

UNIVERSIDADE ESTADUAL PAULISTA

"JÚLIO DE MESQUITA FILHO"

Faculdade de Ciências - Campus Bauru

DEPARTAMENTO DE COMPUTAÇÃO

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

**DESENVOLVIMENTO DE UM PROTÓTIPO DE RASTREADOR DE
BEACONS UTILIZANDO O RASPBERRY PI**

BAURU

2015

GABRIEL LUIZ BASTOS OLIVEIRA

**DESENVOLVIMENTO DE UM PROTÓTIPO DE RASTREADOR DE
BEACONS UTILIZANDO O RASPBERRY PI**

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista Júlio de Mesquita Filho UNESP, Câmpus de Bauru.

Orientador: Prof. Dr. Eduardo Martins Morgado

BAURU
2015

GABRIEL LUIZ BASTOS OLIVEIRA

DESENVOLVIMENTO DE UM PROTÓTIPO DE RASTREADOR DE BEACONS UTILIZANDO O RASPBERRY PI

Trabalho de Conclusão de Curso apresentado ao Departamento de Computação da Faculdade de Ciências da Universidade Estadual Paulista Júlio de Mesquita Filho UNESP, Câmpus de Bauru.

BANCA EXAMINADORA

Aprovado em ____/____/____.

Prof. Dr. Eduardo Martins Morgado
Orientador

Prof. Dr. Aparecido Nilceu Marana

Profa. Dra. Márcia A. Z. Meira e Silva

BAURU
2015

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. Eduardo Morgado, por todo o tempo e trajetória no LTIA¹, pela confiança, apoio e incentivo. Agradeço também pelo auxílio em todas as etapas desse projeto, desde a definição do tema até a apresentação final.

A Henrique Lopes Santos, pela ajuda na realização das fotos e também criação do logotipo para o protótipo.

A Elisa Castro, pelo apoio na correção e melhorias na digitação do texto contido nesse documento.

A Giovanna Aguirre, pelo apoio, incentivo, paciência e companhia nos momentos mais importantes de minha vida.

A todos os colegas do laboratório, pela incrível parceria, trabalho em equipe e organização de projetos. Os quatro anos percorridos com eles foram essenciais para meu crescimento profissional e pessoal.

A todos os professores, pois com muito esforço diário passaram não somente as informações técnicas e práticas de cada disciplina, mas lições que servirão para toda vida.

A todos os colegas de curso, que de alguma forma auxiliaram a chegar ao último semestre da melhor maneira possível.

Aos meus amigos e familiares, pelo apoio e incentivo na escolha e decorrer do curso, com muita compreensão e encorajamento.

¹ Laboratório de Tecnologia da Informação Aplicada

*"Que eu não perca a vontade de ajudar as pessoas,
mesmo sabendo que muitas delas são incapazes
de ver, reconhecer e retribuir esta ajuda."
(Francisco Cândido Xavier)*

RESUMO

FALTA ACERTAR!!!!!! Atualmente a área da internet das coisas tem recebido um foco enorme. Um dos motivos é devido a ser bem nova e também facilitar a integração das pessoas com o mundo real. Essa área tem motivado várias pessoas pela possibilidade de ter um produto concreto após pouco tempo de trabalho. Por conta disso, alguns dispositivos foram criados para auxiliar essas áreas, entre eles o *Arduino* e *Raspberry Pi*. Os *beacons* também são uma criação nova, comunicando com outros dispositivos maiores como o *RPi* para auxiliar a micro-localização dentro de um pequeno espaço, com um baixo custo. O seu foco de uso é pequeno, somente com leituras via *smartphones*. Esse projeto tem como objetivo criar um rastreador de *beacons* utilizando o *Raspberry Pi* para expandir essa capacidade, não necessitando de um celular para que os *beacons* possam ser encontrados.

Palavras-chave: *Beacon*. *Raspberry Pi*. Internet das Coisas.

ABSTRACT

This is the english abstract.

Keywords: Beacon. Raspberry Pi. Internet of Things.

LISTA DE ILUSTRAÇÕES

Figura 1 – <i>RPi A+</i> (esquerda), <i>B+</i> (centro) e 2 <i>Modelo B</i> (direita)	17
Figura 2 – Separação da banda 2.4 GHz para bluetooth e WiFi	19
Figura 3 – Modelo de <i>BLE Advertising Packet</i>	19
Figura 4 – Modelo de <i>beacon</i> proprietário: MPact, da Zebra Technologies Corporation	20
Figura 5 – <i>Payload</i> do pacote iBeacon	22
Figura 6 – Pacote do <i>AltBeacon</i>	23
Figura 7 – <i>RPi 2</i> modelo B utilizado nesse projeto	25
Figura 8 – Orico BTA-406 a esquerda e EDUP N8508GS a direita	25
Figura 9 – Cartão microSD SanDisk Ultra Class 10 utilizado no projeto	25
Figura 10 – Conexão com o <i>RPi</i> via <i>SSH</i>	26
Figura 11 – Moto Maxx (esquerda) e iPad Mini (direita)	27
Figura 12 – <i>Beacon</i> Zebra MPact utilizado para testes	27
Figura 13 – <i>Beacon</i> configurado na <i>Toolbox</i> apresentando a porcentagem de bateria	28
Figura 14 – Caixa de metal utilizada para o protótipo	29
Figura 15 – Parte interna do protótipo	30
Figura 16 – Protoboard na etapa de estudo	30
Figura 17 – Primeiro teste realizado, com <i>RPi</i> e <i>beacon MPact</i>	32
Figura 18 – Software <i>hcitool</i> executando	33
Figura 19 – Software <i>hcidump</i> executando	33
Figura 20 – Teste com movimentação do <i>beacon</i>	34
Figura 21 – <i>RPi</i> posicionado de outra maneira	35
Figura 22 – <i>RPi</i> preso na caixa, com as portas acessíveis	36
Figura 23 – <i>RPi</i> posicionado de outra maneira	37
Figura 24 – Estudo de conexão e programação com display LCD	38
Figura 25 – Conexão com LEDs, display e botão na protoboard	38
Figura 26 – Teste de mesa com todos os componentes	38
Figura 27 – Pintura da caixa de metal - tampa	39
Figura 28 – Pintura da caixa de metal - base	39
Figura 29 – Rascunho de conexão das GPIOs	40
Figura 30 – Modelo de conexão dos botões	40
Figura 31 – Potenciômetros de configuração do display LCD	40
Figura 32 – Parte interna da interface - parcialmente desmontada	41
Figura 33 – Parte interna da interface - completamente montada	41
Figura 34 – Conexão da tampa com a base	42
Figura 35 – Criação e montagem do protótipo finalizada	42

Figura 36 – Definição das informações apresentadas nas telas	43
Figura 37 – LED de alto brilho ao olhar diretamente	44
Figura 38 – LEDs de alto brilho refletindo na mesa	44
Figura 39 – Gráfico ideal de um clique de botão	45
Figura 40 – Gráfico real de um clique de botão	45
Figura 41 – Protótipo finalizado e totalmente funcional	47
Figura 42 – TeXworks, programa utilizado para escrever o código \TeX	52
Figura 43 – BibDesk, programa utilizado para editar as referências do \TeX	53

LISTA DE TABELAS

Tabela 1 – Comparativo entre os modelos de <i>Raspberry Pi</i>	17
Tabela 2 – <i>BLE Physical Layer</i>	18
Tabela 3 – Exemplo de aplicação	21
Tabela 4 – Exemplo de tabela gerada com o \LaTeX	53

LISTA DE ABREVIATURAS E SIGLAS

IoT	<i>Internet of Things</i> - Internet das Coisas
DIY	<i>Do It Yourself</i> - Faça Você Mesmo
RPi	<i>Raspberry Pi</i>
GPIO	<i>General Input and Output</i>
BLE	<i>Bluetooth Low Energy</i>
LTIA	Laboratório de Tecnologia da Informação Aplicada

SUMÁRIO

1	INTRODUÇÃO	13
2	OBJETIVOS	15
2.1	Objetivos Gerais	15
2.2	Objetivos Específicos	15
3	FUNDAMENTAÇÃO TEÓRICA	16
3.1	Internet das Coisas	16
3.2	Raspberry Pi	16
3.3	Bluetooth Low Energy	18
3.4	Beacon	20
3.4.1	<i>iBeacon</i>	21
3.4.2	Outros Protocolos	22
4	MATERIAIS E MÉTODOS	24
4.1	Métodos e Etapas	24
4.2	Materiais Utilizados	24
4.2.1	Raspberry Pi e Acessórios	24
4.2.2	Computadores e Softwares	26
4.2.3	Smartphones e Tablets	26
4.2.4	Beacons	27
4.2.5	Acessórios para o Protótipo	29
4.3	Tecnologias e Ferramentas	30
4.3.1	Raspberry Pi	30
4.3.2	Node.js	30
4.3.3	CouchDB	30
4.3.4	Git	30
4.3.4.1	GitHub	31
4.3.5	Outras Ferramentas	31
5	DESENVOLVIMENTO	32
5.1	Primeira Etapa	32
5.2	Segunda Etapa	35
5.2.1	Problemas Enfrentados	43
5.3	Terceira Etapa	44
5.3.1	Primeiro Módulo	45

5.3.2	Segundo Módulo	45
5.3.3	Terceiro Módulo	46
5.3.4	Quarto Módulo	46
5.3.5	Quinto Módulo	46
5.3.6	Protótipo Final	46
6	CONCLUSÃO	48
	REFERÊNCIAS	49
	APÊNDICE A – LATEX	51
A.1	abnTeX2	51

1 INTRODUÇÃO

A internet cresce dia após dia e cada vez mais diferentes tipos de dispositivos são conectados a essa imensa rede. Há uma estimativa de 26 bilhões de "coisas" conectadas a internet até 2020, comparado a 6 bilhões na década de 2000. Isso foi possível graças ao custo do acesso a internet e banda larga ter diminuído 40 vezes nos últimos 10 anos. Esse *boom* de crescimento também é influenciado pela IoT (*Internet of Things* - Internet das Coisas). (GOLDMAN SACHS, 2014).

Segundo Ashton (2009) o termo IoT surgiu como título de sua apresentação a Procter & Gamble (P&G) em 1999. Esse termo se dá ao uso de internet em diferentes tipos de dispositivos.

"(...) a Internet das Coisas é um conceito no qual dispositivos de nosso dia a dia são equipados com sensores capazes de captar aspectos do mundo real, como por exemplo, temperatura, umidade, presença, etc, e envia-los a centrais que recebem estas informações e as utilizam de forma inteligente." (NASCIMENTO, 2015).

Como exemplo podemos citar: geladeira ligada a internet informando a falta de condimentos, caixa de remédio conectada a internet prevendo o término da caixa de remédio para avisar ao consumidor, entre diversos outros. Um bom exemplo é interligar uma casa por meio de sensores, como termostatos, sistemas de segurança, iluminação, sistemas de entretenimento com uma inteligência por trás para diversas aplicações. (GOLDMAN SACHS, 2014)

Diversas áreas tiveram o seu crescimento alavancado por conta da IoT. O mais marcante é o ramo do DIY (*Do It Yourself*, ou faça você mesmo), em que pessoas criam ou adaptam coisas para suas necessidades. Isso se dá por conta de ter aparecido no mercado dispositivos e componentes que auxiliam a criação e adaptação de eletrônicos e outros. Um ótimo exemplo é o Arduino, um dispositivo que nos ajuda a criar os projetos de eletrônica que consiste de duas partes: o hardware e o software. Com eles é possível construir praticamente de tudo, desde um LED piscante a um robô que envia um *tweet* quando sua planta está sem água. (BEN, 2015). (SORREL, 2008).

Além do Arduino existem diversos outros *devices* com funcionalidades similares ou até complementares. Podemos citar o *Raspberry Pi*, um computador do tamanho de um cartão de crédito e de baixo custo. (RASPBERRY PI, 2015b). Por ser um computador é possível executar um sistema operacional (como Linux, *RISC OS*, *Windows 10 for IoT*). Dependendo do modelo possui portas USB (1 a 4) para conexão com periféricos (como mouse, teclado, adaptador WiFi, Bluetooth), e também porta Ethernet para conexão a in-

ternet cabeadas (exceto modelo A e A+). Também possui de 26 (modelo A e B) a 40 pinos (modelo A+, B+ e 2) para conexões gerais de entrada e saída digitais (GPIO - *General Input and Output*).

Através desses pinos pode-se conectar uma diversidade de componentes eletrônicos como sensores, atuadores, outros dispositivos para comunicação. Dessa forma, suas funcionalidades são expandidas de uma forma absurda, ficando a cargo de cada pessoa montar uma nova aplicação. Uma boa aplicação é a conexão de um adaptador bluetooth pela USB para comunicação com smartphones, tablets, PCs e outros dispositivos tipo sensores sem fio.

Um exemplo desses sensores externos são os *beacons*, pequenos modelos sem fio que se comunicam por meio do Bluetooth 4.0 ou BLE (*Bluetooth Low Energy* - Bluetooth de Baixa Energia). Como o próprio nome diz, esse padrão de comunicação utiliza pouca energia. Desta forma, um *beacon* pode funcionar por anos. "Na prática, ela permite localizar objetos (ou pessoas que carregam esses objetos) com muito mais precisão dentro de ambientes fechados." (TEIXEIRA, 2014).

Os *beacons* foram pouco explorados até o momento, seu uso está sendo mais notado na área de grandes lojas do varejo.

"A Apple (...) já está utilizando a tecnologia em 254 lojas nos EUA. As funcionalidades já estão embutidas na versão oficial do aplicativo da Apple Store para iOS [, sistema operacional de seus smartphones e tablets]. (...) Quando o usuário se aproxima de uma loja física, o aplicativo oferece toda uma camada extra de informações e serviços que são específicos para aquela unidade como por exemplo ofertas locais, tamanho da fila para ser atendido no Genius Bar, eventos e treinamentos que estão agendados ali na loja etc." (TEIXEIRA, 2014).

A Macy's (grande rede norte americana de loja de departamentos) está realizando testes em algumas de suas lojas para enviar alertas a pessoas que entrarem em suas lojas, com promoções e melhores indicações, utilizando o padrão *iBeacon* da Apple. Até o momento esse teste está limitado a usuários de iPhone, e somente quando entrar na loja. Em um teste futuro espera-se que seja possível separar por departamentos, para que quando um usuário percorra a loja apareça as notícias relativo ao local da loja que ela está. (KASTRENAKES, 2013)

2 OBJETIVOS

2.1 Objetivos Gerais

O objetivo geral desse projeto é planejar e desenvolver um protótipo de rastreador de *iBeacons* utilizando o *Raspberry Pi*.

2.2 Objetivos Específicos

- a) Estudar o funcionamento de *beacons*, comunicação via *bluetooth low energy* e *Raspberry Pi*, assim como suas aplicações;
- b) Identificar os requerimentos básicos de funcionamento de *beacons* e *Raspberry Pi*;
- c) Definir os elementos para desenvolvimento do protótipo;
- d) Planejar a estrutura do sistema;
- e) Implementar o protótipo de acordo com a estrutura e elementos planejados.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 Internet das Coisas

A área de IoT está em uma onda crescente, com um grande número de pessoas investindo nesse mercado, e um bom número de profissionais migrando para essa área. Há uma estimativa de que existem 19 milhões de profissionais trabalhando na indústria de desenvolvimento de software, e desses, 19% trabalham em algum projeto relacionado a IoT. (THIBODEAU, 2015).

"A próxima onda na era da computação será fora do domínio do ambiente de trabalho tradicional. No paradigma da IoT, muitos dos objetos que nos rodeiam estarão na rede de uma forma ou de outra. RFID (Radio Frequency Identification - Identificadores via Rádio Frequênci) e as tecnologias de redes de sensores crescerão para enfrentar este novo desafio, em que os sistemas de informação e comunicação estão embutidos nos ambientes que nos rodeiam, de forma invisível.". (GUBBI et al., 2013).

É notável o crescimento dessa área. Há uma expectativa de que, em 2020, o número de carros conectados a internet supere o número de carros não conectados, sendo esses carros possíveis de se comunicar com outros veículos e a infraestrutura das ruas, como os semáforos. (GOLDMAN SACHS, 2014). Segundo Press (2014), em 2014 a IoT substituiu a área da *Big Data* como a tecnologia mais empolgante, ou seja, a tecnologia que mais pessoas iriam migrar e se interessar.

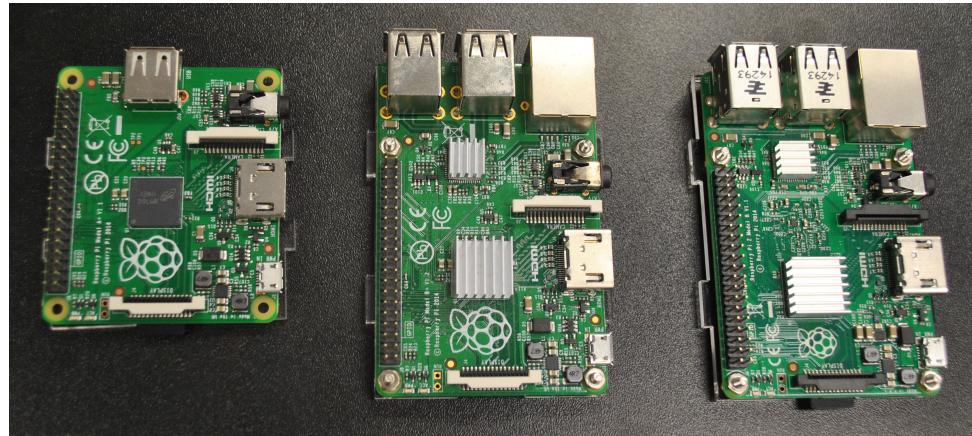
3.2 Raspberry Pi

Quando se pesquisa algo sobre IoT é praticamente impossível não achar relação e links para *Raspberry Pi*, Arduinos, e toda a área de *D/Y*. Atualmente existem três modelos, conforme a tabela Tabela 1 e figura Figura 1.

Segundo Raspberry Pi (2015b), a fundação responsável por esse pequeno dispositivo criou com intuito de ser utilizado por crianças e pessoas carentes do mundo todo para aprender programação e computação. É facilmente configurável, utilizando um cartão de memória como HD para o sistema operacional e dados.

Por meio de suas portas de entrada e saída digitais é possível conectar uma gama ampla de sensores, atuadores, componentes eletrônicos, ficando a cargo do programador e criador do projeto escolher como esses pinos serão conectados e aproveitados. Atualmente existem diversos módulos para expandir os meios de comunicação entre diferentes *devices*, e um deles é via *bluetooth*.

Figura 1 – *RPi A+* (esquerda), *B+* (centro) e 2 *Modelo B* (direita).



Fonte: elaborado pelo autor

Tabela 1 – Comparativo entre os modelos de *Raspberry Pi*

Versão	A+	B+	2 Modelo B
Processador	ARMv6 single core	ARMv6 single core	ARMv7 quad core
Velocidade CPU	700 MHz single-core	700 MHz single-core	900 MHz quad-core
Memória RAM	256 MB	512 MB	1 GB
Portas USB	1	4	4
<i>Ethernet</i>	Não	Sim	Sim

Fonte: (MAKER SHED, 2015)

Atualmente existem diferentes sistemas operacionais portados para o *RPi*.

- a) **Raspbian**: baseado na distribuição *Linux* chamada *Debian*. Atualmente é a suportada oficialmente pela *RPi Foundation*. (RASPBERRY PI, 2015a).
- b) **Ubuntu Mate**: baseado na distribuição *Linux* chamada *Ubuntu*, juntamente com o software *MATE Desktop* para gerenciamento de janelas. (UBUNTU MATE, 2015).
- c) **Snappy Ubuntu Core**: distribuição *Linux* voltada a *cloud* e dispositivos. (CANONICAL LTD., 2015).
- d) **Windows 10 for IOT Core**: versão do *Windows 10* da *Microsoft* para dispositivos voltados a internet das coisas. (MICROSOFT, 2015).

3.3 Bluetooth Low Energy

A tecnologia do *bluetooth* vem sendo amplamente utilizada, principalmente após a criação da versão 4.0, ou *BLE* (*Bluetooth Low Energy* - *Bluetooth de Baixa Energia*), por conta de ter um baixíssimo gasto energético, podendo preservar a bateria do dispositivo que a utiliza. O *BLE* faz parte da tecnologia nomeada *Bluetooth Smart* inteligente e eficiente energeticamente, voltada para *devices* que usam pequenas fontes de energia. (BLUETOOTH SIG, 2015a).

O *BLE* possui similaridades com a versão clássica do *bluetooth*. Ambos utilizam o espectro de frequências de 2.4 GHz (mesmo utilizado pelas redes *WiFi*), mesma modulação GFSK e velocidade de 1 Mbps, porém a indexação de ambos é diferente. A versão clássica possui 79 canais, e a *BLE* possui 40 canais. Além disso, os canais são espaçados de forma diferente, conforme tabela 2. (ARGENOX, 2015).

Tabela 2 – *BLE Physical Layer*

	BLE	Classic
Modulação	GFSK 0.45 a 0.55	GFSK 0.28 a 0.35
Velocidade de Transferência	1 Mbit/s	1 Mbit/s
Canais	40	79
Espaçamento	2 MHz	1 MHz

Fonte: (ARGENOX, 2015)

O espectro de 2.4 GHz para *bluetooth* se extende de 2402 MHz a 2480 MHz, e os canais 37, 38 e 39 (últimos três) são específicos para anúncio (*advertisement*), conforme figura 2. (ARGENOX, 2015).

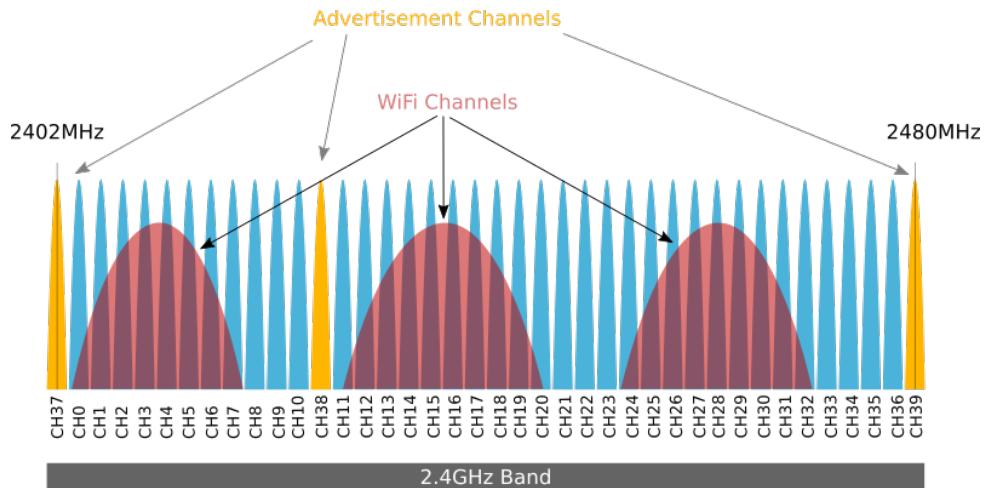
Interessante notar que esses canais estão posicionados em forma bastante estratégica, no começo, final e meio da banda de 2.4 GHz. Isso se deve para aumentar a eficácia, evitando todos os canais ficarem lotados ou com muita interferência. (ARGENOX, 2015).

Os *BLE Advertisement Packets*, ou pacotes de anúncio *BLE* é uma das formas de conexão do *Bluetooth Smart*. Por meio dos anúncios, um *device* transmite pacotes para todos que estão a sua volta, sem necessariamente necessitar de uma conexão direta entre somente outro dispositivo.

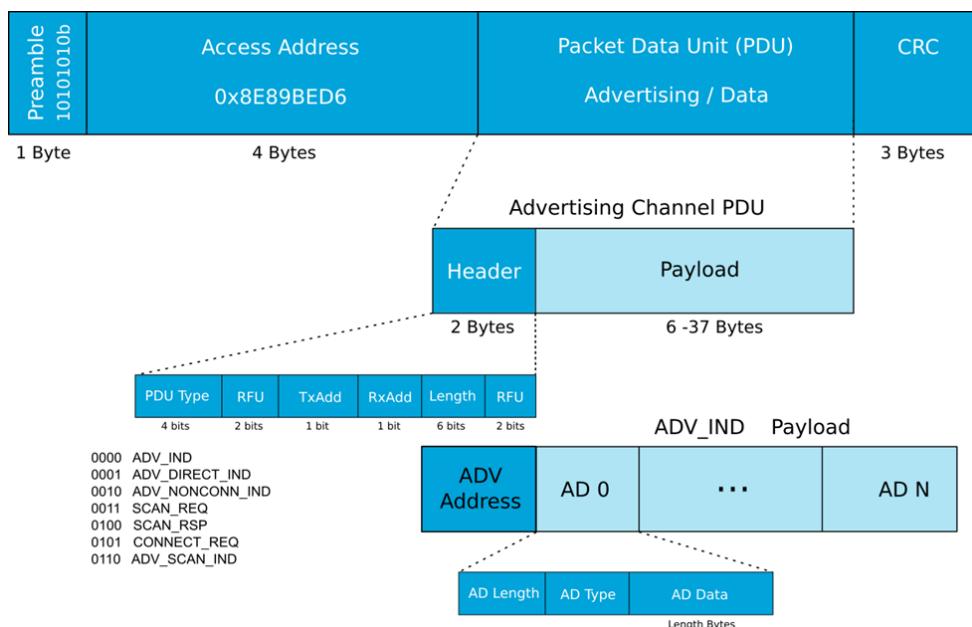
Um *BLE Advertisement Packet* é formado conforme figura 3. O preâmbulo, *access address* e CRC são informações para formação do pacote. Os dados estão de fato dentro do PDU (*Packet Data Unit*). O cabeçalho de 2 bytes informa o tamanho do *payload* (carga de dados), além de informações relevantes como tipo do pacote, tipo de mensagem enviada, entre outros. (ARGENOX, 2015).

O importante do *BLE Advertising Packet* é o tipo de anúncio feito, ou quais são

Figura 2 – Separação da banda 2.4 GHz para bluetooth e WiFi.



Fonte: (ARGENOX, 2015)

Figura 3 – Modelo de *BLE Advertising Packet*.

Fonte: (ARGENOX, 2015)

as informações do pacote. Bluetooth SIG (2015b) apresenta uma tabela com os possíveis valores e também o significado de cada uma. Por exemplo, o valor 0xFF significa que o pacote contém dados específicos do fabricante, ou seja, existe a flexibilidade de manipular o pacote da forma que for preciso, contanto que mantenha a estrutura original de 6 a 37 bytes de payload, conforme figura 3. (ARGENOX, 2015).

3.4 Beacon

Os *beacons* são pequenos sensores que são capazes de identificar objetos com precisão dentro de ambientes fechados. (TEIXEIRA, 2014).

Como muitos espaços fechados (restaurantes, museus, shopping centers, casas de show) possuem estrutura metálica ou utilizam algum tipo de metal em sua construção, é comum que o sinal de GPS fique enfraquecido quando os usuários estão dentro daquele local. Nesse caso, os *Beacons* são uma ótima solução: um hardware relativamente barato, e pequeno o suficiente para ser plugado na parede ou instalado sobre um balcão. (TEIXEIRA, 2014).

Figura 4 – Modelo de *beacon* proprietário: MPact, da Zebra Technologies Corporation.



Fonte: elaborado pelo autor

Segundo Teixeira (2014), os *beacons* utilizam o *BLE* para detectar um dispositivo próximo e transmitir seu identificador único e avisar que está ali presente. Teixeira (2014) também diz que os *beacons* não são inteligentes, toda a interação deve depender do dispositivo que recebe a informação do identificador único.

Atualmente os usos de *beacons* estão restritos a aplicativos em smartphones realizando a leitura e interagindo com o usuário, porém existem ainda diversas áreas a serem exploradas, e um bom exemplo são as casas inteligentes. Segundo Grothaus (2014), a *Apple* está apostando em um kit de desenvolvimento (*HomeKit*) que permita aos desenvolvedores interagirem com *smart devices* presentes no ambiente.

3.4.1 iBeacon

O protocolo *iBeacon* foi apresentado pela Apple juntamente com o iOS 7, versão de seu sistema operacional para dispositivos móveis. É uma tecnologia baseada nos *beacons*, porém adaptada para as necessidades e aplicações de seu sistema móvel.

A Apple adaptou o pacote genérico de *beacon* para transmitir três dados:

- UUID**: Identificador único formado de 16 bytes (128 bits). Focado em ser único para cada aplicação. Cada aplicação deve ter um único UUID.
- Major**: Identificador de 2 bytes que identifica uma sub-região grande. Usado, por exemplo, para dividir as lojas de um grande varejista.
- Minor**: Identificador de 2 bytes que identifica uma sub-divisão de região, ou seja, uma região menor que o Major.

Um exemplo de aplicação é citado na tabela 3. Utiliza-se um único UUID para todas as lojas, um número Major por loja e um Minor por departamento, podendo este último ser repetido entre as lojas.

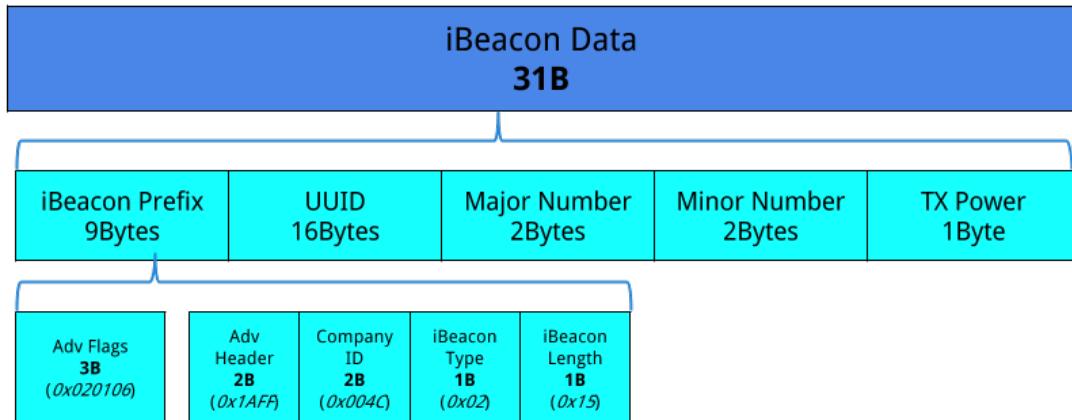
Tabela 3 – Exemplo de aplicação

Localização da Loja	São Francisco	Paris	Londres
UUID	D9B9EC1F-3925-43D0-80A9-1E39D4CEA95C		
Major	1	2	3
Minor (Roupas)	10	10	10
Minor (Utilidades Domésticas)	20	20	20
Minor (Automotivo)	30	30	30

Fonte: (APPLE INC, 2014)

O pacote *BLE* utilizado pela tecnologia *iBeacon* pode ser visto na Figura 5. Segundo Austin (2015), o prefixo determinado para o protocolo *iBeacon* é:

- Adv Flags**: determinam que é um pacote *BLE* de descobrimento geral, e que somente transmite e não permite conexões.
- Adv Header**: determinam que os próximos 26 bytes serão a carga de dados (*payload* de fato). Sempre será 0x1AFF.
- Company ID**: indica que é o ID da Apple junto com a Bluetooth SIG. Essa informação que faz ser dependente da Apple. Sempre será 0x004C.
- iBeacon Type**: ID secundário utilizado por todos *iBeacons* que identificam ser um *beacon* de proximidade. Sempre será 0x02.
- iBeacon Length**: identifica quantos bytes terão em seguida. Sempre será 0x15, ou 21 bytes.

Figura 5 – *Payload* do pacote iBeacon.

Fonte: (AUSTIN, 2015)

Os *iBeacons* foram criados com intuito de serem descobertos por smartphones. Um exemplo de aplicação é uma cafeteria com um *iBeacon* no balcão próximo ao caixa. Quando um consumidor entra na loja e chega próximo ao caixa, um aplicativo em seu celular identifica o *iBeacon* pela sua UUID, identifica pelo Major que se trata da cafeteria número 12 e encontra uma promoção com o Minor de número 26. Em seguida, apresenta uma notificação ao usuário uma promoção e também um cupom válido de desconto para usar no caixa. (AUSTIN, 2015).

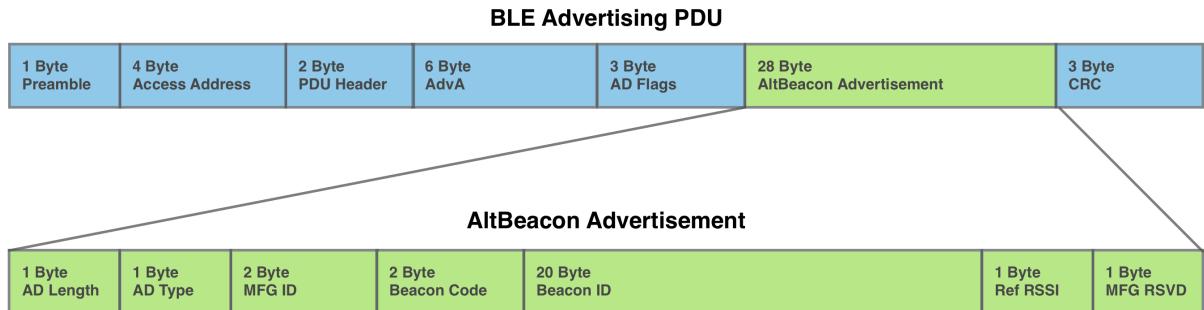
Uma outra aplicação interessante citada por Austin (2015) é a possibilidade do smartphone transmitir pacotes *iBeacon*, sem necessidade de um hardware externo. Dessa forma, pode-se por exemplo automatizar o check-in em um evento e rastrear o movimento dos usuários entre os estabelecimentos.

3.4.2 Outros Protocolos

Existem mais protocolos baseados na tecnologia *beacon*. Dois se destacam por ser abertos e passíveis de alterações: *AltBeacon* e *Eddystone*, este último criado pela Google. O *AltBeacon* possui um modelo bastante parecido com o *iBeacon*, porém com possibilidade de alterar o número do fabricante, ter possibilidade de código de *beacons* diferentes, e também a possibilidade do fabricante colocar sua informação ao final do pacote, conforme figura 6 (AUSTIN, 2015).

Segundo Wandschneider Nirdhar Khazanie (2015), o protocolo *Eddystone* possui três modos de funcionamento:

- a) *Eddystone-UID*: transmite um *beacon* ID, composto de 10 bytes identificando um grupo de *beacons* e 6 bytes identificando um único *beacon*.

Figura 6 – Pacote do *AltBeacon*.

Fonte: (AUSTIN, 2015)

- b) *Eddystone-URL*: transmite um link comprimido, para que o cliente possa acessar um site na internet.
- c) *Eddystone-TLM*: transmite informações de telemetria sobre o *beacon*, como por exemplo voltagem da bateria, temperatura e quantos pacotes foram enviados.

4 MATERIAIS E MÉTODOS

4.1 Métodos e Etapas

O projeto foi dividido em etapas para facilitar o desenvolvimento completo. A primeira etapa foi o levantamento bibliográfico relacionado ao tema, realizadas buscas relacionadas aos assuntos: *Raspberry Pi*, comunicação via *Bluetooth Low Energy*, *beacons*. Concomitantemente foi realizado o estudo das tecnologias, levando em conta suas capacidades, limitações, e aplicações.

A segunda etapa foi planejar o projeto baseado na análise do levantamento bibliográfico, assim como a definição de sua estrutura. Essa etapa foi necessária para facilitar e agilizar a implementação e testes dos componentes nas próximas etapas, definindo assim um escopo inicial de funcionalidades que o sistema terá, assim como outras tarefas a serem realizadas. O processo está detalhado no Capítulo 5 - Desenvolvimento.

A terceira etapa foi a implementação e testes do protótipo. As funcionalidades foram desenvolvidas por parte e em seguida testadas, seguindo a metodologia de desenvolvimento ágil.

Todo o projeto foi desenvolvido no espaço do LTIA¹, da Universidade Estadual Paulista "Júlio de Mesquita Filho" - Campus Bauru.

4.2 Materiais Utilizados

4.2.1 Raspberry Pi e Acessórios

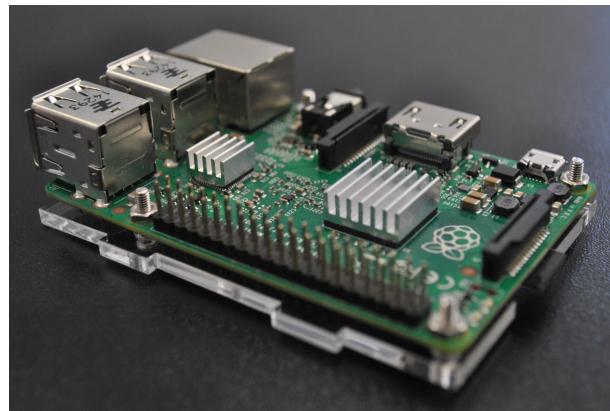
A versão do Raspberry Pi escolhida para o projeto foi a 2 Modelo B, por ter mais processamento e mais memória RAM (Figura 7), conforme informado na seção 3.2.

É necessário o uso de um adaptador *WiFi* para conexão a internet sem necessidade de cabo *Ethernet* e também um adaptador Bluetooth 4.0 com suporte a *BLE* para fazer a busca dos pacotes *beacon*. Os modelos de adaptadores usados foram Orico BTA-406 (*bluetooth*) e EDUP N8508GS (*WiFi*), conforme Figura 8. O *RPi* e os adaptadores são de propriedade do autor.

O sistema operacional executado no *RPi* é instalado em um cartão microSD, com no mínimo 4 GB de espaço. O cartão utilizado nesse projeto é um SanDisk Ultra Class 10 de 8 GB, conforme Figura 9. O sistema utilizado foi o Raspbian Wheezy. Durante o de-

¹ Laboratório de Tecnologia da Informação Aplicada - <<http://www.ltia.fc.unesp.br/>>

Figura 7 – RPi 2 modelo B utilizado nesse projeto.



Fonte: elaborado pelo autor

Figura 8 – Orico BTA-406 a esquerda e EDUP N8508GS a direita.



Fonte: elaborado pelo autor

envolvimento do projeto, a versão Raspbian Jessie foi lançada, com inúmeras melhorias, porém preferiu-se por permanecer na versão Wheezy para evitar possíveis incompatibilidades com os softwares utilizados.

Figura 9 – Cartão microSD SanDisk Ultra Class 10 utilizado no projeto.



Fonte: elaborado pelo autor

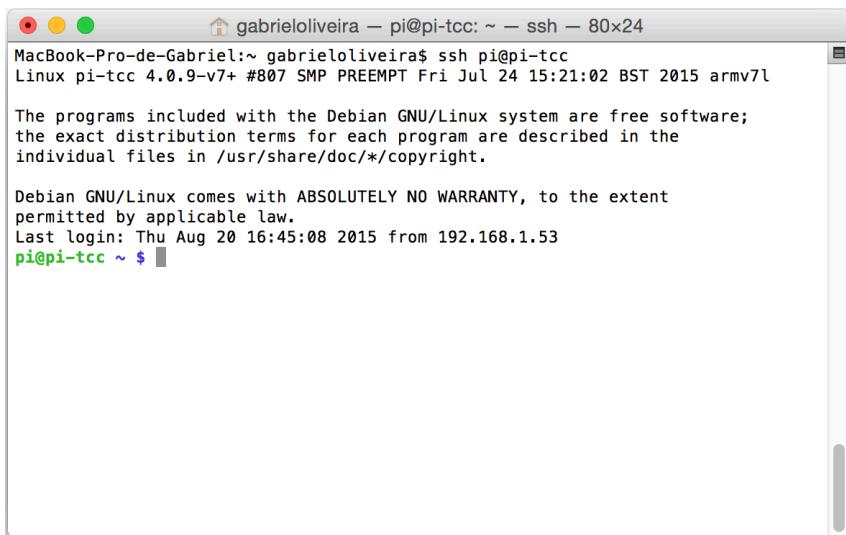
Na seção 4.3 - Tecnologias e Ferramentas, subseção 4.3.1 - Raspberry Pi será

abordado como o *RPi* funciona na prática, como foi utilizado para esse projeto e como suas capacidades foram aproveitadas.

4.2.2 Computadores e Softwares

A conexão com o *RPi* foi realizada utilizando *SSH*², sem necessidade de monitor e teclado. O computador utilizado no projeto foi um MacBook Pro com sistema Mac OS X 10.10 (posteriormente atualizado para 10.11), de propriedade do autor. Nesse sistema o uso de *SSH* é simples, basta abrir o aplicativo Terminal e utilizar o comando "ssh usuário@computador", conforme Figura 10. Esse tipo de abordagem é bastante utilizada para conexão a servidores na nuvem, para executar comandos, softwares, entre outros.

Figura 10 – Conexão com o *RPi* via *SSH*.



```
gabrieloliveira — pi@pi-tcc: ~ — ssh — 80x24
MacBook-Pro-de-Gabriel:~ gabrieloliveira$ ssh pi@pi-tcc
Linux pi-tcc 4.0.9-v7+ #807 SMP PREEMPT Fri Jul 24 15:21:02 BST 2015 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Thu Aug 20 16:45:08 2015 from 192.168.1.53
pi@pi-tcc ~ $
```

Fonte: elaborado pelo autor

4.2.3 Smartphones e Tablets

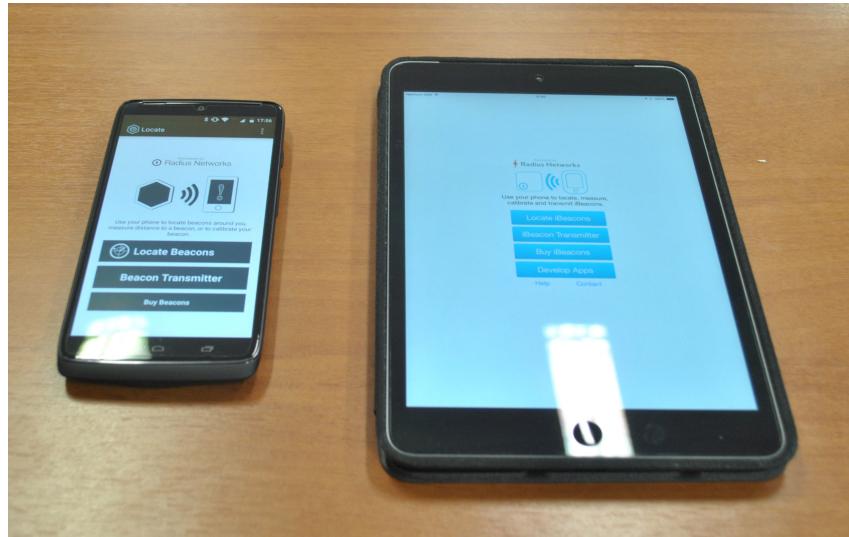
Foi utilizado o smartphone Moto Maxx com sistema Android 5.0.1 e o tablet iPad mini Retina com sistema iOS 8.4 confirme Figura 11.

O aplicativo utilizado em ambos foi o *Locate Beacon* da *Radius Networks*, que nos permite identificar os *beacons* e também simular um, para realizar os testes com diferentes tipos, podendo alterar os valores de UUID, Major, Minor e potência de transmissão.

O Moto Maxx é de propriedade do autor, e o iPad é de propriedade do LTIA, foi utilizado nas dependências do laboratório.

² Secure Shell

Figura 11 – Moto Maxx (esquerda) e iPad Mini (direita).



Fonte: elaborado pelo autor

4.2.4 Beacons

O *beacon* utilizado para testes foi o *Zebra MPact*, conforme Figura 12. Utiliza uma bateria CR2450 para alimentação de energia.

Figura 12 – *Beacon Zebra MPact* utilizado para testes.



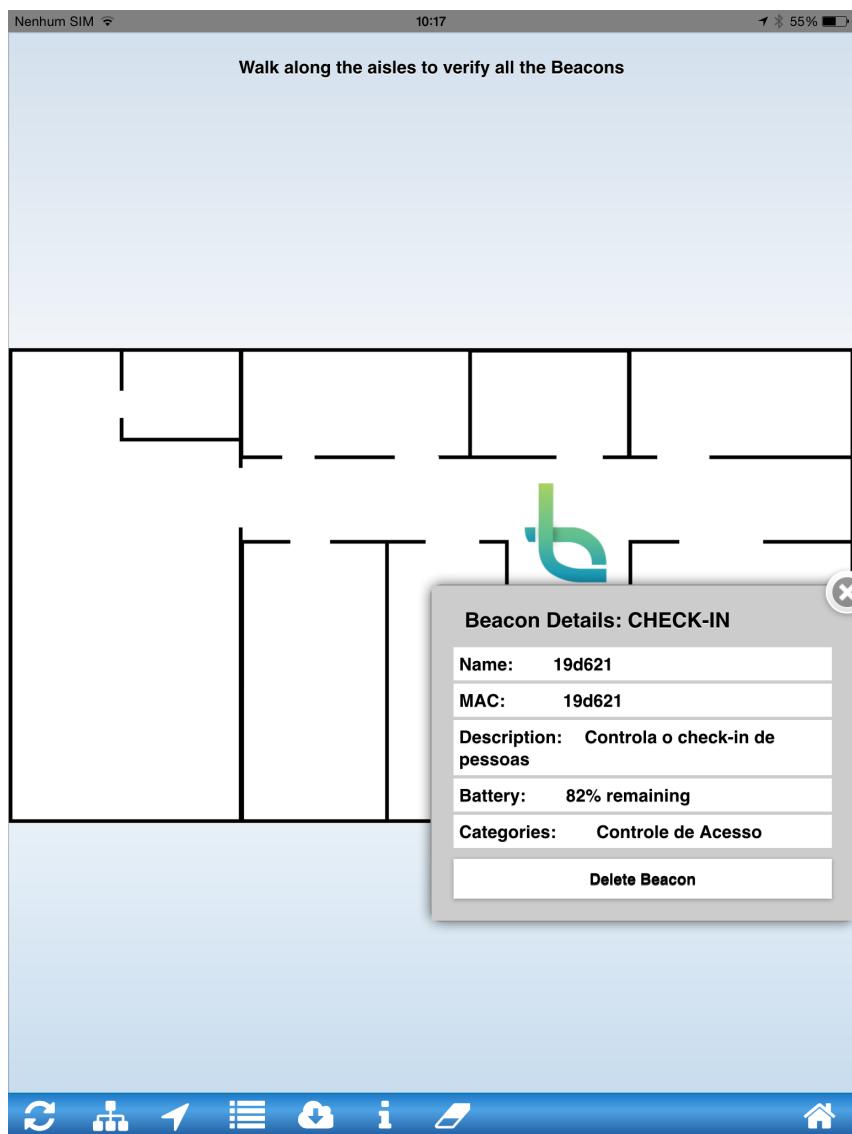
Fonte: elaborado pelo autor

A *Zebra* tem um sistema de administração e gerenciamento nomeado *MPact Toolbox*, instalado em um servidor do LTIA com sistema Debian 8.1. Esse sistema foi utilizado

somente para conhecimento da tecnologia *beacon* e atualização do *firmware*, somente disponível por essa *Toolbox*. Esse software também apresenta a porcentagem de bateria, conforme Figura 13.

Permite a configuração de vários *beacons* simultaneamente, além de alterar o modo de funcionamento, de *iBeacon* para *MPact*, protocolo criado pela fabricante. Os *beacons* são de propriedade do LTIA, e foram utilizados dentro das dependências do laboratório em todas as etapas.

Figura 13 – *Beacon* configurado na *Toolbox* apresentando a porcentagem de bateria.



Fonte: elaborado pelo autor

4.2.5 Acessórios para o Protótipo

O protótipo foi criado em uma caixa de metal (Figura 14), furada conforme projeto e pintada de preto. Esse processo está detalhado na seção 5.2.

Figura 14 – Caixa de metal utilizada para o protótipo



Fonte: elaborado pelo autor

Para o protótipo final foram utilizados seis LEDs, resistores, barra de pinos, um display azul de 16 colunas por 2 linhas, três botões do tipo clique, cabos *flat* do tipo IDE e *floppy*, placas com furação para soldagem de componentes, cabos finos removidos de um cabo de rede *ethernet* para ligações entre os componentes e cola quente para fixação dos componentes internos. Esses materiais podem ser vistos na Figura 15.

Durante a fase de estudo das tecnologias utilizou-se uma protoboard e fios de conexão (Figura 16) para testes de código, modo de conexão, entre outros, detalhado na seção 5.2.

Como o *RPi* tem limitações quanto a quantidade de pinos de entrada e saída digitais (explicados na subseção 4.3.1), os componentes definidos para o protótipo foram determinados conforme possibilidade de conexões. Cada componente necessita de:

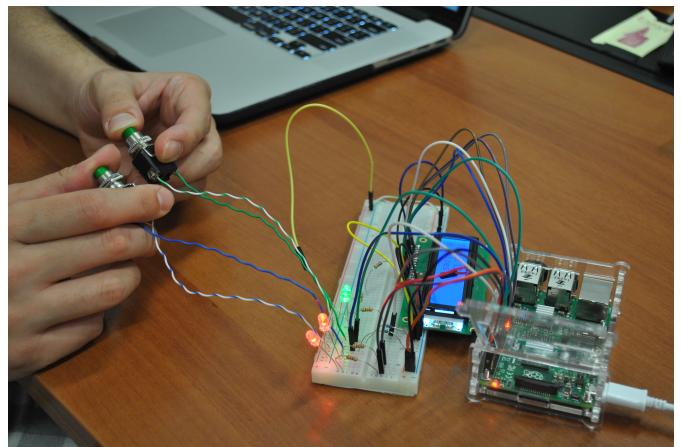
- a) Para cada LED, um pino de saída digital (GPIO) para acender ou apagar;
- b) Para cada botão, um pino de entrada digital para leitura de botão pressionado ou não;

Figura 15 – Parte interna do protótipo



Fonte: elaborado pelo autor

Figura 16 – Protoboard na etapa de estudo



Fonte: elaborado pelo autor

- c) Para o display de LCD, seis pinos de saída no total, sendo quatro para dados, um para *enable* (ou ativar), e outro para *register select*, para posicionar os dados nos registradores corretos.

O uso desses componentes foram determinados na segunda etapa, detalhada na seção 5.2 - Segunda Etapa.

4.3 Tecnologias e Ferramentas

4.3.1 Raspberry Pi

O *RPi* é a tecnologia mais importante desse projeto. Já foi abordado na seção 3.2 e subseção 4.2.1, porém nessa seção outros aspectos serão comentados.

4.3.2 Node.js

4.3.3 CouchDB

4.3.4 Git

Git é um software de controle de versão para repositórios. Registra as mudanças nos arquivos (ou grupo de arquivos) ao longo do tempo, de forma que as versões possam ser recuperadas futuramente. (CHACON, 2014). Se for necessário voltar algumas versões por bug no código, mal funcionamento de uma função recém implementada ou outras razões, as alterações estarão todas salvas.

Atualmente o Git é muito utilizado por projetos de código aberto, como por exemplo o Swift³ que é uma linguagem de programação criada pela Apple voltada para seus

³ <<https://github.com/apple/swift>>

sistemas operacionais, e o AngularJS⁴, framework Javascript mantido pelo Google.

É possível configurar o repositório local para sincronizar com um repositório remoto, por meio de `push` (enviar mudanças) e `pull` (buscar mudanças). Dessa forma promove a divisão do trabalho, removendo o problema de sincronização de código entre diferentes máquinas e pessoas.

4.3.4.1 GitHub

4.3.5 Outras Ferramentas

Durante o desenvolvimento desse trabalho foi necessário a criação de uma proposta e um relatório parcial, ambos para a disciplina de Projeto e Implementação de Sistemas I. Para tal, o autor utilizou a ferramenta denominada LaTeX. Como não está entre os objetivos desse projeto, mas é informação de grande acréscimo, está detalhado no Apêndice A.

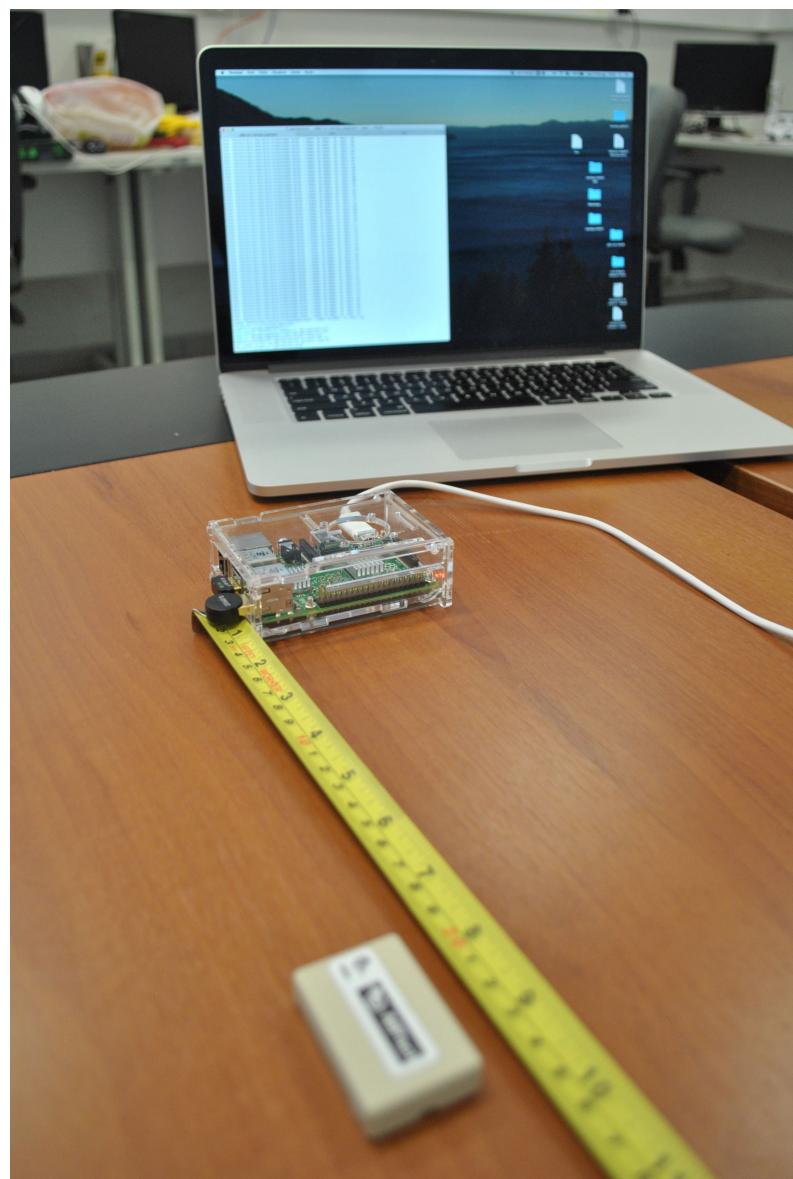
⁴ <<https://github.com/angular/angular.js>>

5 DESENVOLVIMENTO

5.1 Primeira Etapa

Nessa etapa foram realizados os estudos práticos das tecnologias. O primeiro experimento realizado foi a tentativa de identificação de um *beacon* com o *RPi*, para o estudo de funcionamento e comportamento dessas tecnologias. Para isso, o ambiente foi configurado conforme Figura 17.

Figura 17 – Primeiro teste realizado, com *RPi* e *beacon MPact*



Fonte: elaborado pelo autor

O computador ficou conectado via SSH com o *RPi*, recebendo as informações de leitura de pacotes *BLE*. Os softwares utilizados para isso foram, conforme Jjnebeker (2014), os seguintes:

- a) **hcitool**: configurado da maneira *hcitool lescan –duplicates*, faz um scan na frequência *BLE* procurando por dispositivos que estejam transmitindo, conforme Figura 18.

Figura 18 – Software *hcitool* executando

Fonte: elaborado pelo autor

- b) **hcidump**: em conjunto com o *hcitool*, apresenta todos os pacotes escaneados na rede. Executado da maneira *hcidump –raw*, conforme Figura 19.

Figura 19 – Software *hcidump* executando

```
gabrieloliveira ~ pi@pi-tcc: ~ - ssh - 80x24
pi@pi-tcc ~ $ sudo hcidump --raw
HCI sniffer - Bluetooth packet analyzer ver 5.18
device: hci0 snap_len: 1500 filter: 0xffffffff
> 04 3E 2A 02 01 03 00 1E 1F 54 B8 5F D0 1E 02 01 06 1A FF 4C
  00 02 15 FE 91 32 13 B3 11 4A 42 8C 16 47 FA EA C9 38 DB 00
  01 00 52 AF AF
> 04 3E 2A 02 01 03 00 1E 1F 54 B8 5F D0 1E 02 01 06 1A FF 4C
  00 02 15 FE 91 32 13 B3 11 4A 42 8C 16 47 FA EA C9 38 DB 00
  01 00 52 AF AF
> 04 3E 2A 02 01 03 00 1E 1F 54 B8 5F D0 1E 02 01 06 1A FF 4C
  00 02 15 FE 91 32 13 B3 11 4A 42 8C 16 47 FA EA C9 38 DB 00
  01 00 52 AF AF
> 04 3E 2A 02 01 03 00 1E 1F 54 B8 5F D0 1E 02 01 06 1A FF 4C
  00 02 15 FE 91 32 13 B3 11 4A 42 8C 16 47 FA EA C9 38 DB 00
  01 00 52 AF AF
> 04 3E 2A 02 01 03 00 1E 1F 54 B8 5F D0 1E 02 01 06 1A FF 4C
  00 02 15 FE 91 32 13 B3 11 4A 42 8C 16 47 FA EA C9 38 DB 00
  01 00 52 AF AF
^Cpi@pi-tcc ~ $
```

Fonte: elaborado pelo autor

Em seguida, com os softwares em execução e os pacotes sendo analisados, o *beacon* foi movido em cima da mesa para ficar a diferentes distâncias do *RPi*, conforme Figura 20. Esse passo foi necessário para verificar o formato dos pacotes recebidos e também analisar a distância máxima de identificação.

Figura 20 – Teste com movimentação do *beacon*



Fonte: elaborado pelo autor

Com o adaptador Orico BTA-406 e posicionamento na mesa conforme Figura 17 e Figura 20, cerca de 1,3 a 1,4 metros de distância entre o *RPi* e o *beacon* os pacotes já começaram a falhar e a leitura não ser tão constante.

O posicionamento do *RPi* foi alterado para testes, conforme Figura 21. Notou-se uma melhoria na recepção dos pacotes, a 1,5 metros entre o *RPi* e o *beacon* os pacotes começaram a falhar, com recepção notada até 1,6 metros.

Durante essa etapa não foram levado em consideração conexões com LEDs, bo-

Figura 21 – *RPi* posicionado de outra maneira



Fonte: elaborado pelo autor

tões e display LCD para a montagem do protótipo pois esses componentes foram definidos posteriormente. Os estudos e testes desses materiais foram realizados na segunda etapa, para que as escolhas fossem validadas.

5.2 Segunda Etapa

Na segunda etapa iniciou-se o planejamento do protótipo. Para que pudesse ser bem planejado, definiu-se alguns parâmetros:

- a) deve ter uma interface amigável e simples para interação com o software;
- b) deve ser bem apresentável, com cores marcantes e boa construção;
- c) deve mostrar informações relevantes sobre o sistema - aplicação, software e sistema (Linux e *RPi*);
- d) deve ser construído em um único pacote, não muito grande nem muito pequeno;

Seguindo esses parâmetros, a primeira busca foi por um pacote de onde alocar todos os componentes necessários, de forma que tudo ficasse bem protegido. Para tal, escolheu-se uma caixa de metal, derivada de uma antiga caixa de cigarros, apresentado na subseção 4.2.5.

O material foi cortado primeiramente na parte de baixo de forma que acomode o *RPi* deixando todas as conexões disponíveis para futuro desenvolvimento e facilidade de acessar as portas. Foram realizados furos na parte do fundo para fixação do *RPi* com parafusos, arruelas e porcas, conforme Figura 22.

Dessa maneira o *RPi* ficou bem preso na parte de baixo, liberando a tampa para instalação da interface com o usuário.

Figura 22 – *RPi* preso na caixa, com as portas acessíveis



Fonte: elaborado pelo autor

No próximo passo definiu-se como seria a interface com o usuário. Para que os parâmetros fossem cumpridos, os seguintes itens necessários foram elencados:

- a) apresentará uma informação se o sistema está rodando a aplicação corretamente - para tal definiu-se o uso de um LED verde;
- b) apresentará uma informação se o sistema tem conexão com a internet - para tal definiu-se o uso de um LED amarelo;
- c) apresentará uma informação se um beacon está no alcance - para tal definiu-se o uso de um LED azul;
- d) apresentará várias informações textuais sobre o estado atual do sistema, assim como beacons no alcance, histórico, dados do Linux, endereço de IP, temperatura da CPU - para tal, definiu-se o uso de um display LCD de 16 colunas por 2 linhas.

Nota: A cor dos LEDs foi definido por praticidade e diversificação.

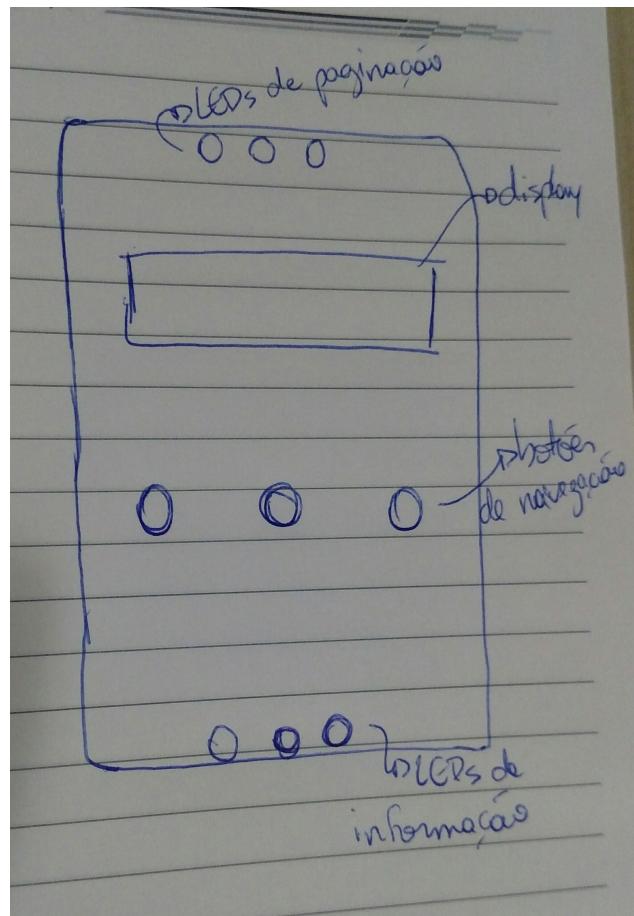
Como o display é pequeno e não apresentaria todas as informações de uma só vez, decidiu-se que existiria alguma forma de navegação por páginas, de forma que as informações fossem alteradas conforme mudasse de página. Para que a navegação fosse realizada de maneira agradável e apresentável, os seguintes itens foram escolhidos:

- a) LEDs indicam qual página o usuário está - definiu-se pelo uso de três LEDs verde, a cor foi escolhida somente por praticidade de já ter os componentes;
- b) botões do tipo clique foram usados para navegar a direita, esquerda, e entre as páginas, similar ao rolar a página.

Seriam utilizado mais LEDs para mais informações serem apresentadas, como mais páginas no display, porém por limitação da quantidade de GPIOs preferiu-se por manter somente três páginas e LEDs.

Com os componentes bem definidos, em seguida a interface foi desenhada no papel, conforme Figura 23.

Figura 23 – RPi posicionado de outra maneira

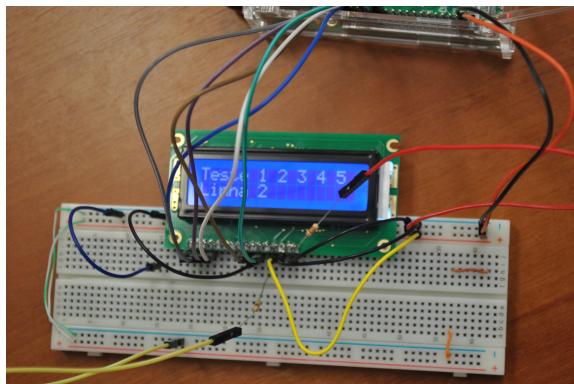


Fonte: elaborado pelo autor

Definido os componentes que fariam parte da interface, foi necessário realizar testes com os LEDs, display LCD e botões na protoboard de forma que todos esses componentes pudessem ser conectados ao mesmo tempo.

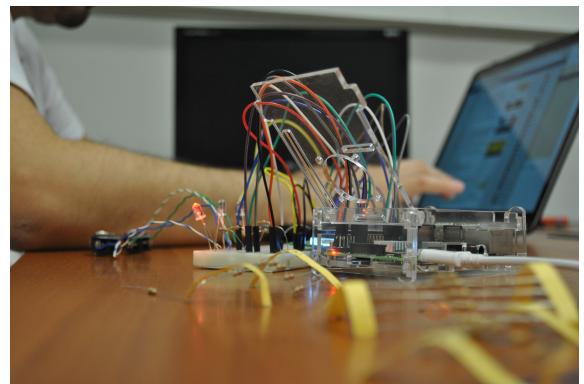
Nesse momento definiu-se o uso de Node.js (detalhado na subseção 4.3.2) para o desenvolvimento, pois foram encontrados diversos tutoriais^{1,2,3} e bibliotecas^{4,5} que facilitariam a conexão com todos os componentes.

Figura 24 – Estudo de conexão e programação com display LCD



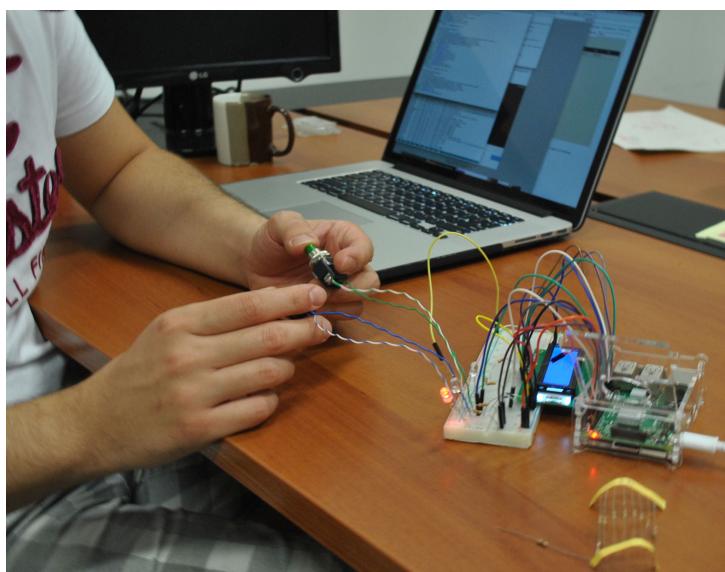
Fonte: elaborado pelo autor

Figura 25 – Conexão com LEDs, display e botão na protoboard



Fonte: elaborado pelo autor

Figura 26 – Teste de mesa com todos os componentes



Fonte: elaborado pelo autor

¹ <<http://thejackalofjavascript.com/rpi-16x2-lcd-print-stuff/>>

² <<http://odesenvolvedor.andafter.org/publicacoes/controlando-gpio-do-raspberry-pi-com-nodejs.html>>

³ <<https://learn.adafruit.com/node-embedded-development/why-node-dot-js>>

⁴ <<https://www.npmjs.com/package/lcd>>

⁵ <<https://www.npmjs.com/package/onoff>>

Após realizar todos os estudos e testes de conexão com sucesso, o próximo passo foi a montagem da interface na caixa de metal. Primeiro os cortes foram feitos com uma ferramenta específica tipo esmeril, em seguida a caixa foi pintada com tinta preta fosca (Figura 27 e Figura 28) para que o protótipo ficasse com boa construção, cor e amigável.

Figura 27 – Pintura da caixa de metal - tampa



Fonte: elaborado pelo autor

Figura 28 – Pintura da caixa de metal - base



Fonte: elaborado pelo autor

Enquanto a tinta secava foi necessário realizar soldas e montagem dos outros componentes em placas para que tudo ficasse organizado e fácil de ser trocado, caso necessário. O rascunho de conexão das GPIOs com os componentes de hardware estão marcados na Figura 29.

Cada LED foi conectado a uma GPIO, a um resistor de 330Ω , e ao GND. Os botões foram conectados seguindo o modelo da Figura 30, com um resistor de $10K\Omega$ e uma GPIO para cada botão. O display foi conectado diretamente a cada GPIO marcada, e também ao +5V e GND para alimentação.

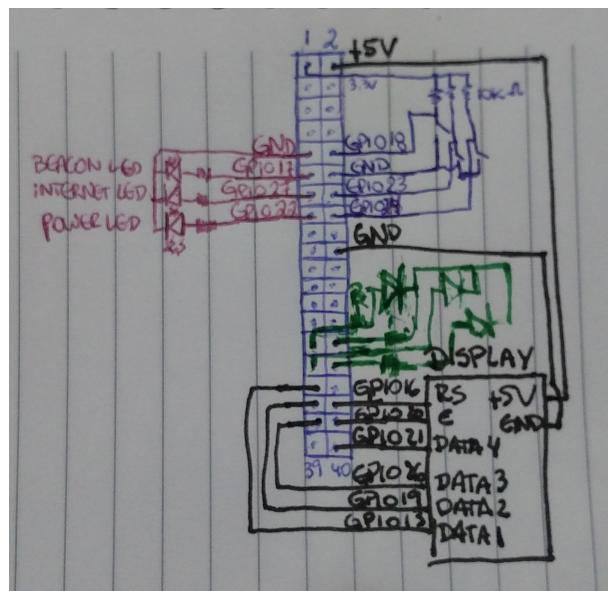
O modelo de display LCD utilizado também tem um pino que utiliza voltagem variável para iluminar cada caractere (contraste de cada letra), e um pino para a iluminação do fundo (azul). Para essas conexões foram utilizados potenciômetros de 10K, para facilitar mudanças caso necessário. Dessa forma, só é necessário girar o pino do potenciômetro para alterar a saída.

Foram utilizados seis GPIOs para os LEDs, três para os botões e seis para o display, totalizando 15 GPIOs de 17 disponíveis. As GPIOs utilizadas foram:

- Para os LEDs de interface (topo), GPIO 12, 6 e 5, respectivamente 1, 2 e 3;
- Para os LEDs de informação (baixo), GPIO 22, 27 e 17, respectivamente Power, Internet e Beacon;
- Para os botões, GPIOs 24, 23 e 18, respectivamente botão esquerdo, central e direito;
- Para o display de LCD, GPIOs 16, 20, 13, 19, 26, 21, respectivamente Register Select, Enable, Data 1, Data 2, Data 3 e Data 4.

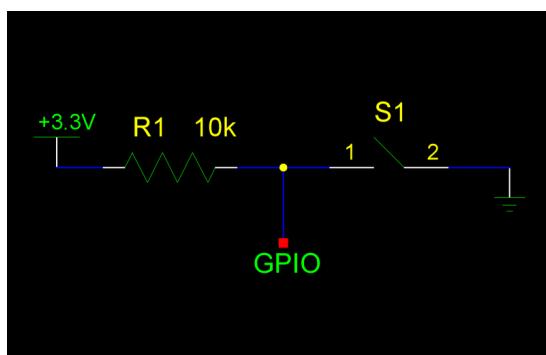
Nota: Os números das GPIOs foram escolhidos devido a proximidade e agrupamento dos componentes.

Figura 29 – Rascunho de conexão das GPIOs



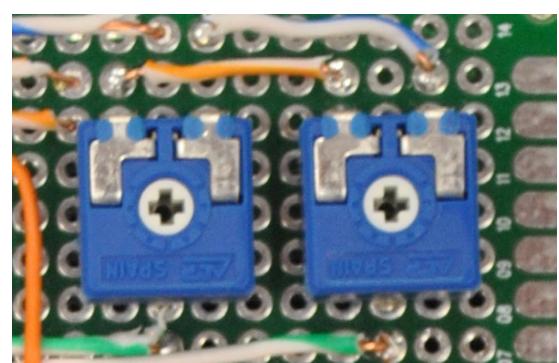
Fonte: elaborado pelo autor

Figura 30 – Modelo de conexão dos botões



Fonte: (SKLAR, 2012)

Figura 31 – Potenciômetros de configuração do display LCD



Fonte: elaborado pelo autor

De forma que a tampa saísse facilmente optou-se por usar um cabo flat frequentemente usado em HDs antigos (padrão IDE) conectado ao *RPi* e a uma placa central com duas fileiras de 20 pinos.

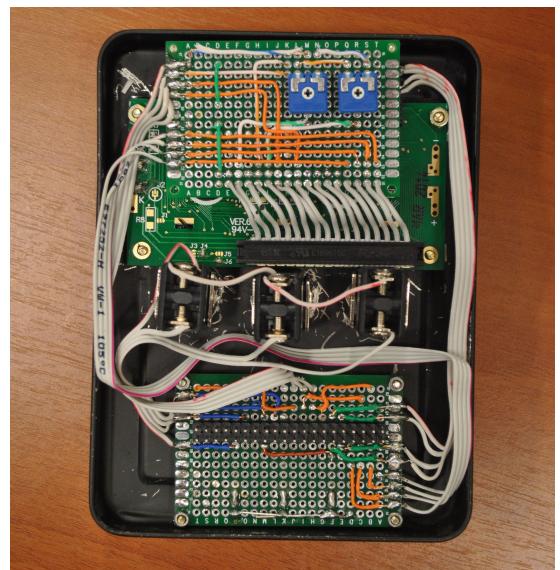
Utilizando os fios internos de cabo de rede para conexão entre os componentes nas placas, o mesmo cabo flat cortado para conexão entre as placas para uma maior flexibilidade e facilidade de mudanças, criou-se o controle da interface, como pode ser visto na Figura 32 e Figura 33.

Figura 32 – Parte interna da interface - parcialmente desmontada



Fonte: elaborado pelo autor

Figura 33 – Parte interna da interface - completamente montada



Fonte: elaborado pelo autor

As placas internas da interface foram afixadas com cola quente, para que os cabos fossem reforçados e não quebrasse nenhuma conexão. Tampa e base foram conectados com o cabo flat (Figura 34) e o protótipo ficou pronto para desenvolvimento da aplicação.

Após o hardware estar totalmente montado, montou-se a estrutura de informações que iriam aparecer nas telas, conforme Figura 36. As telas apresentam as seguintes informações:

- A primeira tela mostra informações sobre os beacons no alcance, ou caso não haja nenhum, apresenta o texto "Aguardando beacons". As subtelas apresentam informações sobre quais beacons estão no alcance, sendo o nome atribuídos a eles ou então o texto "Desconhecido" caso não esteja no banco de dados. A quantidade de subtelas depende da quantidade de beacons no alcance, por isso não se sabe ao certo a quantidade total;
- A segunda tela apresenta um histórico dos beacons encontrados durante a execução do programa, com um contador da quantidade de beacons no histórico, e

Figura 34 – Conexão da tampa com a base



Fonte: elaborado pelo autor

Figura 35 – Criação e montagem do protótipo finalizada



Fonte: elaborado pelo autor

as subtelas apresentam o nome do beacon encontrado em ordem numérica. Se um beacon for encontrado diversas vezes, será repetido nas subtelas. Conforme a primeira tela, a quantidade de subtelas depende da quantidade de beacons no histórico, por isso não se sabe ao certo a quantidade total;

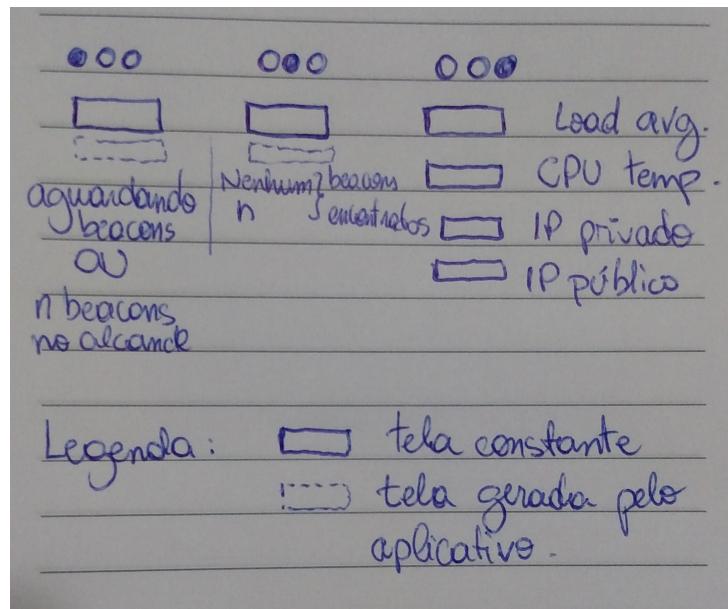
- c) A terceira tela apresenta informações relevantes sobre o sistema, sendo elas:
 - Primeira subtela apresenta o *load average*⁶ do sistema Linux. É relevante para saber se o sistema está muito carregado ou trabalhando tranquilamente;
 - Segunda subtela apresenta a temperatura do processador. É relevante para saber se o sistema está trabalhando a uma temperatura segura ou está sobreaquecendo;
 - Terceira subtela apresenta o endereço de IP privado da rede. É de extrema importância para conexão remota, pois é esse endereço que é utilizado para

⁶ Segundo Passos (2012), essa informação de três números apresenta a média de carga da CPU por um certo tempo. O primeiro número é a média no último minuto, o segundo número é a média nos últimos cinco minutos, e o terceiro número é a média nos últimos quinze minutos

conexão via SSH;

- Quarta subtela apresenta o endereço de IP público da rede. É relevante para saber se está conectado direto a internet (o IP privado será igual ao IP público) ou em uma subrede, como de uma residência ou do laboratório.

Figura 36 – Definição das informações apresentadas nas telas



Fonte: elaborado pelo autor

Após todos esses passos o protótipo estava bem definido para o desenvolvimento.

5.2.1 Problemas Enfrentados

Durante os estudos e testes dos LEDs na protoboard utilizou-se resistores de 330Ω para conexão correta. Quanto maior o valor da resistência, menos brilho o LED terá. Como os componentes estavam na protoboard, não percebeu-se que os LEDs utilizados eram de alto brilho.

Para soldagem final dos componentes, preferiu-se por manter os resistores de 330Ω por questões práticas. Porém os LEDs ficaram com muito brilho, incomodando qualquer pessoa que olhasse diretamente para o protótipo, conforme Figura 37 e Figura 38.

Como os componentes já estavam afixados com cola quente, substituir os resistores levaria muito tempo e possivelmente atrasaria o desenvolvimento do projeto, portanto optou-se por manter dessa maneira. Para um futuro protótipo, resistores maiores, ou LEDs com menos brilho seriam essenciais.

Figura 37 – LED de alto brilho ao olhar diretamente



Fonte: elaborado pelo autor

Figura 38 – LEDs de alto brilho refletindo na mesa



Fonte: elaborado pelo autor

5.3 Terceira Etapa

Na terceira etapa realizou-se o desenvolvimento da aplicação, testes e aquisição de dados. A aplicação foi dividida em módulos por maior facilidade de desenvolvimento, testes e descoberta de erros. Um arquivo de configurações globais, nomeado `globals-config.js`, foi utilizado para todas as definições de constantes de forma a facilitar futuras mudanças, caso necessário.

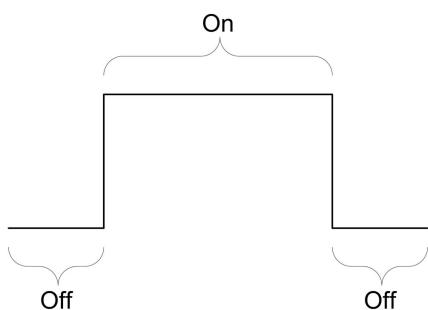
5.3.1 Primeiro Módulo

O primeiro módulo desenvolvido foi a interface, que consiste da conexão com LEDs, botões e display LCD. Optou-se por criar uma biblioteca auxiliar para abstrair métodos repetitivos e também irrelevantes para os outros módulos. Como esse módulo é o meio de acesso do código principal aos componentes de hardware, foram adicionados trechos de teste para acender o LED e apagar o LED pela linha de comando, de forma que os testes de funcionalidade pudessem ser realizados.

Durante a fase de testes do primeiro módulo, um fato interessante ocorreu. Ao clicar uma única vez em um botão, em algumas vezes o código executava dois, três ou até mais cliques de botão. Pesquisando mais a fundo sobre esse erro, uma informação nova surgiu.

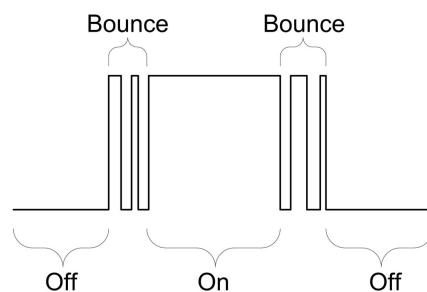
Botões mecânicos não criam ou perdem o contato corretamente, conforme caso ideal e perfeito representado no Gráfico 39. Em vez disso, podem oscilar rapidamente no momento do clique ou ao soltar o botão, conforme Gráfico 40. (PROTOSTACK, 2010). Esse efeito é conhecido como *bounce*, ou ruído.

Figura 39 – Gráfico ideal de um clique de botão



Fonte: (PROTOSTACK, 2010)

Figura 40 – Gráfico real de um clique de botão



Fonte: (PROTOSTACK, 2010)

A solução para resolver esse problema (denominada *debounce*) foi de, em seguida a um clique, bloquear todos os próximos cliques via programação por determinado tempo (em milissegundos), de forma que não interferisse diretamente nos cliques reais do usuário. Esse tempo foi determinado por testes repetitivos e de diferentes usuários em 200ms. Esse valor foi considerado ideal por evitar o ruído e não interferir em cliques seguidos do usuário. Valores maiores, como 300ms afetaram o clique do usuário, e valores inferiores como 100ms não evitaram totalmente o ruído.

5.3.2 Segundo Módulo

Páginas e Navegação entre telas

5.3.3 Terceiro Módulo

Identificação dos Beacons e apresentação nas telas corretas

5.3.4 Quarto Módulo

Procurar beacon no banco de dados local

5.3.5 Quinto Módulo

Salvar beacons encontrados no banco de dados local

5.3.6 Protótipo Final

Mostrar logo e falar sobre protótipo final

Figura 41 – Protótipo finalizado e totalmente funcional



Fonte: elaborado pelo autor

6 CONCLUSÃO

FALTA ACERTAR!!!! Esse projeto tem como objetivo o aprofundamento na área de Internet das Coisas, pelo estudo das tecnologias de *Bluetooth Low Energy*, *Raspberry Pi* e *beacons*. O resultado final será um protótipo de rastreador de *beacons* utilizando o *Raspberry Pi*, ainda em desenvolvimento.

A fundamentação teórica realizada por meio de pesquisa bibliográfica no início do projeto foi essencial para entender como os protocolos *beacons* foram propostas e implementadas, como é o comportamento do *Raspberry Pi*, entre outros. Em conjunto, o estudo das tecnologias foi de extrema importância para entender o funcionamento e utilização do *RPi* e *beacon*.

Certas atividades de extrema importancia ainda serão realizadas, mas o projeto ainda tem muitos frutos para render, caminhando em uma boa direção. A área de Internet das Coisas é uma área extremamente promissora, com áreas de pesquisa e implementação bem vastas.

REFERÊNCIAS

- APPLE INC. *Getting Started with iBeacon*. <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>, 2014.
- ARGENOX. *A BLE Advertising Primer*. <http://www.argenox.com/bluetooth-low-energy-ble-v4-0-development/library/a-ble-advertising-primer/>, 2015.
- ASHTON, K. *That 'Internet of Things' Thing: In the real world, things matter more than ideas*. <http://www.rfidjournal.com/articles/view?4986>, 2009.
- AUSTIN. *Understanding the different types of BLE Beacons*. <https://developer.mbed.org/blog/entry/BLE-Beacons-URIBeacon-AltBeacons-iBeacon/>, 2015.
- BEN. *What is an Arduino?* <https://learn.sparkfun.com/tutorials/what-is-an-arduino>, 2015.
- BLUETOOTH SIG. *Bluetooth Smart Technology: Powering the Internet of Things*. <http://www.bluetooth.com/Pages/Bluetooth-Smart.aspx>, 2015.
- BLUETOOTH SIG. *Generic Access Profile*. <https://www.bluetooth.org/en-us/specification/assigned-numbers/generic-access-profile>, 2015.
- CANONICAL LTD. *Snappy Ubuntu*. <https://developer.ubuntu.com/en/snappy/>, 2015.
- CHACON, B. S. S. *Pro Git*. 2. ed. <https://git-scm.com/book/en/v2>: Apress, 2014.
- GOLDMAN SACHS. *What is the Internet of Things?* <http://www.goldmansachs.com/our-thinking/outlook/iot-infographic.html>, 2014.
- GROTHAUS, M. *How Apple Thinks About Smart Homes*. <http://www.fastcolabs.com/3034919/how-apple-thinks-about-smart-homes>, 2014.
- GUBBI, J. et al. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, <http://www.sciencedirect.com/science/article/pii/S0167739X13000241>, v. 29, n. 7, p. 1645–1660, Sept 2013.
- JJNEBEKER. *Can RaspberryPi with BLE Dongle detect iBeacons?* <http://stackoverflow.com/questions/21733228/can-raspberrypi-with-ble-dongle-detect-ibeacons>, 2014.
- KASTRENAKES, J. *Macy's begins iBeacon shopping test, will send alerts to your iPhone when you enter stores*. <http://www.theverge.com/2013/11/21/5129336/macys-apple-ibeacon-support-herald-union-stores-shopkick>, 2013.
- MAKER SHED. *Raspberry Pi Comparison Chart*. <http://www.makershed.com/pages/raspberry-pi-comparison-chart>, 2015.
- MICROSOFT. *A Internet das suas coisas*. <https://dev.windows.com/pt-br/iot>, 2015.

- NASCIMENTO, R. *O que, de fato, é internet das coisas e que revolução ela pode trazer?* <http://computerworld.com.br/negocios/2015/03/12/o-que-de-fato-e-internet-das-coisas-e-que-revolucao-ela-pode-trazer>, 2015.
- PASSOS, T. *O que significa o load average, do comando top, no Linux?* <http://blog.tiagopassos.com/2012/09/21/o-que-significa-o-load-average-do-comando-top-no-linux/>, 2012.
- PRESS, G. *It's Official: The Internet Of Things Takes Over Big Data As The Most Hyped Technology.* <http://www.forbes.com/sites/gilpress/2014/08/18/its-official-the-internet-of-things-takes-over-big-data-as-the-most-hyped-technology/>, 2014.
- PROTOSTACK. *Debouncing a switch.* <http://www.protostack.com/blog/2010/03/debouncing-a-switch/>, 2010.
- RASPBERRY PI. *Downloads.* <https://www.raspberrypi.org/downloads/>, 2015.
- RASPBERRY PI. *What is a Raspberry Pi?* <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>, 2015.
- SKLAR, B. B. M. *Bread Board Setup for Input Buttons.* <https://learn.adafruit.com/playing-sounds-and-using-buttons-with-raspberry-pi/bread-board-setup-for-input-buttons>, 2012.
- SORREL, C. *Just What Is An Arduino, And Why Do you Want One?* <http://www.wired.com/2008/04/just-what-is-an/>, 2008.
- TEIXEIRA, F. *Tudo o que você precisa saber para começar a brincar com iBeacons.* <http://arquiteturadeinformacao.com/ux-em-espacos-fisicos/tudo-o-que-voce-precisa-saber-para-comecar-a-brincar-com-ibeacons/>, 2014.
- THIBODEAU, P. *Um em cada cinco desenvolvedores já trabalha em projetos de IoT.* <http://computerworld.com.br/um-em-cada-cinco-desenvolvedores-ja-trabalham-em-projetos-de-iot>, 2015.
- UBUNTU MATE. *About Ubuntu Mate.* <https://ubuntu-mate.org/about/>, 2015.
- WANDSCHNEIDER NIRDHAR KHAZANIE, M. A. M. *Eddystone Protocol Specification.* <https://github.com/google/eddystone/blob/master/protocol-specification.md>, 2015.

APÊNDICE A – LATEX

Para o desenvolvimento dessa monografia utilizou-se uma ferramenta denominada TeX, um processador de texto baseado em comandos e macros. Normalmente é grafado usando a macro \TeX, resultando em \TeX.

Não é comum o uso do \TeX por si só. Utiliza-se uma outra ferramenta denominada LaTeX (grafada \LaTeX), uma série de macros definidas para auxiliar o desenvolvimento.

Essa abordagem é muito utilizada no meio acadêmico e científico, pois o ponto mais importante dessa ferramenta é o conteúdo. O design final ficará sempre para o compilador gerar. Isso evita erros de formatação, como por exemplo devido a diferenças entre versões de programas.

Congressos e revistas científicas já disponibilizam arquivos no formato .sty¹ para que os pesquisadores enviem os artigos preenchidos em .pdf formatado conforme modelo próprio. Dessa forma todos os artigos seguirão o mesmo formato, auxiliando também os pesquisadores a escreverem os documentos rapidamente.

Por ser um documento de texto no formato .tex, qualquer editor comum como o Bloco de Notas do Windows pode ser utilizado. Dessa forma a compilação é feita por meio da linha de comando, com algumas opções de compiladores. As mais utilizadas são pdflatex e xelatex.

O meio mais utilizado para escrita de documentos \LaTeX de forma a facilitar a compilação são programas próprios. Existem diversas opções, mas as mais utilizadas são MiKTeX² para Windows e MacTeX³ para Mac OS X. O programa utilizado para esse documento foi o Texworks, um editor de texto para \TeX multiplataforma e de código aberto, conforme imagem 42.

A.1 abnTeX2

Devido a esse Trabalho de Conclusão de Curso seguir as normas ABNT⁴ para escrita e formatação de documentos científicos, utilizou-se a biblioteca de macros abnTeX2 (abnTeX2)⁵ para formatação do documento conforme as normas vigentes. Essa biblioteca é livre, com código fonte aberto e disponível para qualquer um auxiliar.

¹ Modelo de estilos \TeX.

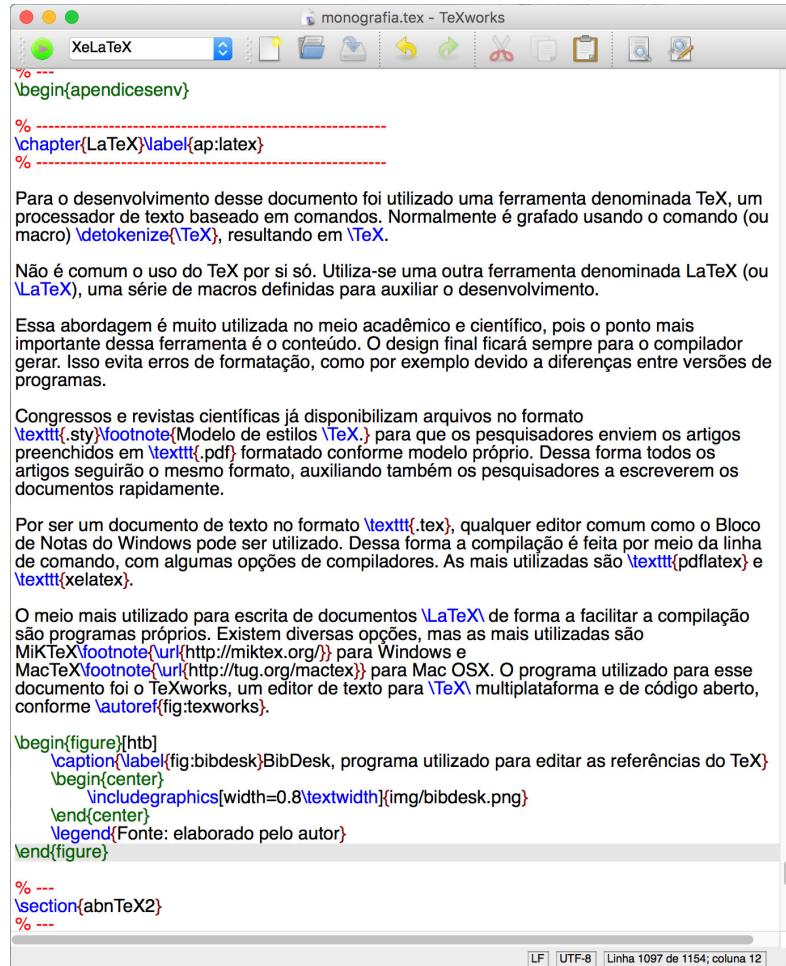
² <<http://miktex.org/>>

³ <<http://tug.org/mactex>>

⁴ Associação Brasileira de Normas Técnicas.

⁵ <<https://github.com/abntex/abntex2>>

Figura 42 – TeXworks, programa utilizado para escrever o código \TeX



Fonte: elaborado pelo autor

Uma ferramenta muito útil disponível com o abn $\text{\TeX}2$ é a criação das Referências Bibliográficas no formato ABNT de forma automática. Para tal, é criado um arquivo .bib com todas as informações de citações e referências utilizadas no texto.

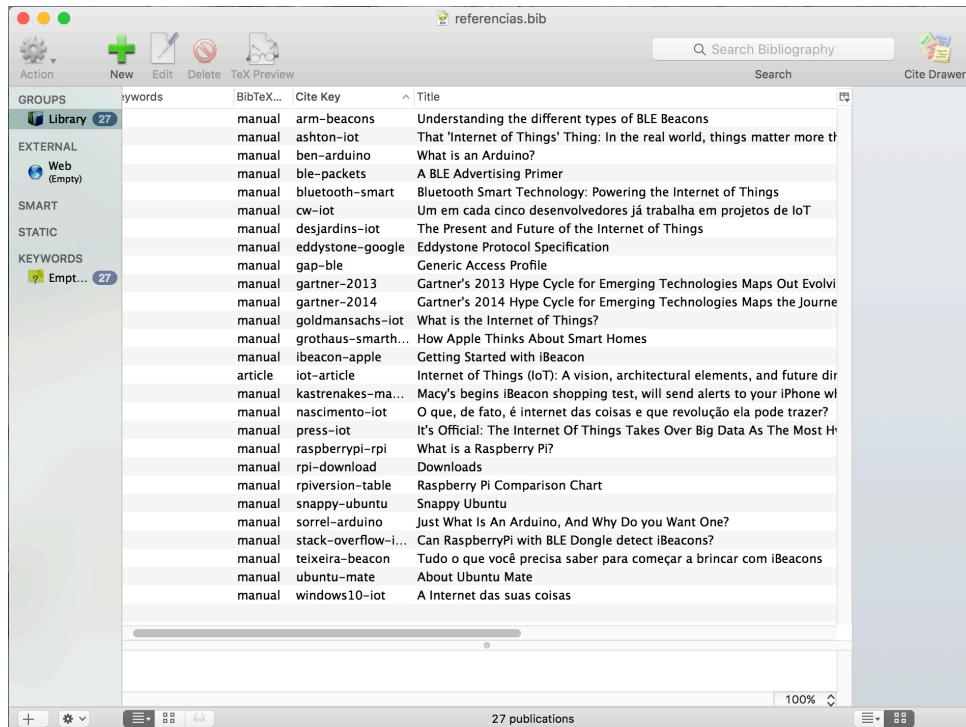
As chaves de citação são os valores que serão utilizados nas macros do abn $\text{\TeX}2$ para que as referências sejam corretamente configuradas. Por exemplo, se desejamos referenciar um artigo de Microsoft (2015), adicionamos todas as informações como nome dos autores, organização, data, edição, entre outros, e adicionamos a chave windows10-iot.

Utilizando a macro \cite {windows10-iot}, no local ficará o texto (MICROSOFT, 2015), e nas referências será adicionado conforme modelo ABNT: MICROSOFT. A Internet das suas coisas. <https://dev.windows.com/pt-br/iot>, 2015. Para citações *inline*, ou na mesma linha, o comando \citeonline {windows10-iot} é utilizado, gerando o texto Microsoft (2015).

O arquivo .bib pode ser criado manualmente ou por meio de programas. Para

esse artigo foi utilizado o programa BibDesk⁶ para Mac OSX, conforme Figura 43. Esse programa já cria o arquivo no formato correto com as chaves de citação configuradas.

Figura 43 – BibDesk, programa utilizado para editar as referências do \TeX



Fonte: elaborado pelo autor

Além de formatar o documento, abn \TeX 2 também disponibiliza macros para criação de conteúdo citados na ABNT. Por exemplo, tabelas que seguem o formato do IBGE⁷ podem ser facilmente criadas com macros disponíveis, como exemplo da Tabela 4.

Tabela 4 – Exemplo de tabela gerada com o \TeX

	BLE	Classic
Modulação	GFSK 0.45 a 0.55	GFSK 0.28 a 0.35
Velocidade de Transferência	1 Mbit/s	1 Mbit/s
Canais	40	79
Espaçamento	2 MHz	1 MHz

Fonte: (ARGENOX, 2015)

O código para geração da Tabela 4, conforme a documentação da abn \TeX 2 é:

⁶ <<http://bibdesk.sourceforge.net/>>

⁷ Instituto Brasileiro de Geografia e Estatística

```
\begin{table}[htb]
\IBGEtab{
    \caption{\textit{BLE Physical Layer}}
    \label{table:ble-physical}
}{

\begin{tabular}{ccc}
\toprule
& BLE & Classic \\
\midrule
Modulação & GFSK 0.45 a 0.55 & GFSK 0.28 a 0.35 \\
\midrule
Velocidade de Transferência & 1 Mbit/s & 1 Mbit/s \\
\midrule
Canais & 40 & 79 \\
\midrule
Espaçamento & 2 MHz & 1 MHz \\
\bottomrule
\end{tabular}
}

\fonte{\cite{ble-packets}}
}

\end{table}
```

Diversas razões podem ser elencadas a favor do uso do L^AT_EX e abnT_EX2 nessa monografia, entre elas:

- a) Facilidade de criação do conteúdo: abnT_EX2 já formata o documento conforme as normas ABNT, necessitando somente colocar o conteúdo interno;
- b) Evitar erros de formatação ao alterar o conteúdo do arquivo: muito comum quando utilizado o Microsoft Word, ao alterar o conteúdo de uma seção, as partes subsequentes facilmente perdem a formatação, sendo necessário uma revisão minuciosa para que esses erros sejam evitados;
- c) Continuação dos outros relatórios: como o autor gostaria de aprender essa tecnologia, utilizou para desenvolver a proposta desse trabalho. Portanto, para facilitar, continuou os outros relatórios utilizando a mesma ferramenta;

O código dessa monografia, assim como a versão final em .pdf pode ser acessado pelo repositório no GitHub⁸. Está disponível para livre consulta, tanto de professores quanto de alunos interessados em saber mais sobre o L^AT_EX.

⁸ <<https://github.com/gabrielb Oliveira/Monografia-Pearson>>