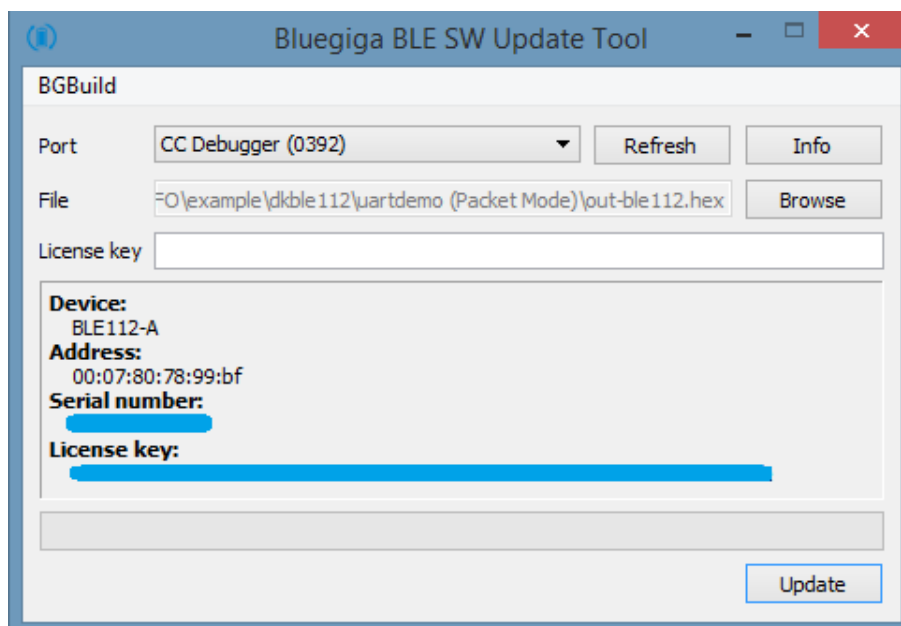


Figura 8 - Ferramenta de Update de Software da Bluegiga.

### Funcionamento do Software no MotelST

A API da Bluegiga está feita de maneira a funcionar com uma programação do tipo “Programação de Retorno de Chamada”. Numa primeira abordagem foi decidido proceder à programação deste sem recurso a micro-kernel. No entanto, problemas com a interface do periférico levantaram-se e para facilitar a interligação com o tipo de programação da API providenciada é mais simples a utilização de um Sistema Operativo. Desta forma, recorreu-se ao sistema operativo FreeRTOS.

O sistema corre dois processos:

- Um processo é responsável por tratar dos dados provenientes do módulo BLE e proceder de acordo com o programado (apresenta-los na janela de *debug*).
- O outro processo trata do *input* do utilizador através da janela de *debug* e procede à construção dos pacotes para posterior envio para o módulo BLE.

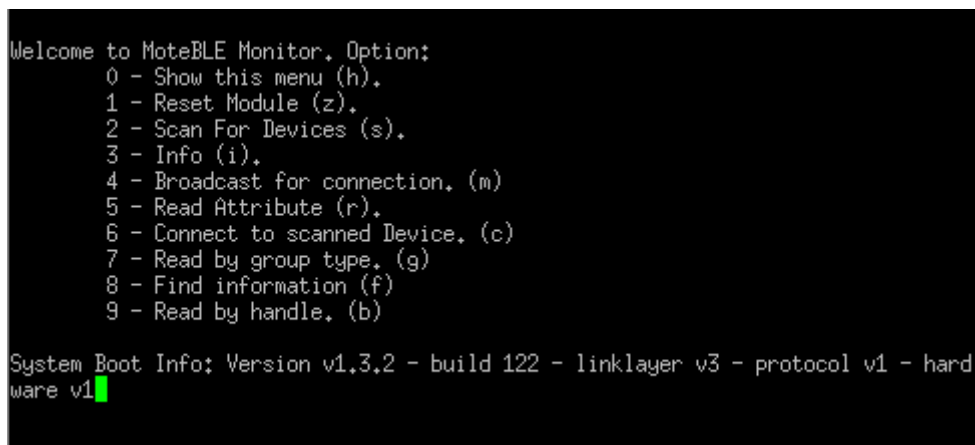
No Capítulo seguinte são mostrados e explicados alguns testes realizados com este módulo.

## Resultados

Os resultados apresentados em seguida mostram o resultado de uma comunicação através de uma aplicação desenvolvida entre o MotelST ligado à placa de desenvolvimento “BLE112 Development Board” e o dongle USB “BLED112”.

É importante notar que, como a API utilizada com o MotelST comunica directamente com a API do módulo (Figura 5) cada dispositivo BLE pode funcionar como Mestre ou Escravo a uma dada altura, de acordo com o especificado em capítulos anteriores através da Figura 1. Neste caso, a única diferença entre dispositivos vai ser a organização do servidor GATT interno ao módulo.

Aquando a inicialização do MotelST é realizado um reset ao módulo (manualmente) por forma a ter conhecimento do estado do módulo. Este reset pode ser realizado através de um pino especificamente para esse propósito (também disponível na placa de expansão). A mensagem apresentada na janela de debug é mostrada na Figura 9.



```
Welcome to MoteBLE Monitor. Option:
0 - Show this menu (h).
1 - Reset Module (z).
2 - Scan For Devices (s).
3 - Info (i).
4 - Broadcast for connection. (m)
5 - Read Attribute (r).
6 - Connect to scanned Device. (c)
7 - Read by group type. (g)
8 - Find information (f)
9 - Read by handle. (b)

System Boot Info: Version v1.3.2 - build 122 - linklayer v3 - protocol v1 - hardware v1
```

Figura 9 - Mensagem apresentada na janela de debug no início de sessão.

Ao mesmo tempo configura-se a ferramenta de interacção visual da Bluegiga “Bluegiga BLE Graphical User Interface Tool”. Num momento mais oportuno seria utilizada a plataforma de desenvolvimento com a placa de expansão desenvolvida, mas como é preciso fazer o update de firmware nesta última, foi decidido exemplificar os dois tipos de comunicação com recurso à ferramenta visual fornecida. A interface pode ser visualizada na Figura 10 onde esta já se encontra ligada ao Dongle USB. Esta interface foi importante no desenvolvimento do software presente no MotelST já que apresenta os comandos enviados pelo módulo na Janela de Dados em Bruto, e com recurso ao

User Manual foi possível a simplificação dos passos a seguir por forma a estabelecer uma ligação entre dois dispositivos BLE.

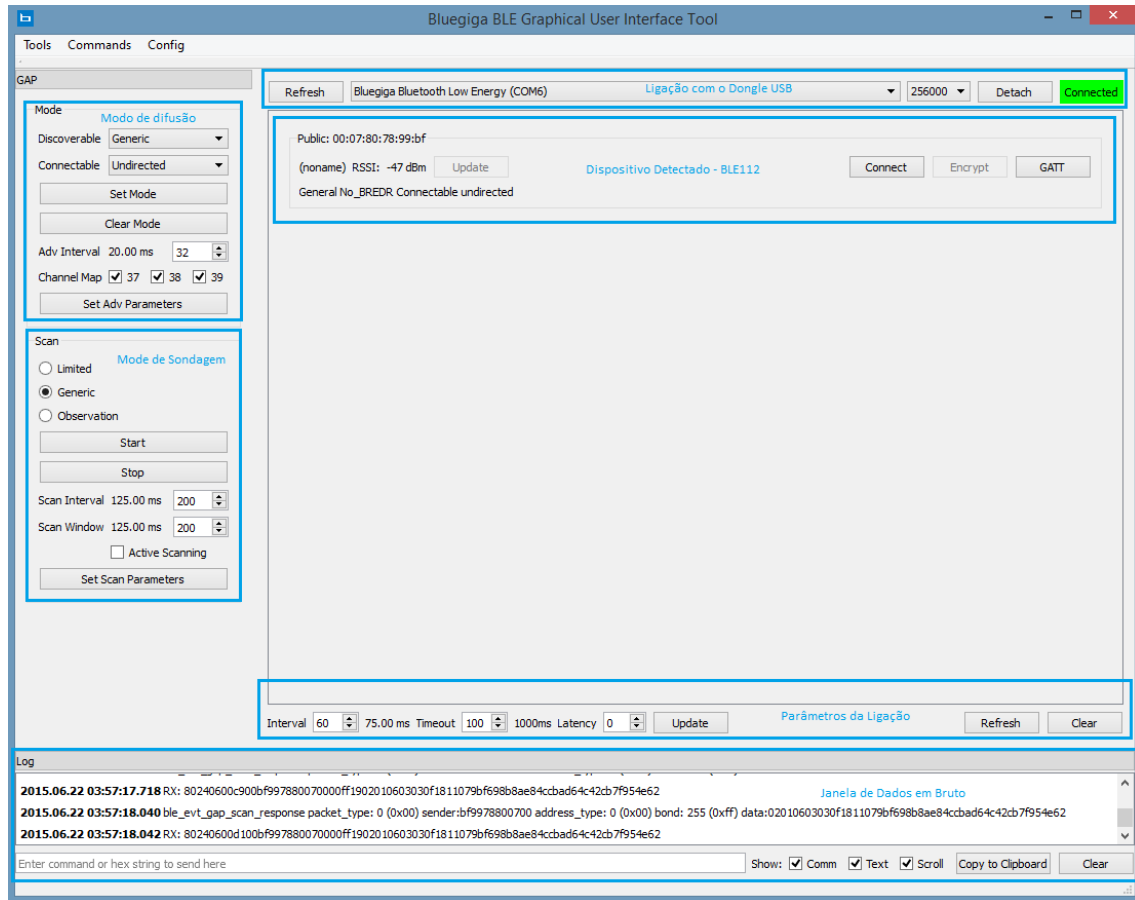


Figura 10 - Janela da interface visual com a descrição dos respectivos campos.

### Demonstração com o MotelST como Servidor GATT.

Partindo do ponto de reset (Figura 9) e à priori da Figura 10 a opção 4 (Broadcast for Connection) foi seleccionada. A esta altura o módulo é detectado e connectável. Carregando em *connect* uma ligação é estabelecida entre o BLE112 e o MotelST. A esta altura é possível ler os Serviços do MotelST carregando em *Service Discovery*.

Um dos Serviços personalizados para esta demonstração aparece com o UUID de 128-bit. Seleccionando este e carregando em *Descriptors Discovery* são apresentados as diferentes Características do Serviço seleccionado. Seleccionando os dois últimos campos (a vermelho) e carregando *Read* obtem-se a mensagem do MotelST visível na Figura 11. O resultado desta interacção com o MotelST como Servidor GATT é visualizado na Figura 12.

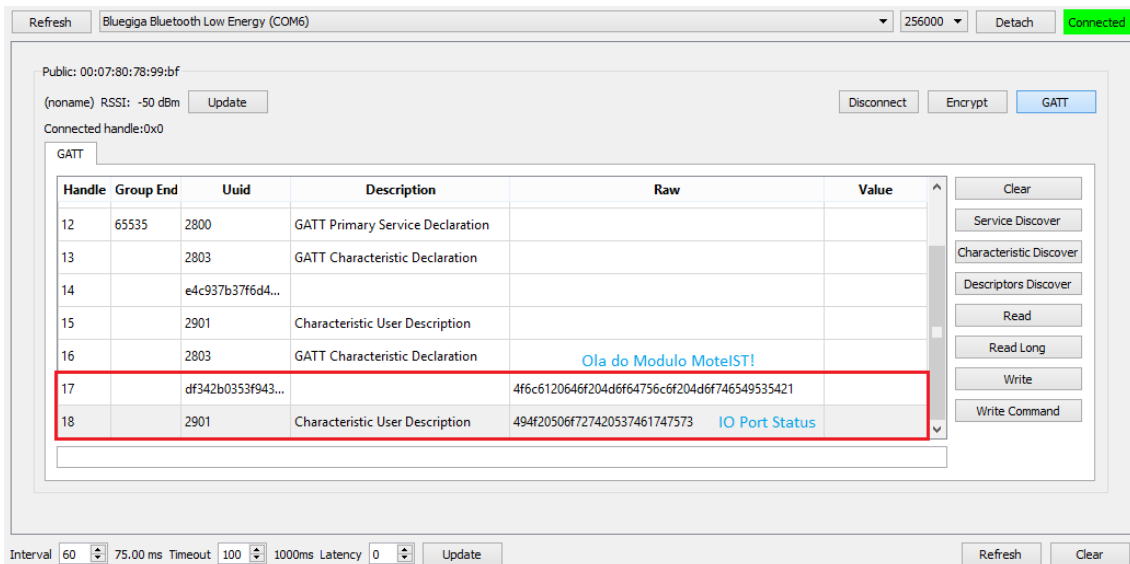


Figura 11 - Resultado final da leitura (a pedido do cliente) da característica IO Port Status do servidor GATT.

```

Welcome to MotelBLE Monitor. Option:
0 - Show this menu (h).
1 - Reset Module (z).
2 - Scan For Devices (s).
3 - Info (i).
4 - Broadcast for connection. (m)
5 - Read Attribute (r).
6 - Connect to scanned Device. (c)
7 - Read by group type. (g)
8 - Find information (f)
9 - Read by handle. (b)

System Boot Info: Version v1.3.2 - build 122 - linklayer v3 - protocol v1 - hardware v1
PASS
Connected to device Address:BE:63:02:80:07:00 Interval 60 - TimeOut 100Sent String

```

Figura 12 - Resultado da interação realizada pelo cliente no Servidor GATT MotelST.

A característica tem o nome “IO Port Status” porque o nome desta não foi modificado no *firmware* carregado para o módulo. Nesta situação, quando o servidor procede a uma leitura ao MotelST o módulo BLE gera um evento e a mensagem é obtida por pedido directo ao MotelST, ao contrário de ser lida directamente da memória local do servidor. Isto deve-se ao parâmetro **type="user"** configurado no ficheiro gatt.xml.



## Demosntração com o MotelST como Cliente GATT.

Partindo do ponto de reset (Figura 9) a interface visual é agora configurada para difundir pacotes por forma a ser possível a ligação como Servidor GATT. Carregando em *Set Mode* na janela de Difusão (Figura 10) o BLED112 começa a difundir pacotes.

Configura-se agora manualmente o MotelST carregando em *Scan For Devices (s)*, seguidamente são apresentados os scans obtidos. Adquirido o dispositivo, ligamo-nos a este *Connect to Scanned Device (c)*. Com uma ligação estabelecida podemos agora fazer operações dedicadas sobre o Servidor GATT do BLED112, nomeadamente obter os Serviços do módulo (*Read by group type (g)*) e posteriormente achar as Características desse Serviço (*Find Information (f)*). O UUID=2A00 é standard com o nome do dispositivo (UUID aparece trocado). Seleccionando a opção *Read by Handle (b)* e introduzindo em seguida o identificador do *Handle 03* conseguimos ler o nome do dispositivo “*Bluegiga BLED112*” visível na Figura 13.

```

Welcome to MoteBLE Monitor. Option:
  0 - Show this menu (h).
  1 - Reset Module (z).
  2 - Scan For Devices (s).
  3 - Info (i).
  4 - Broadcast for connection. (m)
  5 - Read Attribute (r).
  6 - Connect to scanned Device. (c)
  7 - Read by group type. (g)
  8 - Find information (f)
  9 - Read by handle. (b)

System Boot Info: Version v1.3.2 - build 122 - linklayer v3 - protocol v1 - hardware v1
PASS
Device Scanned Address:BE:63:02:80:07:00
Device Scanned Address:BE:63:02:80:07:00
Device Scanned Address:BE:63:02:80:07:00
Device Scanned Address:BE:63:02:80:07:00
Device Scanned Address:BE:63:02:80:07:00
Device Scanned Address:BE:63:02:80:07:00
Connection Handle Acquired!
Connected to device Address:BE:63:02:80:07:00 Interval 40 - TimeOut 100
PASS
NEW GROUP
  Start - 0001 End - FFFF UUID #0018#---Received All Group Information
PASS
Information
  Handle-0001 UUID#0028#---
  Handle-0002 UUID#0328#---
  Handle-0003 UUID#002A#---
  Handle-0004 UUID#0328#---
  Handle-0005 UUID#012A#---Received All Group Information
PASS
  AttHandle-0003 Type-48 Bluegiga BLED112#---

```

Figura 13 - Janela de debug com a interacção entre o MotelST como Cliente GATT e o Dongle BLED112.

## Conclusão

No desenvolvimento deste trabalho foram encontrados alguns obstáculos. Um deles foi a necessidade de update do *firmware* para conseguir usar a UART sem os pinos CTS e RTS. Além disso, outras dificuldades encontradas foram relativas à falta de indicação que a UART necessitava de sinal num pino P0.0 para acordar o módulo por forma a este poder receber dados por este porto. Este problema foi contornado com a utilização do comando *sleep* como se demonstra na Figura 14.

```
<usart channel="1" alternate="1" baud="115200" endpoint="api" flow="false" mode="packet" />
<sleep enable="false" />
<wakeup_pin enable="false" port="0" pin="0" />
```

**Figura 14 - Configurações principais da interface UART do módulo para comunicação com o MotelST.**

De qualquer forma, este pino de *enable* encontra-se presente na placa de expansão e pode ser utilizado o modo em *wake-up* bastando para isso alterar *sleep* para *true* e no campo *wakeup\_pin* substituir para *true*.

O update do *firmware* permite também alterar a organização da base de dados do Servidor GATT e eliminar ou adicionar Serviços e/ou Características.

Outras das dificuldades que se levantaram foi a aprendizagem de como funcionava e estava organizado o acesso e escrita aos dados, tanto remotamente como localmente. Neste sentido o livro “*Getting Started With Bluetooth Low-Energy*” foi muito útil.

Findo este processo de aprendizagem, foi agora necessário aprender a utilizar a API fornecida pela Bluegiga. Uma das principais dificuldades foi o conhecimento dos comandos necessários ao estabelecimento bem sucedido de uma ligação entre dois módulos BLE.

Para este obstáculo o uso do guia de utilizador “Bluetooth Smart Software v1.3 API reference” juntamente com a plataforma de interface gráfica com o Dongle USB revelaram-se imprescindíveis.

### Sobre as Redes de Sensores

Uma das primeiras tentativas à utilização do BLE levou a supor que existia um método de comunicação semelhante ao que era utilizado no Bluetooth normal, o chamado Perfil RFCOMM. Tal não é verdade devido à maneira única como o protocolo BLE funciona.

Foi no entanto achado uma adaptação deste protocolo que realiza uma ligação directa com os dois portos UART, do mesmo estilo que acontecia com o Bluetooth normal. No entanto existem algumas limitações:

- Velocidade de transferência máxima é de aproximadamente 1 kB/s;
- Utilização de controlo de fluxo é obrigatória para transmissão confiável;
- Não permite compatibilidade com outros módulos BLE que tenham configurado um Perfil diferente;

O método mais natural de utilizar o módulo BLE seria com a sua característica típica de configuração em estrela. Para transmissão a outros nós que não pertencem ao cluster pode ser utilizada uma hierarquia em Scatternet. No entanto, é importante notar que sendo a principal diferença entre o BLE e o Bluetooth Clássico é o consumo energético o que afecta em grande parte a velocidade de transmissão de pacotes numa rede BLE. Por este motivo, a utilização do Bluetooth Clássico ao BLE é preferível quando é necessária transmissão considerável de dados, como é exemplo a emissão radiofónica por Bluetooth.

Por este motivo, uma rede BLE não é aconselhada ao uso em redes com muitos nós que realizem muitas retransmissões. Ao invés disso, poderá ser mais vantajoso o uso destes módulos como comunicação entre dados dentro do mesmo Cluster (entre sensores e Cluster-Head) e posteriormente estes são encaminhados para nós acima da árvore através de outro protocolo (por exemplo o 802.15.4, CTP), utilizando outro módulo. Desta forma, e sabendo que uma das suposições na concepção do BLE foi de que o nó Mestre tem menos restrições em termos de energia, os nós designados como Cluster-Head podem ser equipados com baterias de maior capacidade para equilibrar o tempo de vida entre todos os nós da rede.