Felipe Martinez Camargo
Gabriel de Vasconcellos Torres

PROJETO PHARI: UTILIZAÇÃO DE BEACONS PARA BROADCASTING DE INFORMAÇÃO VIA BLUETOOTH LOW ENERGY PARA SMARTPHONES

São Paulo
(2015)

Felipe Martinez Camargo
Gabriel de Vasconcellos Torres

PROJETO PHARI: UTILIZAÇÃO DE BEACONS PARA BROADCASTING DE INFORMAÇÃO VIA BLUETOOTH LOW ENERGY PARA SMARTPHONES

Trabalho de Conclusão de Curso apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Bacharel em Engenharia Elétrica com Ênfase em Sistemas Eletrônicos

Orientadores: Prof. Dr. Marcelo N.P. Carreño
Prof. Dr. Marco Isaias Alayo Chávez
Conrado Leite de Vitor (Pullup)

São Paulo
(2015)

RESUMO

Este projeto propõe o desenvolvimento de um conjunto hardware + software para se comunicar com smartphones fabricados pela Apple e que utilizam o sistema operacional iOS. Isso será possível através do uso de um módulo Bluetooth Low Energy (BLE112), que fará constantemente o envio de dados para o ambiente e de um aplicativo para smartphones que se conectará ao módulo quando entrar em sua zona de transmissão, podendo fazer tanto o envio quanto o recebimento de informações. Sua aplicação principal é ser utilizado em eventos, aeroportos, shoppings, estádios, supermercados e estabelecimentos comerciais em geral, com o objetivo de passar ao cliente informações que tornem sua experiência mais agradável e dinâmica. Para que o usuário receba as informações em seu celular, o bluetooth deve estar habilitado e o aplicativo deverá estar instalado. Com ele é feito o emparelhamento automático e o usuário recebe notificações do tipo *push* enquanto estiver na área de transmissão. Como a distância de transmissão é pequena, frequentemente abaixo de 30 metros, as informações recebidas pelo usuário podem ser delimitadas de forma precisa baseado em sua localização. A programação do módulo BLE é feita através de um dispositivo chamado CC Debugger, que é conectado a um computador. Através dele é possível controlar o tempo entre transmissões, informações que ele transmite, assim como a potência de transmissão. Testes serão feitos variando as diferentes possibilidades de configuração citadas, visando diminuir o consumo de potência e melhorar a interface do aplicativo e experiência do usuário, buscando a melhor conciliação desses parâmetros para cada situação específica.

Palavras-chave: Beacon. Bluetooth Low Energy. BLE. Geolocalização.

ABSTRACT

This project proposes the development of a set hardware + software to communicate with Apple smartphones using the proprietary iOS operating system. This will be possible through the use of a Bluetooth Low Energy Module (BLE112), which will be constantly sending data to the environment and an application for smartphones that will connect to the module when entering its broadcasting area. This will allow both sending and receiving data from the module. Its main application is to be used at events, airports, shopping malls, stadiums, supermarkets and shops in general, in order to give the customer information that makes his experience more enjoyable and dynamic. For the user to receive information on his mobile phone, bluetooth must be enabled and the application installed. The app allows the user to receive push notifications while inside the broadcast area. Due to the fact that the transmission distance is short, often less than 30 meters, the information received by the user can be defined precisely based on his position. The programming of the BLE transmitter is made through a device called CC Debugger, which is connected to a computer, enabling the control of the time between transmissions and the information it conveys. Tests will be done by varying the different configuration possibilities in order to reduce power consumption and enhance the application interface and user experience, seeking better conciliation of these parameters for each specific situation.

Keywords: Beacon. Bluetooth Low Energy. BLE. Geolocation.

FIGURAS

TABELAS

SUMÁRIO

1 INTRODUÇÃO

O marketing digital tem crescido expressivamente nos últimos anos e já ultrapassou o marketing tradicional, responsável por atingir seu público através de jornais, revistas e televisão . Já o marketing digital, por atingir seu público principalmente através de computadores e *smartphones*, têm se beneficiado com o aumento massivo de usuários. Têm como vantagem menor custo, além de permitir um direcionamento maior para o público alvo se comparado com o marketing tradicional.

Apesar de apresentar diversas vantagens, ainda há espaço para melhorias em uma área ainda pouco explorada: direcionamento dos anúncios para um público especifico através de marketing geolocalizado. Marketing geolocalizado consiste em realizar campanhas publicitárias altamente personalizadas com base na localização do público alvo. Com base nesse conceito e por se tratar de um mercado ainda pouco explorado devido à dificuldade de localizar de forma precisa a posição de potenciais clientes, principalmente em espaços fechados onde o uso de GPS não é adequado, ocorreu o surgimento de um novo produto: *beacons*.

Beacons são pequenos módulos transmissores que enviam constantemente pequenos blocos de informações para o ambiente, em busca de usuários que entrem em sua zona de transmissão. Utilizam *Bluetooth* 4.0 (ou *Bluetooth Low Energy*), método de transmissão caracterizado pelo baixo consumo de energia e alta precisão. Desse modo, é possivel definir quando um usuário receberá em seu *smartphone* determinada informação. Por exemplo, para uma loja localizada dentro de um shopping e que possui produtos em promoção, é possível alertar consumidores através do envio de fotos e texto para os celulares de usuários que passarem em frente à vitrine da loja. Também é possivel ser utilizado em eventos, hospitais e feiras, fornecendo informações como mapas e agendas.

O uso de *beacons* vem se popularizando nos Estados Unidos: existem algumas empresas especializadas em produzir o *hardware* e grande parte dos maiores varejistas estudam sua implementação no curto prazo. Segundo estimativas, em 2018 devem existir 4,5 milhões de beacons ativos, movimentando mais de $4 bilhões. No Brasil, no entanto, o mercado ainda é bastante tímido e grande parte das empresas sequer ouviu falar sobre *beacons*. O *hardware* é, na maioria das vezes,

importado, impedindo que sejam feitas customizações para adequar o produto ao mercado brasileiro.

O projeto propõe o desenvolvimento de um *beacon*, ou seja, do *hardware* responsável por fazer o envio das informações via *Bluetooth Low Energy*, e o desenvolvimento de um aplicativo para *smartphones* com sistema operacional iOS, que receberá as informações e as apresentará para o usuário através de uma interface simples e intuitiva.

2 DETALHAMENTO DO PROJETO

2.1. Identificação do Problema

As empresas (clientes), para se adequarem a competitividade contemporânea, buscam de forma sistemática ferramentas e técnicas que as auxiliem atingir o seu público alvo com maior eficiência. Uma das principais estratégias dos anunciantes consiste em fazer com que o maior número possível de pessoas observem seus anúncios.

Apesar de serem utilizados meios nos quais a chance das empresas obterem êxito é maior, o número percentual de pessoas que respondem aos anúncios é bastante reduzido, ficando em 3.4% para correspondência e 0.12% para email. Dados mostram que 86% das pessoas não assistem aos comerciais na televisão e 44% da correspondência não chega a ser aberta pelo consumidor. O problema consiste em não ser capaz de identificar e focar os esforços em um público que têm maior tendência de consumir seus produtos. Ainda não é possível fazer publicidade altamente personalizada com cada usuário, baseada nos lugares que ele frequenta.

2.2. Identificação das Necessidades do Cliente

Hoje em dia a publicidade está tentando cada vez mais aumentar a eficiência em atingir o público-alvo, como por exemplo, as propagandas existentes em sites como o Google são baseadas no perfil que é criado de acordo com os sites que são visitados pelos usuários, produtos que são procurados, etc.

De uma maneira geral, potenciais clientes que estão andando na rua, em shoppings ou outros lugares, que não estejam no computador, não tem essa propaganda personalizada, criando um espaço e uma oportunidade para soluções que personalizem a propaganda para todas as pessoas em todos os lugares. Isso cria a necessidade de conhecer a localização das pessoas. Por exemplo, pessoas em shoppings receberiam informação de promoções e lojas, ou em feiras receberiam informação sobre os estandes que estão à mostra.

2.3. Declaração das Necessidades do Cliente

Os clientes (empresas) necessitam de um meio eficaz de atingir as pessoas baseando-se no padrão de consumo e localização delas. Isso significa utilizar dispositivos que são estrategicamente posicionados, atingindo o público que está na área, ou seja, utilizando a geolocalização para obter maior sucesso na divulgação de seu produto.

A partir da localização precisa do público, a empresa interessada pode direcionar e atingir seu público alvo, alterando a mensagem que cada usuário recebe para se adequar aos seus gostos previamente conhecidos.

O cliente necessita uma plataforma intuitiva e de fácil operação, que permitirá o contato com seu público de maneira ágil e altamente direcionada e personalizável, permitindo que qualquer funcionário da empresa opere o software e faça as modificações desejadas.

Da mesma maneira, necessita que o custo de implantação e operação seja pequeno, a instalação dos *beacons* seja rápida e não seja agressiva (não sendo necessário quebrar paredes, por exemplo) e o seu tamanho seja pequeno, permitindo sua instalação independente do tamanho e do leiaute do estabelecimento.

2.4. Declaração dos Objetivos do Projeto

O objetivo principal do projeto consiste em achar uma maneira de transmitir informações de qualquer tipo (propaganda, promoções, notícias, entre outros) para certo grupo de pessoas de forma geolocalizada, ou seja, para pessoas que estão localizadas em um lugar comum, como feiras ou grandes eventos, em que essas informações sejam interessantes ou importantes. Dessa forma, é possível identificar pessoas próximas do local, que são potenciais clientes ou pessoas interessadas no que se tem para vender ou transmitir. Para atingir esse objetivo, será desenvolvido um circuito de bluetooth de baixa potência (*Bluetooth Low Energy* ou BLE) que funcionará como uma espécie de totem (*beacon*), fazendo transmissões globais (broadcasting) constantemente, sempre em busca de usuários na sua região de transmissão. Ao identificar um usuário, será verificado se ele possui o aplicativo

necessário para receber as mensagens e caso ele possua, um algoritmo verificará a base de dados a partir do ID deste usuário e enviará notificações de interesse a ele.

Outras aplicações que são de interesse do grupo consistem em utilizar a tecnologia de BLE para permitir pagamentos através do celular, seja em supermercados e shoppings ou ao utilizar o transporte público, onde a cobrança da passagem será feita automaticamente.

3 PESQUISA DE LEVANTAMENTO DA SITUAÇÃO

3.1. Tecnologias Existentes

Existem algumas empresas que produzem *beacons* hoje em dia. As principais delas são *Estimote*, *BlueCats* e *Kontakt*. Basicamente a tecnologia funciona por proximidade, ou seja, quando um smartphone chega perto do *beacon*, ocorre uma troca de informações entre os dois, por conexão bluetooth. Os *beacons* tem diversas possibilidades de uso, cuja finalidade principal é fornecer um meio de geolocalizar os usuários, como em shoppings, feiras e ambientes fechados no geral, onde a comunicação via satélite é prejudicada . Podem ser utilizados também para automatização residencial, realizando o controle de equipamentos eletrônicos.

Um exemplo experimental da utilização e da tendência do uso de b*eacons*, consistiu em um evento de jogo de futebol ao qual foram instalados diversos *beacons* dentro do estádio para fazerem propaganda de uma promoção da empresa *Netshoes*. Desta forma, foi possível atingir grupos de pessoas que chegavam ao estádio e recebiam uma mensagem dos *beacons* avisando-lhes de promoções de artigos esportivos associados aos clubes que jogariam, proporcionando aumento considerável nas vendas e no impacto da propaganda da empresa.

Figura 1 – Exemplo de um Beacon produzido pela empresa Estimote

3.2. O Futuro dos *Beacons*

A nova tecnologia de *Bluetooth Low Energy* já está presente nos celulares há alguns anos mas ainda não tem sido largamente utilizada. Com o grande aumento nas vendas de *smartphones* para todas as classes sociais e faixas etárias, espera-se que a utilização de *beacons* aumente consideravelmente em todas as áreas nos próximos anos.

Como pode ser visto na figura abaixo, apesar do uso de *beacons* ainda não ser muito elevado, é estimado que atinja 32 dos 100 maiores varejistas em 2015, chegando a 85 em 2016, um crescimento muito acentuado que demonstra que o mercado percebeu o grande potencial que essa tecnologia possui para auxiliar os lojistas a atingirem o seu público, seja com a aviso de promoções, para auxiliar na localização dos produtos dentro dos estabelecimentos ou simplesmente para enviar informações que melhorem a experiência do usuário quem passa na área de *broadcasting* dos *beacons*.

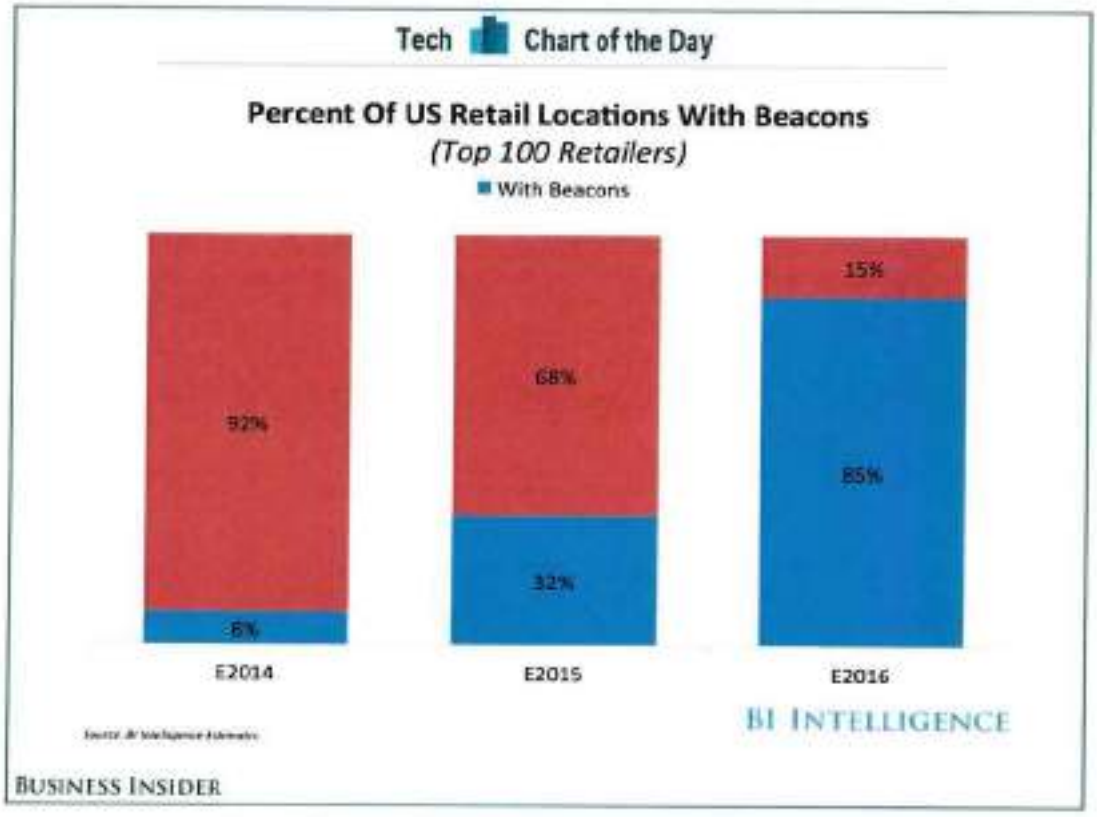

Figura 2 - Projeção do uso de beacons entre os 100 maiores varejistas

Apesar da concepção inicial dos *beacons* ter sido feita visando a sua aplicação no mercado de vendas, essa nova tecnologia também apresenta grande potencial em diversas outras áreas, por exemplo:

- Estádios e teatros, utilizando seus celulares para procurar a localização de seu assento, banheiros, lanchonetes e até mesmo liberar o acesso na entrada somente ao se aproximar das catracas;
- Hóteis, permitindo fazer o check-in, abrir a porta dos quartos e utilizar os celulares para fazer compras em substituição aos cartões internos dos hóteis;
- Parques de diversão, onde os *beacons* informariam o tempo nas filas e informações sobre os brinquedos;
- Compras, onde após sair do estabelecimento o pagamento seja feito de forma automática;
- Museus e zoológicos, o uso de *beacons* permitiria enviar informações detalhadas sobre as obras ou animais ao celular dos usuários quando eles se aproximassem do ponto de observação.

Figura 3 - Aplicativo que auxilia o usuário a se localizar

Como pode ser visto, sua aplicação irá crescer largamente nos próximos anos, sendo um dos limites para o uso de *beacons* a criatividade dos usuários e desenvolvedores além da aderência do mercado e das empresas através de aplicações inovadoras e de alto impacto.

4 HARDWARE

4.1. Bluetooth Low Energy

Bluetooth Low Energy, também conhecido como Bluetooth Smart ou *Bluetooth* 4.0, é a mais nova versão dessa tecnologia que surgiu no final do século XX. Além de trazer melhorias como aumento da velocidade e capacidade de transmissão de dados, aumento na segurança da comunicação, entre outros, também foi criado um novo modo de comunicação, chamado de *advertisement* (propaganda). Como mostrado na figura 4, existem 40 canais espaçados de 2 MHz entre si, sendo 37 canais para transmissão e recepção de dados por comunicação via conexão e emparelhamento, e 3 canais específicos para a função *advertisement* (normalmente os 3 canais de *advertisement* são utilizados pois numa área com muitos transmissores, se apenas 1 canal for utilizado, pode haver interferência entre os transmissores e o receptor pode não reconhecer todos os transmissores presentes no ambiente).

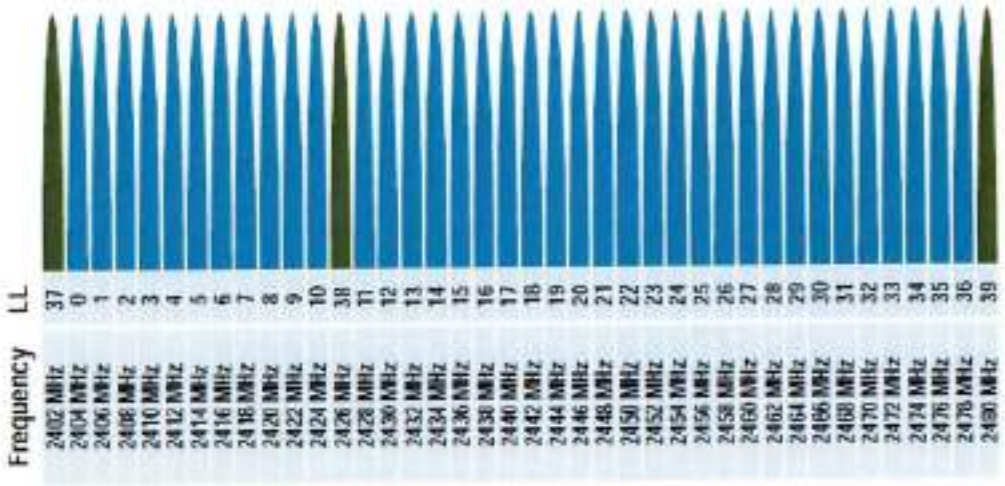

Figura 4 - Canais do Bluetooth 4.0

O modo de *advertisement* funciona com o propósito de não haver necessidade de emparelhamento e/ou conexão entre dois aparelhos *Bluetooth* com a versão 4.0 para que ocorra comunicação entre eles. O dispositivo que estiver em modo *advertisement* envia dados para o ambiente com uma certa frequência para que todos os aparelhos que estiverem em seu raio de transmissão recebam esses dados, sem a necessidade de estabelecer conexão previamente. Os dispositivos

que recebem esses dados podem utilizá-lo para realizar alguma ação, como por exemplo, carregar uma página da internet ou uma tela em um aplicativo. Além de receber os dados, os receptores têm uma função chamada *scan response*, onde a partir do recebimento de certos dados, o dispositivo envia uma resposta ao transmissor. Essa função, que pode ser utilizada para fazer o emparelhamento e conexão entre dois aparelhos sem que haja a necessidade do usuário fazê-la manualmente, ocorre da seguinte maneira: o transmissor envia os dados, informando que deseja estabelecer conexão, o receptor responde, com os valores de emparelhamento e conexão, e o transmissor inicia a conexão automaticamente.

No caso dos *beacons*, eles são utilizados apenas no modo *advertisement* para acionar alguma ação no receptor (*smartphone*). Este receberá os dados, buscará na memória ou em um servidor qual página deverá ser carregada, criará a notificação e abrirá a página.

O modo *advertisement* envia pacotes de dados de tamanho máximo de 31 bytes, como está indicado na figura 5. A empresa *Apple*, que enxerga grande potencial nessa nova tecnologia, estabeleceu uma série de padrões que devem ser obedecidos para que os *beacons* possam ser classificados como iBeacons, e entre eles está a separação que deve ser feita no pacote de dados, que está descrita abaixo:

- 9 bytes de prefixo (dados do fabricante, nome do beacon, flags, entre outros);
- 16 bytes de UUID (número de identificação do beacon, normalmente utilizado 1 por empresa, ou seja, todos os beacons da mesma empresa têm o mesmo UUID);
- 2 bytes de *major* e 2 bytes de *minor* (números que permitem uma identificação com maior precisão entre os *beacons* da empresa, onde cada *beacon* possui um valor único de major e minor);
- 1 a 2 bytes com o valor da potência média recebida a 1 metro de distância do beacon (esse valor pode ser testado ou estimado e permite o cálculo da distância entre o *beacon* e o *smartphone*).

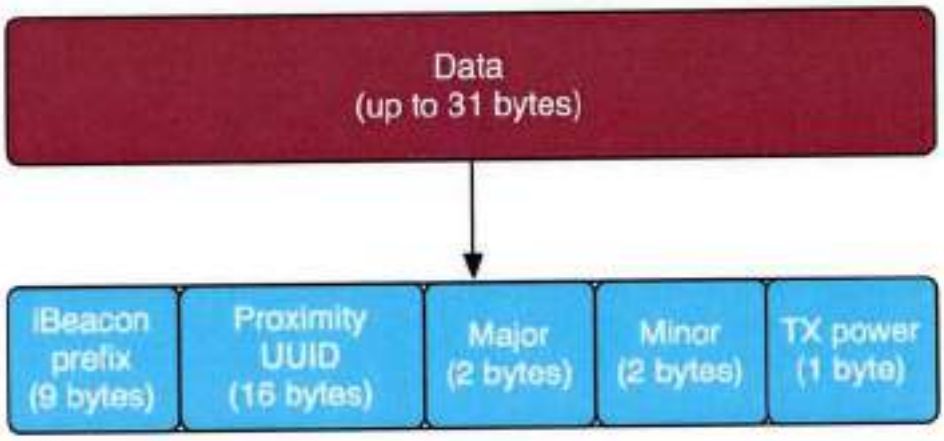

Figura 5 – Composição do pacote de dados do modo advertisement

4.2. Módulo de Transmissão BLE112-A

Ao escolher o módulo que faria a transmissão e recebimento dos dados, buscou-se um dispositivo que atendesse ao maior número possível de requisitos de engenharia. Optou-se por utilizar o módulo BLE112-A, produzido pela B*luegiga*, que utiliza *Bluetooth Low Energy* (*Bluetooth 4.0*) para comunicação e tem seu uso voltado para dispositivos de baixo consumo. Oferece também suporte a perfis de usuário, possui pinos de entrada e saída que permitem o controle de sensores e LEDs e sua memória interna possibilita o armazenamento de informações.

Possui memória *flash*, antena e microcontrolador integrados na mesma placa, não havendo a necessidade de fabricar uma placa de circuito impresso e soldar os componentes, o que torna a implementação mais rápida e fácil, sendo somente necessário programar o módulo para que ele funcione como um *beacon*.

Figura 6 - Módulo transmissor BLE112-A

Principais especificações técnicas:

- Custo: R$40,00;
- Potência de transmissão: +3dBm a -23 dBm;
- Corrente de pico durante transmissão: 27mA (0 dBm);
- Corrente em repouso: 0.4uA;
- Alimentação: 2.0 – 3.6V;
- Taxas de transmissão: +100kbps;
- *Broadcast* para número ilimitado de dispositivos e até 8 conexões em modo master (com envio e recebimento de dados);
- Microprocessador Intel 8051;
- Antena 2.4GhHz;
- Memória flash: 128kB;
- Temperatura de operação: -40° C a +85°C;
- Dimensões: 18.1 x 12.05 x 2.3 mm.

4.3. Programador CC Debugger

CC Debugger é um programador e debugador produzido pela *Texas Instruments* utilizado em SOCs (*System on Chip*) de radiofrequência de baixo consumo que

utilizam o microcontrolador Intel 8051 (utilizado no BLE112) e permite a programação da memória *flash* interna do módulo de transmissão.

O *CC Debugger* é conectado ao computador através de uma conexão USB e ao módulo BLE112 através de um conector 2x5 presente na *Breakout Board*, que será explicada na próxima seção. A programação é feita através do software *SmartRF™ Flash Programmer*, disponibilizado pela *Texas Instruments*.

Figura 7 - Programador CC Debugger

4.4. Sensor de Temperatura ADT7301

O ADT7301 é um chip de monitoramento capaz de medir a temperatura ambiente e fazer o envio dessa informação via interface serial, permitindo que seja utilizado com a maioria dos microcontroladores do mercado. O ADT7301 será responsável por medir a temperatura ambiente, enquanto o BLE112 irá coletar está informação e fazer o envio via *Bluetooth Low Energy* para o aplicativo de celular desenvolvido na plataforma iOS. Após o recebimento dos dados no celular, é feito o armazenamento destas informações em um log para posteriormente ser traçado o histórico de temperatura em determinado periodo. Desse modo pode-se, por exemplo, ter a

temperatura do ambiente em tempo real e diminuir o gasto de energia com ar condicionado.

Optou-se por utilizar este componente devido ao seu baixo consumo de potência, baixo custo, dimensões reduzidas e tensão de alimentação e interface compatível com o módulo de transmissão BLE112.

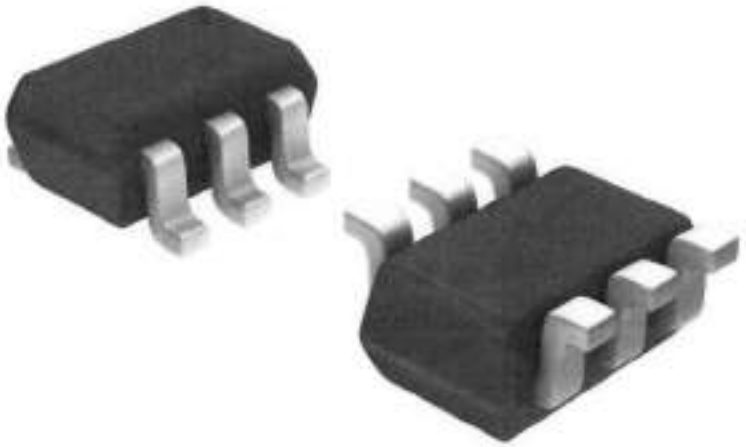

Figura 8 - Sensor de temperatura ADT7301

Principais especificações técnicas:

- Custo: R$12,00;
- Faixa de medição de temperatura: −40°C a +150°C;
- Alimentação: 2.7 – 5.25V;
- Precisão típica: ±0.5°C;
- Conversor analógico-digital de 13 bits permite atingir resolução de 0,03125°C;
- Dimensões: 2.9 x 1.6 x 1.15 mm
- Corrente de repouso: 1 µA;
- Potência de repouso de 4.88 µW.

4.5. Breakout Board

Breakout board são circuitos frequentemente utilizados em projetos eletrônicos. Sua função principal é facilitar o acesso a pinos de componentes como microcontroladores, que possuem diversas portas próximas e de difícil acesso e

servir de base para eventuais componentes necessários para o funcionamento do circuito.

No projeto PHARI, sua função principal é auxiliar na programação do módulo transmissor, que será soldado na *Breakout Board*, assim como resistores, capacitores, botões, LEDs e suporte para a bateria. Além disso, permitirá que o módulo seja alimentado através do programador *CC Debugger*, evitando o gasto de bateria durante o período de testes.

No total foram feitas duas versões da *Breakout Board*. A segunda versão apresenta diferenças significativas que melhoraram o design e resolveram alguns problemas que foram encontrados na primeira versão. Ambas as placas foram desenvolvidas utilizando o software *Altium Designer* e fabricadas no Laboratório de Microeletrônica da USP. Para produção das placas foi utilizado equipamento da empresa *LPKF Laser & Electronics*, capaz de fazer trilhas, furos e recortar placas com até 2 camadas.

4.5.1. Versão Inicial (v1)

A elaboração do esquemático teve início no desenvolvimento do esquema de ligação entre o BLE112 e o *CC Debugger* que é feito através de um conector 2x5 e tem como propósito permitir a programação do módulo transmissor. Caso ocorresse alguma falha como travamentos ou dados corrompidos durante a programação, um botão conectado no pino 7 do conector teria a função de atuar como *Reset*. É importante notar que foi adicionado um resistor *pull-up* para evitar flutuações indesejadas de tensão no pino 7 do *CC Debugger*, mantendo-a em 3V caso o botão não fosse pressionado.

Dois Leds, um vermelho e outro verde, tem como função indicar que o circuito está ligado e fazendo o *broadcasting* de informação. Para evitar a presença de ruídos de alta frequência, foram posicionados 3 capacitores de desacoplamento de 100nF entre o VCC e GND. Por fim, a fixação da bateria na placa de circuito impresso foi feita com um soquete específico para baterias do tipo CR2032 de 3V.

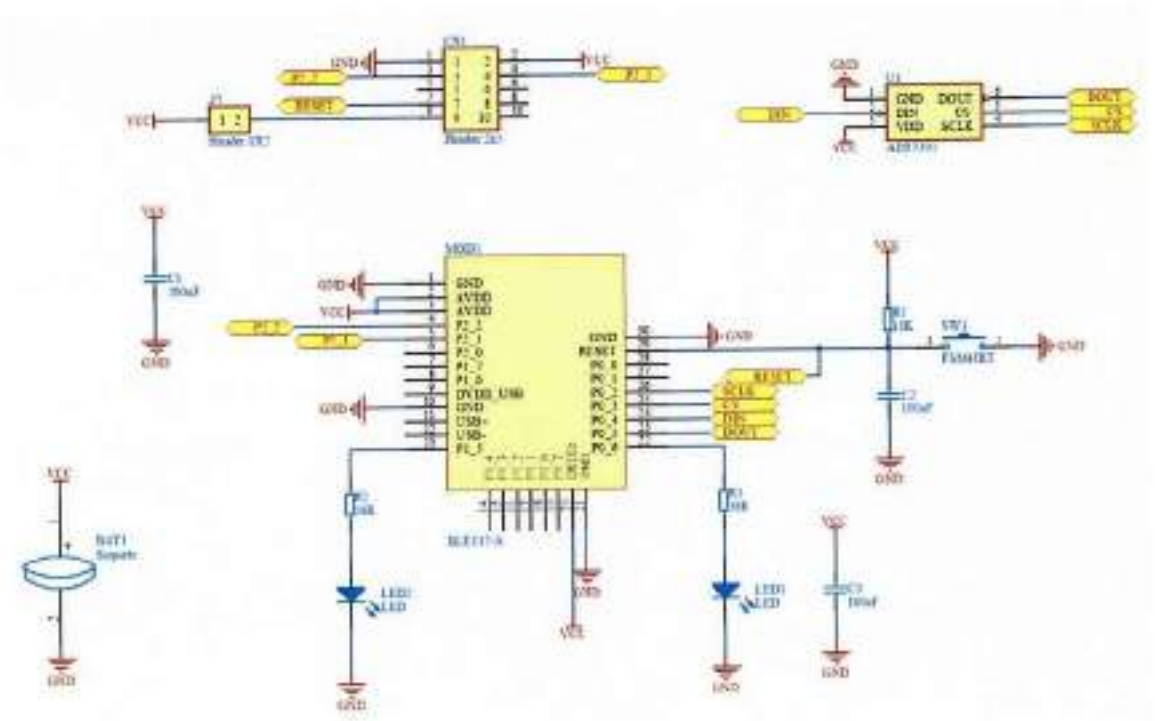

Figura 9 - Esquemático da Breakout Board v1

Ao fazer o posicionamento dos componentes na PCB, o conector do *CC Debugger*, o botão de *Reset* do programador e jumper foram posicionados nas extremidades da placa para facilitar o manuseio e deixá-la mais organizada.

Conforme consta no manual do BLE112, deve-se remover a camada de cobre sobre a área que fica abaixo da antena para garantir o correto funcionamento do módulo. Isso deve ser feito tanto na camada superior como na inferior.

A placa possui duas camadas (*Top* e *Bottom*), dimensões de 32,5 x 38,7mm e trilhas com espessura de 0,25mm. A camada inferior foi utilizada como plano de VCC porém as trilhas que transmitem a tensão VCC na parte inferior da placa não foram removidas e com isso a placa ficou com diversas trilhas redundantes. Além disso, por erro de projeto a camada superior não atuou como plano GND.

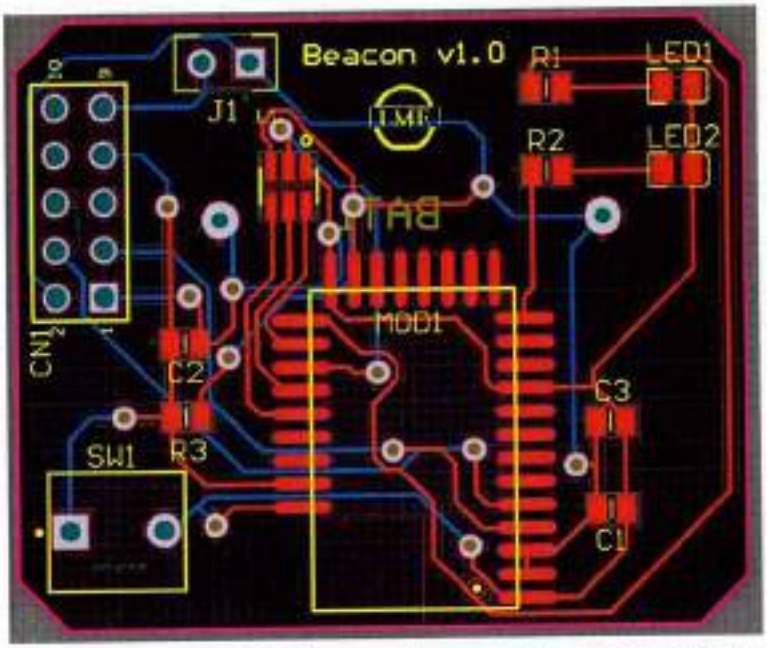

Figura 10 - Versão inicial da placa de circuito impresso. Em vermelho vemos a camada superior e em azul a camada inferior

Figura 11 - Vista 3D da Breakout Board v1. Desenvolvido com o software Altium Designer

Após conclusão do posicionamento dos componentes, foi produzida uma cópia da *Breakout Board* v1. A soldagem dos componentes ocorreu na *Pullup*, empresa incubada no CIETEC, com a ajuda do orientador Conrado Leite, que ensinou técnicas de soldagem e disponibilizou equipamentos específicos para soldagem de componentes SMD.

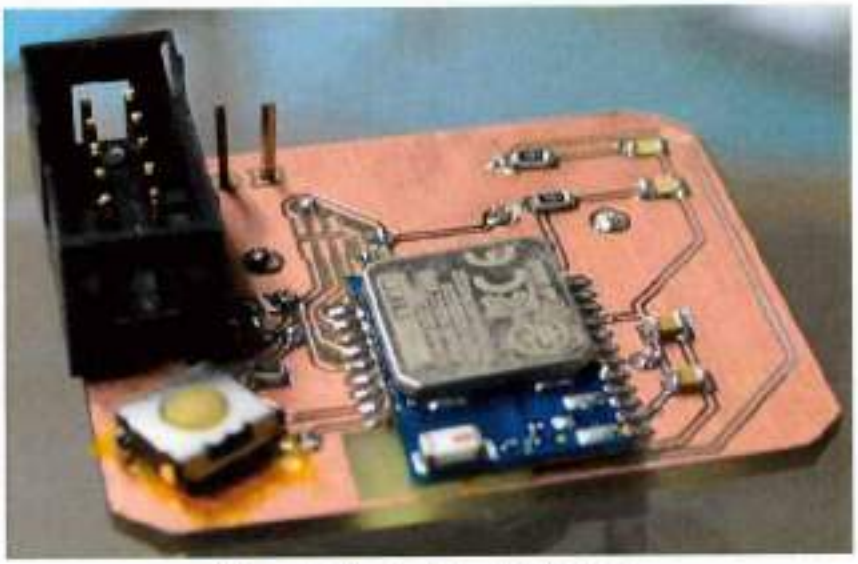

Figura 12 - Breakout Board v1 finalizada

4.5.2. Versão Final (v2)

Após testes com a *Breakout Board* v1, foram levantados alguns pontos que necessitavam melhorias e com isso foi desenvolvida a versão final do esquemático. As principais alterações são:

- Remoção de um dos leds, visto que somente um led é suficiente para indicar que o *beacon* está ligado e transmitindo dados;
- Remoção do botão de *reset* do *CC Debugger* pois caso apareçam erros durante a programação há outras maneiras de reiniciar o *CC Debugger*;
- Adição de botão liga/desliga, evitando que a bateria tenha que ser colocada/removida constantemente. Na *Breakout Board* v1 a solda entre o soquete e a placa não resistiu ao esforço mecânico ao qual o soquete foi submetido e apresentou mal contato.

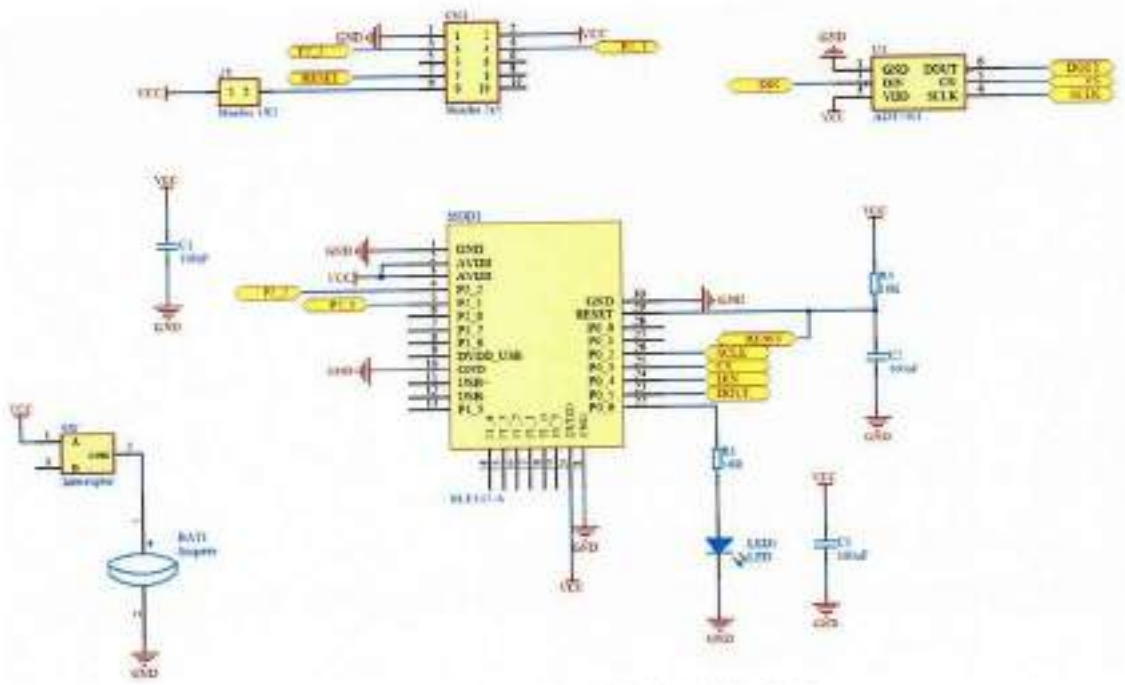

Figura 13 - Esquemático da Breakout Board v2

A seguir estão listadas as alterações que foram feitas no desenvolvimento da placa de circuito impresso em comparação com a sua versão inicial:

- Aumento das dimensões da placa para comportar a adição do botão liga/desliga e soquete da bateria, que foi deslocado para a camada superior. As novas dimensões da PCB são 47,2 x 39,6 mm;
- Alteração da espessura das trilhas para 0,5 mm pois algumas trilhas da versão inicial estavam demasiadamente finas, o que aumenta o risco de problemas no processo de fabricação e soldagem dos componentes;
- Adição de um plano GND na camada superior e remoção de trilhas redundantes (que transmitem tensão VCC na camada inferior e GND na camada superior) o que permitiu expressiva redução na quantidade de trilhas.

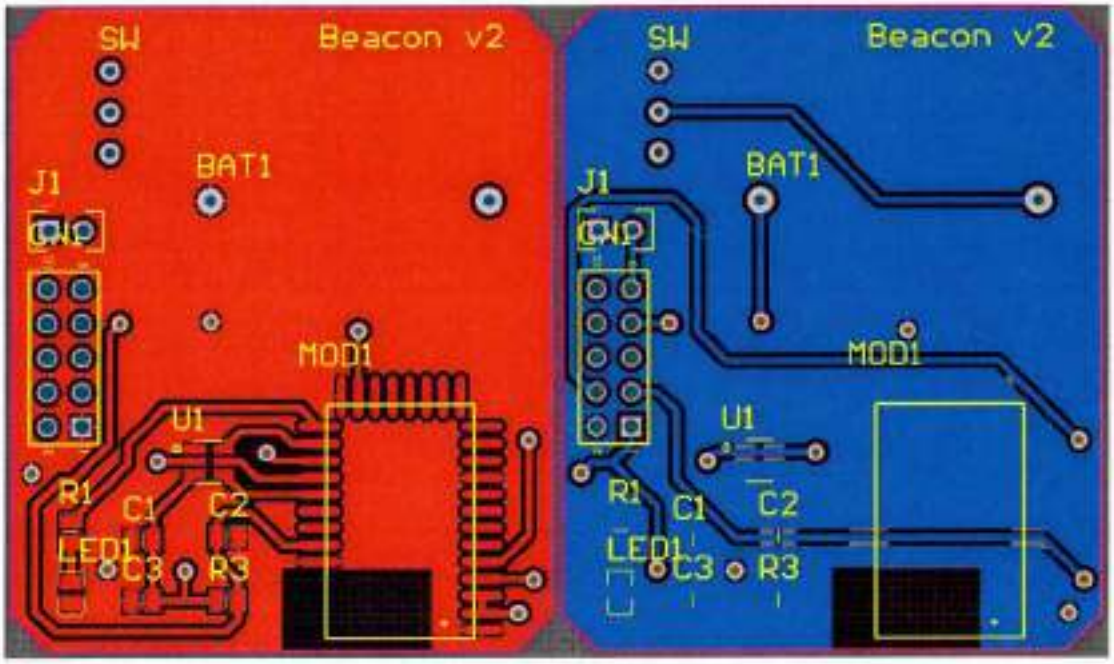

Figura 14 - Versão final da placa de circuito impresso. Em vermelho vemos a camada superior e em azul a camada inferior

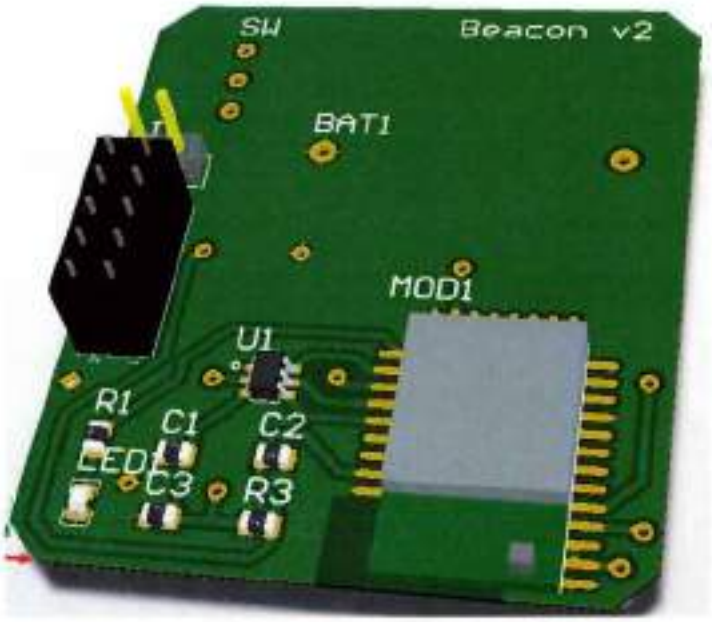

Figura 15 - Vista 3D da Breakout Board v2

Foram produzidas três placas da versão final da *Breakout Board*. A produção e soldagem dos componentes ocorreu no Laboratório de Microeletrônica da USP e foi feito por um técnico com grande experiência em soldagem.

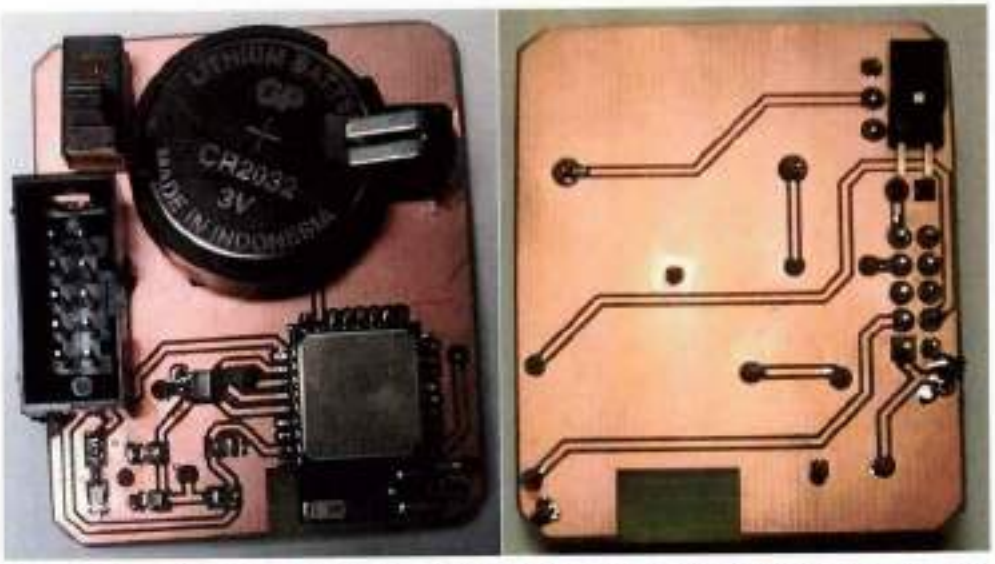

Figura 16 - Breakout Board v2 finalizada. Figura da camada superior e inferior, respectivamente

5 SOFTWARE

O desenvolvimento de *software* compõe grande parte do escopo do projeto PHARI pois está presente em duas frentes: na programação do aplicativo para *smartphones* e na programação do módulo BLE112-A. Cada uma dessas frentes seja descrita abaixo em separado.

5.1. Programação do módulo transmissor BLE112-A

Para a programação do módulo BLE112, são utilizados 3 programas diferentes. O primeiro é um editor de código, utilizamos o *NotePad++*, recomendado pela *Bluegiga*. O segundo programa é o *SmartRF™ Flash Programmer*, da Texas Instruments, cuja função é se comunicar com o *CC Debugger* e fazer a leitura e escrita na memória do *chip*. Por último temos o software da própria *Bluegiga*, o *BLE SW Update Tool*, que é o responsável por fazer a programação do módulo. Esse terceiro programa utiliza o segundo automaticamente para a programação, portanto o *SmartRF* deve ser instalado, apesar de não ter sido utilizado diretamente pelo grupo.

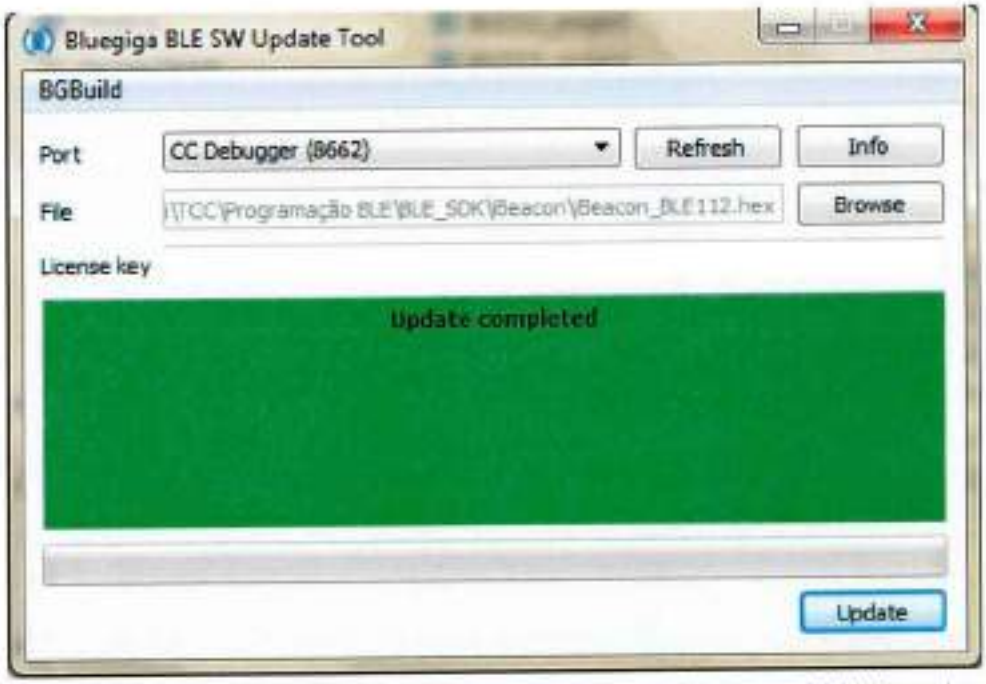

Figura 17 - Programa BLE SW Update Tool ao passar a programação para o módulo transmissor

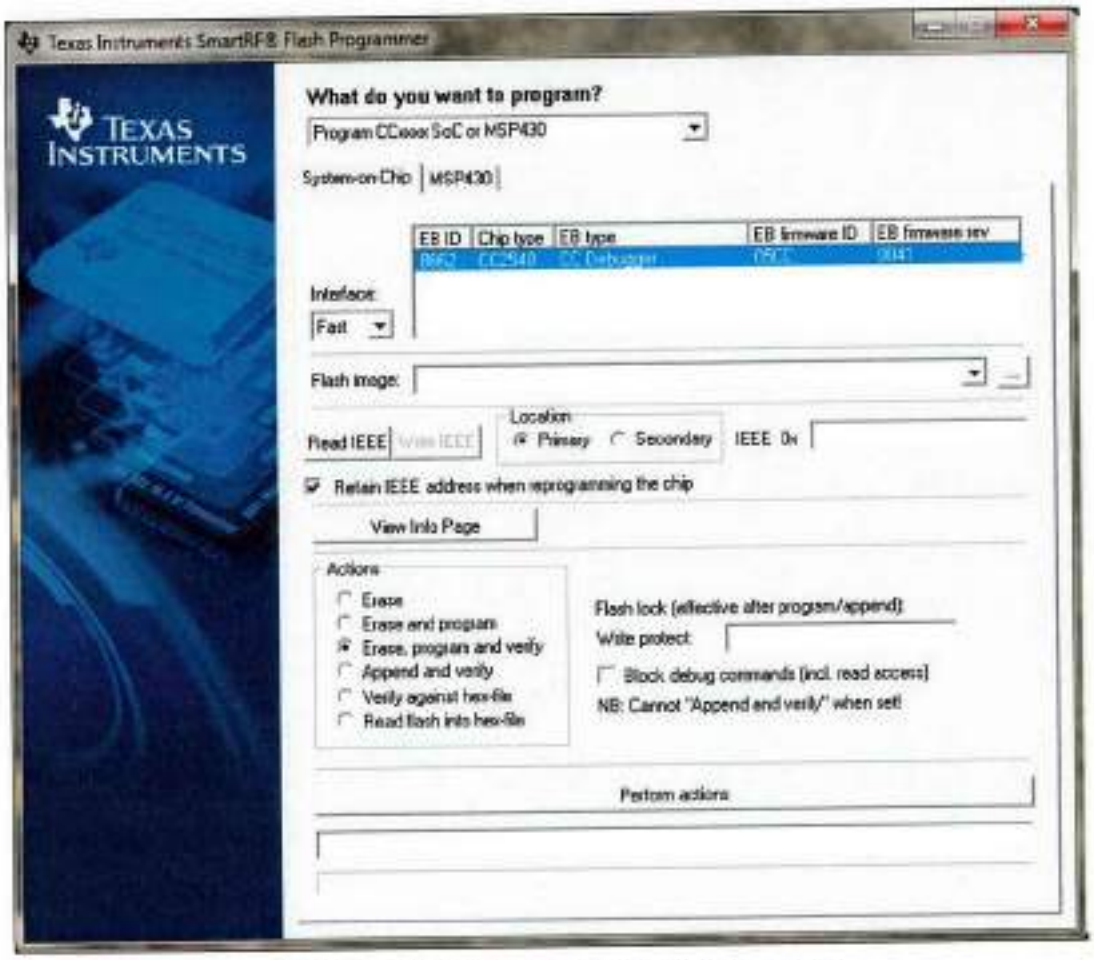

Figura 18 - Programa Smart RFTM Flash Programmer

A programação do módulo BLE112 é feita da seguinte maneira: cria-se um arquivo de extensão *.bgproj*, no formato *xml*, que contém a descrição dos arquivos utilizados, arquivo de saída e definição de alguns parâmetros necessários para o correto funcionamento de módulo transmissor. A figura 24 mostra o arquivo *Beacon_project.bgproj* utilizado no projeto.

Beacon_project.bgproj

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project>
3      <gatt in="gatt112.xml" />
4      <hardware in="hardware.xml" />
5      <script in="Beacon.bgs" />
6      <image out="Beacon.hex" />
7      <device type="ble112" />
8      <boot fw="bootuart" />
9  </project>
```

Figura 19 - Arquivo Beacon_project.bgproj

Na figura, podemos ver que o primeiro arquivo que é mencionado é o gatt.xml. A comunicação *Bluetooth Low Energy* possui um protocolo que é chamado de *Attribute Protocol* e esse protocolo pede que todo sistema que utiliza *Bluetooth Low Energy* tenha um arquivo que descreva as características do sistema, que se chama GATT *profile*.

Em seguida podemos ver o arquivo *hardware.xml*, cuja finalidade é descrever os aspectos físicos que serão utilizados pelo módulo, como frequência de oscilação, potência de transmissão, se será utilizada comunicação USB, entre outros pontos.

O último arquivo utilizado é o *Beacon.bgs*. Esse arquivo contém a programação das funcionalidades que serão passadas para o módulo. A linguagem utilizada é chamada de *bgscript* e foi criada pela empresa *Bluegiga* para programação de seus *chips*.

5.1.1. *Firmware* para envio de UUID, *major* e *minor* e temperatura ambiente

Como já foi dito anteriormente, alguns arquivos compõem a programação do módulo transmissor. No arquivo *hardware.xml* ocorre a definição da potência de transmissão e, como foi utilizado o sensor de temperatura, também é definida a comunicação do sensor com o BLE112, que é feita via SPI (*Serial Peripheral*

Interface), protocolo que permite a comunicação de microcontroladores com outros componentes.

O código do arquivo *hardware.xml* está a seguir. A trecho do código que define a potência de transmissão é *<txpower power="15" bias="5" />*, onde o valor que é colocado pode variar de 0 a 15, onde 0 indica a menor potência (aproximadamente -50 dbm) e 15 a maior potência (0 dbm). A comunicação SPI é definida no trecho *<usart channel="1" mode="spi_master" alternate="1" polarity="negative" phase="0" endianness="msb" baud="57600" endpoint="none" />*, onde o valor de canal e *alternate* são definidos no *datasheet* do BLE112, *endianness* é a ordem de recebimento dos bits, onde *msb (Most Significant Bit)* indica que começa pelo bit mais significativo. Por fim, polaridade e fase representam a borda do *clock* na qual ocorre a leitura (subida ou descida) e *baud* é a taxa de transferência.

```
<?xml version="1.0" encoding="UTF-8" ?>

<hardware>
  <sleeposc enable="true" ppm="30" />
  <usb enable="false" />
  <txpower power="15" bias="5" />
  <port index="0" tristatemask="0" pull="up" />
  <usart channel="1" mode="spi_master" alternate="1" polarity="negative" phase="0"
endianness="msb" baud="57600" endpoint="none" />
  <script enable="true" />
</hardware>
```

Figura 20 - Arquivo hardware.xml

O envio da temperatura lida no sensor é feito através do arquivo *gatt.xml*, que serve para descrever os serviços (envio da temperatura) e características (nome, versão, etc) do *beacon*. O trecho do *gatt.xml* que faz a transmissão do valor lido pelo sensor de temperatura está a seguir. Tanto o serviço quanto a característica têm seu UUID próprio. O valor da característica, visto no código como *id*, recebe o valor *t_data* originado do arquivo *beacon.bgs* e seu valor é escrito em hexadecimal.

```
<service uuid="772c4dd9-1001-47b1-b154-4d8e3e51bb94">
        <description>Sensor de Temperatura</description>

        <characteristic uuid="1848d712-75cf-4eed-9986-ee250cead538" id="t_data">
                <properties read="true" write="true"/>
                <value type="hex">0000</value>
        </characteristic>
</service>
```

Figura 21 - Trecho do arquivo gatt.xml

No arquivo beacon.bgs é onde ocorre a programação do BLE112. Nele é definido o intervalo de transmissão, valores de UUID, *major* e *minor*, ligação do LED e leitura do sensor de temperatura. No código, estão comentadas as principais funções do programa. A trecho onde é definido o intervalo de transmissão e canais de transmissão funciona da seguinte maneira: o intervalo é calculado com o valor utilizado vezes 625 µs, ou seja, 200*625 = 125ms, e a seleção dos canais de transmissão é feita através de uma máscara em hexadecimal, onde cada bit representa 1 canal, 0b111 (todos os canais setados) = 0x7.

```
# ADVertizement data
dim advdata(30)
dim tmp(10)
dim resulta
dim channel
dim tlen

# system boot event listener
event system_boot(major, minor, patch, build, ll_version, protocol_version, hw)

        # Seta o intervalo de transmissão em 125ms.
        # Seleciona os 3 canais de transmissão para serem utilizados
        call gap_set_adv_parameters(200, 200, 7)

        # Configura os pinos P0.6 (LED), P0.4(Saída do Master_SPI), P0.3(Clock) e
P0.2(Chip Select) como saída e P0.5 (Entrada do Master_SPI) como entrada (saída do
sensor)
        call hardware_io_port_config_direction(0, $5c)
        call hardware_io_port_write(0, $4, $4)
        call hardware_io_port_write(0, $40, $40)


        # Leitura inicial do sensor
        call hardware_io_port_write(0, $4, $0)
        call hardware_spi_transfer(1, 2, "\x00\x00")(resulta, channel, tlen, tmp(0:tlen))
```

```
call hardware_io_port_write(0, $4, $4)
call attributes_write(t_data, 0 , 2, tmp(0:2))

# Inicializa o timer para fazer leitura do sensor de 10 em 10 segundos
call hardware_set_soft_timer(327680, 0, 0)

# Initialize iBeacon ADV data
# Flags = LE General Discovery, single mode device (02 01 06)
advdata(0:1) = $02
advdata(1:1) = $01
advdata(2:1) = $06
# Manufacturer data
advdata(3:1) = $1a
advdata(4:1) = $ff
# Preamble
advdata(5:1) = $4c
advdata(6:1) = $00
advdata(7:1) = $02
advdata(8:1) = $15

# UUID: e2c56db5-dffb-48d2-b060-d0f5a71096e0
advdata(9:1) = $e2
advdata(10:1) = $c5
advdata(11:1) = $6d
advdata(12:1) = $b5
advdata(13:1) = $df
advdata(14:1) = $fb
advdata(15:1) = $48
advdata(16:1) = $d2
advdata(17:1) = $b0
advdata(18:1) = $60
advdata(19:1) = $d0
advdata(20:1) = $f5
advdata(21:1) = $a7
advdata(22:1) = $10
advdata(23:1) = $96
advdata(24:1) = $e0

# Major : 08 49
advdata(25:1) = $03
advdata(26:1) = $51

# Minor : 45 17
advdata(27:1) = $11
advdata(28:1) = $a5

# TX Power : -61
advdata(29:1) = $c3

# Seta os dados a serem transmitidos
call gap_set_adv_data(0, 30, advdata(0:30))

#Liga o modo advertisement
call gap_set_mode(4, gap_undirected_connectable)
```

```
end

# Evento recorrente do Temporizador
event hardware_soft_timer(handle)

        call hardware_io_port_write(0, $4, $0) #libera leitura do sensor
        call hardware_spi_transfer(1, 2,"\x00\x00")(resulta,channel,tlen,tmp(0:tlen))
        call hardware_io_port_write(0, $4, $4) #desliga leitura do sensor
        call attributes_write(t_data, 0 , 2, tmp(0:2)) #escreve o valor lido no gatt.xml

end

# Disconnection event listener
event connection_disconnected(handle, result)
        call gap_set_adv_parameters(20, 100, 7)
        #set to advertising mode - with user data
   call gap_set_mode(4, gap_undirected_connectable)
end
```

Figura 22 - Arquivo beacon.bgs

5.2. Programação dos Aplicativos

Para a programação dos aplicativos foram utilizados dois softwares: o *Xcode*, plataforma de desenvolvimento de aplicativos para sistemas iOS e *Beacondo*, plataforma de desenvolvimento especifica para aplicativos que fazem uso de *beacons*.

O *Xcode* foi criado pela *Apple* e utiliza as linguagens *Objective-C* e *Swift*. As telas do aplicativo podem ser montadas através de codificação ou com o uso de uma interface gráfica, o que torna essa tarefa mais simples e rápida para desenvolvedores inexperientes. Já a implementação de ações e funções deve ser feita totalmente através de codificação, o que pode ser um empecilho para desenvolvedores de primeira viagem, já que essa codificação deve ser feita com a linguagem proprietária da *Apple*.

Figura 23 - Interface do software Xcode

O *Beacondo* é um software criado para o desenvolvimento de aplicativos que fazem uso de *beacons* e códigos de barras. Através de uma interface gráfica simples, ele permite que as telas e ações do aplicativo sejam criadas sem que haja a necessidade de escrever qualquer linha de código. Desse modo, um desenvolvedor novato pode criar um aplicativo básico em poucas horas. Apesar de não ser altamente personalizável como o *Xcode*, é uma ferramenta extremamente útil para o desenvolvimento de protótipos.

Figura 24 - Interface do software Beacondo

5.2.1. Aplicativo indicador de distância e temperatura

O desenvolvimento deste aplicativo não tem como objetivo simular uma situação real de uso pelo usuário final, mas visa demonstrar o funcionamento básico dos *beacons* desenvolvidos pelo grupo. Com ele, poderemos verificar a intensidade do sinal recebido com a variação da distância, além de testar o sensor de temperatura. A variação na intensidade do sinal recebido pelo *smartphone* é sinalizada através da mudança de cores na tela, conforme a legenda integrada na interface do aplicativo.

Figura 25 - Interface do aplicativo indicador de distância

Abaixo foi inserido o código que monitora a intensidade do sinal do *beacon* Phari. Para monitorar os beacons Cycle e Brite foi utilizado um código muito similar, somente com a variação do *major* e *minor* corresponde de cada *beacon*. Desse modo optamos por omitir os outros arquivos.

No início do código é definida a região do *beacon*, caracterizada pelo UUID, *major* e *minor*, assim como de um identificador, que nada mais é que o nome do *beacon* escolhido pelo grupo. Ao definir uma região, evitamos que o aplicativo detecte outros *beacons* e erroneamente faça a mudança das cores na tela. Em seguida, é definida a relação cor/intensidade do sinal. Por fim, através de uma função padrão criada pela *Apple*, monitoramos a região do *beacon* que foi criada no início e alteramos a cor da tela de acordo com a intensidade do sinal.

```swift
//  firstContainer.swift
//  Indicador de distancia
//
//  Created by Felipe on 12/4/15.
//  Copyright © 2015 Tevsi LLC. All rights reserved.
//

import Foundation
import UIKit
import CoreLocation

class contPhari: UIViewController, CLLocationManagerDelegate {

    let locationManager = CLLocationManager()
    let region = CLBeaconRegion(proximityUUID: NSUUID(UUIDString: "e2c56db5-dffb-48d2-b060-d0f5a71096e0")!,
        major: 795, minor: 8529, identifier: "Phari")
    let colors = [
        CLProximity.Unknown: UIColor(red: 150/255, green: 150/255, blue: 150/255, alpha: 1),
        CLProximity.Far: UIColor(red: 255/255, green: 102/255, blue: 102/255, alpha: 1),
        CLProximity.Near: UIColor(red: 255/255, green: 255/255, blue: 135/255, alpha: 1),
        CLProximity.Immediate: UIColor(red: 155/255, green: 255/255, blue: 155/255, alpha: 1)
    ]

    override func viewDidLoad() {
        super.viewDidLoad()
        locationManager.delegate = self
        if (CLLocationManager.authorizationStatus() != CLAuthorizationStatus.AuthorizedWhenInUse) {
            locationManager.requestWhenInUseAuthorization()
        }
        locationManager.startRangingBeaconsInRegion(region)

    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    func locationManager(manager: CLLocationManager, didRangeBeacons beacons: [CLBeacon], inRegion
        region: CLBeaconRegion) {
        if (beacons.count > 0) {
            let beaconPhari = beacons[0] as CLBeacon
            self.view.backgroundColor = self.colors[beaconPhari.proximity]
        }
    }

}
```

Figura 26 - Código desenvolvido em linguagem Swift, para o beacon Phari

5.2.2. Aplicativo provedor de informações para feiras e eventos

Para exemplificar uma das funcionalidades que podem ser implementadas através do uso de *beacons*, foi desenvolvido um aplicativo com o software *Beacondo*. Neste aplicativo, foi simulado como seria uma feira com exposição de diversos projetos, onde cada projeto possui um *beacon* posicionado em seu *stand*. Desse modo quando o usuário se aproxima de um *stand*, o aplicativo identifica o sinal enviado pelo *beacon* e envia uma notificação e uma sequência de informações pré-programadas para o usuário.

Para que pudéssemos propiciar uma sensação de uso real, foram fabricados 3 *beacons*, onde cada um atende a um dos projetos de trabalho de conclusão de curso: Phari, Cycle e Brite. Estes grupos foram escolhidos pois são orientados pelos mesmos professores. Todos os *beacons* possuem o mesmo UUID (*E2C56DB5-*

DFFB-48D2-B060-D0F5A71096E0), porém o *major* e *minor* diferem entre eles da seguinte maneira:

- Beacon PHARI:
  - major: 795;
  - minor: 8529;
- Beacon CYCLE:
  - major: 0;
  - minor: 0;
- Beacon BRITE:
  - major: 849;
  - minor: 4517.

A programação foi feita de modo que o usuário só fosse notificado quando estivesse próximo ao *stand*, ou seja, quando a potência do sinal *bluetooth* captado pelo smartphone fosse igual ou superior ao equivalente a 4 metros de distância do *beacon*. No momento em que a intensidade do sinal bluetooth ultrapassa o limite definido, o usuário é notificado, mesmo que o aplicativo esteja fechado. Ao clicar na notificação, o aplicativo abre em sua tela principal, onde existe outra notificação informando qual *beacon* foi encontrado. Optamos por nomear cada *beacon* com o nome do grupo ao qual ele é responsável por fornecer as informações.

Figura 27 - Notificações fora e dentro do aplicativo, respectivamente

Ao clicar na notificação, o usuário é direcionado para a página do grupo. Nesta página existe uma sequência de fotos e vídeos, um breve resumo do trabalho, nome dos integrantes do grupo e nome dos orientadores. Caso o usuário possua interesse em guardar esta página, existe no canto superior direito a opção de adicioná-la aos favoritos. Desse modo mesmo que o aplicativo seja fechado, o usuário poderá consultar esta tela posteriormente. Ao adicionar à lista de favoritos, o aplicativo avisa que a ação foi executada com sucesso através de uma notificação com *feedback*.

Figura 28 -Tela com informações do grupo e opção de adicionar aos favoritos

Ao retornar para o menu principal, o usuário pode realizar 3 ações: ir para a lista de favoritos; ir para o indicador de distância, onde é indicado, através do uso de cores, a distância do *beacon* Phari; ou ir para a tela Sobre, onde é explicado o funcionamento do aplicativo.

Figura 29 - Menu principal, indicador de distância e Sobre, respectivamente

Por fim, temos a lista de favoritos. Nela são armazenados todos os projetos salvos pelo usuário. Como são utilizados dados persistentes, a lista não se perde quando o aplicativo é fechado. Caso haja interesse em apagar somente um ou todos os projetos favoritados, isso também pode ser feito e está exemplificado na imagem abaixo.

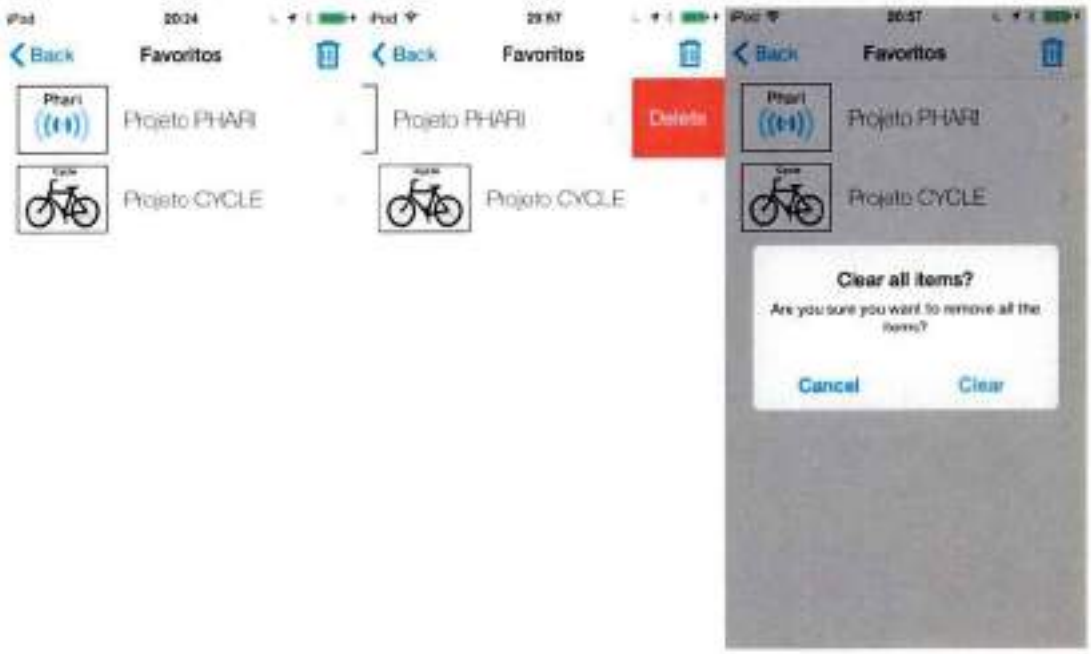

Figura 30 - Lista dos favoritos e opção de apagar um ou todos da lista

6 RESULTADOS

Como a maior parte esforço dedicado a este projeto foi feito na parte de software, com a criação dos aplicativos e otimizações de código, não é pertinente listar linha por linha de código e as modificações que foram necessárias para chegar nos resultados discutidos a seguir. Desse modo, para evitar que a descrição dos resultados se tornasse muito extensa e exaustiva, serão listados principalmente os pontos mais relevantes referente ao escopo do *hardware* do projeto.

6.1. Verificação do funcionamento da *breakout board v1*

A fabricação e soldagem da *breakout board v1* foi concluída antes de finalizarmos a codificação do primeiro aplicativo. Para que pudéssemos dar andamento no projeto e verificar de antemão se o design, fabricação e soldagem dos componentes havia sido bem sucedida, realizamos a programação do módulo transmissor BLE112. Através de um aplicativo chamado *Locate Beacon*, disponível na *App Store* da *Apple*, pudemos comprovar que tanto a programação do BLE112 quanto a breakout board estavam funcionando corretamente.

Figura 31 - Beacon identificado pelo aplicativo Locate

6.2. Aplicativo básico indicativo de distância para um *beacon*

Após finalizar a codificação do primeiro aplicativo (apêndice F.1), que era capaz de detectar somente um *beacon*, exibir a distância na tela do *smartphone* através das mensagens muito próximo, próximo, distante e desconhecido e criar um log no *software Xcode*, com a indicação do UUID, *major*, *minor*, potência e distancia aproximada em metros.

Logo durante os primeiros testes, percebemos uma falha no design da *breakout board v1*: a ausência de um botão liga/desliga. O ato de colocar e tirar a bateria do soquete inúmeras vezes causou o descolamento da trilha na qual o soquete estava soldado. Com isso foi necessário realizar um reparo, como pode ser visto na foto abaixo.

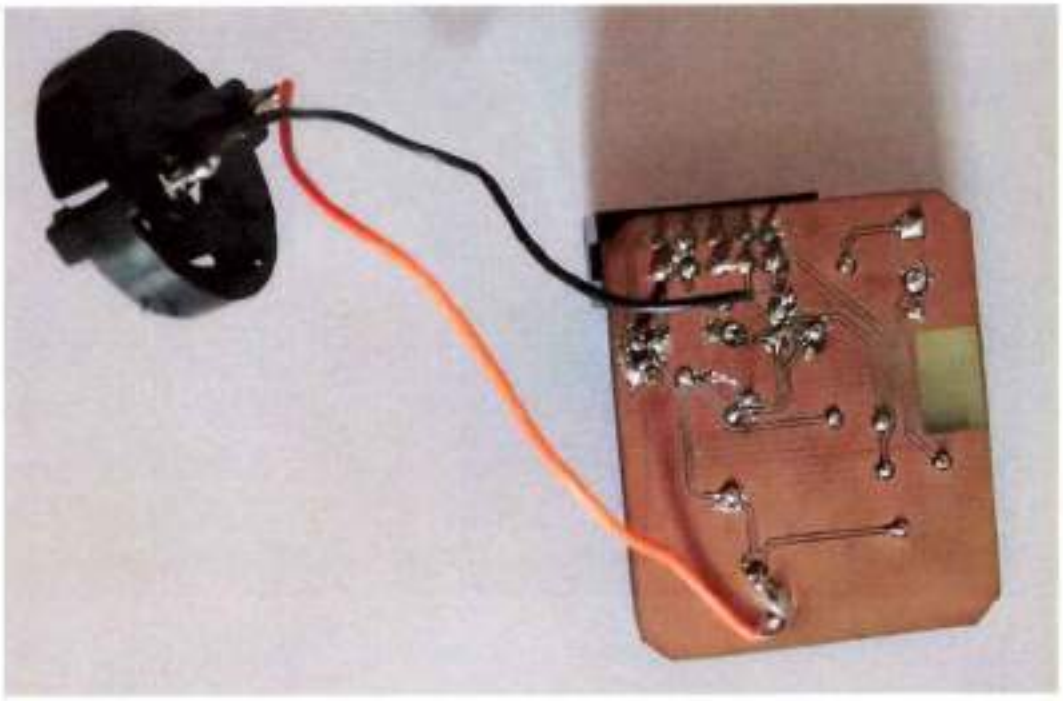

Figura 32- Breakout Board v1 após os testes

Apesar deste contratempo, o aplicativo funcionou sem maiores problemas e pudemos comprovar que não havia erros significativos de projeto. Deste ponto em diante, a base do projeto estava pronta e somente restava realizar melhorias na *breakout board* e desenvolver um aplicativo para demonstrar uma situação real de uso dos *beacons*.

6.3. Aplicativo com notificações para 3 *beacons*

Após corrigir os problemas ne design da *breakout board v1*, fabricamos outras duas placas, totalizando 3 *beacons*. Com isso pudemos codificar um aplicativo que identifica 3 *beacons* diferentes e para um deles exibe uma determinada notificação quando entra em sua região de transmissão, como por exemplo "Bem-vindo" e "Sessão de frutas".

Este foi um importante passo para o projeto pois mostrou que estávamos no caminho correto para a implementação de um aplicativo que executasse determinada ação para cada região de transmissão que o usuário entrasse

6.4. Aplicativo indicador de distância para 3 *beacons* com sensor de temperatura

Seguindo o mesmo princípio de funcionamento do aplicativo do item 6.2, foi desenvolvido um aplicativo que mostra para o usuário ao mesmo tempo a medição de distância dos 3 beacons do projeto. Para evitar que tivéssemos mais um aplicativo cuja função fosse somente indicar a temperatura ambiente, decidimos integrar esse funcionalidade neste aplicativo que já estava pronto.

Para que o beacon enviasse, além do UUID, *major* e *minor*, a temperatura medida pelo sensor, foi preciso modificador a código do *firmware* do BLE112. Ao encontrar dificuldades para fazer o envio da temperatura do sensor para o microcontrolador, notamos que dois pinos do sensor ADT7301 estavam invertidos. Ao romper umas das trilhas e usar um *jumper*, pudemos solucionar esse problema. Prontamente o microcontrolador passou a identificar o sensor de temperatura e a adição da temperatura ambiente no aplicativo foi de fácil implementação.

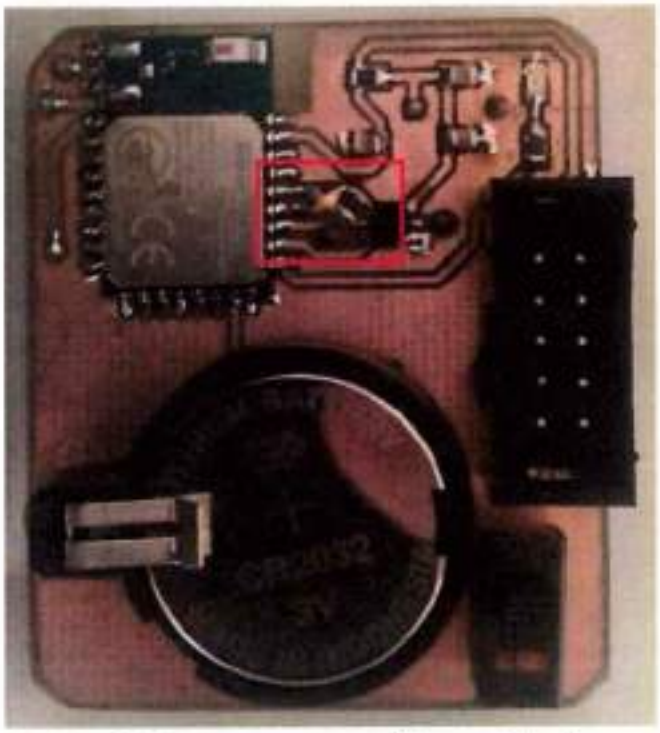

Figura 33 - Breakout Board 2. Destaque para o reparo feito nas trilhas do sensor de temperatura

A fim de verificar a variação da distância com a mudança de cor, foram feitas as seguinte medições, para uma linha de visão livre de obstáculos (*Line of Sight*):

- Muito próximo: d<0,5 m;
- Próximo: 0,5<d<4,5m;
- Distante: 4,5<d<12,5m;
- Desconhecido: d>12,5m.

A partir destas medições, foi desenvolvido o aplicativo provedor de informações para feitas e eventos, onde programamos para que o usuário fosse notificado somente quando entrasse na região definida como próxima, isto é, quando a distância entre o smartphone o beacon fosse inferior a cerca de 4 metros.

CONCLUSÃO

Ao final do período de duração do projeto, os objetivos propostos foram atingidos. Desenvolvemos um protótipo funcional do *beacon*, que poderia ser utilizado comercialmente, assim como um aplicativo que mostra claramente um dos usos que poderia ser dado a esta nova tecnologia.

Na frente de hardware, obtivemos uma versão final do *beacon* com diversas melhorias em relação à versão inicial, como redução do tamanho, adição/remoção de componentes com o objetivo de melhorar o desempenho e reduzir complexidade de fabricação e soldagem, remoção de trilhas redundantes e adição de planos VCC e GND. Por fim, optamos por adicionar um sensor de temperatura, funcionalidade pouco utilizada e que diferencia em relação aos outros *beacons* existentes no mercado. Como reduzimos ao máximo as dimensões da placa, isso incluiu reduzir a grossura das trilhas e aproximar os componentes, o que levou a uma dificuldade adicional ao processo de fabricação e soldagem dos componentes, pois foi a primeira placa de circuito impressa produzida pelo grupo que utilizava componentes SMD.

Na frente de software, desenvolvemos o *firmware* para que o módulo transmissor BLE112 atuasse como *beacon* e implementamos o protocolo de comunicação entre o sensor de temperatura e o microcontrolador. Também concluímos o projeto com 2 aplicativos, onde um deles demonstra como seria a experiência de um usuário em uma feira, e outro aplicativo cuja finalidade é demonstrar a outros desenvolvedores e pessoas com conhecimento técnico como o sinal enviado por beacons se comporta em uma situação real.

Os maiores desafios encontrados foram com relação ao desenvolvimentos dos aplicativos. Apesar dos aplicativos criados apresentarem baixa complexidade, devido à falta de experiência dos integrantes do grupo com as linguagens Swift e Objective-C, foi necessária uma curva de aprendizado bastante íngreme para chegar aos resultados descritos nas páginas anteriores. O uso do *software Beacondo* permitiu que criássemos um aplicativo com mais funcionalidades e de maneira mais ágil se comparado ao desenvolvimento somente através de codificação. No entanto, parte do tempo ganho foi gasto em uma maior bateria de testes em busca erros que poderiam aparecer durante o uso do aplicativo pelo usuários finais. Apesar desses

empecilhos descritos acima, ao fim do projeto obtivemos dois aplicativos que evidenciam o potencial que os *beacons* têm de fazer parte do dia-a-dia dos usuários de *smartphones* em um futuro próximo, seja em automatização residencial, fornecendo informações em feiras e museus, oferecendo promoções ou outros usos que ainda estão por surgir nesse novo mercado.

REFERÊNCIA BIBLIOGRÁFICA

INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS. **IEEE Std 1233 Guide for Developing System Requirements Specifications.** Nova Iorque, 1998.

Ibeacon insider. **What is iBeacon? What are iBeacons?**. Disponível em: <http://www.ibeacon.com/what-is-ibeacon-a-guide-to-beacons/>. Acesso em: 27 de março de 2015.

GRUMAN, G. **What you need to know about using Bluetooth beacons**. Disponível em: <http://www.infoworld.com/article/2608498/mobile-apps/what-you-need-to-know-about-using-bluetooth-beacons.html>. Acesso em: 01 de abril de 2015.

GIBBS, C. **The near future for beacons may not lie in retail**. Disponível em: <http://research.gigaom.com/2014/12/the-near-future-for-beacons-may-not-lie-in-retail/>. Acesso em: 03 de abril de 2015.

ANDREWS, D. **Predictions For The Future Of Beacons**. Disponível em: <http://www.mediapost.com/publications/article/237653/predictions-for-the-future-of-beacons.html>. Acesso em: 03 de abril de 2015.

SMITH, D. **Beacons Are Going To Shape The Future Of Shopping**. Disponível em: <http://www.businessinsider.com/beacons-shopping-2015-1>. Acesso em: 03 de abril de 2015.

DANOVA, T. **BEACONS: What They Are, How They Work, And Why Apple's iBeacon Technology Is Ahead Of The Pack**. Disponível em: <http://www.businessinsider.com/beacons-and-ibeacons-create-a-new-market-2013-12>. Acesso em: 07 de abril de 2015.

AISLELABS. **The Hitchhikers Guide to iBeacon Hardware: A Comprehensive Report by Aislelabs (2015).** Disponível em: <http://www.aislelabs.com/reports/beacon-guide/>. Acesso em: 8 de abril de 2015.

NEALE-MAY, D. **DIRECT MARKETING**. Disponível em: <https://www.cmocouncil.org/facts-stats-categories.php?view=all&category=direct-marketing>. Acesso em: 22 de junho de 2015.

BLUEGIGA. **BLE112 Bluetooth Smart Module**. Disponível em: <https://www.bluegiga.com/en-US/products/ble112-bluetooth-smart-module/>. Acesso em: 22 de junho de 2015.

TEXAS INSTRUMENTS. **CC Debugger User's Guide**. Disponível em: <http://www.ti.com/lit/ug/swru197h/swru197h.pdf>. Acesso em: 23 de junho de 2015.

PORTAL EDUCAÇÃO. **Árvores de problemas e objetivos**. Disponível em: <http://www.portaleducacao.com.br/pedagogia/artigos/42842/arvores-de-problemas-e-objetivos>. Acesso em: 10 de novembro de 2015.

Smartphone OS Market Share, 2015 Q2. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: 12 de novembro de 2015.

REIS, T. **5 dicas para facilitar a criação de uma Estrutura Analítica de Projeto**. Disponível em: <http://www.projectbuilder.com.br/blog-pb/entry/projetos/5-dicas-para-facilitar-a-criacao-de-uma-estrutura-analitica-de-projeto>. Acesso em: 12 de novembro de 2015.

GRAFISCOPIO. **Carta Gantt: para qué sirve y cómo hacerla**. Disponível em: <http://www.grafiscopio.com/carta-gantt-para-que-sirve-y-como-hacerla/>. Acesso em: 12 de novembro de 2015.

DAGES, W. **Getting Started with iBeacon: A Swift Tutorial**. Disponível em: < http://willd.me/posts/getting-started-with-ibeacon-a-swift-tutorial>. Acesso em: 04 de dezembro de 2015.

Beacondo. Disponível em: < http://www.beacondo.com/>. Acesso em: 04 de dezembro de 2015.

ROWBERG, J. **Maximize throughput with BLE modules**. Disponível em: <https://bluegiga.zendesk.com/entries/22400867--HOW-TO-Maximize-throughput-with-BLE-modules>. Acesso em: 04 de dezembro de 2015.

APÊNDICE A: ÁRVORE DE OBJETIVOS

Árvore de objetivos consistem em representações gráficas dos objetivos do projeto. Ela é composta do tronco, que simboliza o objetivo principal e de raizes, que representam os meios que serão utilizados para atingir esse objetivo.

A figura a seguir mostra a árvore de objetivos que foi construída e reflete o projeto abordado:

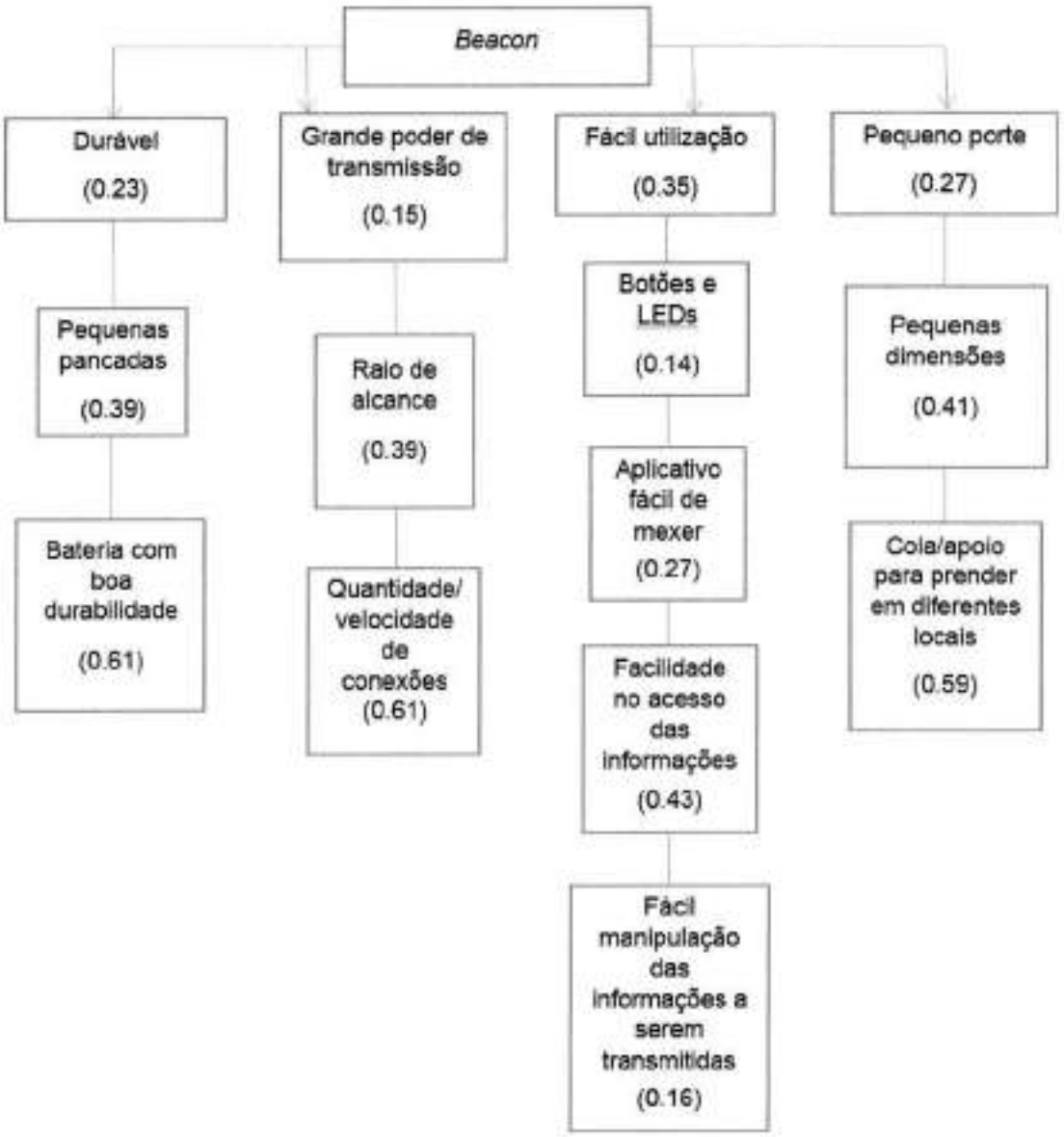

Figura 34 - Árvore de Objetivos com pesos definidos

Tabela 1 - Pesos do 1º nível da árvore de objetivos

	Tecnologia	Durável	Fácil Uso	Pequeno Porte	Mg	ω
Tecnologia	1	0.5	0.3	0.5	0.54	0.15
Durável	2	1	0.5	0.5	0.84	0.23
Fácil Uso	3	1	1	1	1.32	0.35
Pequeno Porte	2	1	0.5	1	1	0.27
						Σ = 1

Tabela 2 - Pesos do tema Tecnologia

	Raio de alcance	Quantidade/velocidade de conexões	Mg	ω
Raio de alcance	1	0.5	0.84	0.39
Quantidade/velocidade de conexões	3	1	1.32	0.61
				Σ = 1

Tabela 3 - Pesos do tema Durável

	Pequenas Pancadas	Bateria Durável	Mg	ω
Pequenas Pancadas	1	0.5	0.84	0.39
Bateria Durável	3	1	1.32	0.61
				Σ = 1

Tabela 4 - Pesos do tema Fácil Uso

	Botões e LED's	Aplicativo de Fácil Utilização	Facilidade de acesso às informações	Facilidade na manipulação das informações	Mg	ω
Botões	1	0.5	0.5	0.5	0.59	0.14
Aplicativo	2	1	0.5	2	1.19	0.27
Acesso	3	2	1	2	1.86	0.43
Manipulação	1	0.5	0.5	1	0.71	0.16
						$\Sigma = 1$

Tabela 5 - Pesos do tema Pequeno Porte

	Pequenas Dimensões	Facilidade para prender em diferentes locais	Mg	ω
Pequenas Dimensões	1	0.5	0.84	0.41
Facilidade em prender	2	1	1.19	0.59
				$\Sigma = 1$

APÊNDICE B: REQUISITOS DO SISTEMA

Os Requisitos do Sistema têm, tradicionalmente, sido visto como um documento que informa os requisitos definidos pelos usuários à comunidade técnica, que irá especificar e construir o sistema. Estes requisitos atuam como uma ponte entre os dois grupos e devem ser definidos claramente para que ambos os grupos tenham pleno conhecimento das metas que são possíveis atingir durante o desenvolvimento do projeto, evitando que ocorram desentendimentos posteriormente.

Abaixo foram especificados os requisitos do produto que está sendo proposto, onde os usuários são responsáveis por fornecer os requisitos de marketing e a comunidade técnica os requisitos de engenharia do produto.

1. Requisitos de Engenharia

O módulo *bluetooth* deve:

1. ser capaz de transmitir e receber dados a uma distância de, no mínimo, 30 metros considerando uma potência de transmissão de -2dbm e em linha reta sem obstáculos (LOS);
2. garantir uma velocidade de transmissão de, no mínimo, 20kbps;
3. fazer *broadcasting* com um intervalo mínimo de 100ms e máximo de 800 ms;
4. operar normalmente com temperaturas de até 60 °C;
5. operar durante, no mínimo, 2 meses somente com o uso de bateria;
6. possuir pelo menos 100 Kb de memória não volátil;

O totem deve:

1. ter um custo final de, no máximo, R$100. O custo final é composto pelo módulo bluetooth, bateria e embalagem protetora de material plástico;
2. possuir um tamanho máximo de 10cm de lado e pesar menos que 200g, permitindo que seja discreto e leve;
3. ser à prova d'água;

2. Restrições Técnicas

    1. o módulo deve utilizar *Bluetooth Low Energy* (*Bluetooth 4.0*) pois esta tecnologia apresenta menor gasto de potência se comparado ao *bluetooth 3.0* e *wireless*. Isso permite que o usuário deixe o bluetooth habilitado durante todo o tempo;

3. Requisitos de Marketing

    1. Durável;
    2. Grande poder de transmissão;
    3. Fácil utilização;
    4. Pequeno porte

Tabela 6 - Tabela de Requisitos

Requisitos de Marketing	Requisitos de Engenharia	Justificativa
2	Ser capaz de transmitir e receber dados a uma distância de, no mínimo, 30 metros.	Deve ser capaz de transmitir e receber informações a uma distância elevada.
2	Garantir uma velocidade de transmissão de, no mínimo, 20kbps.	Deve ser veloz o suficiente para transmitir textos ao aplicativo sem apresentar grandes atrasos.
1	Fazer *broadcasting* com um intervalo mínimo de 100ms e máximo de 800 ms.	O intervalo de *broadcasting* não deve ser muito longo a ponto de causar atrasos e nem muito curto a ponto de gastar muita bateria.
1	Operar normalmente com temperaturas de até 60 °C.	Deve ser capaz de operar tanto dentro de lojas como a céu aberto.
1	Operar durante, no mínimo, 2 meses somente com o uso de	Com baixa corrente de operação, a vida útil da bateria é

	bateria.	maior, evitando que a bateria tenha que ser substituída com muita frequência.
3	Possuir pelo menos 100 Kb de memória não volátil.	Deve possuir memória suficiente para armazenar textos.
3	O totem deve ter um custo final de, no máximo, R$100.	Deve ter um custo baixo para viabilizar a popularização do uso destes dispositivos.
3,4	O totem deve possuir um tamanho máximo de 10cm de lado e pesar menos que 200g.	Pequeno e leve, o dispositivo pode ser instalado sem se destacar do ambiente.
1	Ser à prova d'água.	Para instalações ao ar livre, o dispositivo deve ser capaz de resistir a tempestades.

4. Benchmark Competitivo

O *Benchmark* Competitivo permite estudar e comparar produtos similares que são fabricados por diferentes empresas e estão inseridos em um mesmo mercado, permitindo que se adquira maior conhecimento dos pontos fortes e fracos do produto cujo desenvolvimento está sendo proposto e com isso sejam definidos quais devem ser os pontos de maior atenção.

Tabela 7 - Benchmark Competitivo

	Estimote	*Kontakt*	*Blue Cats*	Nosso
Tamanho(cm)	4.9x9	5.5x5.5x1.5	7.6x2.78	10x10x10
Peso (g)	--	23	50.5	200
Potência de Transmissão (dbm)	+4	-30 a +4	+0	-2
Memória (kB)	256	256	--	100
Preço (R$)	130	100	110	100

APÊNDICE C: GERAÇÃO DE CONCEITOS E SUA AVALIAÇÃO

Após a definição dos Requisitos, o próximo passo consiste em listar as possíveis soluções para os pontos abordados na seção anterior. Em um primeiro momento, é realizada a geração de conceitos, que consiste em desenvolver um diagrama de blocos com as diversas soluções que poderiam ser adotadas, sem realizar uma avaliação previa de sua viabilidade.

Em seguida é feita a avaliação destas soluções através da Avaliação de Forças e Fraquezas, onde são atribuidos pesos e cada conceito gerado é avaliado individualmente e em comparação com outros conceitos.

1. Geração de Conceitos

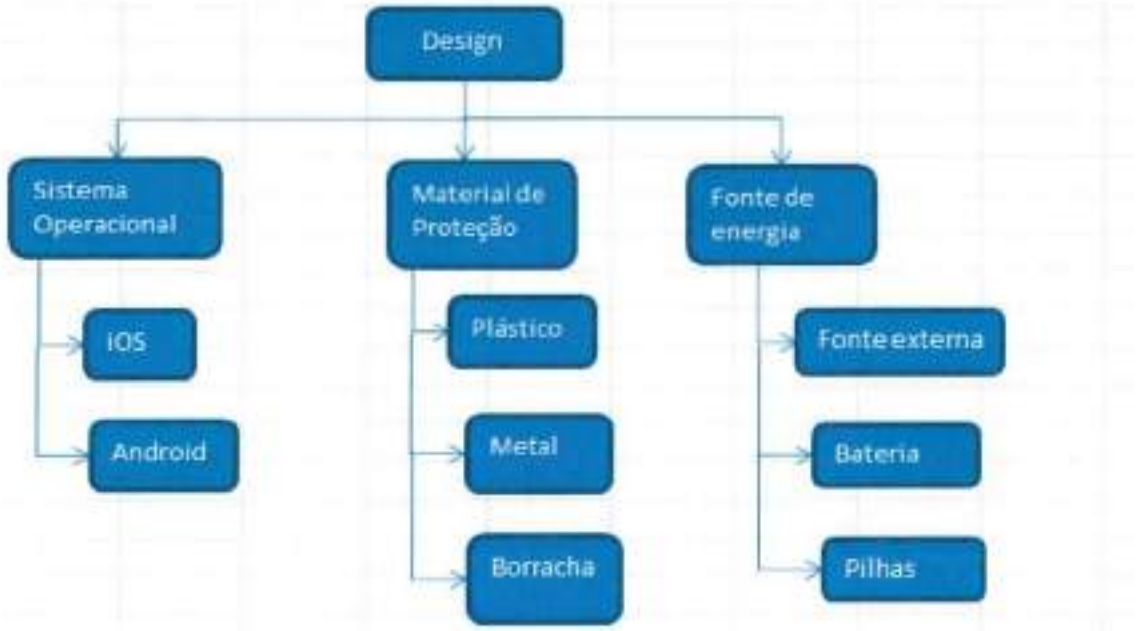

Figura 35 - Diagrama de conceitos

Obtivemos o diagrama acima ao dividir o projeto em três blocos principais: Sistema Operacional, onde ponderamos qual será a interface de software que o usuário utilizará, e em Material de Proteção e Fonte de energia onde são listados os pontos relevantes de *design* relacionados ao hardware do projeto.

Em Sistema operacional, foram escolhidos os dois sistemas que possuem a maior base de usuários. Juntos, Android e iOS somam cerca de 97% dos usuários (dados de 2015T2).

Em Material de Proteção, foram listados os materiais mais comumente utilizados em embalagens protetoras de circuitos eletrônicos. Além disso, os materiais escolhidos são facilmente obtiveis.

E por fim em Fonte de Energia optou-se por comparar os meios mais tradicionais para alimentação de circuitos eletrônicos no mercado.

2. Avaliação de Forças e Fraquezas

Tabela 8 - Avaliação de forças e fraquezas

Método	Forças	Fraquezas
iOS	Público com maior poder aquisitivo (melhor para propaganda) +++	Burocracia de programação -
	Já possuímos material para desenvolvimento em iOS ++	Custo para ser desenvolvedor -
Android	Grande quantidade de exemplos de desenvolvimento ++	Muita variação entre smartphones --
	Não há custo para ser desenvolvedor +	
Plástico	Aguenta pequenas pancadas +	Estética -
	Facilidade de usinagem ++	
	Barato ++	
Metal	Suporta pancadas mais fortes ++	Muita dificuldade de usinagem --
	Estética +	Custo --

Borracha	Suporta pancadas mais fortes ++	Estética --
Fonte externa	Não há necessidade de se trocar bateria/pilha ++	Necessidade de ficar próximo a uma tomada --
		Gasto de espaço na PCB para colocar cabo de tomada --
		Necessidade de um módulo no circuito para baixar a tensão --
Bateria	Durabilidade relativamente alta ++	Necessidade de trocar de tempos em tempos -
	Ocupa pequeno espaço na placa +	
	Custo relativamente baixo ++	
Pilhas	Facilidade na troca +	Durabilidade baixa ---
		Custo relativamente alto --

APÊNDICE D: DECOMPOSIÇÃO FUNCIONAL

O projeto final consiste de uma peça de *hardware* que transmite informação sempre que detecta um novo usuário nas proximidades (*smartphone*). Podemos dividir o projeto em 2 grandes módulos, o desenvolvimento de hardware (*beacon*) e o desenvolvimento de *software* (aplicativo). A decomposição funcional utilizada foi a *Top-down*.

Figura 36 - Nível 0 da decomposição funcional

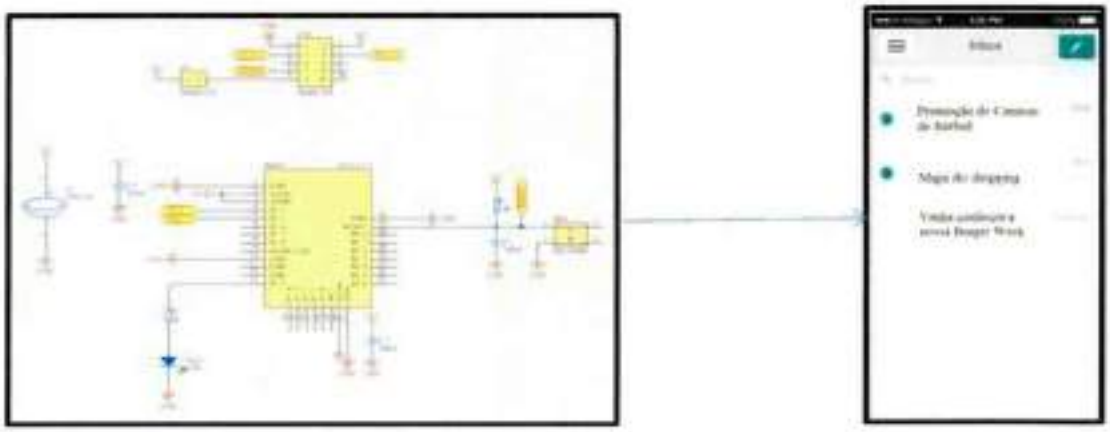

Figura 37 - Níveis 1 e 2 da decomposição funcional

Nas figuras 6 e 7 podemos ver os diferentes níveis da decomposição funcional. No nível 0, temos o projeto como um bloco único que representa o projeto final, tendo como entrada as informações que serão transmitidas e como saída a apresentação dessas informações aos usuários. A figura 7 mostra o nível 1 completo, separado em 2 grandes módulos, desenvolvimento de hardware e desenvolvimento do aplicativo. Dentro do desenvolvimento de hardware vemos o nível 2, que é o esquema elétrico, tendo o módulo de transmissão/processamento, alimentação (bateria), pinos de programação, LED para mostrar funcionamento do sistema e botão liga/desliga.

APÊNDICE E: GERENCIAMENTO DE PROJETO

1. Estrutura Analítica de Projeto (EAP)

EAP é uma ferramenta frequentemente utilizada em gestão de projetos. Ela consiste em decompor o projeto em partes menores a fim de facilitar o seu entendimento e permitir um melhor controle do tempo, escopo e custo. O projeto deve ser decomposto em níveis que permitam o seu fácil gerenciamento, porém a EAP não deve ser demasiadamente detalhada de modo que se torne um novo complicador no processo de gestão.

A gestão de projetos de P&D é de extrema importância, pois assim há um acompanhamento do andamento do projeto. O estabelecimento de responsáveis por cada etapa e de uma metodologia podem ajudar a ter uma ideia de quanto tempo irá levar para certa etapa ficar pronta e para que caso ocorram atrasos, haja um responsável para acelerar ou reportar esse atraso.

A figura a seguir mostra a EAP simplificada para os diferentes módulos do projeto. Ela foi dividida em 3 grupos principais: Apresentações, que é composta pelas entregas dos relatórios, pôster e monografia e pelas apresentações; Aplicativo, que é composto pelo desenvolvimento do *software*; e Módulo, que é composto pelo desenvolvimento do *hardware*.

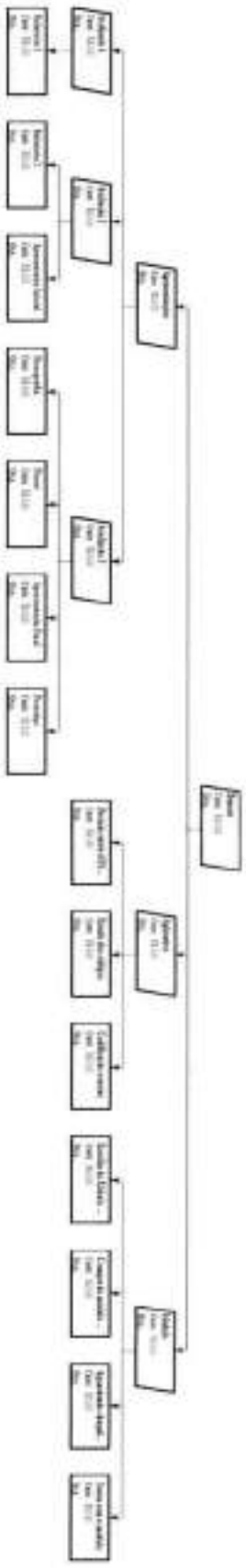

Figura 38 - Estrutura Analítica de Projeto (EAP) desenvolvido

2. Carta de Gantt

Carta de Gantt, assim como a EAP, é uma ferramenta utilizada em gestão de projetos. É utilizada para planejar e programar tarefas que devem ser realizadas durante o decorrer do tempo. Por ser de fácil visualização, permite o acompanhamento e monitoramento de cada etapa do projeto. Na carta de Gantt podemos ver a duração e sequencia das tarefas, além do cronograma completo do projeto e sua data de conclusão.

Na carta de Gantt (figura 9), foi feita a separação em 5 atividades principais: as 3 avaliações, elaboração do aplicativo e elaboração e programação do hardware. Como podemos ver a partir da carta, algumas atividades dependem diretamente da conclusão de outras, de modo que qualquer atraso no cronograma do projeto gera impacto direto no prazo de conclusão.

No caso do *hardware*, temos a definição do módulo de *Bluetooth Low Energy*, a compra do módulo e posteriormente a elaboração do produto. No aplicativo, temos a escolha do sistema operacional, estudo da linguagem de programação e por fim a programação e testes do aplicativo.

3. Análise de Risco

Os maiores riscos estão concentrados no atraso da chegada dos componentes, que por consequência causaria um atraso no inicio dos testes com o módulo e inicio do desenvolvimento do protótipo, afetando a execução da avaliação 3.

Temos também o risco de problemas técnicos, como a dificuldade de programação tanto do *software* (aplicativo) quanto do *hardware* (*beacon*). Ao desenvolver nosso hardware, que terá como função nos auxiliar na programação do módulo transmissor, há o risco de atrasos na produção e envio das placas PCB. Posteriormente, na etapa de soldagem dos componentes, que são do tipo SMD, podem ocorrer danos aos componentes ou até mesmo à placa. Esse risco pode ser mitigado ao encomendar a produção de diversas PCB e compra de componentes extras.

Figura 39 - Carta de Gantt

APÊNDICE F: CÓDIGOS DESENVOLVIDOS NO XCODE

1 Aplicativo indicador de distância para um beacon com log no *Xcode*

O código abaixo foi utilizado durante os primeiros testes. É capaz de monitorar a região de 1 *beacon*, exibir no debugador do *Xcode* se o beacon foi encontrado e mostrar na tela do *smartphone* a distância aproximada através das mensagens muito próximo, próximo, distante e posição desconhecida.

```
//
// ViewController.m
// TesteBeacon
//
// Created by Felipe on 9/22/15.
// Copyright (c) 2015 Felipe & Gabriel Inc. All rights reserved.
//

#import "ViewController.h"

@interface ViewController ()<CLLocationManagerDelegate>

@property (nonatomic, weak) IBOutlet UILabel *messageLabel;


@end

@implementation ViewController

- (void)viewDidLoad
{

  [super viewDidLoad];

  NSUUID *UUID = [[NSUUID alloc] initWithUUIDString:@"e2c56db5-dffb-48d2-b060-
d0f5a71096e0"];
```

```
    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate = self;

    CLBeaconRegion *region = [[CLBeaconRegion alloc] initWithProximityUUID:UUID major:0
minor:0 identifier:@"Phari"];
    [self.locationManager startRangingBeaconsInRegion:region];
    [self.locationManager startMonitoringForRegion:region];

}


- (NSString *)categorizacao:(CLProximity)distancia
{
    switch (distancia)
    {
        case CLProximityFar:
            return @"Beacon distante";
            break;
        case CLProximityImmediate:
            return @"Beacon muito proximo";
            break;
        case CLProximityNear:
            return @"Beacon perto";
            break;
        case CLProximityUnknown:
            return @"Posicao desconhecida";
        break;
    }

}



- (void)locationManager:(CLLocationManager *)manager didRangeBeacons:(NSArray
*)beacons inRegion:(CLBeaconRegion *)region
{
    if (beacons.count > 0)
```

```
    {
        NSLog(@"O beacon Phari foi encontrado.");
    }
    else
        NSLog(@"Nenhum beacon foi encontrado.");


    if (beacons.count > 0)
    {
        CLBeacon *beacon = beacons[0];
        self.messageLabel.text = [self categorizacao:beacon.proximity];
    }

}


- (void)didReceiveMemoryWarning
{

    [super didReceiveMemoryWarning];

}

@end
```

2 Aplicativo indicador de distância com notificações para 3 beacons

O código abaixo foi utilizado para desenvolver o aplicativo que indica a distância do Beacon até o equipamento que roda o aplicativo. Ele é capaz de indicar as posições muito próximo, próximo, distante e posição desconhecida. Também é criado um log que pode ser visto no computador e lista o *UUID*, *Major*, *Minor* e distância em metros. O aplicativo identifica 3 *beacons* que possuem *major* e *minor* únicos e para cada um deles exibe determinada notificação. Foi desenvolvido no software *Xcode* versão 7.0, linguagem *Objective-C*.

```
//
// ViewController.m
// TesteBeacon
//
// Created by Felipe on 9/22/15.
// Copyright (c) 2015 Felipe & Gabriel Inc. All rights reserved.
//

#import "ViewController.h"

@interface ViewController ()<CLLocationManagerDelegate>

@property (nonatomic, weak) IBOutlet UILabel *messageLabel;
@property (nonatomic, weak) IBOutlet UILabel *Label;

@end

@implementation ViewController


- (void)viewDidLoad
{

    [super viewDidLoad];

    NSUUID *UUID = [[NSUUID alloc] initWithUUIDString:@"e2c56db5-dffb-48d2-b060-
d0f5a71096e0"];

    self.locationManager = [[CLLocationManager alloc] init];
    self.locationManager.delegate = self;

    CLBeaconRegion *region = [[CLBeaconRegion alloc] initWithProximityUUID:UUID
major:0 minor:0 identifier:@"Phari"];

    CLBeaconRegion *region2 = [[CLBeaconRegion alloc] initWithProximityUUID:UUID
major:1 minor:1 identifier:@"Phari2"];
```

```
    CLBeaconRegion *region3 = [[CLBeaconRegion alloc] initWithProximityUUID:UUID
major:2 minor:2 identifier:@"Phari3"];

    [self.locationManager startRangingBeaconsInRegion:region];
    [self.locationManager startMonitoringForRegion:region];
    [self.locationManager startRangingBeaconsInRegion:region2];
    [self.locationManager startMonitoringForRegion:region2];
    [self.locationManager startRangingBeaconsInRegion:region3];
    [self.locationManager startMonitoringForRegion:region3];



}

-          (void)          locationManager:(CLLocationManager          *)manager
didEnterRegion:(CLBeaconRegion *) region
{
    UILocalNotification *notification = [[UILocalNotification alloc] init];
    if ([region.major intValue] == 0)&& ([region.minor intValue] == 0)){
        notification.alertBody = @"Welcome";
        notification.soundName = @"Default";
        //notification.alertLaunchImage
    } else if ([region.major intValue] == 1) && ([region.minor intValue] == 1)){
        notification.alertBody = @"Sessao de frutas";
        notification.soundName = @"Default";
    }else if ([region.major intValue] == 2) && ([region.minor intValue] == 2)){
        notification.alertBody = @"Refrigerantes";
        notification.soundName = @"Default";
    }
    [[UIApplication sharedApplication] presentLocalNotificationNow:notification];
}

- (void) locationManager:(CLLocationManager *)manager didExitRegion:(CLBeaconRegion
*) region
{
    UILocalNotification *notification = [[UILocalNotification alloc] init];
```

```
    if (([region.major intValue] == 0) && ([region.minor intValue] == 0)){
        notification.alertBody = @"Ate a proxima";
        notification.soundName = @"Default";
    }

    [[UIApplication sharedApplication] presentLocalNotificationNow:notification];
}

- (NSString *)categorizacao:(CLProximity)distancia
{
    switch (distancia)
    {
        case CLProximityFar:
            return @"Beacon distante";
            break;
        case CLProximityImmediate:
            return @"Beacon muito proximo";
            break;
        case CLProximityNear:
            return @"Beacon perto";
            break;
        case CLProximityUnknown:
            return @"Posicao desconhecida";
        break;
    }

}

- (NSString *)qualificacao:(CLBeaconRegion *)beacon
{
    if (([beacon.major intValue] == 0) && ([beacon.minor intValue] == 0))
        return @"Beacon 0";
    else if (([beacon.major intValue] == 1) && ([beacon.minor intValue] == 1))
        return @"Beacon 1";
    else if (([beacon.major intValue] == 2) && ([beacon.minor intValue] == 2))
        return @"Beacon 2";
```

```
    else
        return @"Nenhum beacon encontrado";
}

- (void)locationManager:(CLLocationManager *)manager didRangeBeacons:(NSArray
*)beacons inRegion:(CLBeaconRegion *)region
{
    NSString *fala = @"Beacon";
    if (beacons.count > 0)
    {
        NSLog(@"O beacon Phari foi encontrado. %@", beacons[0]);
    }
    else
        NSLog(@"Nenhum beacon foi encontrado.");


   if (beacons.count > 0)
    {
        CLBeacon *beacon = beacons[0];
        CLBeaconRegion *beaconRegion = beacons[0];
        self.messageLabel.text = [self categorizacao:beacon.proximity];
        self.Label.text = [self qualificacao:beaconRegion];
        NSLog([fala stringByAppendingString:[beaconRegion.major stringValue]]);
    }

}

- (void)didReceiveMemoryWarning
{

    [super didReceiveMemoryWarning];

}

@end
```