UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Programa de Pós-Graduação em Engenharia Biomédica
PPGEB

FRANCISCO MULLER MACHADO

PROPOSTA DE PROTOCOLO DE TELEMONITORAMENTO SOB DEMANDA DE SINAIS BIOMÉDICOS USANDO INTERNET DAS COISAS, COMPUTAÇÃO MÓVEL E ARMAZENAMENTO EM NUVEM

DISSERTAÇÃO

CURITIBA
2016

FRANCISCO MULLER MACHADO

PROPOSTA DE PROTOCOLO DE TELEMONITORAMENTO SOB DEMANDA DE SINAIS BIOMÉDICOS USANDO INTERNET DAS COISAS, COMPUTAÇÃO MÓVEL E ARMAZENAMENTO EM NUVEM

Dissertação apresentada como requisito parcial à obtenção do título de Mestre em Engenharia Biomédica, do Programa de Pós-Graduação em Engenharia Biomédica, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Miguel Antonio Sovierzoski

CURITIBA

2016

Dados Internacionais de Catalogação na Publicação

M149p 2016

Machado, Francisco Muller
Proposta de protocolo para telemonitoramento sob demanda de sinais biomédicos usando internet das coisas, computação móvel e armazenamento em nuvem / Francisco Muller Machado.-- 2016.
132 f.: il.; 30 cm

Texto em português com resumo em inglês.
Dissertação (Mestrado) - Universidade Tecnológica Federal do Paraná. Programa de Pós-Graduação em Engenharia Biomédica, Curitiba, 2016.
Bibliografia: f. 125-132.

1. Redes de computação - Protocolos. 2. Monitorização fisiológica. 3. Processamento de sinais - Técnicas digitais. 4. Computação móvel. 5. Computação em nuvem. 6. Eletrocardiografia. 7. Sistemas de comunicação sem fio. 8. Simulação (Computadores). 9. Engenharia biomédica - Dissertações. I. Sovierzoski, Miguel Antonio, orient. II. Universidade Tecnológica Federal do Paraná. Programa de Pós-graduação em Engenharia Biomédica. III. Título.

CDD: Ed. 22 -- 610.28

Biblioteca Central da UTFPR, Câmpus Curitiba

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Campus Curitiba

Programa de Pós-Graduação em Engenharia Biomédica

Título da Dissertação Nº 059

"Proposta de protocolo para telemonitoramento sob demanda de sinais biomédicos usando internet das coisas, computação móvel e armazenamento em nuvem".

por

Francisco Müller Machado

ÁREA DE CONCENTRAÇÃO: Engenharia Biomédica

LINHA DE PESQUISA: Instrumentação Biomédica.

Esta dissertação foi apresentada como requisito parcial à obtenção do grau de **MESTRE EM CIÊNCIAS(M.Sc.)** – Área de Concentração: Engenharia Biomédica, pelo **Programa de Pós-Graduação em Engenharia Biomédica (PPGEB),** – da Universidade Tecnológica Federal do Paraná (**UTFPR**), *Campus* Curitiba, às **14h00min** do dia **28 de abril de 2016.** O trabalho foi aprovado pela Banca Examinadora, composta pelos professores:

Prof. Miguel Antonio Sovierzoski, Dr.
(Presidente – UTFPR)

Prof. Rui Francisco Martins Marçal, Dr.
(PUCPR)

Prof. Joaquim Miguel Maia, Dr.
(UTFPR)

Visto da coordenação:

Profª. Leandra Ulbricht.,Drª.
(Coordenadora do PPGEB)

AGRADECIMENTOS

Este trabalho é dedicado a todos que acreditaram na idéia.
Aos muitos mestres e doutores que me ajudaram pelo caminho
Aos muitos que já estão outro plano, mas estavam ao meu lado, apoiando, torcendo.
O sucesso é uma conquista individual,
é uma conquista da família e amigos, que direta ou indiretamente apoiaram
é uma conquista dos meus mentores, que de alguma forma que não
compreendemos, ajudaram a chegar nesse momento
Agradeço especialmente à Isis Koehler,
aos colegas de trabalho, ao professor Eduardo Bertogna pela grande ajuda.
Muito especialmente, agradeço ao incansável professor Miguel Sovierzoski pela disponibilidade, paciência e dedicação.

Obrigado!

RESUMO

MACHADO, Francisco Müller. **PROPOSTA DE PROTOCOLO PARA TELEMONITORAMENTO SOB DEMANDA DE SINAIS BIOMÉDICOS USANDO INTERNET DAS COISAS, COMPUTAÇÃO MÓVEL E ARMAZENAMENTO EM NUVEM**. 2016. 132 páginas. Dissertação (Mestrado em Engenharia Biomédica) - Universidade Tecnológica Federal do Paraná. Curitiba, 2016.

Este trabalho apresenta a proposta de um protocolo de comunicação de dados integrando tecnologias de internet das coisas, computação móvel e armazenamento em nuvem, aplicado ao telemonitoramento sob demanda de sinais fisiológicos. O objetivo do trabalho foi adquirir, transmitir, armazenar, receber e permitir a visualização sob demanda, mantendo a integridade da representação do sinal biomédico para a análise médica, sem alterar a mobilidade do paciente em monitoramento. O trabalho utiliza um canal de ECG, para adquirir o sinal de eletrocardiografia do paciente, que é conectado via Bluetooth com um dispositivo de computação móvel. O dispositivo de computação móvel, com acesso a internet sem fio, envia o sinal fisiológico através do protocolo proposto para uma base segura de armazenamento de informações em nuvem, podendo ser acessada sob demanda por um especialista para realizar a avaliação médica. O telemonitoramento sob demanda permite ao especialista: visualizar e analisar dados recentes, como se fosse um monitoramento remoto em tempo real do paciente; e visualizar e analisar dados anteriores que estão armazenados na nuvem, semelhante à análise de um exame realizado previamente. O protocolo proposto possui características de segurança da informação, mantendo a integridade da representação no tempo do sinal fisiológico, ainda que dados sejam corrompidos. O trabalho envolveu o uso de tecnologias recentes aplicadas às necessidades de registros de sinais fisiológicos. Estas tecnologias estão em evolução, os padrões não estão consolidados, ocorrendo alterações à medida que novas necessidades vão sendo apresentadas, assim como novas soluções venham a ser desenvolvidas. Como dispositivo de computação móvel foi utilizado neste trabalho um tablet, podendo ser utilizado um smartphone. O protocolo para este trabalho foi utilizado com o sinal de um canal de eletrocardiografia, podendo ser modificado para atender outras necessidades, incluindo outros sinais fisiológicos. Neste trabalho foi utilizado como sinal fisiológico o sinal de um simulador de ECG.

Palavras-chave: Telemonitoramento em saúde; Sistema móvel de informação em saúde; Internet das coisas para saúde; Armazenamento em nuvem de sinais fisiológicos; Armazenamento em nuvem de sinais biomédicos.

ABSTRACT

MACHADO, Francisco Müller. **PROTOCOL PROPOSAL FOR ON-DEMAND REMOTE MONITORING OF BIOMEDICAL SIGNALS USING INTERNET OF THINGS, MOBILE COMPUTING AND CLOUD STORAGE**. 2016. 132 pages. Dissertation (Master in Biomedical Engineering) Federal University of Technology - Paraná. Curitiba, 2016.

This work shows a proposal for a data communication protocol integrating internet of things technologies, mobile computing and cloud storage, applied to on-demand remote monitoring physiological signals. The aim of this work was to acquire, transmit, store, receive and allow the on-demand visualization, keeping the integrity of the biomedical signals representation for medical analysis, without changing the mobility of the patient being monitored. The work uses an ECG channel to acquire the patient's electrocardiography signal, which is linked via Bluetooth to a mobile computing device. The mobile computing device, with wireless internet access, sends the physiological signal through the proposed protocol to a safe base of cloud information storage, which can be accessed on demand by a specialist to do the medical evaluation. On-demand remote monitoring allows the specialist to visualize and analyze recent data, as if it were real time remote monitoring of the patient, and visualize and analyze previous data that are stored in the cloud, similar to an analysis of a previously made exam. The proposed protocol has information security features, keeping the integrity of the time representation of the physiological signal, even if the data are corrupted. The work involved the use of new technologies applied to the necessity of physiological signals record. These technologies are evolving. The patterns are not consolidated, and changes are made as new necessities are presented and new solutions developed. For this work, it was used a tablet as a computing mobile device. However, a smartphone could also be used. The protocol was used with an electrocardiography channel signal, which can be modified to attend other necessities, including other physiological signals. As a physiological signal, for this work, it was used an ECG simulator signal.

Keywords: Health remote monitoring, Health mobile information system, Health internet of things, Physiological signal cloud storage, Biomedical signal cloud storage.

LISTA DE SIGLAS E ABREVIATURAS

ACL	*Asynchronous Connection Oriented* – conexão assíncrona orientada
ADC	*Analog-to-Digital Converter* – conversor analógico-digital
ADT	*Android Developper Tools* – ferramentas de desenvolvimento android
API	*Application Programming Interface* – interface de programação da aplicação
APK	*Android Package File* – arquivo-pacote android
CI	Circuito Integrado
CPS	*Cyber Physical System* – sistema físico cibernético
DEX	*Dalvik executable* – arquivo executável Dalvik
DVM	*Dalvik Virtual Machine* – máquina virtual Dalvik
ECG	Eletrocardiografia, eletrocardiógrafo, eletrocardiograma
EPC	*Eletronic Product Code* – código eletrônico de produto
FH-CDMA	*Frequency Hopping - Code Division Multiple Access* – salteamento de frequência – acesso múltiplo por divisão de código
FHS	*Frequency Hopping Synchronization* – sincronização por salteamento de frequência
FH/TDD	*Frequency Hopping/Time Division Duplex* – salteamento de frequência/divisão de tempo duplex
FIR	*Finite Impulse Response* – resposta finita ao impulso
GPS	*Global Positioning System* – sistema de posicionamento global
GSM	*Global System for Mobile communication* – sistema global de comunicação móvel
HCI	*Host Controller Interface* – interface de controle
HTTPS	*Hyper Text Transfer Protocol Secure* – protocolo seguro de transferência de hipertexto
IBGE	Instituto Brasileiro de Geografia e Estatística
IEEE	*Institute of Electrical and Electronics Engineers* – instituto dos engenheiros elétricos e eletrônicos
IETF	*Internet Engineering Task Force* – força tarefa de engenharia de internet
IoT	*Internet of Things* – internet das coisas
IP	*Internet Protocol* – protocolo internet
ISM	*Industrial, Scientific, Medical* – industrial, científico, médico

L2CAP	*Logical Link Control and Adaptation Protocol* – protocolo de adaptação e de controle lógico da conexão
LMP	*Link Management Protocol* – protocolo de gerência de conexão
M2M	*Machine to Machine* – máquina para máquina
MCC	*Mobile Cloud Computing* – computação móvel em nuvem
MCU	*Microcontroller* – microcontrolador
MIT	*Massachusetts Institute of Technology* – Instituto de tecnologia de Massachusetts
NAV	Nó Atrioventricular
NSA	Nó Sino Atrial
OBEX	*OBject Exchange* – intercâmbio de objetos
OHA	*Open Handset Alliance* – aliança aberta de aparelho
ONU	Organização das Nações Unidas
OSI	*Open Systems Interconnection* – interconexão de sistemas abertos
PDP	*Provable Data Possession* – verificação de segurança de dados
PPP	Protocolo Ponto a Ponto
QoS	*Quality of Service* – qualidade de serviço
RAM	*Random Access Memory* – memória de acesso aleatório
RF	*Radio Frequency* – rádio frequência
RFCOMM	*Radio Frequency Communication* – comunicação por rádio frequência
RFID	*Radio Frequency Identification* – identificação por rádio frequência
SCO	*Synchronous Connection Oriented* – conexão orientada assíncrona
SDK	*Software Development Kit* – kit de desenvolvimento de software
SDP	*Service Discovery Protocol* – protocolo de serviço de busca
SIG	*Special Interest Group* – grupo de interesse específico
SSL	*Secure Socket Layer* – camada de soquete seguro
TCP/IP	*Transmission Control Protocol/Internet Protocol* – protocolo de controle de transmissão/protocolo de internet
TCS	*Telephony Control protocol Specification* – especificação do protocolo de controle de telefonia
TDMA	*Time Division Multiple Access* – acesso múltiplo por divisão de tempo
UUID	*Universally Unique Identifier* – identificador único universal
USART	*Universal Synchronous Assyncronous Receiver Transmiter* – transmissor receptor universal assíncrono síncrono

LISTA DE FIGURAS

LISTA DE TABELAS

Sumário

1. INTRODUÇÃO

Internet das coisas (*Internet of Things* - IoT) é uma tendência tecnológica que vem aos poucos se fazendo presente nos últimos anos. Ela foi originalmente proposta em 1999 por pesquisadores do *Massachusetts Institute of Technology* e originalmente tinha a idéia de que cada objeto, fosse ele eletrodoméstico, sensor ou aparelho, tivesse um código para o qual se atrelasse um determinado endereço IP fixo, além de outro código especifico para tal objeto. Havia para tanto o problema da limitação de endereços físicos, que com o IPV6 se fizeram resolver. Na época, a ideia do MIT era usar a tecnologia RFID de reconhecimento de objetos através da sua identificação uma vez exposto o objeto às ondas de rádio de alta frequência emitidas por um transmissor. Uma vez exposto, o objeto respondia com seu código RFID a fim de ser identificado no meio onde se encontrava.

Porém, com o tempo também teve-se um grande desenvolvimento dos smartphones, que foram se sofisticando com mais recursos de hardware e software, tornando-os tão poderosos como um computador de mesa. Com tal poder de processamento e com toda uma tecnologia de informação desenvolvida para dispositivos móveis, através do melhoramento de seus sistemas operacionais, entre os quais android, passou a ser possível pensar em uma revolução na maneira como se pudesse usar tais dispositivos móveis, seja na forma de tablets, smartphones ou phablets, no dia-a-dia, para as mais variadas funções e prestações de serviços. Uma verdadeira quebra de paradigma na sociedade informatizada que vivemos se faz presente, uma vez que se tem a informação detalhada dos objetos, pessoas, situações de status dos objetos, que podem ser usadas por programas específicos para prestar inúmeras informações.

Com novas tecnologias de internet, chegou a IoT, um novo paradigma do mundo moderno que combina as tecnologias emergentes para uma nova abordagem condizente com os dias atuais. Computação ubíqua, protocolos de internet, novos sensores, tecnologias novas de comunicação, entre outros, formam um novo sistema onde os mundos reais e virtuais se encontram, sendo o *smart object* ou objetos inteligentes as novas peças deste novo quebra cabeças que fornecem uma nova visão de IoT no mundo (BORGIA, 2014).

Junto com este desenvolvimento vieram os diversos sistemas de informação, dentre os quais os sistemas de informação de saúde, formado por um número crescente de dispositivos móveis para dar apoio aos profissionais da área em seus consultórios e hospitais. Estes sistemas realizam a troca de informações entre os dispositivos envolvidos em saúde, indo de aparelhos smartphones, tablets, chegando aos sensores sejam eles de ritmo cardíaco, de temperatura, de medida de glicose, entre outros. Desta forma passou-se a ser possível a disponibilização das informações também em dispositivos móveis (POLLARD *et al*., 2001) (KJELDSKOV, SKOV, 2004).

Também tem-se desenvolvido, pelas grandes corporações de dados, grandes servidores de dados como grandes repositórios de informação no mundo. Deste modo, os aplicativos nos dispositivos móveis, que podem tanto funcionar de maneira independente como também podem fazer uso de informações obtidas de servidores na rede, ou de seus sensores, sejam sensores biomédicos ou sensores de status como os de posição geográfica, de velocidade ou posicionamento, vêm usando tais *data centers* no mundo (ESPADA *et al*., 2013).

Analisando de outro lado, o mundo apresenta a cada dia uma população cuja expectativa de vida é crescente ao longo dos anos. Com isto, os cuidados com a saúde se tornam mais frequentes e alguns problemas do dia-a-dia podem ser contornados com as novas tecnologias. Sendo possível um monitoramento remoto das condições de sinais biomédicos, torna-se mais fácil o controle da saúde populacional pela comunidade médica através da rede internet.

Este é o objetivo que se tem com o monitoramento da saúde da população de idosos e dos que, de alguma maneira, tenham algum tipo de deficiência, seja de movimentação, de audição, visão, ou de saúde em geral. Os sistemas biomédicos que emergem das novas tecnologias se baseiam na confiabilidade de sensores, na velocidade de processamento local e na confiabilidade das transmissões através dos protocolos da rede mundial (ESPADA *et al*., 2013).

Desta forma, o desenvolvimento de uma rede de sensores centrada nas pessoas, individualmente, se torna muito apropriada além de factível nos dias atuais, para o controle de saúde por dispositivos móveis através de seus aplicativos, em grandes áreas urbanas. Por isto, a tendência é ver o triunfo da internet das coisas (IoT) nas cidades inteligentes do futuro (ITU, 2015).

Os sistemas eletrônicos biomédicos de saúde consistem basicamente de uma unidade portátil, que possa ser vestida, ou também muito comumente chamada pelo termo em inglês *wearable*, e de consumo de energia baixo, que fique monitorando continuamente algum parâmetro biomédico e enviando ao sistema principal através de ondas de rádio, através de algum tipo de rede de comunicação apropriada.

A idéia proposta neste trabalho consiste no desenvolvimento de uma aplicação que faça uso dos conceitos de internet das coisas, no sentido de identificar tanto o usuário quanto o sinal biomédico ao qual a ele esteja associado.

Através do monitoramento do sinal de Eletrocardiografia (ECG) advindo de um sistema microcontrolado, este sinal é monitorado em tempo real para que não hajam perdas de pacotes de informação durante a transmissão e, uma vez recebidos os dados, eles sejam disponibilizados na rede para análise, tanto de maneira remota quanto de maneira a *posteriori*, afinal os dados pressupõe-se serem gravados em um servidor também chamado de nuvem.

1.1. CONTEXTO

Os dias atuais estão mostrando uma grande notoriedade pelos desenvolvimentos tecnológicos que influencia e altera a vida de todos.

De um lado há o problema de locomoção em grandes centros urbanos, sendo a mobilidade um difícil obstáculo a transpor seja ela para pessoas comuns ou mesmo para meios de locomoção dedicados como no caso de ambulâncias. Com a falta de recursos públicos para a melhoria de transporte particular ou coletivo em longas distâncias a determinadas horas do dia, uma simples ida ao médico pode se tornar um problema, em caso de urgência.

Analisando de outro lado, pessoas mais velhas ou com algum tipo de deficiência poderiam ter disponíveis alguma forma de alarme caso algo emergencial ocorresse, de modo a fornecer tempo de um socorro apropriado. Além disso, deve-se ter em mente que normalmente pessoas mais velhas tem maiores problemas no aprendizado com aplicativos móveis, por mais que eles possam ser simplificados, sobretudo devido a apresentarem algum tipo de deficiência, seja auditiva, táctil, motora ou de visão.

Então, da mesma forma que se desenvolveu a teleconferência para reuniões virtuais entre profissionais das várias áreas, a IoT vem surgindo para integrar todos os entes que fazem parte de uma grande rede emergente em grandes *datacenters* espalhados na rede, que podem, dentre várias funções numa sociedade, também ajudar nos cuidados com a saúde da população, através de um simples monitoramento.

A ideia de usar smartphones ou tablets no trato de saúde se deve ao fato das várias tecnologias terem emergido principalmente na última década, tornando possível que entidades inteligentes façam o contínuo automonitoramento de pacientes e, caso necessário, sejam disparados alarmes para os respectivos parentes responsáveis e/ou profissionais das áreas médicas a eles associados, de modo que também se disponibilizem dados remotamente quando assim seja necessário.

1.2. JUSTIFICATIVA

Para pessoas que possuem problemas relacionados ao monitoramento de sinais biomédicos que mereçam um acompanhamento mais de perto pelos seus respectivos médicos, é comum que elas compareçam frequentemente a consultórios. Muitas vezes gasta-se tempo e dinheiro com os translados apenas para constatar que os sinais elétricos ou os parâmetros fisiológicos do paciente estão em níveis adequados. Porém com a movimentação de pessoas nos grandes centros urbanos além do notório problema que as cidades enfrentam em períodos de maior movimento, seria conveniente e interessante que os pacientes pudessem ser acompanhados remotamente de modo a tornar tudo mais tranquilo e confortável.

Para pessoas que possuem algum tipo de problema crônico, é comum a utilização de aparelhos que venham a monitorar seus respectivos parâmetros fisiológicos através de sensores atrelados ao corpo, preferencialmente de maneira não invasiva. Convém mencionar que muitos aparelhos não possuem comunicação exterior. No máximo gravam dados em cartões de memória externa, como no caso dos *Holters* para monitoramento de pressão cardíaca, por exemplo.

Por isso é conveniente um projeto de pesquisa que vise unir os vários conceitos englobados pela mobilidade que as coisas podem ter nos dias de hoje.

Como já mencionado, a menos que algum exame mais detalhado seja necessário, apenas uma monitoração de um parâmetro biomédico é bastante conveniente para o conforto da população. O intuito deste trabalho é mostrar como resolver alguns dos problemas como garantia de entrega de dados do sensor a um nó local representado pelo dispositivo móvel local, assim como resolver de maneira mais simples e barata possível a transmissão e armazenamento em servidor de dados remoto.

1.3. OBJETIVOS

Os objetivos compreendem a realização de um trabalho que desenvolva a obtenção do sinal eletrocardiográfico como sendo sinal biomédico em questão, transmitindo-o e recebendo-o via transmissão bluetooth de um sistema microcontrolado para um dispositivo móvel em que esteja rodando um aplicativo android. De forma geral, há a verificação do correto recebimento de dados assim como um controle de fluxo dos sinais que chegam ao dispositivo móvel, a fim de garantir que a visualização dos dados na tela não apresente descontinuidades. Em complemento, os dados recebidos e visualizados poderão ser transmitidos remotamente a outro dispositivo móvel via internet com uso da interface WiFi ou 3G. O dispositivo remoto poderá, deste modo, tanto visualizar os dados *online* em tempo real, assim como visualizá-los posteriormente caso tenha-se decidido por armazenar os sinais na nuvem.

1.3.1. Objetivo Geral

Como objetivo geral tem-se o desenvolvimento de um sistema para visualização de sinais bioelétricos, composto por aplicativos de dispositivos móveis na plataforma android, que fazem a integração de sinais biomédicos provenientes de sensores cardíacos junto ao corpo do paciente, transmitidos via bluetooth por um dispositivo inteligente microprocessado, fazendo com que o sinal seja disponibilizado em um tablet local ou remotamente, via internet e armazenamento em nuvem, em smartphone do médico.

1.3.2. Objetivos Específicos

Para alcançar os resultados desejados foram definidos como objetivos específicos as seguintes etapas:

- Estudar o protocolo bluetooth e a interface de comunicação;
- Escolher o sistema microcontrolado de fácil aquisição que seja adequado ao projeto;
- Desenvolver o circuito de tratamento e adequação do sinal de ECG para amostragem;
- Desenvolver o software para o microcontrolador usando suas interfaces para amostrar e digitalizar o sinal de ECG;
- Desenvolver um protocolo de comunicação para sinais biomédicos mantendo a integridade de tempo, via bluetooth, entre o dispositivo com a interface bluetooth e o dispositivo android;
- Implementar o aplicativo android para o dispositivo local que implemente as funções adequadas ao sinal de ECG, com o devido tratamento de correção de erros e de integridade de tempo;
- Estudar sistemas de armazenamento na nuvem para dados em geral, selecionando uma que ofereça suficiente custo-benefício para este projeto;
- Implementar o aplicativo android para o dispositivo remoto que opere nos modos *online* e *on demand*;
- Testar e avaliar o funcionamento do sistema com sinais simulados de ECG.

1.4. ESTRUTURA DO TRABALHO

A estrutura do trabalho foi dividida em capítulos.

O capítulo 1 apresentou a introdução, contextualização, motivação e os objetivos do trabalho.

O capítulo 2 apresenta a fundamentação teórica e as tecnologias envolvidas com o trabalho.

O capítulo 3 apresenta a metodologia desenvolvida no trabalho.

O capítulo 4 apresenta os resultados obtidos.

O capítulo 5 finaliza o trabalho apresentando as conclusões e comentários sobre a plataforma desenvolvida e os resultados alcançados.

2. FUNDAMENTOS E TECNOLOGIAS

Este capítulo apresenta a fundamentação teórica e as tecnologias, destacando-se a fisiologia cardíaca, a plataforma android, o padrão de comunicação bluetooth, o microcontrolador utilizado, a internet das coisas e o interfaceamento com a nuvem.

2.1. FISIOLOGIA CARDÍACA

A maneira como o sangue circula pelo corpo só ocorre devido ao coração bombeá-lo continuamente através da compressão ventricular ritmada e constante. Para que isto ocorra é necessário que pulsos de corrente elétrica se propaguem de maneira rápida pelo coração, ocasionando contração dos músculos deste órgão. Com o atravessar do sinal elétrico pelo coração, diferentes partes deste órgão vão respondendo ao impulso original, cada parte a seu tempo, conforme as características das fibras musculares e nervosas. A atividade elétrica começa no nódulo sino-atrial e se propaga pelo órgão em direção aos ventrículos. Este processo ocorre de maneira contínua, fazendo com que vários pulsos surjam com o passar do tempo (GUYTON, 1992).

A capacidade de o coração bater ocorre devido à presença de células que funcionam como se fosse um marcapasso. As células cardíacas possuem algumas propriedades: a de excitabilidade que corresponde à capacidade dela responder a um impulso; contratibilidade que corresponde à resposta de contração muscular para o estímulo elétrico; condutibilidade que corresponde à capacidade de conduzir o estímulo elétrico do meio externo para uma célula adjacente por meio da membrana celular; e automaticidade que é a capacidade de que algumas células do coração têm de gerar estímulos elétricos automaticamente. Em se tratando de células adjacentes elas possuem a capacidade de se influenciarem mutuamente, fazendo seu potencial interno variar de positivo para negativo, e logo em seguida voltando ao potencial positivo, desencadeando a geração de um pulso que por sua vez se espalha pelo coração de acordo com o respectivo caminho elétrico. Este

pulso percorre desde a parte superior do coração onde foi gerado, no nódulo sino atrial, rumando para as partes inferiores (HODGKIN, HUXLEY, 1952).

O potencial elétrico em condição de regime permanente equivale à aproximadamente -90mV, também chamado de potencial de repouso. Este potencial é negativo porque existe maior concentração de ions de potássio no interior da célula, assim como há uma maior concentração de sódio no exterior da célula. Com isto a diferença entre os potenciais interior e exterior é negativa. No entanto o pulso elétrico medido na célula miocárdica ocorre quando há passagem do fluxo de ions de sódio positivos (Na+) para o interior das células cardíacas, deste modo tornando o exterior delas mais negativo que o interior. Mais tarde, em um processo chamado repolarização, os íons positivos de potássio (K+) migram para fora da célula tornando o exterior dela positivo (HODGKIN, HUXLEY, 1952). As células do músculo cardíaco são auto-excitáveis e conduzem os sinais de impulso pelo órgão. Elas se dividem em: nó sinusal ou sinoatrial (NSA), feixes internodais, feixe de Bachman, nó atrioventricular (NAV), sistema His-Purkinje (feixe de His e fibras de Purkinje) (MALMIVUO, PLONSEY, 1995).

Podem-se considerar como importantes para a eletrofisiologia do coração os seguintes passos, que são registrados pelo eletrocardiograma:

- O estímulo elétrico surge no NSA formado por nervos parassimpáticos. O sinal se propaga pelos feixes intermodais, formados pelas células nodais tipo P que são responsáveis pelo batimento cardíaco e as células de transição (células T) que envolvem as células P. Uma onda de despolarização, propagada a partir dos átrios está representada no ECG como onda P (LEWIS, OPPENHEIMER e OPPENHEIMER, 1910).
- Os feixes Internodais conectam o NSA ao NAV por três vias: anterior, média e posterior. A anterior é responsável por levar fibras para o átrio esquerdo que então se comunica com o feixe de Bachman. Ele conduz o impulso do atrio direito para o esquerdo. A via média leva fibras tanto para o átrio esquerdo quanto para o NAV, enquanto a via posterior leva fibras para o átrio direito e para o NAV (JAMES, 1963).
- O impulso elétrico, uma vez atingindo o NAV, sofre um atraso de 100 milissegundos. O NAV tem a função de conduzir o impulso elétrico até os ventrículos através do feixe de His. O NAV está dividido em três

áreas: primeiro a que faz a transição do musculo atrial com o NAV; segunda é a chamada área compacta; terceira é a responsável pela transição do NAV com o feixe de HIS (CARVALHO *et al*.,1969).

- Do NAV em diante o impulso elétrico segue adiante pelo septo interventricular, através do feixe de His pelo atrioventricular que então é separado em seus ramos esquerdo e direito em direção aos respectivos ventrículos esquerdo e direito. O ramo do lado esquerdo do feixe de His divide-se em subdivisões: fascículo anterior e fascículo posterior. (HECHT, 1973) e acabam em várias fibras chamadas de Purkinje. As fibras de Purkinje se disseminam pelo miocárdio sendo responsáveis pela sua despolarização, indo o sinal do endocárdio para o epicárdico, desencadeando uma contração ventricular (JAMES, 1963).
- O sinal elétrico ou impulso gerado que se propaga do nódulo átrio-ventricular para as fibras de Purkinje é o responsável pela geração do complexo chamado QRS, normalmente usado para a determinação da frequência cardíaca devido à sua variação abrupta de potencial elétrico, mais fáceis de serem percebidas por algoritmos computacionais. A variação com que o sinal desce o miocárdio, de cima para baixo, em termos de velocidade de propagação, pode variar de pessoa para pessoa, conforme ela apresente algum problema no coração. A chamada despolarização ventricular que ocorre antes da contração dos ventrículos está, deste modo, representada graficamente pelo complexo QRS.

Na Figura 1 tem-se representado os vários sinais existentes nas várias partes do coração. O sinal elétrico efetivamente obtido pelos eletrodos na superfície da pele, quando da medida do ECG na prática, é o resultado, em cada instante de tempo, das somas dos vários sinais por superposição. Estes vários sinais já somados estão representados na figura 2 e são consequência da despolarização seguida por repolarização dos potenciais de ação das células do coração.

Pode haver ainda um último trecho do sinal ECG representado pelo trecho branco do gráfico da figura 1, chamado onda U. Ela representa uma repolarização atrasada dos ventrículos (MALMIVUO, PLONSEY, 1995).

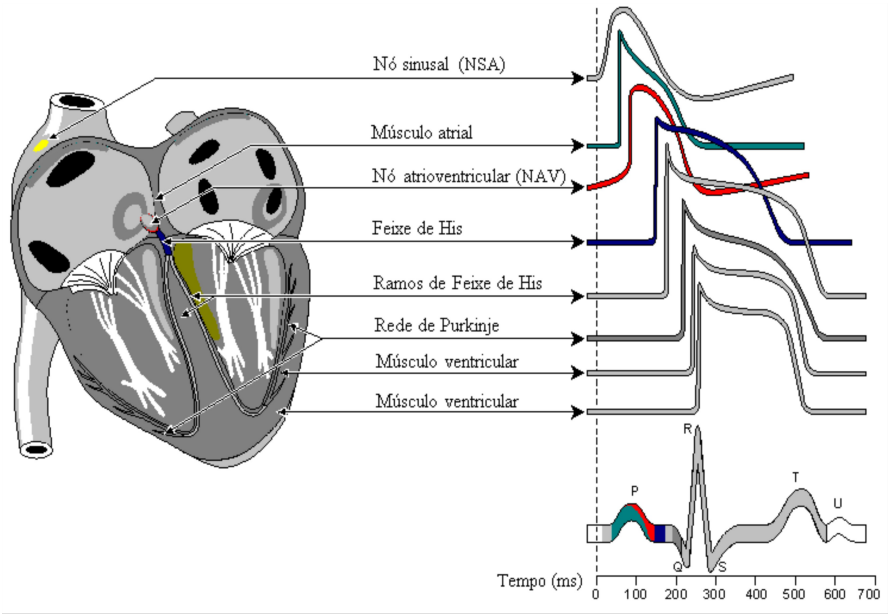

Figura 1 – Sinais elétricos pelo coração e sua composição final.
Fonte: Modificado de Malmivuo e Plonsey (1995).

2.1.1. Formação do Eletrocardiograma

Existem diversas maneiras de se obter o sinal cardíaco, desde maneiras não invasivas a maneiras invasivas. Nas não invasivas, normalmente colocam-se eletrodos no tórax e nos membros, porem existem situações que eletrodos são colocados dentro do esôfago (derivação esofágica), no interior do coração (derivação endocárdica) ou na superfície do coração (derivação epicárdica) (RIEIRA *et al*., 2008). O eletrocardiógrafo ECG mede as correntes elétricas cardíacas formadas em sequência no músculo cardíaco. Para a aquisição dos sinais cardíacos precisa-se de eletrodos de captação na superfície do corpo em posições padronizadas (GUIMARÃES *et al*., 2003).

As derivações existentes se diferenciam em três tipos: bipolares, tipo unipolares e precordiais. As bipolares foram primeiramente sugeridas por Einthoven e medem a diferença de potencial entre dois pontos, sejam eles braço esquerdo e direito, perna direita e ombro esquerdo ou perna direita e ombro direito. As derivações unipolares foram propostas por Wilson e tem três derivações nas extremidades aVR, aVL e aVF.

As derivações precordiais chamadas também de horizontais, de V1 a V6, têm o objetivo de medir a atividade elétrica por diferentes ângulos (MALMIVUO, PLONSEY, 1995) e estão dispostas de acordo com a Figura 2.

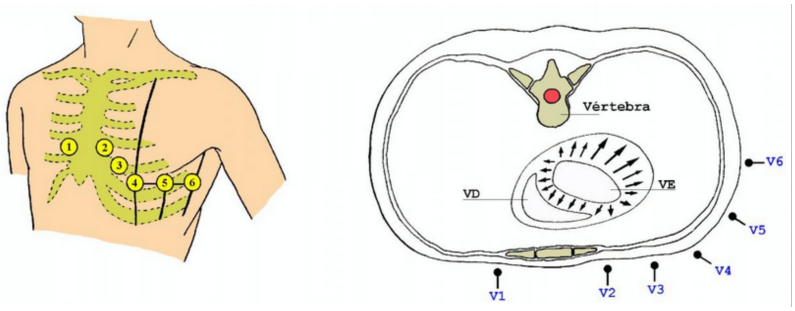

Figura 2 – Derivações precordiais

Fonte: modificado de http://ecg.med.br/assuntos-online/derivacoes.

2.1.2. Meios de Determinação do Complexo QRS

Ao longo dos anos desenvolveram-se muitos algoritmos computacionais para detectar o complexo QRS, determinando precisamente o início de cada ciclo cardíaco. Existem então vários métodos para detecção do complexo QRS. A detecção dele se faz devido a ele ter uma amplitude bastante saliente quando comparada ao resto do ciclo do sinal de ECG. Ainda assim, necessita-se filtrá-lo conforme o grau de precisão do ECG a ser obtido. Esta filtragem pode gerar sobrecarga computacional quando feita por filtros digitais, sendo importante em aplicações em tempo real para monitoramento do coração (ZHANG, LIAN, 2009).

De um lado existem técnicas envolvendo heurísticas bastante sofisticadas do ponto de vista matemático para a determinação do complexo QRS, que não cabem descrição neste trabalho. É perceptível, contudo, o melhoramento obtido no desenvolvimento destas técnicas. Para este trabalho um método baseado nas derivadas primeira e segunda foi utilizado. Os algoritmos baseiam-se na amplitude da primeira derivada, e na segunda derivadas. Este algoritmo foi descrito por Gustavson *et al*. (1977). O algoritmo se baseia nas amplitudes destas derivadas quando calculadas de maneira discretizada conforme a amostragem e também adotam um chamado ponto de decisão para a determinação da provável presença

de QRS. Fraden e Neuman (1980) desenvolveram o método de detecção em que um limiar é calculado como uma fração do valor de pico do sinal de ECG.

Os algoritmos baseados na primeira derivada foram descritos por Holsinger, Kempner e Miller (1971).

Um algoritmo foi apresentado por Engelse e Zeelenberg (1979). Okada (1979) apresentou um método com três filtros de média móvel, seguido por filtragem passa-baixas e limiar para detectar o complexo QRS.

Algoritmos que utilizam duas derivadas foram desenvolvidos por Balda *et al*. (1977). Posteriormente Ahlstrom e Tompkins (1983) propuseram um algoritmo de detecção de complexo QRS em tempo real considerando inclinação, largura dos pulsos e inclinação do sinal. O algoritmo inclui filtros, operações com derivadas, integrações e um referencial de limiar.

2.1.3. Formação de Artefatos no ECG

O sinal do ECG formado dentro do músculo cardíaco que é conduzido até a superfície do corpo pode sofrer interferência de outros sinais elétricos que não o do coração. Todos estes sinais adicionais podem ser considerados como ruídos, mas devido ao processo de captação, somam-se ao ECG gerando pequenas distorções no sinal cardíaco. Suas origens podem ser divididas em três classes: biológica, instrumental e externa (BARBOSA, 2003):

- Artefato de origem biológica: É o artefato gerado a partir de sinais espúrios provindos de atividade biológica como a respiração ou outra atividade muscular. A atividade muscular é a principal fonte que descaracteriza o ECG (SÖRNMO, 1993; BENMALEK, CHAREF e ABDELLICHE, 2010);
- Artefato de origem instrumental: A fonte de alimentação do equipamento pode gerar sinais espúrios no ECG, descaracterizando-o (WEBSTER, 2008). Até mesmo o fato dos cabos que conectam os eletrodos ao equipamento serem muito compridos ou sem a devida blindagem contribuem para a maior quantidade de ruídos, seja da rede ou de qualquer outra fonte elétrica. Além disto, o mau contato do eletrodo e a colocação inadequada de eletrodos também são fontes de

interferência sobre os sinais (WALD, STONE e KHAMBATTA, 1990; BENMALEK, CHAREF e ABDELICHE, 2010);

- Artefato de origem externa: origina-se nas fontes de energia conectadas ao equipamento gerando interferência por indução eletromagnética (WEBSTER, 2008). Em relação ao ruído, podem-se considerar os irradiados e os conduzidos. Os ruídos irradiados são oriundos dos campos eletromagnéticos formados em algum lugar do sistema de modo a se acoplar ao sinal do ECG.

A propagação é feita pelas linhas de fase e neutro do aparelho até a rede elétrica, muitas vezes influenciada por outros equipamentos também conectados à mesma rede de alimentação elétrica (SANCHES, 2003). Os ruídos conduzidos são os propagados por condução. Precisam, portanto, de um meio condutor como, por exemplo, o cabeamento elétrico do próprio equipamento.

2.2. PLATAFORMA ANDROID

Com o avançar das tecnologias apropriadas para dispositivos móveis, acabou-se por criar um novo nicho de aplicativos no mercado de software dedicado a tais dispositivos. Várias são as plataformas existentes, mas a mais disseminada é a da Google, o android, que é um dos sistemas operacionais para dispositivos móveis.

O android é uma plataforma aberta, tipo *Open Source* que foi desenvolvida pela Google visando sua utilização em aplicativos móveis. O seu desenvolvimento se deu desde 2005 por Dan Bornstein entre outros, porém a primeira versão introduzida no mercado de smartphones se deu em 2008. O android tem sua estrutura baseada sobre um núcleo Linux, somando-se às bibliotecas Java. A manutenção deste sistema operacional no ramo dos dispositivos móveis é feita pelo consórcio *Open Handset Alliance* (OHA), formado por mais de 40 empresas envolvidas com telecomunicações, tendo à sua frente a Google Inc. como fornecedora da base android (LECHETA, 2010).

Com o intuito de se desenvolver aplicativos com milhares de adeptos no mundo, ou seja, cuja familiaridade fosse mais ampla, tornando-se de mais fácil utilização para usuários das áreas médicas, foi escolhido o android (LECHETA, 2010).

A plataforma do sistema operacional android funciona tendo como base uma pilha de softwares como ilustrado na Figura 3 (PROJECT, 2015).

A arquitetura da Figura 3 foi elaborada intencionalmente para dar suporte ao desenvolvimento de projetos de software que envolva dispositivos móveis tais como tablets, smartphones, assim como os híbridos tais como os phablets.

A plataforma contém diversas camadas que vão desde a mais baixa envolvendo serviços mais primitivos do sistema operacional, até a camada mais alta que controla o dispositivo propriamente dito, somando-se aos aplicativos fornecidos com o aparelho móvel como um todo, normalmente disponibilizados pelo próprio fabricante conforme o modelo, tais como o que faz as ligações celulares que é naturalmente a função primordial do aparelho (BORNSTEIN, 2008).

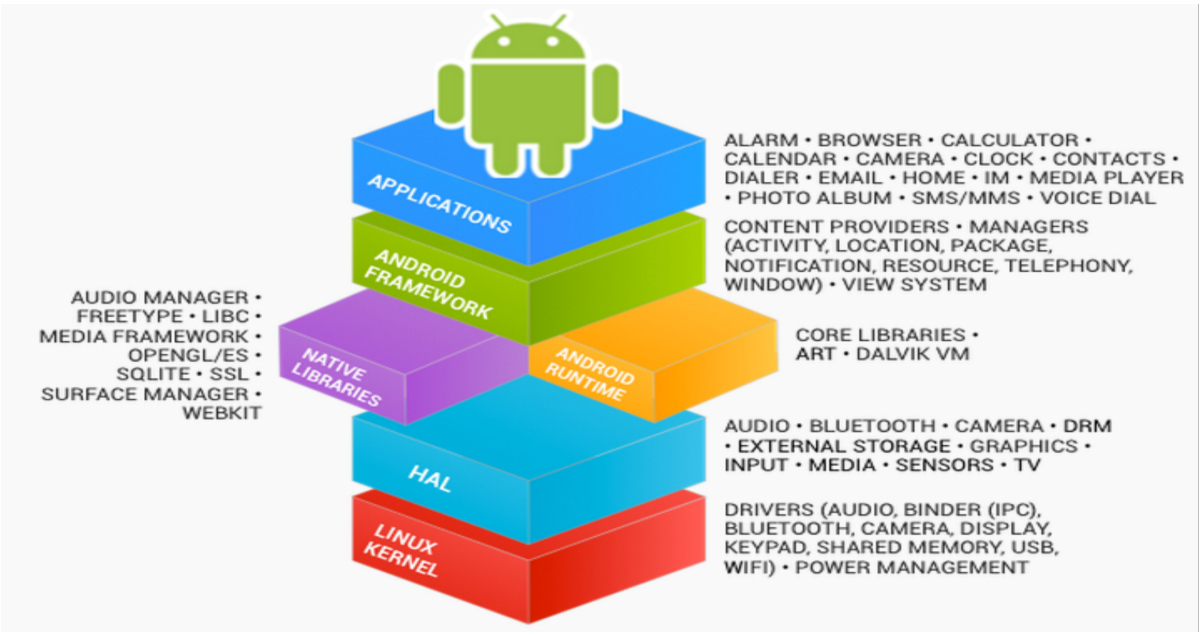

Figura 3 – Pilha Android
Modificado de (PROJECT, 2015).

O objetivo do sistema android é o de se poder desenvolver aplicativos para todos os dispositivos móveis e para isto existem muitas bibliotecas que vão de bibliotecas gráficas, por exemplo, passando pelas que tratam do hardware como câmeras, interfaces USB, NFC, etc, além das ferramentas próprias para desenvolvimento de aplicativos (LECHETA, 2010). Para isto, a todo instante novas bibliotecas são incorporadas ao sistema como um todo na forma de novas Interfaces para Programas Aplicativos ou *Application Programm Interfaces* (APIs), disponíveis para baixar pelo ambiente de desenvolvimento android, a fim de atualizar o ambiente de desenvolvimento com todos os novos dispositivos e tecnologias que surgem e são incorporadas aos dispositivos móveis à medida que o tempo passa (BORNSTEIN, 2008).

É também necessário o kit para desenvolvedores de software chamado de *Software Development Kit* (SDK) que usa todas as bibliotecas e ferramentas disponíveis para o desenvolvimento de um aplicativo a fim de se poder criar os mais diversos aplicativos android. Estes kits variam de acordo com a plataforma alvo para a qual o aplicativo é desenvolvido. No entanto, aplicativos mais simples funcionam em aparelhos sofisticados. Já aplicativos que usam funções mais recentes não rodarão em aparelhos muito simples. Já se está na versão seis do android, chamada *Marschmallow*, mas ainda existem muitos smartphones operando com versões *Ginger Bread*, versão 2.3 (BORNSTEIN, 2008).

As principais versões do android estão apresentadas na Tabela 1.

Tabela 1 – Versões da Plataforma Android.
Modificado de android.com

Versão da plataforma	Nível da API	Nome da versão	Lançamento
Android 6.0	23	*Marschmallow*	Maio/2015
Android 5.1, 5.1.1	22	*Lollipop MR1*	Fevereiro/2015
Android 5.0	21	*Lollipop*	Novembro 2014
Android 4.4, 4.4.2	19	*KitKat*	Outubro 2013
Android 4.3	18	*Jelly Bean MR2*	Julho 2013
Android 4.2	17	*Jelly Bean MR1*	Novembro/2012
Android 4.1	16	*Jelly Bean*	Julho/2012
Android 4.0.3, 4.0.4	15	*Ice Cream Sandwich MR1*	Dezembro/2011
Android 4.0, 4.0.1, 4.0.2	14	*Ice Cream Sandwich*	Outubro/2011
Android 3.2	13	*Honeycomb MR2*	Julho/2011
Android 3.1.x	12	*Honeycomb MR1*	Maio/2011
Android 3.0.x	11	*Honeycomb*	Não disponivel
Android 2.3.4, 2.3.3	10	*Gingerbread MR1*	Fevereiro/2011
Android 2.3.2, 2.3.1, 2.3	9	*Gingerbread*	Dezembro/2010
Android 2.2.x	8	*Froyo*	Maio/2010
Android 2.1.x	7	*Eclair MR1*	Janeiro/2010
Android 2.0.1	6	*Eclair 0 1*	Não disponivel
Android 2.0	5	*Eclair*	Não disponivel
Android 1.6	4	*Donut*	Setembro/2009
Android 1.5	3	*Cupcake*	Abril /2009
Android 1.1	2	*Base 1 1*	Fevereiro/2009
Android 1.0	1	*Base*	Setembro/2008

A fim de se manter o controle sobre como a visualização do aplicativo seria feita para os diversos aparelhos móveis existentes, considerando que cada dispositivo possui sua própria versão de android, pode-se fazer uso do assim chamado *Android Virtual Device* (AVD) que também faz parte do ambiente de desenvolvimento. O AVD é configurado pelo AVD *Manager* dentro dos ambientes Eclipse ou Android Studio. A aparência dele está ilustrada na figura 4. Através dele simula-se o aplicativo em determinado aparelho móvel, independente do fabricante, sem a necessidade do aparelho. Algumas funções, entretanto, não podem ser plenamente testadas, caso elas usem sistema de GPS continuamente, por exemplo (BORNSTEIN, 2008).

Na Figura 4, clicando-se em New abre-se uma nova janela contendo as configurações do smartphone que se queira simular no ambiente Eclipse, por exemplo. Nesta configuração, desde a densidade de pixel por polegada quadrada,

até a quantidade de memória volátil interna ou externa do dispositivo podem ser simuladas. Entretanto, alguns periféricos como o sistema de comunicação por Bluetooth não podem ser simulados usando este ambiente virtual.

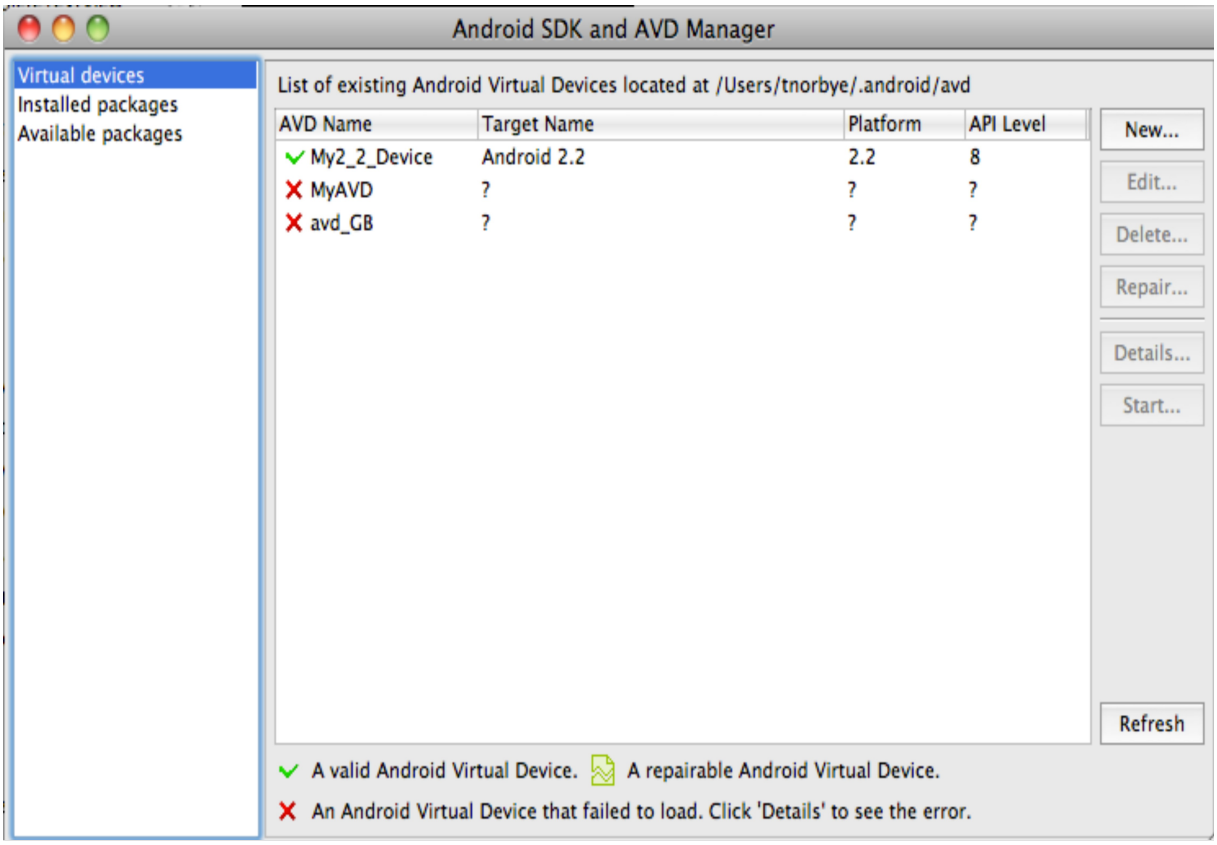

Figura 4 – Visual do Gerenciador Virtual do Android
Fonte: Autoria Própria

2.2.1. Arquitetura Android

A Figura 5 mostra as camadas do sistema android dispostas em uma pilha. A camada inferior da plataforma android corresponde ao *kernel* de Linux. Para facilitar a referência desta camada, este trabalho referencia esta camada como camada 1. Esta camada fornece todos os recursos primitivos sobre as quais o resto do sistema vai se basear, fornecendo subsídios para as camadas superiores, como por exemplo, o sistema de permissões usadas em aplicativos para tornar restrito o acesso aos dados e recursos do sistema somente àqueles processos que possuam as respectivas autorizações (PROJECT, 2015).

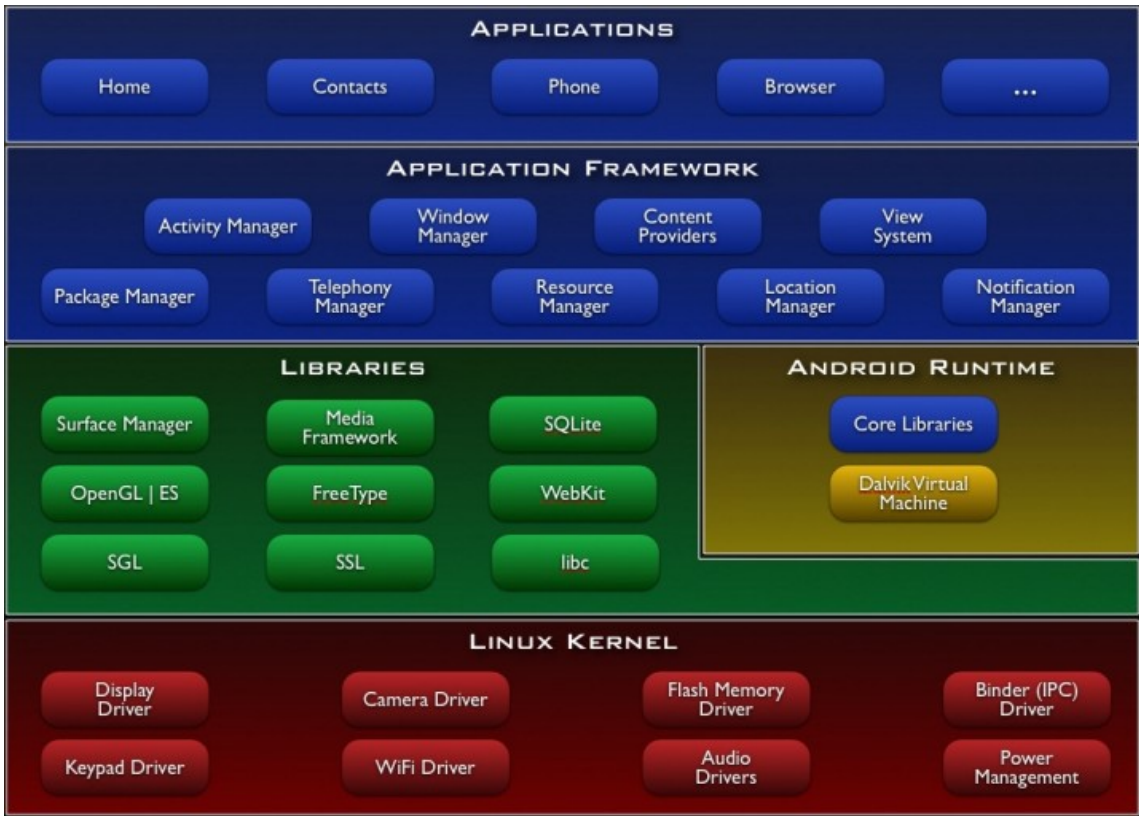

Figura 5 – Camadas do Android
Fonte: adaptado de android.com

Acima da camada 1 tem-se várias outras camadas a saber: bibliotecas do android e *run-time* como camada 2; *framework* do aplicativo como camada 3; e por fim, a camada dos aplicativos como camada 4. Estas denominações de camadas 1, 2, 3 e 4 são apenas a fim de melhor elucidar o conteúdo das páginas que seguem, não fazendo parte do jargão de palavras normalmente usadas em textos explicativos sobre o sistema android.

2.2.2. Linux Kernel Android

A Google baseou-se na versão 2.6 do *kernel* Linux para fazer o *kernel* Android. Apesar disto há uma grande diferença entre os dois sistemas, pois o *kernel* Android possui uma infinidade de pacotes, também chamados de bibliotecas, que são adicionadas ao sistema e que foram acrescidas ao *kernel* Linux, fazendo-o ter assim tantas funcionalidades (ANDROID, 2015).

A camada do *kernel* gerencia a memória, os processos em andamento de forma a permitir que vários processos ocorram simultaneamente sem que um

interfira no outro, além de lidar com detalhes primitivos do sistema quanto ao gerenciamento de arquivos e de sistemas de entrada e saída do dispositivo. O *kernel* é quem controla os *device drivers* permitindo ao android se comunicar com uma gama ampla de componentes externos de hardware, como câmeras, diversos tipos de sistemas de rádio e interfaces dos mais variados tipos (ANDROID, 2015).

O *kernel* Linux ainda inclui alguns componentes mais específicos, como o controlador e gerenciador de energia, pois dispositivos móveis frequentemente esgotam suas baterias rapidamente. O *kernel* também tem seu próprio gerenciador de memória devido a estes recursos serem limitados em sistemas móveis quando comparados a sistemas Desktops. Ainda, o sistema android possui seu próprio mecanismo de comunicação entre processos ou *Interprocess Communication* (IPC) e o *Binder* que permite que processos troquem informações entre si de uma maneira peculiar a sistemas android (BORNSTEIN, 2008).

2.2.3. Bibliotecas e Runtime Android

Na camada 2 tem-se as bibliotecas para banco de dados, tratamento de vídeo, entre outras, como as descritas a seguir, assim como o tratamento dos arquivos gerados para a máquina virtual Dalvik (PROJECT, 2015).

2.2.3.1. Bibliotecas do Android

Na camada 2 estão as bibliotecas do sistema e o sistema de tempo real ou android *run-time system* que normalmente estão escritos em linguagem C ou C++. Por isto estas bibliotecas são frequentemente tratadas como bibliotecas nativas do sistema. Elas controlam atividades centrais do aplicativo, tais como uma simples atualização do display, por exemplo. As bibliotecas disponíveis no android são (PROJECT, 2015):

- *System C Library*: responsável pela criação interna de processos ou *threads*, do processamento computacional matemático, da alocação de memória, entre outros, basicamente definindo as chamadas ao sistema;

- *Surface Manager*: corresponde ao gerenciador de superfícies, responsável pela atualização da tela do dispositivo, compondo gráficos em duas ou três dimensões;
- Bibliotecas 3D: usam aceleração por hardware e renderização por software, através de OpenGL ES1.0 (PROJECT, 2015);
- *Lib Webcore*: é quem renderiza páginas de internet no navegador android;
- *FreeType*: é um gerenciador de fontes para utilização em diversas plataformas (TURNER *et al*., 2006);
- *Media Framework*: fornecem suporte quando são executados os formatos mais comuns de imagem, de vídeo ou áudio ;
- SQLite: coresponde a um banco de dados de código aberto gerenciado pelo sistema, indicado para dispositivos com hardware limitado, tais como dispositivos móveis em geral (ANDROID, 2015);
- O OpenGL é responsável pelo tratamento de gráficos sofisticados a serem mostrados na tela enquanto o SQLite é o responsável por gerenciar os dados armazenados internamente.

2.2.3.2. RunTime do Android

Ainda na camada 2, existe uma biblioteca central de processos android que suporta os processos em andamento. Nesta camada existem dois principais componentes:

- Biblioteca Central Java (*Core Java Library*);
- Máquina Virtual Dalvik (*Dalvik Virtual Machine* ou DVM).

Para que fosse mais fácil a programação android, o sistema permite o uso de java, assim como os pacotes com classes java básicos Java.* e JavaX.* que incluem, por exemplo, estruturas de software usadas na programação Java tais como tipos de dados, mecanismos de processos simultâneos ou *concurrency mechanisms*, e arquivos de entrada e saída (LECHETA, 2010).

Para o android em si, existe o pacote android (Android.*) que gerencia o ciclo de vida de aplicativos móveis, e o pacote ORG (ORG.*) que dá suporte às operações e serviços referentes à internet.

A DVM é o software que executa aplicativos android da mesma forma como a máquina virtual java executa códigos java. O código fonte escrito em java é compilado para vários arquivos tipo java, com seus *bytecodes*. Em seguida algo chamado DX transforma os *bytecodes* em um único arquivo DEX, com um tipo de *bytecode* diferente do java. Este arquivo normalmente consta como classes.dex. Este arquivo do tipo DEX é então empacotado junto com outros recursos do aplicativo para então poder ser instalado no dispositivo móvel. Quando o usuário executa o aplicativo, o *Dalvik Virtual Machine* executa o arquivo DEX do pacote do aplicativo. A máquina virtual Dalvik, ao contrário da máquina virtual java, foi desenvolvida para executar aplicativos em dispositivos com recursos físicos escassos de memória, quantidade de carga de bateria, além de Unidades Centrais de Processamento (CPUs) mais lentas (LECHETA, 2010).

2.2.4. Framework do Aplicativo

A camada 3 corresponde ao chamado *framework* que fornece os recursos para o programador desenvolver novos Aplicativos. Nela estão os elementos para interface com o usuário. Estas partes do sistema são importantes, pois formam as várias partes de um aplicativo android, tais como botões de tela, caixas de diálogo, recursos para o sistema de localização, de notificação, entre outros (ANDROID, 2015). São elas:

- Gerenciador de Pacotes (*Package Manager*) gerencia todos os aplicativos instalados no dispositivo móvel. Cada Aplicativo corresponde a um pacote de aplicação instalado no smartphone. O Gerenciador de Pacotes faz a troca de informações de dados entre aplicativos, assim como permite que um aplicativo requeira um serviço de outro;
- Gerenciador de Janela (*Window Manager*) gerencia as várias janelas que podem ser formadas por um aplicativo. Por exemplo um aplicativo pode mostrar a barra de notificações como sendo uma janela, além da janela da aplicação propriamente dita em andamento, sem mencionar possíveis subjanelas correspondentes aos menus ou caixas de diálogo daquele aplicativo;

- Gerenciador de Recursos (*Resource Manager*) gerencia recursos tais como objetos tipo strings, objetos gráficos, arquivos de layout;
- Gerenciador de Atividades (*Activity Manager*) gerencia as atividades em curso no dispositivo móvel. É bastante frequente termos uma atividade relacionada a uma aplicacção android. Porém existem aplicativos com várias possíveis atividades através das quais o usuário pode ir. Neste caso o gerenciador de atividade coordena e dá suporte para esta navegação;
- Provedores de Conteúdo (*Content Providers*) permitem que aplicativos armazenem e compartilhem dados e informações estruturadas;
- Gerenciador de Localização (*Location Manager*) permite que aplicativos recebam informações sobre localização do dispositivo móvel;
- O Gerenciador de Notificações (*Notification Manager*) permite ou não que aplicativos coloquem informações na barra de notificações. Estas informações se referem aos eventos que tenham ocorrido no sistema, e que sejam relevantes ao aplicativo.

2.2.5. Camada do Aplicativo

A camada 4 contém os aplicativos mais comuns no uso diário que vêm junto com o dispositivo móvel. Como exemplo existe o aplicativo que transforma o sistema em um aparelho celular quando se trata de um smartphone, assim como aplicativo que permite a navegação pela internet, ou a própria tela chamada de *home screen* (ANDROID, 2015).

Ao clicar-se em *home screen*, uma janela semelhante à mostrada na Figura 6 aparece, formada por um conjunto de TABS.

Cada TAB corresponde a uma interface de usuário diferente, seja a TAB que mostra um conjunto de figuras e texto que imita um teclado de celular, seja a que mostra o histórico de ligações feitas e recebidas, seja a dos contatos telefônicos. Cada uma destas TABS mostra componentes do sistema *View* do android, como Botões e *Textviews*. Os *TextViews* são elementos que mostram textos e números impressos na tela, com formatação e localização na tela predeterminados. Ao

pressionar um dos botões do teclado, o sistema "ouve" e reage mostrando na respectiva posição de tela os números sendo formados ou um texto sendo escrito, neste caso correspondente ao nome do contato telefônico. Todos os números telefônicos são informações de conteúdo, gerenciadas pelo *ContentProvider* que as compartilha com outras aplicações que solicitam o uso destes dados, tais como facebook ou twitter, entre outros (BORSTEIN, 2008).

Figura 6 – Ícones de uma *home screen*.
Fonte: Autoria própria.

Tudo o que vai sendo impresso na tela na forma de caracteres ou números são chamados strings. Quando se trata de texto, este pode estar em diferentes versões, uma para cada idioma diferente já que o android é disponível pelo mundo inteiro. Quem controla qual das strings vai aparecer na tela é o *ContentProvider*.

Porém, ao mesmo tempo que o usuário usa este aplicativo, o Gerenciador de Notificações está eternamente "ouvindo" às mensagens do sistema, seja vindos do próprio dispositivo móvel, seja vindo de mensagens sms, mms ou outras. Caso esteja-se digitando um número e alguma mensagem importante apareça, mesmo que vinda de outro aplicativo, o gerenciador de notificações pode apresentar na tela, na área de notificações, tal mensagem recebida (BORNSTEIN, 2008).

Uma das coisas boas que o sistema android tem é que nenhum dos aplicativos que vêm junto com o dispositivo móvel são fixos. Isto quer dizer que

pode-se trocar tal aplicativo por outro feito pelo próprio usuário, caso assim se deseje.

2.2.6. Elementos do programa Android

Os aplicativos android utilizam alguns blocos básicos para a sua construção. As *Activities* são as mais conhecidas, pois normalmente são as classes utilizadas logo no início do aprendizado em android. Existe também a classe *Service* que de maneira diferente das *Activities*, roda normalmente em pano de fundo ou *background*. Tem-se ainda o *Intent Receiver* e o *Content Provider*. Segue uma breve descrição destas partes (ANDROID, 2015).

Muitas vezes se vê definida uma *Activity* como sendo uma tela que apresenta o conteúdo de um programa android. A *Activity* é implementada através de uma classe que mostra ao usuário a interface gráfica do sistema, assim como pode reagir a eventos gerados externamente ou a eventos gerados pelo usuário, como um pressionar de um botão, por exemplo. Toda *Activity* possui um assim chamado ciclo de vida conforme apresenta a Figura 7.

O sistema android que controla todas as janelas faz estas passarem pelos ciclos de criação, início, pausa, continuação, parada e destruição. Existe outro processo chamado recomeço pelo qual uma atividade pode passar. Deste modo, quando um programa android possui várias telas, ao passar de uma tela para outra, internamente é como se estivesse passando de uma classe do tipo *Activity* para outra classe do mesmo tipo, ficando a primeira parada e seus dados colocados em uma pilha interna. Em outras palavras, o estado da primeira janela muda de estado ativo para o estado parado. Esta mudança de estado depende do respectivo método referente à classe atividade em execução. Quem executa isto é o sistema android. Mais tarde, quando o usuário retornar para a primeira tela, ela estará pronta para a consulta, mostrando os valores correntes das variáveis, ainda que alterados. Para se alternar entre as várias telas, usa-se de uma outra classe chamada de *Intent*. Nesta variável tipo *Intent* anexa-se dados, entre os quais a variável que define a nova *Activity* destinatária, e após o chamar desta variável, o sistema responde com a mudança de janela. Esta operação é bastante simples, porém pode ser sofisticada a

ponto de flexibilizar o sistema em termos de dados enviados à nova janela ou em termos de para onde voltar quando a segunda janela terminar (ANDROID, 2015).

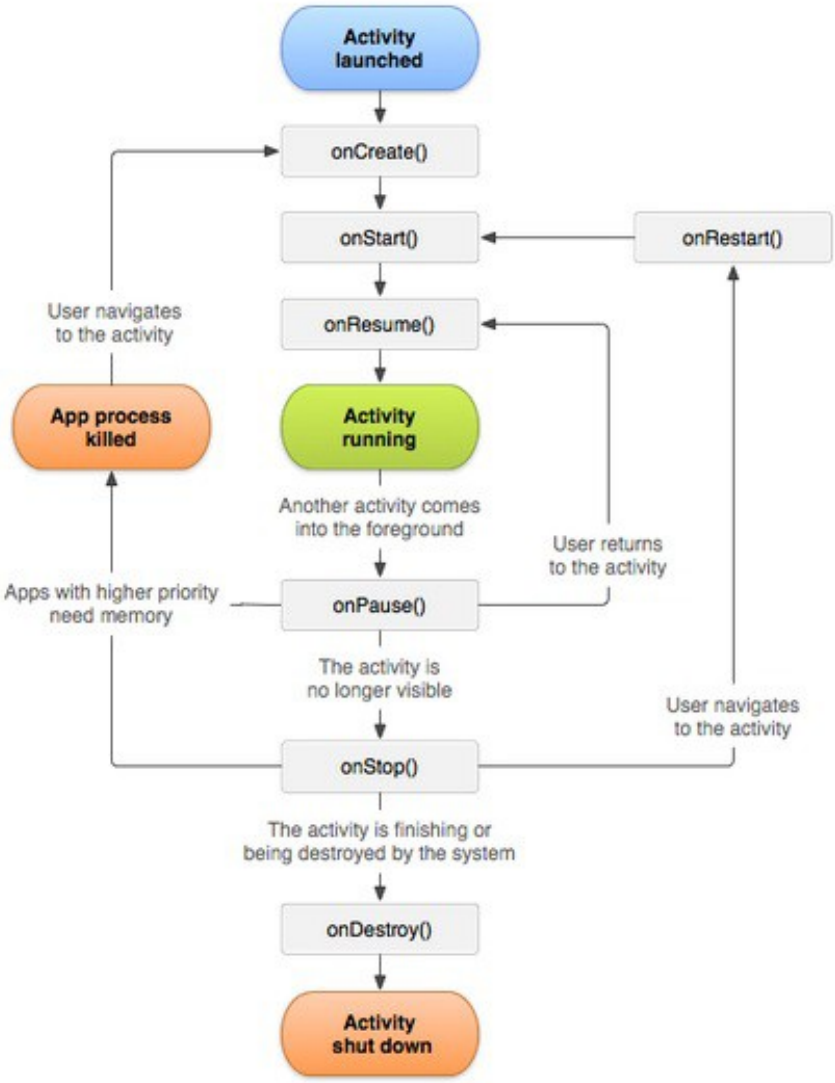

Figura 7 – Ciclos de Vida de uma Atividade Android
Fonte: developer.android.com/reference/android/app/Activity.html

Usa-se o Receptor de Intenções (*Intent Receiver*) quando se necessita que o sistema execute alguma função ao reagir a um evento externo, tal como uma chamada telefônica, ou a descoberta de novos dispositivos bluetooth nos arredores, por exemplo. Neste caso a referida classe do receptor de intenções realiza os procedimentos nela programados. Para o usuário, entretanto, todo este procedimento é transparente. Resumidamente, é como se o sistema android possuísse vários processos em paralelo, cada um ocorrendo ao seu tempo (ANDROID, 2015).

Há o chamado provedor de conteúdos (*Content Provider*) que tem como finalidade compartilhar dados que estão correntemente armazenados com um outro aplicativo. Para tal, assim como toda a classe, esta classe possui vários métodos que permitem o compartilhamento dos dados com outros aplicativos.

Por último há a classe chamada Serviço (*Service*). Existem várias maneiras de se tratar desta classe. Resumidamente, ela trata de códigos de programação que permanecem em execução em segundo plano depois de serem ativados de alguma forma. É possível, por exemplo, chamar uma rotina que fique rodando em pano de fundo com uma música a tocar, enquanto executa-se outra *activity* em paralelo. Ou mesmo é possível deixar um programa ou rotina a ler informações da localização do dispositivo móvel de maneira absolutamente transparente ao usuário, enquanto o usuário faz outras tarefas. Muitos códigos maliciosos podem fazer uso desta característica do sistema android (ANDROID, 2015).

2.3. ARQUITETURA BLUETOOTH

O bluetooth se trata de uma tecnologia de comunicação de dados sem fio que é utilizada tanto para pequenos dispositivos de hardware, quanto em diversos aparelhos móveis, sejam eles tablets ou smartphones. Com o transcorrer dos anos, vários padrões de bluetooth foram sendo aprimorados, começando pelos mais lentos e antigos até os padrões mais novos que atingem muitos megabits por segundo em termos de transmissão de dados, sem mencionar os padrões que dispõe de níveis de consumo de energia muito reduzidos.

O alcance entre os dispositivos é variável, dependendo das condições do ambiente físico em que se encontram, como também da classe bluetooth a que pertencem. A tabela 2 apresenta a relação entre potência e alcance de cada classe. Pode-se enviar e receber dados de maneira simples, procurando atender uma gama muito ampla de aplicações, motivo pelo qual o bluetooth é muito utilizado (CREATIVE, 2015).

Tabela 2 – Classes de potência e alcance do Bluetooth.
Fonte http://www.infowester.com/bluetooth.php

CLASSE	POTÊNCIA MÁXIMA	ALCANCE
1	100 mW	100 metros
2	2,5 mW	10 metros
3	1 mW	1 metro

2.3.1. Frequências de operação do bluetooth

Com a grande difusão de uso do bluetooth pelo mundo, as suas aplicações abrangem uma gama bastante ampla, da industrial até a médica, na faixa ISM Industrial, *Scientific*, *Medical* que vai de 2,4 a 2,5 GHz (BLUETOOTH SIG, 2015). Principalmente na faixa de 2,4 GHz podem surgir conflitos com outras faixas de sinal, como por exemplo, as usadas das redes WiFi, microondas ou telefones sem

fio residenciais. Por isto, uma das preocupações do uso do bluetooth é a de que a frequência utilizada não interfira com frequências de outros serviços de mesma faixa na mesma região de funcionamento. Ainda assim, o bluetooth foi concebido já prevendo que sua utilização viesse a ser em tais ambientes contendo muitos sinais de diferentes origens e operando em faixas de frequência muito próximas. A tecnologia de comunicação bluetooth é mais robusta que as outras tecnologias sem fio, devido à maneira como funciona (CREATIVE, 2015).

Os dispositivos que se comunicam por este protocolo possuem a capacidade de transmitir sinais que trafegam de forma bidirecional para um chamado transceptor bluetooth, ou seja, tanto podem transmitir quanto podem receber sinais. Por este motivo são chamados de canais *full-duplex* (ALECRIM, 2008). Cada canal, seja de entrada ou saída, funciona à parte, ou seja, um não interfere no outro. Uma das técnicas de modulação envolve o espalhamento espectral *Frequency Hoping Spread Spectrum* (FHSS), onde tem-se 79 bandas ou canais com frequências diferentes, cada uma ocupando uma faixa de 1Mz dentro da faixa ISM, sob uma modulação *Gaussian Frequency Shift Keying* (GFSK) (BRAY, STURMAN, 2001).

Para procurar garantir um mínimo de imunidade a ruído, o bluetooth usa a técnica de modulação *Frequency Hopping - Code Division Multiple Access* (FH-CDMA), desenvolvida originalmente durante a segunda guerra mundial para assegurar maior segurança e robustez no controle de torpedos (MILLER, BISDIKIAN, 2001). O mecanismo FHSS é naturalmente mais complexo quando comparado a uma transmissão monofrequencial. As colisões nos dados transmitidos por diferentes fontes, ao mesmo tempo e na mesma subbanda de 1 MHz podem ser atenuadas pelo uso do FHSS, pois é possível fazer-se com que cada *hopping*, ou salto de frequência, tenha uma sequência de saltos diferentes entre as diversas transmissões naquele meio. Mesmo assim, caso haja uma colisão de dados, tem-se a perda de apenas um pacote de informação que pode ser reenviado em outra frequência de *hopping*. Além disto, pelos receptores conhecerem a sequência de *hopping*, eles são capazes de receber todos os pacotes e reconstruir toda a mensagem, cada um de sua respectiva fonte de sinal. O espectro de frequência do bluetooth é dividido em vários canais (ou subcanais), pelos quais os dados são transmitidos em pequenos pacotes de forma sequencial. Com as rápidas alternâncias entre canais através do *hopping*, que ocorrem depois que a conexão

bluetooth é estabelecida, tem-se uma diminuição das possíveis fontes de interferência de sinal sobre o sinal Bluetooth (CREATIVE, 2015).

A Figura 8 apresenta a situação em que se tem *slots* usados por frequências renovadas a cada 625 microsegundos. Desta forma pode-se calcular que se tenha 1600 saltos de frequência por segundo.

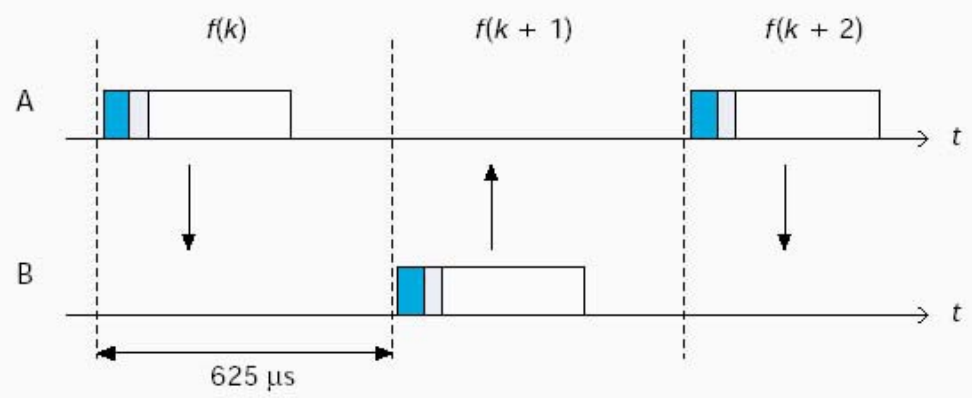

Figura 8 – Canal FH/TDD aplicado ao Bluetooth.
Fonte:http://www.gta.ufrj.br/grad/01_2/bluetooth/bluetooth3.htm.

2.3.2. Topologias de redes bluetooth

Os dispositivos bluetooth podem formar redes entre si. O protocolo permite a existência de redes *piconets* e redes *scatternets*. Na configuração de *piconet* existe um dispositivo mestre que inicia a comunicação com os demais próximos a ele. É ele quem define a maneira, em termos gerais, de como os dados serão enviados aos outros dispositivos que, por sua vez, são os chamados dispositivos escravos (MORIMOTO, 2008). Teoricamente uma *piconet* teria uma limitação de 8 dispositivos conectados, sendo um o mestre e os demais 7, os escravos ao mestre conectados.

O segundo tipo de rede é a chamada de *scatternet*. É uma rede mais espalhada que permite que vários dispositivos escravos se comuniquem entre si, mesmo que pertençam a diferentes redes *piconets* (ALECRIM, 2008). Porém, o dispositivo mestre não tem esta mesma flexibilidade, podendo fazer parte somente da rede a que pertencem. A Figura 9 ilustra estes casos.

Em uma *piconet*, os *slots* de tempo pares permitem transmissão do mestre, enquanto os *slots* de tempo ímpares permitem a transmissão do escravo. De

qualquer forma, as frequências de envio ou recebimento de dados é sempre trocada devido ao sistema de *frequency hopping*. Mais especificadamente, a cada pacote de dados uma frequência diferente dentre as 79 disponiveis é selecionada, formando a sequêcia de *hopping* (CREATIVE, 2015).

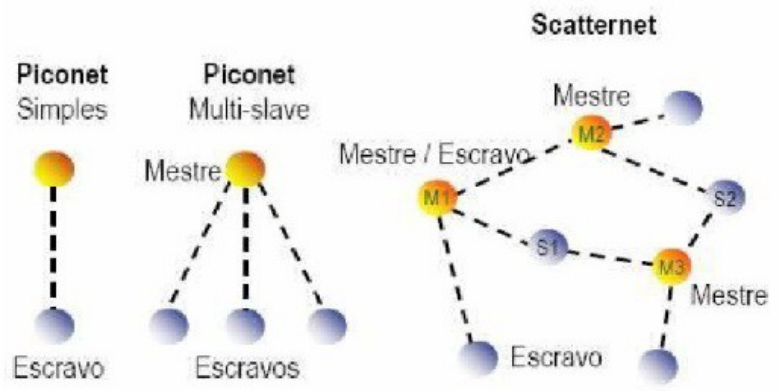

Figura 9 – Topologias de redes bluetooth
Fonte: PRIESS e outros - 2003

Um dispositivo bluetooth pode ser encontrado em um dos estados da Figura 10: em espera, bloqueado, em escuta, página, em solicitação, conectado ou transmitindo (ALECRIM, 2008).

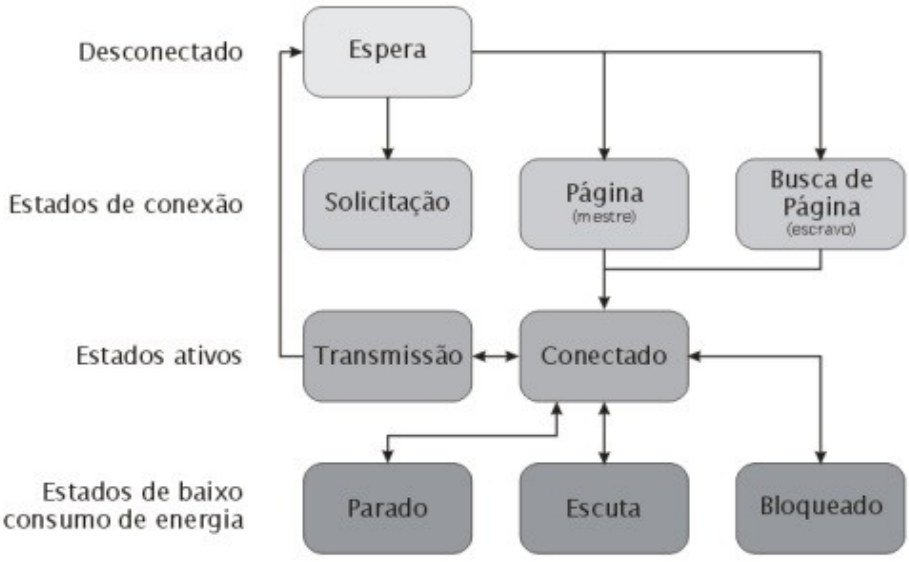

Figura 10 – Diagrama de estados de um dispositivo bluetooth
Fonte:http://www.ic.unicamp.br/~ducatte/mo401/1s2006/T2/057642-T.pdf

Um dispositivo bluetooth inicia em estado de espera ou *stand-by* no momento que é energizado, porém sem ainda estar conectado a nenhuma rede. No momento

em que ele desejar se conectar a alguma rede disponível nas imediações, ele entra em estado de solicitação ou *inquiry*. (BLUETOOTH SIG, 2015).

O sinal *inquiry* é uma mensagem enviada pelo dispositivo que queira buscar os outros dispositivos que estejam na mesma área de alcance para se conectar. Caso algum novo dispositivo tente se conectar em uma rede bluetooth já estabelecida, ele fará uso do sinal *inquiry*. Como resposta ele recebe um pacote chamado *Frequency Hopping Syncronization* (FHS) que fornece a identificação do dispositivo e da sincronização daquela rede. Resumidamente o sinal *page* transmite os pedidos de conexão entre o dispositivos (BLUETOOTH SIG, 2015).

Também existe o sinal *scan* cuja função é fazer com que os dispositivos inativos poupem energia na rede. Esta é uma das vantagens do protocolo bluetooth. Após algum tempo os dispositivos ociosos voltam ao estado ativo a fim de verificar se algum outro dispositivo deseja contactá-lo (BLUETOOTH SIG, 2015).

2.3.3. Versões do padrão bluetooth

Desde que surgiu em 1994 o bluetooth teve melhorias significativas de desempenho. Até hoje conhece-se as versões (ALECRIM, 2008):

- Bluetooth 1.0 e 1.0B: haviam muitas restrições por parte dos fabricantes, principalmente devido à falta de transparência de como um dispositivo se comunicava com outro em uma rede;
- Bluetooth 1.1: é uma primeira versão que teve sucesso do bluetooth. Está definida no padrão IEEE 802.15.1;
- Bluetooth 1.2: compatível com a versão 1.1, tem velocidade de transmissão maior, mais imunidade à interferências, melhor transmissão de aúdio em redes *scatternet*;
- Bluetooth 2.0: velocidade de transmissão aumentada para 3 megabits por segundo. Mais economia de energia;
- Bluetooth 2.1 melhor criptografia, mais economia de energia e o acréscimo de informações ao sinal *inquiry,* o que permitiu melhoramentos nas conexões entre dispositivos;

- Bluetooth 3.0: velocidade de transmissão bastante elevada, chegando a 24 megabits por segundo. Melhoramento no modo de gerenciar energia;
- Bluetooth 4.0: tinha como objetivo a economia de energia. Para isto a sua velocidade de transmissão foi reduzida em relação à versão 3.0, limitada para em torno de 1 megabit por segundo;
- Bluetooth 4.2: é a versão mais recente com novo modelo de segurança de transmissão de dados, usando um algoritmo chamado Curva Elíptica de Diffie Hellman (ECDH) para a geração de chave de segurança. Uma nova maneira para a troca de chaves entre os dispositivos é estabelecida. O bluetooth 4.2 tem a melhor eficiência quanto ao consumo de energia.

2.3.4. Protocolos do bluetooth

O bluetooth possui uma grande variedade de protocolos. Resumidamente existem três grupos em que os protocolos estão divididos: protocolos de aplicação, protocolos de *middleware* e protocolos de transporte (QUEIROZ, 2008).

2.3.4.1. Protocolos de Transporte do Bluetooth

A pilha de protocolos distribuídas entre camadas mais altas e camadas mais baixas é apresentada pela Figura 11. Neste grupo de protocolos faz-se o descobrimento de dispositivos bluetooth propriamente ditos, assim como gerenciam-se ligações físicas e lógicas para os protocolos e aplicações em camadas mais elevadas (CREATIVE, 2015).

A figura 11 apresenta os principais protocolos usados no serviço bluetooth, separados em três arranjos: (a) protocolos de transporte, (b) protocolos de middleware, (c) protocolos de aplicação.

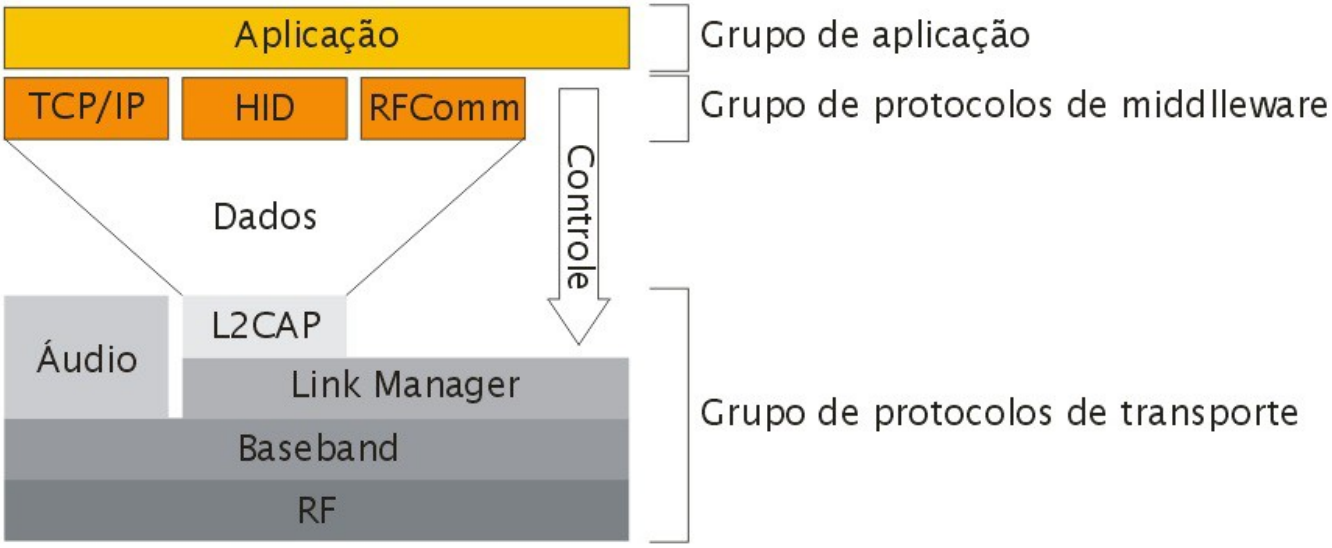

Figura 11 – Pilha de protocolos no Bluetooth
Fonte: http://www.gta.ufrj.br/grad/09_1/versao-final/rssf/padroes_ieee.html

Entre as subcamadas envolvidas no grupo de protocolos de transporte (BLUETOOTH SIG, 2015):

- Camada RF (*Radio Frequency*): Fazendo uma analogia com sistema OSI, esta camada é mais baixa. As frequências das portadoras são controladas pela camada superior. A camada de RF transforma os dados que trafegam para a camada *baseband* nos formatos adequados;
- Camada *Baseband*: aqui especifica-se o controlador de enlace do bluetooth, responsável pelo protocolo de controle e por gerenciar os links síncronos SCO (*Synchronous Connection Oriented*) e os assíncronos ACL (*Asynchronous Connection Oriented*). Esta camada faz parte do *core* do bluetooth, como ilustrado na Figura 12, e pode ser vista como se fosse formada por duas subcamadas:
    - O Controlador do enlace (*link controller*), que se encarrega da codificação e decodificação dos pacotes que vão ou vem da camada de RF, assim como também ajuda na sua sinalização;
    - O *Resource Manager* que é o responsável pelo controle de fluxo de comunicação, dentre os quais os sinais de *acknowlegement* e os pedidos de retransmissão quando necessários.

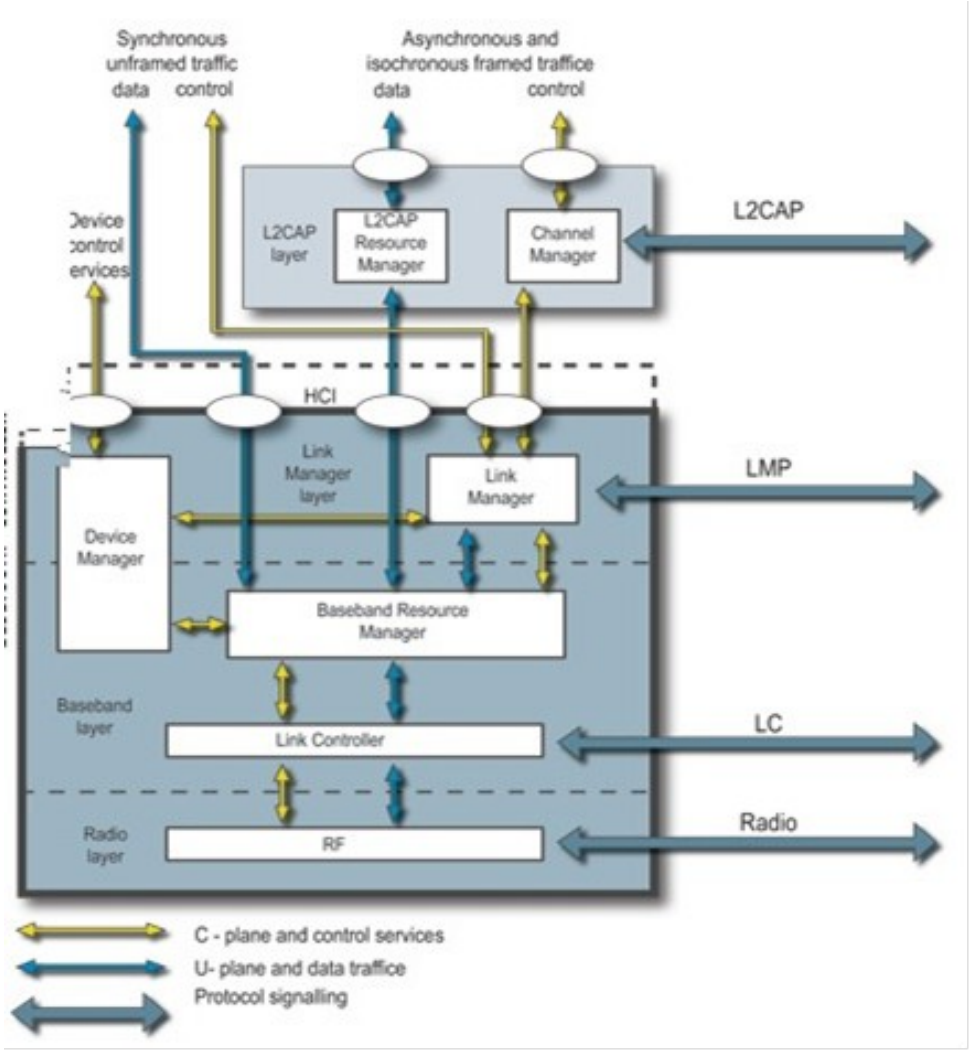

Figura 12 – Arquitetura do núcleo do sistema Bluetooth
Fonte: https://developer.bluetooth.org/TechnologyOverview/Pages/Core.aspx

O *link* SCO é um modo de conexão muito usado para a transmissão de voz.

De outro lado, no modo assíncrono ACL tem-se um link de rádio usado para transmitir pacotes de dados por multiplexação *Time Division Multiple Access* (TDMA) a fim de determinar o meio de acesso. Pela própria definição de TDMA, estabelece-se um *link* com *slots* livres entre um dispositivo mestre e os dispositivos chamados escravos. Indica-se o uso do ACL quando se deseja transmitir informações com garantia de entrega de dados entre dispositivos, pois este modo permite a retransmissão dos pacotes de dados. Sua velocidade de transmissão é de 721 Kbit/s (BLUETOOTH SIG, 2015).

- Camada *Link Manager*: trata-se da camada que gerencia o enlace. Esta camada implementa o *Link Management Protocol* (LMP) para controle do link de rádio entre os dispositivos;

- Camada L2CAP (*Logical Link Control and Adaptation Protocol*): é a camada responsável pela garantia da Qualidade de Serviço (QoS) no lado *host* , além de ser responsável por prover a segmentação, multiplexação, e a remontagem de pacotes;
- Camada HCI (*Host to Controller Interface*): é uma interface responsável pela comunicação entre a pilha do lado *host* com o lado do controlador.

Pelo fato da arquitetura bluetooth poder separar o *Host* do *Controller*, fazendo o intercâmbio de comunicação pelo HCI, o *controller* tem capacidade de armazenamento em buffer menor quando comparada com o *Host*. A finalidade do HCI é tornar possível a operação entre dispositivos e o uso de protocolos de aplicações, sendo feito pelo uso de suas três partes, o *driver* (localizado na parte referente ao *host*), firmware (presente no hardware do controlador) e o módulo de transporte que fica entre ambos (BLUETOOTH SIG, 2015).

2.3.4.2. Protocolos de Middleware do Bluetooth

Há uma seleção de protocolos de padrão industrial entre o grupo de protocolos de *middleware* (CREATIVE, 2015):

- TCS (*Telephony Control protocol Specification*): utilizado para configurar e controlar chamadas de voz e os dados entre dispositivos bluetooth;
- OBEX (*OBject EXchange*): faz a troca de objetos binários entre dispositivos, como por exemplo a que é realizada em dispositivos que usam comunicação infravermelho;
- TCP/IP (*Transmission Control Protocol / Internet Protocol*): permite a comunicação dos dispositivos conectados à internet;
- PPP (*Point-to-Point Protocol*): é um perfil para conexões do tipo ponto-a-ponto;
- SDP (*Service Discovery Protocol*): o SDP é usado para permitir aos dispositivos descobrirem quais serviços são suportados;
- RFCOMM (*Radio Frequency Communication*): é um conjunto de protocolos de transporte sobre o L2CAP que simulam portas seriais.

Transmite um fluxo de dados byte à byte, sem formatação, como se fosse uma transmissão serial assíncrona entre computadores Pode-se efetuar até 60 conexões simultâneas para um mesmo dispositivo bluetooth de cada vez.

O bluetooth tem suporte para TCP/IP de modo a ser utilizado sobre o L2CAP, além da disponibilidade de perfil parao uso do PPP sobre o RFCOMM em aplicativos.

2.3.4.3. Protocolos de Aplicação do Bluetooth

Neste grupo tem-se os aplicativos desenvolvidos que fazem uso dos *link*s bluetooth.

2.3.5. Bluetooth no android

A plataforma android tem suporte ao protocolo bluetooth para envio de dados para outros dispositivos que possuam essa tecnologia embarcada. Tem-se esta funcionalidade através da Bluetooth API, onde são permitidas conexões ponto-a-ponto ou multiponto entre os dispositivos. Através desta API pode-se fazer com que o aplicativo android execute tarefas desde a verificação dos dispositivos próximos, assim como estabelecer canais de comunicação através do protocolo RFCOMM e transferir dados em modo *full-duplex*. Múltiplas conexões também são permitidas (ANDROID, 2015).

Quatro etapas são necessárias para implementar essas interfaces em um aplicativo android:

- Configurar o bluetooth;
- Verificar os dispositivos ao alcance, pareados e os disponíveis;
- Conexão dos dispositivos;
- Transferência de dados.

Todas essas etapas estão contidas em um pacote chamado android.bluetooth. Cada uma das classes do pacote exercem funções específicas em uma conexão bluetooth (ANDROID, 2015):

- *BluetoothAdapter*: através dela faz-se a busca por dispositivos bluetooth, consultas aos dispositivos já pareados e possibilita-se a escuta de comunicação provenientes de outros dispositivos;
- *BluetoothDevice*: descreve os dispositivos já pareados ou os encontrados como nome, endereço MAC. De posse do *BluetoothAdapter* e do *BluetoothDevice,* inicia-se uma conexão através de *sockets*;
- *BluetoothSocket*: representa a interface entre dois dispositivos para um *socket* bluetooth, permitindo que um aplicativo troque dados entre estes;
- *BluetoothServerSocket*: trata-se de um servidor de *socket* que permanece na escuta para requisições de entrada. O dispositivo solicitante necessita abrir um servidor *socket* para que seja retornado um *BluetoothSocket* quando a conexão é aceita.

Para fins de segurança do usuário que utilize um aplicativo, a plataforma android faz que os aplicativos necessitem de permissões para ter acesso a recursos do sistema que se façam necessários. Para o pacote android.bluetooth estas permissões estão declaradas dentro do aplicativo, que somente é executado com a permissão do usuário (ANDROID, 2015):

- android.permission.BLUETOOTH: responsável por solicitar uso das conexões bluetooth e para realizar a comunicação com algum dispositivo (requisição, aceitação, transferência de dados na conexão);
- android.permission.BLUETOOTH_ADMIN: necessária para se executar a busca por dispositivos próximos e por configurar a conexão. Para se usar esta permissão, obrigatoriamente deve-se ter a primeira também autorizada.

A Figura 13 apresenta o exemplo de uma tela de um dispositivo ao solicitar a permissão bluetooth para um aplicativo android. O exemplo ilustra como um aplicativo deve se comportar em um primeiro momento para uso do serviço bluetooth (ANDROID, 2015).

Figura 13 – Caixa de diálogo para habilitar o bluetooth
Fonte: http://developer.android.com/guide/topics/connectivity/bluetooth.html

Existem inúmeras outras permissões possíveis de serem solicitadas, conforme as necessidades do aplicativo. Para todas elas será necessário confirmar a permissão de uso de cada parte do sistema. Como exemplo, tem-se as permissões de acesso a dados via internet, aos contatos telefônicos, às fotos armazenadas no dispositivo, entre outras.

2.4. MICROCONTROLADOR

Uma possível definição para microcontroladores é a de computadores de baixo custo em uma pastilha de silício. Para este trabalho a questão de baixo custo não é critério de escolha, pois em geral todos os microcontroladores da família descrita a seguir são compatíveis com o gasto para uma proposta de trabalho como esta. A questão do baixo custo diz respeito a eles serem mais baratos do que um microprocessador da família Intel ou da AMD, por exemplo, que equipam laptops e desktops.

Para que microcontroladores possam ser utilizados, alguns periféricos devem estar incluídos, tais como portas de entrada e saída, transmissores seriais síncronos e/ou assíncronos, conversores analógico-digitais, entre outros. Deste modo, faz-se principalmente economia de espaço na construção do circuito microcontrolado como um todo, pois todos os circuitos periféricos necessários já estão na própria pastilha do microcontrolador.

2.4.1. Microcontrolador AVR da Atmel

A escolha de uma família de microcontroladores se dá em função de várias características. Dentre elas está o fato de ser acessível em termos de disponibilidade no comércio varejista e preço para projetos universitários, por exemplo, assim como a existência de todos os periféricos já internamente no microcontrolador ou mesmo da velocidade de *clock*, também chamada de relógio, que o microcontrolador funciona.

Com o passar dos anos as velocidades do relógio ou *clock*, que regem a velocidade com que processadores processam instruções, aumentaram. Deste modo, a existência de circuitos internos ao microcontrolador que reduzam sua velocidade de processamento através de divisores de *clock*, também chamados de *prescallers*, é necessário. Além disto, há o fato de que a quantidade de ciclos de *clock* para a execução de cada instrução individualmente é menor. Por isto, muitos microcontroladores hoje em dia funcionam de maneira mais eficiente por assim dizer em relação aos processadores mais antigos que precisavam de muitos ciclos de

clock para executar uma única instrução. Seu esquema simplificado está na Figura 14.

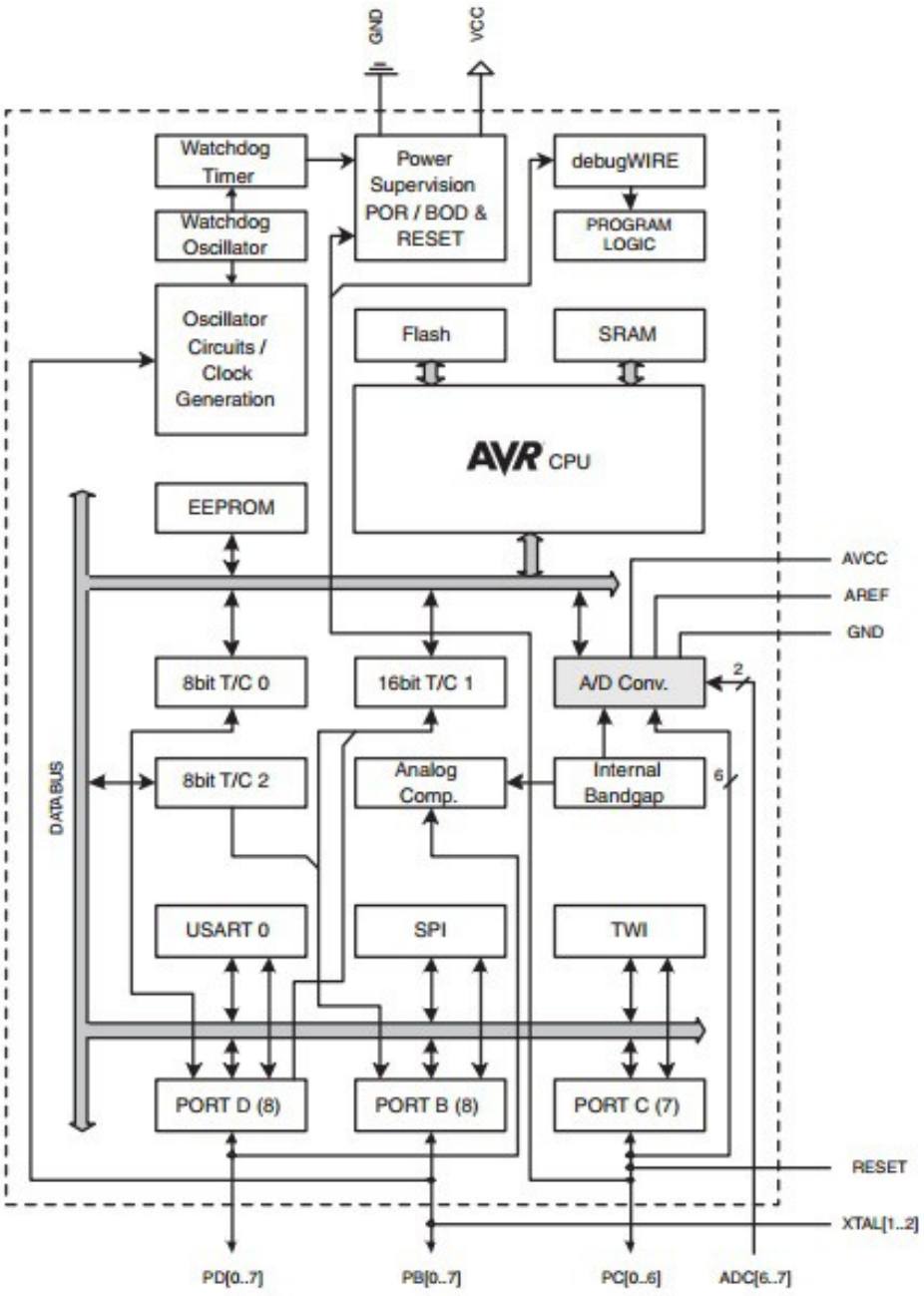

Figura 14 – Diagrama Simplificado de um Microcontrolador Atmega.
Fonte: modificado de (ATMEL, 2016)

Existem várias famílias de microcontroladores, produzidos por diversas companhias diferentes. Alguns fatores como a facilidade de se obter um processador ou outro, disponibilidade de ambientes de desenvolvimento, preço dos componentes e existência de bibliotecas de software disponíveis gratuitamente é que definem qual família de processadores a se usar. Naturalmente a existência de

determinados periféricos, que para algumas aplicações é de fundamental importância, também definem a preferência de um em relação ao outro.

Para o propósito deste trabalho um dos microcontroladores da família AVR foi escolhido. Eles são microcontroladores de 8 bits baseados em arquitetura RISC AVR, que conseguem alta *performance* de até 1 MIPS por MHz, tornando possível o projeto de sistemas de alta *performance* com baixo consumo de energia. Foi desenvolvido em 1996 por dois estudantes do Instituto de Ciência e Tecnologia da Noruega (ATMEL, 2016).

Seguem algumas características que o tornaram apropriado ao projeto:

- Facilidade de aquisição do chip avulso, assim como do chip em placas didáticas para desenvolvimento;
- É uma família de controladores de alta *performance* com baixo consumo de energia;
- Velocidade de processamento de até 32 MHz para alguns membros da família;
- Existência de um ou mais periféricos para transmissão assíncrona (USART);
- Possui 133 instruções, sendo a maioria executada em um único ciclo de maquina;
- Existência de conversor analógico-digital embutido;
- Vários recursos de interrupção interna e externa, como mostrados no capítulo 8;
- Sua memória *flash* interna permite milhares de apagamentos e reescritas de programas, transmitidos via serial a partir de um computador;
- Possui ambiente de desenvolvimento grátis em linguagem C;
- Tem os periféricos necessários para o desenvolvimento deste projeto, e também suporte de software para desenvolvimento de aplicações variadas.

O diagrama da Figura 15 apresenta muito das partes internas do microcontrolador, ilustrando suas interconexões. A grande maioria destas conexões, quando não necessárias ou não usadas podem ser configuradas por software, no início do programa que roda na memória *flash*, de modo a serem desabilitadas.

Conforme a quantidade de cada um dos dispositivos internos necessários, assim como a quantidade de cada tipo de memória, decide-se pelo uso de um ou outro membro da família AVR. Os membros desta família seguem:

- tinyAVR;
- MegaAVR;
- XmegaAVR;
- AutomotiveAVR.

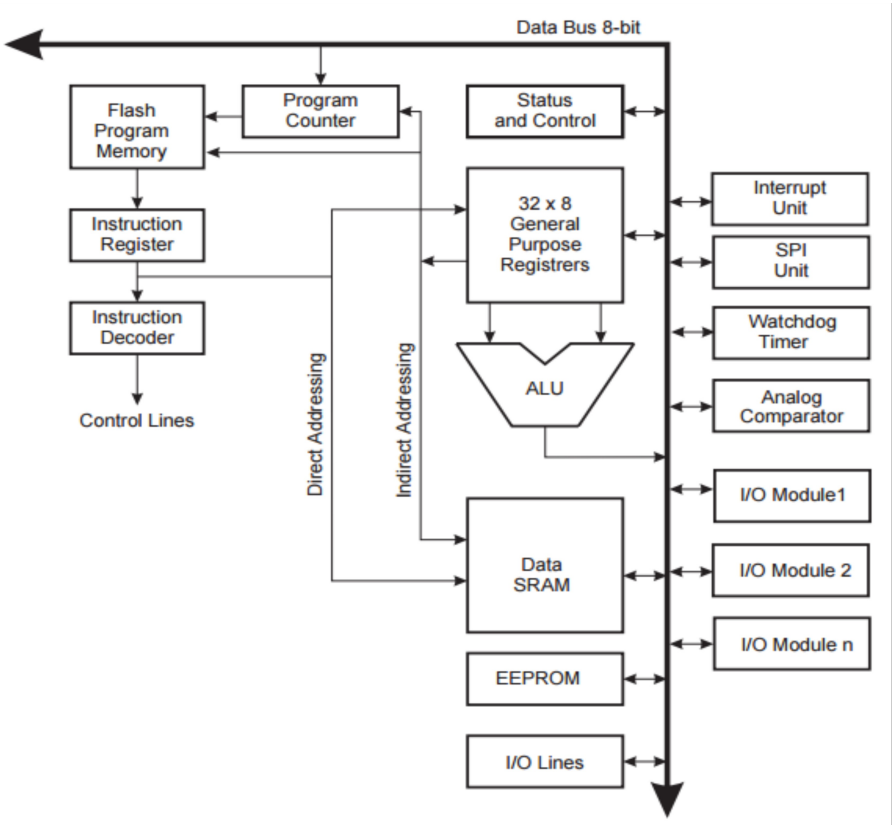

Figura 15 – Diagrama Interno dos microcontroladores com arquitetura AVR
Fonte: atmel.com

2.4.2. Microcontrolador ATmega328P da Atmel

O que varia de uma família para outra são os requisitos de hardware que se fazem necessários para o projeto. Estes requisitos se referem ao número de pinos, presença ou não de USART, SPI ou PWM, dentre outros tipos de interface, a quantidade dos tipos de memória existentes, seja *flash*, ou RAM volátil. Para o

projeto desenvolvido neste trabalho é usado um microcontrolador ATmega328P. Nele tem-se os seguintes elementos de hardware (ATMEL, 2016):

- USART (*Universal Serial Assyncronous Receiver Transmitter*);
- 32 Kbytes de memória *flash*;
- 2 Kbytes de memória RAM;
- 1 Kbyte de memória EEPROM;
- 32 registros de 8 bits;
- 23 pinos programáveis, de entrada e/ou saída, podendo ainda serem configurados para um estado de alta impedância;
- 2 contadores/*timers* de 8 bits, com prescaler individuais (separados), com modo de comparação e de captura;
- 1 contador/*timer* de 16 bits, com prescaler, com modo por comparação e de captura;
- PWM com contador de tempo real com relógio independente;
- *Watch-dog* com oscilador independente;
- Multiplicador por hardware de clocks, gerando 2 ciclos *clock* por ciclo de relógio;
- Conversor ADC de 10 bits;
- Comparador analógico;
- *Clock* até 20 MHz, para o caso do ATmega328.

O diagrama em blocos dos microcontroladores da família AVR ATmega328P está na Figura 16. Nele estão contidos os elementos de hardware presentes em especial na família Atmega328 (ATMEL, 2016).

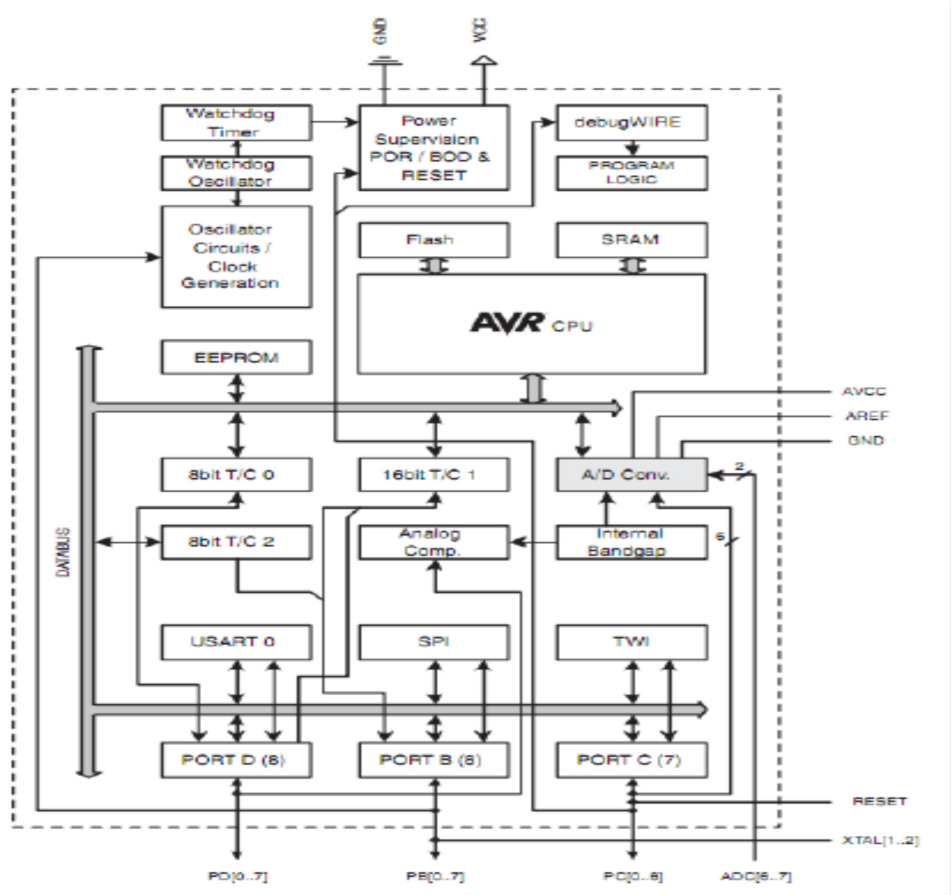

Figura 16 – Arquitetura dos microcontroladores da familia ATmega328
Fonte: atmel.com

A escolha do ATmega328P se deve ao fato da fácil aquisição, disponibilidade de ambiente de desenvolvimento gratuita, além de estar presente em placas na forma de kits didáticos, já com o chamado *BootLoader*. *BootLoader* é um programa que reside na área de *boot* do microcontrolador, que permite que se carregue um programa diretamente pela USB para o microcontrolador. A pinagem externa com seu formato *dual in package* (DIP), com 28 pinos ou também chamado DIP28 facilita seu uso com *protoboards* (ATMEL, 2016).

Nos próximos itens descrevem-se algumas das interfaces, dentre as várias disponíveis, utilizadas para este trabalho.

2.4.2.1. Interface Serial do ATmega328P

O ATmega328P possui uma interface serial, porém com dois registros separados, o que permite que o controlador tanto envie dados através de um pino,

quanto receba informações de modo serial através de outro pino através de uma comunicação *full-duplex*.

2.4.2.2. Modo de Transmissão

A transmissão se dá de maneira mais simples que a recepção, sem o uso de interrupções, bastando-se escrever o dado no registro chamado de UDR. Existe mais uma sequência de bits a serem setados e também verificados para ter-se o dado transmitido. O item 3.3.2.2 apresenta a descrição sucinta deste processo.

2.4.2.3. Modo de Recepção

Uma das maneiras mais simples para receber dados se faz através do monitoramento dos bits de controle da serial, da mesma forma como é feita na transmissão. Porém, para este projeto usou-se uma maneira de receber dados sem que o fluxo do programa fosse interrompido para unicamente aguardar dados recebidos. Usou-se do artifício do uso de chamada de interrupção, ou seja, cada vez que um dado chega e está pronto para ser lido, sua respectiva rotina faz o tratamento de guardar o byte conforme lhe é devido. Maiores informações sobre este processo estão descritas no item 3.3.2.2.

2.4.2.4. Registrador Interno de Contagem

Existem 3 registradores internos que simplesmente contam e podem ser programados para executar alguma tarefa quando algo ocorrer. Como exemplo, usamos um dos contadores de 8 bits no qual introduzimos um valor correspondente a 224. A cada novo ciclo do *clock* interno, este registrador soma 1 a este valor. Quando ele atingir o valor de 256, ele é zerado pois com 8 bits só é possível contar até 255. Isto significa que ele sofreu um *overflow*, o que ocasiona a chamada de uma rotina cada vez que isto ocorre (ATMEL, 2016). O processo serve para gerar a base de tempo de 2 milissegundos para a amostragem do sinal de entrada e também será melhor descrito no item 3.3.2.1.

2.5. INTERNET DAS COISAS

Com o avanço das tecnologias de transmissão sem fio surgiram os serviços chamados de ubíquos, disponibilizando conectividade a qualquer instante em qualquer lugar a dispositivos móveis. Isto levou ao aparecimento de sistemas com intuito de intercomunicar o mundo físico com o mundo virtual (CONTI *et al*., 2012), chamados *cyber-physical systems* (CPS), sistemas integrados de computação e rede (POOVENDRAN, 2010), compreendidos dentro de quatro categorias básicas (PARK *et al*.,2012):

- automação de obtenção de conhecimento;
- internet das coisas – IoT;
- robótica avançada;
- veículos autônomos ou quase autônomos.

Dentro desta gama de aplicações, a internet das coisas tem um valor estimado em 36 trilhões de dolares e é considerada o paradigma dos *cyber-physical systems* (CPS) ou sistemas cibernéticos. Estas aplicações estão ilustradas na Figura 17.

Figura 17 – Cenário Emergente para IoT.
Fonte: Modificado de Borgia (2014).

IoT refere-se a uma maneira de se referenciar objetos através de um tipo de endereçamento, de maneira única, para que possam se comunicar dentro de uma

rede maior. A origem de IoT é atribuída ao centro de Auto-ID do MIT que desenvolveu o sistema de reconhecimento por rádio frequência para qualquer dispositivo que tivesse uma espécie de tarja, chamada RFID. A idéia original era desenvolver uma maneira de se especificar qualquer coisa através de um código implantado no produto pela chipagem com código eletrônico de produto (EPC) (EPCGLOBAL, 2011).

Hoje em dia, contudo, o conceito de coisa no que se refira à internet das coisas já está expandido também para dispositivos, não somente passivos, mas também tablets, dispositivos móveis ou mesmo algo que esteja em movimento. Com isto o conceito se desvinculou do RFID. O cenário com o surgimento de IoT contém variados itens (coisas), com muito heterogeneidade em termos de disponibilidade de recursos computacionais, de coisas, de formas de armazenamento. Isto significa que os serviços de processamento e armazenagem de dados poderão também sofrer uma quebra de paradigma, pois também poderão ser feitos na assim chamada nuvem (CONTI, 2014). Estima-se que até o ano de 2025 cerca de 1000 dispositivos estarão vinculados em cada pessoa, muitos deles chamados *wearables* (BORGIA, 2014). A IoT traz com esta tecnologia muitas vantagens para otimizar situações do dia-a-dia para o individuo, para a comunidade, para o ambiente, com a criação de aplicativos inteligentes que usufruam desta troca de informações, com o devido controle de segurança quanto aos dados dos usuários (GARCIA-MORCHON *et al*., 2013).

Como exemplo de acontecimentos no mundo, recentemente a Google comprou uma empresa que produz termostatos para levar sua marca ao mundo das residências inteligentes (BORGIA, 2014). A IBM também está interessada em soluções inteligentes que envolvam dispositivos e coisas emaranhadas na rede. Estão também muito bem delineados os futuros negócios e estratégias de marketing da Google, Apple, Facebook neste mercado de IoT (BORGIA, 2014).

2.5.1. Visões para um novo paradigma

IoT foi um termo originalmente usado por Kevin Ashton, um dos fundadores no MIT do centro Auto-ID, em uma apresentação na Procter & Gamble (P&G) em 1999. O conceito naquela época era: "uma infraestrutura inteligente que conecta

coisas, pessoas e informações através da rede de computadores", em que a tecnologia RFID seria a base para tudo. Porém em 2005 a ITU - *International Telecommunication Union* levou por primeiro o termo IoT a pesquisadores como um novo significado, como se referindo às tecnologias de comunicação e informação ou *Information and Communication Technologies* (ICT). De lá para cá, houveram inúmeras outras definições para IoT, mas basicamente os autores tem definido IoT em três categorias (BORGIA, 2014):

- Orientada ao Objeto (coisas), onde o foco está no objeto, em como identificá-lo e integrá-lo à rede;
- Orientada à internet, onde o paradigma está em como explorar esta nova rede e seus protocolos de modo a se obter uma boa conectividade entre todos os dispositivos;
- Orientado em Semântica, que se baseia no uso de tecnologias de semântica ao descrever os objetos, ou armazenar e tratar dados, pois o número de objetos a serem interconectados tende somente a aumentar com o tempo.

Em 2009 o *Cluster of European Research Projects* (CERP) propôs sua própria definição de IoT, onde se teria como IoT uma mescla dos diferentes conceitos aplicáveis aos vários componentes técnicos. A definição do CERP foi melhorada para (BORGIA, 2014):

> "Uma rede de infraestrutura global com suas possibilidades baseadas em um padrão de protocolos de comunicação, onde tem-se uma camada física e uma camada virtual de coisas identificáveis com atributos físicos e personalidades virtuais para interface, estando todas integradas em um ambiente de informação."

Esta visão foi recentemente melhorada em (UCKELMANN *et al*., 2011), em que IoT é representada por objetos unicamente identificáveis por seus respectivos representantes virtuais atrelando informações extras destes objetos ao seu status, posição, velocidade ou quaisquer outras informações comerciais, sociais, financeiras, etc, do objeto em questão. Todas as informações do objeto podem ser acessadas a um determinado tempo com um determinado custo.

Além dos objetos mencionados em IoT, outra sua componente fundamental é a comunicação propriamente entre dispositivos, representada pela comunicação

chamada *Machine-to-machine* (M2M). O termo M2M fica aberto à qualquer tipo de tecnologia de comunicação existente, tanto tecnologias wireless quanto de comunicação conectadas por algum tipo de cabo, contanto que permitam a comunicação entre dispositivos. O essencial neste tipo de comunicação é que demande pouca energia, seja de baixo custo, além de precisar de quase nenhuma intervenção de pessoas. O comitê técnico para M2M da ETSI definiu M2M como (BORGIA, 2014):

> "uma comunicação entre duas ou mais entidades que não precisem de intervenção humana".

Para a realização de projetos envolvidos com IoT é inevitável que a infraestrutura da rede seja sempre ampliada. Uma das abordagens muito utilizadas hoje em dia é a chamada silo ou *stove-pipes* devido à abordagem vertical de sistema, ilustrado na Figura 18.

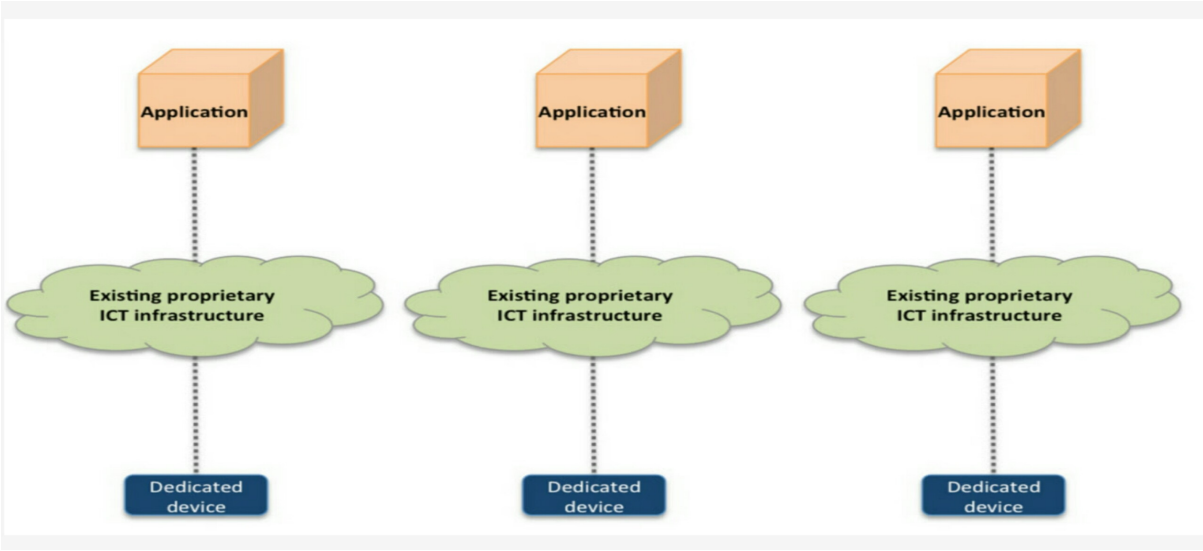

Figura 18 – Representação clássica de aplicativos vistos em silos verticais
Fonte: modificado de Borgia (2014).

Nela cada aplicativo possui sua própria infraestrutura ICT com seus próprios dispositivos. No entanto, esta abordagem vertical deverá com o passar do tempo tender para uma abordagem horizontal onde os aplicativos, serviços, dispositivos de rede e a rede propriamente dita estejam todos distribuídos e disponíveis uns para os outros. A ideia é que os aplicativos não trabalhem mais de forma isolada, mas sim compartilhando rede, serviços e sensores acoplados como mostrado na Figura 19. Para tal precisa-se de uma plataforma operacional em comum que gerencie os

serviços disponíveis e a rede, fornecendo a interconexão entre os diversos tipos de dados para as diversas aplicações (GAMA *et al*., 2012).

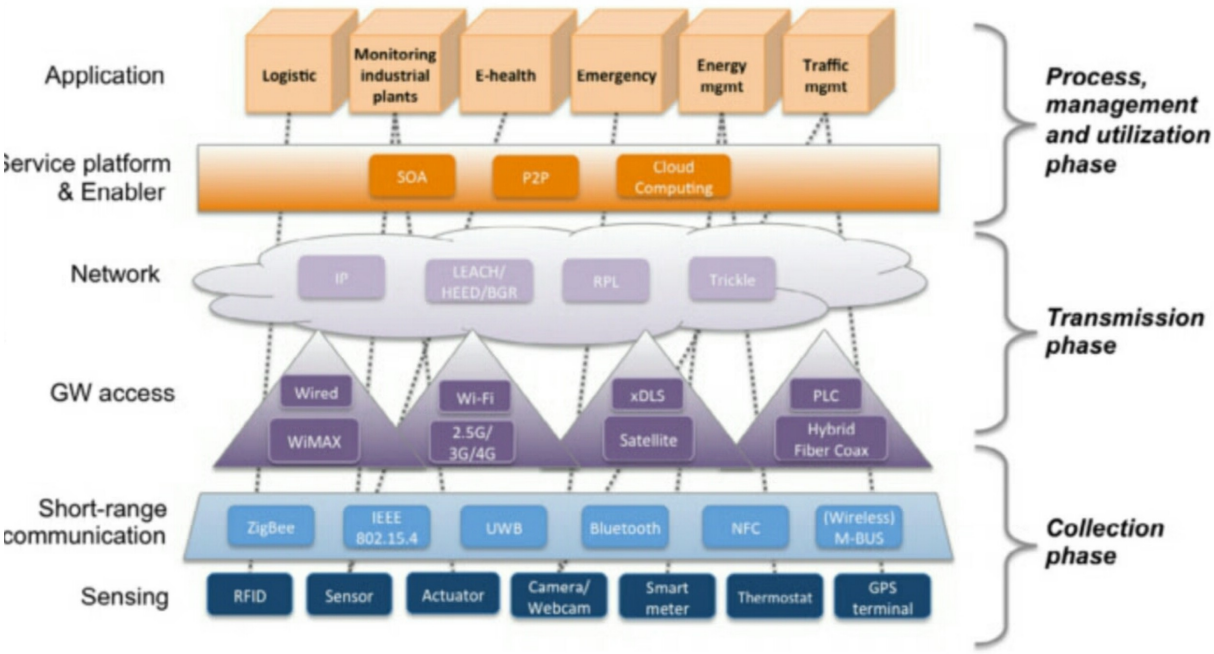

Figura 19 – Representação horizontal para aplicativos que utilizam tecnologias IoT.
Fonte: modificado de Borgia (2014).

2.5.2. Tecnologias para Internet das Coisas

Outra informação disponível na Figura 19 é a representação das camadas que se responsabilizam como plataforma de serviços, que fazem o interfaceamento entre todas as tecnologias e protocolos usados em objetos e nas redes.

Para as tecnologias envolvidas com IoT, consideram-se três fases distintas, descritas do lado direito da Figura 19, responsáveis pelo tratamento do sistema inteligente e como um todo distribuído. São as fases de coleta, transmissão e gerenciamento que correspondem a cada uma das três camadas nas quais estão representados alguns dos protocolos existentes.

2.5.2.1. Fase de coleta

A fase de coleta refere-se à camada dos procedimentos de sensoriamento do ambiente físico, coletando dados em tempo real provenientes dos diversos tipos de

sensores de temperatura, glicose, luminosidade, peso, posicionamento geográfico, batimento cardíaco, frequência cardíaca, sinal EEG, etc.. Estes sinais são então preprocessados localmente e enviados para algum aplicativo nas proximidades pelo uso de variados tipos de transmissão sem fio como bluetooth, zig-bee, WiFi ou qualquer outro, mesmo que proprietário (BORGIA, 2014).

Desta forma surge o conceito de Redes de Sensoreamento Sem Fio, ou *Wireless Sensor Networks* (WSNs), que precisam apresentar robustez suficientes para trabalhar em diversos ambientes (AKYILDIZ, VURAN, 2010). Estas redes consistem de vários nós funcionando como concentradores ou hubs para o sensoriamento em determinada área de certo tipo de sensor com sua respectiva grandeza física, seja temperatura, umidade, glicose, pulso cardíaco. Os sensores, por sua vez, podem precisar de pequenas fontes de alimentação no formato de pequenas baterias, como também podem se alimentar de energia proveniente do bombardeamento por feixe de microondas, como nas etiquetas com tecnologia RFID. Normalmente os sensores não dispõem de muita energia ou memória interna, por isto tem surgido o conceito dos assim chamados elementos móveis ou *Mobile Elementos* (ME). Os MEs são elementos móveis pela rede que nem sempre estão em contato com todos os sensores simultaneamente. Quando quer que eles se aproximem de um ou de outro sensor, coletam os dados dos mesmos (DI FRANCESCO *et al*., 2011). A idéia dos MEs está na otimização do consumo de energia dos sistemas de sensoriamento bem como no uso racional do espectro de frequências para os sensores em áreas pequenas (CONTI *et al*., 2011), de modo a se ter uma distribuição mais uniforme de energia entre os sensores utilizados, fazendo com que a energia deles dure um tempo maior (ANASTASI *et al*., 2009).

As tecnologias de uso para a transmissão de dados durante a coleta podem ser a norma IEEE 802.15.1 correspondente ao bluetooth, ZigBee, NFC ou qualquer outra. Normalmente os métodos de transmissão envolvem protocolos de baixíssima potência, na casa de 2,4 a 2,5 GHz com uma transmissão de dados de algumas centenas de quilobits por segundo. Os protocolos que prevalecem nas WSNs são o bluetooth, zigBee, além do *Wireless Highway Addressable Remote Transducer Protocol* (HART).

Muitas pesquisas começaram a ser feitas desde a última década no que tange à transmissão dos dados vindos dos sensores (DEMIRKOL *et al*., 2006). Para tal, foram estudadas várias maneiras de se endereçar sensores pelos seus MACs,

ou através de protocolos de roteamento (AKKAYA, YOUNIS, 2005) ou de transporte. Foram feitos muitos estudos na transmissão em ambientes hostis à propagação das OEMs como os subaquáticos (DOMINGO, VURAN, 2012) e subterrâneos (AKYILDIZ, STUNTEBECK, 2006). O que é de importância para IoT são as redes de transmissão de baixa potência em áreas chamadas pessoais, ou *Low Power Wireless Personal Area Networks* (LoWPAN) (WANG *et al*., 2012) como por exemplo, as redes bluetooth 4.0 que são de baixa potência ou, como outro exemplo, as redes NFC. As redes NFC operam a 13,56 MHz, ISO/IEC 18000-3 a pouco mais de 400 kbps, apenas com a aproximação dos dispositivos entre si a menos de 4 cm um do outro (BORGIA, 2014). Isto já é o suficiente para que um dispositivo se comunique com o outro.

No entanto, os dispositivos bluetooth se disseminaram pela indústria e transmitem usando protocolo serial de dados com baixas potências de transmissão (BRUNO *et al*., 2002). Os dispositivos bluetooth formam as redes *piconet*, na qual um dispositivo assume o papel de mestre, enquanto os demais dispositivos assumem papel de escravos. As taxas atuais de transmissão vão até 24 Mbits por segundo.

2.5.2.2. Fase de transmissão

Depois que se disponibilizam os dados coletados para as aplicações que as demandem, é necessário transmitir estes dados para algum lugar ou servidor que as armazene para que outros aplicativos possam fazer uso de toda uma base de informação. A maneira como esta transmissão é feita pode ser cabeada ou sem fios. Apesar da rede cabeada, tendo como exemplo o padrão ethernet IEEE 802.3, ser bastante confiável, está-se tornando cada vez mais comum o uso de redes sem fio, como as redes chamadas WLAN. Várias são as tecnologias existentes para tais tipos de transmissão, como as que usam os protocolos WiFi, IEEE802.11 a/b/g/n/ac que operam desde frequências de 2,4 GHz até frequências de 5 GHz. Outra maneira de se transmitir as informações coletadas na etapa anterior é pelo uso de 3G ou 4G disponíveis em muitos aparelhos celulares ou smartphones de hoje. Estas tecnologias têm a vantagem de um alcance muito maior, fazendo hoje proveito dos vários aplicativos já instalados nos smartphones. A questão aqui é deixar disponível

nos dispositivos móveis aplicativos que também façam a coleta das informações da rede WSN de sensoriamento, para estabelecer o que se chama de serviços ubíquos de multimidia ou *Multimedia Ubiquous Service* (CHEN *et al*., 2012), já que a transmissão 3G ou 4G disponíveis em tablets e smartphones possuem alta taxa de transmissão de dados, estando na cada de alguns megabits por segundo.

As tecnologias sem fio que existem e que venham a surgir é que fazem a grande quebra de paradigma para a IoT. A única questão técnica importante é o gerenciamento delas quanto ao uso e/ou disponibilidade do espectro. Vários exemplos de redes, como WiFi, WiMAX, redes xDSL, redes por satélite, redes de comunicação pelas linhas de distribuição elétricas chamadas de *Power Line Communications* (PLC) (GALLI *et al*., 2010) estão ilustradas na Tabela 3.

Tabela 3 – Características das principais redes de acesso para IoT
Fonte: Modificado de Borgia (2014).

Technology	Reference standard	Transmission medium	Frequency bands	Data rate	Maximum distance	Limitations
Ethernet	IEEE 802.3 u/z	Twisted-pair copper wire, coaxial cable, optical fiber	–	10 Mbps, up to 100 Gbps	100 m, up to 50–70 km	Shared medium, physical connection among devices
WiFi	IEEE 802.11 a/b/g/n	Wireless	2.4 GHz, 5 GHz	1–54–600 Mbps	up to 100 m	Sensitive to the presence of household appliances, interference among WiFi communications
WiMAX	IEEE 802.16 a/d/e/m	Wireless	2–66 GHz	up to 70 Mbps	Up to 50–80 km	Low practical data rate, sensitive to weather conditions, high installation and operational costs
xDLS	ADSL, ADSL 2+, VDSL	Twisted-pair copper wire, coaxial cable	Up to 2.2 MHz	12–55 Mbps (d) 1–20 Mbps (u)	5.4–1.3 km	Asymmetrical communication
Cellular	GSM, GPRS, UMTS, HSPA+, LTE	Wireless	900–1800 MHz 2100–1900 MHz 800–2600 MHz	9.6 kbps, 56–114 kbps, 56 Mbps (d)/22 Mbps (u), 300 Mbps (d)/75 Mbps (u)	Macro/micro/pico/ femto cells (10 m to 100 km)	Limited wireless spectrum
Satellite	BSM, DVB-S, DVB-TS	Wireless	4–8 GHz (C band), 10–18 GHz (Ku band), 18–31 GHz (Ka band)	16 kbps to 155 Mbps	GEO sat.: 35,786 km MEO sat.: 500–15,000 km LEO sat.: 200–3000 km	280 ms propagation delay, huge launching cost, almost impossible repairing
PLC	HomePlug AV, IEEE 1901	Electrical power system	1–30 MHz	>100 Mbps	Up to 1500 m to the premise, up to 100 m between devices	Mutual interference with other technologies

Da mesma forma como ocorre na fase de coleta, a fase de transmissão também procura fazer uso das chamadas redes oportunísticas (BOLDRINI *et al*., 2014). Junto com as redes WSNs, elas pertencem à classe de redes móveis multi-*hop ad-doc* (CONTI, GIORDANO, 2014). Estas redes incluem outros conceitos de rede que as tornam auto-organizáveis, muito úteis aos novos paradigmas de IoT. Um exemplo são as redes sem fio MESH (BRUNO *et al*., 2005) ou redes veiculares (HARTENSTEIN LABERTEAUX, 2008) que ainda estão em desenvolvimento (WANG, LIN, 2013).

2.5.2.3. Fase de gerenciamento e utilização de dados

A fase de gerenciamento de rede e a utilização de dados lidam com os processos que analisam toda a informação coletada e transmitida, fazendo com que as muitas diferenças de hardware ou protocolos usados em camadas anteriores se tornem invisíveis nesta etapa. Esta fase é responsável por toda uma abstração das características dos objetos sensoreados e posteriormente transmitidos, podendo também fornecer uma retroalimentação de controle aos aplicativos que fornecem os dados. Tudo que se relacione ao tratamento dos dados, desde análise semântica, filtragem ou outro tipo de tratamento que se faça necessário aos dados é aqui tratado.

Para que estes desafios sejam vencidos, usa-se um conceito aplicado à IoT chamado Arquitetura Orientada a Serviço ou *Service-Oriented Architecture* (SOA), usado em diferentes programas para assim conectar programas que rodam em computadores normais. A idéia básica do SOA está sedimentada em três camadas (GAMA *et al*., 2012):

- A primeira camada sendo responsável pela abstração de objetos, onde um objeto ou uma de suas funcionalidades é abstraída por um *service*, com respectivos métodos para ter acesso às abstrações;
- A segunda camada faz o gerenciamento dos objetos e dos *services*, tornando possível, de alguma maneira, que eles sejam descobertos pelo sistema ou que seus resultados ou status sejam acessados e/ou monitorados. É possível através de funcionalidades extras realizar um

gerenciamento remoto de um objeto ou, por exemplo, a de se saber todas os serviços disponíveis para determinado objeto;

- A terceira camada fornece mecanismos para se manipular objetos e serviços. Como exemplo, pode-se reagrupar um conjunto de objetos criando outro objeto. Outro exemplo é configurar o sistema de modo a se ter certeza de se estar mostrando o status atualizado de um ou mais objetos em uma tela de smartphone.

Para que estas funções todas de IoT sejam bem desempenhadas, torna-se bastante apropriado o conceito de nuvem, onde dados possam ser armazenados, assim como recursos de hardware podem ser alocados dinamicamente conforme as necessidades de um cliente, porém sem que haja intervenção humana, ou seja, como se tudo estivesse automatizado. A isto IoT se refere tanto à necessidade de configuração de hardware apropriada para determinada aplicação, cuja memória ou velocidade de processador podem ser alocados conforme necessidade de espaço ou de poder de processamento, quanto ao fato de se aplicar IoT ao processamento de informações diversas nos ramos da indústria, agricultura, vigilância doméstica ou sistemas de *HealthCare* (PROJECT, 2016).

Algumas destas aplicações estão na Figura 20.

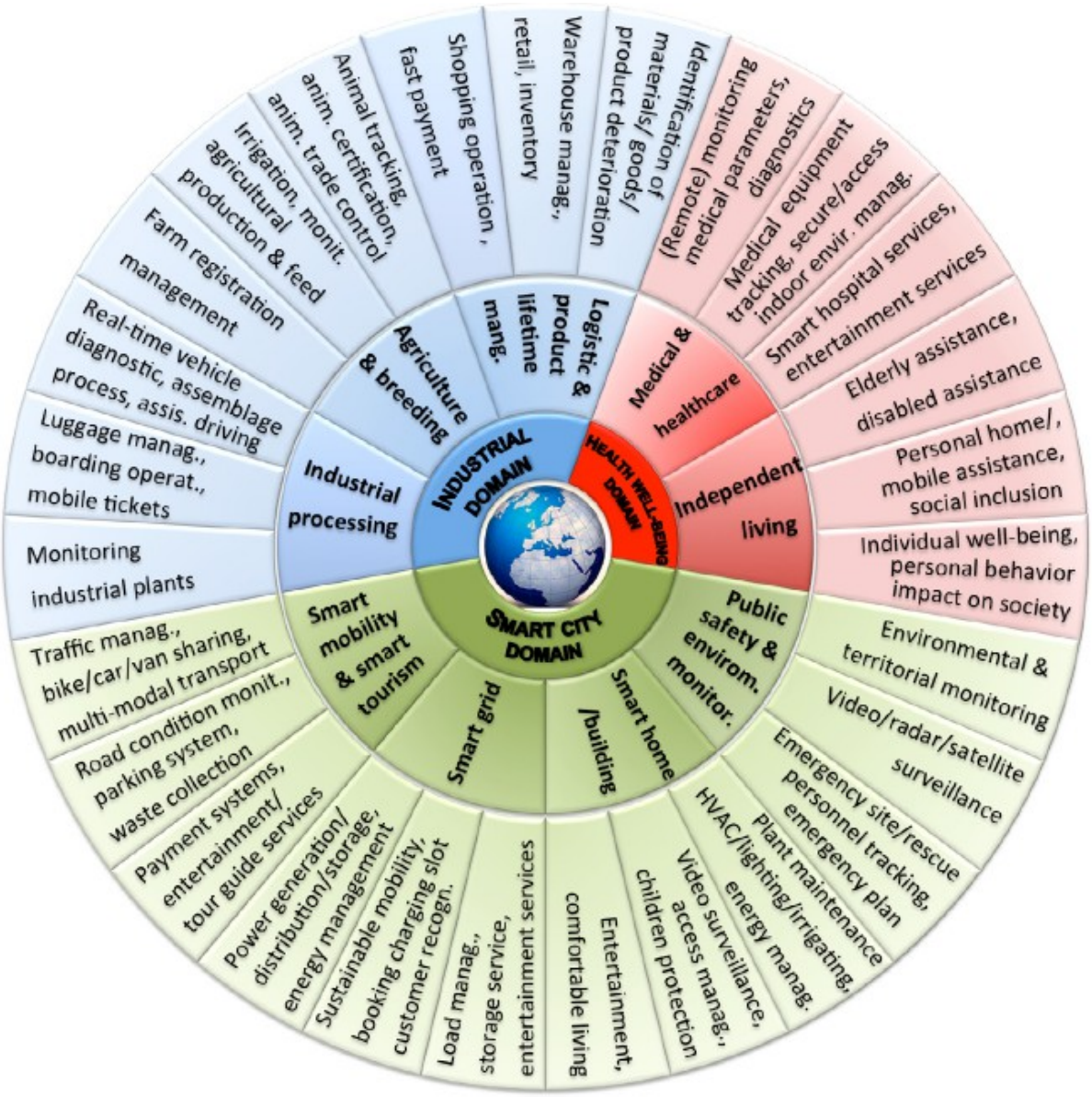

Figura 20 – Aplicações para IoT
Fonte: Modificado de Borgia (2014).

2.6. INTERFACEAMENTO COM A NUVEM

Com o advento das assim chamadas nuvens, várias empresas começaram a fornecer serviços de armazenagem de dados em sites externos. São empresas como a IBM, a Parse, a Google, entre outras, que prestam serviço de armazenagem de dados em seus servidores, garantindo segurança para as aplicações feitas pelos usuários, assim como para os dados que trafegam entre os aplicativos e os servidores.

Porém não é sempre que se tem dados e/ou aplicativos usando serviços em nuvem de maneira tão simples. Existem questões relacionadas à segurança do processamento e à integridade dos dados armazenados

2.6.1. Questões relacionadas à nuvem

Para que se possam introduzir os conceitos de segurança em nuvem é necessário primeiramente compreender as diferenças entre computação em nuvem e *mobile cloud computing* (MCC). Como foi apresentado no capítulo 1, a grande motivação para que os aplicativos aqui desenvolvidos fossem feitos se deve ao fato dos dispositivos móveis terem tido um enorme avanço em termos de poder de processamento, aliado à integração de muitas interfaces sem fio como as de WIFI, bluetooth ou NFC, por exemplo. Com isto, a esta integração dos dispositivos móveis existentes hoje em dia junto com o armazenamento de dados em servidores externos, aqui chamado como computação em nuvem, fez surgir o termo MCC. O termo MCC na realidade atual é, inclusive, usado para o processamento dos dados em nuvem, não se restringindo ao seu simples armazenamento. Este trabalho, no entanto, se restringe à manipulação de dados proprietários, gerados a partir de um sinal de ECG pertencente a um paciente, que foi então armazenado em nuvem para uso posterior (VINH *et al*., 2015).

Computação em nuvem é um novo paradigma para serviços escaláveis para serem usados através da internet, sendo basicamente formados por serviços sob demanda quando o usuário solicita, e apresentam cinco características que os diferenciam de outros serviços emergentes (KHAN *et al*., 2015):

- *Multi-tenancy*: que diz respeito ao fato dos recursos poderem ser compartilhados;
- *Elasticity*: relacionado à propriedade de aumentar ou diminuir os recursos computacionais conforme o número de usuários;
- *Scalability*: é a propriedade do sistema em nuvem poder ampliar a escala para milhares de servidores;
- *Pay-per-use*: trata do fato de se pagar apenas pela quantidade de recursos necessários;
- *Self-provision* ou autoprovisionamento de recursos.

Também pode-se diferenciar entre diferentes tipos de serviços oferecidos em nuvem:

- *Software as a Service* (SaaS) – Neste modelo tem-se um aplicativo específico sendo oferecida pela nuvem ao usuário;
- *Platform as a Service* (PaaS) – Modo pelo qual o usuário desenvolve um aplicativo sobre um ambiente de desenvolvimento fornecido pelo provedor de serviços, ou nuvem;
- *Infrastructure as a Service* (IaaS) – Modo através do qual o provedor de serviços em nuvem fornece toda a infraestrutura ao usuário desenvolver seus aplicativos, permitindo inclusive executá-los em nuvem.

2.6.2. Segurança na nuvem

Com o passar dos últimos anos, muitas empresas mudaram seus aplicativos e seus bancos de dados para plataformas em nuvem (KANMANI, ANUSHA, 2015), que tem entre suas maiores prioridades o cuidado com a segurança dos dados armazenados. Seguem algumas características quanto ao armazenamento de dados:

- Confiabilidade: é a garantia que os dados permaneçam inalterados, a não ser por usuários autorizados;
- Integridade: tenta obter a prova de que os dados se mantém protegidos de modificações maliciosas ou inadvertidas;

- Autenticação: é o processo pelo qual se evita que usuários não autorizados acessem o sistema;
- *Non-repudiation*: relacionado ao estabelecimento de uma comunicação dupla entre o *sender* e o *receiver*. É quem garante que as mensagens sejam corretamente enviadas ao receptor e verifica se o *receiver* deu *acknowledge* ou não para a mensagem;
- Disponibilidade: é a garantia que os recursos computacionais estejam disponíveis para o acesso de usuários.

2.6.3. Autenticidade dos dados em nuvem

O projeto aqui envolvido necessita armazenar dados do sinal de ECG em nuvem para recebê-lo simultaneamente por outro aplicativo conectado através da internet. Para isto, dentre os vários problemas existentes no armazenamento de dados em servidores remotos, aqui enfrenta-se o mesmo problema que muitos outros aplicativos desenvolvidos para os mais diversos fins enfrentam: segurança e integridade dos dados armazenados em nuvem.

A este respeito surge uma nova definição: *Provable Data Possession* (PDP), que trata da garantia de que os dados armazenados em nuvem estejam lá corretamente armazenados (ATENIESE *et al*., 2007). Para garantir que os dados armazenados em nuvem estejam corretamente guardados, uma forma de PDP é usada, pois é possível que haja perda ou alteração de dados por algum artifício malicioso rodando em nuvem. A fim de se verificar a integridade dos dados, alguns métodos foram desenvolvidos (ATENIESE *et al*., 2007):

- Método *Naive* (Ingênuo): Neste método, quando o cliente quiser saber sobre a integridade do arquivo em nuvem, um par (chave K com seu valor *hash*) é liberado. A chave K é enviada ao servidor que recalcula o valor *hash*. O resultado é comparado com o original para saber se o arquivo é original. O problema neste método é o tempo demandado para o cálculo sobre o arquivo inteiro;
- PDP original: Neste método os dados a serem guardados em nuvem são preprocessados gerando um *tag-value* ou metadado para verificação no lado do cliente. Quando se precisar verificar os dados, a

informação desafiante é enviada para obter o metadado. O cliente então compara o metadado de resposta com o metadado local, verificando sua integridade;

- *Proof of Retrievability* (Prova de recuperabilidade): Este foi um novo médodo (JUELS, KALISKI JR., 2007) que ao contrário o método *Naive*, tem-se apenas uma chave de criptografia.

2.6.4. Interface com a nuvem Parse

Com o advento dos servidores em nuvens, várias empresas começaram a fornecer serviços de armazenagem de dados em sites externos. São empresas como a IBM, a Parse.com, a Google, entre outras, que prestam serviço de armazenagem de dados, garantindo segurança às aplicações feitas pelos usuários e ao dos dados que trafegam entre os aplicativos e os servidores externos. Isto se faz pelo uso de bibliotecas fornecidas ao programador pela empresa da nuvem.

A Parse é uma das empresas fornecedoras deste tipo de serviço. Para fazer uso do serviço, basta cadastrar-se como usuário, que pode ser gratuito ou não conforme a quantidade de tráfego que se deseje usar no Parse. Visitando-se o site parse.com e clicando-se em *products* pode-se ver algumas das características de programação que são oferecidas às aplicações. Parse trata-se basicamente de um servidor *backend* como uma prestação de serviço. A Figura 21 mostra a página inicial.

Isto significa que a Parse trata das componentes de rede para as aplicações, cuidando de tudo que é necessário para armazenamento de dados na nuvem, passando por notificações do tipo *push*, integração com o *framework* do facebook ou twitter, quando assim se desejar, além do que se costuma chamar de *cloudcode* ou código nas nuvens que é uma maneira de se colocar a lógica do negócio, ou business logic, como se fosse uma camada para interfacear os dados na nuvem e a sua aplicação.

Para este trabalho usou-se apenas os produtos relativos à armazenagem de dados nos servidores Parse. Para isto fez-se o cadastro através de conta pessoal de email e em seguida pode-se cadastrar os mais variados aplicativos, seja para

Android, IOS, Java, entre outras plataformas de desenvolvimento dentro da sua conta pessoal, conforme ilustrado na Figura 22.

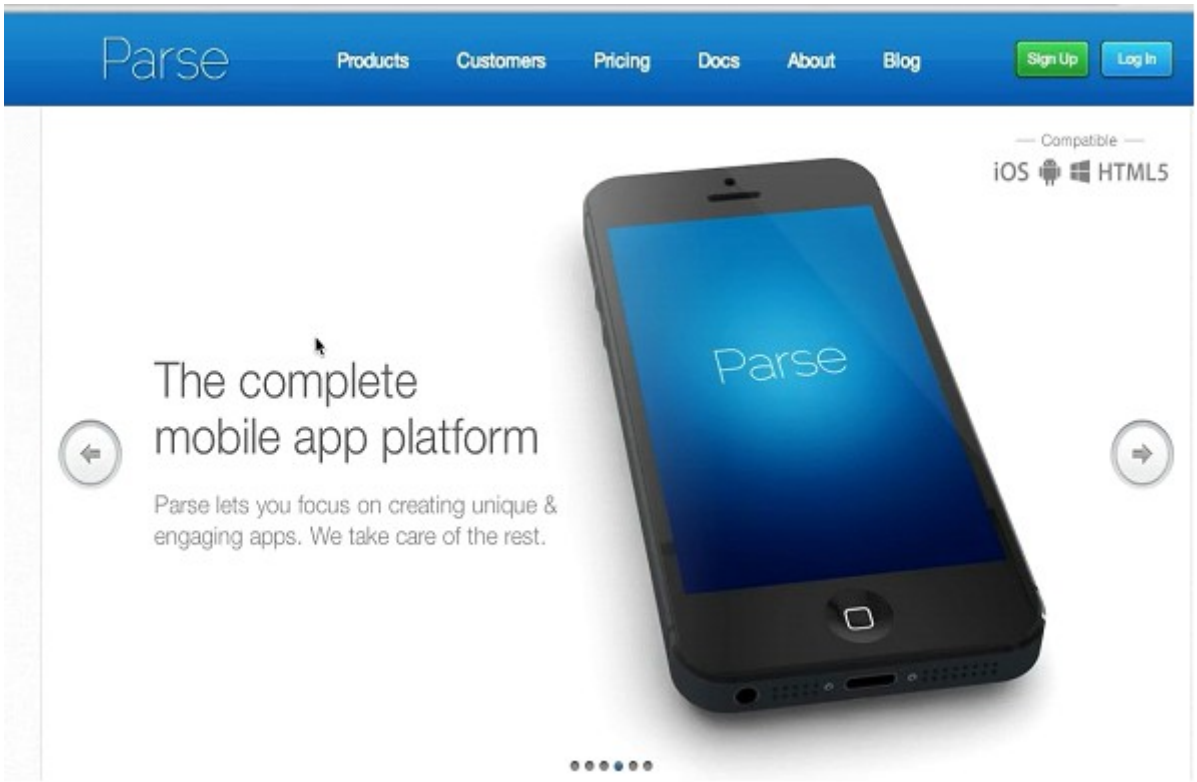

Figura 21 – Página principal de parse.com
Fonte: parse.com

Figura 22 – Plataformas para Parse
Fonte: parse.com

Tem-se as mais variadas plataformas para as quais existem bibliotecas disponíveis para o interfaceamento com Parse. Neste trabalho escolheu-se a plataforma android devido a ser aberta além de estar presente na maioria dos sistemas móveis pessoais. Clicando-se no ícone android da Figura 22, abre-se uma tela como mostra a Figura 23.

Figura 23 – Escolha da Plataforma Parse
Fonte: parse.com

A Figura 23 deixa claro que Parse fornece bibliotecas apropriadas para as várias plataformas hoje existentes, de modo que apenas incorporando-as no projeto do aplicativo, pode-se usar suas classes e métodos para se ter acesso aos servidores da Parse para armazenagem de dados na nuvem.

Existem algumas características do parse.com que seriam muito úteis no controle de dados pessoais que um determinado aplicativo utilize:

- Criação de usuários para a nuvem;
- Conexão e desconexão de usuários da nuvem;
- Correlação de dados entre usuários da nuvem: é um serviço que permite que um usuário siga outro usuário como se fosse um aplicativo parecido com twitter;
- Armazenamento de dados de usuários na nuvem;
- Recuperação de dados de usuários da nuvem.

Para começar o uso destes serviços, deve-se começar com o cadastro pessoal do programador no parse.com, conforme ilustrado na Figura 24.

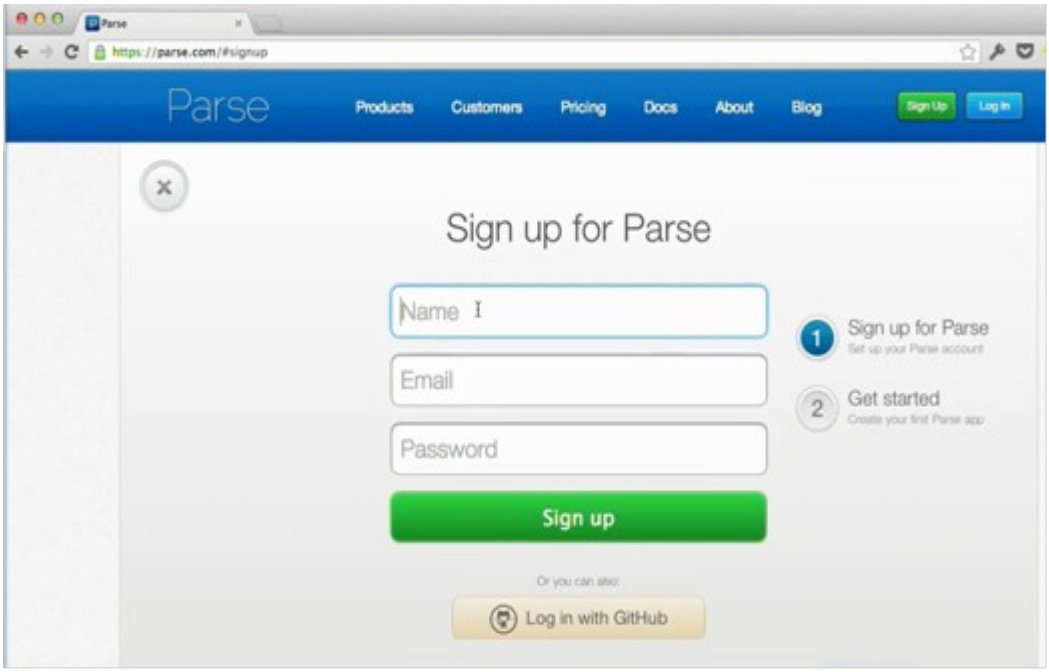

Figura 24 – Tela de Assinatura do serviço na nuvem
Fonte: parse.com

Desta tela segue-se para a Figura 25, onde se clica em *Cloud Store Data*.

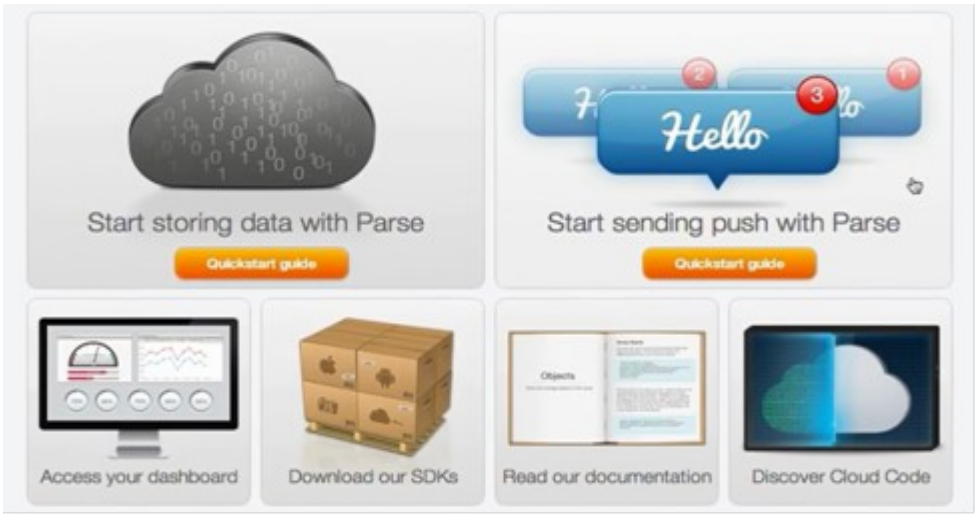

Figura 25 – Tela para Armazenamento de Dados
Fonte: parse.com

Em seguida faz-se o cadastro do aplicativo, conforme apresenta a Figura 26, em desenvolvimento. Pode-se baixar também para um projeto já existente, quando se faria uso apenas das bibliotecas.

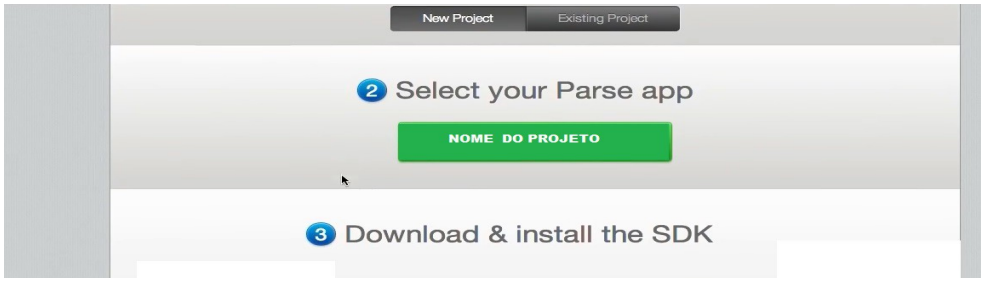

Figura 26 – Tela para escolha do aplicativo a receber o serviço na nuvem
Fonte: parse.com

Logo mais, como mostra a Figura 27, basta baixar o arquivo zipado no qual se obtém uma estrutura básica, esqueleto básico de um aplicativo que use as bibliotecas para armazenamento em parse.com.

Figura 27 – Tela de *download* do kit da nuvem
Fonte: parse.com

O arquivo zipado que foi baixado contém várias coisas, dentre as quais:

- Algumas bibliotecas a serem importadas no projeto, dentro do ambiente de desenvolvimento;
- Um objeto identificador parse;
- Um projeto novo para android.

O projeto chamado *ParseStarterProjetct* deve ser importado para dentro do ambiente eclipse, o ambiente de desenvolvimento android. Ao importar-se, ainda

assim alguns erros aparecerão, pois falta inserir em tal projeto o objeto android que também foi fornecido ao se fazer a inscrição no Parse. Este objeto aparece na Figura 26, item 7, e corresponde à identificação do cliente programador e do aplicativo em questão junto ao Parse. É desta forma que a nuvem sabe quem é o aplicativo que solicita os vários métodos existentes sobre objetos de armazenamento.

Feito isto, basta criar objetos do tipo parse para manipulação de dados. Estes objetos podem ser compostos de conjuntos de pares de valores *key* e *value*. O valor de *key* corresponde a um nome de variável, como o que costuma usar em linguagem C. Value é o valor atribuído à variável *key*. Vários conjuntos de pares *key*, *value* podem ser utilizados em um único objeto tipo parse, de modo que o objeto compreenda uma estrutura complexa de variáveis. Após estes passos, precisa-se armazenar estes dados na nuvem e para isto existem várias maneiras. A forma mais simples é usar-se um método atrelado a objetos parse chamado saveinbackground(). Ele se encarrega de toda a parte relativa aos acessos a servidores, assim como a fazer o armazenamento do objeto criado anteriormente nos servidores Parse.

2.6.5. Segurança oferecida por parse.com

Quando um aplicativo se conecta aos servidores da Parse, duas strings são enviadas: uma com o objetivo de identificar o usuário e a outra identificando o aplicativo. A identificação deste conjunto é feita por um método atrelado a uma classe parse, logo no início do aplicativo. Porém somente estas duas strings não garantem a integridade dos dados, nem o sigilo das informações.

A Parse possui algumas outras características extras que ajudam a garantir o controle de acesso a dados armazenados e ao controle de acesso de usuários. Além disto, todas as conexões são realizadas por objetos criados por classes da biblioteca parse. Para isto, utiliza-se durante a conexão internet estabelecida pelo dispositivo móvel o protocolo HTTPS e SSL. Conexões com servidores Parse que estejam fora destas características são rejeitadas (PARSE, 2015).

O *Hyper Text Transfer Protocol Secure* (HTTPS) põe uma manta de proteção na transmissão de dados entre o servidor e o aplicativo. A comunicação é

criptografada, aumentando a segurança dos dados. Por SSL entende-se *Secure Sockets Layer*, ou seja, uma ferramenta de criptografia que codifica a informação antes dela ser transmitida pela internet ao servidor Parse. O SSL serviu como base para o desenvolvimento do protocolo padronizado pela *Internet Engineering Task* Force (IETF) definido originalmente pelo RFC 2246 de 1999 (IETF, 2016).

Existem garantias extras oferecidas aos objetos criados e gerenciados pelos usuários, pois cada objeto pode conter várias informações referentes ao paciente. Em princípio, os itens de informação relacionados aos pacientes podem ser adicionados ou excluídos, assim pode-se dar autorização aos diversos tipos de usuários do aplicativo que usam serviços de nuvem da Parse para alterá-los. A idéia primordial é que ninguém possa fazer alteração alguma, cabendo somente ao aplicativo enviar os dados referentes aos campos do objeto parse que se referem ao paciente. A Figura 28 ilustra o painel de controle com alguns campos do objeto parse chamado ECG4.

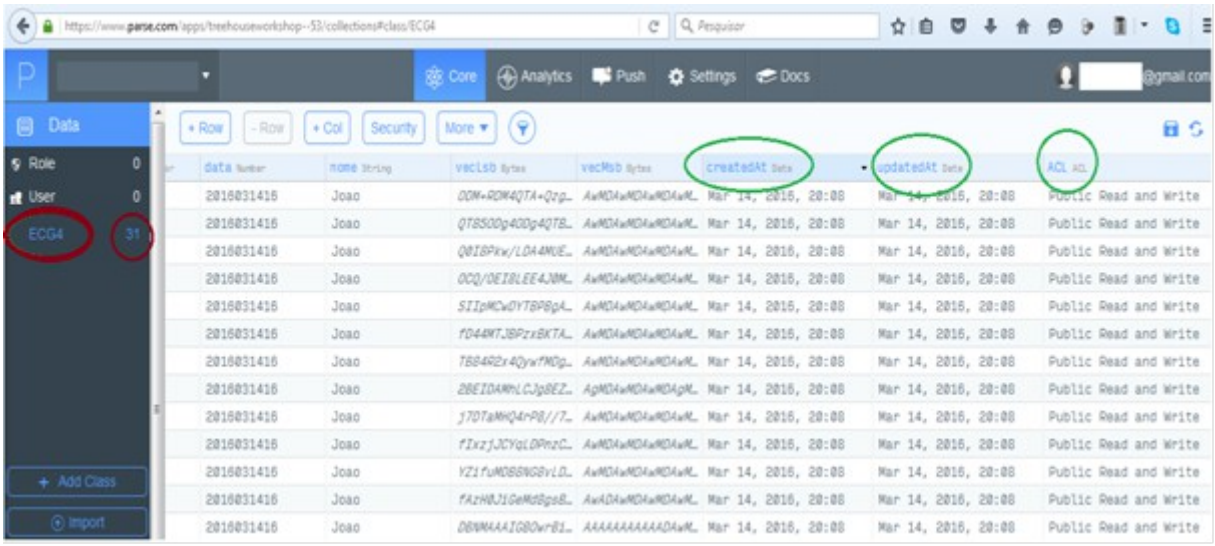

Figura 28 – Exemplo de campos parse
Fonte: Autoria própria

Cada linha da tabela na Figura 28 representa um objeto parse. Cada objeto é gerado a seu tempo. Neste trabalho, cada segundo de sinal de ECG está representado por uma linha representada na Figura 28.

Pode-se notar à direita da Figura 28 três campos circundados em verde. O campo chamado “createdAt” indica o tempo em que foi criado tal objeto no servidor em nuvem. O campo “updatedAt” indica a última atualização do mesmo objeto no

servidor em nuvem. Estes dois campos não são manipuláveis por aplicativos. O campo mais à direita chamado de “ACL” indica as restrições de leitura e/ou escrita de cada objeto Parse para os usuários que fazem o acesso. No exemplo, o aplicativo está autorizado a ler, escrever e alterar os dados. Porém, é possível tornar os dados passíveis apenas de serem lidos, porém sem autorização para modificá-los.

Do lado esquerdo da Figura 28 nota-se o nome dado ao objeto, chamado de “ECG4”. Apesar da Figura 28 conter apenas 13 linhas, significando 13 segundos de gravação do sinal de ECG, pode-se ler do lado esquerdo circundado em vermelho o número 31, que indica que na realidade existem 31 linhas, ou seja, 31 segundos de gravação.

3. METODOLOGIA

Para o desenvolvimento deste projeto, verificaram-se os dados a serem pesquisados através dos objetivos específicos contidos no início deste trabalho. Informação referente ao sistema android e suas diversas classes têm disponível em sua página própria para desenvolvedores, com detalhes sobre o uso das mais novas classes e métodos.

Na escolha do microcontrolador levou-se em consideração o que houvesse de mais acessível em termos de disponibilidade no mercado e tempo para aquisição do microcontrolador, contanto que se mantivesse uma *performance* mínima adequada ao sistema.

Para o sistema remoto de monitoramento várias alternativas existem, indo desde o simples monitoramento até um sistema que usasse armazenamento em servidores, o que foi mais interessante. Alguns protocolos foram implementados quanto às comunicações entre dispositivos e entre dispositivos e a nuvem. Por fim, ao final deste capítulo mostram-se os resultados.

3.1. VISÃO GERAL DO SISTEMA

O projeto desenvolvido envolveu o estudo concomitante de diversas partes do projeto, desde o estudo da interface bluetooth, passando pelo quadro de comunicação de dados até o sistema android que não é algo estático, mas sim de desenvolvimento constante pela Google.

Para que todo o processo de desenvolvimento do sistema começasse, muito estudo-se à respeito da comunicação bluetooth, seja entre dispositivos com hardware isolados como neste caso das interfaces bluetooth, seja quando envolvesse as bibliotecas do sistema android. Uma das premissas a se resolver foi a de encontrar módulos de fácil aquisição. Por isto, partiu-se pela decisão no uso de placas prontas transceptoras de sinal bluetooth. Muitos são os fabricantes destes sistemas prontos para comunicação usando protocolo bluetooth. Estes sistemas fazem o estabelecimento do enlace de rádio, como já descrito em capítulo anterior, de forma automática. No entanto, mesmo com toda a robustez existente na

comunicação bluetooth, tem-se como um dos objetivos o desenvolvimento de um protocolo que controlasse o fluxo de dados recebidos e transmitidos entre as partes envolvidas, sendo elas o microcontrolador e o dispositivo android. O quadro de comunicação desenvolvido envolve a maneira como a informação trafega entre o módulo bluetooth e o dispositivo android, de modo que se possa saber onde e quando a transmissão teve falha ou eventualmente tenha sido interrompida, caso tais erros venham a ocorrer.

Como microcontrolador foi usado o ATmega328P, da Atmel, que também é encontrado em muitas placas didáticas microcontroladas. Ele também pode ser adquirido separadamente caso fosse necessário usar algum sinal de *clock* diferente do usado na placa didática.

Dois outros dispositivos móveis foram usados como equipamentos com sistema android.

Um quadro bastante resumido do funcionamento da comunicação entre as partes que formam este trabalho se encontra na Figura 29.

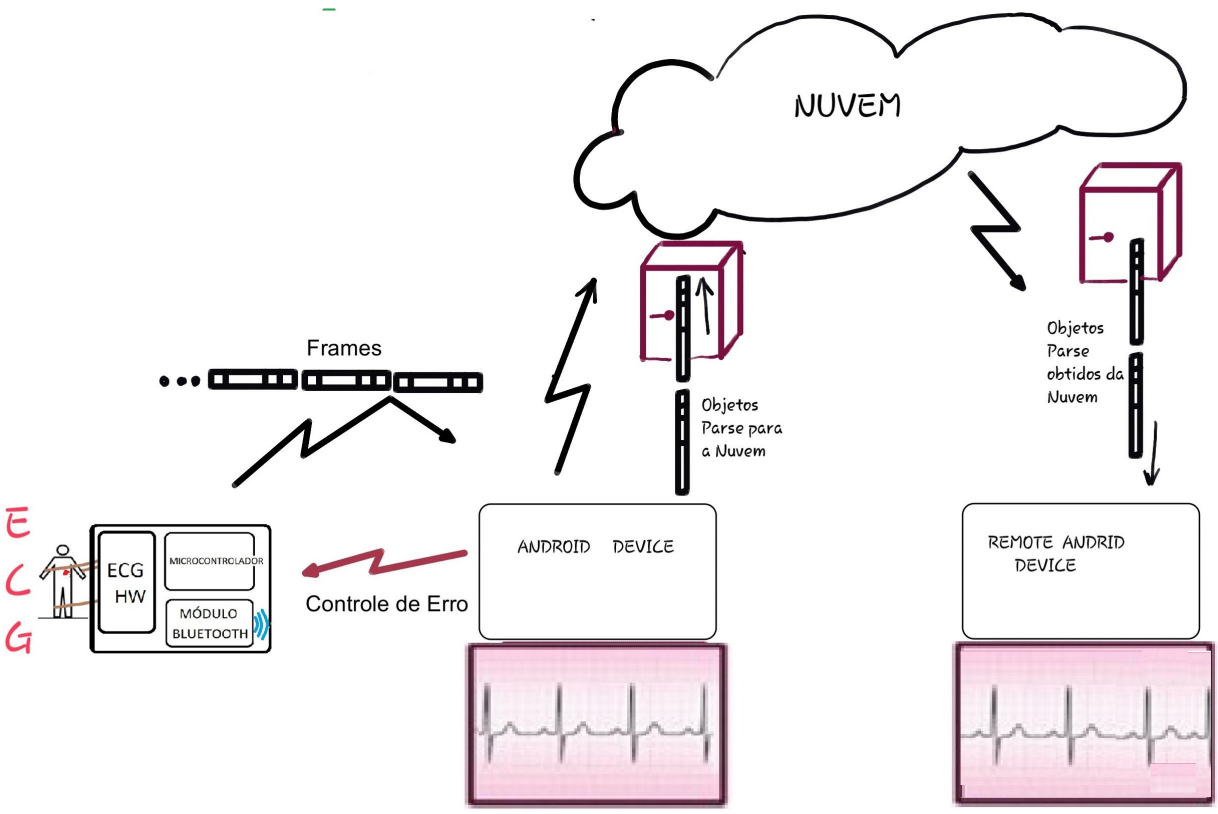

Figura 29 – Visão Geral do Projeto de Monitoramento Remoto de Sinais Biomédicos.
Fonte: Autoria própria

A Figura 29 ilustra todos os elos de comunicação que este trabalho envolve. O link de comunicação bluetooth entre o dispositivo bluetooth e o dispositivo android local estebelecem uma conexão segura, com controle de fluxo e erros fim a fim. Esta conexão segura já é prevista no protocolo bluetooth, porém não é disponível em muitos módulos bluetooth, como no módulo utilizado neste projeto. Por isto um novo *frame* foi implementado de forma a viabilizar a transmissão das amostras do sinal ECG de maneira sequencial, contendo informações adicionais que tornam possível ao dispositivo android perceber se os *frames* estão ou não sendo corretamente recebidos e ao tempo certo. Em caso negativo, solicitações de reenvio do *frame* são enviadas ao microcontrolador via bluetooth. Como se trata de um sinal contínuo no domínio do tempo, existem restrições quanto ao máximo atraso permitido para a recepção das amostras do sinal ECG.

Além da comunicação bluetooth, outros dois links de comunicação via protocolo IP são possíveis de serem feitos através do software implementado para este trabalho. Um primeiro link via protocolo IP é responsável pelo envio do sinal ECG, decodificado pelo dispositivo android local, para o servidor em nuvem. O segundo link via IP serve para que um dispositivo android remoto possa receber a informação referente ao sinal ECG enviado pelo primeiro dispositivo android. Desta forma, o sinal ECG, transmitido via bluetooth é recebido pelo primeiro dispositivo android e recebido pelo segundo dispositivo android.

Para que se garantisse a entrega dos pacotes com amostras do sinal ECG à nuvem, usou-se as bibliotecas da parse.com como mostrado mais à frente, enquanto que o link bluetooth usa uma comunicação assíncrona entre as respectivas partes. Os links IP, por usarem a biblioteca disponível da Parse, não necessitam de controle de fluxo de dados. Tanto o processo de armazenagem de dados em nuvem quanto o processo de leitura dos dados da nuvem são feitos por métodos já existentes dentro das classes java fornecidas pela Parse.

3.2. SIMULADOR DE SINAIS DE ECG DA BIO-TEK INSTRMENTS

Para o desenvolvimento e testes do projeto foi utilizado o equipamento simulador de sinais de ECG da Bio-Tek Instruments, Inc. modelo ECGF-Plus.

A vantagem de uso do simulador é a não variabilidade do batimento cardíaco. O simulador gera sinais com uma determinada configuração selecionada de batimentos, sendo mais interessante durante os testes realizados no projeto.

A figura 30 apresenta o simulador utilizado.

Figura 30 – Simulador de sinais de ECG da Bio-Tek modelo ECG-Plus.
Fonte: Autoria própria

3.3. CAMADA MICROCONTROLADA E SENSOR DE ECG

Esta camada está relacionada com todo o hardware desenvolvido para este trabalho. Ela contém um circuito para pretratamento do sinal de ECG, o microcontrolador e um módulo transceptor bluetooth.

3.3.1. Condicionamento do sinal de ECG

O sinal do ECG vem de uma placa de hardware externa. Esta placa faz a adequação e filtragem do sinal para que ele possa ser amostrado pelo conversor analógico-digital do microcontrolador. As principais questões envolvidas neste hardware são os filtros estão ilustradas na figura 31, assim como o módulo de aquisição de sinal de ECG, o amplificador de instrumentação INA102, ilustrados na Figura 31.

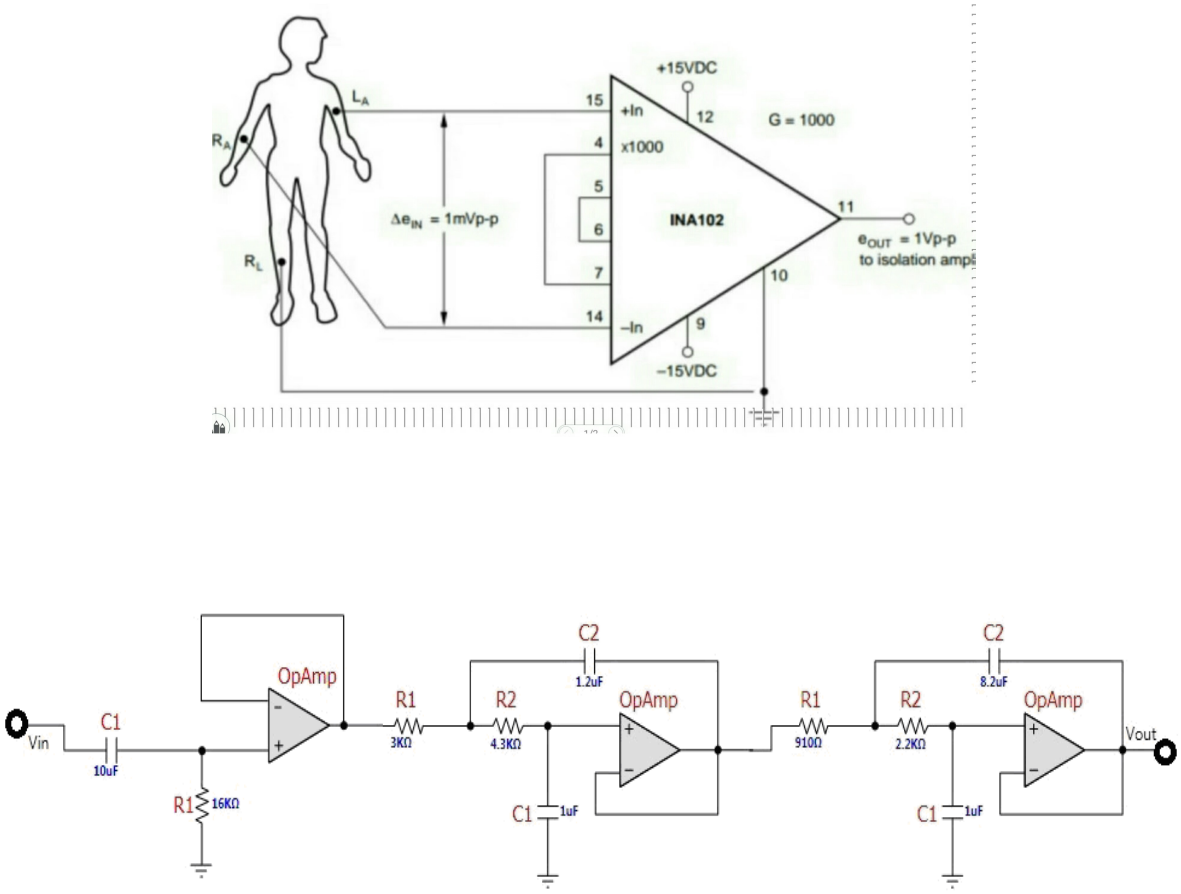

Figura 31 – Hardware externo para aquisição do sinal de ECG.
Fonte: Autoria própria

Após o tratamento do sinal de ECG, faz-se a amostragem do sinal de ECG no microcontrolador através do pino de entrada ligado internamente ao conversor analógico-digital.

3.3.2. Sistema microcontrolado com ATMEGA32 da Atmel

O sistema microcontrolado é responsável pelo envio via bluetooth do sinal recebido dos circuitos de condicionamento do sinal de ECG, baseando-se em 3 rotinas principais:

- Rotina geradora de 2 milissegundos;
- Rotina controladora do conversor analógico-digital;
- Rotina de recepção de dados seriais via bluetooth.

Neste trabalho foi utilizado o microcontrolador AtMega328P contido em um kit de arduíno, chamado de arduino-nano. Está ilustrado na figura 32.

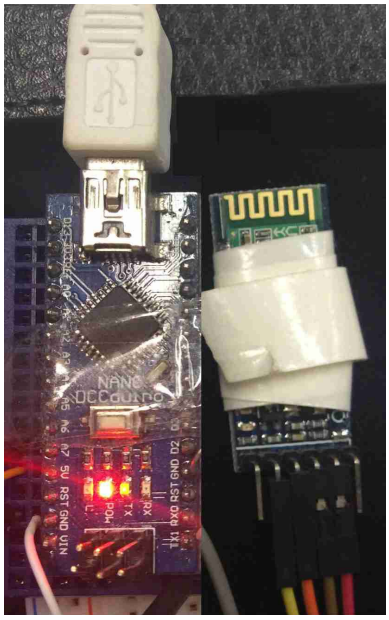

Figura 32 – Sistema Microcontrolado e módulo HC-05.
Fonte: Autoria própria

O funcionamento básico do sistema microcontrolado é ilustrado na Figura 33, resumindo-se à geração de amostras que são armazenadas em um *frame* de 4 amostras para então serem transmitidas.

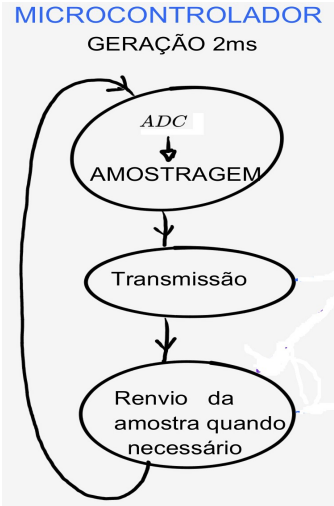

Figura 33 – Rotina de 2 ms
Fonte: Autoria própria

A Figura 33 ilustra o funcionamento do software do microcontrolador de forma resumida. Por ADC da Figura 33 entenda-se o conversor analógico-digital. O ADC

faz amostragens do pino externo, convertendo-as em informações de 10 bits sendo armazenadas em dois bytes. A conversão se dá pela rotina padrão como ilustrado pela Figura 34. Por simplicidade, o byte menos significativo é somado ao *checksum* para atualizar seu valor.

```
// Read the AD conversion result
unsigned int read_adc(unsigned char adc_input)
{
ADMUX=adc_input | ADC_VREF_TYPE;
// Delay needed for the stabilization of the ADC input voltage
delay_us(10);
// Start the AD conversion
ADCSRA|=(1<<ADSC);
// Wait for the AD conversion to complete
while ((ADCSRA & (1<<ADIF))==0);
ADCSRA|=(1<<ADIF);
adH=ADCH;
adL=ADCL;
return ADCW;
}
```

Figura 34 – Rotina de leitura do ADC
Fonte: Autoria própria

Na Figura 33 o bloco intermediário refere-se à transmissão via bluetooth. Aqui inclui-se naturalmente os dados a serem enviados. À medida que os dados são amostrados, vão sendo agrupados a cada 4 amostras, formando um novo *frame* ou quadro de dados é enviado pela serial ao bluetooth junto com as outras informações de índice e *checksum*. A formação do *frame* é vista logo mais em seguida. A última parte da Figura 33 refere-se à retransmissão do *frame*, quando for necessária. Na mesma rotina da transmissão também tem-se a possibilidade de retransmissão. Em suma, quando tem-se a primeira, segunda ou terceira amostras, verifica-se se há a necessidade de retransmissão do *frame* retido no buffer interno. Na quarta amostra só é possível transmitir o *frame* corrente.

Para que a rotina da transmissão seja ativada a cada 2 milissegundos aproximadamente, ou seja gerando uma taxa de amostragem de aproximadamente 500 amostras por segundo, é necessário configurar o contador interno do Atmega328 que dispara um serviço de interrupção a cada vez que este *timer* sofrer um *overflow* na sua contagem, ou seja, que o seu registro interno passa de 255 para zero.

3.3.2.1. Geração do sinal de amostragem

A rotina de 2 ms é chamada através de um serviço de interrupção do microcontrolador que, executando a rotina principal, pára tudo o que fazia para começar a executar a rotina de amostragem.

Os dois milissegundos são gerados a partir de sucessivas divisões do *clock* principal de 16 MHz do sistema. O sinal de 16 MHz de *clock* interno correspondem a 0,0625 µs de período. Este *clock* é configurado para ser dividido por 1024, gerando com isto um sinal cujo período é de 0,0625 x 1024, resultando em 64 microssegundos, ou 15.625 Hz. Este é o *clock* que alimenta um contador interno do Atmega328. Porém este contador interno é configurado para começar sua contagem com valor igual a 224. Deste modo, depois de 32 períodos de *clock* tem-se 224 somados a 32 resultando em 256, gerando um overflow e por consequência, um sinal de interrupção que aciona a rotina de 2 ms correspondente à rotina de transmissão.

Resumidamente tem-se 64 µs do período do *clock* aplicado ao *timer* vezes 32 ciclos do *timer* até o seu *overflow*, resultando em 2,048 ms. Note-se que o período de amostragem não é exatamente de 2 ms devido ao sinal de *clock* do próprio controlador.

3.3.2.2. Considerações para o circuito de transmissão serial

Uma das partes importantes de projeto foi a programação da interface serial, mais conhecida como USART. Necessita-se a mais alta taxa de transmissão serial em bits por segundo disponível pelo microcontrolador, que é refletida diretamente pela velocidade de modulação mais comumente chamada de *baudrate*. Porém esta taxa de *baudrate* tem que ser compatível com as taxas disponíveis de entrada para o módulo bluetooth. Chegou-se à conclusão que a maior taxa possível para os 16 MHz de *clock* é uma transmissão a 115.000 *bauds*, apesar da interface serial do microcontrolador poder oferecer mais, assim como também o módulo bluetooth. No entanto, usando o dobro desta taxa, que equivale a 230.000 *bauds* no microcontrolador já não seria compatível com os 250.000 *bauds* disponíveis para o

módulo bluetooth. Por esta razão adotou-se os 115.000 *bauds*. É a máxima velocidade de comunicação serial comum aos dois dispositivos, ao microcontrolador e ao módulo Bluetooth.

No microcontrolador existe um circuito que, a partir do *clock* do sistema de 16 MHz divide por um valor contido em um registrador chamado de UBRR (*UART Baud Rate Register*), que é o registrador da USART responsável pela velocidade de transmissão assíncrona, gerando a taxa desejada. Este processo segue as equações (1) e (2).

(a) $$BAUD = \frac{Fosc}{(16x(UBRR + 1))} \quad (1)$$

(b) $$BAUD = \frac{Fosc}{(8x(UBRR + 1))} \quad (2)$$

Para este trabalho foram utilizados os dados de 16 MHz de *clock* e o *baudrate* de 115.000 *bauds*.

Pela equação (1), calcula-se UBRR como 7,39. Então:

- Adotando-se 7 para UBRR, BAUD fica em 125.000 *bauds*, muito distante dos 115.000 *bauds* desejados;
- Adotando-se 8 para UBRR, BAUD fica em 111.111 *bauds*;
- Pela equação (2), UBRR fica em 16,39. Note-se que um byte deve ser inteiro;
- Caso se adotasse 16 para UBRR, o *baudrate* seria 117.647 *bauds*, com erro de +2,3% em relação aos 115.000 *bauds*;
- Caso se adotasse 18 para UBRR, o *baudrate* seria 111.111 *bauds*, com erro de -3,3% em relação aos 115.000 *bauds*.

Conclui-se que o uso da equação (2) é mais adequada, com o registrador em 16 ou #10h em valores hexadecimais. Com isto o erro na taxa de transmissão (*baudrate*) gerado em relação aos 115.000 *bauds* desejados ficou em 2,3%.

Resultados mostraram que este erro é superado pelo circuito de controle interno do microcontrolador.

3.3.3. Módulo bluetooth HC-05 da Wavesen

O módulo Bluetooth HC-05 da WaveSen (WAVESEN, 2016) atua tanto como dispositivo mestre em uma rede bluetooth, como pode atuar sendo dispositivo escravo.

O item 3.3.2.2 descreve a geração do *baudrate* responsável pela transmissão assíncrona enviada ao módulo bluetooth. Por uma questão de facilidade de aquisição no mercado, escolheu-se o HC-05 ilustrado na Figura 35 (WAVESEN, 2016).

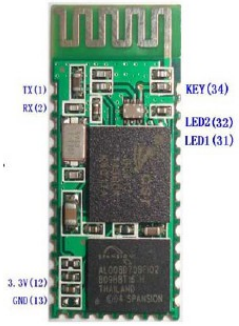

Figura 35 – Módulo bluetooth HC-05 da WaveSen
Fonte: www.wavesen.com

Este módulo já vem soldado a uma placa que por sua vez disponibiliza os pinos chamados de RX, TX, KEY, GND e VCC (recepção, transmissão, chave, terra e positivo da alimentação), conforme apresenta a Figura 36.

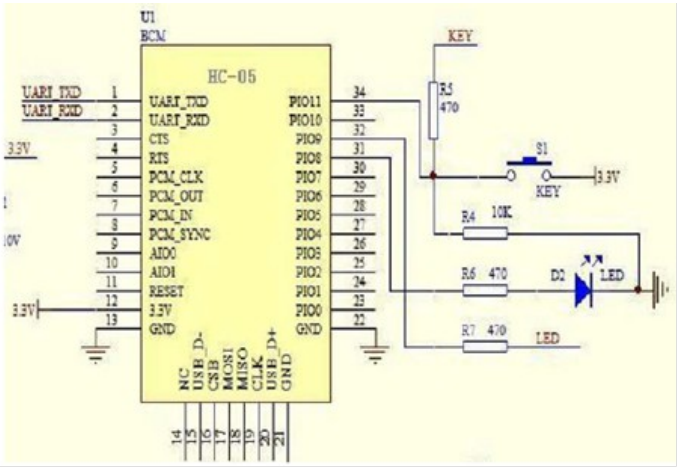

Figura 36 – Esquema elétrico do módulo bluetooth HC-05 para interfaceamento.
Fonte: www.wavesen.com

No circuito da Figura 36, a chave é adotada externamente somente no ato da programação interna do módulo para que opere em 115.000 bauds, de acordo com a taxa pré programada para o microcontrolador. Depois de programado, usa-se apenas os pinos de transmissão e recepção (pinos 1 e 2, respectivamente), conectados na recepção e transmissão do microcontrolador, respectivamente.

3.3.4. Estrutura do quadro de transmissão de dados

A maneira mais simples de se transmitir dados é pelo envio de dados, sem nenhum controle de fluxo ou de erro, também chamada de transmissão *simplex*. Esta transmissão é feita amostra por amostra, do microcontrolador para o módulo bluetooth, ou seja de uma parte à outra (PAULO, 2014). Contudo, verifica-se que muitas soluções bluetooth podem parar de operar pelo aumento da distância entre os dispositivos ocasionando a desconexão. Se o aplicativo não foi desenvolvido para prever o tratamento desta exceção, pode ocorrer que o aplicativo pare de funcionar ou trave, bloqueando o sistema operacional.

Para solucionar este problema adotou-se um quadro próprio de comunicação que possui um grau de robustez suficiente para garantir que sinais em tempo real, na amostragem condizente com sinais cardíacos, sejam retransmitidos quando na eventualidade da perda de algum pacote ou *frame* de informação. A interface serial do microcontrolador foi configurada para transmissões assíncronas de 8 bits via interface serial. Acrescentando-se o *start bit* e o *stop bit* formam-se 10 bits enviados por byte conforme ilustrado na Figura 37. Estes bits são relevantes na hora da transmissão, pois ocupam espaço na linha do tempo e devem ser multiplicados pelo número de bytes de informação em um *frame* de dados.

Figura 37 – Estrutura do byte enviado pela Serial
Fonte: Autoria própria

O funcionamento geral do lado microcontrolado, em conjunto com o sistema android, está ilustrado na Figura 37.

Em uma comunicação bluetooth em modo assíncrono, caso alguma informação seja perdida, ela não será mais retransmitida. Por isto a sugestão proposta aqui, conforme mostra a Figura 38 é garantir que os pacotes recebidos com erro peçam retransmissão. Para cada pacote perdido uma mensagem do tipo "nack" é enviada para o microcontrolador pedindo retransmissão do *frame* com erro. Estes erros são armazenados em buffer para a retransmissão logo que possível. Esta retransmissão deve ocorrer rapidamente, senão será descartada na hora da montagem final do sinal na tela do dispositivo android. A Figura 39 ilustra no tempo o que o diagrama de estados da figura 39, em termos mais gerais, elucida.

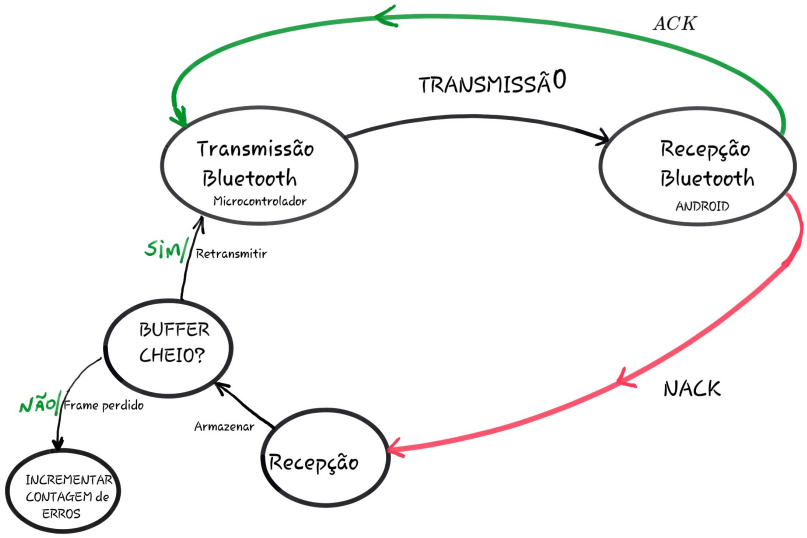

Figura 38 – Diagrama de estados do microcontrolador e do sistema android
Fonte: Autoria própria

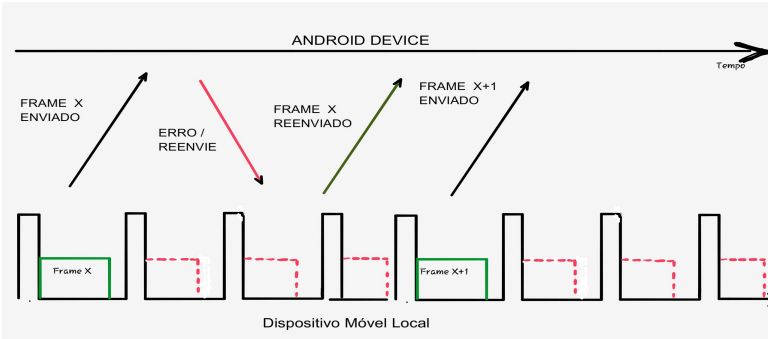

Figura 39 – Diagrama no tempo do envio e reenvio de *frames* ao android.
Fonte: Autoria própria

O envio dos *frames* com erro, prearmazenados em buffer, ocorre nos períodos das primeiras três amostras. A Figura 40 mostra que somente na quarta

amostra envia-se o *frame* corrente. Nas amostras primeira, segunda ou terceira, só há transmissão caso haja algum *frame* no buffer aguardando retransmissão.

O tamanho em bits de cada *frame* possível de transmissão depende, entre outras coisas, da taxa de modulação medida em *bauds* pela USART do microcontrolador e do tamanho em bits do *frame* enviado.

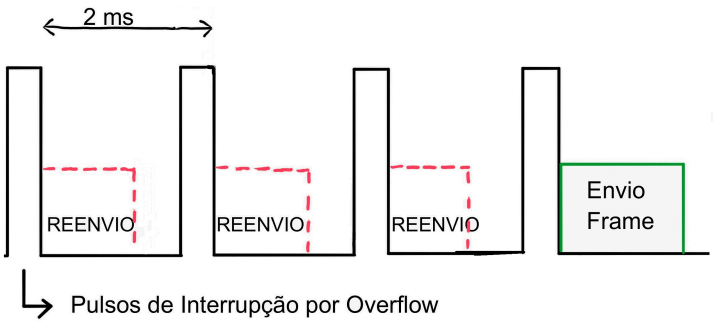

Figura 40 – Sequência de *frames* durante a transmissão.
Fonte: Autoria própria.

Na seção 3.3.2.2 foi visto que a taxa de transmissão é 2,3% superior aos 115.000 *bauds*. Na figura 39 pode-se ver o *frame* completo formado pelo preâmbulo de 8 bits, índice de 8 bits, pelo valor da taxa de amostragem de 8 bits, pelas quatro amostras de 16 bits cada e um *checksum* de 8 bits final, totalizando os 96 bits formadores do *frame*. Para efeitos de transmissão serial, deve-se levar em conta o *start bit* e o *stop bit*. Desta forma, para cada 8 bits tem-se na verdade uma transmissão de 10 bits. Com isto, os 96 bits quando somados aos *start bits* e *stop bits* geram o equivalente a 120 bits transmitidos por cada *frame* enviado. O tempo para a transmissão do *frame* é apresentada pela equação (3).

$$TEMPOframe = \frac{(120bits)}{(115255))} = 1043\mu s \qquad (3)$$

O valor de 1043 µs equivalente à metade dos 2 ms entre 2 *frames* é adequado para o *frame* chegar ao android. Desta forma existe tempo para verificar sua integridade e há tempo para um pedido de retransmissão ainda no mesmo intervalo de 2 ms, caso necessário.

Os períodos de reenvio servem para transmitir quaisquer *frames* contendo quatro amostras cada, *frames* estes que não tenham sido transmitidos com sucesso ao dispositivo android.

Ao *frame* note-se que são adicionados dados de controle e de fluxo, como o *checksum* e o valor de indexamento, além do preâmbulo e do valor da taxa de amostragem para que o sistema android possa identificar qual a procedência do sinal enviado. A formação do *frame* segue na Figura 41. Caso o sistema android receba as mensagens com erro, uma mensagem de reenvio é solicitada ao microcontrolador, também via bluetooth, pelo aplicativo android.

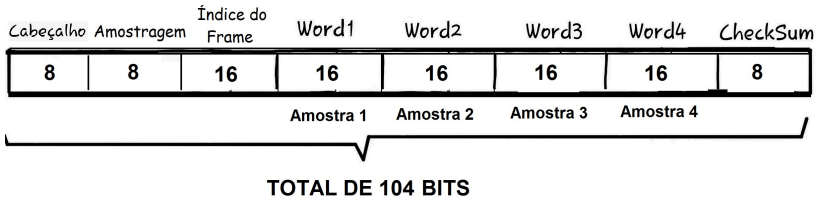

Figura 41 – Estrutura do *frame* com 4 amostras na comunicação bluetooth.
Fonte: Autoria própria.

A maneira como os *frames* chegam ao aplicativo android, assim como a maneira como eles podem retornar ao microcontrolador é mostrado na Figura 42. Nela estão os *frames* enviados na cor preta e os eventuais *frames* reenviados na cor verde. Os pacotes informativos de recebimento incorretos estão na cor vermelha.

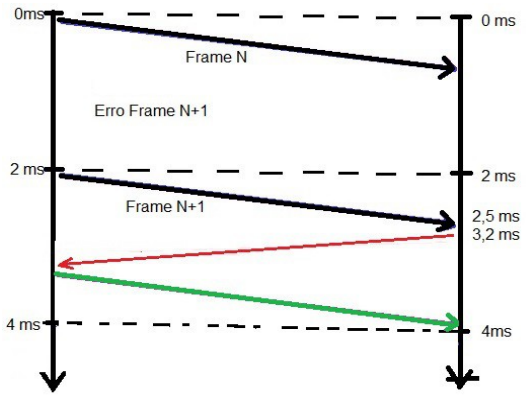

Figura 42 – Diagrama no tempo do fluxo de pacotes/*frames.*
Autoria Própria

A Figura 42 mostra algumas informações de tempo que não são possíveis de se determinar, como por exemplo, o exato tempo em que a informação da seta vermelha, de recebimento incorreto do *frame* N+1, se deu. Tão pouco o momento exato quando da sua correta retransmissão (seta em verde). Nas transmissões de informação de um lado para o outro existe um buffer de envio e outro buffer de recebimento, intrínsecos à transmissão bluetooth em si, tanto do lado micro-

controlador quanto do lado da aplicação android. Por este motivo foi verificado que até um tempo de 1000 ms é aceitável para o sistema ficar sem trocar pacotes ou *frames* entre as partes, antes de desencadear uma rotina de reconhecimento de falha geral na comunicação do lado android.

Em relação ao *timming* do sistema de transmissão, os únicos tempos precisos de serem medidos são os usados para o envio de informação, representados pelas setas escuras, da esquerda para a direita. Porém, como ilustrado na Figura 43, outras variáveis como o tempo de processamento de hardware no dispositivo móvel do *frame* enviado ou recebido, ou o tempo de processamento dentro do aplicativo android também não são definíveis. É possível que o aplicativo android por algum motivo demore a conferir o *checksum*, atrasando um eventual pedido de retransmissão de *frame*. Mesmo outros aplicativos e serviços que estejam ativos no sistema android podem forçar atrasos no processamento das informações transmitidas e recebidas, gerando inclusive erros em casos extremos. Por este motivo a rotina de falha geral desencadeada pelo *timer* do sistema android incorporado ao aplicativo se faz necessário. O *timer* é apenas uma classe criada que conta até certo valor de *time-out*, fazendo parte do programa android. Caso o sistema trave por algum motivo, a contagem de *time-out* é atingida desencadeando sua respectiva rotina.

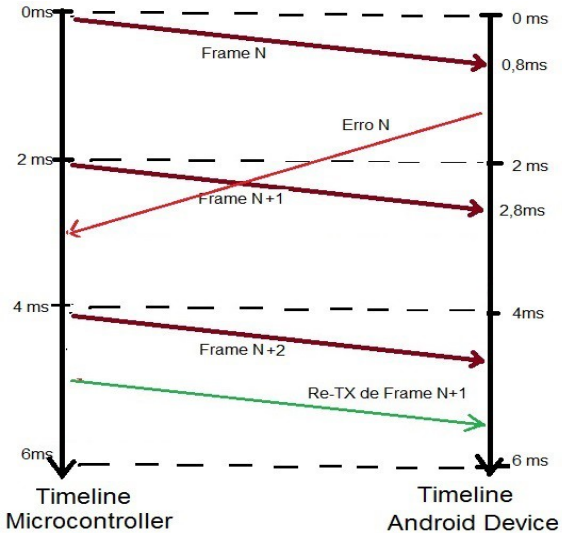

Figura 43 – Diagrama de tempo só com Atrasos.
Fonte. Autoria Própria.

Do lado do microcontrolador tem-se ainda uma rotina que recebe os erros. Os erros são enviados a partir do aplicativo android quando os valores de *checksum*

calculado e o recebido não conferem. A Figura 44 ilustra simplificadamente o funcionamento.

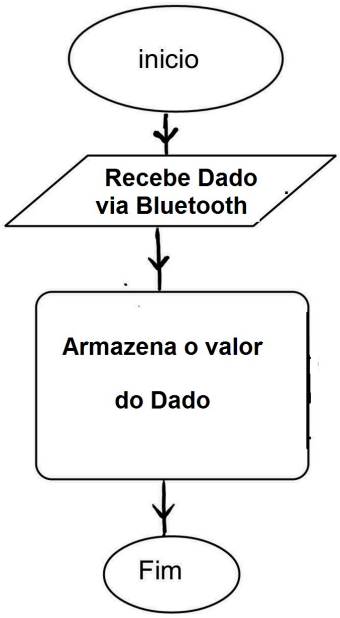

Figura 44 – Recepção de dados pelo microcontrolador via USART
Fonte: Autoria própria

Esta rotina é ativada por interrupção toda a vez que cheguar algum dado via serial. Isto se justifica a fim de não comprometer o fluxo de leitura do sinal de ECG enquanto se aguarda por um eventual código de erro do dispositivo android. Sendo assim o sistema pode estar enviando um *frame* enquanto ele interrompe o que estiver em execução para receber um código de erro do aplicativo android. As informações vindas se referem ao número do índice do *frame* que foi transmitido e mal recebido. Como o pedido de reenvio transmitido pelo android, quando houver, tem apenas um byte de tamanho, o tempo para sua interrupção é muito curto. Este tempo é necessário para que o número de índice seja colocado em ordem crescente em uma fila dentro de um vetor E, para retransmissão futura.

Na rotina ilustrada na Figura 45, desencadeada a cada 2 ms, verifica-se se o número da amostra é primeira, segunda, terceira ou quarta.

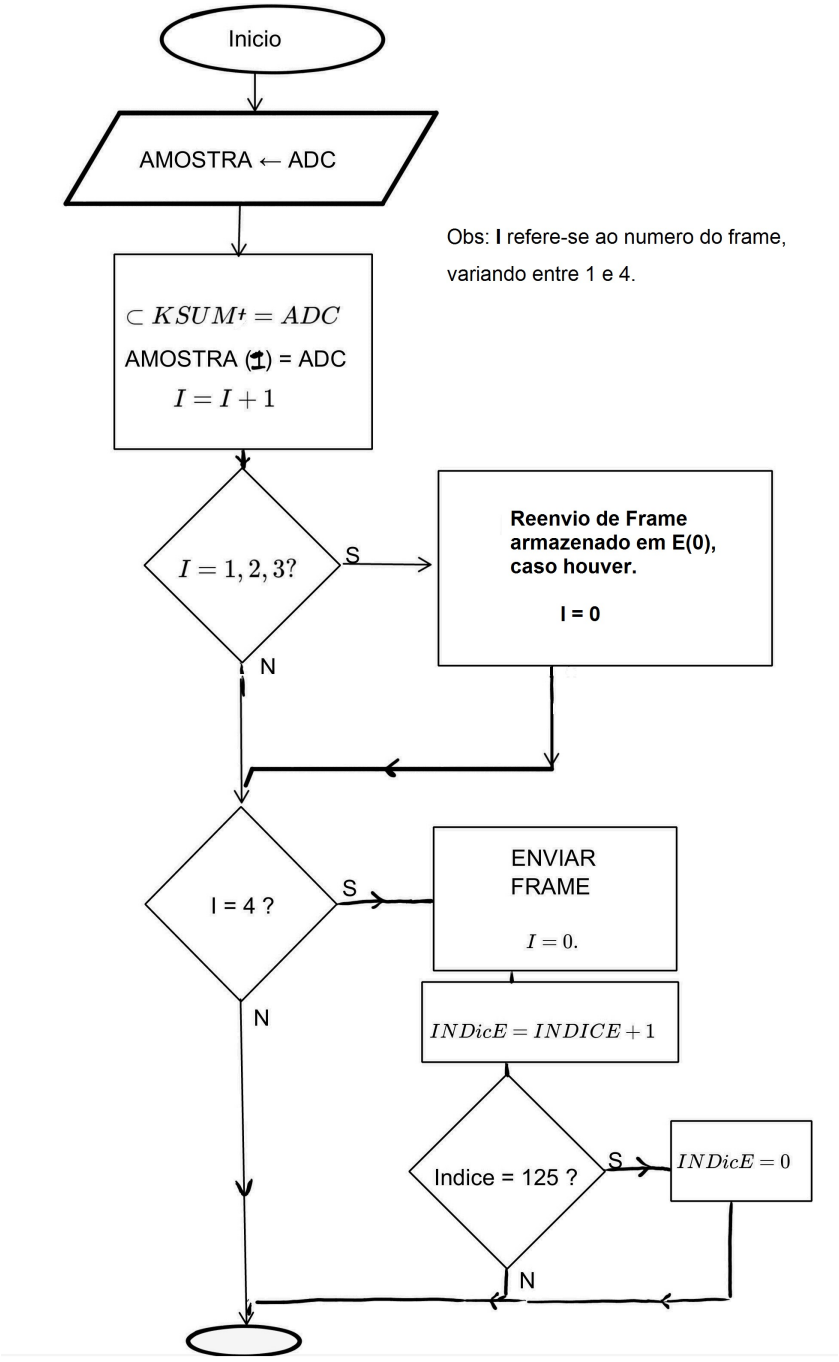

Figura 45 – Rotina de transmissão do *frame* pelo microcontrolador
Fonte: autoria própria

Sendo uma das primeiras três, é então também verificado se o primeiro elemento do vetor E é diferente de zero, o que implicaria que o valor que lá consta é o número de índice com prioridade a ser reenviado. Caso esta primeira posição do vetor E seja igual a zero, significa que não há nada a se retransmitir, seguindo o fluxo da rotina adiante. A figura 46 se refere a este processo de retransmissões de um mesmo *frame*.

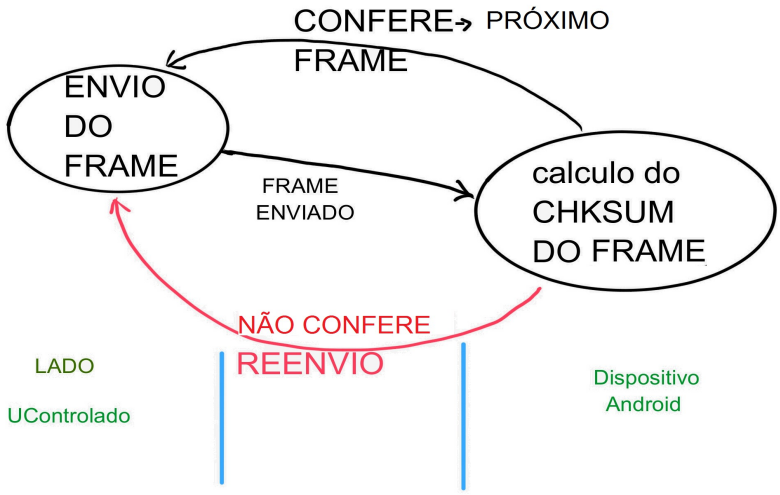

Figura 46 – Diagrama de reenvio de *frames*.
Fonte: Autoria Própria.

Transmite-se o *frame* novamente com o mesmo número de índice. Caso a transmissão volte a ter problemas, o aplicativo android verificará então o erro, enviando novamente o mesmo número de índice ao controlador que então, pela sua rotina de interrupção receptora de dados vindos da serial, recolocará tal índice a fim de ser retransmitido pela segunda vez. Pode-se ver pela figura 46 que não há limites para o reenvio de um mesmo *frame*. Porém, volta-se ao caso que esta situação só poderia ocorrer caso o nível de ruído ambiente seja tão alto que não somente uma, mas muitas amostras viriam com erro. O que ocorrerá nesta situação é que um indicador de número de erros na janela principal acusará valores elevados.

3.4. RECEPÇÃO DE DADOS NO ANDROID LOCAL

O sinal vem via bluetooth para dentro do android que por sua vez faz o processamento em tempo real do sinal de ECG com seu respectivo controle de erro e de fluxo, além de disponibilizá-lo na tela junto às demais informações de batimento cardíaco em gráficos. A segunda função do mesmo primeiro aplicativo é disponibilizar o sinal via internet para que o segundo aplicativo possa receber. A figura 47 mostra esta iteração entre as partes do primeiro aplicativo e o módulo microcontrolado.

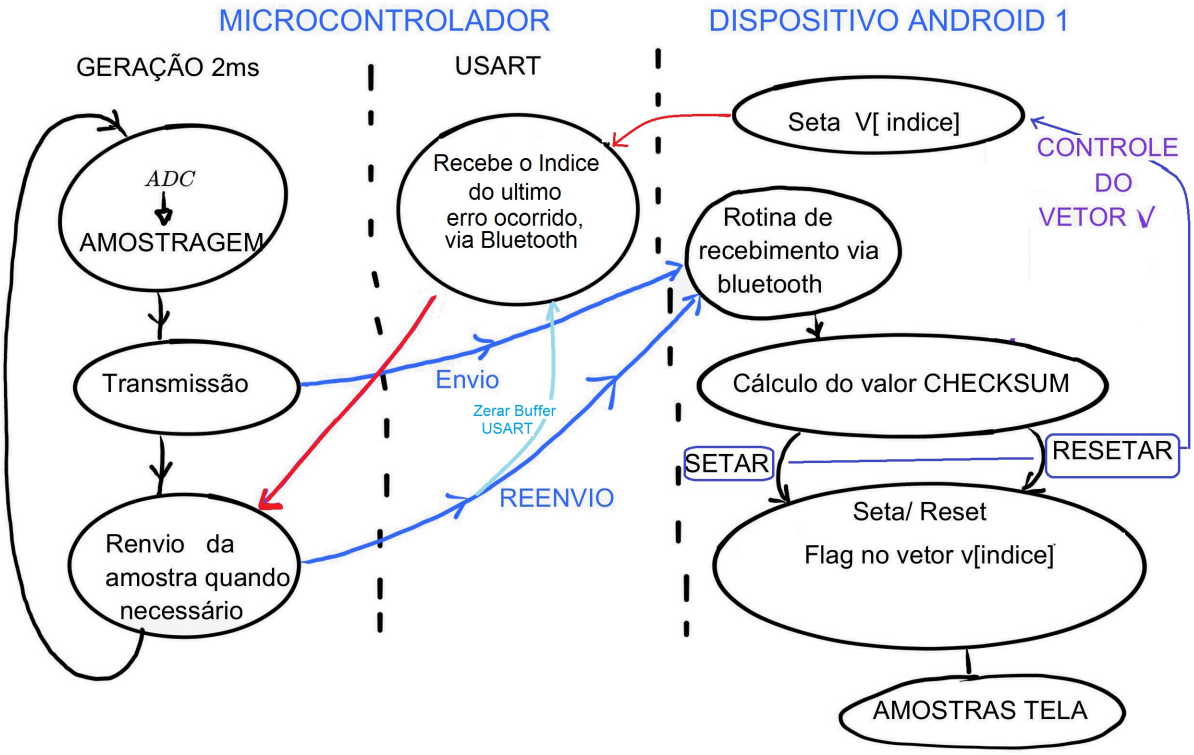

Figura 47 – Diagrama de estados do projeto
Fonte: Autoria própria

O aplicativo envolvido nesta primeira parte é bastante flexível quanto aos requisitos para sua utilização. Apenas é necessário que haja hardware para recepção bluetooth no primeiro dispositivo móvel, caso contrário ele encerra automaticamente. Antes de se colocar o aplicativo para rodar, todo um processo de verificação de hardware, assim como verificação se tal hardware está funcional são realizados. Ainda é necessário procurar dispositivos ao alcance assim como escolher com qual deles se conectar, além de se efetuar a conexão. Este processo está descrito 3.6.

Analisando-se o trajeto percorrido pelos *frames* de amostras do controlador, estes chegam ao dispositivo móvel rodando android que, através de um processo à parte e contínuo, lê as amostras contidas no *frame* recebido. Uma rotina dentro do processo que corre concomitantemente à atividade principal da tela, desmonta o *frame*, sendo seu conteúdo armazenado em variáveis locais no android. Estas variáveis correspondem ao número de índice do *frame* e de seu *checksum*, além das amostragem, de seu *cheksum*, além das amostras propriamente ditas. Com as amostras individuais, internamente o android recalcula o *checksum* a fim de comparar com o *checksum* recebido no *frame*. Sendo igual, então não houve erros.

Caso haja alguma discrepância entre o *checksum* recebido e o calculado, um código referente ao índice recebido com erro é enviado pelo processo android de volta ao sistema microcontrolador que então o armazena para reenvio.

A figura 48 ajuda a compreender melhor o processo no aplicativo android. O processo permanece lendo a interface bluetooth enquanto tiver algo a ser lido. Funciona como se fosse um processo rodando eternamente até que ou um erro ocorra, como o erro correspondente a estar fora de alcance, ou o sistema seja desligado pelo usuário. Havendo uma destas situações, uma situação de exceção ocorre, desencadeando a rotina de *time-out*.

No caso de não haver nada no buffer interno de bluetooth do aparelho android, ou em caso de algum tipo de falha, uma mensagem de desconexão será mostrada na tela da atividade principal. Esta possibilidade foi apenas prevista, pois em termos de software ela existe, apesar de pouco provável.

Existe uma segunda possibilidade também prevista no projeto que ocorre quando o tablet sai fora de alcance do transmissor de sinal. Para esta situação, um *timer* interno permanece em contagem continua, sendo zerado a cada tempo equivalente a 500 amostras, ou seja, a cada 125 *frames* de dados. Este *timer* é representado por um processo à parte dentro do programa android local e está ilustrado na figura 49. O *timer* conta em intervalos de 100 milissegundos e, ao chegar ao ponto predeterminado de contagem igual a 1000 ms, aciona rotina de erro correspondente ao fato de estar fora de alcance. Entretanto, caso o processo de envio e recebimento de pacotes estiver acontecendo de forma correta, a contagem do *timer* é zerada à medida que amostras vão chegando à atividade principal. Isto acontece devido ao uso de um método da classe Timer responsável pelo *reset* da contagem do mesmo.

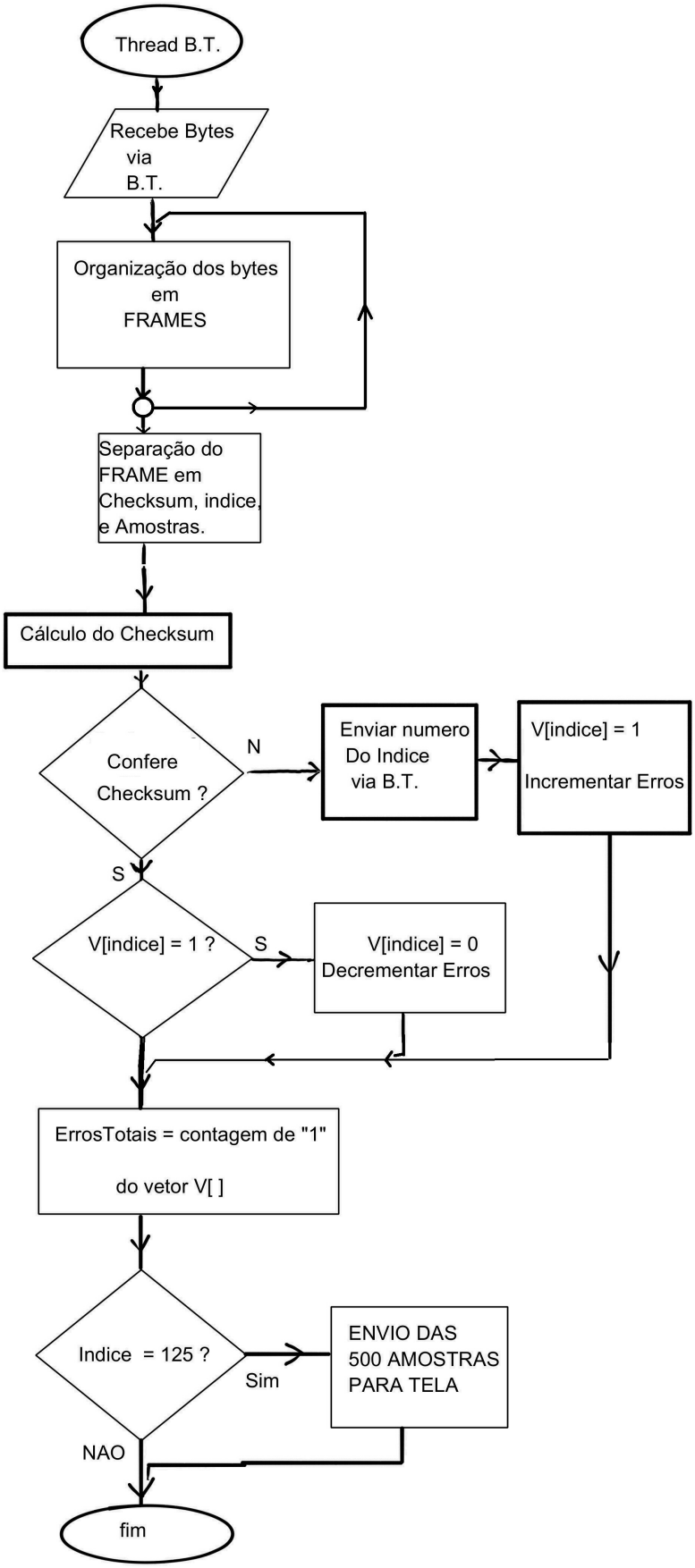

Figura 48 – Diagrama de Fluxo de Recepção de Dados
Fonte: Autoria Própria.

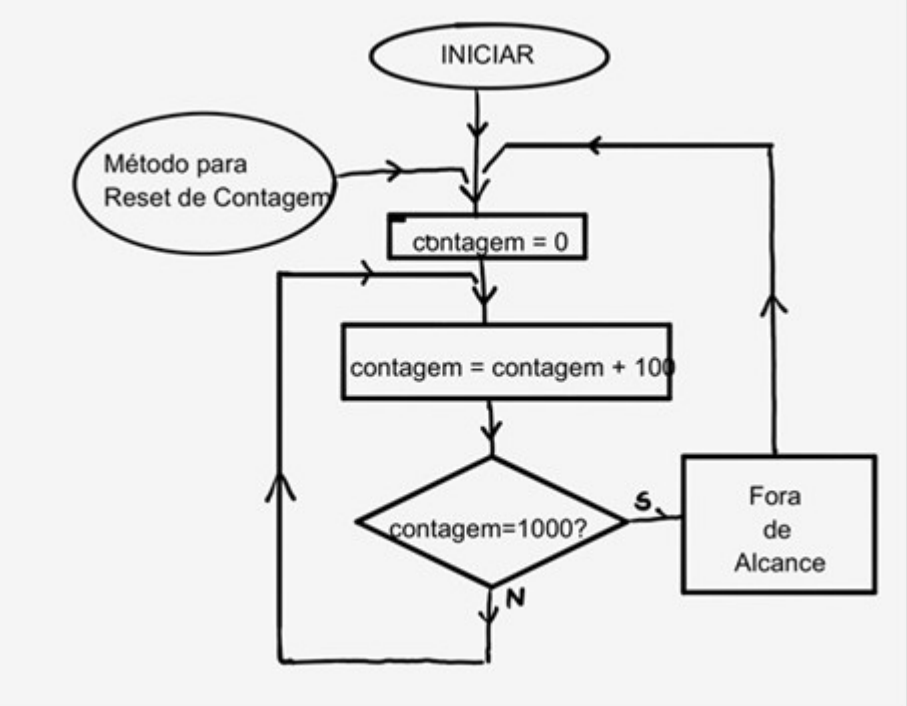

Figura 49 – Diagrama de funcionamento do *timer*
Fonte: Autoria própria.

3.5. FORMATO DO BANCO DE DADOS NA NUVEM

Os dados do sinal de ECG vão sendo agrupados para, quando equivalerem a um período de 1000 ms de tempo, sejam enviados à atividade principal que os desenha na tela. Simultaneamente eles são enviados para um servidor em nuvem, à exceção dos dados incorretamente recebidos, cujos valores são atribuídos o valor zero. Isso facilita ao aplicativo remoto identificar os erros ocorridos.

Este processo faz parte da atividade principal, chamada de *MainActivity* no aplicativo android. A vantagem do uso das bibliotecas de parse.com incorporadas ao projeto do aplicativo android se mostram relevantes neste ponto do projeto. É muitas vezes recomendado uso de processos ou *threads* extras para a gravação de dados na nuvem. Porém, as bibliotecas do Parse permitem também guardar todos os dados em *background* sem interromper a atividade principal. Trata-se do método chamado saveinbackground(), pertencente à classe de objetos suportados pela Parse.

O sinal do microcontrolador é recebido pelo aplicativo android que gera, para este projeto, 500 amostras do sinal de ECG a cada segundo. Estas amostras se

encontram na forma de variáveis de 16 bits, ou *short*. São vetores de 500 *words*, portanto. Como objetos parse não aceitam 16 bits de tamanho, geram-se dois vetores de 500 bytes cada, um para os 8 bits mais significativos e outro vetor para os 8 bits menos significativos. Deste modo são enviados pouco mais de 1000 bytes/segundo aos servidores em nuvem na forma de objetos parse, correspondentes aos dois vetores de 500 bytes cada, à informação do nome do paciente em questão, além da informação do ano, mês, dia, hora, minuto e segundo daquela informação e, da taxa de amostragem. Outras informações como as de identidade e segurança são internamente geradas em nuvem.

A figura 50 é um retrato fiel de como os objetos armazenados para o sinal de ECG, representados pelas linhas de dados, ficam estruturados no banco de dados. A figura 49 contém todos os campos do objeto: o identificador do respectivo segundo de armazenagem; a taxa de amostragem; o número sequencial do quadro; o campo com a data da amostragem no formato ano, mês, dia, hora; o campo com o tempo em segundos do calendário; nome do paciente; os vetores com os dados do sinal de ECG; a data de criação em nuvem; a data da última modificação dos dados; e, por último, o controle de acesso do usuário aos dados chamado de ACL que contém as permissões de leitura e escrita daquele usuário e daquele aplicativo em específico.

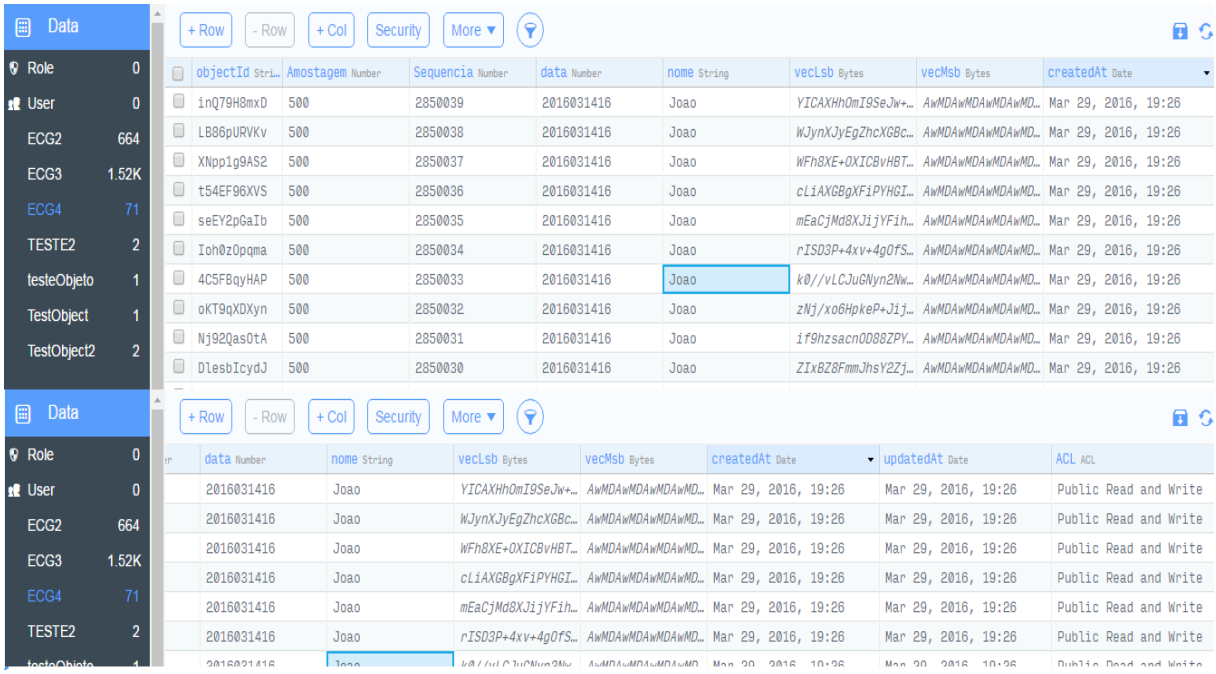

Figura 50 – Banco de dados de sinais biomédicos na nuvem.
Fonte: Autoria própria.

3.6. SISTEMA REMOTO DE RECUPERAÇÃO DE DADOS

O segundo aplicativo, responsável pela leitura remota dos dados de ECG, uma vez inicializado, recebe os sinais via internet enviados pelo primeiro aplicativo e os mostra na tela. Para isto, basta ele estar conectado através de uma rede WiFi ou 3G/4G à rede internet. O aplicativo, no entanto, não pede nenhuma permissão extra além da permissão para ter acesso à rede de dados. Ele tem sua operação quase independente do primeiro aplicativo, se não fosse pelo fato dele depender dos dados enviados pelo primeiro.

Duas situações de exceção são possíveis de serem percebidas pelo aplicativo remoto:

1. Quando existem erros em amostras individuais transmitidas pelo aplicativo local para a nuvem;
2. Quando existe interrupção unilateral do envio das amostras do sinal de ECG, pelo aplicativo local, para a nuvem.

Para o primeiro caso, a detecção de que houve erro quando ao menos uma amostra transmitida pelo aplicativo local para a nuvem e então recebida pelo aplicativo remoto se faz pelo recebimento de valores iguais a zero referentes às amostras decodificadas pelo aplicativo remoto que são visualizadas na tela.

Para o segundo caso, o alarme para o aplicativo remoto indicando que algum tipo de interrupção para o sinal de ECG ocorreu se faz pela verificação do número de sequência da amostra lida. O aplicativo remoto dispara uma situação de exceção caso nenhuma amostra seja recebida, pelo aplicativo local, em um período igual ou superior a 7000ms. Para o aplicativo remoto, por estar recebendo dados via rede internet, deve-se considerar este tempo extra como margem de tolerância antes de disparar internamente no aplicativo remoto uma situação de exceção semelhante. Este tempo deve ser superior aos 1000ms adotados pelo aplicativo local e deve ser inferior ao tempo necessário para que o aplicativo local se reconecte ao microcontrolador via bluetooth. Arbitrou-se um tempo de 7 segundos para este tempo máximo.

Da mesma forma como é feito no aplicativo local, um contador de tempo é zerado a cada conjunto de 500 amostras recebidas da nuvem. Caso este contador

de tempo supere 7000ms, o aplicativo mostra uma mensagem de possível erro ocorrido. No entanto, a decisão de desconectar o aplicativo da rede internet é do usuário.

O aplicativo remoto é apenas um reprodutor das amostras armazenadas em nuvem a cada segundo. Por isto, mesmo que ocorra grandes atrasos na transmissão das amostras do sinal de ECG para a nuvem, no pior caso a imagem no aplicativo remoto fica congelada. Passados 7000ms de congelamento, uma possível situação de erro é mostrada. Mesmo que a conexão à rede internet não seja desfeita e as amostras voltem, por algum motivo, a serem transmitidas, elas voltaram então a serem visualizadas, porém agora com um atraso maior.

3.7. OPERAÇÃO LOCAL DO SISTEMA

O sistema foi planejado para ser de fácil operação. Inicialmente, liga-se o dispositivo móvel local, que para fins deste trabalho foi usado um tablet modelo P7500 de 10 polegadas de tela, da Samsung. Por ser um aparelho com quatro anos de uso, implica em dizer que a maioria dos dispositivos atuais contendo sistema operacional android terão compatibilidade com o software.

Pressionando-se o ícone do aplicativo, a primeira tela ilustrada na figura 51 surge e pede para que a interface bluetooth seja ligada, caso não estivesse previamente. Em caso de não autorização, o aplicativo finaliza.

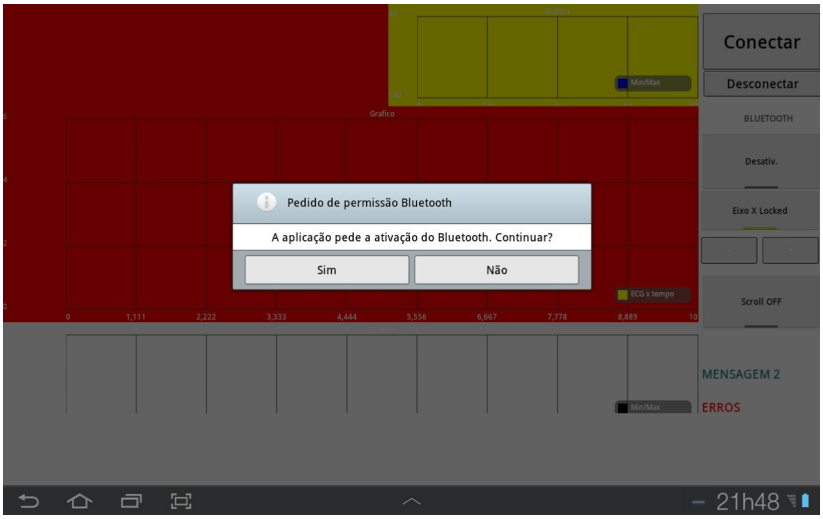

Figura 51 – Tela de inicialização do bluetooth no dispositivo local
Fonte: Autoria própria.

Uma vez aceita a solicitação de acionamento do bluetooth local, a conexão tentará ser feita, conforme ilustrada na figura 52.

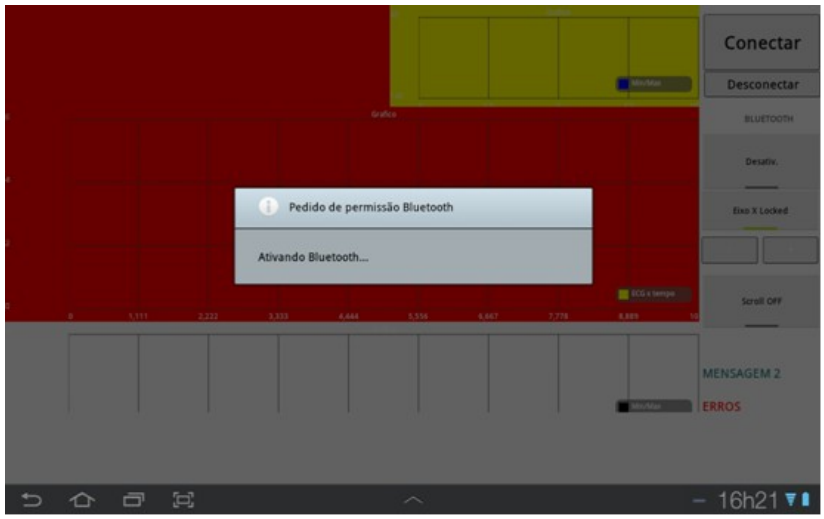

Figura 52 – Tela principal ativando o bluetooth local
Fonte: Autoria própria.

Estabelecida a conexão, a tela principal do aplicativo aparece, ilustrada na figura 53.

Figura 53 – Tela principal do aplicativo local.
Fonte: Autoria própria

Com o aplicativo aberto, pode-se conectar os eletrodos no paciente antes de ligar o lado microprocessado. Estando o paciente conectado liga-se o circuito do MCU e, logo em seguida, pode-se pressionar o botão “CONECTAR” da tela

principal. Desta forma, inicia-se uma busca por todos os dispositivos bluetooth nas imediações e uma tela com os nomes deles surge, conforme apresenta a figura 53.

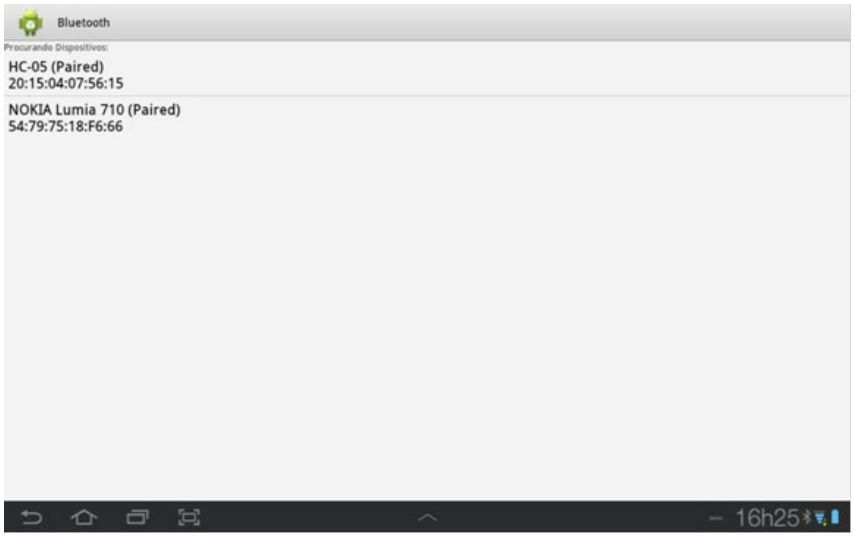

Figura 54 – Tela de seleção do dispositivo bluetooth
Fonte: Autoria própria.

A figura 54 ilustra a maneira como o dispositivo bluetooth usado neste trabalho aparece quando aparece na tela de seleção, surgindo com o nome “HC-05”. Ao se clicar sobre ele, tenta-se realizar a ponte de comunicação bidirecional entre o dispositivo android local e o circuito amostrador de sinal de ECG.

A tela principal apresenta algumas partes bem definidas. O sinal de ECG, quando surgir, será mostrado na área vermelha do display. Há também uma área com comandos à direita, em tom cinza, conforme mostra a figura 55.

Figura 55 – Comandos do aplicativo principal
Fonte: Autoria própria

Os botões disponíveis pela interface android ao usuário tem suas funcionalidades definidas como:

- CONECTAR: Uma vez pressionado, começa uma busca por todos os dispositivos bluetooth nas imediações;
- DESCONECTAR: Desfaz a conexão bluetooth existente para que uma nova busca possa ser feita;
- DESATIV: Este comando tem a função de, uma vez estabelecida a conexão bluetooth, ativa a transmissão de dados ou desativa, caso pressionado novamente. Quando a transmissão é desativada apenas o sinal fica congelado na tela. A conexão bluetooth, entretanto, continua a existir;
- Eixo x *Locked*: Esta função permite ao sinal correr ao longo do tempo pela tela disponibilizando sempre a amostra mais recente do lado direito. Se pressionado novamente aciona o modo de varredura da esquerda para a direita, voltando para a esquerda com a tela limpa quando a plotagem chega do lado direito;
- Campo MENSAGEM2: Apresenta os erros totais percebidos pelo sistema de recepção android;
- Campo ERROS: Apresenta o total de erros corrigidos por retransmissão.

3.8. OPERAÇÃO REMOTA DO SISTEMA

Como mencionado, o aplicativo remoto tem sua operação bastante simplificada. Uma vez inicializado o aplicativo, ele já procura conexão com a rede internet. Os botões de conexão e desconexão ilustrados na figura 56 servem para conectá-lo e desconectá-lo, respectivamente, do servidor parse.

O botão de leitura *off-line* foi simplificado neste trabalho para exibir a última leitura de dados previamente armazenados na nuvem.

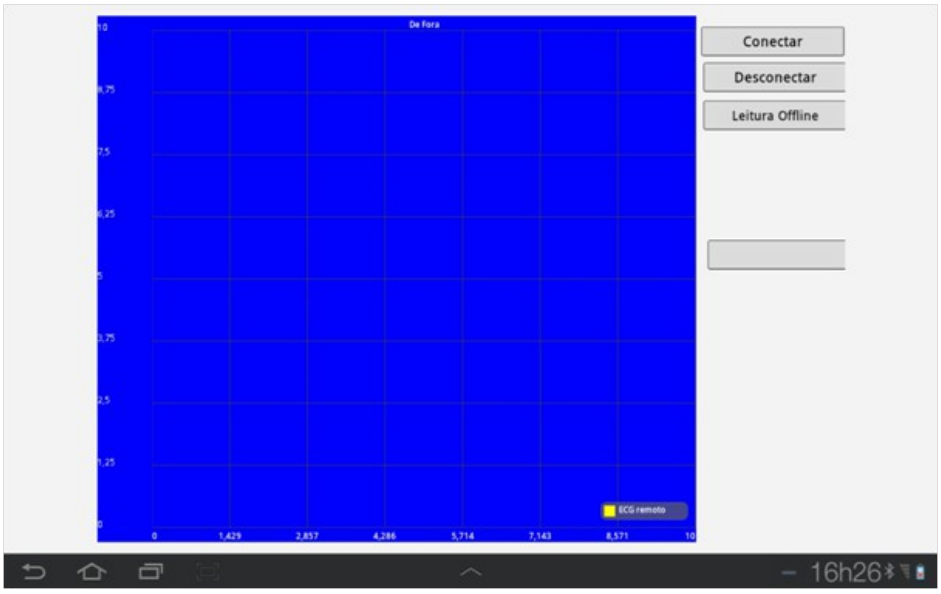

Figura 56 – Tela do aplicativo remoto
Fonte: Autoria própria

Os dados referentes ao último segundo disponível de sinal de ECG que tenham sido armazenados no servidor em nuvem é que serão mostrados no *display* de tela azul. A rotina de dados *online* é um *loop* eterno de verificação do número de sequência presente em cada objeto tipo parse. Caso este número de sequência tenha se alterado, todo o objeto é recuperado e mostrado na tela.

De outro lado, quando se desejar dados sob demanda, os dados *offline* são recuperados através de rotina semelhante. Neste caso os dados de cada segundo de gravação do sinal de ECG em nuvem, representados pelos seus respectivos números de sequência, são recuperados um a um, até que um próximo número de sequência igual a zero seja recebido, correspondente ao primeiro segundo de gravação do próximo registro. Em seguida o sistema apresenta os quadros um após o outro, dando um intervalo de 1000 ms entre eles.

4. RESULTADOS

Para os testes foi usado o gerador de sinal de ECG ilustrado na figura 57. As formas de onda obtidas, assim como a interpretação dos resultados seguem adiante.

4.1. MONTAGEM DO SISTEMA LOCAL

Na figura 57 está ilustrado o simulador de sinais de ECG, cujos três conectores fornecem o sinal das três derivações cardíacas de onde se obtém o sinal de ECG para amostragem.

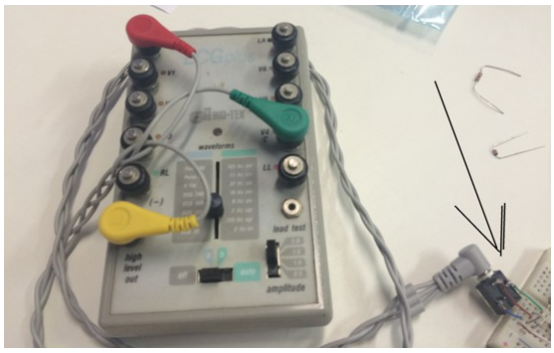

Figura 57 – Simulador de sinais de ECG.
Fonte: Autoria própria

O sinal obtido é adequado de forma a ser aplicado ao conversor ADC do microcontrolador. A amplitude de pico a pico, do sinal amostrado foi condicionada a valores de 1 Vpp para o pino de entrada do conversor ADC do MCU.

A figura 58 replica a mesma ilustração contida no capítulo 8 do sistema microcontrolado. O led vermelho acesso presente na figura 58 representa o microcontrolador em funcionamento.

Depois do sistema ligado, ligou-se um osciloscópio à saída do sinal de ECG já condicionado, porém antes da etapa de amostragem pelo microcontrolador. Este sinal está ilustrado na figura 59.

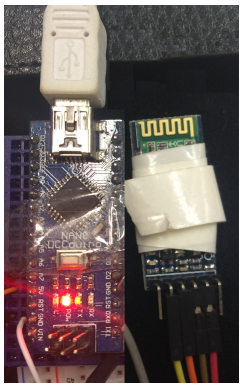

Figura 58 – Módulo microcontrolado e módulo bluetooth HC-05
Fonte: Autoria própria

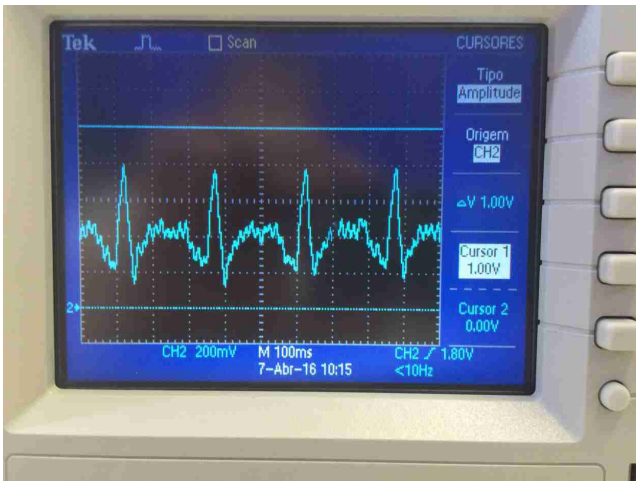

Figura 59 – Tela do osciloscópio com o sinal de ECG a 240 bpm.
Fonte: Autoria própria

Uma vez estabelecida a conexão bluetooth entre o MCU e o tablet que funcionou como dispositivo local, as formas de onda provenientes do gerador de ECG eram transmitidas por bluetooth ao dispositivo local. O tablet era um P7500 da Samsung, com tela de 10 polegadas e sistema android versão 3.2., com resolução de tela de 800 por 1280 pixels.

Uma das leituras do tablet está mostrada na figura 60. O tablet estava em modo de varredura, não estando no modo de leitura contínua. Por isto, no momento da foto a parte direita na estava ainda plotada no display. Leituras mais detalhadas seguem nas próximas figuras.

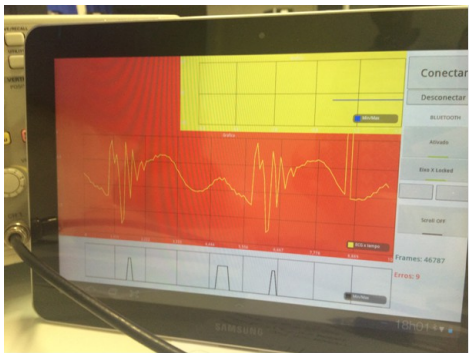

Figura 60 – Tela do tablet Samsung P7500 local, recebendo o sinal de ECG.
Fonte: Autoria própria

Note-se a linha azul com o batimento cardíaco, simulado, constante. Note-se também que erros de transmissão bluetooth também acontecem como está ilustrado na tela do tablet.

A formação completa, ao longo do tempo de 1 segundo de sinal de ECG pode ser vista na figura 61.

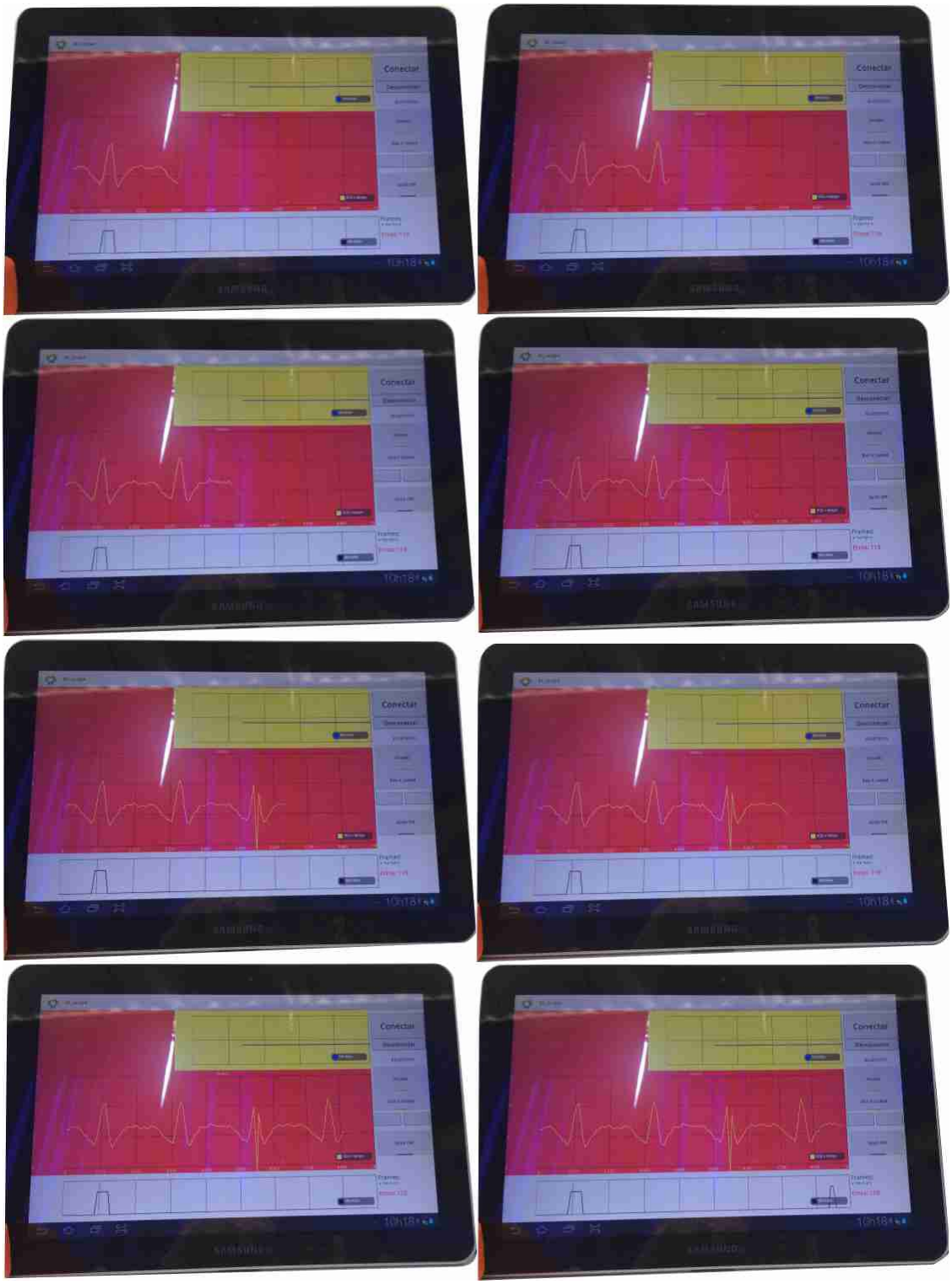

Figura 61 – Um segundo de formação do sinal de ECG no tablet.
Fonte: Autoria própria

Nas figuras 62 a 64 ilustram-se algumas das imagens do osciloscópio com o tablet, simultaneamente.

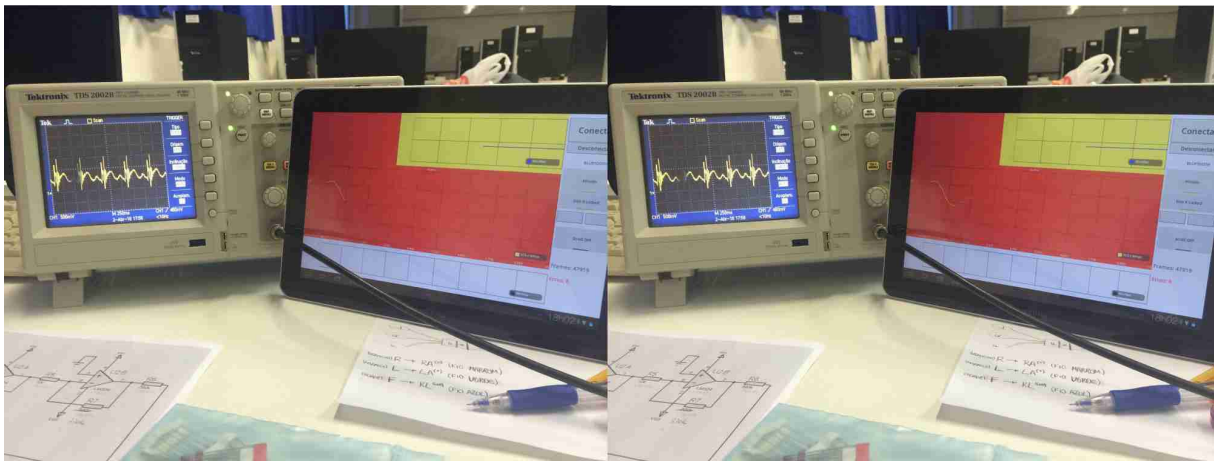

Figura 62 – Snapshots 1 e 2 de um segundo do sinal no tablet.
Fonte: Autoria própria

a seguir ilustram-se apenas algumas das imagens

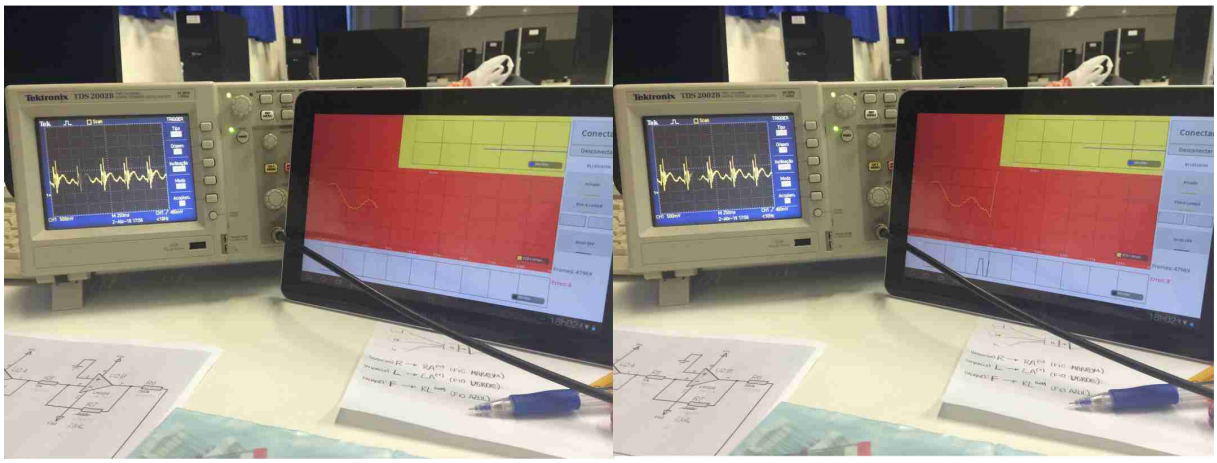

Figura 63 – Snapshots 3 e 4 de um segundo do sinal no tablet.
Fonte: Autoria própria

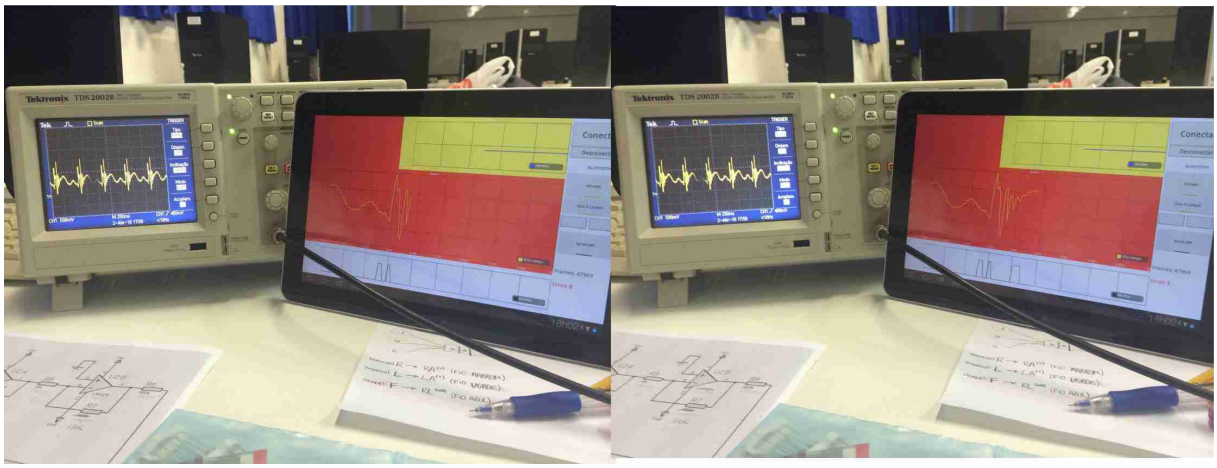

Figura 64 – Snapshots 5 e 6 de um segundo do sinal no tablet.
Fonte: Autoria própria

Em seguida conectou-se o aparelho smartphone à rede internet para que ele servisse como dispositivo remoto. O aparelho usado foi um smartphone MEU AN 400, contendo dois núcleos de 1,2 GHz, com tela de 4 polegadas e resolução 480x800 pixels, e sistema android versão 4.2.2.

Como o smartphone está conectado à rede internet através de WIFI, apenas com o pressionar do botão “Conectar”, o sinal de ECG começa a ser mostrado na tela, conforme apresenta a figura 65.

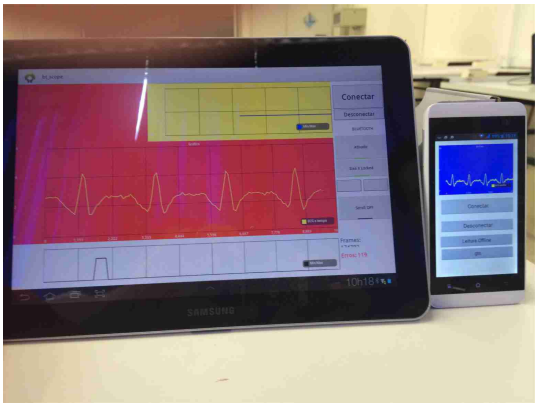

Figura 65 – Snapshot 7 de um segundo do sinal no tablet e no smartphone.
Fonte: Autoria própria

Verificou-se que, alterando-se a frequência de batimento cardíaco proveniente do simulador de sinal de ECG, a resposta apresentada na tela do aplicativo remoto quando colocado, lado a lado, junto ao tablet apresentava um atraso quase imperceptível, porém mensurável. Não chegou-se a medir este atraso devido ao reflexo humano para tal medida ser muito lento perto do atraso ocorrido. Os dois aparelhos estavam conectados no mesmo hub WIFI do laboratório, o que justifica uma transmissão rápida do pacote de informação do aplicativo local até o aplicativo remoto, passando antes através do servidor em nuvem.

Os dados carregados em nuvem, chamados de “ECG4”, permanecem lá até que alguém os apague ou o servidor Parse interrompa o serviço de armazenagem. Foram gravados sinais de ECG com frequência cardíaca variando de 60 a 240 batidas por minuto (BPM).

5. COMENTÁRIOS FINAIS

Este capítulo finaliza o trabalho com conclusões, as dificuldades encontradas durante o desenvolvimento do trabalho, e idéias de continuidade com trabalhos futuros.

5.1. DISCUSSÃO

Para que os objetivos fossem alcançados, várias etapas foram sendo vencidas, referentes aos diversos blocos que fazem parte deste projeto. As dificuldades começam pela etapa de filtragem do sinal de ECG, cuja maior interferência é a do sinal de 60 Hz presente nas linhas da rede elétrica. Outras interferências também se fazem presentes, como as da própria alimentação do laptop quando usada para alimentar o módulo Arduíno. A fonte de alimentação do laptop causava grande atenuação do sinal de ECG amostrado. Até mesmo o cabo de alimentação do laptop, quando simplesmente conectado ao mesmo, ainda que desconectado da rede elétrica, provoca interferências no sinal de ECG medido na entrada do conversor ADC, como se o cabo de alimentação do laptop fosse uma antena captadora de sinais espúrios presentes no ambiente do laboratório.

Apesar dos filtros de 60 Hz e de sua montagem já serem de domínio público, nem sempre a montagem de protótipos funciona de imediato como previamente projetado. Também o sinal presente na saída de módulos captadores de sinal de ECG, como o módulo vendido pela Sparkfun com o AD8232 já embutido e montado conforme recomendação da Analog Devices, não disponibilizou o formato do sinal de ECG obtido na saída do respectivo módulo tão perfeito quanto a propaganda dele mostrava. Foi necessário usar-se de filtros extras, como o passa-altas e os passa-baixas em cascata.

De outro lado, o sistema android também sofreu mudanças em seu ambiente de desenvolvimento. No início, estudou-se sobre o ambiente Eclipse que era o padrão adotado pela Google. Porém desde 2015 mudou-se para o ambiente Android Studio, cuja filosofia de programação, em termos da formação das pastas do projeto, são diferentes. Ainda assim, apesar de ter tido que formatar o laptop contendo os

projetos android dos dois aplicativos, optou-se por manter a instalação do ambiente eclipse com toda a sua configuração feita de maneira manual, do que instalar o Android Studio e ter que reaprender o funcionamento deste novo ambiente.

Também novas interfaces de programação vão sendo lançadas ao longo do ano, as *Application Programm Interfaces* (APIs). Sendo o tablet mais antigo (cerca de 4 anos de uso), optou-se em continuar a usar as bibliotecas android antigas, por exemplo, que controlam o hardware do bluetooth tanto no tablet quanto no smartphone, mas que continuam válidas, ainda que o tablet contenha um android 3.2., e o smartphone contenha um android 4.2.2. Novas bibliotecas são lançadas para android 5 e 6, porém tudo o que foi feito para rodar desde os mais antigos, continuam a rodar nos mais novos. O raciocínio reverso, contudo, não é válido.

Enfim, a tecnologia é algo em constante evolução, mudanças, atualizações e desenvolvimento visando maior eficiência e aplicabilidade em termos gerais.

5.2. CONCLUSÔES

Algumas etapas poderiam ser melhoradas no projeto, como por exemplo a robustez do software nos dispositivos móveis, por exemplo. Contudo, isto está sugerido adiante.

De acordo com o que está originalmente proposto no capítulo 1, tem-se os vários objetivos que foram sendo concluídos um a um.

Se fez necessário um módulo bluetooth flexível quanto à sua possibilidade de configuração por software, de modo que ele funcione de maneira apropriada de acordo com as velocidades de transmissão e recepção de dados vindos do microcontrolador. Fez-se o módulo bluetooth HC-05 trabalhar a aproximados 115.000 bits por segundo, o que o possibilita transmitir o *frame* de 104 bits em tempo suficiente entre as amostras e ainda aguardar por respostas no mesmo *frame*. É um módulo de baixo custo, facilmente encontrado no mercado, e facilmente configurável e interfaceável.

O sistema microcontrolado é de fácil aquisição, além de baixo custo, o que facilita seu uso em meios acadêmicos futuros para estudantes que eventualmente se interessem em melhorar o sistema aqui proposto. O sistema de ADC possui

resolução de 10 bits, porém foi útil, pois a finalidade deste projeto envolve a confiabilidade da transmissão dos dados.

O sistema de condicionamento do sinal de ECG envolve um tratamento bastante conhecido pela respectiva literatura, tratando-se de circuitos seletores de frequência a saber: um filtro passa-altas de primeira ordem cascateado com um filtro passa-baixas de quarta-ordem em configuração Sallen-Key, com função de aproximação de Butterworth, e frequência de corte de 40 Hz.

O software desenvolvido como aplicativo local levou mais tempo para ser terminado até porque teve que ser aprendido sobre o funcionamento do sistema operacional android desde o início deste trabalho. O aplicativo local realiza todas as funções necessárias, inicialmente propostas para ele desempenhar. O software faz a conexão bluetooth com a interface do módulo bluetooth HC-05, faz sua leitura, assim como também envia informações quanto aos erros ocorridos na recepção, recupera as amostras do *frame* transmitido, monta-os em um pacote maior a ser enviado para a nuvem, representa-os na tela do aplicativo principal e os envia para um servidor de dados em nuvem. Isto pode ser verificado pelo simples acesso aos servidores Parse, através do *login* do autor deste trabalho.

No lado do aplicativo remoto, pode-se constatar a autenticidade dos dados enviados para a nuvem através do envio de sinal de ECG, constatando-se que, através da mudança de frequência ou de forma do sinal transmitido, a mesma alteração se verifica no aplicativo remoto, porém com um atraso de transmissão mínimo enquanto os dois aplicativos usarem um mesmo hub de internet.

Conforme é visto na figura 64, o sinal de ECG é recebido com qualidade suficiente para se poder determinar o complexo QRS, o que ilustra que os filtros estão projetados a contento do projeto que trata de sinais de ECG.

O sistema de armazenamento em nuvem é uma das variáveis mais importantes para o funcionamento do projeto envolvido neste trabalho, quando o assunto envolve acesso remoto. Além disto, a tecnologia está em evolução constante. Por isto, deve-se estar atento ao surgimento de outras provedores do serviço em nuvem, talvez com ainda maior facilidade de uso. Existem vários outros sistemas de armazenagem em nuvem surgindo para que alternativas sejam possíveis à medida que eventuais custos de operação de armazenagem em nuvem a tornem inviáveis para meios acadêmicos. O sistema escolhido da Parse se mostra confiável, rápido, e com bastante flexibilidade quanto à maneira do envio de dados

para seus servidores, dispondo de uma biblioteca de classes e métodos bastante ampla. Além disto, o sistema disponibilizado pela Parse é bastante transparente quanto à visualização dos dados lá armazenados.

Existem várias pesquisas recentes envolvendo sinais vitais, tais como o sinal de ECG, computação móvel, computação na nuvem e, IoT. Porém são ainda poucos os que disponibilizam os dados em servidores remotos. Hsieh *et al.* (2013) levantou uma pesquisa mostrando um resumo dos vários estudos pelos quais os dispositivos móveis fazendo uso de MCC podem auxiliar diagnósticos de eletrocardiograma, ecocardiograma, além dos exames de imagens para médicos fora dos seus consultórios. Mohammed *et al.* (2014) desenvolveram um aplicativo para leitura *online* do sinal de ECG enviando os dados para uma nuvem particular do usuário para futura análise. Os dados eram apenas enviados, porém sem controle do sinal no tempo. Hsieh *et al.* (2012) desenvolveu um sistema de tele-diagnose para o sinal de ECG de doze terminais, dando ênfase para o uso em áreas afastadas dos grandes centros. Neste estudo o sistema transmitia *online* com doze fios, porém sem o uso de armazenamento das nuvens.

5.3. TRABALHOS FUTUROS

Com o passar do tempo, vai-se percebendo que muito há para se implementar nos aplicativos, tanto no remoto, quanto no local, com a finalidade de torná-los aplicativos mais robustos para o uso diário. Mais agilidade na conexão bluetooth, por exemplo, pode ser feita, agilizando-se a maneira como o bluetooth se conecta entre o módulo bluetooth e o dispositivo android sem a necessidade de fazê-lo manualmente.

Um trabalho futuro poderia tratar da criptografia dos sinais antes de enviá-los para a nuvem, da mesma forma como decifrá-los da maneira correta no aplicativo remoto.

Outro trabalho importante envolve o uso racional dos servidores em nuvem e do canal de comunicação de dados. Com o passar do tempo, mais e mais dados tendem a serem armazenados em nuvem, então uma maneira de se comprimir estes dados de forma a otimizar tanto o uso dos servidores em nuvem, como otimizar o

uso dos canais de transmissão de dados de faz necessário. Tudo se traduz em economia de custos futuros.

REFERÊNCIAS BIBLIOGRÁFICAS

Ahlstrom, M.L., Tompkins, W.J. (1983) "Automated High-Speed Analysis of Holter Tapes with Microcomputers", IEEE Transactions on Biomedical Engineering, volume BME-30, issue 10, pp. 651–657.

Akkaya, K., Younis, M., (2005) A survey on routing protocols for wireless sensor networks, Ad Hoc Networks, 3(3), pp. 325–349.

Akyildiz, I.F., Stuntebeck, E.P., (2006) Wireless underground sensor networks:research challenges, Ad Hoc Networks, 4(6), pp. 669–686.

Akyildiz, I., Vuran, M.C., (2010) Wireless Sensor Networks, John Wiley & Sons Inc, New York, NY, USA.

Alecrim, E. (2008) Tecnologia Bluetooth. Disponível em: http://www.infowester.com/ bluetooth.php

Anastasi, G. Conti, M. Di Francesco, M., (2009) Reliable and energy-efficient data collection in sparse sensor networks with mobile elements, Perform. Eval. 66(12), pp. 791–810.

ANDROID DEVELOPERS, INC. (2015) Disponível em: http://developer.android.com/about/ index.html.

Ateniese, R., Brurns, R., Curtmola, J., Herring, L., Kisner, Z., Peterson and D. Song, (2007) “Provable Data Possession at Untrusted Stores.”, ACM Conference Computer and Communication Security (CCS´07), pp. 598-609.

ATMEL (2016) ATMEL 8-BIT MICROCONTROLLER, http://www.atmel.com /images/atmel-8271-8-bit-avr-microcontroller-atmega48a-48pa-88a-88pa-168 a-168pa-328-328p_datasheet_summary.pdf, acessado em 02/04/2016.

Balda, R.A., Diller, G., Deardorff, E., Doue, J., Hsieh, P. (1977) "The HP ECG Analysis Program", in: IFIP Working Conference Trends in Computer Processed Electrocardiograms, North Holland Publish, pp.197–204.

Barbosa, P.R.B. (2003) "Efeitos da Ponderação da Média Coerente e da Filtragem na Detecção de Potenciais Tardios Ventriculares no Eletrocardiograma de Alta Resolução", tese, Universidade Federal do Rio de Janeiro,193 páginas.

Benmalek, M., Charef, A., Abdelliche, F. (2010) "Preprocessing of the ECG Signals using the His-Purkinje Fractal System", in: 7th International Multi-Conference on Systems Signals and Devices, pp.1–5.

BLUETOOTH SIG. (2015) Disponível em: <http://www.bluetooth.com/ Pages/Bluetooth-Home.aspx>.

Boldrini, C., Lee, K., Önen, M., Ott, J., Pagani, E., (2014) Opportunistic networks, Computer Communications, 48, pp. 1–4.

Borgia, E. (2014) “The Internet of Things vision: Key features, applications and open issues”, in Computer Communications, pp.1-31.

Bornstein, D. (2008) “Dalvik VM Internals”, disponível: www.youtube.com/watch? v=p tjedOZEXPM

Bray, J., Sturman, C. F., (2001) Bluetooth – Connect Without Cables. 1.ed. Upper Saddle River, New Jersey, Printice Hall.

Bruno, R., Conti, M., Gregori, E., (2002) Bluetooth: architecture, protocols and scheduling algorithms, Cluster Computing, 5 (2), pp. 117–131.

Bruno, R., Conti, M. ,Gregori, E., (2005) Mesh networks: commodity multihop ad hoc networks, IEEE Communications Magazine, 43 (3), pp. 123–131.

Carvalho, A.P., Hoffman, B.F., Carvalho, M.P. (1969) "Two Components of the Cardiac Action Potential", The Journal of General Physiology, volume 54, pp. 607–635.

Chen, M., Leung, V.C.M., Hjelsvold, R., Huang, X., (2012), Smart and interactive ubiquitous multimedia services, Computer Communications, 35 (15), pp. 1769–1771.

Conti, M., Chong, S., Fdida, S., Jia, W., Karl, H., Lin, Y.-D., Mähönen, P., Maier, M., Molva, R., Uhlig, S., Zukerman, M. (2011) Research challenges towards the future internet, Computer Communications, 34 (18), pp. 2115–2134.

Conti, M, Das, S K., Bisdikian, C., Kumar, M., Ni, L.M., Passarella, A., Roussos, G., Tröster, G., Tsudik, G., Zambonelli, F., (2012), Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber-physical convergence, Journal Pervasive and Mobile Computing, vol. 8, Issue 1, pg 2–21.

Conti, M., (2014) Computer communications: present status and future challenges, Computer Communications, 37, pp. 1–4.

Conti, M., Giordano, S., (2014) Mobile ad hoc networking: milestones, challenges, and new research directions, IEEE Communications Magazine, 52 (1), pp. 85–96.

CREATIVE (2015) CREATIVE COMMONS, Funcionamento do Bluetooth. Kioskea.net < http://pt.kioskea.net/contents/bluetooth/bluetooth-fonctionnement.php3 >, acesso em 01/04/2016.

Demirkol, I., Ersoy, C., Alagoz, F., (2006) Mac protocols for wireless sensor networks: a survey, IEEE Communications Magazine, 44 (4), pp. 115-121.

Di Francesco, M., Das, S.K., Anastasi, G. (2011) Data collection in wireless sensor networks with mobile elements: a survey, ACM Trans. Sensor Networks, 8 (1), pp. 7:1–7:31.

Domingo, M.C., Vuran, M.C., (2012) Cross-layer analysis of error control in underwater wireless sensor networks, Computer Communications, 35 (17), pp. 2162–2172.

EPCGLOBAL (2011), GS1 EPC Tag Data Standard 1.6, disponível em <http://www.gs1.org/gsmp/kc/epcglobal/tds/tds16-RatifiedStd-0110922.pdf>, acesso em 01/04/2016.

Engelse, W.A.H., Zeelenberg, C. (1979) "A Single Scan Algorithm for QRS-Detection and Feature Extraction", Computers in Cardiology, volume 6, pp. 37–42.

Espada, J.P., Díaz, V.G., Crespo, R.G., Martínez, O.S., G-Bustelo, B.C.P., Lovelle, J.M.C., (2015) Using extended web technologies to develop Bluetooth multi-platform mobile applications for interact with smart things, Journal Information Fusion, vol. 21, pp. 30-41.

Fraden, J., Neuman, M.R. (1980) "QRS Wave Detection", Medical and Biological Engineering and Computing, vol. 18, pp. 125–132.

Galli, S., Scaglione, Wang, A. Z., (2010) Power line communications and the smart grid, in: Proceedings of First IEEE Smart Grid Communications (SmartGridComm), pp. 303–308.

Gama, K., Touseau, L., Donsez, D. (2012) Combining heterogeneous service technologies for building an Internet of Things middleware, Computer Communications, vol. 35, Issue 4, pp. 405–417.

Garcia-Morchon, O., Kuptsov, D., Gurtov, A., Wehrle, K., (2013) Cooperative security indistributed networks, Computer Communications, 36 (12), pp. 1284–1297.

Guimarães, J.I., Moffa, P.J., Uchida, A.H., Barbosa, P.B. (2003) "Normatização dos Equipamentos e Técnicas para a Realização de Exames de Eletrocardiografia e Eletrocardiografia de Alta Resolução", Arquivos Brasileiros de Cardiologia, volume 80, número 5, págs. 225–234.

Gustavson, D., Willsky, A.S., Mitter, S.K., Wang, J.Y., Akant, A., Kessel, W.C., Doerschuk, P.C. (1977) “Automated VCG Interpretation Studies Using Signal Analysis Techniques, Charles Stark Draper Lab. Report R-1044, Cambridge, Massachusetts.

Guyton, A.C. (1992) "Tratado de Fisiologia Médica", 8ª edição, Editora Guanabara Koogan.

Hartenstein, H., Laberteaux, K. (2008) A tutorial survey on vehicular ad hoc networks, IEEE Communicationns Magazine, 46 (6), pp. 164–171.

Hecht, H.H. (1973) "Atrioventricular and Intraventricular Conduction: Revised Nomenclature and Concepts", The American Journal of Cardiology, volume 31, issue 2, pp. 232–244.

Hodgkin, A.L., Huxley, A.F. (1952) "A Quantitative Description of Membrane Currents and its Application to Conduction and Excitation in Nerve", The Journal of Physiology, volume 117, issue 4, pp. 500-544.

Holsinger, W.P., Kempner, K.M., Miller, M.H. (1971) "A QRS Preprocessor Based on Digital Differentiation", IEEE Transactions on Biomedical Engineering, volume BME-18, issue 3, pp. 212–217.

Hsieh, J., Hsu, M., (2012), A cloud computing based 12-lead ECG telemedicine service, US National Library of Medicine – National Institute of Health , doi 10.1186/1472-6947-12-77

Hsieh, J., Li, A-H., Yang, C-C. (2013) Mobile, Cloud, and Big Data Computing: Contributions, Challenges, and New Directions in Telecardiology, International Journal of Environmental Research and Public Health, doi: 10.3390/ijerph10116131

ITU (2015). Report on Internet of Things: Executive Summary: https://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf.

IETF (2016) RFC 2246. Disponível: https://www.ietf.org/rfc/rfc2246.txt Acessado: 29/03/2016.

James, T.N. (1963) "The Connecting Pathways Between the Sinus Node and the A-V Node and Between the Right and Left Atrium in the Human Heart", American Heart Journal, volume 66, issue 4, pp. 498–508.

Juels, A., Kaliski Jr., B.S., (2007) Pors: proofs of retrievability for large files, in: Proceedings of the 14th ACM conference on Computer and Communications Security, pp. 584-597.

Kanmani, P., Anusha, S., (2015) “A Novel Integrity Scheme for Secure Cloud Storage”, IEEE Ninth International Conference on Intelligent Systems and Control, pp. 1-3.

Khan, A. M., Ahmad, S., Haroon, M., (2015) “A Comparative Study of Trends in Securitiy in Cloud Computing”, IEEE Fifth International Conference on Communication Systems and Network Technologies, pp. 586-590.

Kjeldskov, J., Skov, M.B., (2001) Supporting work activities in healthcare by mobile electronic patient records. Lecture Notes in Computer Science, pp. 191-200.

Lewis, T., Oppenheimer, B.S., Oppenheimer, A. (1910) "The Site of Origin of the Mammalian Heart Beat: the Pacemaker in the Dog", Heart, volume 2, pp. 147–169.

Lecheta, Ricardo R., (2010) Google Android: aprenda a criar com aplicações para dispositivos móveis com Android SDK, 2. Ed., São Paulo: Novatec Editora.

Malmivuo, J., Plonsey, R., (1995) "Bioelectromagnetism - Principles and Applications of Bioelectric and Biomagnetic Fields", Oxford University Press, New York, 1st edition.

Miller, B.A.; Bisdikian, C; (2001) Bluetooth Revealed. 1.ed. Upper Saddle River, New Jersey: Prentice Hall.

Mohammed , J., Thakral, A., Ocneanu, A. F. (2014) Internet of Things: Remote Patient Monitoring Using Web Services and Cloud Computing, IEEE International Conference on Internet of Things (iThings), DOI: 10.1109/iThings.2014.45

Morimoto, Carlos Eduardo. (2008) Redes, guia prático. Porto Alegre: Sul Editores.

Okada, M., (1979) "A Digital Filter for the QRS Complex Detection", IEEE Transactions on Biomedical Engineering, volume BME-26, issue 12, pp. 700–703.

PARSE (2015), Parse Cloud Database, disponível em https://parse.com.

Paulo, Jean Vitor de. (2014) Desenvolvimento de um aplicativo Android e de uma interface bluetooth para um dinamômetro biomédico. 93 f. Dissertação (mestrado) - Universidade Estadual Paulista Júlio de Mesquita Filho, Faculdade de Engenharia de Ilha Solteira.

Park, K.-J., Zheng, R., Liu, X., (2012) Cyber-physical systems: milestones and research challenges, Computer Communications, 36 (1), pp 1–7.

Pollard, J.K., Rohman, S., Fry, M., (2001) A web-based mobile medical monitoring system. International Workshop on Intelligent Data Acquisition on Advanced Computing Systems. Technology and Aplications, IEEE Press.

Poovendran, R., (2010) Cyber-physical systems: close encounters between two parallel worlds, Proc. IEEE, 98 (8), pp. 1363–1366.

Priess, W., Resende, J. F. de, Pirmes, L., Carmo, L. F. R. da C. (Um Mecanismo de Escalonamento Parametrizavel para Scatternets Bluetooth, XXI Simpósio Brasileiro de Redes de Computadores, pp 5-20.

PROJECT. (2015), Android open source. Disponível em: http://source.android.com , acesso em 01/04/2016.

PROJECT (2016), Cloud of Things for empowering the citizen clout in smart cities, 2013-2016 <http://clout-project.eu/>

Queiroz, L. E. C. (2008) Protocolo de Redes Bluetooth. Distribuição e Integração de Sistemas. Barcarena.

Rieira, A.R., Ferreira, C., Ferreira Filho, C., Dubner, S., Schapachnik, E., Uchida, A.H., Moffa, P.J., Zhang, L., Luana, A.B. (2008) "Wellens Syndrome Associated with Prominent Anterior QRS Forces: an Expression of Left Septal Fascicular Block", Journal of Electrocardiology, volume 41, issue 6, pp. 671–674.

Sanches, D. (2003) "Interferência Eletromagnética", Editora Interciência, 124 páginas.

Sörnmo L. (1993), Time-varying digital filtering of ECG baseline wander. Med Biol Eng Comput. 31(5), pp.503-508.

Turner, D., Wilhelm, R., Lemberg, W., (2006) “The Free Type Project Freetype 1.”, disponível em http://freetype.sourceforge.net/freetype1/index.html, acesso em 01/04/2016.

Uckelmann, D., Harrison, M., Michahelles, F., (2011), Architecting the Internet of Things, vol. 1, Springer Verlag, Berlin Heidelberg, Chapter. An Architectural Approach Towards the Future Internet of Things.

Vinh, T.L., Bouzefrane, S., Farinone, J-M., Attar, A., Kennedy, B.P. (2015) “Middleware to Integrate Mobile Devices, Sensors and Cloud Computing” , in The 6th International Conferencce on Ambient Systems, Networks and Technologies (ANT-2015), Procedia Computer Science, vol. 52. pp. 234-243.

Wald, A., Stone, J.G., Khambatta, H.J., (1990) Plastic Induced ECG Noise On Cardiopulmonary Bypass, Proceedings of the Twelfth Annual International Conference of the IEEE Engineering in Medicine and Biology Society, pp. 1998-1999.

Wang, X., Zhong, S., Zhou, R., (2012) A mobility support scheme for 6 lowpan, Computer Communications, vol. 35 (3), pp. 392–404.

Wang, S.-S., Lin, Y.-S., (2013) Passcar: a passive clustering aided routing protocol for vehicular ad hoc networks, Computer Communications, vol 36 (2), pp. 170–179.

WAVESEN (2016) Wavesen, disponível em wavesen.com, acesso em 01/04/2016.

Webster, J.G. (2008) "Encyclopedia of Medical Devices and Instrumentation", volume 2, Wiley-Interscience.

Zhang, F., Lian, Y. (2009) “QRS Detection Based on Multiscale Mathematical Morphology for Wearable ECG Devices in Body Area Networks”, IEEE Transactions on Biomedical Circuits and Systems, volume 3, issue 4, pp. 220–228.