UNIVERSIDADE DE SÃO PAULO

Instituto de Ciências Matemáticas e de Computação

Rastreamento e Localização de Pessoas

Marcelo Shinye

São Carlos – SP

Rastreamento e Localização de Pessoas

Marcelo Shinye

Orientador: *Prof. Eduardo do Valle Simões*

Monografia de conclusão de curso apresentada ao Instituto de Ciências Matemáticas e de Computação – ICMC-USP – para obtenção do título de Bacharel em Engenharia de Computação.

USP - São Carlos

Novembro de 2016

Dedico esta monografia aos meus pais,

meus familiares e minha namorada

que sempre me ajudaram

e me ampararam em todos momentos.

Agradecimentos

Dedico meus sinceros agradecimentos para:

- Meus pais pelo amor e incentivo incondicional.

- Prof. Eduardo do Valle pelo conhecimento e guia.

- Minha namorada pelo amor e carinho.

- Meus amigos que fizeram parte do meu preparo.

Resumo

Neste trabalho se desenvolveu um estudo de soluções e desenvolvimento de um sistema de comunicação entre processadores embarcados para a realização de localização de pacientes e monitoramento de seus dados, visando um melhor controle hospitalar e sinalização automatizada para contenção de epidemias e redução de riscos de contágio.

Palavras-chave: Bluetooth, Beacon, Embarcado, BLE

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

USP	*Universidade de São Paulo*
ICMC	*Instituto de Ciências Matemáticas e Computação*
BLE	*Bluetooth low energy*
API	*Application Programming Interface*
GAP	*Generic Access Profile*
SIG	*Special Interest Group*

1 Introdução

1.1 Contextualização e Motivação

Em um ambiente onde a competição entre várias organizações é altamente acirrada, o fator qualidade de serviço pode ser o diferencial entre a escolha de empresas, esta característica pode ser relacionada a vários elementos mas principalmente com toda experiência que o consumidor teve no processo de utilização do serviço/produto que deve se traduzir com apenas o sentimento de satisfação. De muitas maneiras pode se atingir qualidade: Tecnologia, Melhoria de Processos, Qualificação de Pessoas e Diminuição de Riscos. Como todas as organizações o hospital procura formas de atingir um nível de excelência, porém é necessário uma mudança de foco de como atingir novos níveis de qualidade, centralizando ações no cliente ao invés de concentrar todos os esforços na doença, como por exemplo a segurança do usuário. Para a gerência hospitalar não existe uma forma de manter um atendimento personalizado 24/7 para cada paciente, é inviável contratar uma pessoa por paciente. Desta forma vários problemas em relação à localização e falta de informação de quem está recebendo cuidados podem acontecer: Risco de fuga de um paciente para um lugar perigoso, colocando ele mesmo em perigo; Falta de localização ao paciente em caso de acidentes; Risco de sequestro, principalmente para jovens pacientes. Tendo em vista estes problemas e riscos apresentados, o presente projeto propõe a implementação de um sistema de localização indoor hospitalar que apresente baixo custo e de fácil utilização, o sistema será baseado nas plataformas Raspberry Pi e Arduino, a localização será feito através do módulo *Bluetooth* HM-10, módulos que possuem um baixo consumo energético. O *Back End* do sistema foi feito em NodeJs com Mysql, sistemas com estruturas simples porém eficientes para dado sistema, e o *Front End* foi criado com *framework*

Bootstrap.

1.2 Objetivos

Realização da localização hospitalar de pacientes utilizando processadores embarcados e *Bluetooth*, com um sistema de monitoramento das informações em tempo real visando uma melhor qualidade na gestão hospitalar e no atendimento dos pacientes.

1.3 Organização do Trabalho

Esta monografia é dividida em 4 capítulos, sendo que no primeiro capítulo de introdução, as considerações iniciais do tema e objetivos.

2 *Revisão Bibliográfica*

2.1 *Bluetooth*

Bluetooth é uma tecnologia desenvolvida que provê uma forma de realizar a troca de informação sem fio entre dispositivos como celulares, laptops, computadores, impressora de uma forma segura e com *hardware* de baixo custo que utiliza frequência de banda baixa. Desenvolvida originalmente pela empresa Ericsson, a tecnologia na época foi desenvolvida para que permitisse a comunicação entre telefones celulares e acessórios utilizando sinais de rádio de baixo custo. Atualmente o padrão é mantindo por uma SIG *Special Interest Group* formada por mais de 25 mil empresas (*Bluetooth* SIG, [200-?]), vide símbolo do grupo na figura 2.1.

Figura 2.1: Símbolo Bluetooth SIG

Para poder atender os variados dispostivos a tecnologia foi dividida em três classes, vide tabela 2.1 :

A rede de bluetooth transmite dados ondas de baixo comprimento com frequência na banda de 2,4 GHz, que é uma frequência utilizada por vários aparelhos no mercado, esta faixa de

Classe	Potência Máxima (mW)	Alcance (m)
1	100	100
2	2.5	10
3	1	1

Tabela 2.1: Classificação de Classes Bluetooth

banda (mais especificamente entre 2.402 GHz e 2.480 GHz) por ter esta caracteríStca apresentou desde o começo do desenvolvimento este grande problema, e um dos objetivos da tecnologia era buscar uma maneira de como realizar a comunicação com esta banda e não ser interferido por outros sinais. Uma das maneiras encontrada foi pela emissão de sinais de baixa potência e portanto sinais limitados a distância (10 Metros) mas que não sofrem interferência na transmissão de dados. Outra técnica utilizada para evitar esta interferência é a chamada *frequency hopping spread spectrum*, esta faz com que o aparelho bluetooth troque aleatoriamente por 79 frequências, isso quer dizer que em média o transmissor troca de frequência 1600 vezes a cada segundo, permitindo assim que mais aparelho consigam utilizar esta banda de frequência que é tão disputada por gadgets. Tipos de Aplicações:*Headsets* sem fio;Trasnferência de arquivo entre aparelhos;Alto Falantes sem fio. Como qualquer outra tecnologia, o *Bluetooth* foi estruturado em cima de vários protocolos a fim de padronizar todo o meio de comunicação e transporte de informação. Elas são dividas em algumas camadas e chamadas de Camadas de Transporte:

- RF (*Radio Frequency*): Camada responsável pelo uso de radiofrequência;

- *Baseband*: Camada de determinação da comunicação via Bluetooth. Por exemplo com esta camada que se define como dispositivos master e slave se conectam dentro de uma piconet;

- LMP (*Link Manager Protocol*): Camada que responde por definições da comunicação, lidando com parâmetros de por exemplo autenticação, taxas de transferência de dados, criptografia;

- HCI (*Host Controller Interface*): Camada de interface de comunicação com *hardware Bluetooth*, possibilitando interoperabilidade entre dispositivos;

- L2CAP (*Logical Link Control and Adaptation Protocol*): Camada que lida com a conexão entre camadas superiores e inferiores, trabalha com parâmetros de QoS (Quality of Service Qualidade de Serviço).

Após o transporte de informação ser realizado, entra em ação as Camadas *Middleware* que permitem a compatibilidade entre aplicações já criadas, mesmo com o uso de outros padrões e de outros protocolos como por exemplo o IP, e WAP. O crescimento de dispositivos com

bluetooth continua a crescer, sempre focado em intercomunicação de aparelhos proximos, este crescimento podemos verificar com suas constantes evoluções, como o BLE *Bluetooth Low Energy* que será explicado na próxima subseção.

2.1.1 BLE - *Bluetooth Low Energy*

A tecnologia *bluetooth low energy* como o próprio nome ja diz foi criado para fornecer um consumo ainda menor de energia e custo, mantendo um alcance de comunicação similar ao *Bluetooth* Clássico.Apesar de ser constantemente apresentado como uma versão otimizada da versão Clássica do bluetooth na realidade o BLE, foi desde o início desenvolvido com outra ideia de linhagem e outra ideia de design, com foco principal para que a indústria do silício implementasse este tipo de tecnologia. Esta foi criada por volta de 2010 e sua adoção foi muito grande devido à tecnologia estar fortemente ligada aos *smart devices* como celulares e *tablets*. Por um valor baixo de desenvolvimento e de *hardware* é possível criar vários tipos de produtos criativos e inovadores, custos menores comparando à outras tecnologias de comunicação sem fio como por exemplo *WiFi,GSM,Zigbee*. Outra vantagem é que pelo fato dessa tecnologia ser mantida pela SIG a maioria dos projetos BLE serão compatíveis com inúmeros aparelhos desenvolvidas pelo própria, variando de aparelhos *Apple* até *tablets* Androids. Talvez o fator menos decisivo no sucesso de adoção da tecnologia, mas não menos importante, tenha sido em como o BLE foi desenvolvido, focando na ideia de ser um *framework* de troca de dados, criando assim APIs em baixo nível que facilitaram a utilização e agregação da tecnologia sem contar na possibilidade de projetos mais personalizados e permitindo assim mais liberdade aos desenvolvedores para que usem o *framework* da maneira que mais desejar. Esta diferença de consumo que foi citada nesta subseção é a maior diferença entre a tecnologia de Bluetooth Clássica e BLE, com essa característica aplicações podem rodar com pequenas baterias por quatro a cinco anos, apesar disso não ser o caso ideal para celulares ainda é um fator vital para programas que necessitam pequenas trocas de dados periodicamente. Igualmente ao Bluetooth Clássico, BLE opera na faixa de 2.4Ghz ISM, porém diferentemente do seu antecessor o BLE permanece em estado de hibernação exceto quando uma conexão é iniciada. A conexão dura por volta de 100ms e a razão dessa conexão ser tão curta é pelo fato da transmissão de dados ser alta,

Specifications	**Bluetooth**	**BLE(Bluetooth Low Energy)**
Rede	*Scatternet*	*Star Bus*
Consumo de Energia	Baixo (menos que 30 mA)	Muito Baixo (menos que 15 mA)
Velocidade	700 Kbps	1 Mbps
Range	<30 m	50 m(150 m em campo aberto)
Frequencia RF	2400 MHz	2400 MHz
Canais de Frequência	79 canais de 2.400 GHz até 2.4835 GHz com 1 MHz espaçamento	40 canais from 2402MHz até 2480 MHz
Modulação	GFSK (índice de modulação em 0.35) , pi/4 DQPSK, 8DPSK	GFSK (índice de modulação em 0.5)
Latência	Aprox. 100 ms	Aprox. 3 ms
Tamanho da mensagem(bytes)	358 (Max.)	8 to 47
Detecção de erro	8 bit CRC(header), 16 bit CRC, 2/3 FEC, ACKs	24 bit CRC, ACKs
Segurança	64b/128b, definido pelo usuário	128 bits AES definido pelo usuário
Taxa de Transferência	0.7 até 2.1 Mbps	menor que 0.3 Mbps
Nós	7	Ilimitado

Tabela 2.2: Tabela de Comparação Blutooth e BLE

em torno de 1Mb/s. Tipos de Aplicações:iBeacons;Aplicações de Transporte Público;Monitores de pressão. Pela Tabela 2.2 é possível verificar as principais diferenças entre as tecnologias de Bluetooth e BLE.

Além das diferenças de aplicações e técnicas, o BLE possui um modo de divulgação de informação, a camada responsável é chamada de *Generic Access Profile* (GAP). Uma das aplicações que utilizam fortemente esta camada são os chamados *iBeacons*, que serão explicados na próxima subseção.

2.1.2 *iBeacons*

Desenvolvido pela *Apple*, o *iBeacon* é uma tecnologia que tem como principal função realizar a divulgação de informações, em pequenos pacotes de dados que são regularmente propagados. Esta comunicação é de uma via somente, *iBeacons* que querem ser descobertos continuam a realizar o *broadcast* de informação, este dados que são enviados para todos devem ser coletados por aparelhos como celulares *smartphones*, que terão aplicativos capazes de processar esses dados e realizar certas ações ao recebê-las, como por exemplo envio de mensagens e chamada de funções. A figura 2.2 exemplifica a ideia do funcionamento da tecnologia.

[1]

Com esta tecnologia foi também criado um padrão de como os dados seriam propagados, esta padrão foi divido em 4 partes e cada parte destes dados fornece informações fundamentais para o funcionamento de todo o sistema. O campo UUID possui 16 bytes e é usado para

[1] Figuras 2.2 e 2.3 retiradas do site www.ibeacon.com

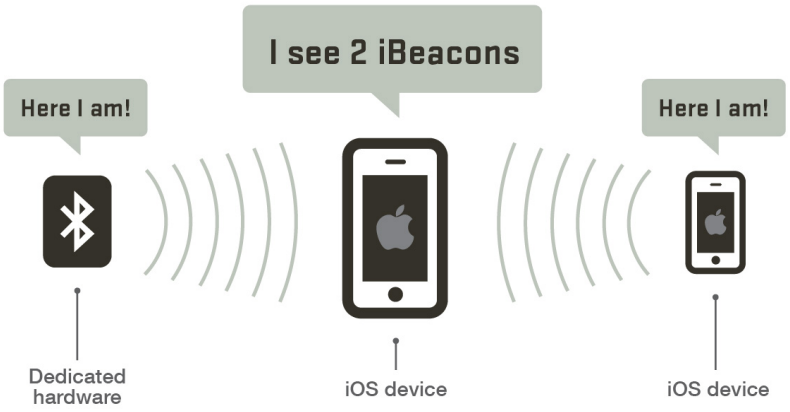

Figura 2.2: Modo Funcionamento de iBeacons

diferenciar de grandes grupos de *beacons*. Por exemplo se a proposta é criar uma grande rede de *iBeacons*, todos possuirão o mesmo UUID, o campo *Major* possui 2 bytes e é utilizado para diferenciar de grupos menores em um grande grupo, o penúltimo campo *Minor* possui 2 bytes e sua função é para diferenciar os menores grupos de uma coleção de *iBeacons*. O último campo é o *TX Power*, um valor em dBM que define a força que um sinal chega em exatamente um metro de distância, este é muito utilizado de referência para um cálculo de estimativa de distância.

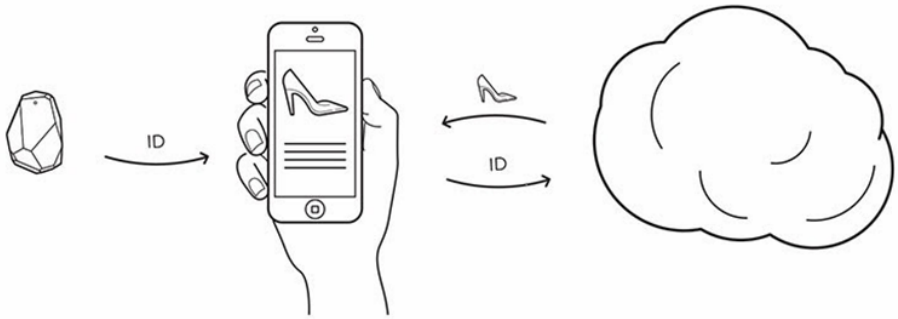

Figura 2.3: Ideia Básica de um Projeto iBeacon

Pela figura 2.3 tem se a ideia típica de um projeto com *iBeacons*, o consumidor com seu aplicativo rodando que busca *iBeacons* próximos, ao serem encontrados o próprio aplicativo dispara uma ação para o server, resultando em uma ação como uma simples mensagem ao usuário,

ou em outro caso enviar fotos do produto realizando uma propaganda de certo produto. As possibilidades e oportunidades são ilimitadas, pois esta tecnologia provê uma extensão digital do mundo real e mudará a forma como as empresas se comunicam com o consumidor final.

2.2 Gestão de Qualidade Hospitalar

A mensuração da qualidade serviços é feita normalmente por meio de pesquisas com os clientes em relação à sua satisfação, desta maneira elas são a principal maneira de percepção de qualidade do lado do cliente que é possível obter, tais pesquisas são fundamentais para uma corporação pois permitem ter uma visão realista de como anda a qualidade de seus serviços e são a forma mais eficaz para um gestor tomar ações em cima, dando uma direção de como tornar os serviços mais qualificados para seus clientes.(STRAW, 2002;ALÁSTICO, 2011). Este método e pensamento faz com que o constante contato e informações providas pelo cliente levarão sempre a melhoria contínua dos processos e portanto uma maior competitividade entre os seus concorrentes. Entretanto nem todas organizações precisam qualificar seus processos apenas pela competitividade, como por exemplo o hospital, que tem atrelado também à qualidade o bem estar de seu público. Tempos atrás, poucos médicos se interessavam na questão de qualidade em relação à saúde, inicialmente tal avaliação foi criada apenas para padronizar os resultados com pacientes por meio da criação de processos de avaliações médicas. E com os estudos de Donabedian (1990) foi desenvolvido um quadro conceitual para o entendimento da avaliação da qualidade da saúde, a partir dos conceitos em estrutura, processos e resultados. A parte estrutural é derivada de alguns pilares como recursos humanos, recursos físico e recursos tecnológicos. Por muito tempo a capacitação técnica do recurso humano era o que ditava o nível de qualidade do serviço de saúde(GIARELLI, 2002). Porém com a evolução da tecnologia, esta apresentou muitas inovações que possibilitaram o aumento de qualidade na gestão hospitalar e suprir algumas necessidades apresentadas pela área de saúde. Conforme Godoy(2011,p.3), ”Deficiências no processo de gestão hospitalar como, por exemplo, no controle de equipamentos e materiais, podem afetar drasticamente a qualidade dos serviços (VISSERS; BEECH, 2005). Assim, a justificativa principal do trabalho é a sistematização da implantação

da tecnologia RFID no controle de ativos móveis em ambientes hospitalares como, por exemplo, cadeiras de rodas, macas, disfibriladores, bombas etc, identificando informações sobre estes recursos que poderiam ser utilizadas como base de melhoria dos processos."A implementação de tecnologias como o caso do autor traz benefícios e maior qualidade de serviço aos hospitais, portanto é um pilar que apresenta inúmeras oportunidades de melhoria.

2.3 Localização

"Orientação espacial é a capacidade que o indivíduo tem de situar-se e orientar-se, em relação aos objetos, às pessoas e o seu próprio corpo em um determinado espaço. É saber localizar o que está à direita ou à esquerda; à frente ou atrás; acima ou abaixo de si, ou ainda, um objeto em relação a outro. É ter noção de longe, perto, alto, baixo, longo, curto"(ASSUNÇÃO; COELHO, 1996, p.91-96). Nesta citação o autor fala da capacidade que o ser humano possui para realizar a orientação em relação ao espaço porém mais que uma capacidade o ser humano possui uma necessidade de saber sua orientação, saber sua localização para conseguir algum serviço a partir daquela posição. A expansão da microeletrônica, computação e meios de comunicação permitiram a criação de tecnologias que como dito acima, elabora novos serviços que possibilitaram a localização exata em que o usuário se encontra. Mas para que tudo isso ocorra é necessário pensar em toda infra-estrutura de comunicação que suportará estas novas aplicações que resultarão na localização física do objeto. Pensando em um ambiente *outdoor* a tecnologia GPS (*Global Positioning System*) é a mais requisitada, pois consegue sob qualquer condição atmosférica e em qualquer lugar da terra desde que o dispositivo esteja no campo de visão de pelo menos 3 satélites, a tecnologia desenvolvida tem a capacidade de definir sua posição no globo, como visto pela figura 2.4. Para o ambiente *indoor* o GPS não consegue atuar com a mesma eficácia que no ambiente outdoor, por causa dos sinais de microondas que são fortemente atenuados pelo teto e paredes das casas, por isso outras alternativas foram feitas para este tipo de ambiente como por exemplo a técnica usada com a tecnologia *Wifi* que a partir da intensidade do sinal do receptor pode se estimar a distância do ponto de acesso. Parâmetros como SSID e endereço MAC são utilizadas para o cálculo, e a precisão depende muito dos dados que

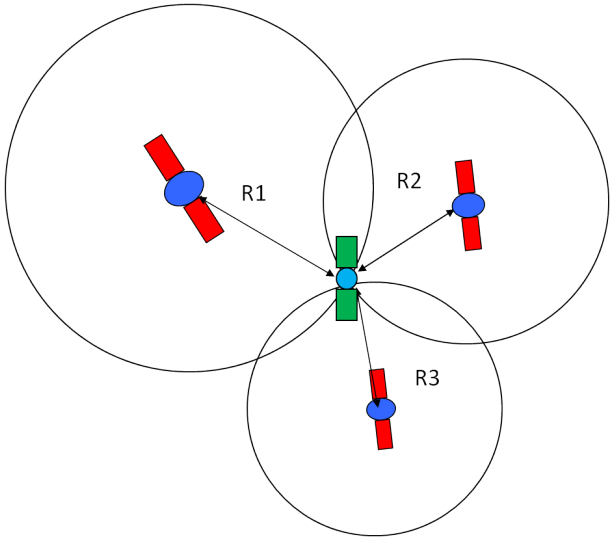

Figura 2.4: Funcionamento Básico de um GPS

já foram previamente inseridos no banco de dados. Um dos problemas é que a variação do sinal implica em erros para o cálculo diminuindo portanto a precisão deste tipo de localização. Outra técnica utilizada é a de ângulo de recepção, ela consiste basicamente no ângulo em que o sinal chega no receptor. Este ângulo é estabelecido medindo o tempo entre a chegada do mesmo sinal entre várias antes de recepção. E um outro método que será comentada no próximo capítulo é a estimativa de distância pelo indicador de força de sinal (RSSI). As ondas de rádio se propagam de acordo com a variação com o inverso do quadrado, portanto sua distância pode ser estimada dependendo do nível de sinal entre o receptor e transmissor, porém como as ondas são refletidas e absorvidas geralmente este valor apresenta muito ruído por isso apresenta uma grande dificuldade na estimativa da localização e por isso que uma das áreas de pesquisa desta tecnologia que mais cresce é a de redução de ruídos através de filtros.

2.4 Sistema Embarcado

Um computador pessoal é utilizado para várias funções, navegar pela internet, digitar textos, ouvir músicas, editar fotos, etc. Para cada atividade que é executada uma parte de seus recursos é alocada e por isso que necessitam de uma grande quantidade de recursos para conseguir atender toda esta demanda. Um sistema embarcado tem um propósito diferente de um computador pessoal, todo seu *hardware* e *software* são desenvolvidos com um objetivo específico.

Principais Características:

Menor Custo/Menor Consumo

O menor custo por unidade e menor consumo de energia acontece justamente pelo *hardware* ser dedicado é necessário menos recursos para o sistema, porém por causa desta qualidade o desenvolvimento do sistema é diferente pois toda arquitetura deve ser pensada de modo a administrar os recursos eficientemente.

Interface

Muitas vezes o sistema embarcados não apresenta interface para o usuário, principalmente com sistemas dedicados para uma tarefa somente, entretanto geralmente a interface apresentada para o usuário é feita com botões, telas LCDs e LEDs, tudo isto para evitar utilizar recursos com a parte gráfica, que é computacionalmente cara.

Disponibilidade

É esperado que sistemas embarcados trabalhem continuamente por muito tempo, e que consigam se recuperar sozinhos depois de algum erro, pode se ver este tipo característica em sistemas críticos, onde são necessários 4 dimensões de confiança: Disponibilidade, Confiabilidade, Segurança e Proteção. Porém sempre focando na confiança pois esta qualidade reflete a credibilidade do usuário em relação ao sistema como também as expectativas do usuário que não haverá falhas conforme a operação ocorre.

Sensores

A presença de sensores é a forma que sistemas embarcados se comunicam com o meio externo, estes periféricos enviam dados para o processador embarcado e variam desde simples sensores de contato(mecânico) até sonares.

Atuadores

Atuadores são as ferramentas que possibilitam sistemas embarcados de atuarem em seu meio, como motores e ventiladores. Geralmente atuadores funcionam com base em informações que são processadas de sensores, este tipo de sistema é conhecido como malha fechada, que

utilizam a realimentação de informação para poderem atuar de alguma forma. Pela Figura 2.5 é possível detalhar alguns componentes principais de sua estrutura.

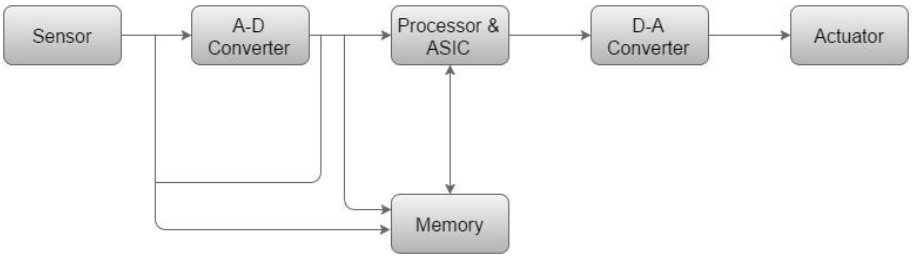

Figura 2.5: Sistema Embarcado Básico

Conversor A/D

Um conversor que transforma sinais analógicos dos sensores em um formato digital.

Processador e ASIC

Componente que processa dados e guarda em sua memória.

Conversor D/A

Um conversor que transforma sinais digitais do processador em um formato analógico.

A tecnologia embarcada é usada em vários ramos como por exemplo no controle de vôo, telefones celulares, videogames, PDAs, Calculadoras, etc. Uma grande maioria dos aparelhos computacionais que existem são embarcados, e a estimativa é que em 2020 40 bilhões de aparelhos sejam também.

3 Desenvolvimento do Trabalho

3.1 Projeto

O objetivo do projeto consiste na localização de pessoas em um ambiente hospitalar, sendo que cada paciente/agente de saúde carregará um iBeacon que gerará um sinal único que será escaneado por um processador embarcado (Raspberry Pi), esta ficará localizada em cada sala do hospital. Cada processador embarcado executará o *scan* de sua área e estes dados serão processados e inseridos em um banco de dados central, todo o *software* desenvolvido foi feito em C, uma linguagem com muitas bibliotecas e bom desempenho para certas aplicações. Este mesmo servidor será responsável pela manutenção dos dados, como também manter um *web-server* em node js, possibilitando que outros acessem uma interface que processe todos estes dados enviados pelos processadores embarcados e consiga apresentar de uma forma simples todo o resultado da aplicação, a parte do *back-end* do servidor foi desenvolvido com Javascript e a interface foi criada com o *framework* Bootstrap. Pode se verificar toda esta estrutura pela figura 3.1.

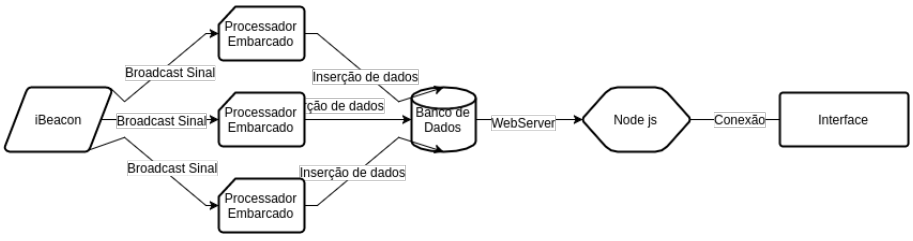

Figura 3.1: Visão Alto nível de Localizador BLE

3.1.1 extithardware

Arduino Nano

Arduino é uma plataforma *open-source* usada para prototipação ou até mesmo a construção de projetos eletrônicos. Ele é separado basicamente por duas partes, uma é o microcontrolador que é o circuito programável e um *software*, este pode ser desenvolvido em alguma máquina pessoal e depois enviado para a placa programável através de uma interface que pode ser inclusive a USB. A facilidade de desenvolvimento e de aprendizado foram as razões por ter sido escolhida este tipo de tecnologia na construção deste projeto, facilidades devido à extensa comunidade ativa, grande quantidade de conteúdo desenvolvido e principalmente pelo fato de ser uma ferramenta com código aberto. A plataforma Arduino possui vários tipos de modelos, cada um com suas qualidades e portanto cada um focado em certos tipos de aplicações, eles diferem em quantidade de pinos de entrada/saída, tamanho, interface de comunicação serial e tipo de microcontrolador. Pensando em todos estes fatores o modelo escolhido foi o Arduino Nano, tendo em vista seu tamanho reduzido (veja figura 3.2) e baixo consumo e a presença de uma interface com porta mini USB que facilita muito os testes já que este tipo de conexão é comum no mercado. Porém o fator decisivo para tal tecnologia ser escolhida foi pelo baixo custo por unidade, em torno de R$ 22,00.

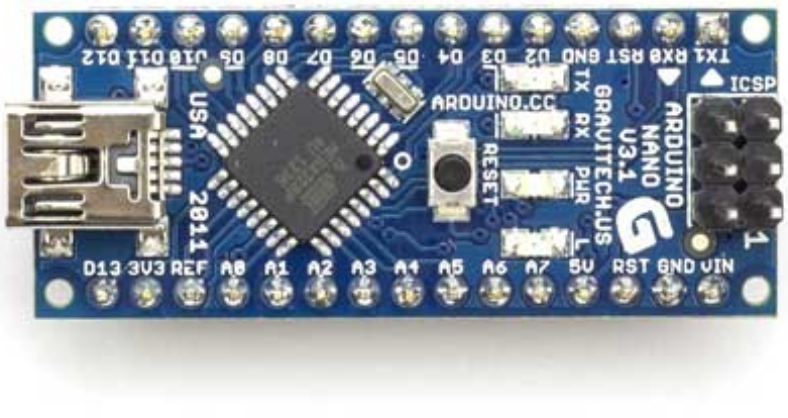

Figura 3.2: Arduino Nano

Na tabela 3.1 é possível verificar todas características do modelo Nano.

CPU	Microcontrolador Atmel Atmega 328
Tensão Mínima de Alimentação	3.3V
Tensão Máxima de Alimentação	20V
Tensão Ideal para Alimentação	7V-12V
Pinos Digitais I/O	14 pinos, no qual 6 destes também servem para saídas PWM
Pinos de Entrada Analógica	8 entradas com 10 bits de resolução
Corrente DC Por Pino I/O	40mA
Clock	16 MHz

Tabela 3.1: Características de Arduino Nano

Sua utilização no projeto será limitada a interface de configuração do *iBeacon*, ou seja uma vez que foi realizada a configuração não será mais necessária a intervenção da plataforma Arduino, portanto somente um Arduino Nano é necessário para todo o projeto.

Raspberry Pi

Raspberry Pi é um computador de tamanho reduzido que possui todas as funções que um computador pessoal normal, originalmente desenvolvido com a ideia de um aparelho de baixo custo que ajudariam as pessoas a melhorarem seus conhecimentos em programação e em eletrônica, mas por causa do seu tamanho reduzido muitos começaram a utilizar para projetos eletrônicos que necessitam mais que um simples microcontrolador. Apesar da aparência o funcionamento dele é igual a de um computador normal ou seja, é possível conectar a porta HDMI de vídeo, o mouse, o teclado e o usuário conseguirá utilizar através de uma interface gráfica todos recursos disponíveis como navegador, calculadora e terminal do sistema operacional. O modelo utilizado neste projeto foi o Raspberry Pi 1 Model B (Figura 3.3), pela tabela 3.2 pode se verificar todas as qualidade do modelo citado.

A plataforma Raspberry foi escolhida primeiramente por ser uma tecnologia portátil e portanto pode ser posicionada em lugares estratégicos para ter uma melhor recepção do sinal dos *iBeacons*, segundo que é um computador com um custo reduzido e que consegue suprir todas necessidades em relação ao processamento de dados, por fim o ultimo ponto a considerar é pelo fato de ser uma tecnologia flexível e de fácil aprendizado e que possui uma grande comunidade ativa. O modelo utilizado neste projeto é uma versão antiga da plataforma, que foi substituído

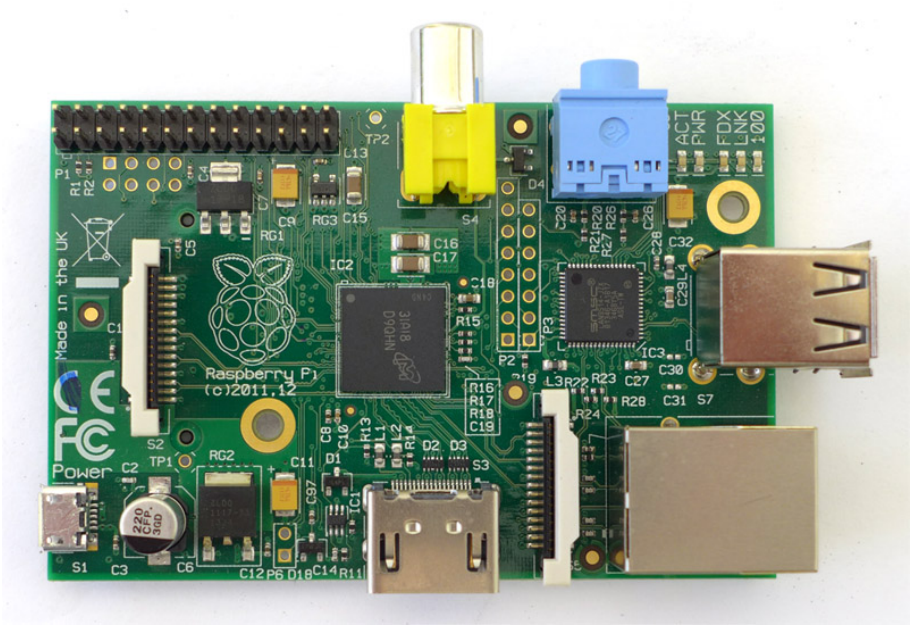

Figura 3.3: Raspberry Pi 1 Model B

Processador	Broadcom BCM2835
Clock do Processador	700 MHz
RAM disponível	256 MB
Conector de Rede	*Ethernet* 10/100
Conector de Vídeo	HDMI, RCA
Conectores USB	2.0 (2x)
Pinos de I/O	26
Consumo	5V; 700mA; 3,5W
Dimensões	85 x 56 x 17 mm

Tabela 3.2: Características Raspberry Pi 1 Model B

pelo modelo Raspberry Pi 3 porém todo o desenvolvimento neste projeto é compatível com qualquer versão da plataforma, o único ponto a se comentar é pelo fato das novas versões de OS da Raspberry possuirem uma configuração diferenciada de seus pinos UART (Universal asynchronous receiver/transmitter) e portanto é necessário fazer uma configuração dos mesmos.

Buetooth HM-10

O módulo HM-10 é o elemento central deste projeto, desenvolvido pela JNHuaMao Technology Company o componente que implementa a tecnologia BLE é capaz de transmitir informação *wireless* utilizando o padrão de comunicação 4.0. Como explicado no capítulo anterior este padrão tem como principal qualidade o baixo consumo de energia. Em seu modo de

funcionamento comum dois dispositivos conseguem trocar informações entre si, primeiramente eles devem fazer a conexão (pareamento) e após este passo os dados que chegam são enviados para sua porta serial (RX). Para realizar o envio de informação as informações devem ser encaminhadas para uma porta *serial* TX. Porém para este projeto em específico o módulo HM-10 foi implementado em modo *iBeacon* e um outro em modo *Master Scanner*, diferente do modo comum, neste projeto não haverá pareamento para troca de dados, pois o *iBeacon* realiza um *broadcast* de informação logo não é necessário nenhum tipo conexão entre os módulos. Veja a figura 3.4 da estrutura especificada e simplificada, sendo que o *bluetooth* da esquerda o *iBeacon* e o da direita o *Scanner*.

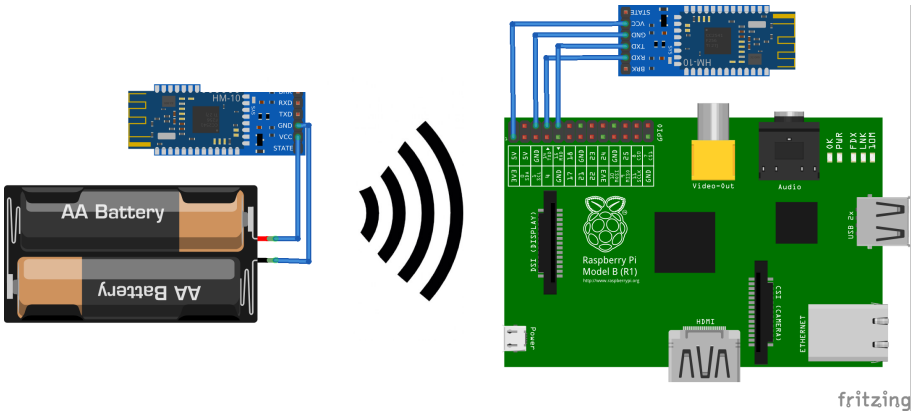

Figura 3.4: Estrutura de Bluetooth

As características do módulo podem ser verificadas na tabela 3.3,os principais pinos utilizados para este sistema foram os de transmissão e recepção *serial*, este caso so aplica para o dispositivo *Master Scanner*, pois só ele necessita transmitir algum dado para o processador embarcado enquanto o *iBeacon* somente realizará o *broadcast* de informação portanto so necessitará de uma fonte de energia para ficar ligado.

Versão Bluetooth	4.0
Frequência de Operação	2,4GHz
Modulação	GFSK (Modulação Gaussiana)
Voltagem de Operação	3,3V
Corrente de Operação	50mA max; 8,5mA em operação
Corrente em Modo Sleep	400uA

Tabela 3.3: Características HM-10

3.1.2 *Software*

O *software* deste projeto foi dividido em basicamente duas partes, uma seria o *software* de controle que está implementado na Raspberry Pi e o outro módulo implementado no servidor de dados. Como explicado na seção anterior uma parte fundamental do programa é a troca de informação entre o módulo BLE e o processador embarcado e por isso a necessidade de um *software* para controlar toda essa sincronização de informação, como também uma estruturação da mesma para que não haja erros de integração entre esses dois dispositivos. Por fim uma interface simples para o usuário final foi necessária para visualização do resultado final e uma forma fácil de obter este valor, bem como também uma forma de guardar os dados. As duas subseções a seguir cobrirão com mais detalhes cada módulo.

Software Controle

Esta fração do código tem como objetivo realizar o controle do módulo HM-10 e organizar as informações para que possam ser inseridas em um banco de dados central, ou seja tornar todo processo de controle/inserção de informação o mais confiável possível. O programa foi desenvolvido na Raspberry Pi 1 Model B, na linguagem C utilizando basicamente bibliotecas *standard* C. O funcionamento da deste módulo será exemplificado pela figura 3.5.

O Software Controle inicia sua execução executando o *scan* de informações dos *iBeacons*, este procedimento retornará as informações necessárias para identificação e cálculo de distância, porém enquanto não for recebido totalmente a informação o algoritmo não seguirá em frente. Após receber toda a informação necessária estes dados que são passados como uma *string* serão organizados em estruturas para que essas informações possam ser processadas de uma maneira adequada, em poucas palavras estamos transformando uma *string* em vários vetores de *float* e *string*. Com as informações estruturadas é possível realizar o cálculo de estimativa de distância, basicamente é uma fórmula de regressão exponencial e quer será mais detalhada na próxima seção. O próximo passo é realizar o *cache* de informações, isto é feito para diminuir os erros de estimativa e consequentemente aumentar a precisão. Caso a informação seja suficiente o sistema insere as informações em um Banco de Dados (Mysql) que será utilizado

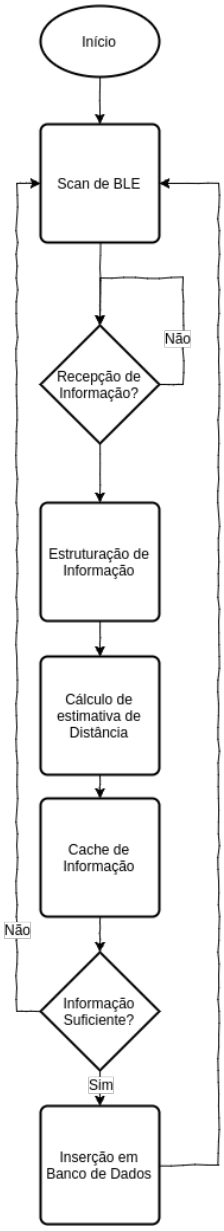

Figura 3.5: Funcionamento *Software* de Controle Raspberry Pi

por uma interface.

Server/Client Side

Este módulo realiza toda a parte de salvamento de dados, interface com o usuário(*Front-End*) e uma interface com o Banco de Dados(*Back-End*). E cada tecnologia é responsável por uma pequena divisão, o banco de dados mysql cuida do servidor de dados, ele possui uma tabela seguindo a estrutura da figura 3.6

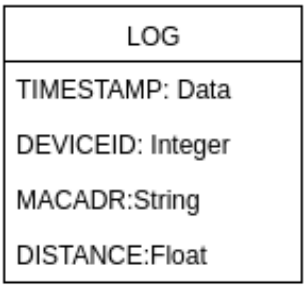

Figura 3.6: Estrutura de Banco de Dados do Projeto

O projeto do banco de dados será com dados simples e seu único objetivo é guarda-los:

- *Timestamp*: Campos que indicará o horário em que scan foi realizado;

- *DeviceID*: Campo que indicará qual dos processadores embarcados fez a inserção deste registro;

- *MacAdr*: Campo que mostrará qual o endereço físico do bluetooth que foi encontrado;

- *Distance*: Distância em que foi estimada para o dispositivo *iBeacon*.

O *Back-end* foi desenvolvido em Node Js, que é uma tecnologia *open-source* construida em cima do motor Javascript que permite a construção de aplicações de rede rápidas e escaláveis, podendo até ser utilizado para levantar *webservers* simples. Esta técnica foi utilizada de forma preparar o *back-end* do servidor a receber *requests* do *front-end*. O *Front-end* assim como *back-end* foi desenvolvido com javascript, porém utilizando outras técnicas baseadas nesta tecnologia, a parte visual da interface foi feita com o bootstrap, que é um *framework* para criação de *websites* utilizando HTML e CSS. A estrura simplificada de todo o *server side* seria como a figura 3.7.

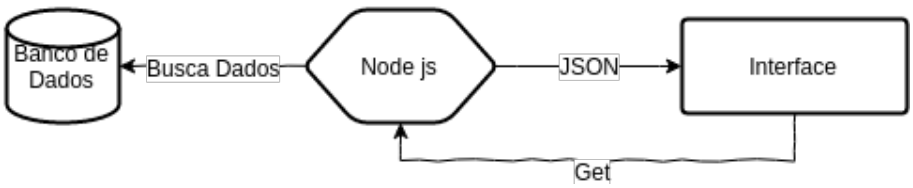

Figura 3.7: Estrutura de *Server/Client Side*

O funcionamento da estrutura é simples, quando o usuário fizer uma requisição (GET), dados serão carregados para o *webserver* em Node Js, e neste *back end* será construído uma

estrutura chamada JSON para que as informações possam ser transferidas ao usuário, quando a produção da estrutura estiver pronta o arquivo é transferido ao *Client-side* e exibida no browser do usuário. Na próxima seção será abordada todo o processo de desenvolvimento em detalhe.

3.2 Descrição das Atividades Realizadas

3.2.1 Configuração BLE em *iBeacon*

O módulo HM-10 tem como configuração de fábrica o funcionamento no modo clássico em modo *slave* porém para este projeto deve se configurar para cada paciente um *iBeacon* único e portando para cada dispositivo BLE será necessário realizar o procedimento a seguir. Como explicado no capítulo anterior, estes dispositivos HM-10 possuem um *framework* que facilita a comunicação e configuração dos mesmos, a ideia geral é que você consiga realizar todas estas ações através de comandos que são enviados através da porta RX de cada dispositivo BLE, por ser uma porta *serial* os comandos são enviados como uma *string*. Na tabela 3.4 são apresentados alguns comandos básicos do módulo.

Comando	**Descrição**
AT	Comando para testar se módulo está funcionando.
AT+BAUD?	Receber valor de taxa de bits.
AT+MODE?	Retorna o modo que o dispositivo está trabalhando.
AT+NAME?	Retorna o nome do dispositivo.
AT+VER?	Retorna a versão do firmware.

Tabela 3.4: Comandos básicos do Módulo HM-10

Para enviar estes comandos de uma maneira simples para o módulo foi utilizado um Arduino Nano como interface para o envio de comandos, porém seria possível utilizar uma interface UART-USB, pois basicamente a idéia de configuração dos *iBeacons* é enviar algumas *strings* de comando para o módulo HM-10 portanto não necessariamente é preciso usar uma plataforma Arduino. Pela figura 3.8 é apresentada a estrutura que foi montada para configuração dos *iBeacons*.

No Arduino Nano foi utilizado um código exemplo da IDE Arduino chamado *SoftwareSerialExample*, o código configura as portas 10 e 11 do Arduino respectivamente como Entrada

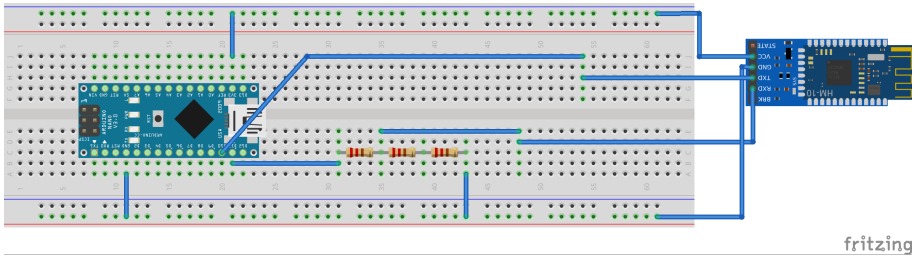

Figura 3.8: Estrutura de Módulo Arduino com HM-10

Serial e Saída *Serial*, portanto com toda parte de comunicação feita é necessário uma interface para o usuário que neste caso será utilizado da IDE Arduino também. No menu *Tools -¿ Serial Monitor* é possível monitorar a porta escolhida verificando a interação com o módulo HM-10, e por fim enviar os comandos listados a seguir para configurar o iBeacon.

1. AT+RENEW - Voltar para as configurações de fábrica.

2. AT+RESET - Reinicializar módulo.

3. AT - Confirmar se módulo esta ativo.

4. AT+MARJ0x1234 - Configurar *Major* de *iBeacons* para hexa 0x1234.

5. AT+MINO0xFA01 - Configurar *Minor* de *iBeacons* para hexa 0xFA01.

6. AT+ADVI5 - Configurar o tempo de intervalo de funcionamento para 5 (546.25 milisegundos).

7. AT+NAMEBEACON - Definir nome do módulo para BEACON.

8. AT+ADTY3 - Tornar não conectável para economia de bateria.

9. AT+IBEA1 - Habilitar modo *iBeacon*.

10.AT+DELO2 - Habilitar modo de *broadcast* de informação apenas.

12.AT+RESET Reinicializar módulo.

Com este procedimento o módulo HM-10 foi configurado como *iBeacon*, e nunca mais precisará ser conectado a outra interface, portanto poderá ser retirado a entrada e saída *serial*, deixando apenas conectado a alguma fonte de energia.

3.2.2 Funções Desenvolvidas

Software Controle

Função receiveString A função receiveString tem a função de realizar a Interface com funcoes de Porta *Serial* e *Bluetooth*, preparação de dados para inserir no Banco de dados, e possui como parâmetro msgString que é um ponteiro para recepção de dados da porta *Serial* e tem como retorno a quantidade de *devices* que foram encontrados. Pela figura 3.9 é possível verificar o fluxograma desta função.

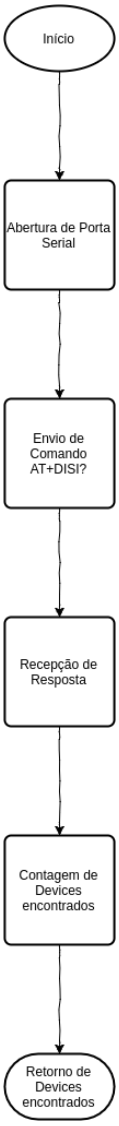

Figura 3.9: Função receiveString

A função inicialmente abre sua porta *serial* para enviar uma cadeia de *string*, o comando AT+DISI? que tem como função retornar todos dispositivos *bluetooths* encontrados naquele

momento, quando os resultados chegarem na porta *serial* o resultado ficará alocado em uma *string*. E com este valor começará o primeiro processo de estruturação de informação, na qual ocorre a contagem de aparelhos encontrados pela resposta do dispositivo BLE.

Função countSubstr A função countSubstr tem a função de Contar quantos *Devices Bluetooths* foram encontrados pela *String* mensagem, e possui como parâmetro cSubString para identificar módulos e strMsg que é *String* principal da Mensagem. Pela figura 3.10 é possível verificar o fluxograma desta função.

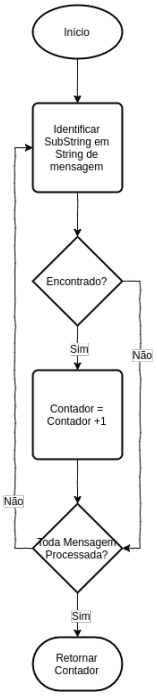

Figura 3.10: Função countSubstr

A função deste algoritmo é identificar certas cadeias de caracteres que indicam o encontro de um dispositivo *bluetooth*, representada por esta *string* ”OK+DISC:”. Todos os casos da substring ”OK+DISC:”significa que mais um *device* foi localizado, e este processamento é feito até que a *string* inteira seja trabalhada.

Função getStr A função getStr tem a função de receber *String* que *device Bluetooth* está enviando pela porta *serial*, e possui como parâmetro Ponteiro de char buff que receberá a *String* da mensagem. Pela figura 3.11 é possível verificar o fluxograma desta função.

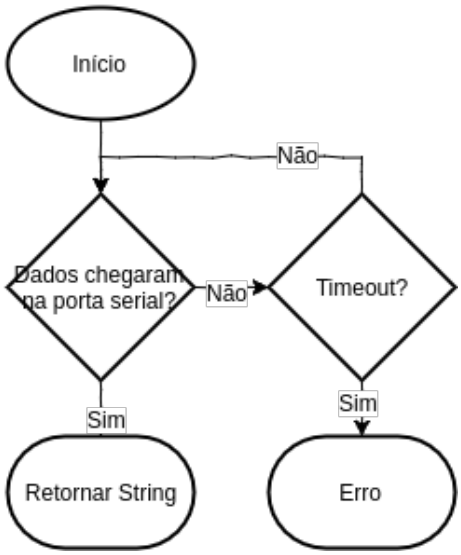

Figura 3.11: Função getStr

Esta simples função opera em cima da abertura da porta *serial*, onde chegarão os dados. Enquanto os dados não chegarem a porta continuará aberta até que as informações sejam transmitidas para a porta ou até que o *timeout* de espera seja atingido, levando a função a finalizar com um erro.

Função insertDatabase Esta função tem como função realizar a preparação de dados e inseri-los em um banco de dados. Possui como parâmetro a *string* msgString que é a mensagem do *scan* de informações do *bluetooth* HM-10 e um número inteiro qtdDevice que representa a quantidade de dispositivos encontrados. Pela figura 3.12 é possível verificar o fluxograma desta função.

Esta função inicialmente realiza toda a estruturação da *string* do resultado enviado pelo dispositivo HM-10, porém esses dados não são enviados logo em seguida pois é necessário filtrar as informações para que os resultados tenham mais precisão assim sendo um *cache* de informação foi implementado, e somente quando este *cache* for preenchido que os dados serão inseridos no banco de dados.

Função median Função que calcula a mediana de um vetor com tamanho n, possui como parâmetros n indicando o tamanho do vetor e o vetor x que contém os valores, o retorno é a

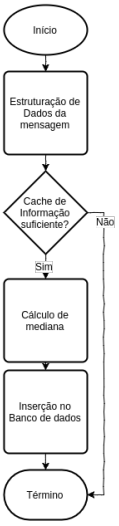

Figura 3.12: Função insertDatabase

mediana calculada.

Função getDistanceRSSI getDistanceRSSI é o procedimento responsável por estimar a distância entre *iBeacon* e *Bluetooth* por regressão exponencial, possui como parâmetros o valor de rssi em db e o valor de transmissão de energia e por fim o retorno da função é o valor estimado da distância. A fórmula utilizada para a estimativa de distância foi encontrada empiricamente, pelo método de regressão exponecial. Outro método foi implementado pela relação teórica entre os valores que seria:

$$\dot{R}SSI = -10 * n * (\log_{10} d/d0) + A \tag{3.1}$$

Porém pelo fato do valor teórico ter uma variância grande e logo o erro também estar grande, foi necessário verificar outro método resultando neste algoritmo:

$$\dot{R}atio = RSSI/Meas.Power \tag{3.2}$$

$$\dot{D}istancia = 0.89976 * ratio^{7.7095} + 0.111 \tag{3.3}$$

Server/Client Side

Back-End Engine O *Back-End* do sistema foi desenvolvido com a tecnologia Node Js, e a sua responsabilidade é de atuar como uma interface com o banco de dados para que o clinte não

realize a busca diretamente no *database*, permitindo assim um controle maior das informações que serão acessadas por cada usuário. O funcionamento deste é bem simples, já que a estrutura do banco de dados não é complexa, o *webserver* em node responde por eventos então ele ficará em espera até que uma requisição do cliente chegue. Para o caso deste projeto foi desenvolvido apenas uma função que retornará os dados do banco de dados e as enviará em formato JSON para o usuário. O fluxograma desta função é apresentado pela figura 3.13.

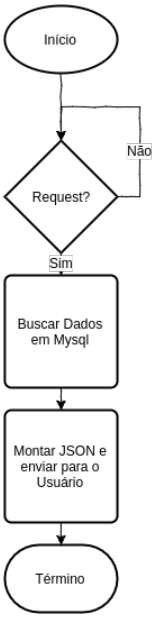

Figura 3.13: Fluxograma *Back-End*

Front-End Engine O *Front-End* do sistema foi construído em HTML/CSS com o *framework* Bootstrap e a função principal deste módulo é de mostrar o resultado final de todo o processo, apresentação dos resultados em uma forma simples e que possa acessada por todos. A engine foi construída com javascript/jquery, e seu funcionamento é basicamente enviar a *request* para o *Back-End*, receber os resultados e apresentar todos os valores de uma forma clara para o usuário. O fluxograma desta função é apresentado pela figura 3.14.

Na próxima seção serão detalhados os resultados deste projeto e análise em relação aos objetivos iniciais que foram estabelecidos.

Figura 3.14: Fluxograma Front-End

3.3 Resultados Obtidos

3.3.1 Precisão e Alcance

O projeto de localização de pessoas no ambiente hospitalar tem como parte central o trabalho entre os dispositivos *iBeacons* e o *Scanner* de *iBeacon* portanto para que seja possível criar todo o projeto e que estes consigam se comunicar entre distâncias até 30m, o que é bem comum para um ambiente hospitalar porém a presença de obstáculos físicos atrapalham a leitura do device, gerando alguns erros. Pensando em tudo isto foram feitos alguns testes para certificar que o funcionamento para a determinação da precisão para certas distâncias como também testando o alcance do sistema. E verificando estes resultado nota se que com o aumento do alcance a precisão cai muito, chegando a valores que ultrapassam o dobro do valor medido, o tipo de aplicação é sensível a grandes distâncias porém o fator que mais influenciou foi a adição de obstáculo, o que fazia os resultados variarem muito mesmo estando em distâncias curtas.

Distância em 50cm

Para este caso o funcionamento do sistema apresentou resultados estáveis com apenas pequenas variações, os resultados foram apresentados na tabela 3.5. O experimento foi realizado com uma distância real medida em 50cm sem obstáculos.

Timestamp	**DeviceID**	**Endereço Físico**	**Distância(m)**
2016-10-20 01:34:28	1	88C25532CF72	0.009408
2016-10-20 01:34:44	1	88C25532CF72	0.012283
2016-11-06 16:59:46	1	88C25532CF72	0.232907
2016-11-06 17:00:09	1	88C25532CF72	0.232907
2016-11-06 17:00:30	1	88C25532CF72	0.342167
2016-11-06 17:00:50	1	88C25532CF72	0.342167
2016-11-06 17:01:11	1	88C25532CF72	0.191064
2016-11-06 17:01:31	1	88C25532CF72	0.342167
2016-11-06 17:01:51	1	88C25532CF72	0.412494

Tabela 3.5: Tabela Experimento para Distância em 50cm

Distância em 3m

Para este caso o funcionamento do sistema apresentou resultados estáveis porém com uma variância maior, os resultados foram apresentados na tabela 3.6. O experimento foi realizado com uma distância real medida em 3m com poucos obstáculos.

Timestamp	**DeviceID**	**Endereço Físico**	**Distância**
2016-11-06 17:12:33	1	88C25532CF72	0.842868
2016-11-06 17:12:55	1	88C25532CF72	1.27444
2016-11-06 17:13:15	1	88C25532CF72	1.60297
2016-11-06 17:13:35	1	88C25532CF72	1.79556
2016-11-06 17:13:56	1	88C25532CF72	1.27444
2016-11-06 17:14:35	1	88C25532CF72	1.60297
2016-11-06 17:15:26	1	88C25532CF72	1.13524
2016-11-06 17:15:44	1	88C25532CF72	2.24659
2016-11-06 17:16:05	1	88C25532CF72	2.00945

Tabela 3.6: Tabela Experimento para Distância em 3m

Distância em 5m

Para este caso o funcionamento do sistema apresentou resultados estáveis porém com uma variância maior e a presença de um ponto fora da curva, os resultados foram apresentados na

tabela 3.7. O experimento foi realizado com uma distância real medida em 5m com obstáculos e a presença de uma parede.

Timestamp	DeviceID	Endereço Físico	Distância
2016-11-06 17:21:51	1	88C25532CF72	4.28784
2016-11-06 17:22:12	1	88C25532CF72	3.1195
2016-11-06 17:22:32	1	88C25532CF72	3.1195
2016-11-06 17:22:53	1	88C25532CF72	4.28784
2016-11-06 17:23:13	1	88C25532CF72	4.28784
2016-11-06 17:23:33	1	88C25532CF72	3.8609
2016-11-06 17:23:54	1	88C25532CF72	3.8609
2016-11-06 17:24:18	1	88C25532CF72	18.0776

Tabela 3.7: Tabela Experimento para Distância em 5m

Distância em 14m

Para este caso o funcionamento do sistema apresentou resultados variados mas no geral a estimativa está dentro do esperado, os resultados foram apresentados na tabela 3.8. O experimento foi realizado com uma distância real medida em 14m com obstáculos e a presença de uma parede.

Timestamp	DeviceID	Endereço Físico	Distância
2016-11-06 17:24:18	1	88C25532CF72	18.0776
2016-11-06 17:26:28	1	88C25532CF72	13.819
2016-11-06 17:26:48	1	88C25532CF72	8.65181
2016-11-06 17:27:09	1	88C25532CF72	13.819
2016-11-06 17:27:29	1	88C25532CF72	13.819
2016-11-06 17:27:49	1	88C25532CF72	13.819
2016-11-06 17:28:09	1	88C25532CF72	9.52156

Tabela 3.8: Tabela Experimento para Distância em 14m

3.3.2 Resultados Experimentais

O teste de validação do sistema foi feito para provar se todos objetivos propostos anteriormente foram atingidos e verificar também como se porta com vários módulos funcionando ao mesmo tempo, o cenário é de dois processadores embarcados (Raspberry Pi) atuando como *Scanners* de *iBeacons* separados em dois móveis, e um *Bluetooth* configurado como *iBeacon* que foi movimentado de uma sala para outra. A parte de infraestrutura foi pensada simulando

um hospital logo cada Raspberry Pi estava conectada na mesma rede em que o Servidor de Dados. O teste consistiu em verificar o resultado final na interface, se o que acontecia na realidade era verificada pela resposta do programa. O sistema se comportou bem e apresentou bons resultados, principalmente quando o *iBeacon* se aproximava mais de cada *Scanner* de *Ibeacons*, justamente pelo fato de menores distâncias apresentarem menores erros. Pela figura 3.15 é apresentada a tela para o usuário demostrando o resultado.

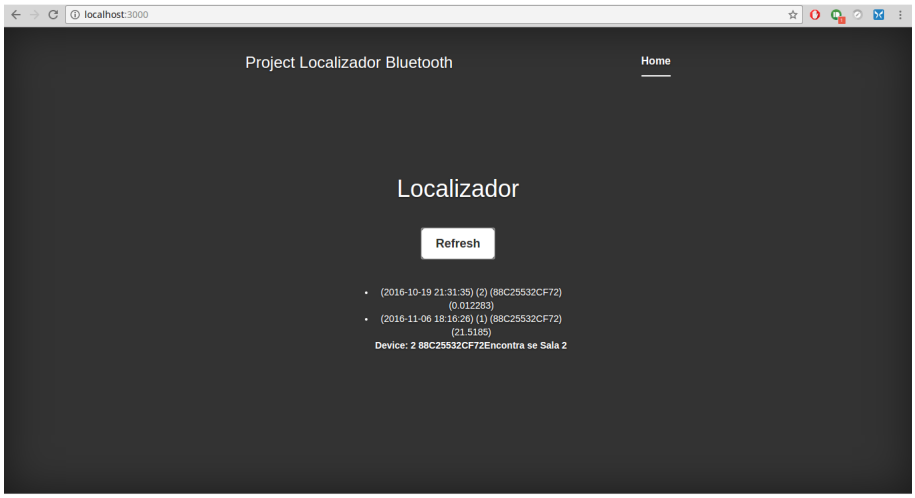

Figura 3.15: Tela para usuário final

3.4 Dificuldades e Limitações

A presença do módulo HM-10 para este projeto foram experiências tanto positivas como negativas, positivas pelo fato de apresentarem uma grande comunidade que utiliza este módulo e por apresentar um baixo custo. Porém a principal dificuldade com este dispositivo foi pelo fato de muitos fabricantes venderem como originais sendo que não o são, pela figura 3.16 é possível diferenciar o módulo original da cópia. O problema raiz é a falta de suporte dessas empresas, não existe material nem *datasheet* confiável para conferir as características do produto. Um caso que ocorreu neste projeto foi a necessidade de atualização de *firmware* para ter acesso à uma nova funcionalidade do produto *Bluetooth* 4.0, pela falta de documentação da empresa não foi possível efetuar a atualização da cópia de *bluetooth* pois apesar de possuir um extithardware similar e funções similares, o *firmware* não suportava a atualização. A solução foi obter a versão do aparelho com suporte e documentação correta. Uma outra limitação encontrada para

o dispositivo é para a função nativa de *scanner* de *iBeacons*, acreditava se que a performance deste *scan* seria de alguns milisegundos, porém o encontrado foi de alguns segundos o que limitou muito a performance o sistema inteiro pois este dispositivo que gerará informação para o sistema inteiro, a interação central do sistema inteiro transformou se em um gargalo.

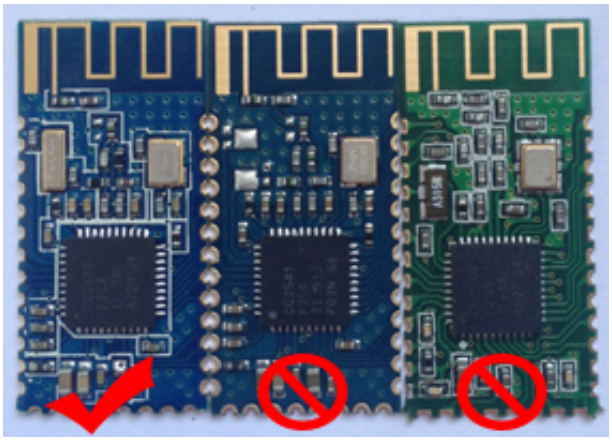

Figura 3.16: Módulos HM-10 Original e Cópias

4 Conclusão

Com o presente projeto implementado julga se que atingiu se o objetivo de realizar a localização usando esta arquitetura, mesmo considerando os erros dos resultados foram satisfatórios, o *software* foi robusto e se portou bem em relação aos resultados apresentados. Porém a performance não foi a esperada, o gargalo apresentado pelo módulo HM-10 não era esperado e impacta o sistema como um todo pois este projeto é um sistema em tempo real, por isso quanto maior a frequência de *refresh* de informação mais preciso será o sistema em relação ao mundo real. Todo a parte de *software* e todo *hardware* utiliza ferramentas *open source*, o que traz muitas vantagens para o sistema como um todo pois permite uma customização de cada módulo e principalmente melhor aprendizado e entendimento de cada ferramenta chegando ao nível mais baixo que é o código. Outro fator importante é o fato da ferramenta open source apresentar uma grande comunidade que pode auxiliar novos usuários para a utilização. E por isso que todo este projeto foi implementado com esta ideia, e será disponibilizada em um *website online* para que qualquer um que quiser utiliza-lo ou modificá-lo. O desenvolvimento deste projeto trouxe muito conhecimento para a formação acadêmica, visto que foi necessário aprender mais sobre certas áreas, principalmente a de sistemas embarcados. Esta experiência com embarcados trouxe além de conhecimentos, revelou oportunidades que podem ser criadas, a combinação de processadores embarcados com *hardware* é algo que pode ser utilizado em todas as áreas e pelo fato do material ter baixo custo facilita a implementação de qualquer tipo de projeto, basta ter uma ideia.

Todo conteúdo desenvolvido pode ser encontrado em :

`https://github.com/simoesusp/Workshop-Raspberry/tree/master/SoftwareRasp/`

`iBeacon`

4.1 Trabalhos Futuros

Os principais gargalos do sistema poderão ser resolvidos com a nova implementação do *Bluetooth* 5.0, pelos requisitos informados no site oficial da SIG *Bluetooth* a nova tecnologia apresentará o quadruplo do alcance, o dobro da taxa de velocidade e aumentando a velocidade de transmissão de informação por broadcast em 8 vezes, é esperado que o lançamento desta tecnologia seja em 2017.

5 *Referência*

Use HM-10 as a Central Device or an iBeacon. Disponível em:

http://blog.blecentral.com/2015/05/13/hm-10-central-ibeacon/

Acesso em : 2 Outubro 2016

Bluetooth low energy. Disponível em:

https://en.wikipedia.org/wiki/Bluetooth_low_energy

Acesso em : 2 Outubro 2016

Getting Started with Bluetooth Low Energy. Disponível em:

https://www.safaribooksonline.com/library/view/getting-started-with/97814919005
ch01.html

Acesso em : 2 Outubro 2016

Bluetooth 4.0 BLE module Datasheet. Disponível em:

https://www.seeedstudio.com/wiki/images/c/cd/Bluetooth4_en.pdf

Acesso em : 15 Setembro 2016

Upgrading Firmware to HM-10 CC2541 BLE 4.0. Disponível em:

https://suryaigor.wordpress.com/2016/02/05/upgrading-firmware-to-hm-10-cc2541-b

Acesso em : 15 Outubro 2016

PARTHEL,N.**Comunicação sem fio entre microcontroladores na automação residencial.**

2015, Trabalho de conclusão de curso - Curso Engenharia Mecatrônica, USP, São Carlos

GODOY,P.**Tecnologia RFID: Uma proposta de sistematização na gestão hospitalar**

2011, Trabalho de conclusão de curso - Curso Engenharia Elétrica, USP, São Carlos