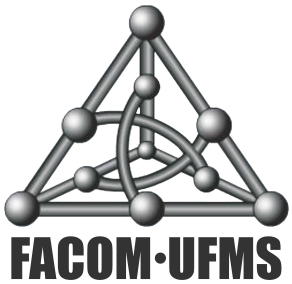
Dissertação de Mestrado

# BEP 1.0: Aplicativo Android para visualização e armazenamento de informações fisiológicas e ambientais

João Paulo Gomes de Andrade

**Orientação:** Prof. Dr. Evandro Mazina Martins
**Co-orientação:** Dra. Fabiana Villa Alves

Mestrado Profissional em Computação Aplicada
Área de Concentração: Tecnologias Computacionais para Agricultura e Pecuária




Faculdade de Computação
Universidade Federal de Mato Grosso do Sul
16 de Fevereiro de 2017

BEP 1.0: Aplicativo Android para visualização e armazenamento de informações fisiológicas e ambientais

Campo Grande, 16 de fevereiro de 2017

Banca examinadora:

- Professor Dr. Evandro Mazina Martins (FAENG/UFMS) - Orientador
- Pesquisadora Dra. Fabiana Villa Alves (Embrapa Gado de Corte) - Coorientadora
- Professor Dr. Milton Ernesto Romero Romero (FAENG/UFMS)
- Professor Dr. Ricardo Ribeiro dos Santos (FACOM/UFMS)

# Agradecimentos

Agradeço primeiramente a Deus, por minha vida e pelas oportunidades que tive.

Aos meus pais pela educação dada para os seus três filhos, onde nunca nos deixou faltar nada, sempre nos apoiando para fazermos tudo da melhor maneira possível.

Aos meus irmãos que sempre me apoiaram.

Á minha noiva Érica pela paciência e companheirismo nos momentos de dificuldade.

Aos professores Dr. Evandro Mazina Martins e Dr. Milton Ernesto Romero Romero pela dedicação e apoio para a conclusão deste trabalho.

Ao professor Dr. Ricardo Ribeiro dos Santos pela presteza e ajuda nos momentos de necessidade.

À pesquisadora Dra. Fabiana Villa Alves pelos ensinamentos, motivação e sugestões para o desenvolvimento deste trabalho.

Aos meus colegas do LSCAD, principalmente aqueles que atuaram diretamente para a conclusão do projeto e que sempre estiveram a disposição para me ajudar, Patrik, Cleison,

Emílio, Rafael, Leonardo, Ricardo.

Aos professores da FACOM pelos conhecimentos transmitidos.

À UFMS pela qualidade na educação prestada à sociedade.

À Embrapa Gado de Corte por todo o apoio e suporte, principalmente pelos profissionais qualificados que contribuíram de forma significativa nos trabalhos.

A todos os amigos e pessoas que me ajudaram e colaboraram na construção deste trabalho.

# Resumo

Nos tempos atuais, o monitoramento animal se faz necessário para melhorar a produção bovina garantindo o bem-estar animal. Devido à isso, existem hoje em dia, diversas soluções de monitoramento animal. Contudo, tratando de animais não-sedados e não-contidos, poucas são as soluções existentes, pois não é uma tarefa fácil. Este trabalho propõe a criação de um aplicativo *mobile*, para *smartphones* e *tablets* que contêm o sistema operacional Android, capaz de transmitir, receber e calcular informações fisiológicas e ambientais de sensores presentes em bovinos não-sedados e não-contidos, como as frequências cardíaca e respiratória, e visualizá-las de forma fácil e eficaz por parte do criador, podendo assim ter um maior controle em relação ao seu rebanho. Para a obtenção das frequências cardíaca e respiratória, algoritmos foram utilizados para estimar esses dados. Através dos *samples* capturados por um sensor oxímetro e desses algoritmos consegue-se obter uma estimativa da quantidade de batimentos cardíacos e da quantidade de respirações por minuto dos bovinos. Os sensores necessários para a aquisição das variáveis fisiológicas e ambientais bovinas fazem parte de um sistema eletrônico constituído de software e hardware embarcado, desenvolvido no escopo de outros projetos dentro do grupo de pesquisa. Parte dos projetos e desenvolvimentos realizados no âmbito deste trabalho constituíram um pedido de patente de número BR10201600222.

# Abstract

In the current times, animal monitoring is necessary to improve a bovine production guaranteeing animal welfare. Due to this, there are nowadays, several animal monitoring solutions. However, dealing with non-sedated and non-contained animals, few are as existing solutions, as it is not an easy task. This work proposes the creation of a mobile application, for smartphones and tablets that contain the Android operating system, capable of transmitting, receiving and calculating physiological and environmental information of sensors present in non-sedated and non-contained cattle, such as Respiratory, and visualize them in an easy and effective way on the part of the creator, thus as well as a greater control in relation to his herd. To obtain heart and respiratory frequencies, the algorithms were used to estimate these data. Through the samples captured by an oximeter sensor and the algorithms an estimate of the number of heart beats and the number of breaths per minute of the cattle can be obtained. The sensors necessary for the evaluation of the physiological and environmental variables of bovines are part of an electronic system consisting of embedded software and hardware, developed without the scope of other projects within the research group. Part of the projects and developments carried out within this work constituted a patent application number BR10201600222.

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Capítulo 1

# Introdução

Segundo a Associação Brasileira das Indústrias Exportadoras de Carnes, **ABIEC**, no cenário atual mundial, o Brasil é um dos principais produtores e exportadores de carne bovina. Além disso, o mercado global está cada vez mais exigente no que diz respeito à qualidade da carne e ao armazenamento de informações sobre os animais desde o nascimento até o momento do abate. Dessa forma, a demanda internacional pede por um sistema que respeite o animal prezando o bem-estar do nascimento ao abate, pois atualmente o bem-estar animal, junto às questões de segurança alimentar, são considerados como um dos maiores desafios da pecuária e agricultura atual. Assim, para ajudar a garantir esta qualidade necessária na carne e o bem-estar animal, o uso da tecnologia da informação se torna uma aliada no manejo dos rebanhos, visando aumentar o controle sobre esses animais. Portanto, soluções tecnológicas podem ser desenvolvidas, com o intuito de facilitar a extração e visualização de informações à respeito dessas questões tão importantes. O monitoramento animal surgiu como solução para estas exigências. Um

modelo ideal seria onde cada animal da propriedade fosse monitorado de forma constante e automatizada, colhendo informações úteis aos produtores, evitando situações as quais possam afetar a qualidade da carne e o bem-estar animal.

Diante do cenário apresentado e da real necessidade de acompanhamento dos animais por meio dos produtores, este projeto tem como objetivo principal o desenvolvimento de um aplicativo *mobile* para *smartphones* e *tablets* com o sistema operacional *Android*, capaz de receber, armazenar, visualizar, processar e calcular dados capturados em bovinos, de forma *online*, através de uma solução integrada de *hardware* e *software*, não invasiva, desenvolvida por outros projetos de acadêmicos de mestrado profissional em computação aplicada da Universidade Federal de Mato Grosso do Sul [1][2]. Este sistema eletrônico, integrando *hardware* e *software* é composto de sensores avançados, adaptados e acoplados à pele dos animais a fim de obtenção dos sinais referentes às variáveis fisiológicas, como temperatura, frequência cardíaca e frequência respiratória, além de variáveis ambientais, para auxiliar a identificação de estresse térmico em bovinos. Junto a ele, tem-se a utilização de *hardware* embarcado que controla os sensores, obtendo as amostras capturadas, para então transmití-las para o aplicativo, que processará os dados possibilitando a visualização de forma intuitiva para o usuário final.

Com um aplicativo *mobile* capaz de receber, realizar a configuração do *hardware* definindo a frequência e os períodos de captura das amostras, processar e visualizar as informações de forma *online*, será possível um monitoramento animal eficaz e a obtenção

de respostas mais completas e precisas sobre o comportamento dos animais analisados.

Na Figura 1.1 encontra-se esquematizado o sistema completo.

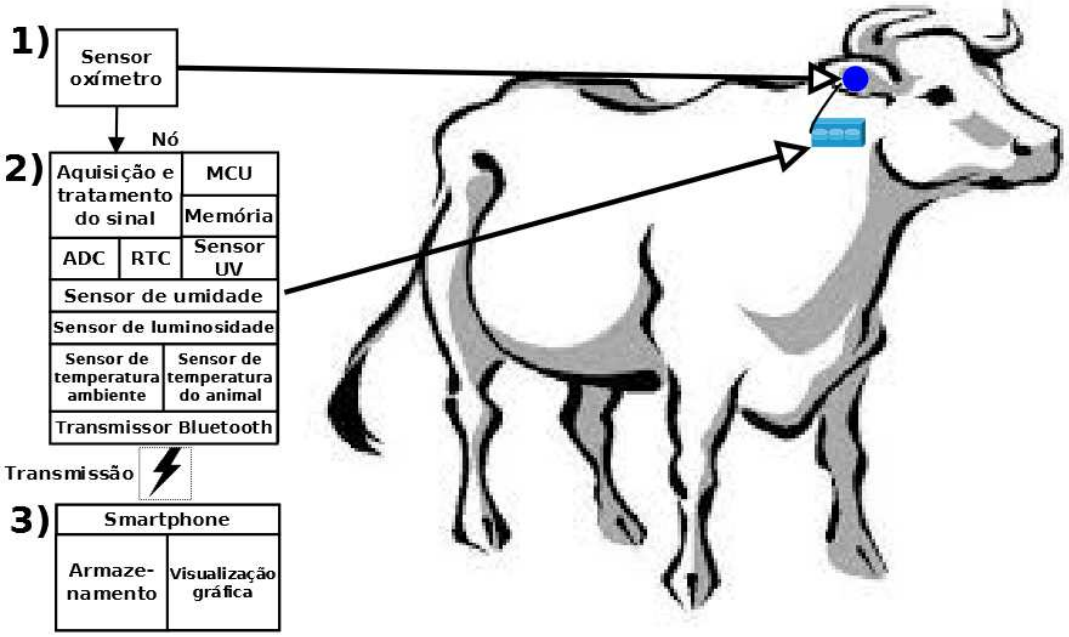



Figura 1.1: Figura esquemática do sistema [1]

Ressalta-se que as atividades do desenvolvimento da solução de *hardware* e *software*, além dos sensores e da placa contendo o microcontrolador utilizados no projeto, foram desenvolvidas no âmbito de outros trabalhos no grupo do projeto. Especificamente neste trabalho, foi desenvolvido um aplicativo *mobile* com as características citadas anteriormente.

## 1.1 Organização do Trabalho

Os capítulos desta dissertação de mestrado estão organizados da seguinte forma:

- O Capítulo 2 apresenta a revisão bibliográfica sobre as tecnologias utilizadas no desenvolvimento do aplicativo *mobile*, além da motivação para suas escolhas.

- No Capítulo 3 é apresentado detalhadamente o desenvolvimento do aplicativo em sua versão 1.0.

- Por fim, o Capítulo 4 apresenta as considerações finais deste trabalho, as dificuldades encontradas e propostas de trabalhos futuros.

# Capítulo 2

# Revisão Bibliográfica

Este capítulo abordará a revisão bibliográfica envolvida nesta dissertação. Nas seções 2.1 e 2.1.1 são apresentadas a tecnologia de comunicação sem fio *Bluetooth* e o módulo *Bluetooth* utilizado no projeto, tal como suas características e limitações. Na seção 2.2 é apresentado o sistema operacional Android. Em 2.3 são apresentadas as motivações para as escolhas destas tecnologias no projeto.

## 2.1 *Bluetooth*

*Bluetooth* é uma tecnologia de comunicação sem fio simples e segura. Surgiu da cooperação de várias empresas tendo como finalidade a comunicação entre dispositivos utilizando baixo consumo de energia e eliminando cabos de conexão entre eles. Suas principais características são: [3]

- Baixo consumo de energia, pois trabalha em baixas tensões;

- Suporta transmissões de voz e dados;
- Tem curto alcance;
- *Hardware* com dimensões reduzidas;
- Baixo custo;

As conexões entre dispositivos eletrônicos habilitados para *Bluetooth* permitem que esses dispositivos se comuniquem sem fio através de curto alcance, geralmente de 1 à 100 metros, dependendo de sua versão, usando transmissões de rádio nas faixas de 2,4 GHz até 2,485 GHz[4]. Para evitar interferências com outras fontes de rádio frequência, o *Bluetooth* utiliza a técnica de *frequency hopping spread spectrum*. Esta técnica divide a faixa de frequência disponível em 79 canais de 1 MHz cada. Desta forma, os dados que serão transmitidos são divididos em pacotes que são enviados cada um em um canal diferente, através de saltos. A sequência desses saltos é única para a *piconet*[1] e é definida pelo dispositivo mestre. Esta mudança de canais ocorre em até 1600 vezes por segundo[6]. Como o meio de transmissão é o ar, qualquer dispositivo poderia receber as informações transmitidas. Entretanto, a técnica *frequency hopping spread spectrum* garante que essas informações transmitidas entre os dispositivos sejam seguras, pois é necessário ter o conhecimento prévio dos saltos que serão dados na frequência, ou seja, somente quem conhece os saltos (dispositivos dentro da *piconet*) conseguem receber os dados corretamente. A

[1]É uma rede *Bluetooth* com topologia dinâmica, onde estão conectados dois ou mais dispositivos[5].

Figura 2.1 demonstra a técnica *frequency hopping spread spectrum*. Nela é possível ver que cada espaço de frequência é utilizado por vários canais (A-D).

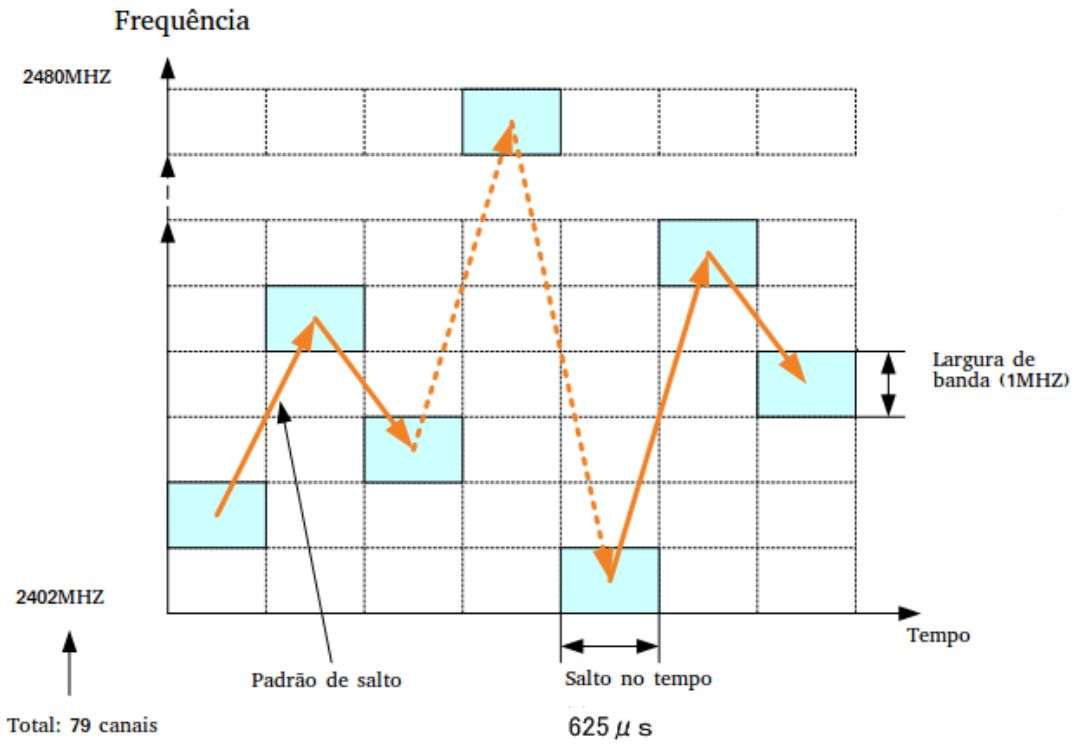



Figura 2.1: *Frequency hopping spread spectrum*[7]

A comunicação entre os dispositivos é baseada no princípio mestre-escravo. Nesta arquitetura tem-se um único dispositivo fazendo o papel de mestre (*master*), e outros dispositivos fazendo o papel de escravo (*slave*). Conexões envolvendo apenas dois dispositivos são chamadas de ponto-a-ponto. Já conexões envolvendo mais de dois dispositivos são chamadas de ponto-a-multiponto. Quando dois ou mais dispositivos existentes estão utilizando o mesmo canal, eles formam uma *piconet*[5]. Cada *piconet* pode ter apenas um dispositivo mestre. Porém, um dispositivo pode ser mestre em uma *piconet* e escravo em

outra. Um conjunto de duas ou mais *piconets* formam uma *scatternet*, conforme pode ser visto na figura 2.2.

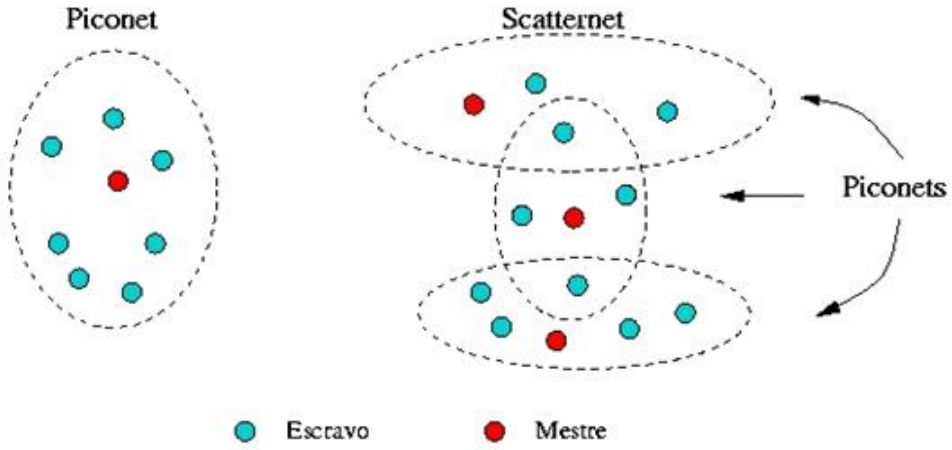



Figura 2.2: *Piconet* e *Scatternet*[8]

O dispositivo mestre tem como papel o controle do acesso dos dispositivos escravos no canal e o início da transmissão de dados. Em uma *piconet*, toda a comunicação ocorre entre o dispositivo mestre com os escravos, ou seja, não há comunicação entre dispositivos escravos[9]. Em uma mesma *piconet* podem estar conectados ao dispositivo mestre até 255 dispositivos escravos, porém, destes, apenas 7 podem estar ativos simultaneamente, o restante deve estar em modo estacionado (*parked*). A figura 2.3 ilustra este tipo de arquitetura.

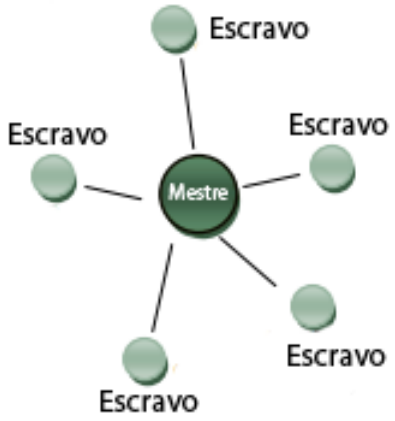



Figura 2.3: Arquitetura mestre-escravo

O *Bluetooth* tem diferentes versões, alcances e velocidades de transmissões. Na tabela 2.1 são apresentadas as potências de transmissão do *Bluetooth* e também seus alcances. Já na tabela 2.2 são apresentadas as versões existentes para o *Bluetooth* e suas velocidades de transmissão.

| Classe | Potência de transmissão do *Bluetooth* | Alcance aproximado |
|---|---|---|
| 1 | 100 mW | Até 100 metros |
| 2 | 2,5 mW | De 10 à 20 metros |
| 3 | 1 mW | Até 1 metro |

Tabela 2.1: Potências de Transmissão e alcances aproximados [10]

| Versão do *Bluetooth* | Velocidade máxima |
|---|---|
| 1.2 | 1 Mbit/s |
| 2.0 + Enhanced Data Rate (EDR) | 3 Mbit/s |
| 3.0 + High Speed | 24 Mbit/s |
| 4.0 | 1 Mbit/s |

Tabela 2.2: Velocidade de transmissão de dados [11]

O protocolo do *Bluetooth* é considerado confiável em sua transmissão, pois garante que pacotes perdidos ou que foram corrompidos sejam retransmitidos. O *Bluetooth* utiliza o método *Cyclic Redundancy Check - CRC* para a identificação de erros nos pacotes[9]. Este método é muito utilizado em redes de computadores.

### 2.1.1 Módulo *BlueSMiRF Silver*

*BlueSMiRF Silver* é um módulo *Bluetooth* capaz de receber e enviar dados para qualquer outro dispositivo *Bluetooth* que suporte SPP (*Serial Port Profile*), permitindo que a comunicação serial sem fio seja de forma transparente. Para isso é necessário a utilização do módulo acoplado a um microcontrolador. No caso do projeto, um Arduíno Uno foi escolhido para ser utilizado. Ele mede apenas 45 x 16,6 x 3,9mm e consegue operar em temperaturas entre -40C até +70C.

O módulo *BlueSMiRF Silver* possui as seguintes características[12]:

- Utiliza a classe 2 do *Bluetooth*;
- Suporta a versão 2.0 + EDR do *Bluetooth*;

- Alta integridade da informação (correção de erros para garantia de pacotes enviados);
- Comunicação serial de 2400bps à 115200bps;
- Utiliza conexão encriptada;
- Opera nas voltagens de 3,3V à 6V;
- Antena interna integrada;
- Baixo consumo de energia (26$\mu$A modo ocioso, 3mA quando conectado, 30mA quando transmitindo);
- Módulo de rádio transmissão com tamanhos reduzidos (0,15x0,6x1,9");

Ele possui um modo de comando onde é possível configurá-lo. Facilmente este modo é acessado pelo microcontrolador quando envia-se para o módulo o conjunto de caracteres *$$$*, através do terminal de comando. Ao entrar no modo de comando, é possível alterar algumas de suas características, tais como o nome do dispositivo, a velocidade de transmissão dos dados, o valor do código *PIN*, o qual é utilizado para realizar o pareamento entre o módulo com outro dispositivo, entre outros itens. A placa inclui dois *leds*. Um vermelho, indicando o estado corrente do módulo, e um outro verde, indicando se algum dispositivo está conectado atualmente ao módulo. Ele permite estar conectado a apenas um dispositivo por vez.

A figura 2.4 representa o módulo *Bluetooth BlueSMiRF Silver* utilizado no projeto.

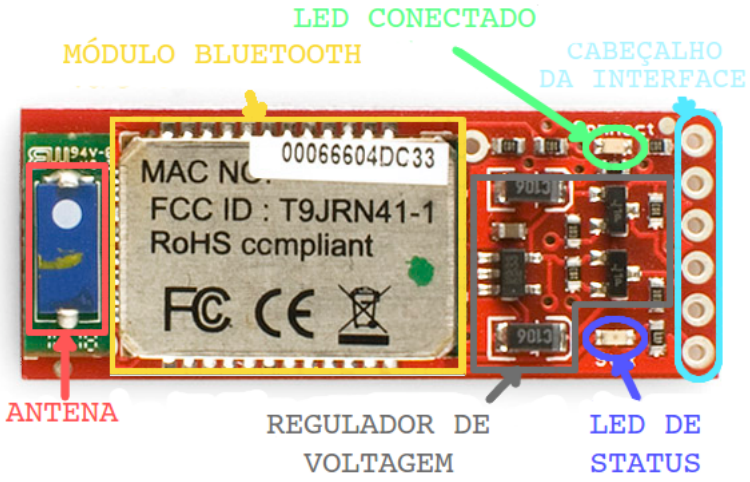



Figura 2.4: Módulo *Bluetooth BlueSmirf Silver*[12]

## 2.2 Sistema Operacional Android

Android é uma plataforma, a qual é composta por um sistema operacional e ferramentas de desenvolvimento para dispositivos móveis. Foi desenvolvido pelo Google em parceria com diversas outras empresas do ramo da tecnologia. Possui seu código fonte aberto, onde está sob a licença *Apache Software License*[2]. Atualmente é a plataforma *mobile* mais popular do mundo, onde encontra-se na versão 7. Em relação ao ano de 2016, a plataforma do Google esteve presente em mais de 80% dos *smartphones* comercializados [14]. A figura 2.5 e a tabela 2.3 mostram a diferença da presença no mercado entre os sistemas operacionais *mobile* atuais.

[2] *Apache Software License* é uma licença onde permite que qualquer desenvolvedor faça alterações no código fonte, não exigindo que estas alterações sejam expostas.[13]

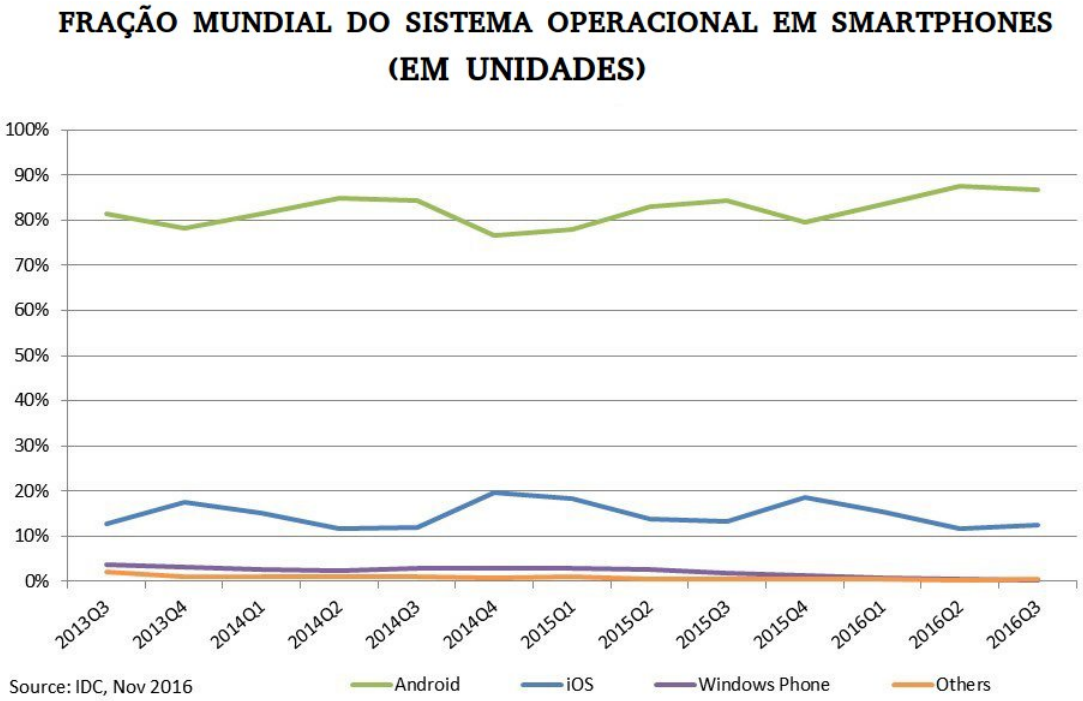



Figura 2.5: Presença no mercado dos sistemas operacionais *mobile* atuais [14].

| Trimestre/Ano | Android | iOS | Windows Phone | Outros |
|---|---|---|---|---|
| 4º/2015 | 79,6% | 18,7% | 1,2% | 0,5% |
| 1º/2016 | 83,5% | 15,4% | 0,8% | 0,4% |
| 2º/2016 | 87,6% | 11,7% | 0,4% | 0,3% |
| 3º/2016 | 86,8% | 12,5% | 0,3% | 0,4% |

Tabela 2.3: Diferenças de presença no mercado dos sistemas operacionais *mobile* [14].

## 2.3 Escolha das tecnologias empregadas

Esta seção aborda a motivação para a escolha destas tecnologias para o projeto.

### 2.3.1 Motivação para escolha do *Bluetooth*

Primeiramente, foi decidido pela utilização de uma tecnologia sem fio, com a finalidade de evitar cabos e facilitar a implantação e o transporte dos dados. Conforme estudos realizados em cima de tecnologias sem fio, chegou-se a conclusão que o *Bluetooth*, por suas características, é o que melhor atendia aos requisitos do projeto. Essa escolha para transmissão dos dados processados, ao invés de outras tecnologias sem fio como *ZigBee*, deve-se principalmente por seu baixo custo, baixo consumo de energia, taxas de transmissão de dados suficientes e fácil integração com a plataforma Arduino, sendo apenas necessário a utilização de um módulo extra com esta tecnologia na plataforma, visto anteriormente.

Além das vantagens citadas, a tecnologia já está inserida na maioria dos *smartphones* atuais [11], não necessitando de placas externas conectadas aos aparelhos para transmitir e receber os dados. Como desvantagens, porém irrelevantes para o projeto, estão o curto alcance e a limitação, por parte do módulo *BlueSmirf Silver*, de dispositivos conectados simultaneamente.

### 2.3.2 Motivação para escolha do Android

Atualmente estão sendo lançados dispositivos móveis com *hardwares* cada vez mais robustos, com maiores capacidades de processamento e com valores mais acessíveis ao público, consequentemente, os dispositivos móveis estão ficando cada dia mais populares. O uso de *smartphones* e *tablets* está em constante crescimento. A Figura 2.6 mostra

evolução dos tipos de dispositivos no mercado[14]. É possível notar que para 2017 espera-se que 87% do mercado serão de *tablets* e *smartphones*.

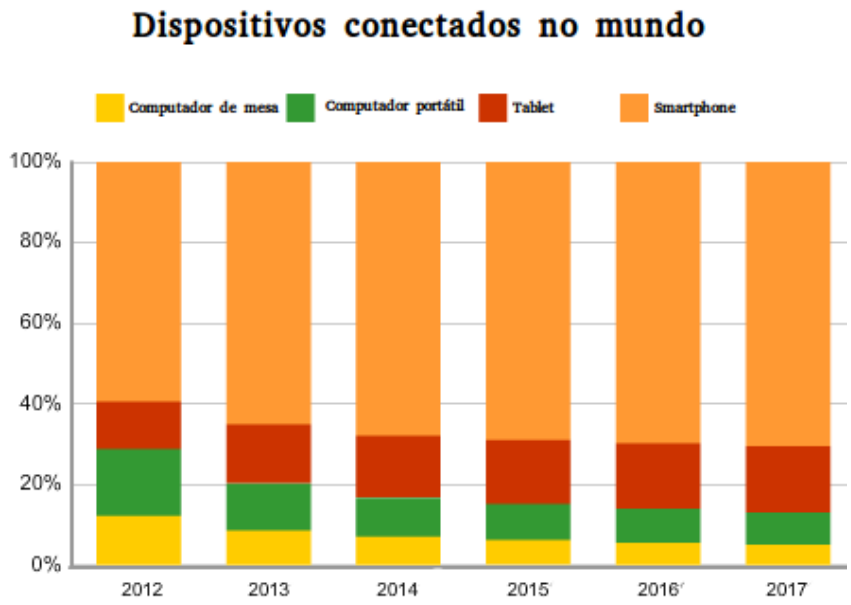



Figura 2.6: Dispositivos conectados no mundo [14]

Têm-se diversas escolhas de plataformas de desenvolvimento para dispositivos móveis que poderiam ser adotadas ao projeto. Contudo o sistema operacional Android foi definido pois se destacou em relação ao demais pelo fato de estar presente na maior parte dos dispositivos móveis existentes, por possuir uma ampla comunidade de desenvolvimento e uma documentação bem definida e de fácil acesso. Além disso, a portabilidade que um *smartphone* provê é um fator extremamente relevante que deve ser levado em conta. Devido à proposta deste projeto, a utilização de estações de trabalho para processamento e visualização das informações no meio do pasto não seria uma escolha interessante.

# Capítulo 3

# Desenvolvimento do aplicativo

Este capítulo apresentará os caminhos seguidos para o desenvolvimento do aplicativo *mobile* em sua versão 1.0, as ferramentas utilizadas e as funcionalidades presentes.

## 3.1 Passos do desenvolvimento

Após as escolhas das tecnologias que seriam utilizadas no decorrer do projeto, iniciou-se a fase de desenvolvimento do aplicativo.

Inicialmente o módulo *BlueSmirf Silver* foi inserido no microcontrolador, fazendo as ligações necessárias para o correto funcionamento. Após isso, foi realizada uma configuração padrão do mesmo através do terminal de comando do microcontrolador, deixando-o preparado para a conexão por *bluetooth*. Em seguida, foram realizados pequenos testes de transmissão entre um *smartphone* e o módulo e vice-versa, através de um aplicativo genérico disponível na internet.

Dado este primeiro passo, o desenvolvimento do aplicativo foi iniciado utilizando a

ferramenta de desenvolvimento integrada *Android Studio IDE* do Google. Esta ferramenta permite que o código-fonte desenvolvido seja implantado diretamente em um *smartphone*, podendo o aplicativo ser testado em um ambiente real gerando *logs* para depuração. No caso deste projeto foi utilizado para o desenvolvimento e testes o *smartphone LG Google Nexus 4*, cuja sua configuração é: CPU *quad-core Qualcomm APQ8064 Snapdragon S4 Pro*, com *clock* de 1,5Ghz, 2GB de memória RAM, 16GB de armazenamento interno para dados, *bluetooth* na versão 4.0 e sistema operacional Android na versão 5.0.2. A primeira tarefa desenvolvida foi realizar a comunicação entre o *smartphone* e o módulo *bluetooth*, utilizando a taxa de transmissão padrão de 9600bps. Foram enviados dados reais de *samples* fotoplestimográficos (PPG) capturados anteriormente através dos sensores presentes nos bovinos e armazenados em um cartão de memória *SD card* presente na plataforma. Para armazenar esses dados no *smartphone*, foi utilizado o banco de dados SQLite, presente nativamente no Android. No primeiro momento não houve preocupação com normalização ou algumas outras regras de consistência de dados para a modelagem do banco de dados, pois futuramente novas informações apareceriam e o banco deveria ser remodelado. Portanto, a modelagem foi realizada utilizando-se uma única tabela chamada AMOSTRA para gerenciar todos os dados. Esta tabela continha apenas três campos, conforme pode ser visto na Figura 3.1.

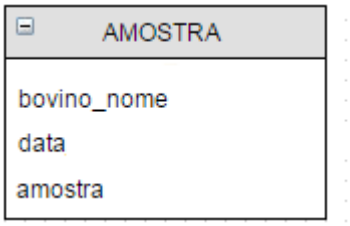



Figura 3.1: Tabela única na versão 1.0

Cada *sample* recebido é representado por um número inteiro positivo. Os dados lidos pelo microcontrolador no cartão de memória SD são transmitidos *byte* a *byte* para o *smartphone* por *bluetooth*. Após o recebimento dos dados, estes são armazenados no banco de dados.

O sensor oxímetro está configurado para ler *samples* na frequência de 250Hz, ou seja, para cada um segundo 250 *samples* PPG (*Photopletismogram*) são capturados. Em apenas um minuto 15000 *samples* PPG são gravados no cartão de memória. Devido à esse volume de dados, a taxa de transmissão padrão de 9600bps tornava o processo de transferência dos dados, entre a placa e o aplicativo, muito demorado. Conforme informado no Capítulo 2, o módulo utilizado para transferência por *bluetooth* aceitava configurações de taxas de transmissão de dados de 2400bps até 115200bps. Foram realizados testes de transmissão na taxa máxima de 115200bps, porém sem sucesso, pois os dados chegavam no destino corrompidos. A maior taxa de transmissão onde os dados chegavam íntegros no destino foi 57600bps. Tendo uma melhora em 6x menos tempo de transmissão. Após essa etapa, iniciou-se o desenvolvimento da parte de visualização das informações.

### 3.1.1 Visualização das Informações armazenadas

Para essa etapa realizou-se buscas de bibliotecas que pudessem gerar gráficos de linhas baseado nos *samples* PPG recebidos pelo *smartphone*. Algumas foram testadas e a biblioteca *open-source* AChartEngine[15] foi escolhida, a qual atendeu bem os requisitos. Os dados dos *samples* eram lidos do banco de dados e enviados para a biblioteca, onde ela "plotava"cada amostra gerando o gráfico. É possível ver na Figura 3.2 um exemplo do gráfico gerado.

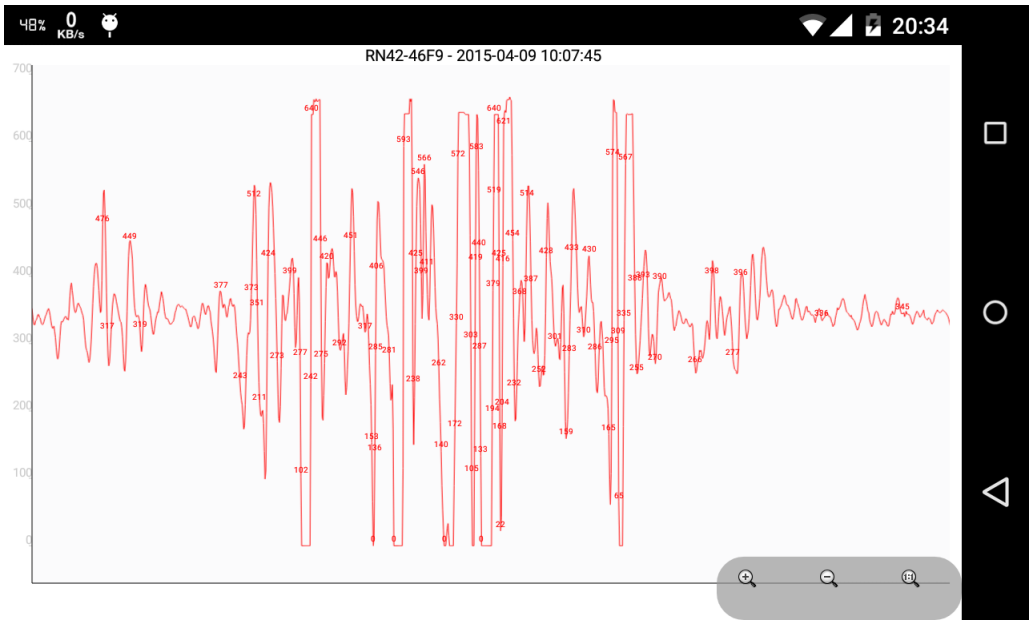



Figura 3.2: Gráfico com *samples* PPG gerado pela biblioteca AChartEngine

Através do gráfico gerado é possível ver a forma de onda PPG gerada pelos *samples* capturados.

Além da visualização da informação pelo *smartphone*, uma outra funcionalidade in-

teressante foi implementada, possibilitando a transmissão dessas informações capturadas dos sensores, para uma possível visualização em um computador ou apenas para armazenamento. Essa funcionalidade possibilita a exportação de um arquivo com formato csv (*Comma Separated Value*), contento os dados dos *samples* PPG, e o envio deste arquivo por e-mail.

Após isso, o banco de dados foi remodelado adotando os padrões de normalização[1], para melhorar sua estrutura e também para ganhar desempenho, pois o mesmo já apresentava sinais de lentidão conforme os *samples* PPG armazenados nos testes em laboratório cresciam. Além disso, essa alteração possibilitou ser possível o cadastro de bovinos e o armazenamento de *samples* PPG e de temperatura para cada um independentemente. Para isso ao invés de uma tabela, o modelo foi alterado para 5 tabelas, seguem elas:

- BOVINO
- DATA_PPG_BOVINO
- AMOSTRA_PPG
- DATA_TEMP_BOVINO
- AMOSTRA_TEMP

Na tabela BOVINO, são armazenados os dados do bovino. Os campos presentes

[1] A normalização do banco de dados é o processo de organizar as colunas (atributos) e tabelas (relações) de um banco de dados relacional, de modo a reduzir a redundância e melhorar a integridade dos dados. [16]

na tabela são: bovino_id (identificador único do bovino e chave primária da tabela) e bovino_nome (nome para identificação de cada bovino).

Na tabela DATA_PPG_BOVINO são armazenadas as datas das amostras, por bovino. Os campos presentes na tabela são: data_id (identificador único e chave primária da tabela), bovino_id (chave estrangeira representando o animal na tabela BOVINO) e data representando a data da amostra. Na tabela AMOSTRA_PPG são armazenadas as amostras com seus valores. Os campos presentes na tabela são: amostra_id (identificador único da amostra e chave primária da tabela), data_id (chave estrangeira representando a data na tabela DATA_PPG_BOVINO) e amostra (representando o valor de um *sample* PPG).

As tabelas DATA_TEMP_BOVINO e AMOSTRA_TEMP apresentam as mesmas estruturas das tabelas DATA_PPG_BOVINO e AMOSTRA_PPG respectivamente. A única diferença é que na tabela AMOSTRA_TEMP são armazenadas as amostras de temperatura ao invés dos *samples* PPG.

Através da Figura 3.3 é possível ver o diagrama entidade relacionamento do banco de dados.

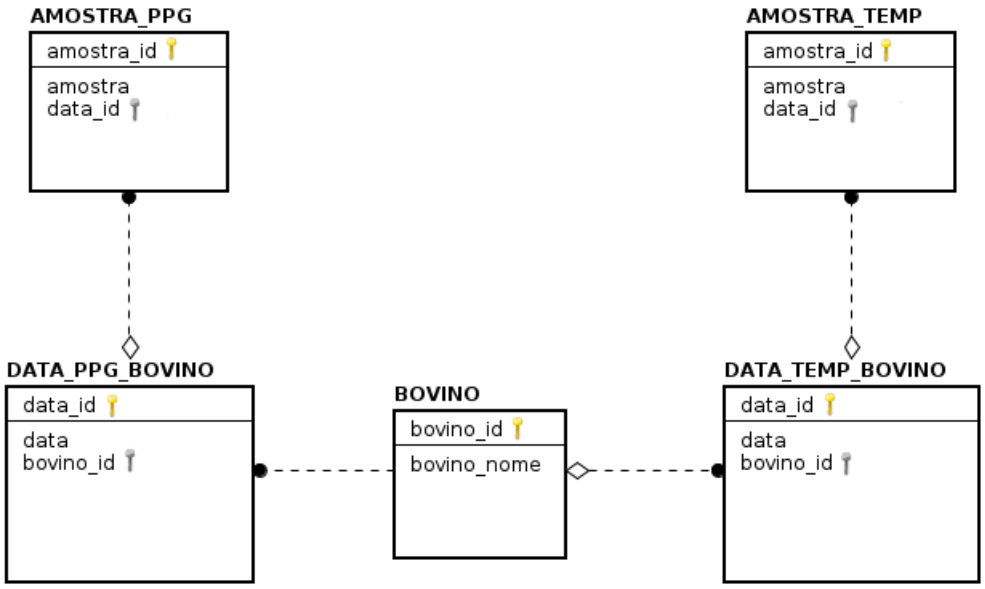



Figura 3.3: Diagrama entidade relacionamento

### 3.1.2 Configurações dos sensores

Uma funcionalidade desenvolvida para o aplicativo foi a possibilidade de configuração do início e fim de captura dos dados dos sensores. Para isso, foi implementado uma central de solicitações. Através de comandos enviados do aplicativo para o microcontrolador por *bluetooth* é possível requisitar que seja iniciado a coleta de dados dos sensores oxímetro (PPG) e IButton (Temperatura), além de solicitar a transmissão desses dados para o aplicativo. Através da Figura 3.4 é possível ver a tela desta funcionalidade.

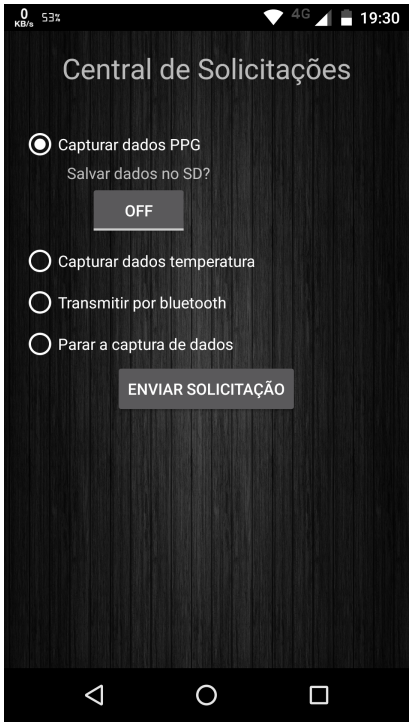



Figura 3.4: Central de solicitações

Ao clicar no botão "ENVIAR SOLICITAÇÃO"quando selecionado a opção "Capturar dados PPG", é enviado para o microcontrolador o caractere "O", indicando que o microcontrolador deve começar a captura dos dados PPG através do sensor oxímetro. A opção "Salvar dados no SD?"é utilizada para indicar que os dados capturados através do microcontrolador devem ser salvos no cartão de memória.

Ao clicar no botão "ENVIAR SOLICITAÇÃO"quando selecionado a opção "Capturar dados temperatura", é enviado para o microcontrolador o caractere "I", indicando que o microcontrolador deve começar a captura dos dados de temperatura através do sensor IButton.

Ao clicar no botão "ENVIAR SOLICITAÇÃO"quando selecionado a opção "Transmitir por bluetooth", é enviado para o microcontrolador o caractere "B", indicando que o microcontrolador deve começar a transmissão dos dados já armazenados para o *smartphone*.

Ao clicar no botão "ENVIAR SOLICITAÇÃO"quando selecionado a opção "Parar a captura de dados", é enviado para o microcontrolador o caractere "C", indicando que o microcontrolador deve parar a captura os dados.

Para a utilização dessas solicitações o aplicativo deve estar conectado ao microcontrolador por *bluetooth*. A Figura 3.5 ilustra um diagrama de sequência indicando as possíveis ações.

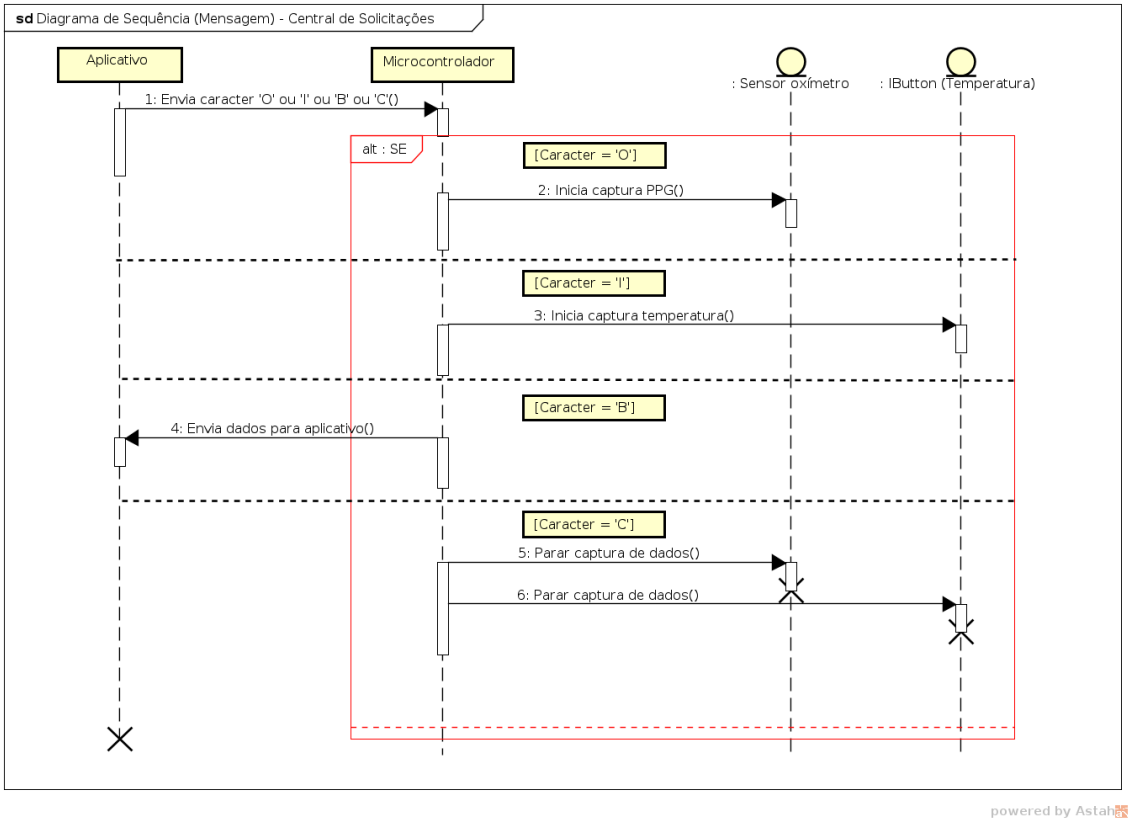



Figura 3.5: Diagrama de sequência da central de solicitações

Ao solicitar a transmissão dos dados, o aplicativo faz um acesso ao seu banco de dados buscando a última data de amostras existentes para o bovino solicitado. Essa data é enviada para o microcontrolador onde ele realiza uma busca dentro do seu cartão de memória, trazendo todos os arquivos contendo datas posteriores a data recebida, para este bovino. Caso encontre arquivos que devam ser enviados, ele os abre e transmite os *samples* para o aplicativo. Este, por sua vez, recebe os *samples* e os grava no banco de dados. O diagrama de sequência na Figura 3.6 descreve os passos desta solicitação.

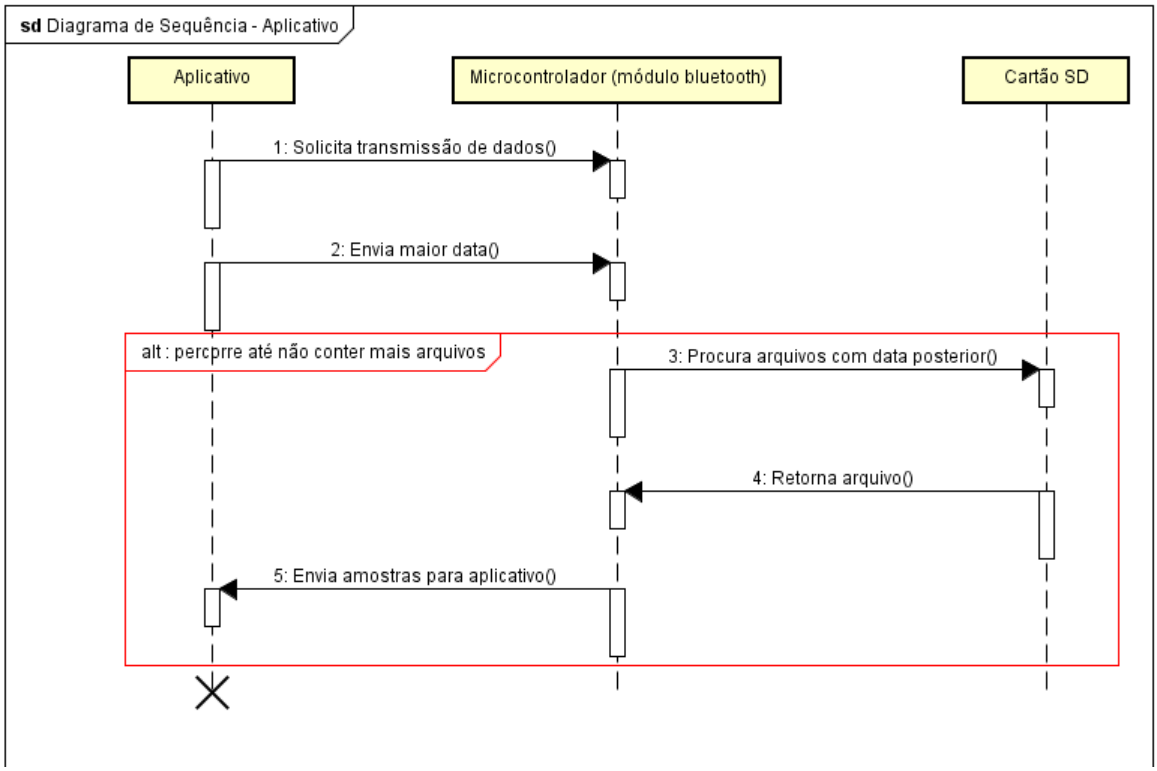



Figura 3.6: Diagrama de sequência da solicitação de transmissão de dados

### 3.1.3 Recebimento das informações

Outra funcionalidade desenvolvida para o aplicativo foi a do recebimento de informações de maneira *online*. Isso significa que conforme os dados estejam sendo capturados pelos sensores presentes no bovino, ao mesmo tempo eles já estão sendo enviados pelo microcontrolador, através do módulo *bluetooth*, para o aplicativo. No aplicativo os dados são recebidos e visualizados. Através da Figura 3.7 é possível ver a tela de acesso à essa funcionalidade. Já nas Figuras 3.8 e 3.9 podem ser vistas a tela de visualização de temperatura e *samples* PPG de maneira *online*.


Figura 3.7: Tela de acesso à funcionalidade de recebimento *online*

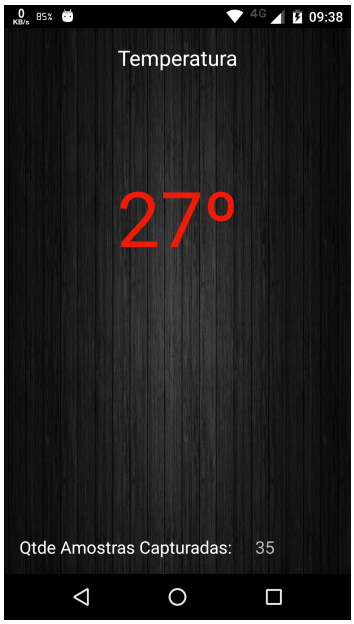



Figura 3.8: Temperatura *online*

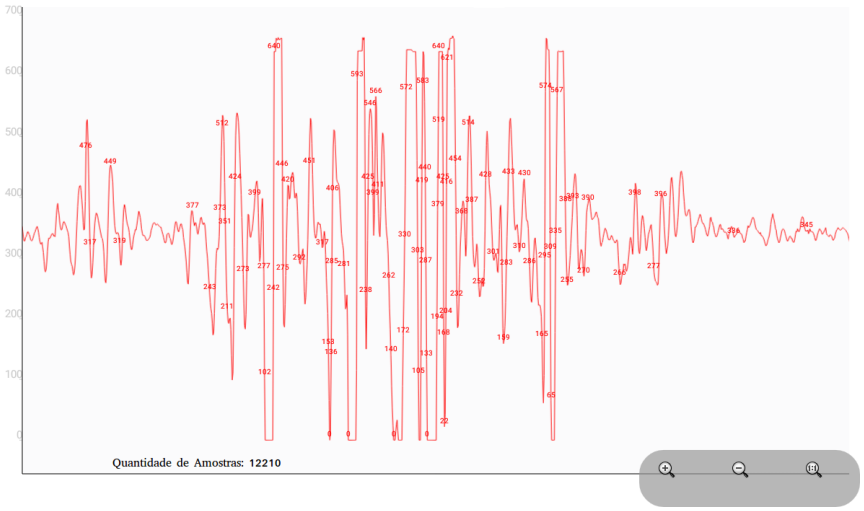



Figura 3.9: *Samples* PPG *online*

Esta funcionalidade é muito importante para que uma análise *online* seja realizada. Assim, é possível verificar, além dos dados PPG e de temperatura dos animais, se algum sensor está posicionado de forma incorreta, não capturando os dados adequadamente.

### 3.1.4 Algoritmos para obtenção das frequências cardíaca e respiratória

Os valores dos *samples* PPG capturados através dos sensores, de maneira isolada, não dizem informações interessantes a respeito do animal. Entretanto, através deles é possível que sejam extraídas informações muito importantes como as frequências cardíaca (batimentos cardíacos por minuto - BPM) e respiratória (respirações por minuto - RPM). Mas para isso é necessário que os *samples* PPG armazenados sejam a entrada de algoritmos que estimem esses dados. No trabalho [1], o autor desenvolveu os algoritmos para estimativa das frequências cardíaca e respiratória. Entretanto, os algoritmos foram de-

senvolvidos em linguagens de programação diferentes da utilizada no desenvolvimento do aplicativo. Dessa forma, neste trabalho os algoritmos foram reescritos para a linguagem de programação Java, a qual é adota pelo sistema operacional *Android*. Além de terem sido testados e validados.

**Algoritmo de frequência cardíaca**

O algoritmo de frequência cardíaca originalmente foi desenvolvido na linguagem de programação C. A entrada do algoritmo são os *samples* PPG armazenados ou recebidos *online*. Estes são passados individualmente, um a um, para o algoritmo, que realiza o cálculo para estimar a frequência cardíaca por minuto.

O Algoritmo 1 descreve os passos para obtenção da quantidade de BPM a partir dos *samples* PPG.

**Algoritmo 1:** Algoritmo para cálculo dos batimentos por minuto

```
   Entrada: sinal PPG: sinal
   Saída: batimentos por minuto: BPM
 1 se sinal < limiar E N > (IBI/5) * 3 então
 2 |   se sinal < T então
 3 |   |   T ← sinal
 4 |   fim
 5 fim
 6 se sinal < limiar E sinal > P então
 7 |   P ← sinal
 8 fim
 9 Função Verificar_Pulso(N)
10 Função Inicializa_IBI(taxa[])
11 contagem_total ← 0
12 para i ← 0; i <= 8; i++ faça
13 |   taxa[i] ← taxa[i+1]
14 |   contagem_total ← contagem_total + taxa[i]
15 fim
16 taxa[9] ← IBI
17 contagem_total ← contagem_total / 10
18 BPM ← 60000 / contagem_total
19 QS ← VERDADEIRO
20 Função Reset(sinal, limiar, P, T)
```

- A variável *limiar* é inicializada com 512, pois refere-se à 50% da amplitude da onda (1024).

- A variável $N$ recebe um contador que mantém o tempo decorrido (em milissegundos), subtraindo o tempo da última batida cardíaca pela penúltima.

- As variáveis P e T representam a retenção do pico e o valor mínimo, respectivamente.

- Entrada: Sinal individual PPG

- Saida: Quantidade de batimentos cardíaco por minuto
- **Linha 1 a 8:** atualiza as variáveis $T$ e $P$ caso as condições sejam verdadeiras.
- **Linha 9:** é chamada a função *Verificar_Pulso*, passando a variável N para verificar se já existe um pulso.
- **Linha 10:** é chamada a função *Inicializa_IBI*. Esta função inicializa o vetor *taxa[]* com os 10 últimos IBIs (*Inter beat interval*).
- **Linha 11:** a variável *contagem_total* recebe o valor 0.
- **Linha 12 até 15:** soma-se todos os IBIs na variável *contagem_total* e desloca-se todos os valores do vetor *taxa[]* uma posição para a esquerda.
- **Linha 16:** o vetor *taxa[]* na última posição recebe o valor do IBI atual.
- **Linha 17:** faz-se a média dos últimos 10 IBIs.
- **Linha 18:** os batimentos cardíacos são obtidos nessa linha pela divisão de 60000 (1 minuto = 60000 milissegundos) pela média dos 10 IBIs.
- **Linha 19:** seta a variável *QS* como VERDADEIRO indicando que o batimento cardíaco foi encontrado.
- **Linha 20:** chama a função *Reset* para limpar as variáveis para não ocorrer o problema de determinar um batimento, pois quando o sinal cruza o limiar descendo, é necessário limpar as variáveis e atualizar a variável limiar com a nova marca de 50%

que será encontrada novamente durante a subida do sinal do sensor. As variáveis $P$ e $T$ são reinicializadas para o novo limiar.

Após a implementação do algoritmo dentro do aplicativo, diversos testes foram realizados para garantir que a portabilidade do algoritmo estava correta. Para isso utilizou-se o algoritmo real (já validado pelo autor do trabalho em [1]) e o algoritmo reescrito. Esses testes tiveram a intenção de verificar se os dados obtidos dos batimentos cardíacos, utilizando o mesmo conjunto de entrada em ambos os algoritmos, eram iguais. Após a comparação dos resultados, concluiu-se que o algoritmo estava de acordo com o original.

**Algoritmo de frequência respiratória**

O algoritmo de frequência respiratória originalmente foi desenvolvido utilizando a ferramenta matemática para engenharia, denominada MATLAB. Essa ferramenta tem uma linguagem de programação própria bem diferente das mais conhecidas, com muitas funções nativas para cálculos que tiveram que ser implementadas no aplicativo. Dessa forma, a portabilidade do código para a linguagem de programação Java foi complexa e trabalhosa.

Foi definido por [1] o uso do algoritmo de análise da largura de pulso (PWV - *Pulse Width Variability*) como mecanismo para identificar os momentos de início e fim dos ciclos respiratórios em bovinos, ao invés de outros. Para a obtenção da frequência respiratória através do algoritmo PWV, é necessário a identificação de alguns pontos de interesse dentro do sinal PPG. Entretanto, o sinal PPG pode conter ruídos que dificultam a identi-

ficação desses pontos. Assim sendo, houve a necessidade de implementar estratégias para filtragem dos ruídos indesejáveis no sinal, o que não foi necessário para o algoritmo de frequência cardíaca.

O algoritmo para extração da frequência respiratória está dividido em 4 etapas:

- Filtro do sinal PPG;
- Obtenção dos pontos de interesse;
- Algoritmo PWV para obtenção da nova forma de onda correspondente aos ciclos respiratórios;
- Algoritmo para calcular a quantidade de respirações por minuto baseado no retorno dos dados de retorno do PWV;

**Filtragem dos ruídos do sinal PPG**

A partir de análises sobre amostras do sinal PPG, percebeu-se que haveria a necessidade de implementar uma filtragem desse sinal, a fim de adequá-lo às faixas de valores, onde a forma de onda encontra-se livre de ruídos. A Figura 3.10 ilustra um sinal com ruídos contendo 5140 *samples* coletados em 102,8 segundos.

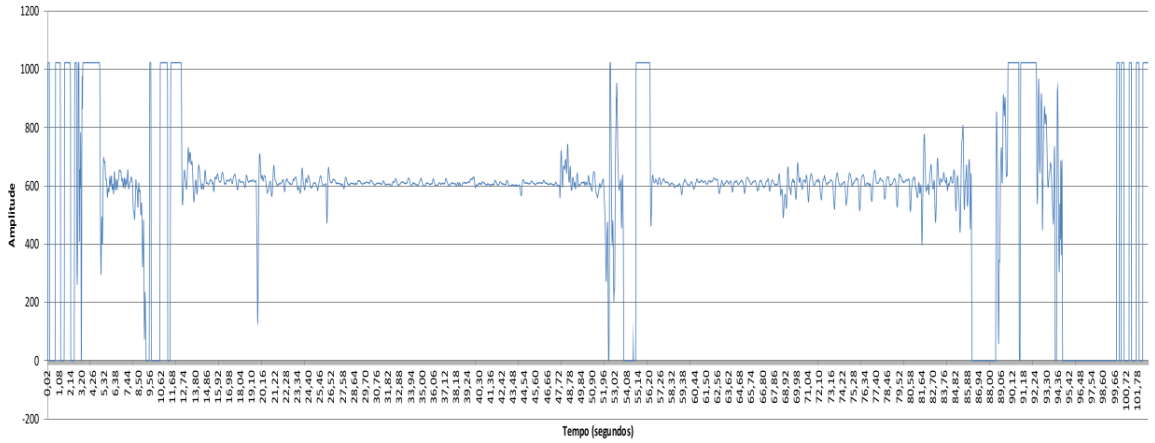



Figura 3.10: Sinal PPG com ruídos [1]

Essa oscilação da amplitude da forma de onda, e também os ruídos, devem-se a alguns fatores como a movimentação do animal, o posicionamento do sensor, a intensidade com a qual o sensor é pressionado na pele do animal, entre outros. Por exemplo, vários *samples* PPG com valores zeros alocados de forma consecutiva são considerados ruídos. Outro exemplo de ruído são valores dentro dos *samples* PPG que estão fora da faixa de frequência estabelecida. Devido à isso, esses dados devem ser descartados. Para esse procedimento, quatro passos são realizados em ordem:

- Identificação dos pontos de interesse no sinal PPG com ruídos;
- Marcação das posições onde contenham valores zeros seguidos no sinal PPG;
- Marcação das posições que estão com valores fora da faixa de frequência estabelecida;
- Geração de um novo vetor para o sinal PPG filtrado;

A seguir é apresentado o Algoritmo 2, desenvolvido por [1] para a detecção dos pontos de interesse no sinal PPG. Esses pontos de interesse serão utilizados para o cálculo da frequência respiratória através do algoritmo PWV.

**Algoritmo 2:** Algoritmo de filtragem de sinal e detecção dos pontos de interesse

**Entrada:** sinal PPG: PPG[]
**Saída:** Pontos de interesse: vetor_indicesNa, vetor_indicesNb, vetor_indicesNo, vetor_indicesNe

```
1  [indice_Na, valor_na] ← Função peakfinder(PPG, parâmetros_ajustaveis)
2  contador ← 1
3  k ← indice_Na(contador)
4  enquanto indice_Na(contador) = VERDADEIRO faça
5      enquanto PPG(k-1) < PPG(k) E range > range_No faça
6          valor_atual ← PPG(k-1)
7          flag← 1
8      fim
9      enquanto PPG(janela) < PPG(k) E range > range_No faça
10         valor_atual ← PPG(janela)
11         flag← 1
12     fim
13     se flag = 1 então
14         Função Valida_No
15         se PPG(k-1) <= PPG(k) então
16             enquanto PPG(k-1) <= PPG(k) faça
17                 valor_atual ← PPG(k-1)
18             fim
19         fim
20         Função Valida_Nb
21         valor_atual← valor_Na(contador)
22         flag← 0
23         enquanto PPG(k+1) <= PPG(k) E range > range_Ne faça
24             valor_atual ← PPG(k+1)
25             flag← 1
26         fim
27         enquanto PPG(janela) <= PPG(k) E range > range_Ne faça
28             valor_atual ← PPG(janela)
29             flag← 1
30         fim
31         se flag = 1 então
32             Função Valida_Ne
33         fim
34     fim
35     contador ← contador + 1
36     k← indice_Na(contador)
37 fim
```

- Entrada: Sinal PPG completo;
- Saída: Os pontos de interesse: vetores de índices Na, Nb, No e Ne;
- **Linha 1:** invoca-se a função *peakfinder*, que recebe o sinal PPG e alguns parâmetros ajustáveis, como amplitude base para definição dos picos. Ela retornará os índices e valores dos picos definidos do sinal PPG;
- **Linha 2:** inicializa-se a variável *contador*;
- **Linha 3:** variável $k$ recebe o valor do índice do primeiro pico;
- **Linha 4:** inicia-se o laço de repetição, onde houver um pico válido, haverá pontos de interesse;
- **Linha 5:** identifica-se os pontos *No* comparando os *samples* à esquerda do pico de cada onda;
- **Linhas 6 à 7:** a variável *valor_atual* recebe o valor anterior e a variável *flag* recebe o valor 1, enquanto a condição da linha 5 for atendida;
- **Linhas 9 à 12:** após encontrar um possível ponto *No*, é necessário certificar que esse ponto encontrado não é um ruído. Dessa forma a variável *janela* serve como índice para o vetor de *samples* PPG. Ela pode ser atribuída com os valores 2 ou 3, por exemplo, significando que ao invés de analisar o ponto subsequente, dois ou três pontos a partir do ponto atual são analisados.

- **Linha 13:** Verifica se algum ponto *No* foi encontrado.
- **Linha 14:** Caso tenha sido encontrado algum ponto *No*, este deve ser validado.
- **Linhas 15 à 19:** Busca-se o ponto de interesse *Nb*, verificando se o ponto à esquerda é menor ou igual ao ponto da direita, pois dessa forma, o declínio da onda já se estabilizou.
- **Linha 20:** Caso tenha sido encontrado algum ponto *Nb*, este deve ser validado.
- **Linhas 21 à 22:** Nessa linha, os pontos à esquerda do pico *Na* já foram encontrados. Dessa forma, é necessário encontrar o único ponto a direita referente à esse pico, o ponto *Ne*. Assim, a variável *valor_atual* recebe o pico *Na* para começar a descida à direita. A variável *flag* recebe o valor 0.
- **Linhas 23 à 34:** É realizado os mesmos passos das Linhas 5 à 12, só que agora para o lado direito. Na Linha 26 é realizado a validação do ponto *Ne*.
- **Linhas 35 à 36:** Por fim, a variável *contador* é incrementada, e a variável $k$ recebe a posição do próximo pico *Na*.

O sinal PPG filtrado, resultante do Algoritmo 2, pode ser visto através da Figura 3.11.

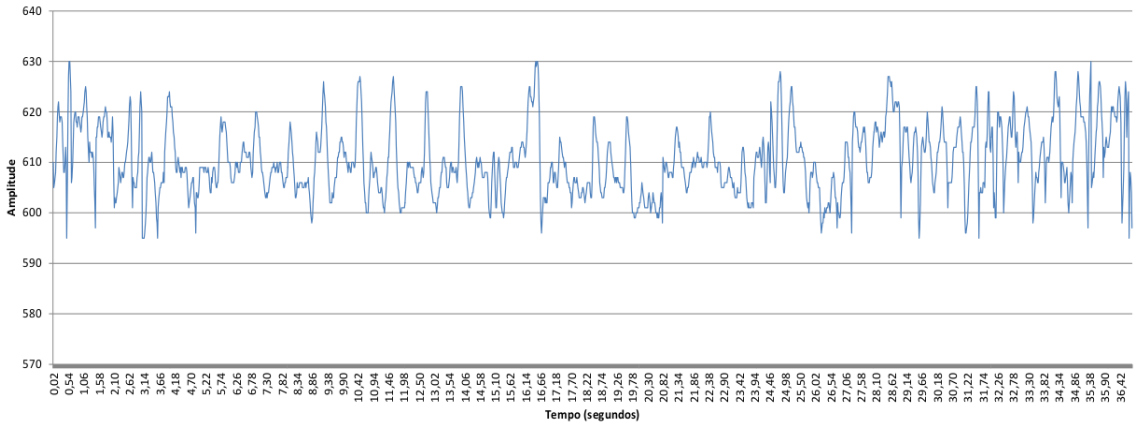



Figura 3.11: Sinal PPG filtrado [1]

### Algoritmo PWV para obtenção dos dados de frequência respiratória

Com os pontos de interesse definidos pelo Algoritmo 2, conforme a Figura 3.12 [1] ilustra, uma nova forma de onda pode ser extraída através do algoritmo PWV, o qual é apresentado a seguir, no Algoritmo 3 [1]. Através dessa nova forma de onda PWV gerada, a qual corresponde aos ciclos respiratórios, é possível calcular sua frequência respiratória.

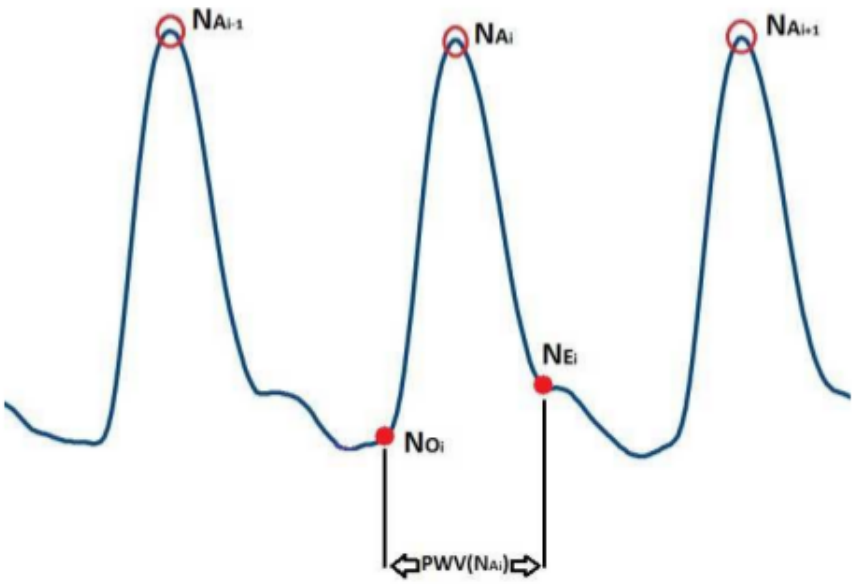



Figura 3.12: Pontos de interesse para cálculo da frequência respiratória no algoritmo PWV [1]

**Algoritmo 3:** ALGORITMO PWV

**Entrada:** Pontos de Interesse: indice_Na, indice_No, indice_Ne, tamanho_vetor_PPG
**Saída:** FR_Derivada

```
1 tamanho_vetor_Na ← TAMANHO(indice_Na)
2 para k ← 1; k <= tamanho_vetor_Na; k++ faça
3 |   para j ← indice_Ne + 1; j <= tamanho_vetor_PPG; j++ faça
4 |   |   se j <= indice_Ne[k+1] então
5 |   |   |   PWV[j] ← (1/50) * (indice_Ne[k] - indice_No[k]) * (j - indice_Na[k])
6 |   |   fim
7 |   fim
8 fim
```

- Entrada: Pontos de interesse: indice_Na, indice_No, indice_Ne, tamanho_vetor_PPG
- Saída: PWV[] (Nova forma de onda PWV)
- **Linha 1:** A variável *tamanho_vetor_Na* recebe o tamanho do vetor de índices Na.

- **Linha 2:** Laço de repetição que percorrerá todas as posições dos vetores Na, Ne, No.

- **Linha 3:** Neste laço de repetição, a variável $j$ recebe a posição subsequente, onde encontra-se o ponto de interesse *N[k]*.

- **Linha 4:** A condição contida nessa linha garante que o cálculo realizado na linha 35 seja feito dentro do intervalo da forma de onda posterior.

- **Linha 5:** Realiza-se o cálculo da nova forma de onda PWV.

A Figura 3.13 ilustra a forma de onda PWV resultante de dados reais, *samples* PPG capturados em testes de campo, de um bovino. Esta forma foi obtida após a execução dos Algoritmos 2 e 3.

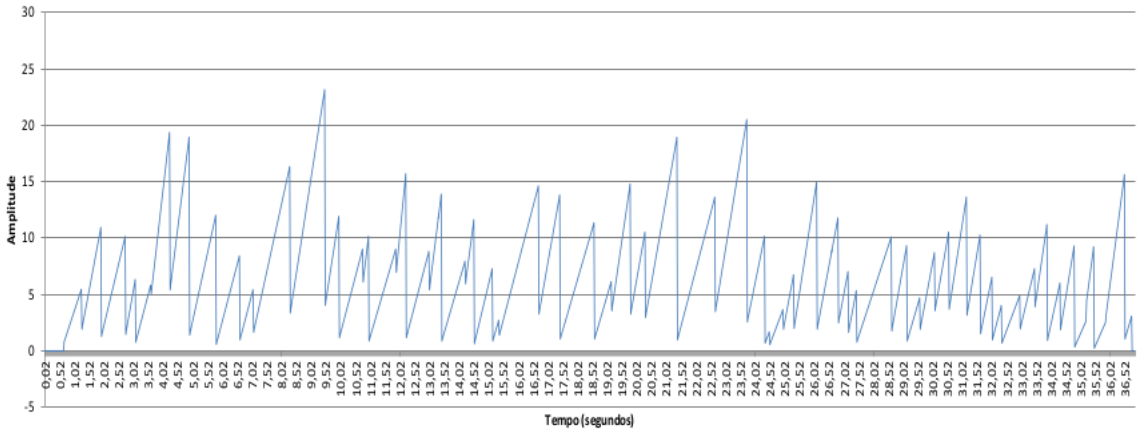



Figura 3.13: Forma de onda PWV [1]

Para conseguir calcular a frequência respiratória é necessário ainda que uma nova forma de onda seja criada baseado na derivação dos picos da onda PWV. Após isso uma

nova onda com pontos suavizados é gerada, conforme pode ser visto na Figura 3.14.

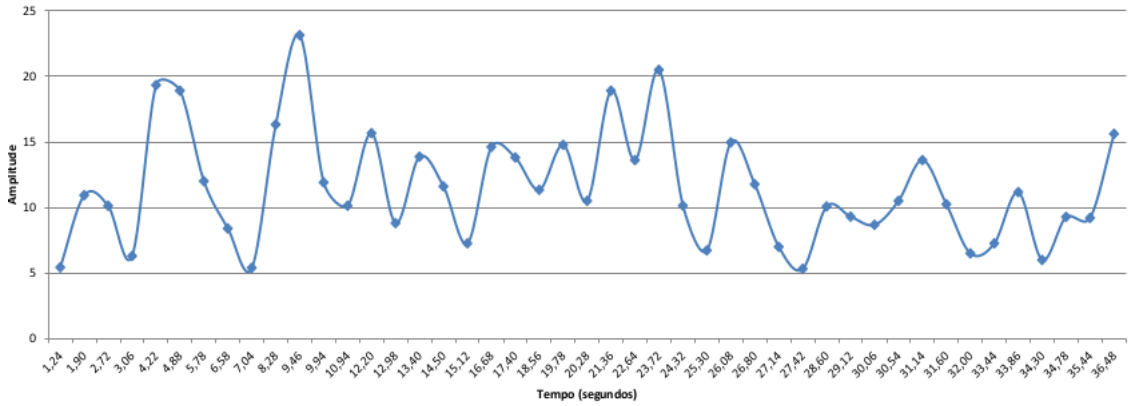



Figura 3.14: Forma de onda derivada a partir dos picos da onda PWV [1]

Baseado nesta nova forma de onda gerada, é possível identificar cada ciclo respiratório, ou seja, cada período desta onda representa uma respiração bovina. Analisando os ciclos respiratórios da Figura 3.14, este animal realizou 14 respirações em aproximadamente 36,82 segundos, resultando em uma frequência respiratória de 22,8 respirações por minuto. O Algoritmo 4, desenvolvido neste trabalho, calcula esses ciclos respiratórios, determinando assim a quantidade de respirações que o animal deu neste período.

**Algoritmo 4:** Algoritmo para obtenção da quantidade de respirações baseado na forma de onda dos picos da onda PWV

**Entrada:** Forma de onda derivada da onda PWV: PWV_Picos[]
**Saída:** qtdeRespiracoes

```
1  qtdeRespiracoes ← 0
2  tamanho_vetor ← TAMANHO(PWV_Picos)
3  se PWV_Picos[1] < PWV_Picos[2] então
4  |  subindo ← VERDADEIRO
5  fim
6  senão
7  |  subindo ← FALSO
8  fim
9  para k ← 1; k < tamanho_vetor; k++ faça
10 |  se subindo = VERDADEIRO então
11 |  |  se PWV_Picos[k] > PWV_Picos[k+1] então
12 |  |  |  subindo ← FALSO
13 |  |  |  qtdeRespiracoes ← qtdeRespiracoes + 1
14 |  |  fim
15 |  fim
16 |  senão
17 |  |  se PWV_Picos[k] < PWV_Picos[k+1] então
18 |  |  |  subindo ← VERDADEIRO
19 |  |  |  qtdeRespiracoes ← qtdeRespiracoes + 1
20 |  |  fim
21 |  fim
22 fim
```

- Entrada: Forma de onda derivada da onda PWV: PWV_Picos[]
- Saída: A quantidade de respirações dada pelo animal: qtdeRespiracoes
- **Linha 1:** A variável *qtdeRespiracoes* é inicializada com o valor 0.
- **Linha 2:** A variável *tamanho_vetor* recebe o tamanho do vetor *PWV_Picos*.
- **Linha 3:** Verifica se o valor da primeira posição do vetor *PWV_Picos* é menor que

o valor da segunda posição do vetor *PWV_Picos*.

- **Linha 4:** Caso seja, seta a variável *subindo* como verdadeiro.
- **Linha 6 à 8:** Caso não seja, seta a variável *subindo* como falso.
- **Linha 9:** Laço de repetição que percorrerá todas as posições no vetor *PWV_Picos*.
- **Linha 10 até 22:** Verifica os momentos onde a onda altera sua direção, contando assim uma respiração.

**Considerações finais**

Todos os algoritmos ilustrados neste capítulo, após terem sido reescritos para se adaptarem ao aplicativo *mobile*, foram testados com amostras reais e comparados seus resultados com as saídas obtidas através dos algoritmos originais, desenvolvidos por [1]. Dessa forma os algoritmos foram validados. Entretanto, conforme descrito pelo autor, verificou-se que os dados obtidos através das saídas dos algoritmos não foram tão precisos, comparados com ferramentas utilizadas para estes fins, conforme [1]. Isso ocorre principalmente devido aos ruídos do sinal capturado pelo *hardware* utilizado atualmente no projeto, pois são equipamentos para serem utilizados em seres humanos. Além disso, a forma como o animal reage à colocação do sensor oxímetro em sua pele, fazendo movimentos bruscos e se mexendo também impacta na qualidade do sinal.

### 3.1.5 Testes em campo

No decorrer deste projeto, diversas visitas foram realizadas à Embrapa Gado de Corte, situada no município de Campo Grande/MS. Através delas foi possível validar o funcionamento dos sensores oxímetro e *IButton*, coletar amostras PPG e de temperatura, além da transmissão dos dados para o aplicativo *mobile* desenvolvido.

As Figuras 3.15 (a e b) e 3.16 (a e b) ilustram a coleta do sinal PPG nos animais.

(a) (b)

Figura 3.15: Coleta dos sinais PPG no animal

(a)

(b)

Figura 3.16: Coleta dos sinais PPG no animal

Nos primeiros momentos os animais se mantiveram calmos. Entretanto, no decorrer do tempo, estes passaram a ficar inquietos e impacientes com a colocação do sensor. Um dos problemas apresentados foi que a forma de onda obtida através do sensor PPG continha muitos ruídos, dificultando a utilização dos algoritmos para extração da frequência cardíaca e respiratória.

A Figura 3.17 ilustra o teste de transmissão dos dados para o aplicativo *mobile*.

Figura 3.17: Teste de transmissão de dados pelo aplicativo *mobile*

Devido ao campo aberto, a transmissão dos dados da plataforma para o aplicativo ocorreu conforme esperado, sem perdas de dados e dificuldades. Os dados foram transmitidos, e puderam ser visualizados na tela do *smartphone*. Dessa forma, a transmissão dos dados foi bem sucedida.

# Capítulo 4

# Considerações Finais

Este projeto objetivou o desenvolvimento de um aplicativo *mobile* para o sistema operacional Android, onde através dele e de um sistema integrado de *hardware* e *software* fosse possível obter dados de variáveis fisiológicas de bovinos, como frequência cardíaca, frequência respiratória e temperatura e esses dados pudessem ser visualizados de modo *online*. Dessa forma proporcionando uma alternativa para auxiliar no manejo bovino, visto que a captura dessas informações é realizada de forma não-invasiva, com os animais não-sedados e não-contidos. Ressalta-se que não houve trabalhos encontrados na literatura que abordavam a utilização do *bluetooth* para captura dessas informações.

O aplicativo *mobile* para *smartphones* e *tablets* foi desenvolvido, contendo as diversas funcionalidades a seguir:

- Transmissão de dados por *bluetooth*.
- Armazenamento das informações em banco de dados.

- Visualização das informações históricas de frequência cardíaca, frequência respiratória e temperatura, separadas por animal.
- Recebimento e visualização dessas informações de forma online.
- Cálculo das frequências cardíaca e respiratória.
- Central de solicitações para inicio e fim de captura de informações dos sensores.

Para a estimativa das informações de frequência cardíaca e respiratória de bovinos, algoritmos criados pelo autor em [1] foram reescritos neste projeto para o aplicativo *mobile*. Experimentos e testes foram realizados visando a validação dos dados adquiridos através dos sensores e dos algoritmos.

Parte dos projetos e desenvolvimentos realizados no âmbito deste trabalho constituíram um pedido de patente de invenção, solicitado junto ao INPI, com pesquisadores da UFMS e da Embrapa Gado de Corte. Esse pedido de patente foi intitulado como: "Equipamento, método e sistema não intrusivos de monitoramento de sinais fisiológicos de animais e parâmetros ambientais". Número do registro: BR10201600222, data de depósito: 20/01/2016.

## 4.1 Contribuições do trabalho desenvolvido

As contribuições deste trabalho podem ser elencadas como desenvolvimento da versão 1.0 do aplicativo *mobile*, com todas as funcionalidades informadas no Capítulo 3, a por-

tabilidade dos algoritmos de frequência cardíaca e respiratória de outras linguagens de programação para a linguagem de programação Java, além de testes e experimentos para garantia do funcionamento correto dos algoritmos, e ajuda no desenvolvimento do código fonte existente na placa de circuito impresso (*firmware*) utilizada neste projeto.

## 4.2 Dificuldades encontradas

No decorrer do desenvolvimento deste trabalho, surgiram algumas dificuldades que tiveram que ser superadas, tais como:

- Geração dos gráficos com os *samples* PPG *online*: foram testadas algumas bibliotecas que plotassem os *samples* da forma necessária e que fossem atualizando conforme mais pontos fossem inseridos.

- Portabilidade do algoritmo de frequência respiratória: foi difícil portar o algoritmo, devido o algoritmo ter sido desenvolvido utilizando a ferramenta MATLAB, a qual tem uma linguagem própria, com funções nativas que tiveram que ser interpretadas, reescritas e testadas dentro do aplicativo.

- Teste do algoritmo de frequência respiratória: testes foram realizados para garantir que o algoritmo portado estava de acordo com o algoritmo original. A dificuldade maior foi a validação de cada trecho do algoritmo, onde comparações dos resultados de saída tiveram que ser realizadas entre os algoritmos.

- Os parâmetros de entrada do algoritmo de frequência respiratória: o algoritmo de frequência respiratória exige que alguns parâmetros de entrada sejam definidos para que a estimativa seja realizada. Entretanto esses parâmetros têm que ser definidos manualmente para cada conjunto de *samples* informado, ou seja, os parâmetros definidos podem ser adequados para um conjunto de *samples* e não ser úteis para outro conjunto. Assim, houve uma grande dificuldade para encontrar corretamente esses parâmetros de entrada.

## 4.3 Trabalhos futuros

Como trabalhos futuros, pode-se citar:

- Implementar mais funcionalidades no aplicativo conforme demanda.
- Refatorar código fonte.
- Melhorar *layout* do aplicativo, visando a usabilidade por parte do usuário.
- Melhorar os algoritmos de estimativa de frequência cardíaca e respiratória.
- Tornar os parâmetros de entrada do algoritmo de frequência respiratória dinâmicos.
- Implementar ações em lotes, como configuração dos sensores.
- Possibilitar a conexão do aplicativo com mais de um dispositivo por vez.
- Desenvolver tela de login restringindo as funcionalidades por perfil.

- Armazenar os dados na nuvem (*cloud computing*).

- Desenvolver o aplicativo para o sistema operacional móvel iOS.

# Referências Bibliográficas


[1] P. O. Bressan, "Algoritmos para obtenção de frequência cardíaca e respiratória em bovinos." *Dissertação de Mestrado. Universidade Federal de Mato Grosso do Sul, Faculdade de Computação - FACOM. Campo Grande - MS*, 2015.

[2] C. Marin, "Sistema para aquisição e processamento de temperatura cutânea em bovinos e variáveis ambientais." *Dissertação de Mestrado. Universidade Federal de Mato Grosso do Sul, Faculdade de Computação - FACOM. Campo Grande - MS*, 2016.

[3] P. McDermott-Wells, "What is bluetooth?." *IEEE Potentials*, vol. 23, pages. 33-35, 2005.

[4] T. S. Siqueira, "Bluetooth – características, protocolos e funcionamento." *Instituto de Computação, Universidade Estadual de Campinas*, 2006.

[5] C. Y. Kobayashi, "A tecnologia bluetooth e aplicações." *Computação Móvel, IME - USP*, 2004.

[6] H. Jaap, "Bluetooth—the universal radio interface for ad hoc, wireless connectivity."

[7] "Seekic web site." Disponível em: http://www.seekic.com/blog/project_solutions/2011/08/18/Wireless_Connectivity__ISM_WLAN_WMAN_Bluetooth_et_al_3.html. Acesso em: 20-12-2016.

[8] "UFRJ web site." Disponível em: http://www.gta.ufrj.br/grad/09_1/versao-final/bluetooth/Page323.htm. Acesso em: 20-12-2016.

[9] F. H. Nishihara, "Bluetooth." 2004.

[10] M. S. Kay, "Sistema embarcado reconfigurável para aquisição de sinais de eletrocardiograma." *Dissertação de Mestrado. Universidade Federal de Mato Grosso do Sul, Faculdade de Computação - FACOM. Campo Grande - MS*, 2014.

[11] "Bluetooth." Disponível em: http://www.bluetooth.com. Acesso em: 20-12-2016.

[12] "Bluesmirf silver web site." https://learn.sparkfun.com/tutorials/using-the-bluesmirf. Acessado em 20-12-2016.

[13] "Apache Software License web site." Disponível em: http://www.apache.org/licenses/LICENSE-2.0. Acesso em: 20-12-2016.

[14] "IDC Corporate USA web site." Disponível em: http://www.idc.com/. Acesso em: 20-12-2016.

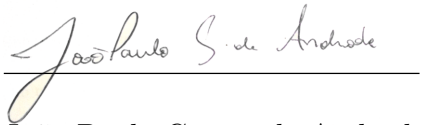
[15] "AChartEngine web site." Disponível em: http://www.achartengine.org/. Acesso em: 20-12-2016.

[16] E. F. Codd, "A relational model of data for large shared data banks." *IBM Research Laboratory, San Jose, California*, 1970.

# Declaração de Autoria e Responsabilidade

João Paulo Gomes de Andrade, residente e domiciliado na cidade de Campo Grande, Estado do Mato Grosso do Sul, portador do RG de nº 001350636 e CPF nº 019.481.461-00, declaro que a dissertação apresentada, com o título "BEP 1.0: Aplicativo Android para visualização e armazenamento de informações fisiológicas e ambientais"é de minha autoria e assumo a total responsabilidade pelo seu conteúdo e pela originalidade do texto. Declaro que identifiquei e referenciei todas as fontes e informações gerais que foram utilizadas para construção do presente texto. Declaro também que este artigo não foi publicado, em parte, na íntegra ou conteúdo similar em outros meios de comunicação, tendo sido enviado com exclusividade para a defesa da dissertação do curso de Mestrado Profissional em Computação Aplicada da Universidade Federal de Mato Grosso do Sul (UFMS).

Campo Grande, 16 de Fevereiro de 2017.



João Paulo Gomes de Andrade

RA 20143672