UNIVERSIDADE DE SÃO PAULO
Instituto de Ciências Matemáticas e de Computação

Departamento de Sistemas de Computação

Desenvolvimento de sistema de localização de equipamentos hospitalares baseado em iBeacons de Bluetooth

Danilo Augusto Yudi Horikome

São Carlos - SP

Desenvolvimento de sistema de localização de equipamentos hospitalares baseado em iBeacons de Bluetooth

Danilo Augusto Yudi Horikome

Orientador: Eduardo do Valle Simões

Monografia referente ao projeto de conclusão de curso dentro do escopo da disciplina SSC0670 - Projeto de Formatura I do Departamento de Sistemas de Computação do Instituto de Ciências Matemáticas e de Computação – ICMC-USP para obtenção do título de Engenheiro de Computação.

Área de Concentração: Controle e Automação

USP – São Carlos
06/11/2016

“O sucesso é uma consequência e não um objetivo”

(Gustave Flaubert)

Agradecimentos

Aos meus pais e irmã, por todo apoio, toda educação, valores e ensinamentos dados a mim. São minha base e pilar de tudo que possa ter conquistado em minha vida, a qual serei eternamente agradecido.

A meus amigos de longa data, que continuam a estar presente em minha vida não importando o tempo ou distância.

A todas amizades feitas durante a graduação e por toda companhia, sejam elas nas festas, nos estudos, nas conversas ou em momentos de dificuldade.

Aos famosos *"rushes do boi doido"*, capaz de unir todo uma turma para estudar em conjunto e que possibilitou muito aprendizado.

Aos professores que se dedicaram ensinar e compartilhar seus conhecimentos.

Ao meu orientador Eduardo Simões, por auxiliar no desenvolvimento desse projeto, além de prover o suporte e equipamentos para que fosse possível a sua conclusão.

Para todos aqueles que contribuíram de alguma forma por toda essa jornada.

Um muito obrigado.

Resumo

Um problema recorrente encontrado em diversos hospitais do país é falta de informações sobre os equipamentos hospitalares pertencentes ao inventário que o compõe. As consequências ocasionadas podem ser muitas, entre elas não saber sua localização em momentos de sua necessidade, com uma demora para realizar o rastreamento. Isso afeta o atendimento aos pacientes, que pode levar mais tempo, ou até mesmo o desaparecimento desses equipamentos. Isso motivou a ideia desse trabalho, que é criar um sistema de localização de equipamentos hospitalares, onde através de uma interface visual pode-se rapidamente localizá-los. Com isso dar aos usuários do sistema um controle de todo os aparelhos hospitalares e agilizar a tomada de decisão relativo a recepção dos pacientes. A forma encontrada para tornar possível esse projeto foi através dos *beacons*, pequenos dispositivos de baixo consumo de energia, capaz de enviar dados através da tecnologia *Bluetooth*. Com ele acoplado nos equipamentos é possível rastreá-los, onde computadores extremamente portáteis chamados *Raspberry Pi* são responsáveis por essa função e por processar todos os dados, que são enviados para um banco de dados. Então a interface visual web fica a cargo de extrair as informações dessa base e os exibir de forma simples e prática para os usuários. Durante o decorrer dessa monografia, serão detalhados todos os aspectos abordados para o desenvolvimento do trabalho, desde os conceitos e ideias utilizados, até implementação e testes feitos para validação desse sistema.

Sumário

Lista de Abreviaturas

BLE.....*Bluetooth Low Energy*

SCO.....*Synchronous Connection-Oriented*

ACL.....*Asynchronous Connection-Less*

Lista de Gráficos

Lista de Figuras

CAPÍTULO 1: INTRODUÇÃO

1.1. Contextualização e Motivação

Uma parte considerável dos hospitais no Brasil atualmente, principalmente as da rede pública, carecem muitas vezes de infraestrutura adequada para melhor atender seus pacientes. Há falta de espaço, de profissionais, equipamentos e até medicamentos dentro do ambiente hospitalar, uma infeliz realidade, que frequentemente é noticiado nos diversos meios de comunicação. Uma das consequências desses problemas pode ser vista nas longas filas de espera que as pessoas são submetidas para serem atendidas, onde em casos mais graves de saúde podem rapidamente levar a piora do quadro clínico e até a fatalidade, caso paciente não seja tratado agilmente.

Uma forma de amenizar esse tempo de espera do atendimento é uma boa administração dos recursos disponíveis nos hospitais, como distribuição correta dos médicos, enfermeiros e pacientes, em conjunto da utilização inteligente de salas de operação, pronto-socorro e atendimento. A locomoção das pessoas que precisam de cuidados, para as salas onde os recursos necessários (como equipamentos hospitalares) estão disponíveis de forma de fácil acesso para utilização dos profissionais, faz com que menos tempo seja usado para alocação, e sim com o atendimento propriamente dito. Isso diminui a espera dos pacientes pelo tratamento, principalmente em casos mais graves em que cada minuto faz diferença entre salvar ou perder uma vida.

Outro grande problema, que está ligado diretamente aos equipamentos hospitalares, é a escassez de seu monitoramento, onde a falta de dados sobre a localização pode levar a muitos casos de perda mesmos. Como muitos desses instrumentos possuem um alto custo de obtenção, o prejuízo financeiro trazido em consequência dessas possíveis perdas é enorme. Isso junto com a falta de verba em muitos hospitais brasileiros, onde já existem dificuldades para obter todos os equipamentos necessários para o seu pleno funcionamento, acaba por só agravar a situação. Por isso é preciso monitorar de forma eficiente todos os

aparelhos hospitalares presentes no hospital, para manter assim um controle de todo seu inventário.

Com esse controle e levantamento dos equipamentos é possível distribui-los de maneira eficiente pelo ambiente hospitalar. Por exemplo através de um estudo, se pode destinar cada equipamento para o local mais adequado de acordo com sua frequência de uso, e garantir um rápido acesso a esse recurso pelos profissionais e otimizar o tempo dos atendimentos dos pacientes. Há o ganho também de velocidade na tomada de decisão, pois uma vez que se sabe a posição de cada equipamento, pode-se alocar o paciente a ser atendido na sala mais próxima equipado com os aparelhos necessários para realizar os procedimentos médicos. Consequentemente há uma rápida resposta, principalmente para os pacientes em estados mais graves.

Diante desse contexto, a motivação principal por trás desse trabalho é ajudar os profissionais que atuam nos hospitais, a conseguirem otimizar o tempo de atendimento dos pacientes. Além disso ter todo um controle de inventário e auxiliar na tomada de decisão dentro do hospital, utilizando a tecnologia vinda dos *beacons* como *core* do projeto.

1.2. Objetivos

O projeto tem por função desenvolver, implementar, testar e validar um sistema de localização dos equipamentos hospitalares. Os principais objetivos são configurar os módulos *Bluetooth* HM-10 para atuarem como *beacons* e implementar um algoritmo para que a *Raspberry Pi* consiga identificá-los, além de receber e processar os dados proveniente deles, para finalmente enviá-los a um banco de dados. Por último criar uma interface visual através de uma página web, para extrair as informações dessa base e os exibir de forma simples e prática para os usuários, e consequentemente facilitar a visualização sobre a localização dos equipamentos, além de trazer maiores detalhes para o usuário, como data e hora da localização, qual o ambiente anteriormente o aparelho foi rastreado e a que distância se encontra.

1.3. Organização do Trabalho

No Capítulo 2 é apresentada a revisão bibliográfica dos principais conceitos utilizados durante o desenvolvimento do trabalho. O Capítulo 3 cita as ferramentas, utilitários e dispositivos de *hardware* usados, enquanto o Capítulo 4 descreve em detalhes como o projeto foi estruturado e desenvolvido, testes feitos, os resultados e a conclusão obtida. Finalmente no Capítulo 5 são apresentadas as relações do curso com o projeto, além de ideias para trabalhos futuros.

CAPÍTULO 2: REVISÃO BIBLIOGRÁFICA

2.1. Considerações Iniciais

Este capítulo tem como objetivo explanar os principais conceitos adotados para o desenvolvimento do projeto e que ajudaram no rumo e decisões a serem tomadas, de forma que as ideias sigam uma abordagem correta para soluções dos problemas.

2.2. iBeacon

É dado o nome *iBeacon* para o padrão criado pela *Apple* para um sistema de proximidade em ambientes fechados, com objetivo de fazer com que aplicativos móveis consigam perceber a sua geolocalização em pontos específicos, como dentro lojas, mercados e restaurantes por exemplo. Isso dá poder aos aplicativos, que podem entregar um conteúdo baseado no contexto do local em que o usuário se encontra, e com isso uma experiência mais imersiva. Um exemplo prático em como essa tecnologia pode ser aplicada são em museus, que podem disponibilizar um aplicativo que os visitantes podem instalar em seus *smartphones*. Através da tecnologia *iBeacon* quando o usuário estiver perto de uma obra de arte específica, o aplicativo é capaz de detectar essa proximidade e assim enviar informações como autor, característica da obra e curiosidades. Há uma infinidade de funcionalidades que se tornam possível com o advento dessa tecnologia, desde potencializar vendas de um produto, como usá-lo como localizador de equipamentos dentro de hospitais que é o objetivo desse projeto (IBEACON, 2016).

O conceito do *iBeacon* se dá a partir da comunicação de aplicativos com os *beacons*, que são pequenos dispositivos transmissores de sinais, através de tecnologia *Bluetooth Low Energy* (BLE) ou *Bluetooth* 4.0, que é descrito no tópico subsequente. Esses sinais contêm pequenos pacote de dados sobre o *beacon* emissor, que no padrão de protocolo da *Apple Inc.* possui quatro blocos de informações principais na seguinte ordem (APPLE INC., 2014):

UUID: É uma *string* de 16 bytes com propósito de diferenciar grandes conjunto de *beacons* relacionados. Por exemplo se uma loja de roupas utiliza uma rede de *beacons*, todos eles possuem o mesmo UUID, assim caso um aplicativo processe os sinais emitidos, é possível identificar que se trata dos *beacons* de propriedade dessa loja.

Major: *String* com 2 bytes que é usado para identificar um grupo menor de *beacons* dentro de um conjunto maior. Dentro do contexto da loja de roupas, os *beacons* possuem o mesmo Major dentro de um departamento por exemplo.

Minor: Novamente uma *string* de 2 bytes que é um identificador único para cada *beacon*. Na loja de roupas poderia ser identificador referente para cada modelo de roupa.

TX Power: Usado para determinar a proximidade do *beacon*, através da intensidade do sinal em decibéis que é enviado do *beacon* emissor a uma distância de 1 metro do receptor, sendo esse valor constante para cada dispositivo e é usado como referência. A partir desse dado, pode-se estimar a distância entre os dispositivos.

Se pode perceber que o protocolo segue uma hierarquia simples, com intuito de identificar desde grandes grupos de *beacon* até individualmente cada um. Com os pacotes de dados, eles podem ser processadas e diferentes aplicações podem ser desenvolvidas a partir das informações obtidas. É importante ressaltar que dispositivos que irão interagir com os *beacons* necessitam possuir também o padrão BLE para que haja a comunicação entre eles.

2.3. Bluetooth

O *Bluetooth* é um padrão para comunicação sem fio através de *streaming* de dados contínua, usando frequência de rádio de curto alcance na faixa de 2.4 GHz a 2.485 GHz. Ou seja, provê uma forma de se conectar e trocar informações entre diferentes dispositivos a curta distância. O objetivo principal do *Bluetooth* é ser um protocolo padrão de comunicação, com baixo consumo de energia, baixo alcance e com microchips transmissores de baixo custo. Como a troca de dados é feita por radiofrequência, um dispositivo pode detectar outro

independentemente da posição, sendo necessário apenas que estejam dentro do limite de proximidade. O alcance máximo foi dividido em três classes (ALECRIM, 2013):

- **Classe 1:** potência máxima de 100 mW (miliwatt), alcance de até 100 metros;
- **Classe 2:** potência máxima de 2,5 mW, alcance de até 10 metros;
- **Classe 3:** potência máxima de 1 mW, alcance de até 1 metro.

Classes distintas podem se comunicar entre si sem problemas, desde que estejam dentro do alcance da respectiva classe. As taxas de transmissão de dados variam bastante, com as primeiras versões com velocidade de 1 Mb/s (megabit por segundo) chegando até 25 Mb/s nas últimas versões (ALECRIM, 2013).

Pelo fato de a faixa de frequência de rádio utilizada é aberta, são necessárias providências para que não haja interferências e nem que as mesmas sejam geradas. O método de transmissão de sinal FH-CDMA (*Frequency Hopping - Code-Division Multiple Access)*[1], usado pelo *Bluetooth*, faz essa proteção, dividindo a frequências em vários canais. Quando uma conexão for estabelecida, o dispositivo alterna entre esses canais de uma maneira muito rápida, o que é chamado *frequency hopping* ou "salto de frequência", permitindo uma baixa largura de banda e diminuindo as chances de interferência. São permitidos até 79 canais dentro da faixa utilizada, com intervalos de 1 MHz entre elas (ALECRIM, 2013).

A tecnologia *Bluetooth* é *full-duplex*, ou seja, pode tanto receber quando enviar dados. A transmissão é intercalada em *slots* de envio e recebimento, no chamado *Frequency Hopping / Time Division Duplex* (FH/TDD). Os *slots* são os canais divididos em períodos de 625 µs (microssegundos) e a cada salto de frequência é ocupado por um *slot*, onde em apenas 1 segundo há 1600 desses saltos.

[1] FH-CDMA é uma técnica de modulação básica utilizada na transmissão de sinal de espectro espalhado.

Na relação entre emissor e receptor, o *Bluetooth* utiliza dois padrões, *Synchronous Connection-Oriented* (SCO) e *Asynchronous Connection-Less* (ACL). No primeiro tipo é estabelecido um *link* sincronizado entre os dispositivos que vão emitir e receber os dados, com separação de *slots* para cada. No padrão SCO não é possível retransmitir um pacote de dados perdidos, o que pode gerar ruídos, portanto é mais indicado para uma transmissão contínua de dados. Com o padrão ACL o link feito entre os dispositivos é assíncrono, pois os *slots* ficam previamente livres para serem gerenciados. Ao contrário do SCO, nesse padrão é possível reenviar pacotes de dados perdidos, o que garante a integridade das informações a serem trocadas.

A comunicação entre dois ou mais dispositivos utilizando o *Bluetooth*, formando uma rede, dá se o nome de *Piconet*. O dispositivo que inicia a comunicação é chamado de mestre e todos os outros dispositivos que se comunicam com ele são os escravos. O papel do mestre é regular toda a transmissão de dados e fazer o sincronismo entre os dispositivos na rede (ALECRIM, 2013).

A seguir será listado as principais camadas do protocolo de transporte utilizadas no *Bluetooth*:

Radio Frequency: camada responsável pelos aspectos da utilização de radiofrequência.

Link Manager Protocol: tem a função de gerenciar diferentes aspectos da comunicação propriamente dita, administrando a taxa de transferência, parâmetros de autenticação, níveis de potência, criptografia, entre outros.

Baseband: camada que determina como diferentes dispositivos se localizam e comunicam uns com os outros por *Bluetooth*. Aqui se gerencia como os mestres e escravos se relacionam na *Piconet* e onde os padrões SCO e ACL atuam.

***Host Controller Interface*:** é a interface de comunicação com o *hardware* emissor do sinal *Bluetooth*, para garantir que diferentes dispositivos sejam compatíveis entre si.

Logical Link Control and Adaptation Protocol: tem a função de lidar com parâmetros de *Quality of Service* (QoS) e fazer a ligação entre camadas superiores e inferiores.

Os protocolos de *middleware* visam permitir a compatibilidade de aplicações já existentes por meio de protocolos e padrões de outras entidades, como *Point-to-Point Protocol, Internet Procotol, Wireless Application Procotol e Object Exchange.* Por fim os protocolos de aplicação, são as diferentes aplicações que fazem uso do *Bluetooth* em seus equipamentos, como por exemplo os fones de ouvidos sem fio (ALECRIM, 2013).

2.3.1. Bluetooth Low Energy (BLE ou 4.0)

Esta é a versão mais recente utilizada atualmente da tecnologia *Bluetooth* e sua principal novidade é o menor consumo de energia em aparelhos que não exigem a transferência de grandes volumes de dados. Isso permite ser alimentado por baterias como pilhas convencionais e ainda tenha autonomia de várias semanas ou até meses dependendo do seu uso, pois seu consumo médio é de apenas 1 μA (miliampère). Mas com a melhoria da gestão energética, a taxa de transferência sofreu uma diminuição para 1 Mb/s em comparação da taxa de 3 Mb/s da versão 3.0. Uma grande diferença em relação a essa versão e o *Bluetooth* convencional é que esta permanece em modo *Sleep* constantemente, exceto quando uma conexão é iniciada. O tempo de conexão dura poucos milissegundos, diferentemente do *Bluetooth* convencional que pode durar mais de 100 milissegundos. Apesar de características em comum com *Bluetooth*, o BLE não atua da mesma forma que o mesmo, e, portanto, não é compatível com a versão anterior. Em resumo o *Bluetooth* é capaz de lidar com grande quantidade de dados, mas consome mais energia e tem um custo maior. Já o BLE foi desenvolvido para aplicações que não requisitem uma grande troca de dados, podendo funcionar com uma bateria pequena por muito tempo e a um baixo custo (BLUETOOTH..., 2016).

2.5. Banco de Dados

Segundo (KORTH e SILBERSCHATZ, 1994), um banco de dados "é uma coleção de dados inter-relacionados, representando informações sobre um domínio específico", ou seja, pode-se dizer que quando há um grupo de informações que possuem relação entre si e que englobam um mesmo tema, existe um banco de dados. Esse conceito visa poder armazenar os dados de forma estruturada e de diminuir ao máximo a redundância.

Com o constante crescimento da informatização no mundo, a geração de dados cresce vertiginosamente, e um banco de dados tem a utilidade de disponibilizar a seus usuários a possibilidade de realizar consultas, atualizações e inserções / remoções de uma grande quantidade de dados digitais de uma maneira simplificada.

Para criar e gerenciar os bancos de dados existem *softwares* com essa finalidade de administrar e manipular as informações provenientes dessas bases e controlar o acesso dos usuários. São chamados sistema de gerenciamento de banco de dados (SGBD) e temos como alguns exemplos: PostgreSQL, MySQL, Oracle, SQL Server e DB2.

Os bancos de dados e SGBD são partes do sistema de banco de dados, que possui quatro componentes: os dados, *software*, *hardware* e usuários, que são englobados como mostrado na Figura 3. O objetivo do sistema é abstrair os dados, ou seja, isolar o usuário de detalhes da implementação do banco de dados e fazer com que esses dados sejam independentes em relação às aplicações. Para garantir essa abstração, são feitas em três níveis:

- **Nível do Usuário:** quais partes do banco de dado os usuários, ou um grupo deles, podem acessar de acordo com suas necessidades.
- **Nível Conceitual:** quais dados serão armazenados no banco e como é a relação entre eles.
- **Nível Físico:** é o nível mais baixo, e define como os dados serão armazenados fisicamente de fato.

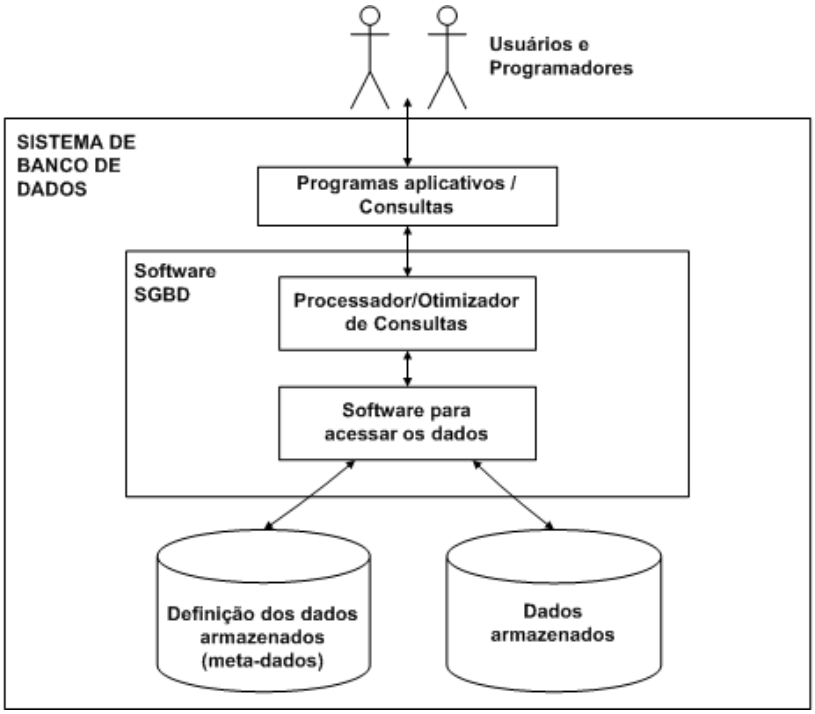

Figura 1: Sistema de Banco de Dados.

Fonte: Página do professor André Sanchez do IME da Universidade de São Paulo[2].

2.4. Considerações Finais

Este capítulo trouxe uma explanação dos principais conceitos usados no decorrer do projeto. No próximo capítulo serão listados as ferramentas e dispositivos de *hardware* utilizados para tornar possível o rastreamento dos equipamentos hospitalares.

[2] http://www.ime.usp.br/~andrers/aulas/bd2005-1/aula5.html

CAPÍTULO 3: METODOLOGIA

3.1. Considerações Iniciais

Este capítulo tem o objetivo de ilustrar os principais dispositivos de *hardware* e ferramentas de *software* utilizadas ao longo do projeto. Os módulos *bluetooth* HM-10 são componentes chave para o desenvolvimento por sua atuação como *beacon* e a *Raspberry Pi* possui grande importância por processar os dados e enviá-los para o banco de dados. Também é comentado sobre o servidor *Apache* e o banco de dados *MySQL*.

3.2. Módulo Bluetooth HM-10

É o hardware utilizado no projeto capaz de emitir dados através do padrão *Bluetooth* 4.0 ou *Bluetooth Low Energy* (BLE). Esse módulo é baseado no *chipset* CC2541 da *Texas Instruments* e responde por comandos AT[3], podendo operar nos modos Mestre / Escravo. Seu uso se dá por comunicação serial, possuindo pinos de RX e TX para se conectar no microcontrolador / dispositivo que possua pinos UART[4]. Algumas de suas características são (KARYDIS, 2015):

- Utiliza *Bluetooth Low Energy* 4.0
- Frequência de Operação de 2.4 GHz da banda ISM (do inglês *industrial, scientific and medical*)

[3] Derivado do padrão "Heyes", usado para que computadores pudessem interagir com conexões telefônicas através de um modem.

[4] *Universal Asynchronous Receiver/Transmitter* – Dispositivo de comunicação serial a grandes distâncias e com sincronização feita por software.

- Fonte de Alimentação de 2 a 3.7 Volts
- Entrada e Saída (Serial RX e TX) opera com tolerância até 3.3 Volts
- Alcance de até 100 metros em campo aberto

Os módulos utilizados são de extrema importância para o desenvolvimento do projeto, pois são configurados para atuarem como *beacons*. Excepcionalmente um módulo HM-10 foi utilizado juntamente com a *Raspberry Pi 1 Model B*, para que fosse possível fazer a comunicação do mesmo com os *beacons*, uma vez que o modelo disponível para uso não possui módulo *Bluetooth* integrado a sua placa como nas versões mais recentes. O esquemático do módulo com seus pinos é mostrado na Figura 2.

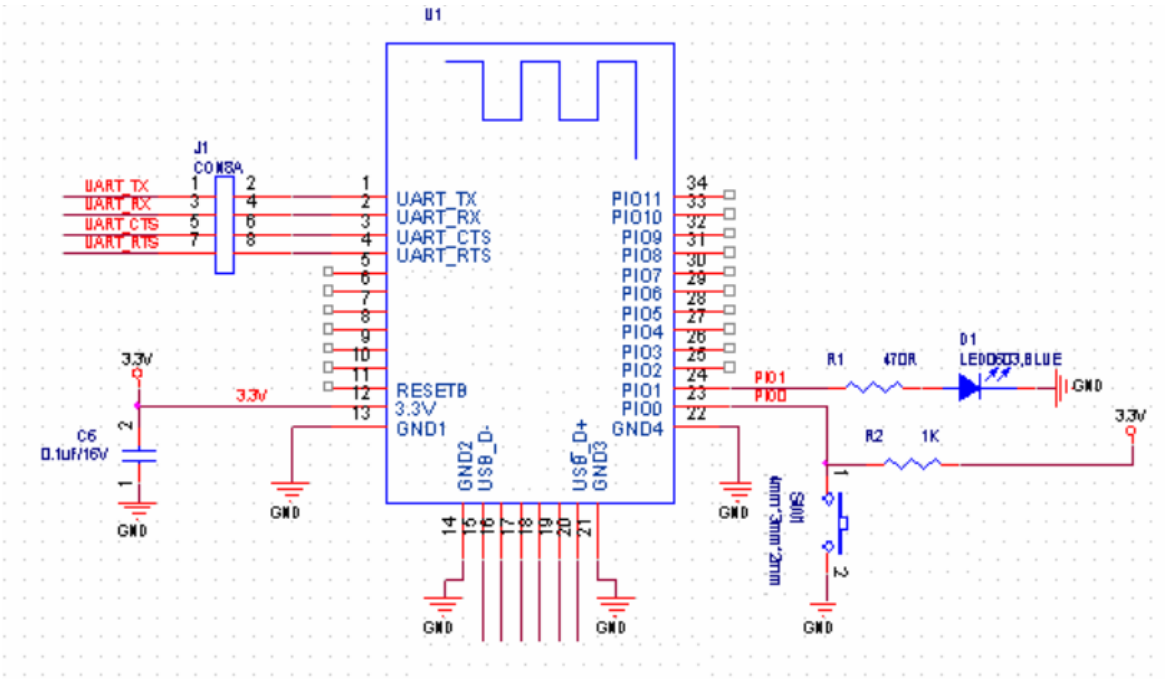

Figura 2: Esquemático do Módulo HM-10.

Fonte: Datasheet do Módulo HM-10.

3.3. Arduino

Arduino é um projeto de plataforma open-source de prototipagem eletrônica baseado em kits de placas com microcontroladores da marca *Atmel AVR*, com suporte de entrada e saída e usando linguagem de programação C/C++. Tem como objetivo ser uma ferramenta simples de aprender e ser usada, voltado a todas as pessoas interessadas em criar interatividade com objetos e ambiente.

Há diversos modelos de *Arduino*, sendo que o modelo utilizado nesse projeto é o *Arduino Nano*, que apresenta as seguintes características (ARDUINO, 2014):

- **Microcontrolador:** ATmega328 de 16 MHz.
- **Voltagem de Operação (nível lógico):** 5 Volts.
- **Voltagem de Entrada (recomendado):** 7-12 Volts.
- **Voltagem de Entrada (limite):** 6-20 Volts.
- **Pinos de Entrada/Saída Digitais:** 14 (das quais 6 provém saída PWM[5]).
- **Pinos de Entrada Analógicos:** 8
- **Corrente Contínua por Pino E/S:** 40 mA.
- **Memória Flash:** 32 KB com 2 KB usado para bootloader.
- **SRAM:** 2 KB.
- **EEPROM:** 1 KB.

[5] Do inglês *Pulse Width Modulation*, que é uma técnica onde se fixa a frequência de uma onda quadrada e que se varia o tempo que sinal permanece em nível lógico alto.

O *Arduino Nano* pode ser alimentado via conexão *Mini-B USB*, ou fornecida por uma fonte externa não regulada de 6-20 Volts no pino 30 ou então por uma fonte externa regulada de 5 Volts no pino 27. A fonte de alimentação é automaticamente selecionada pela maior voltagem. O esquemático com os pinos pode ser visto na Figura 3.

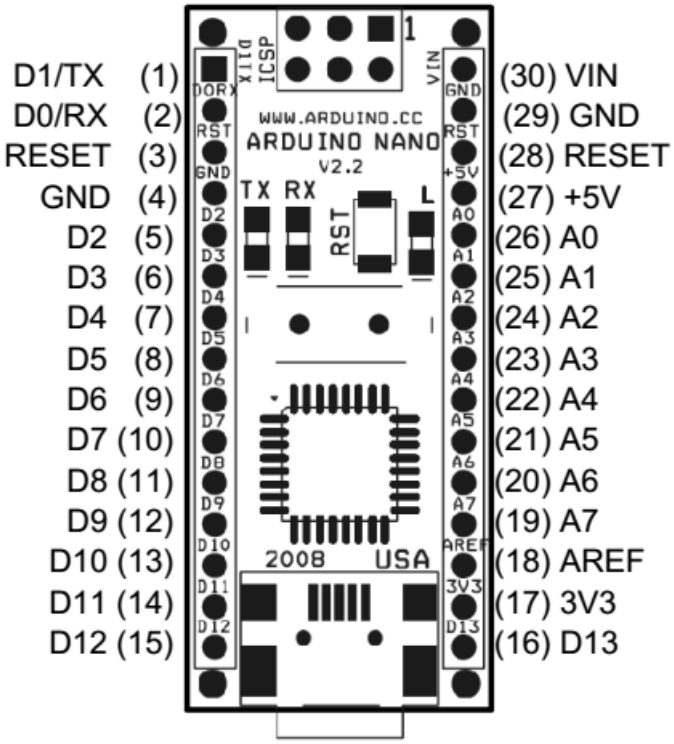

Figura 3: *Layout* dos pinos do *Arduino Nano*.

Fonte: Datasheet do Arduino Nano.

A entrada e saída de dados podem ser feitas nos 14 pinos digitais (D0 – D13) presentes no *Arduino Nano*. A seguir são descritos detalhes dos pinos utilizados para se conectar e comunicar com o módulo HM-10.

- **Serial:** pinos 1 (TX) e 2 (RX) usados para receber (RX) e transmitir (TX) dados pela TTL Serial.
- **Alimentação:** pinos 29 (terra) e 13 (tensão de saída 3.3 Volts).

Para fazer a comunicação entre o *Arduino Nano* e outros dispositivos, o microcontrolador *ATmega328* disponibiliza comunicação serial UART TTL de 5 volts, através dos pinos digitais 0 (RX) e 1 (TX).

Para o desenvolvimento de programas para o *Arduino* é disponibilizado uma IDE (do inglês *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado) própria, chamado *Arduino Software*. Nele é possível escrever códigos, compilá-los e enviar para o *Arduino*, de maneira simplificada. Dentro da IDE existem ferramentas, entre elas o *Serial Monitor*, uma interface simples para comunicação serial entre *Arduino* e o dispositivo a ele conectado.

3.4. Raspberry Pi

Raspberry Pi é um computador totalmente funcional de tamanho muito reduzido se comparado a *desktops* e *notebooks* convencionais, onde em uma única placa está todos os componentes de *hardware*. É desenvolvido no Reino Unido pela *Fundação Raspberry Pi*[6] e teve como intenção inicial ser uma ferramenta de educação para jovens alunos se aprofundarem na área de programação e entendimento de *hardware*. Mas devido a seu tamanho compacto e preço acessível, o produto atraiu atenção de públicos de diversas áreas, interessados em projetos que necessitem mais do que microcontroladores básicos como o *Arduino* citado no tópico 2.3, pois o *Raspberry Pi* é capaz de rodar o sistema operacional *Linux / Windows* e proporcionar todos os recursos e funcionalidades advindas disso, com um baixo consumo de energia.

Há diversos modelos do *Raspberry Pi* que podem ser adquiridos, onde cada modelo traz diferentes especificações técnicas. Para este projeto foi utilizado um dos primeiros modelos lançados, o *Raspberry Pi 1 Model B* exibida na Figura 4.

[6] www.raspberrypi.org

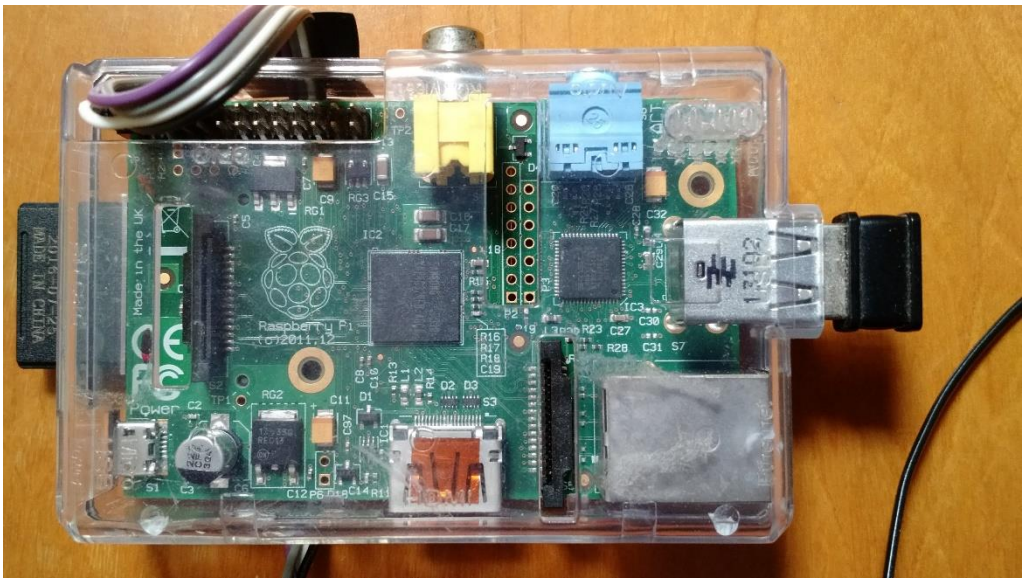

Figura 4: Raspberry Pi 1 Model B.

Sua especificação técnica são (RASPBERRY, 2016):

- **Arquitetura:** ARMv6 de 32 bits.
- **Processador:** ARM1176JZF-S Single Core de 700 MHz.
- **Placa Gráfica:** Broadcom VideoCore IV.
- **Memória SDRAM:** 256 MB compartilhada com o precessador.
- **USB**: Duas portas USB 2.0.
- **Entrada de Vídeo:** Conector CSI de 15 pinos.
- **Saída de Vídeo:** Porta HDMI 1.3
- **Saída de Áudio:** Áudio analógico com entrada para conector 3.5 mm e áudio digital pelo HDMI.
- **Armazenamento:** *Slot* para cartões SD, MMC, SDIO.

- **Rede:** Entrada para cabo Ethernet de 10/100 Mbit/s.
- **Consumo de Energia:** 700 mA (3.5 W).
- **Fonte de Energia:** 5 Volts via Micro USB.

Assim como o *Arduino*, está disponível pinos para entrada e saída de dados, que é mostrado na Figura 5 e suas respectivas funcionalidades.

GPIO#	2nd func.	Pin#	Pin#	2nd func.	GPIO#
	+3.3 V	1	2	+5 V	
2	SDA1 (I²C)	3	4	+5 V	
3	SCL1 (I²C)	5	6	GND	
4	GCLK	7	8	TXD0 (UART)	14
	GND	9	10	RXD0 (UART)	15
17	GEN0	11	12	GEN1	18
27	GEN2	13	14	GND	
22	GEN3	15	16	GEN4	23
	+3.3 V	17	18	GEN5	24
10	MOSI (SPI)	19	20	GND	
9	MISO (SPI)	21	22	GEN6	25
11	SCLK (SPI)	23	24	CE0_N (SPI)	8
	GND	25	26	CE1_N (SPI)	7

Figura 5: Pinos da Raspberry Pi 1 Model B.

Fonte: Página sobre Raspberry Pi no Wikipedia[7].

Para o trabalho foi necessário utilizar apenas os pinos para conectar o módulo HM-10 com a *Raspberry Pi*, sendo eles os pinos 1 (+3.3V), 6 (terra), 8 (Serial TX) e 10 (Serial RX).

O *Raspberry Pi* possui suporte para sistemas operacionais, principalmente baseado em *Linux*, que atualmente estão disponibilizados diversas distribuições otimizadas para o mesmo. Um dos mais utilizados é o *Raspian* que tem como base o *Debian*, com uma usabilidade e interface simples e com grande quantidade de ferramentas para o usuário comum utilizar, que foi a escolhida para ser usada no trabalho.

[7] https://en.wikipedia.org/wiki/Raspberry_Pi

3.5. Servidor Apache

O servidor *Apache* é o servidor web mais utilizado no mundo. Em uma pesquisa feita em maio de 2010 pela *Netcraft* (NETCRAFT, 2010) mostra que o Apache foi usado em 54,68% de todos os sites existentes e em mais de 66% dos milhões dos sites com mais acessos que estão disponíveis na web. Por ser um *software* livre, além de ser gratuito para uso, os seus usuários podem fazer alterações no código fonte para adequarem às suas necessidades, e até criar novas funcionalidades.

O *Apache* tem como função básica processar solicitações HTTP (*Hyper-Text Transfer Protocol*), que é o protocolo padrão utilizado na web, e retornar para o solicitante, geralmente os navegadores web, o conteúdo e recursos resultante. Além disso o *Apache* consegue processar vários outros tipos de protocolos, como FTP (*File Transfer Protocol*), HTTPS que é o HTTP juntamente com SSL (*Secure Socket Layer*) que é uma camada de segurança, entre muitos outros.

Uma característica que levou a seu uso no projeto foi sua capacidade de interpretar códigos em PHP, linguagem utilizada na criação da página web desenvolvida. Juntamente com o banco de dados MySQL, se torna uma poderosa combinação para aplicações web. Outras qualidades que podem ser citadas são a excelente performance, não exigindo um *hardware* avançado, compatibilidade com diversas plataformas, segurança e de fácil instalação, sendo que a versão utilizada foi o *Apache 2.4*, em ambiente *Windows 10*. É disponibilizado um tutorial de instalação do *Apache 2.4* no Apêndice A.

3.6. Banco de dados MySQL

MySQL é sistema de gerenciamento de banco de dados (SGBD), com uso da linguagem SQL (do inglês *Structured Query Language*) como interface. Suas principais características são sua alta portabilidade, com suporte para a maioria das plataformas utilizadas atualmente, compatibilidade com diversas linguagens de programação através de módulos de interface, um excelente desempenho e estabilidade. Além disso é muito bem

otimizado, por isso não requer muitos recursos de *hardware*, com a vantagem de ser um *software* livre e possuir interface gráfica para facilitar a utilização e gerenciamento do banco de dados (MySQL, 2016). Para o projeto foi instalado a versão 5.7 do servidor MySQL, em ambiente *Windows 10*, através do MySQL *Installer 5.7*, que conta também com o *Workbench* (interface gráfica para manipulação do banco de dados). Para o passo-a-passo de como instalar e configurar o MySQL, é disponibilizado um tutorial no Apêndice B.

CAPÍTULO 4: DESENVOLVIMENTO DO TRABALHO

4.1. Considerações Iniciais

Este capítulo apresenta os detalhes de como o módulo HM-10 foi configurado para atuar como *beacon*, a implementação do sistema de processamento na *Raspberry Pi*, detalhes da interface visual web, além dos resultados dos testes realizados, as dificuldade e limitações encontradas.

Para facilitar a compreensão dos passos seguidos, será dado uma visão geral do projeto como um todo e suas premissas. Para ser possível a realização de um sistema de localização de equipamento hospitalares, é utilizado os *beacons* para serem atrelados a cada equipamento individualmente. Isso significa que cada *beacon* é único e está ligado fisicamente a um único equipamento, e sua função é enviar pacotes de dados para *Raspberry Pi*, que realiza escaneamento de um ambiente específico, recebe esses pacotes, executa processamento desses dados, calcula a distância que se encontra cada *beacon* encontrado e finalmente envia os dados processados para o banco de dados. Cada *Raspberry Pi* é responsável apenas por um local fixo, portanto todos os registros advindos de um aparelho em específico são referentes a aquele ambiente apenas.

Com os dados gerados e armazenado no banco de dados, a interface visual web é responsável por extrair a informação dessa base e exibi-los de maneira intuitiva para o usuário final.

Todos os códigos desenvolvidos estão disponíveis em https://github.com/danilohorikome/Beacon.

4.2. Módulo HM-10 Implementado como *Beacon*

Para poder alimentar e enviar comandos para o módulo HM-10, primeiramente foi feita a ligação com o *Arduino Nano,* onde pinos 1 (TX), 2 (RX), 29 (terra) e 13 (3.3 Volts) do *Arduino Nano* são ligados nos pinos 2 (RX), 1 (TX), 13 (terra) e 12 (VCC) do HM-10 respectivamente. É preciso atenção quanto a tensão de 5 Volts vinda do pino TX do *Arduino,* pois o módulo HM-10 suporta tensões até 3.3 Volts, sendo suscetível a perda do mesmo quando submetido a tensões maiores. Portanto é necessário um divisor de tensão[8] entre os dois pinos, feitas com três resistores de 1 kilohms em série.

Uma vez ligado o HM-10 com o *Arduino Nano,* a comunicação serial é feita através do IDE *Arduino Software*, que possui a ferramenta *Serial Monitor*. Através dessa interface é possível enviar os comandos AT a qual o módulo responde e obter sua resposta. Os seguintes comandos foram usados para o HM-10 ter a funcionalidade de *beacon*:

1. **AT:** Apenas para verificar se o módulo está ligado, deve receber "OK" como resposta.

2. **AT+ADVI5:** Configura o intervalo de envio de pacotes de dados para 5 (equivalente a 546.25 milisegundos).

3. **AT+ADTY3:** Faz com que o módulo não possa ser conectado por outros dispositivos *Bluetooth*. Como *beacon,* não é necessário se fazer essa conexão, pois é preciso apenas que os outros dispositivos consigam identificá-lo e estejam "cientes" da sua presença.

4. **AT+NAMEBEACON:** Atribui o nome BEACON para o módulo. Obs: nome arbitrário.

5. **AT+IBEA1:** Habilita o modo *iBeacon*.

[8] Divisor de tensão, é uma técnica de projeto utilizada para criar uma tensão elétrica de saída que seja proporcional à uma tensão de entrada.

6. **AT+DELO2:** Modo apenas de transmissão do *beacon*, usado para economizar energia.

7. **AT+PWRM0:** Coloca o *beacon* em *auto-sleep*, não recebendo mais comandos AT. Isso gera uma grande economia de energia pois seu consumo cai de aproximadamente 8 para 0.18 mA apenas.

8. **AT+RESET:** Reinicializa o módulo para garantir que as alterações sejam executadas.

Com esses poucos comandos o módulo HM-10 já está pronto para ser utilizado como *beacon*. Como configurado, a cada aproximadamente 500 milisegundos um sinal contendo pacote de dados é enviado para que dispositivos possam "enxergar" sua presença.

4.3. *Raspberry Pi* como central de processamento

Como o modelo *Raspberry Pi 1 Model B* utilizado não possui *Bluetooth* integrado em seu circuito, foi preciso ligá-lo ao módulo HM-10. Semelhante ao procedimento com o *Arduino*, é necessário ligar os pinos 1 (+3.3V), 6 (terra), 8 (TX) e 10 (RX) das *Raspberry Pi* nos pinos 12 (VCC), 13 (terra), 2 (RX), 1 (TX), e do HM-10 respectivamente.

Os algoritmos criados para serem executados na *Raspberry Pi* utilizam a linguagem C, juntamente com a biblioteca *wiringSerial*, que permite fazer a comunicação serial com o módulo HM-10. Um ponto de atenção é que esse módulo utilizado atua como um *scanner* de *beacons*, portanto é configurado de maneira distinta:

1. **AT+ROLE1:** Faz o módulo atuar como Mestre central.

2. **AT+IMME1:** Quando o módulo é ligado, responde apenas aos comandos AT, e não faz mais nada até **AT + START** for recebido.

Esta configuração permite a utilização do comando **AT+DISI?**, na qual o HM-10 realiza um *scan* no ambiente, recebendo um pacote de dados de cada *beacon*, contendo sete parâmetros:

1. **ID de Fábrica:** campo com 8 caracteres alfanuméricos. Não utilizado no trabalho.
2. **UUID:** Campo com 32 caracteres alfanuméricos. Não utilizado no trabalho.
3. **Major:** Campo com 4 caracteres alfanuméricos. Não utilizado no trabalho.
4. **Minor:** Campo com 4 caracteres alfanuméricos. Não utilizado no trabalho.
5. **TX Power:** Campo com 2 caracteres alfanuméricos (valor em hexadecimal). Necessário para medir a distância entre o *beacon* e *Raspberry Pi*.
6. **Endereço MAC:** Campo com 12 caracteres alfanuméricos. Endereço único de cada módulo, usado para identificação.
7. **RSSI:** Campo com 4 caracteres alfanuméricos. Intensidade do sinal do *beacon*. Usado como parâmetro para medição de distância.

O padrão da string da resposta é: **OK+DISI | OK+DISC: ID de Fábrica: UUID: Major | Minor | TX Power: Endereço MAC: RSSI | OK+DISCE**

Com o módulo HM-10 configurado devidamente, foi desenvolvido as funções para que a *Raspberry Pi* se tornar o sistema central de processamento dos dados proveniente dos *beacons*, descritos a seguir.

4.3.1. Função iniciaSerial

int iniciaSerial (char *msg)

A função iniciaSerial estabelece comunicação com a porta serial com *baud rate* de 9600 e envia o comando "AT+DISI?" para o módulo HM-10. A resposta obtida é

armazenada em forma de *string* no *array* **msg** passado por parâmetro. Caso ocorra erro na comunicação ou no recebimento dos dados é retornado -1, caso contrário retorna 1.

4.3.2. Função contaBeacon

int contaBeacon (char *msg)

Faz a contagem do número de que a vezes a *string* "OK+DISC:" aparece no *array* **msg** passado por parâmetro. O motivo é que a resposta obtida do *scan* feita pelo HM-10 inicia com essa sequência de caracteres para cada *beacon* descoberto. O retorno é o valor obtido da contagem, equivalente ao número de *beacons* achados.

4.3.3. Função calculaDist

double calculaDist (int RSSI, int TxPower)

Calcula a distância entre o *beacon* e *Raspberry Pi* baseado nos parâmetros da função, sendo o **RSSI** o nível do sinal e **TxPower** o valor de calibragem inerente a cada módulo do *beacon*. O valor retornado é a distância calculada.

A fórmula utilizada abaixo, desenvolvida e disponibilizada pela empresa *Radius Network* em sua página para desenvolvedores (YOUNG, 2014) se baseia na regressão linear de uma tabela que contém diversos dados de distância / RSSI, que é específico para cada dispositivo.

$$\text{Distância} = A * (\text{RSSI} / \text{TxPower})^{B} + C$$

Onde A, B e C são constantes, com base na melhor curva de valores das diversas medições de vários níveis de intensidade de sinal. Para uma melhor precisão deve ser feito essas medições para cada módulo HM-10, em diversas distâncias diferentes. Por questões técnicas não foi possível fazer as medições necessárias nos módulos HM-10 utilizados, portanto foi usado os valores de medições feitas para um *smartphone Nexus 4*, publicados

pela própria empresa em seu site de desenvolvimento (YOUNG, 2014) e há maiores detalhes em sua página do *Github* (RADIUS NETWORKS, 2015), onde inclusive há informações detalhadas da fórmula descrita, além de informações de como realizar as medições e calcular as constantes usadas.

4.3.4. Função distanciaRecalc

double distanciaRecalc (double *vet)

Esta função recebe como parâmetro um vetor de valores tipo *double* com dez posições, que então é ordenado por valores de forma crescente. Então são descartados dois valores extremos inferiores e superiores para realizar a média aritmética dos valores restantes, que é retornado para o usuário.

4.3.5. Função formataString

void formataString (char *msg, int numBeacons)

Recebe como parâmetros o *array* com a *string* resposta do *scan* realizado e o número de *beacons* descobertos. Matrizes e vetores temporários são alocadas de acordo com o valor recebido por **numBeacons**, e são usadas para guardar o endereço MAC, intensidade RSSI e valor Tx Power de cada *beacon*, lidos e copiados da *array* **msg**. Em sequência é chamado a função **calculaDist**, passando os parâmetros RSSI e Tx Power, na qual o valor de distância retornado é guardado em um vetor global, visível por todas as funções, onde existe um vetor independente para cada endereço MAC. Quando algum vetor está com dez valores de distância armazenados, a função **baseDados** é chamada para fazer a inserção dos dados na base de dados.

4.3.6. Função baseDados

void baseDados ()

Para o projeto foi criado um servidor local MySQL no *laptop* utilizado durante o desenvolvimento. Essa função faz uso da biblioteca **mysql.h** para conectar ao banco de dados chamado **beacon**, presente no servidor MySQL, onde são inseridos os dados referentes aos *beacons* descobertos, na tabela **info_beacon** (criada para o projeto), com os campos Localização, Endereço MAC e Distância. É verificado quais vetores globais estão cheios (ou seja, com 10 valores de distância armazenados) e feito a inserção dos dados referentes a esses vetores na tabela. Para o campo Distância é utilizado a função **distanciaRecalc**, que é passado como parâmetro o vetor global. Ao chegar no final da função, é desconectado do servidor MySQL. Um detalhe que requer atenção é de que cada *Raspberry Pi* é responsável pelo *scan* de apenas um ambiente hospitalar, portanto quando é feito a inserção do campo Localização, é utilizado um identificador único para cada *Raspberry Pi*, sendo que o utilizado no projeto utiliza o identificador "Sala1" para esse campo. Portanto cada nova inserção na base de dados feita por esse dispositivo específico, o campo Localização terá como registro "Sala1". Para as demais *Raspberrys* devem ser escolhidos também um identificador único para cada.

4.4. Interface Visual Web

A partir da análise direta do banco de dados é possível extrair a informação dos dados contido na tabela **info_beacon**, mas que demanda certo tempo e não é intuitivo para o usuário. Com a finalidade de facilitar e tornar muito mais fácil a visualização da localização dos equipamentos hospitalares, é disponibilizado uma interface visual através de página web, com a planta do hospital na qual é exibida um indicador para cada equipamento e sua localidade e uma tabela com as seguintes informações:

- **Equipamento:** Exibe o nome do equipamento vinculado a um *beacon*.
- **Localização Atual:** Qual ambiente se encontra o equipamento.

- **Distância:** Mostra a distância calculada pela *Raspberry Pi*.
- **Erro:** Desvio padrão considerado para a distância calculada.
- **Última Localização:** Último ambiente que o equipamento foi detectado antes da atual.
- **Data e Hora:** Data e horário da última detecção presenciada pela *Raspberry Pi*.

A interface da página web é exibido na figura 6.

Figure 6: Captura da Interface Visual Web.

Para a construção da página web foi utilizado a linguagem PHP, que significa *Hypertext Preprocessor*, amplamente usada para desenvolvimento web, mesclado dentro de um código HTML (do inglês *HyperText Markup*), linguagem de marcação de texto também usado para desenvolvimento de páginas web. Como o PHP é uma linguagem interpretada

pelo lado do servidor, foi instalado o servidor Apache versão 2.4 para realizar essa interpretação e poder exibir interface visual do projeto corretamente. Uma das vantagens do PHP é ter uma biblioteca nativa para acessar e manipular dados de um banco de dados MySQL, portando sua integração com o trabalho é muito facilitada.

O código escrito faz importação de uma parte da planta do Hospital Escola de São Carlos em formato de imagem PNG, para fins de demonstração e visualização, uma vez que a planta completa é demasiadamente grande. Então é feita conexão com a base de dados **beacon** presente no servidor MySQL e realizado uma *query* para trazer os dados da tabela **info_beacon**, a uma taxa de atualização de 10 segundos. Esta *query* traz apenas as últimas inserções feitas, com a localização de todos os *beacons* descobertos. Como múltiplas *Raspberry Pi*`s podem ter realizado o *scan* de um mesmo *beacon*, o critério utilizado para determinar qual a localização correta é qual dos registros possuem a menor distância calculada. Assim, por exemplo, caso as *Raspberry Pi* A e B, presente em ambientes distintos, descubram um mesmo *beacon*, a localização correta é dada pela que possui menor distância resultante.

A **Distância** é o valor de distância calculado pela *Raspberry Pi* responsável pela inserção dos dados coletados.

Para a **Localização Atual** é pressuposto que cada *Raspberry Pi* esteja posicionado fixamente em um ambiente do hospital, e a partir do identificador único registrado no campo Localização da tabela, é exibido a posição do equipamento. A cada *query* realizada é atualizada essa informação, e caso a **Localização Atual** do equipamento mude, a localidade antiga é mostrada em **Última Localização**.

O **Equipamento** parte da premissa de que cada *beacon* é atrelado a um único equipamento hospitalar. Como o endereço MAC é único para cada módulo HM-10, pode-se atribuir um equipamento para cada endereço. Para o trabalho foram utilizados apenas dois *beacons*, portanto simulado dois equipamentos hospitalares arbitrários, o Desfibrilador e Eletrocardiógrafo.

Erro é resultado dos testes feitos com os *beacons*. Foram coletadas amostras e realizado o cálculo de desvio padrão para diferentes distâncias. Maiores detalhes estão no tópico de Resultados.

Data e Hora é apenas a data e horário em que foram inseridos os dados no servidor MySQL.

4.5. Resultados

Para realizar os testes foram utilizados um módulo HM-10 configurado como *beacon* e uma *Raspberry Pi 1 Model B* em comunicação serial com um módulo HM-10. Para simular o ambiente hospitalar, foi usado uma sala de tamanho aproximado de 7 por 7 metros, onde a *Raspberry Pi* foi fixada em um ponto da sala e o *beacon* posicionado em diferentes distâncias. Com o algoritmo na *Raspberry Pi* rodando em loop, foi sendo coletado os dados do *beacon* junto com sua distância relativa, e armazenando no banco de dados criado. Foram ao todo coletados 100 amostras para cada posição do *beacon*, com distâncias de 1, 2, 4 e 6 metros. Ao final da coleta, foi realizado a média aritmética de todas as amostras para cada distância, o desvio padrão e um gráfico com a distribuição normal de cada teste, cujo os resultados são mostrados a seguir.

4.5.1. Distância de 1 metro

Os resultados obtidos para o *beacon* posicionado a 1 metro da *Raspberry Pi* mostram que há uma boa precisão da distância, com desvio padrão aceitável.

- **Valor de distância mínimo:** 0.49 metros
- **Valor de distância máximo:** 2.52 metros
- **Média aritmética:** 1.01 metros

- **Desvio padrão:** +/- 0.28 metros

No Gráfico 1 mostra a distribuição normal das amostras de distância 1 metro, e pode-se aferir que a probabilidade da distância calculada estar próximo da distância real de 1 metro é alta, enquanto os extremos apresentam baixa probabilidade de ocorrência.

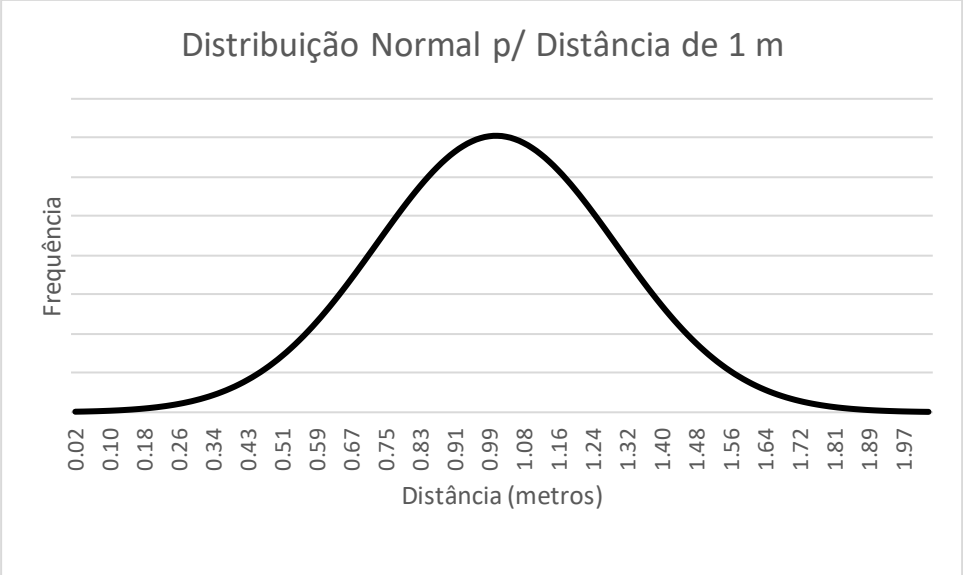

Gráfico 1: Distribuição normal das amostras com 1 metros de distância.

4.5.2. Distância de 2 metros

Com aumento da distância, nota-se um aumento do desvio padrão, ocasionando em uma maior variância dos valores encontrados. Em média a amostra se encontra próximo da distância real de 2 metros, com um erro de 0.24 metros, que ainda é aceitável para uma localização precisa.

- **Valor de distância mínimo:** 0.70 metros
- **Valor de distância máximo:** 4.25 metros
- **Média aritmética:** 2.24 metros
- **Desvio padrão:** +/- 0.74 metros

No gráfico 2 é exibido a distribuição normal das amostras com distância de 2 metros. As distribuições de valores apresentam uma menor concentração no centro, indicando menos precisão comparado com as amostras de distância 1 metro.

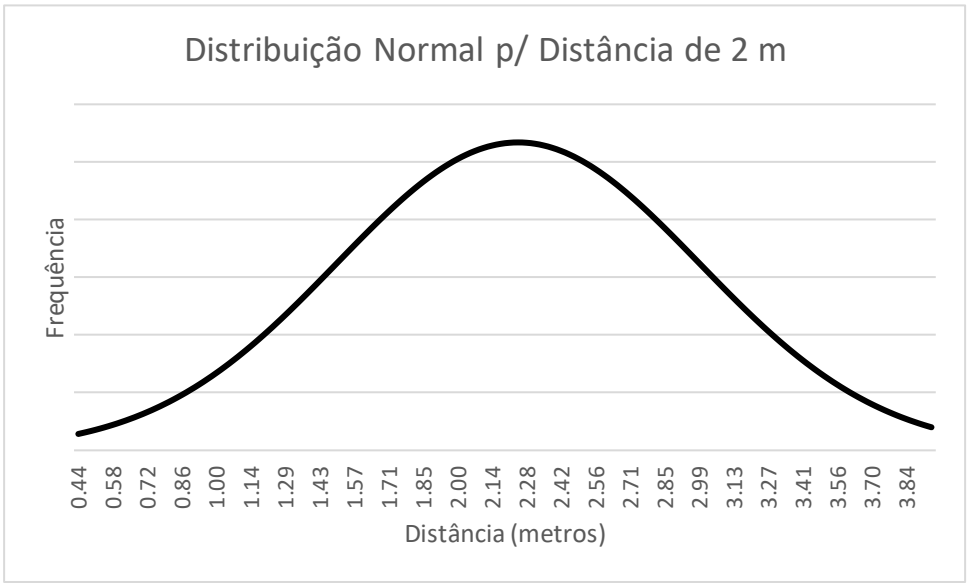

Gráfico 2: Distribuição normal das amostras com 2 metros de distância.

4.5.3. Distância de 4 metros

Não houve grandes diferenças com os resultados das amostras de distância 2 metros. A média de valores continuam estáveis, assim como o desvio padrão, com nível de precisão ainda aceitáveis.

- **Valor de distância mínimo:** 2.69 metros
- **Valor de distância máximo:** 6.84 metros
- **Média aritmética:** 4.03 metros
- **Desvio padrão:** +/- 0.72 metros

No Gráfico 3 é exibida a distribuição normal das amostras com distância de 4 metros, com distribuição de valores comparáveis com as amostras de distância 2 metros.

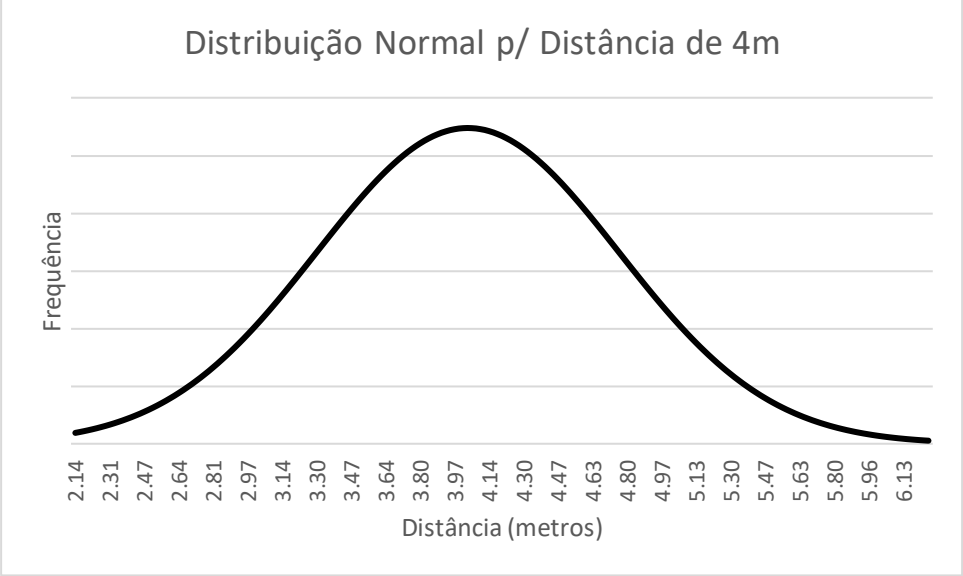

Gráfico 3: Distribuição normal das amostras com 4 metros de distância.

4.5.4. Distância de 6 metros

Essas amostram registram um grande aumento do desvio padrão, assim como os valores extremos são bem distantes da distância real. Como consequência há uma enorme variação das distâncias calculadas, diminuindo a confiabilidade das informações.

- **Valor de distância mínimo:** 2.93 metros
- **Valor de distância máximo:** 14.49 metros
- **Média aritmética:** 5.84 metros
- **Desvio padrão:** +/- 2.48 metros

No Gráfico 4 é exibida a distribuição normal das amostras com distância de 6 metros, com a distribuição de valores bem desigual, o que indica instabilidade das distâncias.

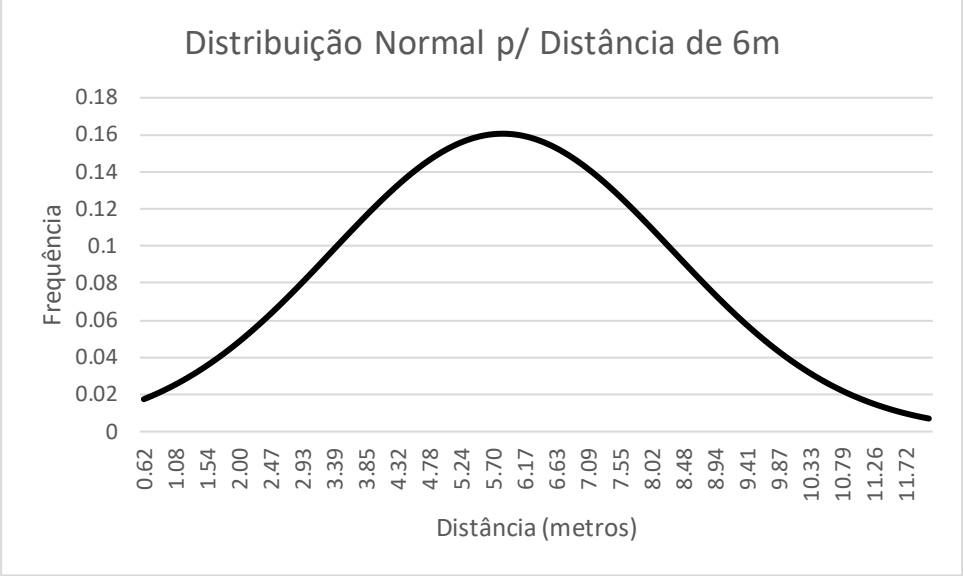

Gráfico 4: Distribuição normal das amostras com 6 metros de distância.

4.5.5. Distância Superiores a 6 metros

Para distâncias entre o *beacon* e a *Raspberry Pi* superiores a 6 metros não foi possível colher amostras suficientes para análise mais aprofundada. Isso se deve a limitação física do espaço utilizado para testes, além da dificuldade de acesso à internet para poder enviar os registros para o banco de dados criado e falta de tomadas elétricas para alimentar os dispositivos, nos possíveis ambientes de maiores dimensões. Contudo ainda foram feitas medições com posicionamento do *beacon* a mais de 6 metros, ainda que com uma menor amostragem, onde foi verificado uma maior variação das distâncias calculadas. Conforme o distanciamento entre os dois dispositivos é aumentado, percebe-se uma piora da qualidade dos dados gerados relativo a distância. A partir de aproximadamente 12 metros se torna inviável a utilização desses dados, pois a discrepância entre a distância real e a calculada é muito grande. Outro fator observado que afeta de maneira brusca os resultados são obstáculos, como paredes, móveis e objetos que interferem no sinal enviado pelo *beacon*, possivelmente criando ruído e interferindo no resultado do cálculo final.

4.5.6. Teste da Interface Visual Web

Com as amostras devidamente coletadas e armazenadas no banco de dados, foram feitas execuções da interface visual web, que mostrou com sucesso as informações do *beacon* utilizado para o teste, exibindo corretamente, o equipamento associado ao endereço MAC, tanto sua localização mais recente quanto a anterior, o erro (desvio padrão) relativo a distância e a data e hora do registro. As marcações na imagem da planta do hospital, com indicação da posição, também foram exibidas de maneira satisfatória.

4.6. Dificuldade e Limitações

Uma das maiores dificuldades foi a escassez de dispositivos disponibilizados para a realização do projeto. Ao todo foi utilizado apenas uma *Raspberry Pi 1 Model B*, dois módulos HM-10 e um *Arduino Nano*. Isso impacta diretamente na realização de testes, onde não foi possível ter uma simulação mais real da que é encontrada no hospital diariamente.

Há limitação também no algoritmo desenvolvido para cálculo de distância, pois usam como base o nível do sinal RSSI e o valor de referência Tx Power. As observações feitas mostram que o RSSI é extremamente instável, suscetível a muitas variações devido a presença de objetos, móveis e pessoas no ambiente, afetando o resultado final calculado. Como mostra os resultados, essa variância dos níveis de sinal aumenta quanto mais afastado se encontra o *beacon* da *Raspberry Pi*. Portanto a confiabilidade dos dados gerados é comprometida para distâncias maiores.

CAPÍTULO 5: CONCLUSÃO

5.1. Contribuições

O sistema desenvolvido faz contribuição para hospitais e seus profissionais, possibilitando um controle maior do inventário hospitalar e possivelmente facilitando a tomada de decisão relativo ao atendimento e alocação dos pacientes, de médicos e enfermeiros para salas adequadas. Espera-se que esse trabalho traga um maior uso de tecnologia nos hospitais, que pode ser benéfico tanto para equipe médica e demais profissionais, quanto para os pacientes.

Toda a pesquisa e desenvolvimento trazido por esse projeto pode ser usado como base para projetos, estudos e aplicações futuras. Há também a possibilidade de expandir o uso desse sistema para outros propósitos, devido a gigante gama de opções de uso ainda não exploradas oferecidas pelos *beacons*, ou até mesmo continuar com a expansão e melhoria desse trabalho.

5.2. Relacionamento entre o Curso e o Projeto

O curso de Engenharia da Computação traz uma grade curricular muito ampla, que possibilita ao aluno ter conhecimento em diversas áreas ligadas a computação. Em particular para o desenvolvimento desse projeto, as disciplinas relacionas a programação, lógica, controle e automação proveram uma boa base para que fosse possível a realização do trabalho e superar desafios, dificuldades encontradas durante todo o percurso.

5.3. Considerações sobre o Curso de Graduação

O curso de Engenharia da Computação oferecida pela Universidade de São Paulo, campus São Carlos, traz a seus alunos toda a competência exigida para atuar em diferentes áreas da engenharia ou computação, graças a sua vasta grade de disciplinas. Aqueles que desejam atuar tanto em área acadêmica quanto profissional, recebem uma base sólida para que exerçam um trabalho excepcional na carreira escolhida. Isso é evidenciado no reconhecimento dos vários prêmios e reconhecimentos conquistados dentro e fora do Brasil.

Porém, há uma crítica a ser feita quanto ao aprofundamento proveniente das disciplinas. Apesar de ser um curso vasto em questão de áreas abordadas, falta a possibilidade de poder se especializar e aprofundar os conhecimentos em certos tópicos, dentro da estrutura oferecida pela faculdade, onde muitas vezes é passado uma visão superficial da matéria dada. A diminuição da carga horária de disciplinas obrigatórias e ofertar um maior leque de disciplinas optativas ajudaria a melhorar esse cenário.

5.4. Trabalhos Futuros

Para trabalhos futuros pode se listar as seguintes sugestões:

- Estudar e implementar um novo algoritmo para cálculo da distância relativa entre os *beacon* e *Raspberry Pi*, visto que a fórmula utilizada no trabalho está propensa a muitas variações de valores dependendo do ambiente que se encontra.

- Incluir novas funcionalidades para a interface visual web, como filtro para equipamentos ou um grupo deles, um melhor indicador visual da localização na planta do hospital.

- Melhorar o algoritmo presente na *Raspberry Pi*, com melhor suporte a identificação dos *beacons* e um critério de seleção dos valores calculados de distância que indiquem realmente o melhor valor.

REFERÊNCIAS

ALECRIM, E. Tecnologia Bluetooth: o que é e como funciona? **Infowester**, 2013. Disponivel em: <http://www.infowester.com/bluetooth.php>. Acesso em: 10 Outubro 2016.

APPLE INC. GETTING Started with iBeacon. **Apple Developers**, 2014. Disponivel em: <https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf>. Acesso em: 08 Outubro 2016.

ARDUINO Nano. **Arduino**, 2014. Disponivel em: <https://www.arduino.cc/en/Main/ArduinoBoardNano>. Acesso em: 14 Outubro 2016.

BLUETOOTH low energy. **Wikipedia**, 2016. Disponivel em: <https://en.wikipedia.org/wiki/Bluetooth_low_energy>. Acesso em: 11 Outubro 2016.

IBEACON. **Wikipedia**, 2016. Disponivel em: <https://en.wikipedia.org/wiki/IBeacon>. Acesso em: 08 Outubro 2016.

KARYDIS, T. Bluetooth Interfacing with HM-10. **FabCentral**, 2015. Disponivel em: <http://fab.cba.mit.edu/classes/863.15/doc/tutorials/programming/bluetooth.html>. Acesso em: 14 Outubro 2016.

KORTH, H. F.; SILBERSCHATZ, A. **Sistemas de Banco de Dados**. edição revisada. ed. [S.l.]: Makron Books, v. 2a, 1994.

MYSQL. **Wikipedia**, 2016. Disponivel em: <https://pt.wikipedia.org/wiki/MySQL>. Acesso em: 20 Outubro 2016.

NETCRAFT. May 2010 Web Server Survey. **Netcraft**, 2010. Disponivel em: <https://news.netcraft.com/archives/2010/05/14/may_2010_web_server_survey.html>. Acesso em: 14 Outubro 2016.

RADIUS NETWORKS. Android Beacon Library. **Radius Networks Github Page**, 2015. Disponivel em: <https://altbeacon.github.io/android-beacon-library/distance-calculations.html>. Acesso em: 12 Setembro 2016.

RASPBERRY Pi. **Wikipedia**, 2016. Disponivel em: <https://en.wikipedia.org/wiki/Raspberry_Pi>. Acesso em: 13 Outubro 2016.

YOUNG, D. G. Fundamentals of Beacon Ranging. **Radius Networks Developer Blog**, 2014. Disponivel em: <http://developer.radiusnetworks.com/2014/12/04/fundamentals-of-beacon-ranging.html>. Acesso em: 10 Setembro 2016.

APÊNDICE A – Instalação do Servidor Apache 2.4

Para instalação do Apache 2.4 no ambiente *Windows 10* foi utilizado os binários disponibilizados pelo *Apache Lounge,* pois os binários não são oferecidos para esse sistema operacional de forma oficial pelo *Apache*.

Foram seguidos os seguintes passos:

1. ***Download* do binário:** Acessar https://www.apachelounge.com/download/ e escolher a versão de acordo com a versão do ambiente *Windows* que será rodado (32 ou 64 bits).

2. **Descompactar:** O arquivo está em compactado em formato .zip, descompacte em diretório de preferência. Obs: certificar de estar a instalado a última versão do C++ Redistributable Visual Studio 2015 (disponível em https://www.microsoft.com/en-us/download/details.aspx?id=53840)

3. Abrir a janela de comandos (Windows + R, digite cmd e tecle ENTER), vá para o diretório bin do diretório descompactado (Ex. C:\Apache24\bin) e execute httpd.exe. Caso apareça o *warning* "Could not reliably determine the server's fully qualified domain name", pode se ignorar a mensagem.

4. É possível aparecer a mensagem do *Firewall* do *Windows* perguntando sobre as permições do Apache, permita o acesso.

5. Abra *browser* de sua escolha e vá para o endereço **http://localhost**, se uma página com a mensagem "It`s Works!" aparecer, o Apache foi instalado com sucesso.

APÊNDICE B – Instalação do Servidor MySQL

Para a instalação do banco de dados MySQL no sistema operacional *Windows* 10 foram seguidos os seguintes passos:

1. Faça o *download* do instalador "mysql-installer-web-community-xxxxx.msi" no endereço: http://dev.mysql.com/downloads/installer/

2. Siga os passos do instalador e caso o uso seja para desenvolvimento então a configuração "Developer default" seja a melhor opção.

3. Será instalado também o *MySQL Workbench* que é a interface visual nativo para o usuário.

4. Após a instalação será perguntado questões sobre configuração, sendo recomendado selecionar a opção "Open firewall port for network access" caso o banco de dados vá ser acessado por outras máquinas.

5. Defina sua senha de usuário *root*, que irá ser seu usuário padrão para desenvolvimento.

6. O servidor MySQL está instalado com sucesso.