CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Relatório de Iniciação Científica

PIBIC/CNPq

Yulli Dias Tavares Alves

DESENVOLVIMENTO DE UMA INTERFACE TÁTIL BASEADA EM PYSIDE PARA INTERAÇÃO COM SISTEMAS DE MONITORAÇÃO DE SINAIS VITAIS DE NEONATOS

Orientador: Profª. Drª. Ana Paula Batista

Coorientador: Prof. Dr. Giovani Guimarães Rodrigues

Belo Horizonte,

Outubro de 2018

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

Relatório de Iniciação Científica

PIBIC

Yulli Dias Tavares Alves

DESENVOLVIMENTO DE UMA INTERFACE TÁTIL BASEADA EM PYSIDE PARA INTERAÇÃO COM SISTEMAS DE MONITORAÇÃO DE SINAIS VITAIS DE NEONATOS

Relatório técnico apresentado para a conclusão da iniciação científica, projeto orientado pela Dra. Ana Paula Bastista e coorientado pelo Dr. Giovani Guimarães Rodrigues, como introdução ao meio da pesquisa e metodologia científica.

Orientador: Prof[a]. Dr[a]. Ana Paula Batista

Coorientador: Prof. Dr. Giovani Guimarães Rodrigues

Belo Horizonte,

Outubro de 2018

Resumo

Em aplicações biomédicas, interfaces de usuário são utilizadas em geral para disponibilizar informações sobre variáveis fisiológicas, no entanto, o crescimento do número de variáveis em análise e a alta probabilidade de falsos alarmes tem aumentado as exigências do clínico responsável. Fato este que, destaca a importância do desenvolvimento de dispositivos adequados que auxiliem a equipe clínica na tomada de decisões. Nesse projeto foi desenvolvido uma interface tátil utilizando a linguagem Python, o *framework* PySide e a biblioteca PyQtGraph para interação com sistemas de monitoração de sinais vitais em neonatos. A interface desenvolvida exibe gráficos de temperatura, umidade, frequência cardíaca e saturação de oxigênio, sendo capaz de exibir outros gráficos. Os resultados obtidos neste trabalho foram satisfatórios havendo a interação adequada da interface com os sensores para exibir os gráficos em tempo real. Em um trabalho futuro pode-se emitir alarmes quando as variáveis fisiológicas apresentarem anormalidade. A interface tátil desenvolvida nesse projeto será utilizada em aplicações na área biomédica em UTIs neonatal. Entretanto, as habilidades desenvolvidas possuem aplicação no desenvolvimento de sistemas para diversas áreas da engenharia.

Palavras-chave: Interface Gráfica, PySide, Neonato, UTI neonatal, PyQtGraph, Interface tátil, Python, Sinais vitais .

Lista de Figuras

Sumário

Capítulo 1

Introdução

1.1 Apresentação do Problema

A recente difusão de microcomputadores de baixo custo e razoável poder de processamento abre novos caminhos para o desenvolvimento de sistemas embarcados com propósitos diversos. O uso de hardware de aplicação livre como Raspberry Pi possibilita o desenvolvimento de sistemas com baixo custo e relativa flexibilidade (BATISTA, 2017).

O Raspberry Pi 3 Model B possui um processador quad-core de 64 bits, 1 GB memória RAM, portas USB e HDMI, interface para câmera e *display*, além de módulos de comunicação em rede cabeada e sem fio (FOUNDATION, 2018). O acoplamento do Raspberry a um módulo de aquisição de dados permite monitorar e registrar a evolução temporal de variáveis do mundo real. Com essa configuração, é possível, por exemplo, registrar o comportamento de uma grandeza durante um dado período, registrar os valores em um banco de dados, processar os dados para extrair tendências ou mesmo compactar a informação e por fim transmitir os resultados para exibição em um software gráfico. Esse arranjo constitui então um módulo de monitoramento com inteligência para extrair informações dos dados registrados (BATISTA, 2016).

Uma área potencial para a aplicação de um sistema embarcado desse tipo é a monitoração de variáveis fisiológicas de recém nascidos (MOUSTAPH, 2007; POUPYREV; MARUYAMA, 2003). O monitoramento adequado dessas variáveis fisiológicas e ambientais é de extrema importância para a sobrevivência dos neonatos. Dessa forma, um módulo de monitoração destes sinais pode ser acoplado a sistemas de controle automático das variáveis relacionadas ou mesmo emitir alarmes quando essas atinjam limites que possam ameaçar a saúde da criança. É importante que os sons de alerta não sejam ouvidos dentro da UTI neonatal, pois os efeitos sonoros podem causar prejuízo à saúde do neonato (PEDRON; BONILHA, 2008). De acordo com (COSTA et al., 2015) o recém-nascido prematuro exposto a altos níveis de ruído pode sofrer dano sensório neural, estresse e distúrbios de linguagem ou auditivos, sendo que os ruídos entre 70 dBA e 80 dBA podem causar apneia, bradicardia e hipertensão. Sendo assim, o ruído causado pelos alarmes deve ser controlado, evitando risco a saúde dos recém nascidos.

A partir do controle da umidade e temperatura, a incubadora neonatal deve manter

um ambiente termo neutro, semelhante ao útero materno. Além disso, a manutenção da temperatura é importante para a proteção contra hipoglicemia, desconforto respiratório, distúrbios de coagulação e apneias causadas pela hipo ou hipertermia (AMORIM et al., 2014). Sendo assim, o controle da temperatura precisa ser feito adequadamente, por isso a visualização gráfica das variáveis de temperatura e umidade podem auxiliar a equipe médica no controle adequado das variáveis de ambiente. A partir dos gráficos a equipe clinica poderá controlar a temperatura da incubadora de forma que a saúde do recém nascido não seja prejudicada.

Há alguns anos iniciou-se uma preocupação em desenvolver interfaces interativas com uso de dispositivos sensíveis ao toque (MOUSTAPH, 2007; POUPYREV; MARUYAMA, 2003). Em (POUPYREV; MARUYAMA, 2003) é apresentado a implementação de uma interface sensível ao toque para dispositivos móveis. Em (KANG, 2014), uma pesquisa sobre design de interface de usuário (GUI), projeto de experiência do usuário (UXD) e design de informação, com enfoque para área de serviços de saúde de militares é apresentada. Em (JIA et al., 2006) é proposto uma interface de usuário inteligente que auxilia no diagnóstico de epilepsia a partir do processamento de sinais de eletroencefalografia e magnetoeletroencefalogria. Em aplicações biomédicas, monitores são utilizados para disponibilizar informações sobre variáveis fisiológicas contribuindo para o diagnóstico adequado do paciente pela equipe médica (BATISTA, 2017). Dessa forma, os gráficos exibidos são de suma importância para o acompanhamento da saúde dos pacientes.

Nesse projeto foi desenvolvido uma interface sensível ao toque para interagir com sensores responsáveis pela monitoração de sinais vitais de bebês recém-nascidos. Os sinais como: temperatura, umidade, frequência cardíaca e saturação de oxigênio no sangue são medidos em tempo real e enviados para a interface por meio da interligação dos Raspberrys formando uma rede de computadores. Nessa rede há um banco de dados, onde os dados que identificam os dispositivos são mantidos.

Temos como desafios desse projeto a continuação de um trabalho previamente iniciado pelo aluno de Engenharia de Computação João Guilherme Monteiro Guimarães e a comunicação com os sensores desenvolvidos pelos alunos de Engenharia Elétrica Ana Carolina Duque de Almeida e Mark Carmo Testi Moreira, todos discentes do CEFET-MG. A interface utiliza, também, um banco de dados que teve a contribuição dos alunos Cristóvão Olegário de Castro da faculdade PUC Minas, Gabriela Ramalho Santos e Raylander Frois Lopes do CEFET-MG.

A interação com uma tela sensível ao toque reduz número de periféricos acoplados a interface, tais como teclado e mouse, acompanhando a tendência do mercado e proporcionando uma interação simplificada com a tela se a interface for amigável. No desenvolvimento utilizou-se as funcionalidades do PySide para desenvolver uma interface tátil capaz de facilitar a monitoração e visualização dos sinais vitais de neonatos para auxiliar a equipe médica. O baixo custo para implementação do sistema e a disponibilização dos

códigos-fonte são os diferenciais desse projeto que tem o objetivo de contribuir com a área biomédica.

1.2 Objetivos

O objetivo principal deste projeto é desenvolver uma interface tátil que será executada em um dispositivo touch acoplado a um Raspberry PI para interação com sistemas de monitoração de sinais vitais em neonatos. Essa interface será utilizada em aplicações na área biomédica em UTIs neonatais. O desenvolvimento da interface será feito em Python utilizando o *framework* PySide.

1.3 Estrutura do relatório

Este relatório está organizado em seis capítulos e dois apêndices. Este capitulo tem a intenção de contextualizar e apresentar os objetivos e motivações da pesquisa. O segundo capítulo introduz o conceito de interface e apresenta pesquisas anteriores, que serviram de motivação para este estudo. No terceiro capitulo há uma explicação sobre conceitos importantes abordados nesse projeto, sendo eles o PySide, a *thread* e o *socket*.

No quarto capítulo é apresentado uma descrição do desenvolvimento da interface, bem como as decisões tomadas durante a sua criação. No quinto e no sexto capítulo são apresentados os resultados desse trabalho e uma discussão sobre o projeto e sua contribuição para a comunidade. No Apêndice A é apresentado um tutorial sobre a instalação do Raspbian e no Apêndice B é explicado o procedimento para utilizar o Pyreverse e gerar um diagrama de classes a partir do código fonte.

Capítulo 2

Revisão bibliográfica interfaces biomédicas

2.1 O estado da arte

2.1.1 Interfaces convencionais

A interface de usuário de sistema consiste em aspectos com a qual o usuário tem contato, tanto psicológica, perceptiva ou conceitualmente (MORAN, 1981, tradução nossa). A interface engloba tanto *hardware* quanto software. Os componentes de *hardware* são dispositivos com os quais os usuários realizam as atividades motoras e perceptivas. Entre eles estão tela, teclado, mouse, *tablets*, monitores e impressoras. O software da interface é a parte do sistema que implementa os processos computacionais necessários para controle dos dispositivos de *hardware*, construção dos objetos de com os quais o usuário possa interagir, geração dos diversos símbolos e mensagens que representam as informações do sistema, e interpretação dos comandos dos usuários. A interface tem por objetivo fundamental determinar como o usuário pode efetivamente interagir com o sistema e mostrar para o usuário quais as funções da aplicação o sistema oferece (LEITE, 2001).

Será considerado neste trabalho como interface convencional aquela que não possui o recurso tátil e necessita de periféricos como teclado, mouse e botões físicos para compor os elementos de *hardware* da interface. Nesse tipo de interface foi desenvolvido em (CIFUENTES; PRIETO; MÉNDEZ, 2011) um sistema de treinamento para a equipe médica utilizando a linguagem Python. Nesse sistema é simulado as possíveis patologias de neonatos e os principais sinais biomédicos como ECG, pressão, pulso, saturação de CO_2 e frequência cardíaca. Na interface gráfica são mostrados os resultados gerados e ajustados por modelos matemáticos que descrevem os sinais vitais de neonatos patológicos e não. O monitor neonatal é interativo e permite gerar diferentes cenários para assistir professores e alunos durante o treinamento de diagnóstico médico. O trabalho preliminar foi satisfatório, apresentando um erro inferior a 5% quando comparados os dados reais com os sinais obtidos pelo simulador.

Os sinais pulso cardíaco, saturação de oxigênio no sangue e tempo de gestação foram utilizados em (CAMARGO et al., 2014) para detectar a apneia de prematuridade. Quando pausas na respiração são detectadas, o sistema automaticamente ativa um dispositivo vibratório que realiza a estimulação vibrotátil e promove o retorno de movimentos respiratórios.

Desenvolvido na plataforma LabView, o sistema denominado "Anjo"foi processado em um notebook, nele são exibidos os gráficos de SpO2 e frequência de pulso e suas frequências de alarmes. Utilizou-se o armazenamento em arquivos das variáveis monitoradas para que posteriormente sejam descritos cálculos estatísticos, gráficos e tabelas.

2.1.2 Interfaces táteis

Hoje a tela pôde tornar-se móvel ofertando expressivas possibilidades de interação, produtividade e entretenimento, a partir da sua associação com a tecnologia *touch*. As interfaces táteis permitem que a interação seja feita utilizando como *hardware* somente uma tela sensível ao toque, dispensando a necessidade do uso de fios, hardwares, botões físicos e demais periféricos como teclado e *mouse*. O acesso facilitado aos dispositivos móveis permitiu o uso de celulares como auxiliares no acompanhamento da saúde dos pacientes fora das clínicas e hospitais.

O recurso tátil foi utilizado em (YEN; CHANG; YU, 2013) em um aplicativo Android que monitora ECG em tempo real e reconhece a arritmia instantaneamente. A mobilidade dos *smartphones* e o grande número de usuários Android contribuem para uma solução com um custo inferior aos produtos que existem no mercado. Além disso a interface sensível ao toque dos *smartphones* diminui a quantidade de periféricos agregados a solução. Nela são exibidos um gráfico de ECG e uma classificação do batimento cardíaco.

Em (GRADL et al., 2012) também foi desenvolvido um aplicativo capaz de monitorar o sinal de ECG em tempo real e classificar os batimentos cardíacos como normais ou anormais. Os dados podem ser transmitidos para visualização em tempo real ou o banco dados para os sinais que foram gravados. Na visualização em tempo real os gráficos são exibidos na interface, onde o sinal de ECG e a frequência cardíaca são mostrados na parte inferior da tela. Utilizou-se as cores verde e vermelho para indicar batimentos normais e anormais, respectivamente.

Um aplicativo desenvolvido em (GRADL et al., 2013) para o sistema Android, foi utilizado para classificar as fases do sono de pacientes e servir como ferramenta de prevenção. O sono desempenha um papel fundamental na vida humana, um terço da vida gasta-se dormindo, mas é necessário ter a quantidade e qualidade adequada para uma vida saudável, pois a falta de sono é associada a algumas doenças crônicas, como a diabetes, depressão e obesidade. Podendo ser utilizado pelo paciente em casa, o aplicativo apresenta um hipnograma e estima a progressão das fases do sono.

Capítulo 3

Conceitos abordados neste projeto

3.1 PySide

Um *framework* provê uma solução para uma família de problemas semelhantes. O PySide é um *framework* de interface gráfica para a linguagem Python lançado em 2009 que utiliza as bibliotecas do Qt [1] escritas em C++ para o seu próprio funcionamento. O PySide apresenta a vantagem de ser um software livre, diferente de outros *frameworks* comerciais (BATISTA, 2017).

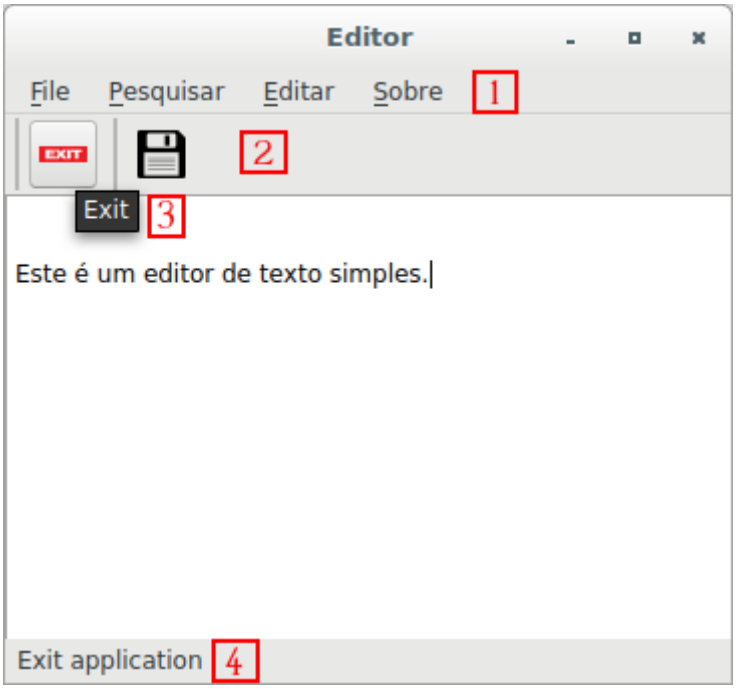
Inicialmente foi feita uma revisão de conceitos sobre os componentes básicos de tela utilizando o *framework* PySide. Seguindo tutorial (ZETCODE, 2011), foi implementado um editor de texto simples para testar os conceitos aprendidos. A classe PySide.QtGui.QTextEdit provê uma *widget* que é usada para editar e exibir textos. O editor de texto foi desenvolvido utilizando essa classe, e os menus, botões, barra de ferramentas e de status foram implementadas. Os itens destacados na figura 3.1 indicam:

1. Menu
2. Barra de ferramentas
3. Label do botão "*Exit*"
4. Mensagem na barra de status

Além disso, foi estudado o mecanismo de comunicação entre objetos utilizando *signals* e *slots*. *Signal* é um sinal que é emitido quando um determinado evento ocorre, avisando que o estado de um objeto mudou. Já o *slot* é uma função que é chamada quando um sinal em que o mesmo esta conectado é emitido. Um exemplo simples de utilização desse tipo de comunicação acontece quando clicamos no botão *Exit*, que aparece na imagem 3.1. Esse botão possui um *slot* que chama a função de encerrar a aplicação quando o mesmo é pressionado.

[1] O Qt é um *framework* comercial multi-plataforma que auxilia na criação e construção de aplicações nativas e interfaces de usuários.

Figura 3.1: Editor de texto criado durante o estudo inicial do Framework PySide

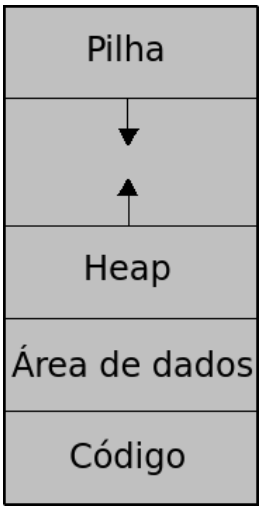


Fonte: Elaborada pela autora

3.2 *Thread*

Essa seção foi escrita com base no livro de (SILBERSCHATZ; GALVIN; GAGNE, 2012). Um processo é um programa em execução que inclui a área de código, a pilha onde os dados locais são temporariamente alocados, a área de dados que contém as variáveis globais e o *heap* que contém os dados alocados dinamicamente durante a execução do processo. A estrutura de um processo na memória é mostrado na figura 3.2.

Figura 3.2: Estrutura de um processo na memória



Fonte: Elaborada pela autora.

A *Thread* é a unidade básica de utilização da CPU. Ou seja, se houver um processo com mais de uma *thread*, cada *thread* do processo será executada separadamente, pois a CPU não processa mais do que uma *thread* por vez.

As *threads* de um mesmo processo compartilham área de código, área de dados e *heap*, cada *thread* possui sua própria pilha, onde as variáveis locais são armazenadas. Um processo *multithread* possui mais de uma *thread* e pode executar mais de uma tarefa ao mesmo tempo. Esse recurso é utilizado neste trabalho para atualizar mais de um gráfico, se não houvesse o recurso *multithread* somente um gráfico poderia ser atualizado por vez.

Os benefícios de um processo com múltiplas *threads* é maior em uma arquitetura com múltiplos processadores, onde as *threads* podem executar paralelamente em diferentes núcleos. Considere uma aplicação com quatro *threads*. Em um processador com um único núcleo, as *threads* serão executadas concorrentemente, porque um núcleo só é capaz de executar uma *thread* por vez. Em um sistema com quatro núcleos teremos todas as *threads* executando em paralelo, cada uma em um núcleo.

3.3 Comunicação entre processos através de uma rede de computadores

O *socket* permite uma comunicação ponto a ponto entre processos que pertencem a mesma rede, podendo ou não estar no mesmo computador (HARSCHUTZ, 2015). Diversas aplicações utilizam *socket* para se comunicar. O navegador web utiliza *sockets* para requisitar páginas e um sistema quando se integra a um banco de dados abre um *socket* (PANTUZA, 2017).

Existem dois tipos de *socket*: TCP e UDP. No protocolo TCP os dados enviados são confirmados quando recebido pelo destinatário, enquanto no protocolo UDP a transmissão de dados é feita sem verificar que o destinatário recebeu os dados. A ausência da etapa de verificação dos dados contribui para que o UDP seja mais rápido que o protocolo TCP. A escolha entre os protocolos dependerá do objetivo da aplicação, em uma transação bancária onde a confiabilidade dos dados é importante, deve-se utilizar o TCP. Já em uma transmissão de vídeo, o UDP é mais adequado, porque receber os dados atuais é mais importante do que atrasar a transmissão para receber os dados perdidos. Neste projeto utilizaremos o *socket* UDP.

Capítulo 4

Metodologia

Este capítulo aborda os passos adotados durante o andamento da pesquisa. Constam aqui tarefas e decisões tomadas para sua realização, bem como os recursos utilizados para tal. Ressalta-se a importância do *framework* Pyside, com um pacote de interface gráfica que permite aos desenvolvedores do Python o acesso as bibliotecas do Qt, provendo recursos como *thread* e abstração de rede. Também utilizou-se o PyQtGraph, uma biblioteca de interface gráfica e de usuário para Python que provê gráficos rápidos e interativos para exibir dados. Na interação entre os dispositivos utilizou-se o *socket* por permitir uma comunicação entre processos em diferentes dispositivos.

4.1 Detalhamento do desenvolvimento Inicial da interface

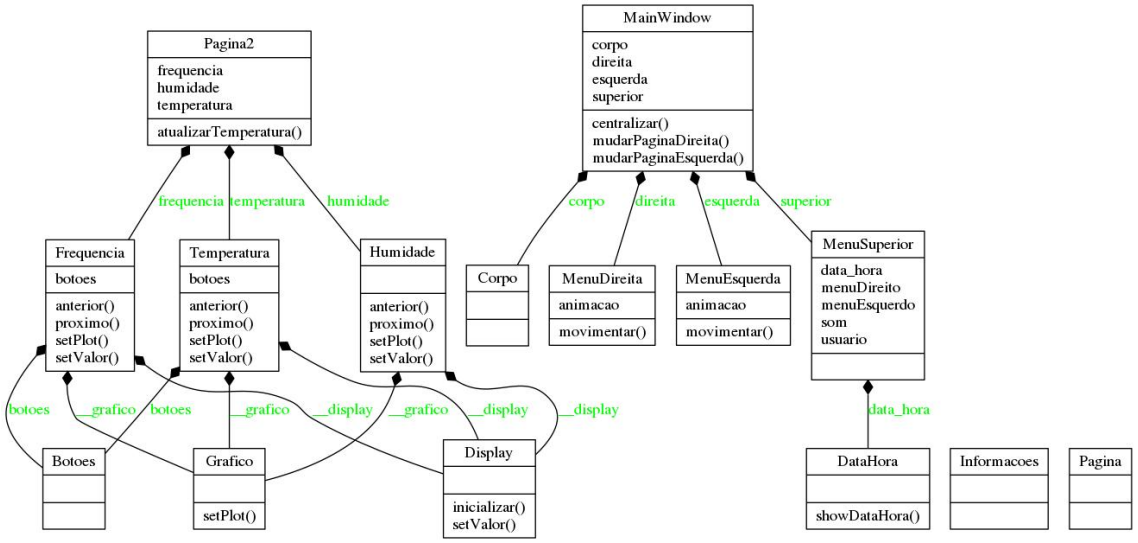
O desenvolvimento de uma interface preliminar começou com o aluno João Guilherme Monteiro Guimarães no Centro de Pesquisa Inteligente (CPEI). Ele foi o responsável por desenvolver a parte visual inicial da interface. Com o objetivo de obter o diagrama de classe do projeto preliminar mostrado na figura 4.1, utilizou-se o Pyreverse, o procedimento utilizado para gerar o diagrama está descrito no apêndice B. No diagrama mostrado na figura 4.1 não foram representados os construtores, por isso classes que só possuem o método construtor aparecem em branco.

As classes descritas a seguir, estão representadas no diagrama 4.1. Na classe MainWindow são definidos a cor de fundo, o título e o tamanho da tela. Ela é composta pelas classes Corpo, MenuDireita, MenuEsquerda e Menu Superior. Na classe Corpo são criadas as telas para exibir os gráficos e mostrar o status da conexões. Não existe um teste para validar essas conexões com a interface.

A classe MenuEsquerdo controla a exibição do menu mostrado na figura 4.3 e a classe MenuDireito exibe a opção de alterar a configuração da hora do sistema, figura 4.4, esse recurso não foi implementado. Esses menus possuem o método movimentar, o responsável pela animação de recolher ou expandir o menu.

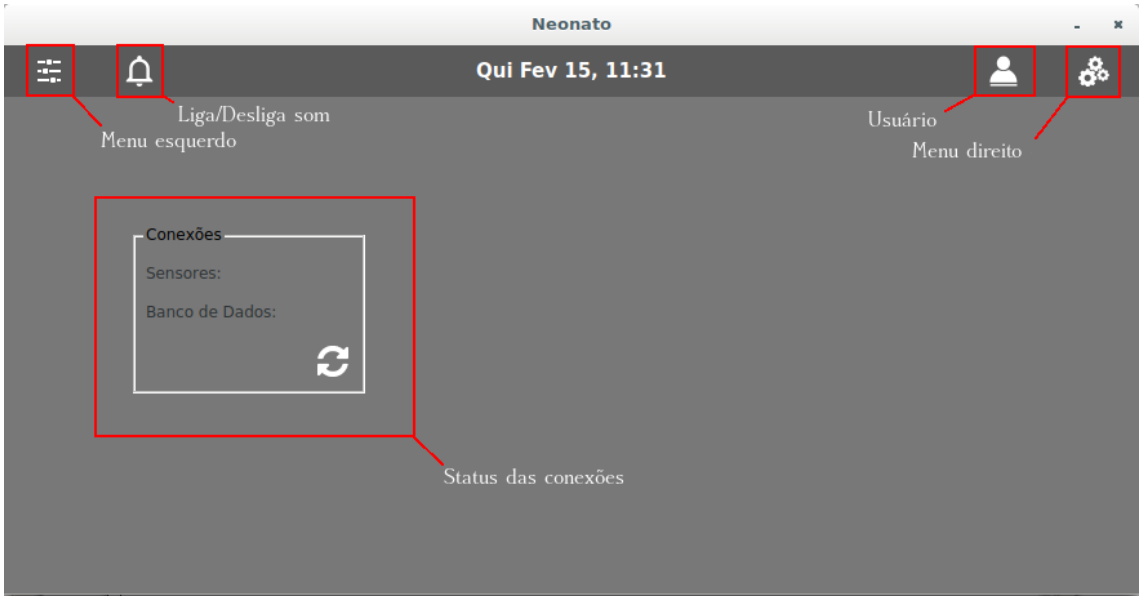
Os ícones que controlam o som e exibem o perfil do usuário podem ser vistos no topo da janela na figura 4.2. As ações desses botões não haviam sido implementadas e, portanto, não realizam nenhuma função. A classe MenuSuperior é composta pela classe DataHora que atualiza o horário exibido na interface de acordo com a hora do sistema.

Figura 4.1: Diagrama de Classe da interface preliminar gerado utilizando o Pyreverse



Fonte: Elaborada pela autora

Figura 4.2: Interface preliminar desenvolvida em trabalho anterior - Tela inicial

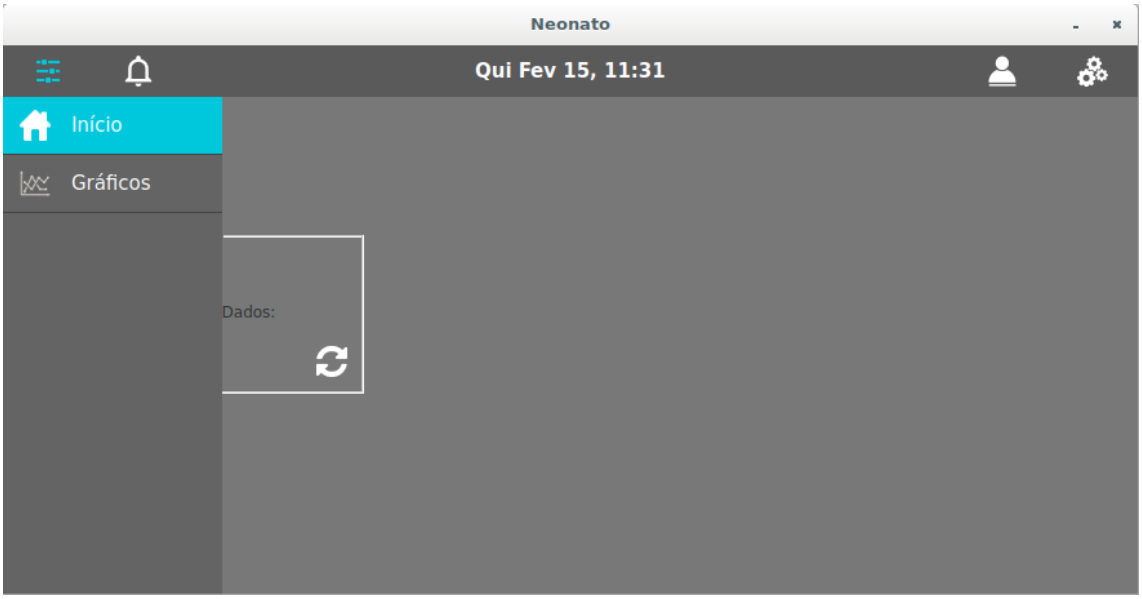


Fonte: Elaborada pela Autora

A classe Corpo cria os objetos das classes Informacoes e Pagina2, pertence a classe PySide.QtGui.QStackedWidget e por isso cria uma pilha com os seus objetos, mostrando apenas um por vez. A classe informações mostra na tela de status da conexão 4.2 e a Pagina2 contém os Gráficos.

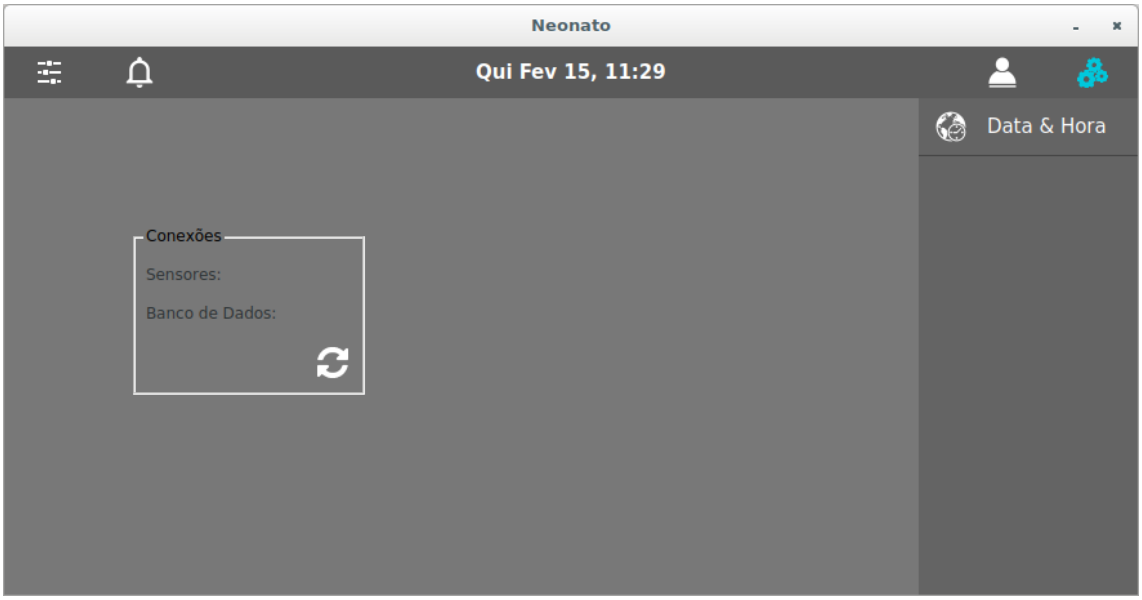
A classe Pagina2 também possui um objeto da classe PySide.QtGui.QStackedWidget, sendo composta pelas classes Temperatura, Frequencia e Humidade. Essas classes são responsáveis por atualizar os gráficos e possuem dois métodos: anterior e próximo. Esses métodos são chamados quando os respectivos botões do gráfico 4.5 são pressionados.

Figura 4.3: Interface preliminar desenvolvida em trabalho anterior - Menu Esquerdo



Fonte: Elaborada pela autora

Figura 4.4: Interface preliminar desenvolvida em trabalho anterior - Menu Direito



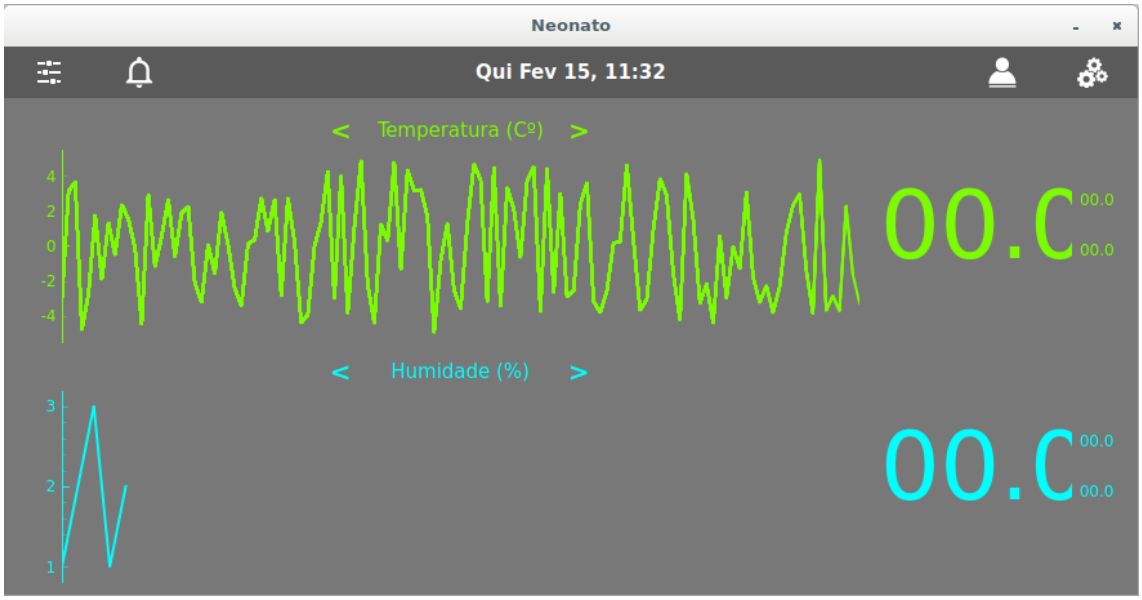
Fonte: Elaborada pela autora

O método setPlot() chama o método de mesmo nome da classe Gráfico, para o traçar. O método setValor() é responsável por atualizar os valores dos *displays*, números que aparecem ao lado do gráfico. Apesar de haver essa função, os valores dos *displays* não são alterados durante a execução. Na figura 4.5 podemos ver os gráficos Temperatura e Humidade. Apenas o gráfico de temperatura está sendo atualizado com valores aleatórios, em um tempo fixo pré-definido, pelo método atualizarTemperatura presente na classe Pagina2 da figura 4.1, os outros gráficos recebem valores iniciais que não são modificados pela

interface.

Embora apenas um gráfico seja atualizado durante a execução, a interface trava. Isso ocorre porque o método atualizarTemperatura acrescenta um valor aleatório no fim do vetor a cada 50 milissegundos, fazendo-o crescer infinitamente. Dessa forma o processo atinge seu limite de memória e sua execução é interrompida.

Figura 4.5: Interface preliminar desenvolvida em trabalho anterior - Gráficos



Fonte: Elaborada pela autora

4.2 Novas funcionalidades agregadas a interface

Nessa seção é feito um comparativo com a interface preliminar desenvolvida em trabalho anterior descrita na seção 4.1. Além disso, são utilizado os conceitos descritos no capítulo 3 para explicação das funcionalidades agregadas a interface. Será abordado também as decisões de projeto utilizadas no desenvolvimento da interface tátil que tem por objetivo auxiliar a equipe médica na monitoração de sinais vitais coletados a partir da interação com sensores e exibidos em tempo real.

Algumas mudanças no *desing* da interface foram feitas. O botão responsável por ligar ou desligar o som, exibido na figura 4.2, foi retirado porque quando a interface for capaz de emitir alarmes, espera-se que os mesmos não sejam silenciados. Com o objetivo de alertar a equipe médica, os alarmes serão soados quando as variáveis de ambiente não estiverem adequadas para a saúde do neonato. Dessa forma, desligar o som pode trazer um risco a vida do recém nascido, pois os alertas serão emitidos pela interface mas não serão ouvidos pela equipe médica.

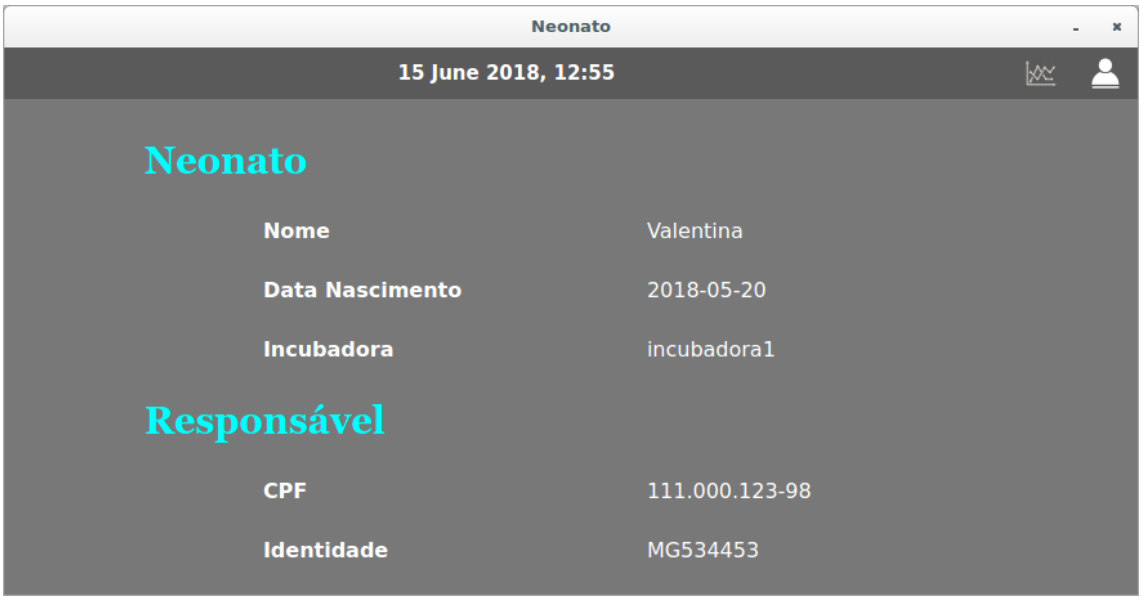
Após essas alterações a interface ficou com duas telas que podem ser acessados por botões no menu superior, sendo elas a tela de gráficos 4.6 e a tela de perfil do neonato 4.7. Os menus esquerdo e direito também foram retirados. O menu esquerdo da figura 4.3 se

tornou desnecessário ao retirar a pagina inicial (figura 4.2) e o menu direito (figura 4.4) foi retirado porque a data e hora da interface são definidas a partir das informações do sistema. Dessa forma, a hora do Raspberry deve ser ajustada corretamente.

Figura 4.6: Tela de gráficos exibindo temperatura e umidade

Fonte: Elaborada pela autora.

Figura 4.7: Tela de perfil exibindo dados de um neonato fictício

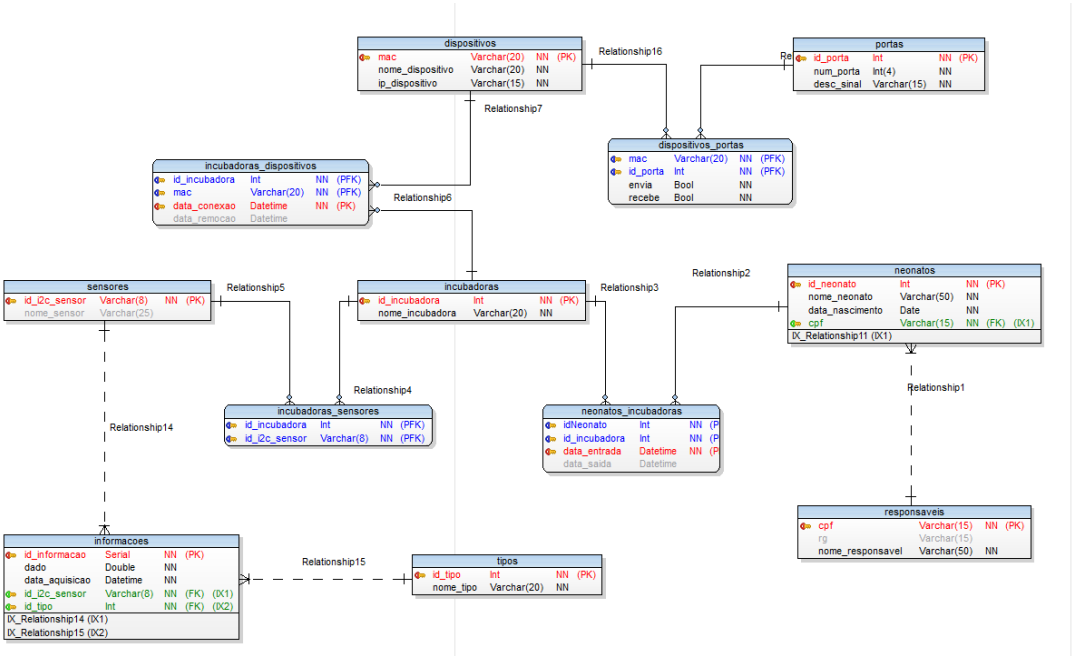


Fonte: Elaborada pela autora.

Os dados exibidos na figura 4.7 foram coletado do banco de dados desenvolvido pelo Raylander Frois Lopes, apresentado na figura 4.8, onde podemos identificar os relacionamentos entre as tabelas. Esse banco tem por objetivo armazenar os dados coletados pelos

sensores, determinar a interação entre os dispositivos conectados a incubadora e armazenar os dados dos neonatos.

Figura 4.8: Modelo relacional do banco de dados utilizado neste projeto



Fonte: Raylander Frois Lopes

Os dados coletados pelos sensores são simultaneamente exibidos na interface e armazenados no banco de dados. Nas tabelas responsáveis, neonatos, neonatos_incubadoras, incubadoras, incubadora_dispositivos, dispositivos, dispositivos_portas, portas e seus relacionamentos são obtidas as informações necessárias para interação da interface com os sensores. Essas tabelas são utilizadas para se obter informações do neonato, da incubadora e dos dispositivos conectados a ela.

Além disso a tabela dispositivos_portas é usada pela interface ao utilizar o *socket* desenvolvido pela Gabriela Ramalho Santos. Permitindo a comunicação entre processos que estão sendo executados em diferentes computadores, o *socket* também permite a comunicação independente da linguagem utilizada. O *socket* UDP foi desenvolvido para a interação da interface com os *software* de ECG, temperatura e SpO2.

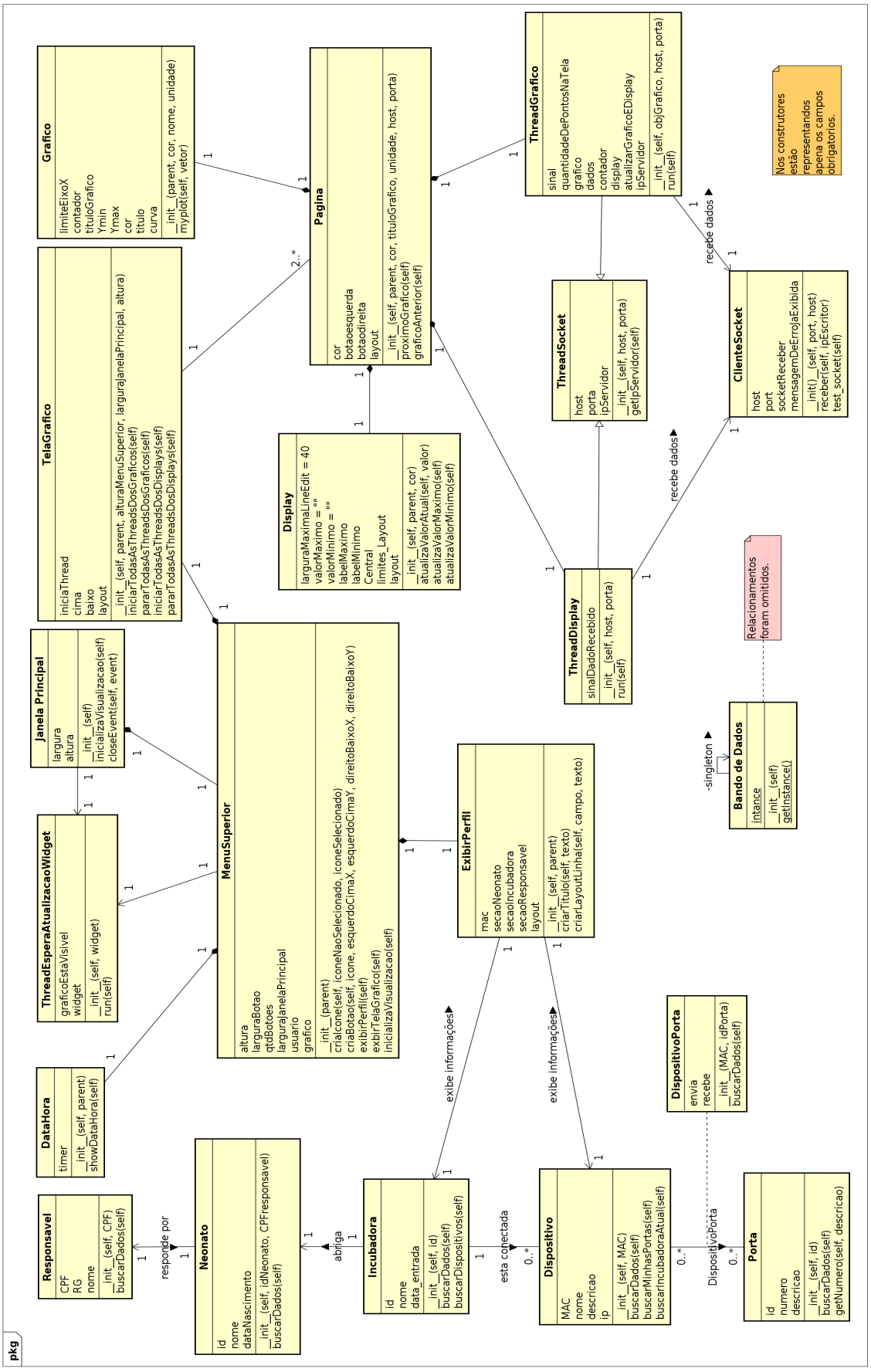
A interface utiliza o *socket* para receber os dados enviados pelos sensores e utiliza o banco de dados para descobrir de quais dispositivos ela deve receber dados. O cadastro dos Raspberrys que interagem com a interface trás um fator de segurança ao sistema, uma vez que o *socket* recebe dados através de uma porta e qualquer dispositivo poderia enviar dados para essa porta, bastando conhecer o IP do Raspberry conectado a interface. Dessa forma, a interface poderia receber dados indevidos e exibi-los nos gráficos ou *displays*, o que prejudicaria a avaliação médica trazendo um risco a saúde do paciente.

Para evitar essa situação os Raspberrys devem ter um IP estático e estarem cadastrados

no banco para interagir com a interface. Somente um dispositivo pode enviar dados para uma determinada porta, dessa forma não haverá dois dispositivos enviando dados para uma determinada porta. Caso o mesmo dispositivo necessite enviar mais de um dado, poderá fazer isso enviando cada dado para a sua porta adequada.

O diagrama de classe obtido do trabalho anterior, representado na figura 4.1, foi o ponto de partida para entender o funcionamento do sistema desenvolvido previamente e conseguir remodelá-lo para adequá-lo a implementação das novas funcionalidades da interface como a exibição dos gráficos em tempo real, a atualização dos *displays*, a utilização de *threads* e a comunicação com os sensores. O diagrama deste projeto pode ser visto na figura 4.9, onde é possível identificar as classes, atributos e relações entre os objetos do sistema.

Figura 4.9: Diagrama de Classe da Interface Tátil desenvolvida nesse Projeto



Fonte: Elaborada pela Autora

Os objetos da classe ClienteSocket recebem os dados de uma porta e caso o IP do dispositivo que enviou o dado seja o esperado, os objetos da classe ThreadDisplay ou ThreadGrafico receberão os dados. A *thread* que atualiza o gráfico pode atualizar também o *display* caso seja necessário que o mesmo dado exibido no gráfico seja visto no valor atual do *display*. Se for desejado que o valor atual seja atualizado por um valor recebido pelo *socket*, então deve-se utilizar a classe ThreadDisplay. Para facilitar a identificação do que é entendido como *display* na interface, o mesmo foi destacado na figura 4.10.

O recurso de exibir um valor no *display* que seja diferente do valor representado no gráfico pode ser necessário, por exemplo, para exibir os batimentos cardíacos obtidos a partir de uma modelagem matemática, sendo enviados para o *display* através do *socket*. Dessa forma, os batimentos cardíacos poderiam ser exibidos no *display* ao lado do gráfico de ECG.

Os valores máximos e mínimos destacados na figura 4.10 são definidos pelo o usuário e podem ser utilizados em um trabalho futuro para emitir sons de alerta. Nessa figura 4.10 podemos identificar a "Página", representada no diagrama 4.9 pela classe de mesmo nome e composta pelas classes ThreadGrafico, ThreadDisplay, Grafico e Display.

Figura 4.10: Explicação dos componentes principais da tela de gráficos

Fonte: Elaborada pela autora.

Capítulo 5

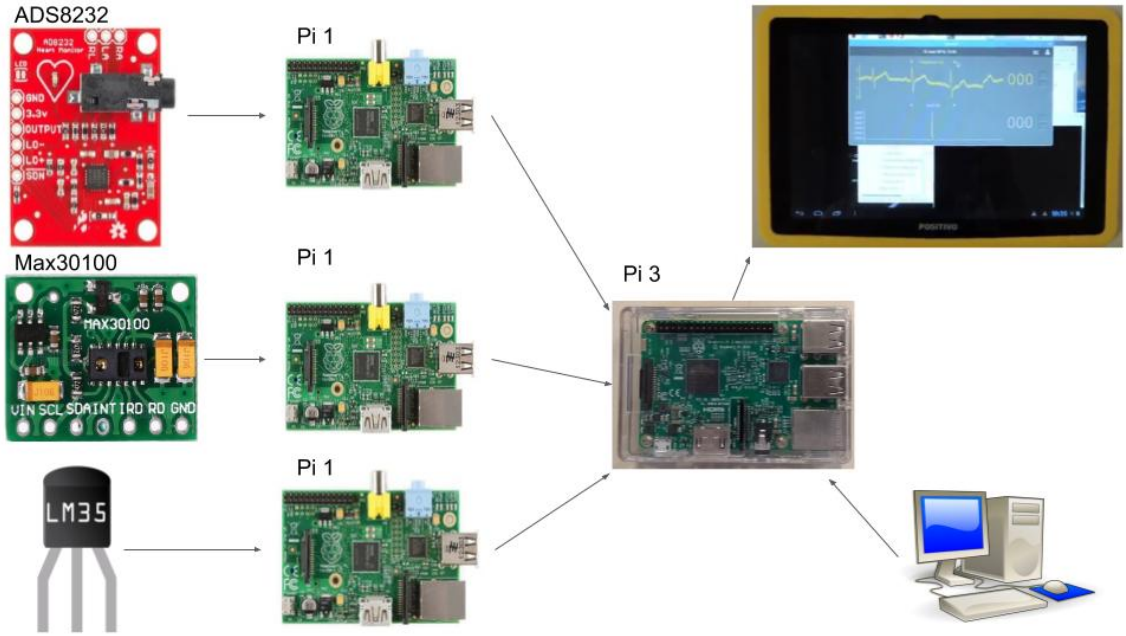
Resultados

5.1 Equipamentos utilizados

Para a validação da interface, projetou-se um experimento em que três sensores conectados a unidades Raspberry distintas. Utilizou-se três Raspberry Pi 1 Model B e um Raspberry Pi 3 Model B, sensores, *tablet* e um computador de mesa. Os sensores LM35, Max30100 e ADS8232 foram utilizados para monitorar respectivamente os sinais de temperatura, SpO2 e ECG, o processamento dos sensores foi realizado no Raspberry Pi 1. Também utilizou-se um *tablet* para exibir a tela do Raspberry Pi 3, onde foi processada a interface gráfica. Como não havia sensor de umidade, o *display* correspondente foi atualizado com valores aleatórios por um programa executado no computador de mesa.

A figura 5.1 mostra a interligação dos equipamentos no experimento de teste da interface. Veja que os sensores são conectados aos Raspberrys Pi 1 onde um software é responsável por coletar os dados, processar e enviar para o Pi 3. Um computador foi utilizado para representar a funcionalidade dos *displays* de receberem dados pela rede. O Raspberry Pi 3 foi utilizado na interface por possuir uma maior capacidade de processamento através de um processador com 4 núcleos, enquanto o Pi 1 possui somente um núcleo de processamento. Dessa forma, é possível ter no Pi 3 quatro *threads* executando no sistema enquanto no Pi 1 haverá uma.

Figura 5.1: Interligação dos equipamentos utilizados na apresentação de resultados



Fonte: Compilação elaborada pela autora. ADS8232 (ELETRONICS, 2018), Max20100 (GEARBEST, 2018), LM35 (CIRCUITO.OI, 2018), Raspberry Pi 1 (HüMMLER, 2014), Computador de mesa (WIKICLIPART, 2018), Raspberry Pi 3 e *Tablet* são fotos da autora.

5.2 Configurações necessárias para a realização do experimento

5.2.1 Banco de Dados

A partir da versão 8.0 do MySQL o *plugin* de autenticação padrão não é compatível com a versão 2.7 do Python. Para resolver esse problema de compatibilidade é necessário alterar o *plugin* de autenticação padrão para o aquele utilizado nas versões anteriores. Em (MYSQL, 2018) é apresentado o que deve ser feito para mudar o *plugin* padrão.

Os Raspberrys conectados aos sensores e a interface devem ser cadastrados no banco e devem estar conectados na mesma incubadora. Os dados recebidos pela interface respeitam a estrutura definida no banco. Conforme podemos ver no diagrama 4.8, as tabelas incubadoras, incubadora_dispositivos, dispositivos, dispositivos_portas e portas devem ser necessariamente preenchidas para que seja possível a interação dos sensores com a interface.

5.2.2 Raspberry conectados a Sensores

Os Raspberrys devem pertencer a mesma rede para que seja possível a comunicação dos dispositivos. Dentro dessa rede somente um dispositivo deve ser capaz de enviar para uma porta e somente a interface pode receber, isso deve ser garantido pelo banco de dados. Os softwares dos sensores devem consultar a tabela portas 4.8 para encontrar qual porta devem enviar os dados processados. Encontrado a porta adequada ao sinal transmitido, é

necessário fazer uma consulta na tabela dispositivo_porta para encontrar o IP da interface. Com os dados corretos de porta e IP, será possível utilizar o *socket* para comunicação com a interface.

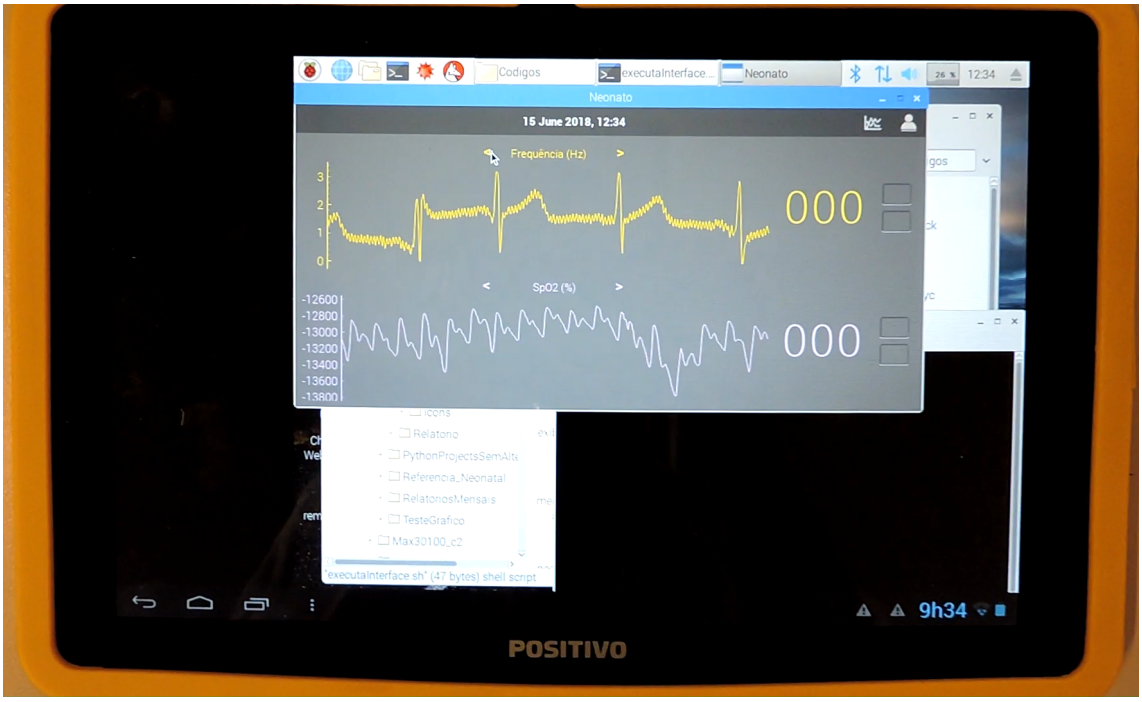
5.2.3 *Tablet*

O *tablet* também deve estar na mesma rede dos dispositivos. Na apresentação de resultados utilizamos o *tablet* que possuía conexão wi-fi para exibir a tela do Raspberry. Por isso utilizou-se a comunicação *wireless* do *tablet* para que fosse possível usá-lo como um monitor do Raspberry Pi 3. As instruções necessárias para tal configuração estão disponíveis em (MILTON, 2017).

5.3 Análise dos resultados

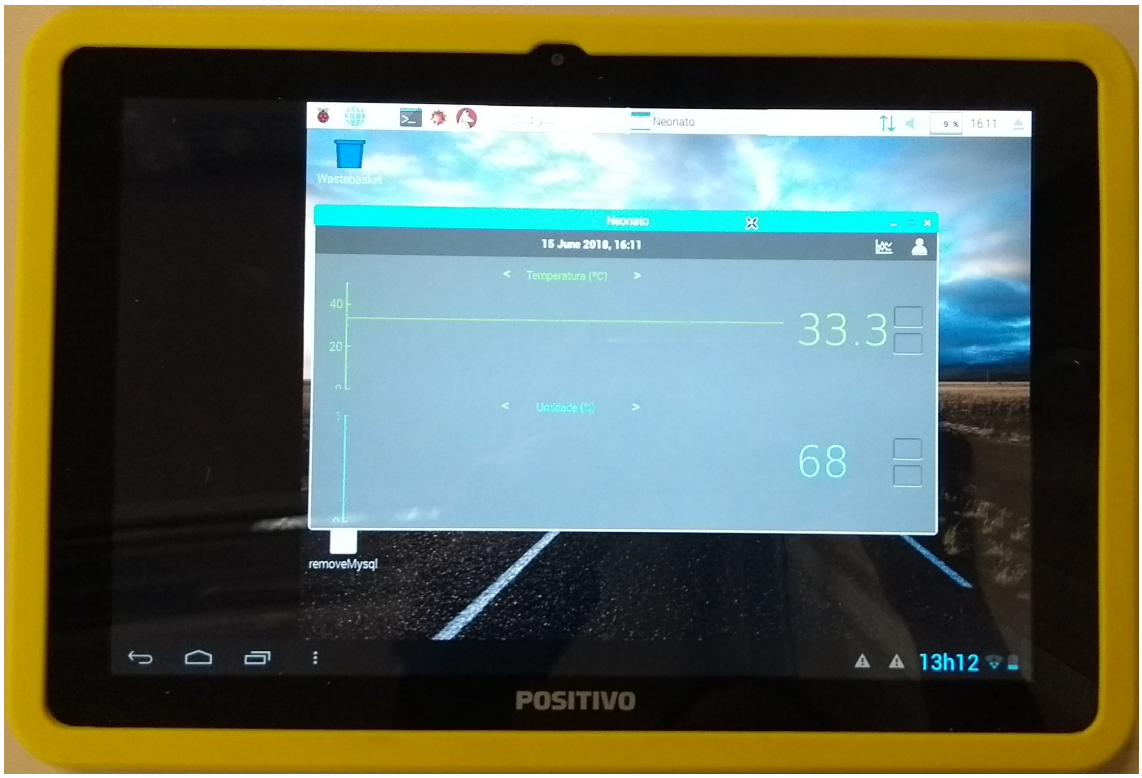
A comunicação com os sensores funcionou de acordo com o esperado e a atualização em tempo real da interface funcionou corretamente ao exibir valores adequados aos coletados pelos sensores. As figuras 5.2, 5.3 e 5.4 mostram a exibição da interface no *tablet*.

Figura 5.2: Exibindo os gráficos de ECG e Spo2 no *tablet*



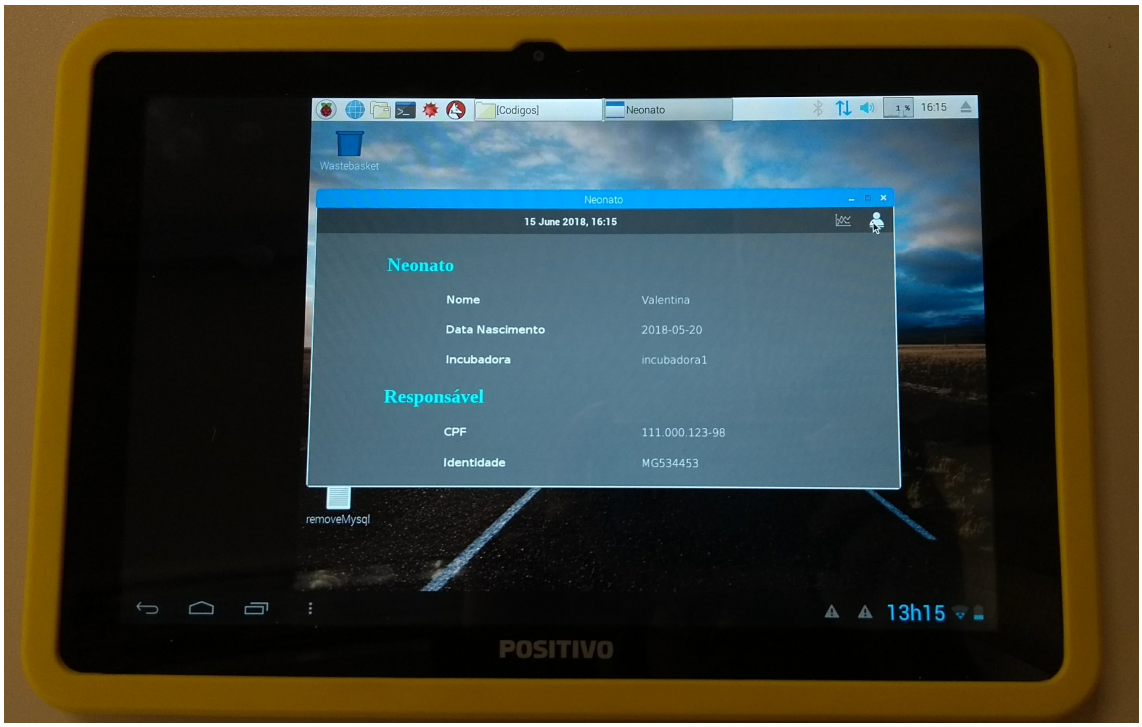
Fonte: Elaborada pela autora.

Figura 5.3: Exibindo os gráficos de temperatura e umidade no *tablet*



Fonte: Elaborada pela autora.

Figura 5.4: Exibindo o perfil do neonato com dados fictícios no *tablet*



Fonte: Elaborada pela autora.

Para melhorar a visualização da interface na tela, pode-se utilizar o *display touchscreen* para o Raspberry Pi como o mostrado na figura 5.5. Ao utilizá-lo seria possível exibir a interface no modo *fullscreen*. Como esse tablet é conectado diretamente no Raspberry, não seria necessário realizar as configurações descritas na seção 5.2.3.

Figura 5.5: *Display* Raspberry Pi *Touchscreen* 7"

Fonte: Compilação elaborada pela autora. *Display* (FILIPEFLOP, 2018a), *Display* conectado ao Raspberry (FILIPEFLOP, 2018b).

Capítulo 6

Conclusão

Os bebês prematuros que vão para UTI neonatal precisam ter seus sinais vitais monitorados por diversos sensores enquanto são mantidos em uma ambiente adequado na incubadora. Nesse projeto foi desenvolvido uma interface capaz de exibir os sinais vitais de um neonato e auxiliar a equipe médica.

Como desafios do projeto tínhamos o desenvolvimento de uma interface que fosse capaz de exibir os sinais recebidos de sensores em tempo real utilizando o processador do Raspberry. Além disso, a programação multitarefas foi um fator importante para que os gráficos fossem atualizados paralelamente. Nesse projeto foi feito uma interface tátil que exibe os sinais de temperatura, umidade, ECG e SpO2 em tempo real e interage corretamente com os sensores capazes de coletar esse sinais.

Algumas funcionalidades podem ser agregadas a interface, tais como a emissão de alarmes para alertar a equipe médica. Os sons podem ser emitidos quando algum dos sinais monitorados apresentarem um valor fora da faixa adequada para a sobrevivência do neonato. Além disso, pode-se fazer uma diferenciação de sons dos alarmes de acordo com o risco envolvido, podendo variar o som ou até mesmo o volume. Pode-se também utilizar modelos matemáticos para detectar doenças a partir dos sinais recebidos pela interface.

A utilização do Raspberry para processamento dos sensores e da interface proporciona uma solução de baixo custo para aplicação em UTIs neonatais. Além do baixo custo para implementação da interface, pretende-se que a mesma seja facilmente reproduzida tanto em termos de *hardware* quanto *software*. Para isso o código fonte do projeto será disponibilizado no repositório do GitHub disponível em <https://github.com/yullidias/MonitorTatilNeonatal>.

Embora esse trabalho seja voltado para a área biomédica com foco na monitoração de sinais vitais de neonatos, a interface desenvolvida pode ser utilizada para a comunicação com sensores em geral, exibindo gráficos e valores nos *displays* conforme desejado.

Referências Bibliográficas

AMORIM, M. et al. Umidade relativa em incubadora neonatal: Implicações sobre a sensação térmica do neonato. 2014.

ANCLIN, E. *Pyreverse : UML Diagrams for Python*. 2008. Disponível em: <https://www.logilab.org/blogentry/6883>. Acesso em: 14 fev 2018.

BATISTA, A. P. *DESENVOLVIMENTO DE UM SISTEMA BASEADO EM RASPBERRY PI PARA MONITORAMENTO E PROCESSAMENTO DE SINAIS EXPERIMENTAIS*. 2016.

BATISTA, A. P. *DESENVOLVIMENTO DE UMA INTERFACE TÁTIL BASEADA EM PYSIDE PARA INTERAÇÃO COM SISTEMAS DE MONITORAÇÃO DE SINAIS VITAIS DE NEONATOS*. 2017.

CAMARGO, V. C. et al. Instrumentation for the detection and interruption of apnea episodes for premature newborn. In: *2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. [S.l.: s.n.], 2014. p. 2127–2130. ISSN 1094-687X.

CIFUENTES, J.; PRIETO, F.; MÉNDEZ, L. C. SIMULATION OF A NEONATAL MONITOR FOR MEDICAL TRAINING PURPOSES. *Revista EIA*, scieloco, p. 9 – 27, 12 2011. ISSN 1794-1237. Disponível em: <http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S1794-12372011000200002&nrm=iso>.

CIRCUITO.OI. *LM35 Analog Temperature Sensor*. 2018. Disponível em: <https://blog.circuito.io/wp-content/uploads/2017/02/LM35-illustration.png>. Acesso em: 20 jul 2018.

COSTA, E. J. L. et al. Uso adequado de incubadora neonatal na assistência em saúde. *Revista enfermagem*, v. 18, n. 4, 2015. ISSN 2238-7218.

ELETRONICS, U. *Arduino Single Lead Heart Rate Monitor - AD8232*. 2018. Disponível em: <https://uge-one.com/image/cache/catalog/catalog/0%20UGE%20Heart%20Rate%20Monitor%20AD8232-b-500x375.jpg>. Acesso em: 20 jul 2018.

FILIPEFLOP. *Display Raspberry Pi Touchscreen 7"Raspberry Pi*. 2018. Disponível em: <https://uploads.filipeflop.com/2017/07/Pi_Screen-01.jpg>. Acesso em: 20 jul 2018.

FILIPEFLOP. *Display Raspberry Pi Touchscreen 7"Raspberry Pi*. 2018. Disponível em: <https://uploads.filipeflop.com/2017/07/101754_e4f252f1-56e8-4a1f-b2e4-d16f7c1bd60c_large.jpg>. Acesso em: 20 jul 2018.

FOUNDATION, R. P. *Raspberry Pi 3 Model B*. 2018.

GEARBEST. *MAX30100 Pulse Oximeter Heart Rate Sensor Module - GREEN*. 2018. Disponível em: <https://gloimg.gbtcdn.com/gb/pdm-product-pic/Electronic/2016/08/30/thumb-img/1472500902191079205.jpg>. Acesso em: 20 jul 2018.

GRADL, S. et al. Real-time ecg monitoring and arrhythmia detection using android-based mobile devices. In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. [S.l.: s.n.], 2012. p. 2452–2455. ISSN 1094-687X.

GRADL, S. et al. Somnography using unobtrusive motion sensors and android-based mobile phones. In: *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. [S.l.: s.n.], 2013. p. 1182–1185. ISSN 1094-687X.

HARSCHUTZ, J. *network socket*. 2015. Disponível em: <https://whatis.techtarget.com/definition/sockets>. Acesso em: 05 ago 2018.

HüMMLER, T. *Workshop: Raspberry Pi als Mailserver*. 2014. Disponível em: <https://www.admin-magazin.de/var/ezflow_site/storage/images/das-heft/2014/05/workshop-raspberry-pi-als-mailserver/ita_0514_p10_00.jpg/134249-1-ger-DE/ITA_0514_P10_00.jpg_leadimage.jpg>. Acesso em: 20 jul 2018.

JIA, W. et al. An intelligent user interface system for diagnosis of epilepsy. In: *Proceedings of the IEEE 32nd Annual Northeast Bioengineering Conference*. [S.l.: s.n.], 2006. p. 131–132. ISSN 2160-6986.

KANG, J. Healthboard: A graphic user interface for patient centered healthcar. *Parsons Journal for Information Mapping*, New York, United States, 2014.

LEITE, J. C. *Design de Interfaces de Usuário*. 2001. Disponível em: <https://www.dimap.ufrn.br/~jair/ES/c6.html>. Acesso em: 15 jul 2018.

MILTON, A. *Como usar seu tablet / celular / mobile / android como monitor, teclado e mouse de um Raspberry / Raspbian*. 2017. Disponível em: <https://devblog.drall.com.br/como-usar-seu-tablet-celular-mobile-android-como-monitor-teclado-e-mouse-de-um-raspberry-raspbia
Acesso em: 22 jul 2018.

MORAN, T. P. The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man-Machine Studies*, v. 15, n. 1, p. 3 – 50, 1981. ISSN 0020-7373. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0020737381800223>.

MOUSTAPH, H. *Ambient Touch: Designing Tactile Interfaces for Handheld Devices, Visual Comput*. [S.l.: s.n.], 2007. v. 23.

MYSQL. *2.10.1.2 Changes Affecting Upgrades to MySQL 8.0*. 2018. Disponível em: <https://dev.mysql.com/doc/refman/8.0/en/upgrading-from-previous-series.html#upgrade-caching-sha2-password-compatibility-issues>. Acesso em: 22 jul 2018.

PANTUZA, G. *O que são e como funcionam os Sockets*. 2017. Disponível em: <https://blog.pantuza.com/artigos/o-que-sao-e-como-funcionam-os-sockets>. Acesso em: 05 ago 2018.

PEDRON, C. D.; BONILHA, A. L. d. L. Práticas de atendimento ao neonato na implantação de uma unidade neonatal em hospital universitário. *Revista gaúcha de enfermagem*, v. 29, n. 4, p. 612–618, 12 2008. Disponível em: <http://hdl.handle.net/10183/23611>. Acesso em: 25 set 2017.

POUPYREV, I.; MARUYAMA, S. Tactile interfaces for small touch screens. In: *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: ACM, 2003. (UIST '03), p. 217–220. ISBN 1-58113-636-6. Disponível em: <http://doi.acm.org/10.1145/964696.964721>.

SILBERSCHATZ, A.; GALVIN, P. B.; GAGNE, G. *Operating System Concepts*. 9th. ed. [S.l.]: Wiley Publishing, 2012. ISBN 1118063333, 9781118063330.

WIKICLIPART. 2018. Disponível em: <http://wikiclipart.com/wp-content/uploads/2017/01/Computer-clip-art-free-download-clipart-images.jpg>. Acesso em: 20 jul 2018.

YEN, T. H.; CHANG, C. Y.; YU, S. N. A portable real-time ecg recognition system based on smartphone. In: *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. [S.l.: s.n.], 2013. p. 7262–7265. ISSN 1094-687X.

ZETCODE. *PySide tutorial*. 2011. Disponível em: <http://zetcode.com/gui/pysidetutorial/>. Acesso em: 21 ago 2017.

Apêndice A

Instalação Raspbian

A instalação do Raspbian foi realizada utilizando o sistema operacional Debian GNU/-Linux. Para realizá-la, é necessário, preparar o cartão de memória formatando-o. Para isso foi utilizado o GParted, um editor de partições gráfico.

Inicialmente, verifique se o GParted está instalado digitando no terminal o seguinte comando:

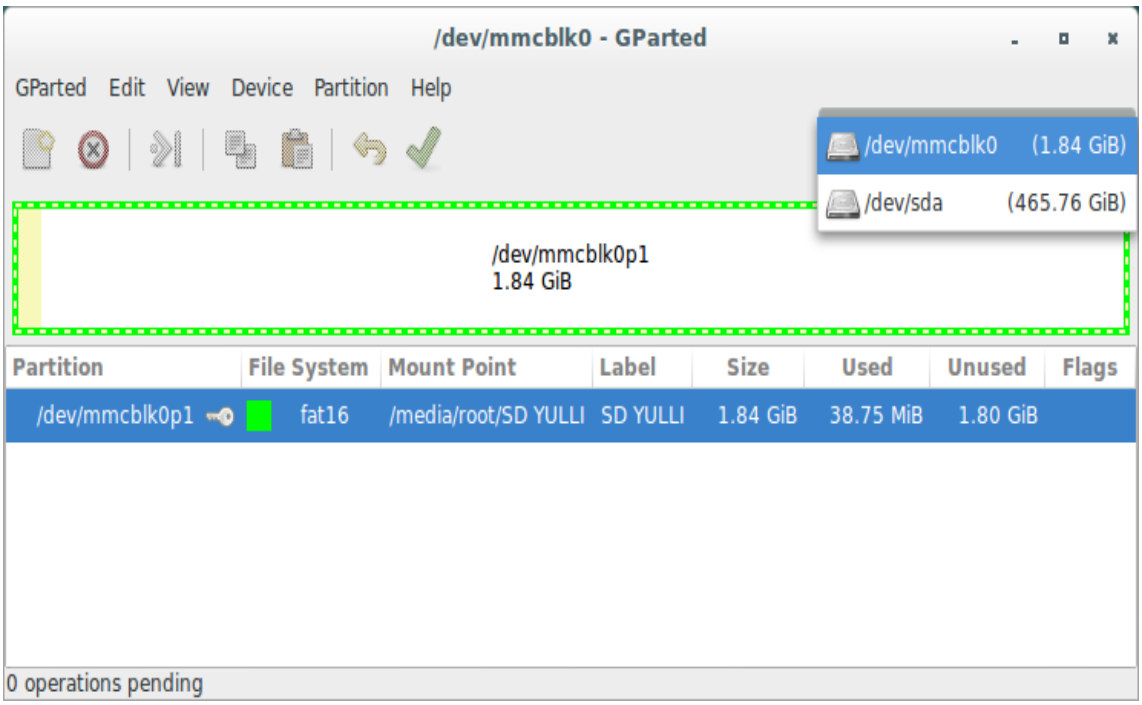
```
$ dpkg -l | grep gparted
```

Se nada for retornado, será necessário instalar o GParted, basta digitar:

```
$ sudo apt-get install gparted
```

Inicie o GParted e escolha o dispositivo que deseja formatar na lista no canto superior direito da tela, figura A.1.

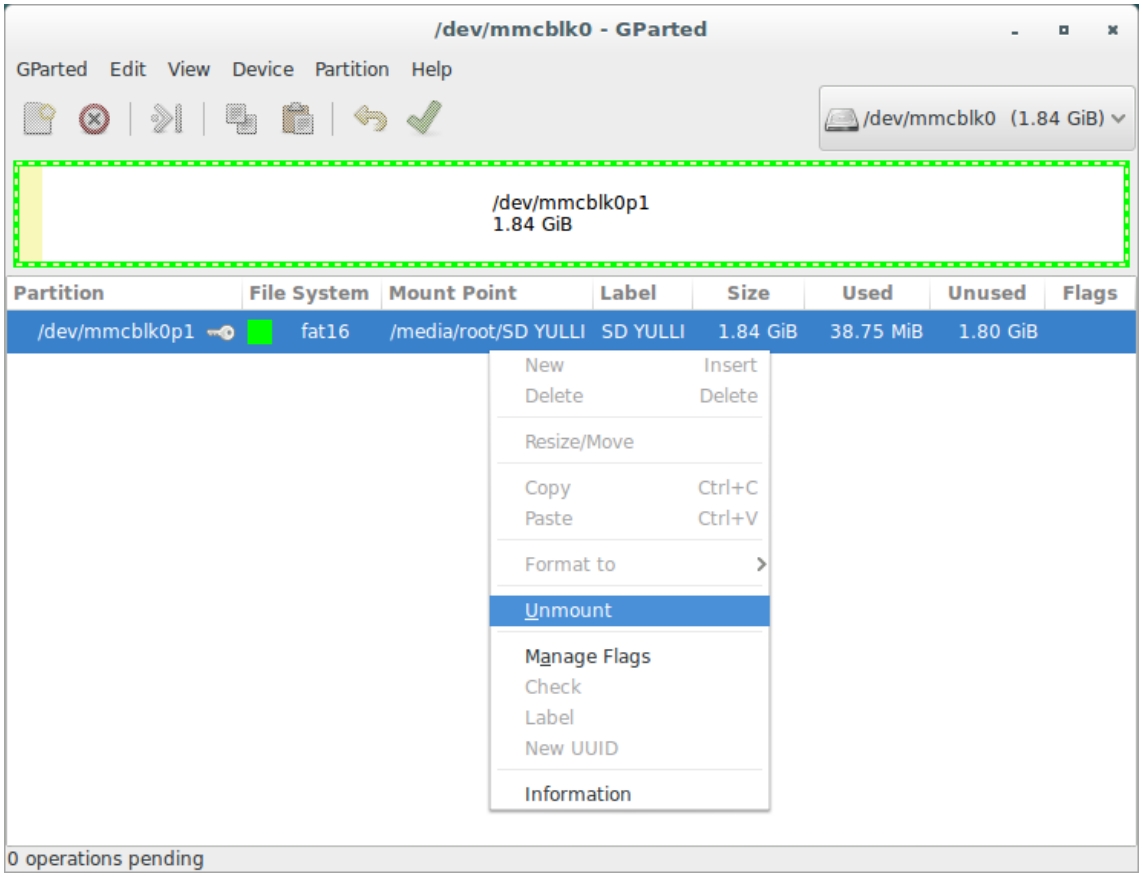
Figura A.1: Selecionando o dispositivo que será formatado no GParted



Fonte: Elaborada pela autora.

Desmonte o cartão de memória clicando com o botão direito do mouse na partição. De acordo com a figura A.2.

Figura A.2: Desmontando o dispositivo que será formatado no GParted



Fonte: Elaborada pela autora.

Clique com o botão direito na partição e selecione o sistema de arquivos FAT32. Depois vá no menu superior Edit e selecione "*Apply All Operations*"para iniciar a formatação.

Concluída a formatação, acesse [1] e faça o download do "*Raspbian Stretch with desktop*". Descompacte o arquivo, monte o dispositivo e liste os dispositivos através do comando:

```
$ lsblk
```

O cartão de memória normalmente fica localizado em /dev/mmcblk, mas caso haja dúvidas, retire o cartão e digite o comando anterior. Observe os resultado e insira novamente o cartão, seu dispositivo será o que apareceu na lista. Essa etapa deve ser feita com total atenção, pois o comando dd sobrescreve qualquer partição da máquina. Identificado o cartão corretamente, use o comando dd para copiar a imagem, da seguinte forma:

```
$ sudo dd if=caminho\_arquivo.img of=caminhoDispositivo.
```

Exemplo:

[1] https://www.raspberrypi.org/downloads/raspbian/

```
$ sudo dd if=2016-05-27-raspbian-jessie.img of=/dev/mmcblk0
```

A cópia é demorada e não é exibida mensagens sobre o andamento do processo, dessa forma aguarde pacientemente sua finalização. Quando for finalizado, retire o cartão do computador, insira-o no Raspberry e realize a instalação do sistema.

Apêndice B

Obter Diagrama de classe com o Pyreverse

O Pyreverse é um software que gera diagramas a partir dos códigos fonte em todos os formatos que o graphviz/dot conhece (ANCLIN, 2008). Inicialmente precisamos instalar os pacotes Pylint e Graphviz como o seguinte comando:

```
$ sudo apt-get install pylint graphviz
```

Após a instalação utilize o comando Pyreverse com o parâmetro -S para mostrar recursivamente todas as associações entre as classes.

```
$ pyreverse -S *
```

Esse comando deve ser executado na pasta onde estão os códigos fonte, veja que o '*' no comando se refere a todos os arquivos dessa pasta. É possível passar para o Pyreverse somente os arquivos que são do python usando '*.py' no lugar de '*'. Depois, utilize o comando dot para gerar a saída no formatado JPEG.

```
$ dot -Tjpeg classes_No_Name.dot -o class.jpeg
```

Pronto. No arquivo class.jpeg estará o diagrama correspondente ao código fonte passado para o pyreverse.