

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 Facul-
dade 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

Atualmente a área da Internet das Coisas tem sido foco de muitos estudos e aplicações. 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 recentes, permitindo a comunicação com outros dispositivos 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 teve como objetivo criar um rastreador de *beacons* utilizando o *Raspberry Pi* para expandir a capacidade de identificação. Divido em etapas de estudo e levantamento, planejamento e desenvolvimento, o protótipo foi concebido de componentes de baixo custo, permitindo uma futura utilização em massa. Trabalhou-se com Node.js para o desenvolvimento da aplicação e CouchDB como banco de dados do histórico de identificação dos *beacons*. Testes foram executados, comprovando a eficiência do projeto. O protótipo pode ser facilmente expandido para outras aplicações em várias áreas de atuação.

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

ABSTRACT

Currently the area of the Internet of Things has been the focus of many studies and applications. This area has motivated several people because it makes possible to have an specific product after a short time job. For this reason, some devices were created to assist these areas, including the Arduino and Raspberry Pi. The beacons are also recent, allowing communication with other devices like the RPi to help micro-locating within a small space at a low cost. Its usage focus is small, only with readings via smartphones. This project aimed to create a beacon tracker using the Raspberry Pi to expand its identification capacity. Divided into study steps and survey, planning and development, the prototype was designed to use low-cost components, allowing future mass usage. It was developed with Node.js and with CouchDB database for the beacons identification history. Test were executed, proving efficiency to the project. The prototype can be easily expanded to other applications in various fields.

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	18
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 <i>iBeacon</i>	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 – Conexão com o <i>RPi</i> via <i>SSH</i>	26
Figura 10 – Moto Maxx (esquerda) e iPad Mini (direita)	27
Figura 11 – <i>Beacon</i> Zebra MPact utilizado para testes	27
Figura 12 – <i>Beacon</i> configurado na <i>Toolbox</i> apresentando a porcentagem de bateria	28
Figura 13 – Caixa de metal utilizada para o protótipo	29
Figura 14 – Parte interna do protótipo	29
Figura 15 – Protoboard na etapa de estudo	29
Figura 16 – Pinos físicos de conexão	31
Figura 17 – Numeração dos pinos de conexão	31
Figura 18 – Futon - manipulação dos bancos de dados CouchDB	33
Figura 19 – Primeiro teste realizado, com <i>RPi</i> e <i>beacon MPact</i>	35
Figura 20 – Software <i>hcitool</i> executando	36
Figura 21 – Software <i>hcidump</i> executando	36
Figura 22 – Teste com movimentação do <i>beacon</i>	37
Figura 23 – <i>RPi</i> posicionado de outra maneira	38
Figura 24 – <i>RPi</i> preso na caixa, com as portas acessíveis	39
Figura 25 – Estudo de conexão e programação com display LCD	40
Figura 26 – Conexão com LEDs, display e botão na protoboard	40
Figura 27 – Pintura da caixa de metal - tampa	41
Figura 28 – Pintura da caixa de metal - base	41
Figura 29 – Modelo de conexão dos botões	41
Figura 30 – Potenciômetros de configuração do display LCD	41
Figura 31 – Circuito de conexão das GPIOs	42
Figura 32 – Parte interna da interface - parcialmente desmontada	43
Figura 33 – Parte interna da interface - completamente montada	43
Figura 34 – Conexão da tampa com a base	44
Figura 35 – Criação e montagem do protótipo finalizada	44

Figura 36 – Logotipo criado para o protótipo	45
Figura 37 – Logotipo menor criado para o protótipo	45
Figura 38 – LED de alto brilho ao olhar diretamente	46
Figura 39 – LEDs de alto brilho refletindo na mesa	46
Figura 40 – Gráfico ideal de um clique de botão	47
Figura 41 – Gráfico real de um clique de botão	47
Figura 42 – Protótipo finalizado e totalmente funcional	51
Figura 43 – TeXworks, programa utilizado para escrever o código T _E X	56
Figura 44 – BibDesk, programa utilizado para editar as referências do T _E X	57

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	57

LISTA DE ABREVIATURAS E SIGLAS

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

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	<i>Beacon</i>	20
3.4.1	<i>iBeacon</i>	21
3.4.2	Outros Protocolos	22
4	METODOLOGIA	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	<i>Beacons</i>	27
4.2.5	Acessórios para o Protótipo	28
4.3	Tecnologias e Ferramentas	30
4.3.1	Raspberry Pi	30
4.3.2	Node.js e npm	31
4.3.3	CouchDB	32
4.3.4	Git	33
4.3.5	Outras Ferramentas	34
5	DESENVOLVIMENTO	35
5.1	Primeira Etapa - Estudo Prático das Tecnologias	35
5.2	Segunda Etapa - Planejamento do Protótipo	38
5.2.1	Logotipo e Nome para o Protótipo	45
5.2.2	Problemas Enfrentados	45
5.3	Terceira Etapa - Desenvolvimento e Testes da Aplicação	46
5.3.1	Primeiro Módulo - Biblioteca Auxiliar	46

5.3.2	Segundo Módulo - Páginas e Navegação	47
5.3.3	Terceiro Módulo - Identificação dos <i>Beacons</i>	47
5.3.3.1	Problemas Enfrentados	48
5.3.4	Quarto Módulo - Busca no Banco de Dados	48
5.3.5	Quinto Módulo - Salvar no Banco de Dados	49
5.3.6	Testes e Resultados	49
5.3.7	Protótipo Final	50
6	CONCLUSÃO	52
	REFERÊNCIAS	53
	APÊNDICE A – LATEX	55
A.1	abnTeX2	55

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 FOUNDATION, 2015). 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 internet cabeada (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 Frequência) 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.

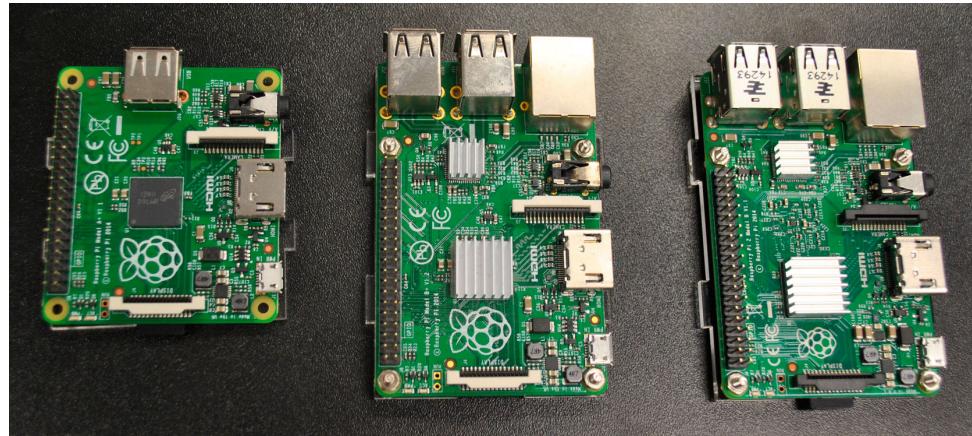
3.2 Raspberry Pi

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 1 e Figura 1.

Segundo Raspberry Pi Foundation (2015), esse pequeno dispositivo foi criado 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
Ethernet	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, 2015);
- 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).

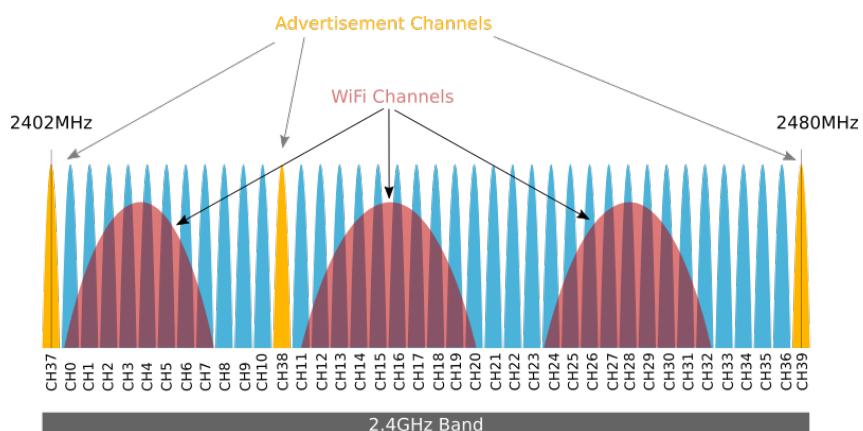
Tabela 2 – *BLE Physical Layer*

	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).

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



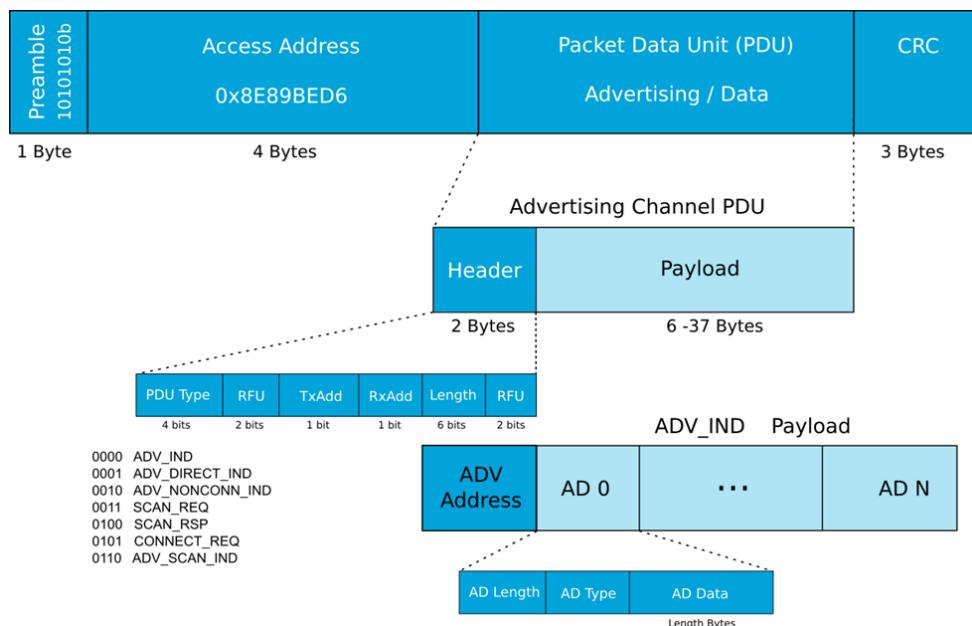
Fonte: (ARGENOX, 2015)

Interessante notar que esses canais estão posicionados em forma bastante estratégica, no começo, final e meio da banda de 2.4 GHz, com a finalidade de 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).

Figura 3 – Modelo de *BLE Advertising Packet*.



Fonte: (ARGENOX, 2015)

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

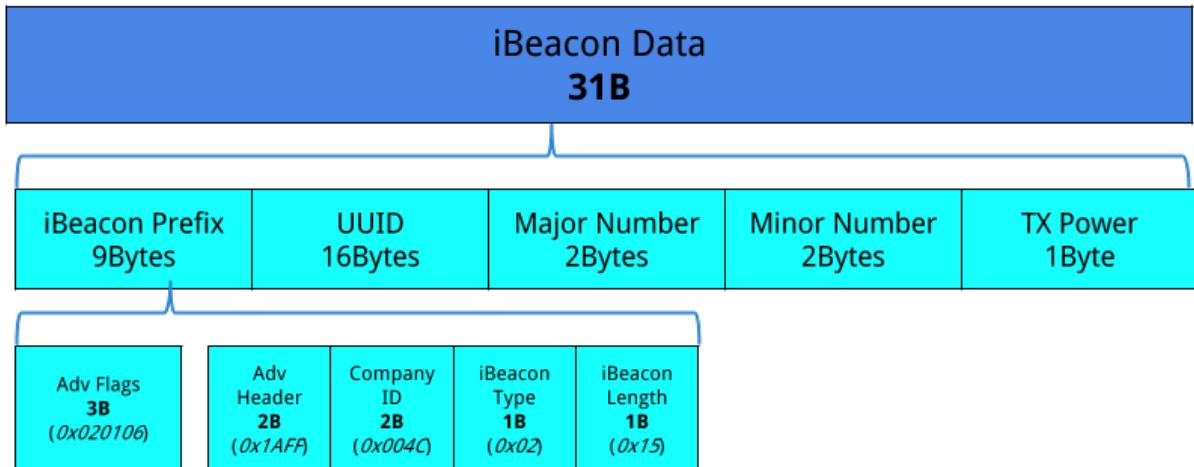
Tabela 3 – Exemplo de aplicação

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

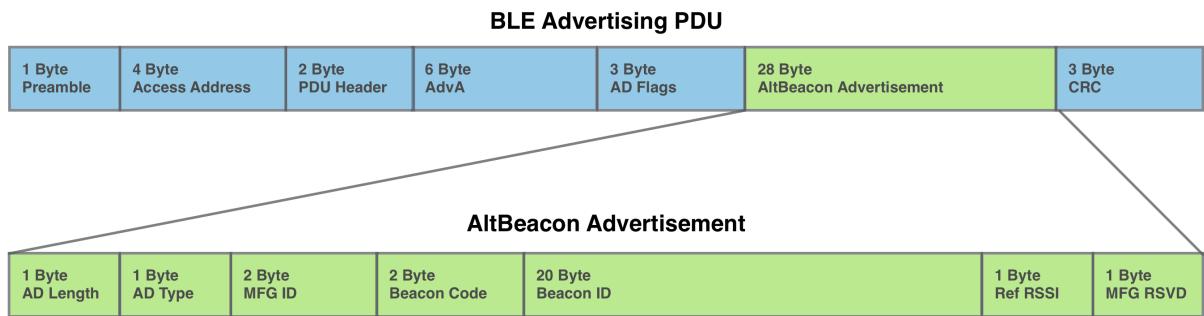
Existem mais protocolos baseados na tecnologia *beacon*. Dois se destacam por ser abertos e passíveis de alterações: *AltBeacon* e *Eddystone*, este último criado pela Google. O *AltBeacon* possui um modelo bastante parecido com o *iBeacon*, porém com possibilidade de alterar o número do fabricante, 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:

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

- 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.

Figura 6 – Pacote do *AltBeacon*.



Fonte: (AUSTIN, 2015)

4 METODOLOGIA

4.1 Métodos e Etapas

O projeto foi dividido em três 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 prático 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 divididas em módulos para facilitar os testes e detectar possíveis erros e *bugs*.

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.

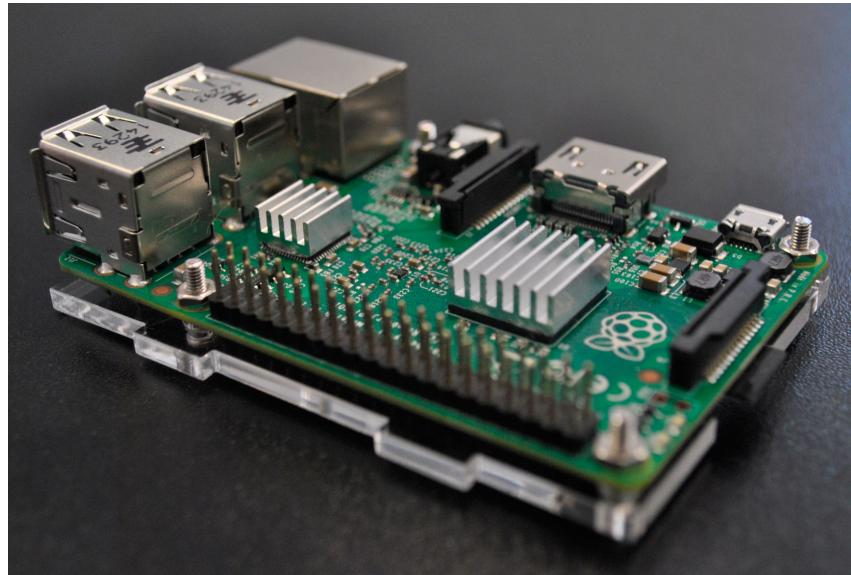
O sistema operacional executado no RPi foi instalado em um cartão microSD, com no mínimo 4 GB de espaço. O cartão utilizado nesse projeto foi um SanDisk Ultra Class 10 de 8 GB.

O sistema utilizado foi o Raspbian Wheezy. Durante o desenvolvimento do projeto, a versão Raspbian Jessie foi lançada, com inúmeras melhorias, porém preferiu-se por

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

permanecer na versão Wheezy para evitar possíveis incompatibilidades com os softwares utilizados.

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

Na seção 4.3 - Tecnologias e Ferramentas, subseção 4.3.1 - Raspberry Pi aborda-se 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.

Nesse sistema o uso de *SSH* é simples, basta abrir o aplicativo Terminal e utilizar o comando "*ssh usuario@computador*", conforme Figura 9. Esse tipo de abordagem é bastante utilizada para conexão a servidores na nuvem, para executar comandos, softwares, entre outros.

Figura 9 – 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 10.

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.

² Secure Shell

Figura 10 – Moto Maxx (esquerda) e iPad Mini (direita).

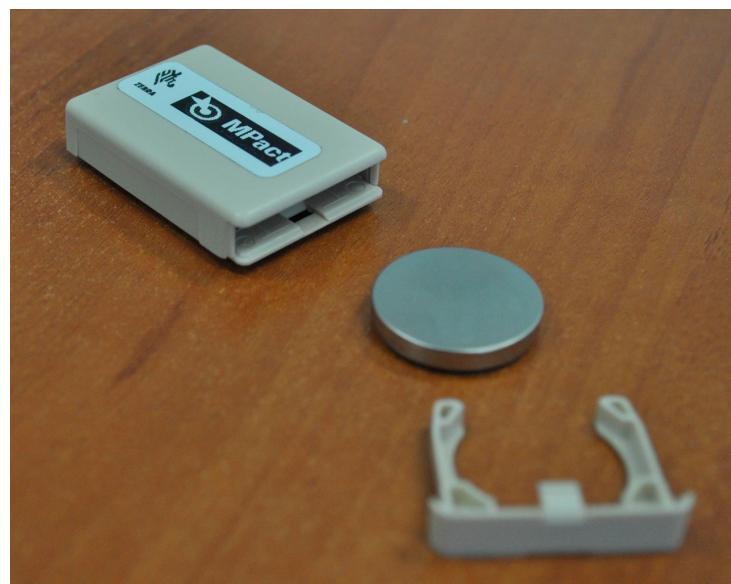


Fonte: elaborado pelo autor

4.2.4 Beacons

O *beacon* utilizado para testes foi o *Zebra MPact*, conforme Figura 11. Utiliza uma bateria CR2450 para alimentação de energia.

Figura 11 – *Beacon Zebra MPact* utilizado para testes.



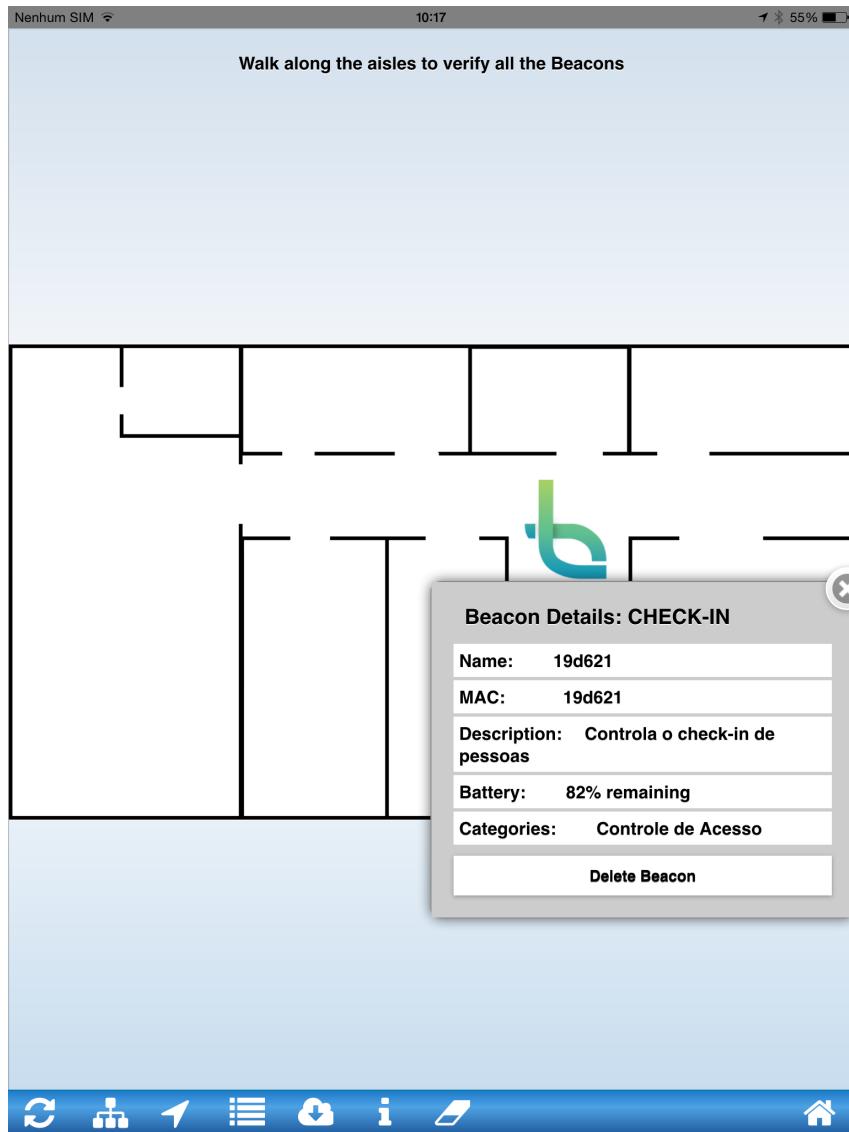
Fonte: elaborado pelo autor

A empresa *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 neste projeto somente para conhecimento da tecnologia *beacon* e atualização do *firmware*, disponível somente via *Toolbox*. Esse software também apresenta a porcentagem de bateria, conforme Figura 12.

Permite a configuração de vários *beacons* simultaneamente, além de alterar o modo de funcionamento, de *iBeacon* para *MPact*, protocolo criado pela fabricante.

Figura 12 – *Beacon* configurado na *Toolbox* apresentando a porcentagem de bateria.



Fonte: elaborado pelo autor

4.2.5 Acessórios para o Protótipo

O protótipo foi adaptado em uma caixa de metal (Figura 13), furada conforme projeto e pintada de preto. Esse processo está detalhado na seção 5.2.

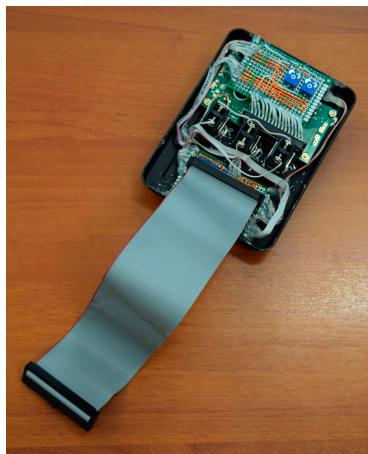
Figura 13 – Caixa de metal utilizada para o protótipo



Fonte: elaborado pelo autor

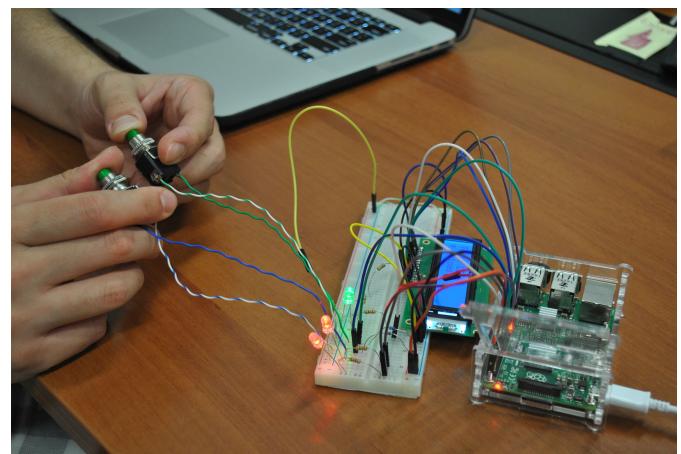
Para o protótipo final foram utilizados seis LEDs, resistores, barra de pinos, um display azul de 16 colunas por 2 linhas, três botões do tipo clique, cabos *flat* do tipo IDE e *floppy*, placas com furação para soldagem de componentes, cabos finos removidos de um cabo de rede *ethernet* para ligações entre os componentes e cola quente para fixação dos componentes internos. Esses materiais podem ser vistos na Figura 14.

Figura 14 – Parte interna do protótipo



Fonte: elaborado pelo autor

Figura 15 – Protoboard na etapa de estudo



Fonte: elaborado pelo autor

Durante a fase de estudo das tecnologias utilizou-se uma protoboard e fios de conexão (Figura 15) 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) um pino de saída digital (GPIO) para acender ou apagar cada LED;
- b) um pino de entrada digital para leitura do estado de cada botão;
- c) seis pinos de saída no total, sendo quatro de dados, um *enable* (ou ativar), e outro *register select*, para o display LCD.

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

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 parte outros aspectos serão abordados.

A arquitetura da CPU do *RPi 2 modelo B*, utilizado nesse projeto, é baseada na *arm-v7*. Consequentemente todos os softwares devem ser compilados em *arm* para que possam ser executados com sucesso.

Alguns softwares foram selecionados para utilização, porém não haviam versões *arm* compatíveis ou estáveis para utilização. Entre eles, o banco de dados MongoDB foi descartado pela versão de instalação *arm* não ter sido testada no *RPi*, e gerada pela comunidade. Diversos relatos de *bugs*, mal funcionamentos e incompatibilidade levaram a mudar a tecnologia de banco de dados.

Uma ferramenta muito importante disponível no *Raspberry Pi* são as GPIOs, ou *General Purpose Input/Output* - portas de entrada e saída de uso geral (Figura 16).

"Esses pinos são a interface física de conexão entre o Pi e o mundo. No mais baixo nível, você pode pensar delas como interruptores que você pode ligar ou desligar (entrada) ou que o Pi possa ligar ou desligar (saída). Dos 40 pinos, 26 são pinos de GPIO e os outros são pinos de energia ou terra.". (RASPBERRY PI FOUNDATION, 2016)

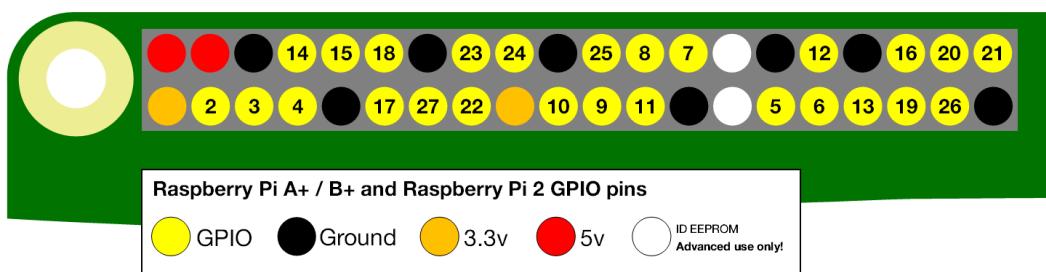
Seguem um padrão definido de numeração (Figura 17). Esses números são utilizados diretamente na programação para acessar os pinos e interagir com o que estiver conectado ali.

Figura 16 – Pinos físicos de conexão



Fonte: (RASPBERRY PI FOUNDATION, 2016)

Figura 17 – Numeração dos pinos de conexão



Fonte: (RASPBERRY PI FOUNDATION, 2016)

4.3.2 Node.js e npm

Node.js³ é uma plataforma de aplicações baseada no motor de Javascript do Google Chrome chamado V8. (MOREIRA, 2013). A linguagem Javascript normalmente é utilizada no navegador, no chamado lado do cliente, porém o Node leva essa linguagem de programação diretamente para o sistema operacional, permitindo que aplicações sejam executadas direto no lado servidor. Seu código fonte está disponível⁴ para acesso, colaboração e comentários.

Utiliza um modelo de aplicações voltado a eventos, ou seja, não bloqueia a execução do código com entrada e saída, cálculos, etc. Em vez disso dispara as funções quando os eventos forem acionados. (MOREIRA, 2013).

Essa plataforma possui um aplicativo chamado *Node Package Manager* (denominado *npm*). Esse aplicativo auxilia a instalação e distribuição de pacotes e bibliotecas auxiliares, normalmente criadas pela comunidade e de código livre. Qualquer pessoa pode criar sua biblioteca, enviar ao GitHub, e enviar ao *npm*.

O *npm* cria o arquivo *packages.json* e salva todas as informações do projeto. Nome do projeto, do criador, link para o repositório no GitHub, dependências de bibliotecas com a

³ <<https://nodejs.org/>>

⁴ <<https://github.com/nodejs/node>>

versão específica instalada, entre outros. Facilita o trabalho de outra pessoa, pois quando utilizar o código fonte, executando o comando *npm install* o mesmo baixa e instala todas as dependências para executar aquele aplicativo.

Diversos pacotes foram utilizados no desenvolvimento desse projeto:

- a) **bleacon**: para descoberta e transmissão de pacotes *beacon*;
- b) **Lcd**: auxilia na conexão com displays de LCD;
- c) **nano**: auxilia na conexão com o banco de dados CouchDB;
- d) **onoff**: disponibiliza funções para conexões com LEDs, botões e hardwares no geral - específico para Raspberry Pi e BeagleBone⁵;
- e) **node-constants**: facilita o uso de constantes entre arquivos e funções;
- f) **numeral**: funções para manipulação avançada de números inteiros e floats;
- g) **is-online**: função para validar conexão com a internet;
- h) **internal-ip**: função para retornar o endereço de IP da rede interna;
- i) **public-ip**: função para retornar o endereço de IP da rede externa, ou IP público;

Quando foi instalado no Raspbian Wheezy, a versão mais recente, compatível e recomendada para o *RPi* era a v0.12.6. Atualmente já existem versões mais novas, porém essa versão permaneceu por questões de compatibilidade com o software desenvolvido.

Para que o sistema executasse constantemente e automaticamente toda vez que o *RPi* fosse ligado, um software desenvolvido em Node.js foi utilizado, denominado pm⁶. Esse aplicativo permite iniciar um software desenvolvido em Node.js, fornece ferramentas para monitorar, reiniciar, parar, entre outras funcionalidades.

4.3.3 CouchDB

CouchDB⁷ é um sistema gerenciador de banco (SGBD) de dados voltado para a *web* e *apps mobile*. Não é um SGBD baseado em SQL, em vez disso utiliza documentos no formato *JSON* (JavaScript Object Notation)⁸, as conexões e requisições são feitas por HTTP (similar a uma conexão a um site), além de permitir servir páginas web direto do banco de dados.

Esse sistema foi utilizado neste projeto pela facilidade de uso com o Node.js. O pacote *nano* para Node abstrai toda a conexão com o banco de dados, permitindo que pouco

⁵ Computador do tamanho de um cartão de crédito, similar ao Raspberry Pi

⁶ <<https://github.com/Unitech/pm2>>

⁷ <<http://couchdb.org/>>

⁸ Modo de formatação dos dados. Muito leve, utilizada para troca de informações entre máquinas, por ser de fácil interpretação e geração. (JSON.ORG, 2016).

código seja utilizado. Os documentos gerados no formato JSON no Javascript são enviados diretamente ao CouchDB sem necessidade de tratamento, manipulação ou configuração.

Disponibiliza uma ferramenta web para acesso aos bancos de dados criados denominada *Futon*, a partir da instalação já funciona normalmente acessando o endereço http://localhost:5984/_utils/ (Figura 18).

Figura 18 – Futon - manipulação dos bancos de dados CouchDB

Name	Size	Number of Documents	Update Seq
_replicator	8.1 KB	1	1
_users	20.1 KB	1	5

Fonte: (COUCHDB, 2016)

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). Caso necessário voltar algumas versões por bug no código, mal funcionamento de uma função recém implementada ou outras razões, as alterações estarão documentadas.

Atualmente o Git é muito utilizado por projetos de código aberto, como por exemplo o Swift⁹ que é uma linguagem de programação criada pela Apple voltada para seus sistemas operacionais, e o AngularJS¹⁰, 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 usuários.

⁹ <<https://github.com/apple/swift>>

¹⁰ <<https://github.com/angular/angular.js>>

4.3.5 Outras Ferramentas

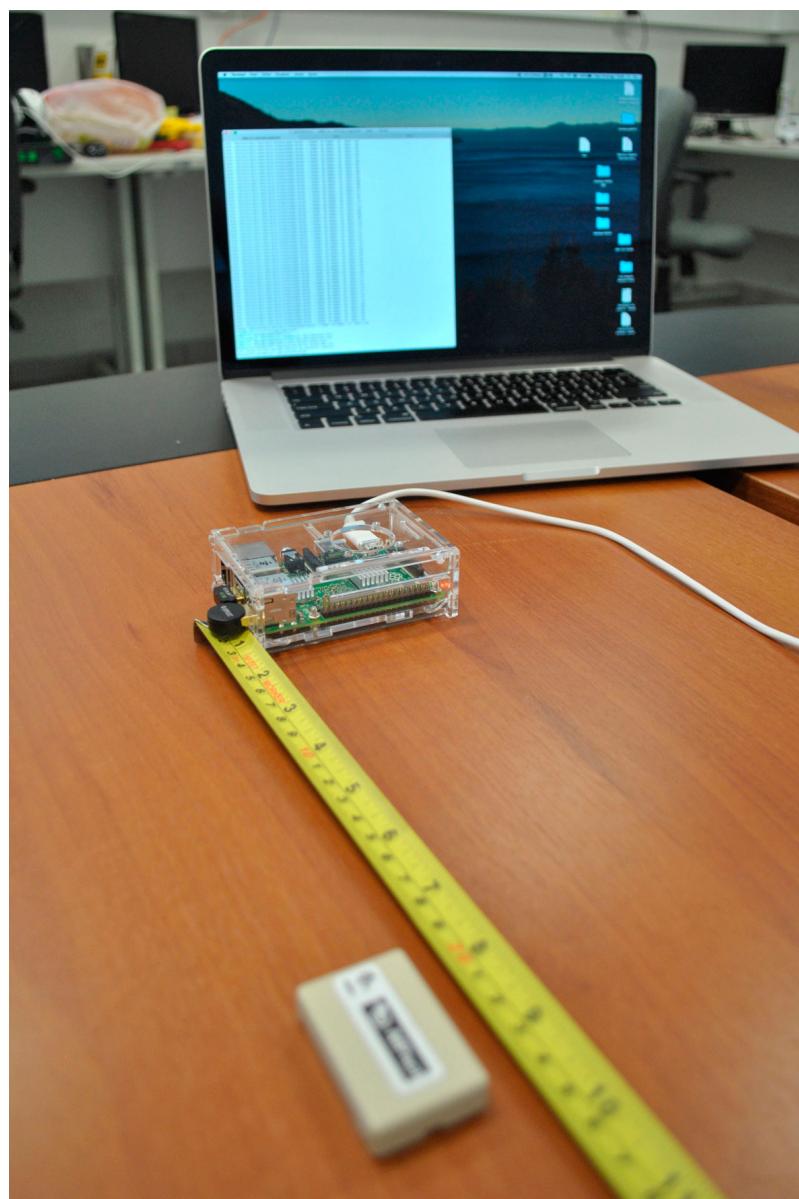
Durante o desenvolvimento desse trabalho foi necessário a criação de uma proposta e um relatório parcial, ambos para a disciplina de Projeto e Implementação de Sistemas I. Para tal, o autor utilizou a ferramenta denominada LaTeX. Como não está entre os objetivos desse projeto, mas houve um grande crescimento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

5 DESENVOLVIMENTO

5.1 Primeira Etapa - Estudo Prático 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 19.

Figura 19 – Primeiro teste realizado, com *RPi* e *beacon MPact*

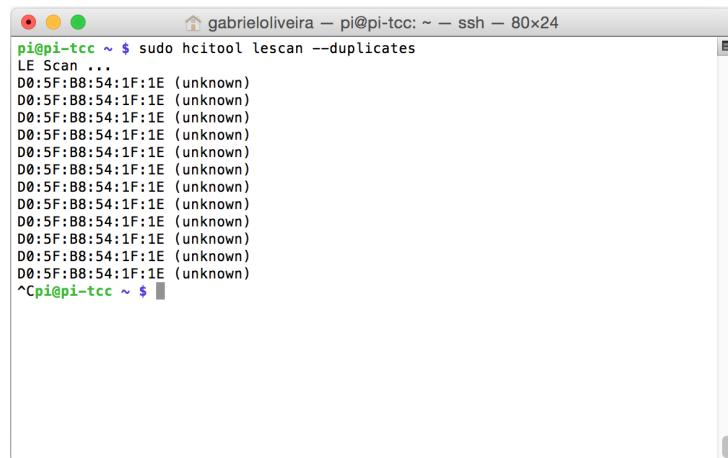


Fonte: elaborado pelo autor

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 20.

Figura 20 – Software *hcitool* executando

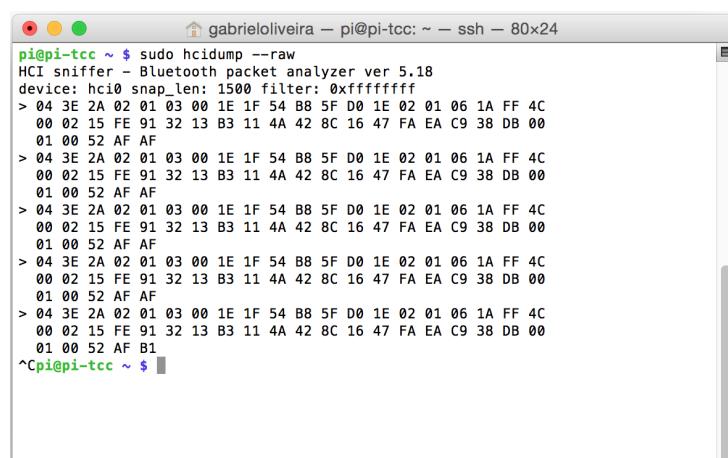


```
pi@pi-tcc ~ $ sudo hcitool lescan --duplicates
LE Scan ...
D0:5F:B8:54:1F:1E (unknown)
^Cpi@pi-tcc ~ $
```

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 21.

Figura 21 – Software *hcidump* executando



```
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 B1
^Cpi@pi-tcc ~ $
```

Fonte: elaborado pelo autor

Em seguida, com os softwares em execução e os pacotes sendo analisados, o *beacon* foi movido ao longo mesa para ficar a diferentes distâncias do *RPi*, conforme Figura 22. Esse passo foi necessário para verificar o formato dos pacotes recebidos e também analisar a distância máxima de identificação.

Figura 22 – Teste com movimentação do *beacon*



Fonte: elaborado pelo autor

Com o adaptador Orico BTA-406 e posicionamento na mesa conforme Figura 19 e Figura 22, a 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 foram tão constante.

O posicionamento do *RPi* foi alterado para testes, conforme Figura 23. Notou-se

uma melhoria na recepção dos pacotes, a 1,5 metros entre o *RPi* e o *beacon* os pacotes começaram a falhar, com recebimento de informações notada até 1,6 metros.

Figura 23 – *RPi* posicionado de outra maneira



Fonte: elaborado pelo autor

Durante essa etapa não foram levados em consideração conexões com LEDs, botões 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 - Planejamento do Protótipo

Para que o protótipo pudesse ser bem planejado, parâmetros foram definidos:

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

Seguindo esses parâmetros, a primeira busca foi por um recipiente para 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 acomodasse 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 24.

Figura 24 – RPi preso na caixa, com as portas acessíveis



Fonte: elaborado pelo autor

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

O próximo passo foi para definir a interface com o usuário. Para que os parâmetros fossem cumpridos, os seguintes itens necessários foram elencados:

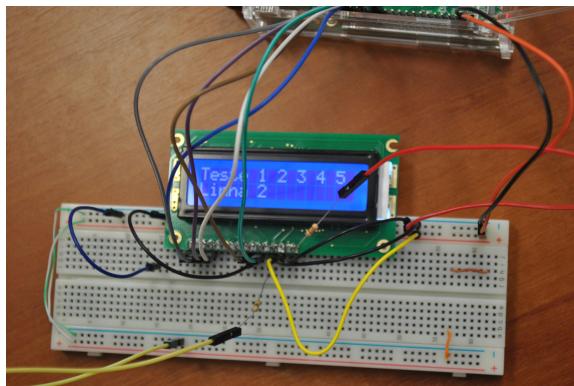
- a) apresentar informação em que consistia se o sistema está executando a aplicação corretamente - para tal definiu-se o uso de um LED verde;
- b) apresentar informação em que consistia se o sistema tem conexão com a internet - para tal definiu-se o uso de um LED amarelo;
- c) apresentar informação em que consistia se um *beacon* está no alcance - para tal definiu-se o uso de um LED azul;
- d) apresentar 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.

Como o display era pequeno e não apresentou todas as informações de uma só vez, decidiu-se pela utilização de navegação por páginas, de forma que as informações fossem alteradas conforme avançasse nas páginas. Para que a navegação fosse realizada de maneira agradável e apresentável, os seguintes itens foram escolhidos:

- LEDs indicando qual página está - definiu-se pelo uso de três LEDs verde;
- botões do tipo clique para navegar a direita e esquerda nas páginas principais, e entre as páginas secundárias.

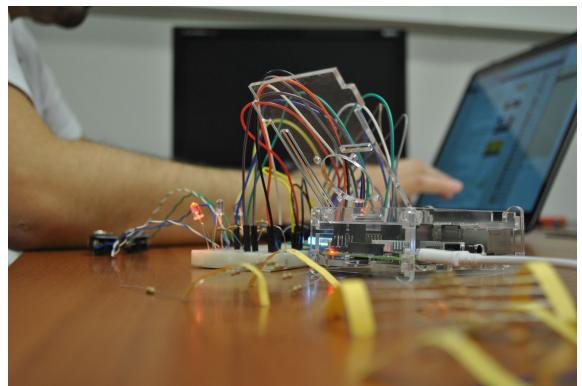
Com os componentes bem definidos, a interface foi concebida inicialmente em rascunhos no papel. Foi necessário realizar testes com os LEDs, display LCD e botões na protoboard de forma que todos esses componentes pudessem ser conectados ao mesmo tempo (Figura 25 e Figura 26).

Figura 25 – Estudo de conexão e programação com display LCD



Fonte: elaborado pelo autor

Figura 26 – Conexão com LEDs, display e botão na protoboard



Fonte: elaborado pelo autor

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 facilitaram a conexão com todos os componentes.

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 e visualização.

Foram necessários soldas e montagem dos outros componentes em placas para que tudo ficasse organizado e fácil de ser trocado, caso necessário. As conexões das GPIOs com os componentes de hardware estão marcados na Figura 31.

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

² <<http://odesenvolvedor.andafter.org/publicacoes/controlando-gpio-do-raspberrypi-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>>

Figura 27 – Pintura da caixa de metal - tampa



Fonte: elaborado pelo autor

Figura 28 – Pintura da caixa de metal - base

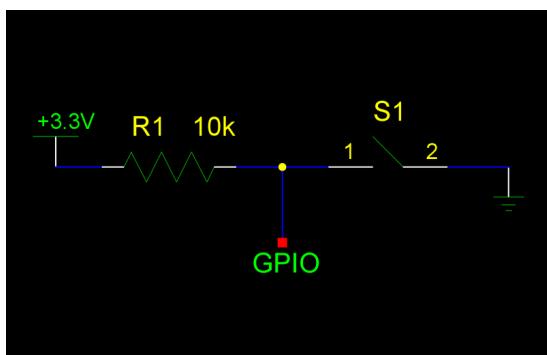


Fonte: elaborado pelo autor

Cada LED foi conectado a uma GPIO, a um resistor de 330Ω , e ao GND. Os botões foram conectados seguindo o modelo da Figura 29. O display foi conectado diretamente a cada GPIO marcada, e também ao +5V e GND para alimentação.

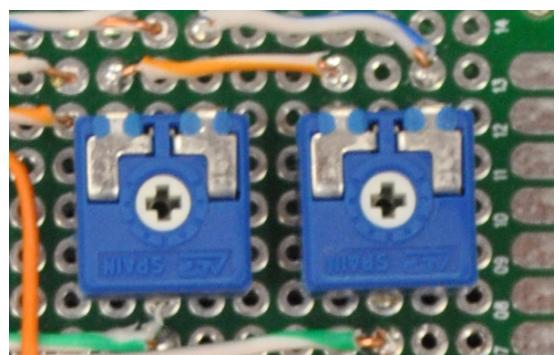
O modelo de display LCD utilizado também tem um pino que utiliza voltagem variável para iluminar cada caractere (contraste de cada letra), e um pino para a iluminação do fundo (azul). Para essas conexões foram utilizados potenciômetros de 10K, para facilitar mudanças caso necessário (Figura 30). Dessa forma, só é necessário girar o pino do potenciômetro para alterar a saída.

Figura 29 – Modelo de conexão dos botões



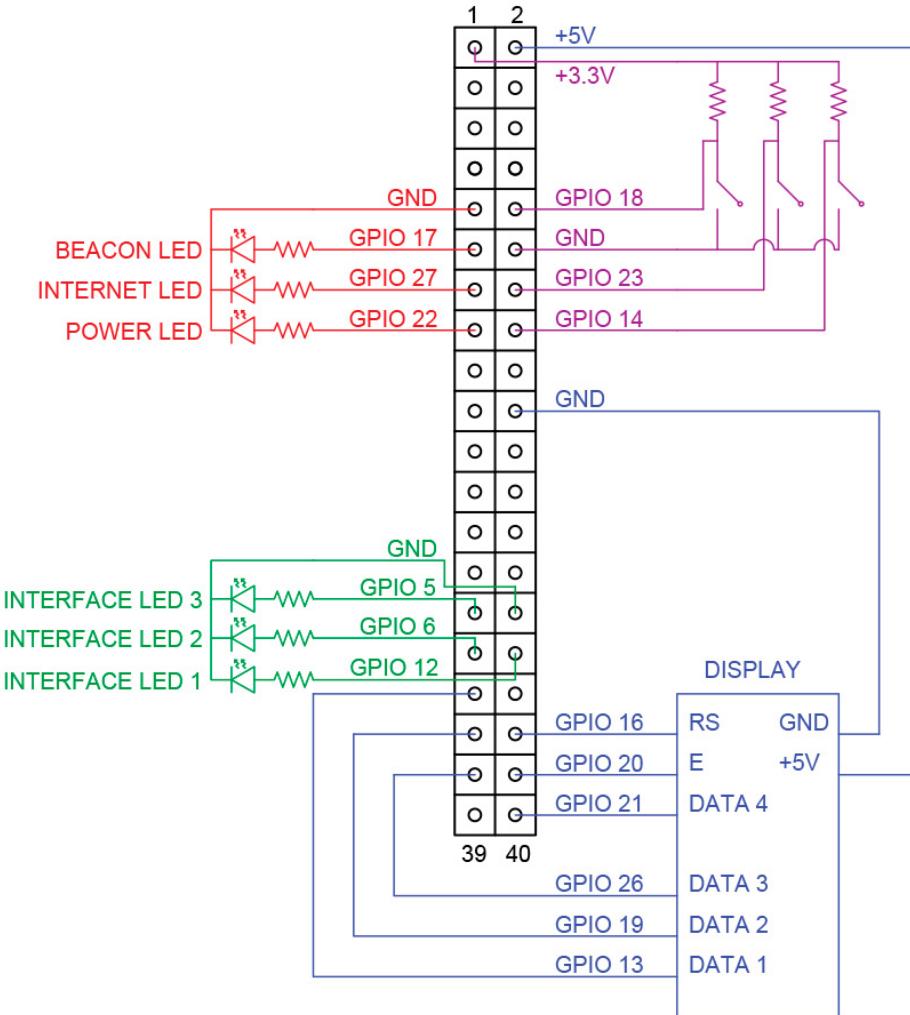
Fonte: (SKLAR, 2012)

Figura 30 – Potenciômetros de configuração do display LCD



Fonte: elaborado pelo autor

Figura 31 – Circuito de conexão das GPIOs



Fonte: elaborado pelo autor

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.

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.

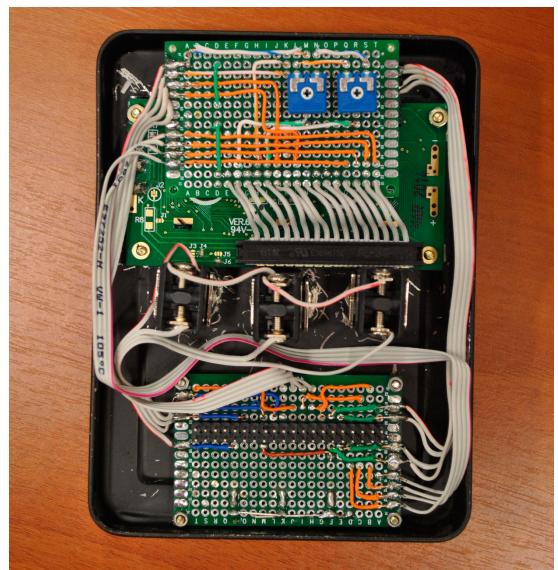
Utilizando os fios internos de cabo de rede para conexão entre os componentes nas placas, o mesmo cabo flat cortado para conexão entre as placas para uma maior flexibilidade e facilidade de mudanças, criou-se o controle da interface, como pode ser visto na Figura 32 e Figura 33.

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 nenhuma conexão fosse quebrada. 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, as informações apresentadas nas telas foram elaboradas:

- 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 as subtelas apresentam o nome do *beacon* encontrado em ordem numérica. Se

Figura 34 – Conexão da tampa com a base



Fonte: elaborado pelo autor

Figura 35 – Criação e montagem do protótipo finalizada



Fonte: elaborado pelo autor

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 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.

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

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

5.2.1 Logotipo e Nome para o Protótipo

Por ser um produto, decidiu-se que o protótipo deveria ter um nome e logotipo. Durante algumas reuniões de *brainstorm*, o nome *Peacon* surgiu e foi o escolhido. Esse nome deriva das palavras *Pi* + *beacon*.

Juntando todos os atributos do Peacon, Henrique Lopes Santos, aluno do curso de Design da Faculdade de Arquitetura, Artes e Comunicação da Universidade Estadual Paulista "Júlio de Mesquita Filho", criou um logotipo que identificasse muito bem todas as funcionalidades do protótipo (Figura 36). Adesivos foram criados para colar no protótipo e identificar o produto. Também foi criado um logotipo menor, para colar em componentes secundários (Figura 37).

Figura 36 – Logotipo criado para o protótipo



Fonte: elaborado pelo autor

Figura 37 – Logotipo menor criado para o protótipo



Fonte: elaborado pelo autor

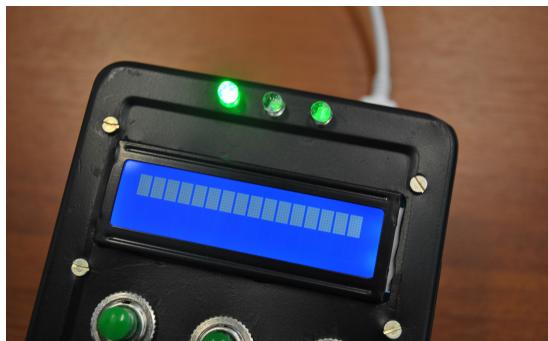
5.2.2 Problemas Enfrentados

Durante os estudos e testes dos LEDs na protoboard utilizou-se resistores de 330Ω para conexão correta. Quanto maior o valor da resistência, menor brilho o LED apresenta. 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 38 e Figura 39.

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 podem ser utilizados.

Figura 38 – LED de alto brilho ao olhar diretamente



Fonte: elaborado pelo autor

Figura 39 – LEDs de alto brilho refletindo na mesa



Fonte: elaborado pelo autor

5.3 Terceira Etapa - Desenvolvimento e Testes da Aplicação

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.

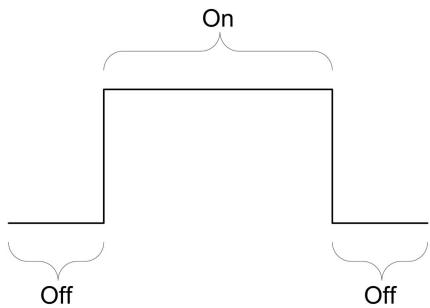
5.3.1 Primeiro Módulo - Biblioteca Auxiliar

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.

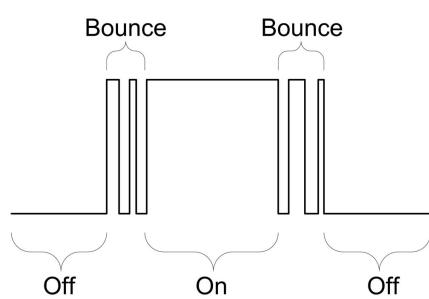
Botões mecânicos não criam ou perdem o contato corretamente, conforme caso ideal e perfeito representado no gráfico apresentado na Figura 40. Em vez disso, podem oscilar rapidamente no momento do clique ou ao soltar o botão, conforme gráfico apresentado na Figura 41. (PROTOSTACK, 2010). Esse efeito é conhecido como *bounce*, ou ruído.

Figura 40 – Gráfico ideal de um clique de botão



Fonte: (PROTOSTACK, 2010)

Figura 41 – Gráfico real de um clique de botão



Fonte: (PROTOSTACK, 2010)

A solução para resolver esse problema (denominada *debounce*) foi de bloquear todos os cliques após o primeiro clique 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 por 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

O segundo módulo consistiu da criação e navegação entre páginas. Para que as páginas secundárias pudessem ser dinamicamente inseridas nas telas, utilizou-se uma matriz de funções para lidar com a troca de informações na interface.

Esse método foi muito eficaz e adaptou-se muito bem na metodologia de desenvolvimento para Node.js. O ato de mudar a tela executa a função na próxima posição do vetor. Isso permite que cada tela, com suas particularidades, seja de fácil criação e alteração, incluindo-se ou retirando-se páginas primárias e secundárias.

Nos testes desse módulo não surgiram problemas, erros ou *bugs*, pela sua simplicidade de desenvolvimento. Nos próximos métodos, trechos criados nesse passo sofreram apenas pequenas alterações de adaptação.

5.3.3 Terceiro Módulo - Identificação dos Beacons

Nesse módulo foram implementadas as funções de identificação dos *beacons*. Quando um *beacon* entra no alcance, é salvo em um vetor auxiliar.

Como os *beacons* transmitem pacotes de tempos em tempos, houve necessidade de criar uma lógica para verificar se ainda estavam no alcance depois de determinado tempo. Para isso, no momento em que um pacote *beacon* é identificado, dispara um evento

a ser executado após um determinado tempo (em segundos). O valor do tempo de espera foi descoberto por meio de seguidos testes, e o valor ideal encontrado foi de 3 segundos.

O próximo passo foi apresentar nas páginas principais e secundárias a quantidade de *beacons* descobertos, e implementar o histórico da aplicação. Nesse módulo os *beacons* descobertos ainda não são salvos, permanecendo somente durante a execução do programa.

5.3.3.1 Problemas Enfrentados

O módulo de identificação dos *beacons* apresentou um problema constante em todas as primeiras execuções após reinicialização do *RPi*. O módulo bluetooth não reportava nenhum pacote *beacon*, e nenhuma descoberta era possível.

Ao analisar a fundo esse problema, percebeu-se que o *driver* desse módulo estava defeituoso, e nada poderia ser feito. A solução encontrada foi de reiniciar a interface bluetooth toda vez que o software fosse executado. Com essa solução, os pacotes *beacon* apareceram e o software funcionava normalmente.

Após algumas semanas, ao avançar no desenvolvimento e testes, percebeu-se que o problema se agravou. Em alguns momentos aleatórios durante a execução do programa o módulo parava de funcionar totalmente, sendo necessária a reinicialização da interface.

Essa solução resolia o problema momentaneamente. Não foram encontrados outros drivers ou solução para o problema, porém como não afetou o desenvolvimento e testes do protótipo não foi necessário a troca do módulo *bluetooth*.

5.3.4 Quarto Módulo - Busca no Banco de Dados

Até o momento o software somente apresentava a quantidade de *beacons* no alcance, e não identificava os mesmos. Nesse módulo foi implementado a busca no banco de dados pelas informações do *beacon* para possível identificação.

O CouchDB tem uma busca muito fácil e prática por identificadores de documento. Para aproveitar essa funcionalidade, um padrão de identificadores e documentos foi criado:

Documento JSON para beacons conhecidos

```
[{"_id": "UUID-Major-Minor",
  "uuid": "numero-uuid",
  "major": "numero-major",
  "minor": "numero-minor",
  "name": "nome-beacon"}]
```

O campo `_id` é formado do número UUID, seguido pelo número `Major`, e finalizado pelo número `Minor`, todos separados pelo caractere - (menos). Exemplo de documento:

Exemplo de documento JSON para beacons conhecidos

```
[{
    "_id": "D9B9EC1F392543D080A91E39D4CEA95C-Major-Minor",
    "uuid": "D9B9EC1F392543D080A91E39D4CEA95C",
    "major": "2",
    "minor": "3",
    "name": "Beacon de LTIA"
}]
```

Dessa forma o terceiro módulo foi aprimorado para apresentar o nome dos *beacons*, permitindo a identificação dos mesmos. Se um *beacon* não fosse encontrado no banco de dados, o texto "*Desconhecido*" seria apresentado.

5.3.5 Quinto Módulo - Salvar no Banco de Dados

Para o quinto módulo, último a ser desenvolvido, restou a parte de salvar os *beacons* encontrados. As informações relevantes a serem salvas foram: *uuid*, *major*, *minor*, data inicial (momento inicial que o *beacon* foi encontrado), tempo total no alcance (em segundos).

Documento JSON para salvar beacons encontrados

```
[{
    "uuid": "numero-uuid",
    "major": "numero-major",
    "minor": "numero-minor",
    "initialDate": "data-inicial-em-timestamp",
    "totalTime": "tempo-total-em-segundos"
}]
```

Como o valor de `_id` não necessita de uma formatação, optou-se por utilizar o CouchDB para gerar um UUID aleatório e atribuir ao `_id`. Não se deve confundir o UUID gerado pelo CouchDB pelo UUID do *beacon*.

5.3.6 Testes e Resultados

Após o desenvolvimento de todos os módulos, a aplicação estava pronta para testes. Como os testes de erros e *bugs* foram realizados após cada módulo, com sucesso, nessa parte somente foi testado a capacidade e geração dos dados.

O software pm2 foi configurado para executar o aplicativo, monitorar erros, fechamentos inexperados, entre outros, e os testes foram iniciados na seguinte sequência:

- a) **Primeiro teste:** descoberta de cada *beacon* MPact individualmente, a diferentes distâncias;
- b) **Segundo teste:** descoberta de mais de um *beacon* MPact simultaneamente, alternando um, dois e três a diferentes distâncias;
- c) **Terceiro teste:** *beacons* MPact em movimento, a diferentes distâncias;
- d) **Quarto teste:** mistura de *beacons* MPact com iPad e Moto Maxx, diferentes quantidades e distâncias;
- e) **Quinto teste:** diferentes localizações dos *beacons* MPact, no bolso da calça com o usuário parado e em movimento.

Todos os testes foram repetidos durante diferentes momentos do dia, durante duas semanas. Os resultados foram:

- a) misturar os dispositivos (*beacon* MPact e smartphones/tablets) não afetou em nenhum ponto - a recepção de todos continuavam normalmente, sem interferência;
- b) o protótipo suportou quatro *beacons* no alcance, simultaneamente, sendo três MPact e o Moto Maxx transmitindo pacotes;
- c) *beacons* em movimento apresentaram pouca ou nenhuma recepção - quando parados, apresentavam recepção constante e também um maior alcance;
- d) quando no bolso da calça, o alcance ficou extremamente limitado - inclusive parado, a distância até o protótipo era muito pouca, no máximo 30cm. Isso se deve pelo *beacon* ser de baixa potência.

5.3.7 Protótipo Final

O protótipo foi desenvolvido com sucesso (Figura 42), todos os parâmetros foram satisfeitos e os testes e resultados apresentaram uma boa implementação do código. Melhorias para futuros protótipos podem ser elencadas:

- a) Baterias internas para facilitar a execução em ambientes sem tomada próxima;
- b) Utilizar LEDs mais fracos ou resistores mais fortes;
- c) Utilizar um melhor adaptador *bluetooth*, de preferência com antenas removíveis para adaptação das antenas dentro do protótipo;
- d) Colocar o *RPi* totalmente dentro da caixa, deixando somente as portas necessárias.

Figura 42 – Protótipo finalizado e totalmente funcional



Fonte: elaborado pelo autor

Interessante notar que todas essas melhorias não são cruciais, mas para uma evolução e maior praticidade do produto final.

6 CONCLUSÃO

Esse projeto teve como objetivo o aprofundamento na área de Internet das Coisas, estudo das tecnologias de *Bluetooth Low Energy*, *Raspberry Pi* e *beacons*. O resultado final é um protótipo de rastreador de *beacons* utilizando o *Raspberry Pi* totalmente funcional, cumprindo com os parâmetros definidos na fase de planejamento.

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*.

A etapa de planejamento do protótipo auxiliou na etapa de desenvolvimento, pois com a estrutura criada foi possível dividir a aplicação em módulos e acelerar a programação e testes do produto final. O protótipo apresentou-se eficaz nos aspectos propostos.

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.
- COUCHDB. *1.7. Futon: Web GUI Administration Panel*. <http://docs.couchdb.org/en/1.6.1/intro/futon.html>, 2016.
- 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.
- JSON.ORG. *Introdução ao JSON*. <http://www.json.org/json-pt.html>, 2016.
- 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-square-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.
- MOREIRA, R. H. *O que é Node.js?* <http://nodebr.com/o-que-e-node-js/>, 2013.
- 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 FOUNDATION. *What is a Raspberry Pi?* <https://www.raspberrypi.org/help/what-is-a-raspberry-pi/>, 2015.
- RASPBERRY PI FOUNDATION. *GPIO: Models A+, B+ and Raspberry Pi 2.* <https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/>, 2016.
- 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 (Figura 43).

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.

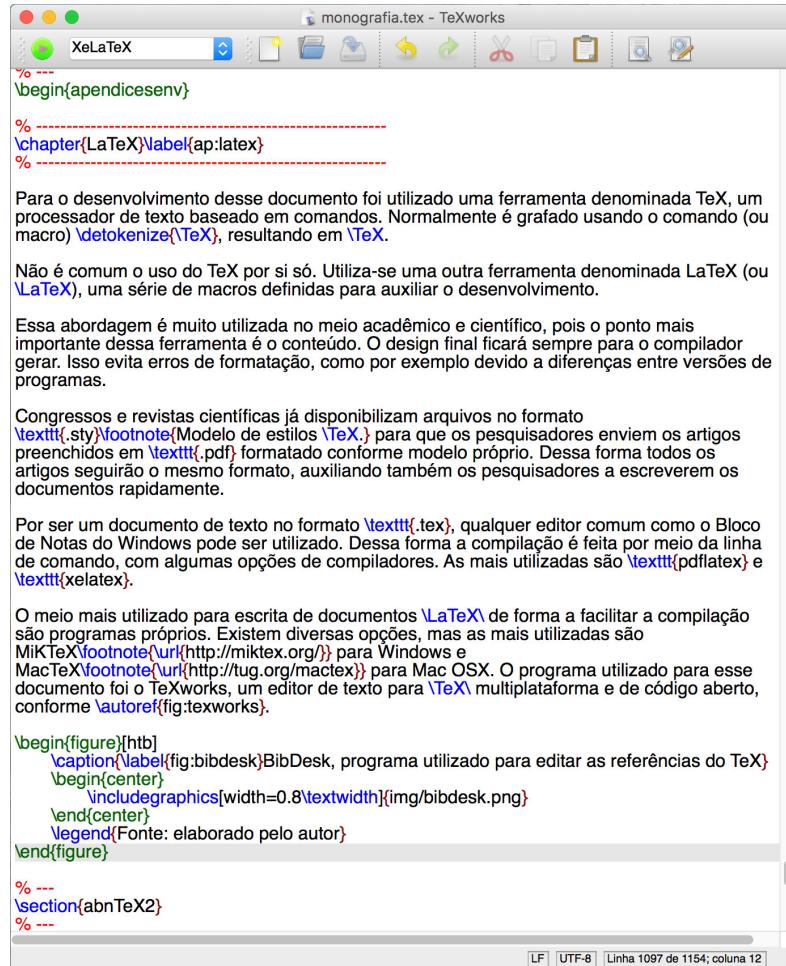
² <<http://miktex.org/>>

³ <<http://tug.org/mactex>>

⁴ Associação Brasileira de Normas Técnicas.

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

Figura 43 – 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.

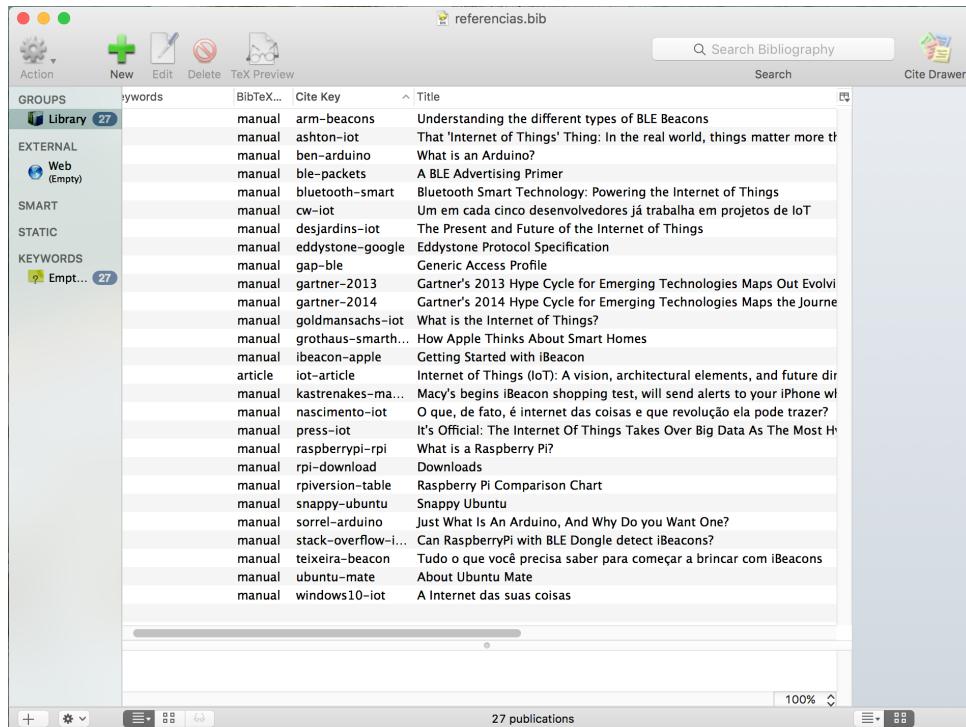
As chaves de citação são os valores que serão utilizados nas macros do abn $\text{\TeX}2$ para que as referências sejam corretamente configuradas. Por exemplo, se desejamos referenciar um artigo de Microsoft (2015), adicionamos todas as informações como nome dos autores, organização, data, edição, entre 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 44. Esse programa já cria o arquivo no formato correto com as chaves de citação configuradas.

Figura 44 – BibDesk, programa utilizado para editar as referências do \TeX



Fonte: elaborado pelo autor

Além de formatar o documento, abn $\text{\TeX}2$ 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 $\text{\TeX}2$ 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 \LaTeX e abn \TeX 2 nessa monografia, entre elas:

- a) Facilidade de criação do conteúdo: abn \TeX 2 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 \LaTeX .

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