INSTITUTO FEDERAL
Paraíba
Campus João Pessoa

CURSO SUPERIOR DE BACHARELADO EM ENGENHARIA ELÉTRICA

ERON ALMEIDA VIEIRA DA SILVA

TRABALHO DE CONCLUSÃO DE CURSO

PROTÓTIPO DE MONITORAMENTO DE LEITO MÉDICO HOSPITALAR

João Pessoa
2022

Eron Almeida Vieira da Silva

PROTÓTIPO DE MONITORAMENTO DE LEITO MÉDICO HOSPITALAR

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Superior de Bacharelado em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Orientador:
Michel Coura Dias

João Pessoa
2022

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Nilo Peçanha do IFPB, *campus* João Pessoa

S586p Silva, Eron Almeida Vieira da.

Protótipo de monitoramento de leito médico hospitalar / Eron Almeida Vieira da Silva. – 2022.
45 f. : il.

TCC (Graduação – Bacharelado em Engenharia Elétrica) – Instituto Federal de Educação da Paraíba / Coordenação do Curso Superior em Engenharia Elétrica, 2022.

Orientação : Prof. MSc. Michel Coura Dias.

1. IOT. 2. UTI - equipamento. 3. Dashboard. 4. Grafana. 5. PostgreSQL. I. Título.

CDU 004.738.5:615.471(043)

Bibliotecária responsável: Lucrecia Camilo de Lima – CRB 15/132

Eron Almeida Vieira da Silva

Protótipo de monitoramento de leito médico hospitalar

Trabalho de Conclusão de Curso submetido à Coordenação do Curso Superior de Bacharelado em Engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Trabalho Aprovado em 15 / 08 / 2022 pela banca examinadora:

ICP Brasil
Documento assinado digitalmente
MICHEL COURA DIAS
Data: 05/02/2025 10:00:21-0300
Verifique em https://validar.iti.gov.br

Prof. Msc. Michel Coura Dias
IFPB (Orientador)

gov.br
Documento assinado digitalmente
HELDER ROLIM FLORENTINO
Data: 31/01/2025 15:36:48-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Cleumar da Silva Moreira
IFPB (Membro da Banca)

gov.br
Documento assinado digitalmente
CLEUMAR DA SILVA MOREIRA
Data: 09/12/2024 14:49:11-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Helder Rolim Florentino
IFPB (Membro da Banca)

Dedico este trabalho às diversas pessoas que um dia disseram que seria impossível, aos que também me desmotivaram e aqueles que me pediram para desistir dos meus sonhos.

AGRADECIMENTOS

Quero agradecer primeiramente a Deus e a minha família(pai, mãe e irmãos) que direta ou indiretamente sempre me apoiaram e acreditaram juntamente comigo no meu grande sonho.

RESUMO

Este trabalho descreve a utilização de tecnologias IOT para criação de um sistema de monitoramento em um ambiente hospitalar integrado a equipamentos presentes nos leitos de UTI. No entanto, o foco principal do projeto é uma solução conjunta para acompanhamento de informações integrada a diversos leitos hospitalares de maneira simultânea. Além disso, é possível gerar um relatório com dados históricos em tempo real, o que permite que os profissionais de saúde o utilizem como uma ferramenta auxiliar de uso rápido para o diagnóstico de enfermidades dos pacientes, mostrando-se assim, particularmente útil em ambientes hospitalares de nível critico como é o caso das unidades de terapia intensiva. Sua implantação também possui um baixo custo e pode ser facilmente integrado a equipamentos mais antigos, que se evidencia como sendo grandes vantagens da solução, ainda mais em um cenário de pandemia onde a ocupação de ambientes deste tipo aumentou drasticamente. Tanto a solução como a sua validação são descritas neste documento.

Palavras-chave: IOT, UTI, Dashboard, Grafana, PostgreSQL , Grafana, OpenCV.

ABSTRACT

This work describes the use of IoT technologies to create a monitoring system in a hospital environment integrated with equipment present in ICU beds. However, the main focus of this project is a joint solution integrated information monitoring to several hospital beds simultaneously. In addition, it’s possible to generate a report with historical data in real-time, which allows health professionals to use it as a quick-to-use auxiliary tool for diagnosing patients' illnesses, thus proving to be particularly useful in critical hospital environments such as is the case of intensive care units. Its implementation also has a low cost and can be easily integrated with older equipment, which is evident as a great advantage of the solution, even more so in a pandemic scenario where the occupation of environments of this type has increased dramatically. Both the solution and its validation are described in this document.

Key-words: IoT, ICU, Dashboard, Grafana, PostgreSQL, OpenCV

Lista de Ilustrações

Lista de Abreviaturas e Siglas

PEP	Prontuário Eletrônico do Paciente
IoT	Internet of Things
UTI	Unidade de Tratamento Intensivo
VGA	Video Graphics Array
USB	Universal Serial Bus
HDMI	High-Definition Multimedia Interface
FPS	Frames Por Segundo
OCR	Reconhecimento de Caracteres Ópticos
HP	Hewlett-Packard
SO	Sistema Operacional
SGBD	Sistema Gerenciador de Banco de Dados
TCP	Transmission Control Protocol

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos com o auxílio da Tecnologia da Informação diversos setores foram drasticamente alterados. Com um acesso rápido e tráfego gigantesco de informações, o mercado teve que se adaptar e melhorar o seu fluxo de dados. Todavia, o setor hospitalar não tem recebido grandes avanços técnicos quando o quesito é o tratamento e tráfego de dados de forma inteligente em países de média e baixa renda. Já nos países com rendas superiores existem uma série de normas éticas e duelos de interesses comerciais entre os estados e empresas, como diz o boletim da OMS(2015).

Não há dúvidas de que as tecnologias de informação e comunicação estão nos conduzindo a uma nova era, contudo não podemos deixar de nos preocupar com os limites da difusão dessas tecnologias, sobretudo na área de saúde: será que todos se beneficiarão do aumento das disponibilidades de recursos de saúde online, ou estamos caminhando para uma sociedade dos que têm e dos que não têm informações? (Thomas et al., 1998).

As tecnologias utilizadas no setor de saúde são responsáveis pela prevenção, tratamento e cuidado, incluindo medicamentos, procedimentos e suporte para a assistência do cliente/paciente. É grande o desafio do Sistema de Saúde se aprimorar e, sobretudo, garantir a incorporação e difusão das tecnologias em saúde, principalmente devido à realidade limitada de recursos econômicos.

1.1 MOTIVAÇÃO

Por muito tempo o atendimento hospitalar foi realizado apenas guardando os dados do paciente em uma folha de papel no período em que ele estava enfermo dentro da unidade hospitalar. Após a sua alta ou óbito, dificilmente esses dados poderiam ser novamente encontrados depois de um longo período de tempo. Essas informações simplesmente não eram armazenadas ou não tinham como serem guardadas devido a grande quantidade produzida diariamente por diversos pacientes.

Porém, como a tecnologia vem avançando rapidamente nos últimos anos, a estrutura desse tipo de atendimento e armazenamento dos dados mudou bruscamente.

Hoje um paciente ou médico consegue ter acesso ao seu PEP(Prontuário Eletrônico do Paciente) mesmo após anos a consulta ter sido realizada. Um médico pode realizar uma teleconsulta com um paciente a quilômetros de distância e conseguir diagnosticar a sua enfermidade.

Equipamentos conhecidos pela infraestrutura de “Internet das Coisas" ou “IoT”, podem se conectar à rede e extrair dados diários de um paciente com precisão. Essas informações podem compor padrões e identificar possíveis enfermidades sendo diagnosticados automaticamente por sistemas e ainda gerar relatórios e enviar automaticamente para médicos ou enfermeiros que tratam de pacientes no modo “Home Care”.

Toda essa tecnologia já existe e está implementada em algumas poucas unidades considerando de forma global, todavia, o preço para criação ou adaptação em alguns hospitais saem totalmente dos padrões financeiros das unidades de forma inicial o que é então considerado desnecessário para os gestores. O que não é avaliado em sua grande maioria é que a adoção do tratamento desses dados trará uma enorme redução de custos a longo prazo e muitas vezes passa despercebido pelos administradores do projeto como é bem descrito no artigo “Controle de custos assistenciais na saúde suplementar utilizando *big data* e *analytics* para prever comportamentos e antecipar cuidados aos beneficiários” escrito por pesquisadores da UFC (Universidade Federal do Ceará).

1.2 JUSTIFICATIVA

Os sistemas de atendimento hospitalar tem sofrido bastante devido ao breve contexto de pandemia mundial causado pelo COVID-19.

Segundo Mancini: “Os grandes reflexos dessa situação, segundo pesquisas, estão comprovados agora. Diante dos graves sintomas respiratórios que a Covid-19 provoca, especialmente com os pacientes que são acometidos na forma mais grave da doença, existe a necessidade de serem hospitalizados em leitos de Unidade de Terapia Intensiva (UTI), pois quanto antes eles são entubados, melhor é a resposta do tratamento e mais rápido diminui o índice de mortalidade da doença”.

A falta de estrutura adequada, equipamentos e tecnologias que dão suporte ao tratamento desta doença tem sido uma das principais causas de mortes no mundo. Neste ínterim a superlotação dos leitos hospitalares e a falta de profissionais de saúde para operá-los é uma realidade. Assim, o uso de ferramentas tecnológicas para otimizar a sua operação pode ser uma alternativa relevante para diminuir o impacto deste pico de ocupação das estruturas de atendimento médico mesmo em cenários de telemedicina.

Tentando aliar o tratamento remoto e seguro dos leitos hospitalares e de UTIs a ideia de desenvolvimento e criação de um software que centralizasse os dados e contribuísse de forma positiva para a melhoria no tratamento do paciente se torna algo cada vez mais essencial para o nosso cenário atual.

Com cada vez mais informações, a Saúde do futuro vai exigir maior organização e capacidade de análise desses dados. A expectativa é que, juntas, tecnologias de big data e analytics sejam um marco de transformação do segmento, já que é capaz de centralizar, entender e permitir a tomada de decisões mais assertivas, tanto do corpo clínico, quanto da parte administrativa(SAÚDE BUSINESS, 2020)

Para vislumbrar esse tipo de sistema é necessário a integração de várias ferramentas, sendo elas de software e hardware que serão abordadas a seguir.

1.3 Objetivo

Desenvolver uma solução para monitoramento centralizado e análise das informações dos pacientes para fins de auxílio ao diagnóstico médico.

1.4 Organização Textual

Esse trabalho de conclusão de curso está dividido nos seguintes capítulos e sub tópicos expostos a seguir incluindo o assunto a ser abordado:

- Capítulo 1 - Introdução, Motivação, Justificativa e Objetivos: Onde foram tratados todas as informações que motivaram e instigaram a desenvolver este projeto.

- Capítulo 2 - Fundamentação Teórica: serão descritos os conhecimentos que serviram com base para o desenvolvimento da solução técnica descrita neste trabalho..
- Capítulo 3 - É o capítulo que descreve a solução técnica fruto deste trabalho.
- Capítulo 4 - Apresentação e análise de resultados
- Capítulo 5 - Serão apresentadas as considerações finais do trabalho.

2 Fundamentação Teórica

2.1 Equipamentos de UTI

As UTI são unidades hospitalares destinadas ao atendimento de pacientes graves ou de risco que dispõem de assistência médica e de enfermagem ininterruptas, com equipamentos específicos próprios, recursos humanos especializados e que tenham acesso a outras tecnologias destinadas a diagnóstico e terapêutica. Estas unidades podem atender grupos etários específicos: neonatal (de 0 a 28 dias), pediátrico (de 28 dias a 14 ou 18 anos) e adultos que atendem pacientes maiores de 14 ou 18 anos (APARECIDA, 2004).

Este setor dispõe de equipamentos eletrônicos essenciais na manutenção da vida dos pacientes, utilizados na monitoração dos parâmetros fisiológicos ou como opção terapêutica integrada ao sistema de gases. Essas instalações devem ser integradas com a fonte de emergência que rapidamente assumem alimentação garantindo que o fornecimento não seja interrompido (CAMPOS, 2020).

A fim de centralizar e esmiuçar todos os equipamentos disponíveis em uma unidade, tivemos como objetivo principal centralizar todos os dados para que seja possível o rápido diagnóstico mesmo com os enfermeiros e médicos em outros ambientes aumentando a eficácia do tratamento na saúde dos pacientes.

Os principais equipamentos encontrados nas unidades são:

- Monitor cardíaco: É utilizado para fazer a leitura de diversos sinais vitais do paciente.

- Oxímetro de pulso: Pode ser integrado ao monitor cardíaco ou ser utilizado de forma isolada para análise de oxigênio no sangue.
- Sonda nasoenteral: Sondas necessárias para procedimentos, mas que não possuem considerados equipamentos que podem ser integrados a este projeto.
- Sonda vesical: Sondas utilizadas também no leito, mas que não possuem forma de integração a este projeto.
- Máscara e cateter de oxigênio: É utilizado para administrar oxigênio em pacientes sejam eles adultos ou pediátricos, no entanto, não podem ser integrados a este projeto.
- Cateter central: São sondas que são utilizadas para administração de medicamentos e que não possuem forma de integração com este projeto.
- Tubo orotraqueal: Tubo utilizado para ventilação mecânica, utilizado no procedimento conhecido como “entubação”. Não possui integração a este projeto.
- Ventilador mecânico: Utilizado para pacientes com insuficiência respiratória ele auxilia na inspiração e expiração do ar nos pulmões e que pode ser integrado a este projeto dependendo do modelo utilizado.
- Bomba de infusão: Possui função semelhante ao cateteres só que é utilizado de forma controlada e automática mediante a programação dos enfermeiros. Pode também ser integrado ao sistema integrador deste projeto.
- Cama motorizada: As camas nos leitos de UTI possuem motores para que possam ser regulados a sua altura e inclinação de acordo com a necessidade de cada caso.

Destes equipamentos podemos realizar a integração do monitor cardíaco, oxímetro de pulso, bomba de infusão e mais algum equipamento que tenha alguma forma de comunicação serial com alguma rede.

Figura 1 – Monitor multiparâmetros da Alfamed.

Fonte: https://alfamed.com/produto/produto-03/.

Devido a indisponibilidade de integrar todos os equipamentos de uma UTI foi utilizado em nosso protótipo apenas o monitor cardíaco para ser integrado ao sistema de centralização dos dados.

O monitor possui a leitura simultânea de diversos dados no corpo do paciente. É amplamente utilizado não só em UTIs como também em outros tipos de leitos devido a sua versatilidade e praticidade de leitura e exibição dos dados.

Monitor multiparamétrico como também é conhecido é um dos principais equipamentos médicos hospitalares utilizados por profissionais na área para visualizar e acompanhar indicadores de saúde do paciente. São dispositivos capazes de realizar medidas de parâmetros variados do corpo, exibindo em uma tela dados como eletrocardiograma (ECG), frequência cardíaca (FC), saturação de oxigênio (SpO2),

frequência respiratória (FR), temperatura (ºC), pressão arterial não invasiva, dentre outros(RODRIGUES, 2019).

Conforme a figura 1, o monitor do fabricante Alfamed do modelo Vita 1200, podemos extrair por exemplo os dados de: Frequência cardíaca, pulsação, oximetria, pressão não invasiva e a sua média, pressão arterial e a sua média, pressão parcial de CO2, temperatura corporal, traçado de capnografia e diversos outros dados. Além de possuir interfaces seriais como VGA, USB e um ponto de RJ45 para conexão com a rede. Utilizando a interface VGA citada anteriormente é que iremos desenvolver este projeto.

2.2 IoT

O IoT, mais conhecido como internet das coisas, apareceu com o surgimento da indústria 4.0, que é a utilização de pequenos equipamentos que se conectem à rede gerando dados com um baixo tráfego de rede e que facilitem a integração do homem com a tecnologia.

A Internet das coisas é a tecnologia em que os objetos, bem como os seres humanos são fornecidos com os identificadores exclusivos e pode transferir dados através de uma grande rede sem sequer exigir interação homem-a-computador ou interação homem-a-homem. No conceito de rede, isto refere-se à rede sempre crescente que tem objetos físicos com um IP, direcionado para conectividade com a Internet, e a comunicação que ocorre entre essas coisas e outros dispositivos de rede e sistemas(JOG, 2015).

Esses dados, por sua vez, são enviados para a nuvem para que possam ser gerados gráficos, estudados posteriormente e até utilizados para tomadas de decisão de forma automática. Podemos considerar desta forma equipamentos de IoT, como por exemplo: celulares, smartwatches, geladeiras, computadores de bordo em carros, TVs, etc.

Esta regra se aplica aos mais diversos equipamentos, desde que tenham acesso à rede de internet e possam gerar algum tipo de dado. Ou seja, na IoT, os objetos inteligentes estão verdadeiramente conectados. Deste modo, os objetos podem prover

serviços tais como quaisquer outros dispositivos na Internet, por exemplo, um objeto inteligente pode ser um servidor Web. Qualquer usuário da Internet, seja humano ou máquina, poderá ter acesso aos recursos dos objetos inteligentes(SANTOS, 2019).

Os tipos de equipamentos que já estão sendo empregados na categoria de Home Care, são sensores de ambiente ou sensores de sinais vitais do paciente, de modo que seja gerado com precisão: alarmes, ativação ou desligamento de outros serviços do ambiente de cuidados do paciente em casa.

Estes serviços são suportados por processos eletrônicos chamados de e-saúde. Uso de dispositivos da Internet das coisas em monitoramento remoto e notificação em casos de emergência. (JOG, 2015).

2.3 RASPBERRY

Apesar de não ser enquadrado especificamente como um microcontrolador(Microcontrolador é um pequeno computador, System-on-a-chip (SoC)1 , em um único circuito integrado, o qual contém um núcleo de processador, memória e periféricos programáveis de entrada e saída), a Raspberry Pi deve ser levado em consideração quando o assunto é automação e monitoramento, pois o potencial do pequeno equipamento para prototipação é rápido e estável(BRAZILIAN JOURNAL OF HEALTH REVIEW, 2020).

A raspberry é um equipamento que faz a integração entre o meio de software e hardware. Possui portas analógicas e digitais para o programador iniciante poder integrar sensores ou diversos outros atuadores que trabalham com comunicação serial.

Para ser capaz de realizar tantas atividades, o Raspberry Pi possui um sistema operacional oficial baseado no Kernel do Linux, cujo o nome é Raspbian, porém existem outros sistemas operacionais que também podem ser instalados: Arco, Ubuntu, RISC OS e o Windows 10 para IoT CORE, sendo este último um sistema projetado para IoT. Algumas das linguagens suportadas são C/C++, Java, Python, dentre outras (MOREIRA, 2019).

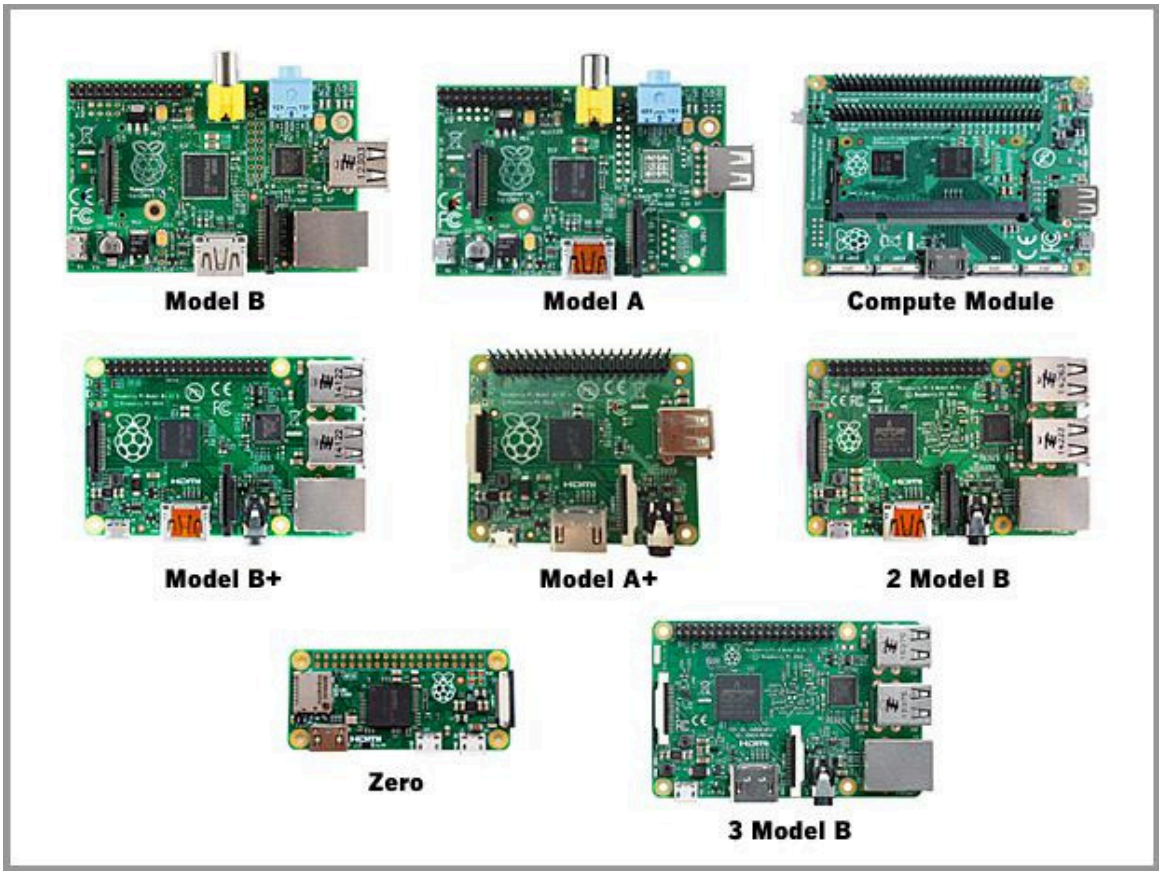
Esse pequeno computador possui portas HDMI e USB para conexão com periféricos normais como mouse, teclados, monitores e etc. Alimentado com um tensão

de 5v e um baixo consumo de corrente(300 mA), a raspberry se torna extremamente versátil para projetos de IoT's de baixo custo. Possui ainda diversos sistemas operacionais que podem ser instalados para facilitar a assimilação de determinados softwares.

De acordo com a figura 2, temos 8 modelos, no entanto, já existem outros como por exemplo a Raspberry Pi 4 que não está presente na imagem e possui uma memória de 4 GB LPDDR4 e um processador Quad Core Cortex A72 de 1.5 Ghz que melhora ainda mais o desempenho dos projetos desenvolvidos com este tipo de plataforma.

A raspberry é utilizada neste projeto como um conceito de integração entre os dois meios, satisfazendo as primeiras solicitações do software. Tratando alguns dados e enviado para o servidor central.

Figura 2 – Modelos de Raspberry Pi.



Fonte: https://www.bernabauer.com/como-identificar-a-versao-do-hardware-do-raspberry-pi/.

2.4 OPENCV E TESSERACT

A biblioteca OpenCV, desenvolvida pela intel em 2000, permite a manipulação de dados de imagens, manipulação de matrizes e vetores, desenvolvimento de rotinas de álgebra linear, estruturas de dados dinâmicas, desenvolvimento de algoritmos de processamento de imagem, análise estrutural, calibração de câmera, análise de movimento(tracking), reconhecimento de objetos, GUI Básica, e rotulagem de imagem. Sua vantagem em relação ao MATLAB é o fato de poder ser usada, para programar em várias plataformas, como C/C++, Python Visual Basic, Ruby, permitindo uma integração mais fácil com outros programas, evitando problemas de integração e facilitando no caso de desenvolvimento de softwares embarcados.(PONCIANO, 2009)

A biblioteca está disponível com o código fonte e os executáveis (binários)otimizados para os processadores Intel. Um programa OpenCV, ao ser executado, invoca automaticamente uma DLL (Dynamic Linked Library) que detecta o tipo de processador e carrega, por sua vez, a DLL otimizada para este. Juntamente com o pacote OpenCV é oferecida a biblioteca IPL (Image Processing Library), da qual a OpenCV depende parcialmente, além de documentação e um conjunto de códigos exemplos(MARENGONI, 2010).

A biblioteca está dividida em cinco grupos de funções: Processamento de imagens, análise estrutural, análise de movimento e rastreamento de objetos, reconhecimento de padrões e calibração de câmera e reconstrução 3D(MARENGONI, 2010).

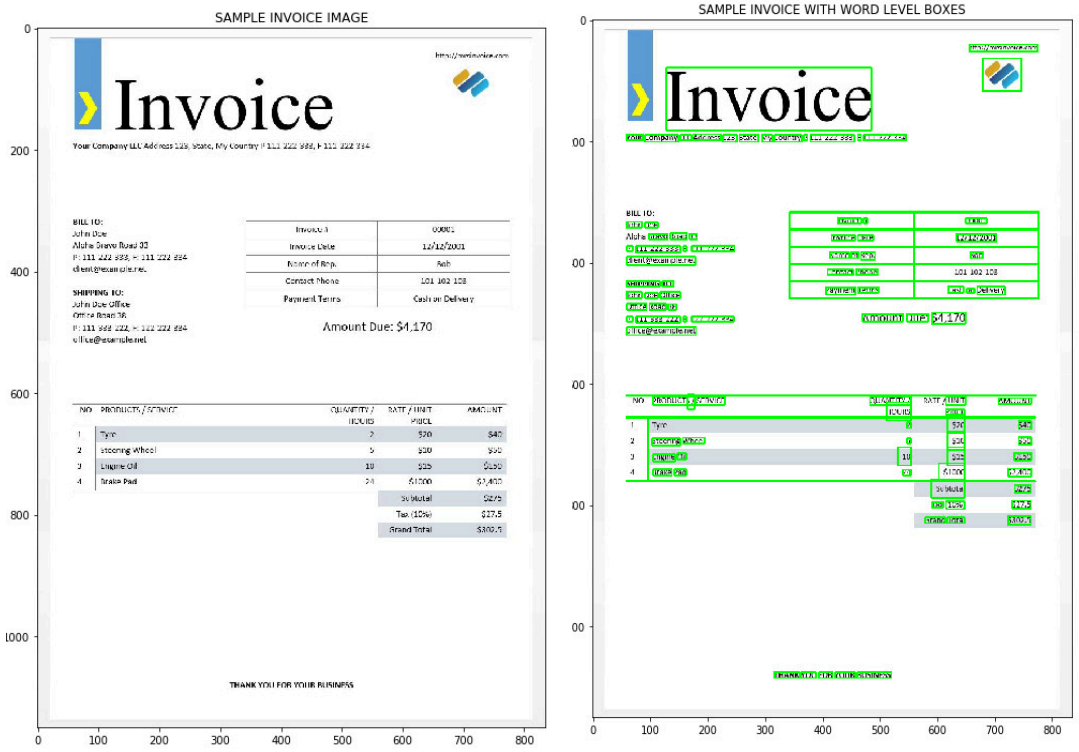
Para captura de um determinado vídeo o OpenCv destrincha o vídeo como sendo a composição de várias imagens(frames) e depois analisa pixel por pixel em formato de tabela da esquerda para direita e de cima para baixo. Se pegarmos um exemplo de uma câmera que registra vídeos a uma taxa de 30 FPS (30 frames por segundo) estaremos dizendo que essa câmera registra “30 fotos por segundo” e realiza a junção delas criando um vídeo único. Após isso a biblioteca de visão computacional irá realizar o

processo inverso, irá desmontar o vídeo para que possa ser analisado cada frame gerando em um segundo e obter os dados necessários.

E então e não menos importante é o “Tesseract” - Reconhecimento de Caracteres Ópticos(OCR) que também é um software livre. Criado inicialmente pela Hewlett-Packard (HP) e foi mantido pelo google para detecção de textos em dispositivos móveis, vídeos e textos de spam no Gmail. Atualmente o software está disponível no repositório do GitHub para os desenvolvedores que desejarem a aplicação.

Aliando o OpenCv e o Tesseract podemos ter a leitura de textos em mais de 100 idiomas em um vídeo corrente. E todo esse conhecimento distribuído no formato de open source facilita ainda mais o desenvolvimento de novas tecnologias no ambiente hospitalar conforme é o intuito deste projeto.

Figura 3 – Reconhecimento de caracteres utilizando o Tesseract em uma imagem.



Fonte: https://github.com/NanoNets/ocr-with-tesseract/blob/master/tesseract-tutorial.ipynb.

Na Figura 3 como um exemplo foi aplicado o software de reconhecimento de caracteres (Tesseract) na imagem e juntamente com o OpenCV esses caracteres foram mapeados em caixas de texto na cor verde.

Esta é uma técnica bastante utilizada pelo programador para identificar possíveis falhas de treinamento na biblioteca, pois, obtendo o que está sendo lido ele pode selecionar partes específicas da imagem e tratar individualmente informações que provavelmente não foram identificadas em um primeiro contato.

2.5 DOCKER

O Docker oferece a capacidade de empacotar e executar um aplicativo em um ambiente vagamente isolado denominado contêiner. O isolamento e a segurança permitem que você execute vários contêineres simultaneamente em um determinado host. Os contêineres são leves e contém tudo o que é necessário para executar o aplicativo, portanto, não é preciso depender do que está instalado atualmente no host. É possível compartilhar contêineres facilmente enquanto são desenvolvidos(DOCKER, 2020).

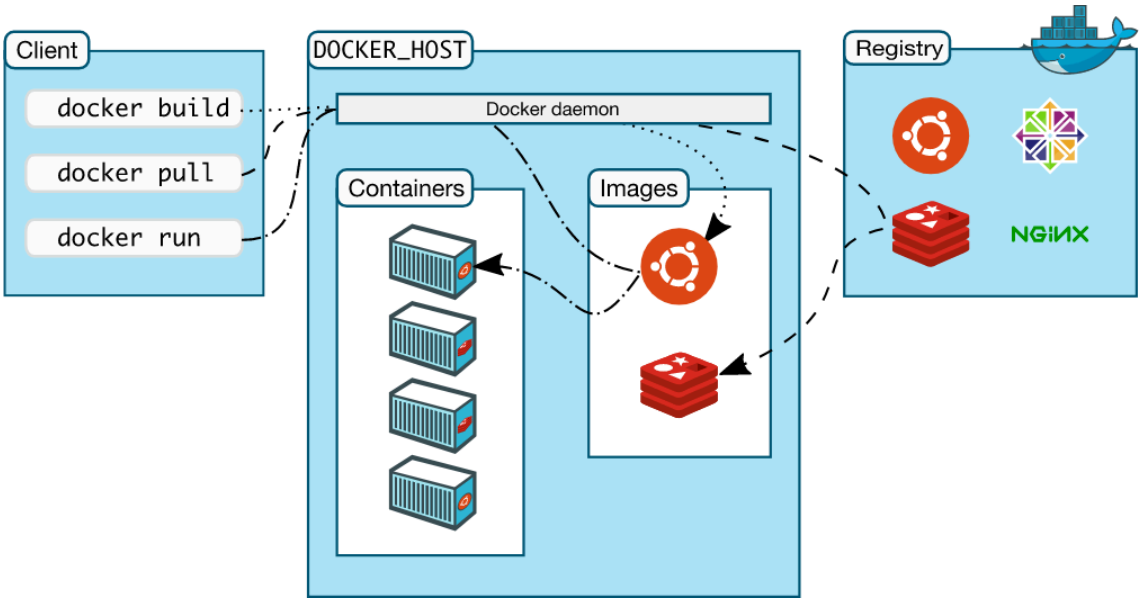
Partindo do ponto de vista estrutural das aplicações criadas atualmente, a implementação de diversos sistemas demanda bastante tempo para instalação e configuração em determinados servidores. Desde a seleção do sistema operacional a ser utilizado, até as bibliotecas necessárias para rodar uma determinada aplicação consomem um determinado tempo do técnico ou analista de implantação de projetos.

Para facilitar e reduzir este consumo de recurso foi criado o “Docker”, desenvolvido pelo grupo Google, porém pela empresa Docker, Inc, em linguagem GO. Ele possui uma estrutura de conteinerização, ou seja, após instalar a aplicação do Docker podemos simplesmente baixar um container que possui todas as configurações e bibliotecas necessárias para rodar por exemplo o OpenCV e o Tesseract, tudo isso com apenas um comando. Esse conceito é chamado “virtualização de ambientes”.

Virtualização é uma combinação de software e hardware visando a criação de máquinas ou ambientes virtuais em uma única máquina física (BURGUER, 2012). Na prática, as máquinas virtuais oferecem os mesmos serviços que uma máquina física, onde um sistema operacional pode ser utilizado com todos os seus recursos. Porém, máquinas ou ambientes virtuais só existem logicamente (ALECRIM, 2012). Com isso, é possível executar diversas máquinas sobre um mesmo hardware, o que aumenta o aproveitamento computacional e, consequentemente, diminui o consumo energético e infraestrutura.(VINCCI, 2017)

Após instalado o container podemos então colocar em funcionamento onde ele terá uma interface virtual de rede e poderá se comunicar através dela com as outras aplicações. Além de poder coletar os logs de erro em tempo real e poder entrar neste Sistema Operacional(SO) emulado, esse tipo de estrutura permite a segmentação das camadas da aplicação facilitando a manutenção da infraestrutura do servidor e sistema em geral.

Figura 4 – Arquitetura docker.



Fonte: DOCKER, 2020.

Docker usa uma arquitetura cliente-servidor. O cliente Docker se comunica com o daemon Docker , que faz o trabalho pesado de construção, execução e distribuição de

seus containers Docker. O cliente Docker e o daemon podem ser executados no mesmo sistema ou você pode conectar um cliente Docker a um daemon remoto do Docker. O cliente Docker e o daemon se comunicam usando uma API REST, por soquetes UNIX ou uma interface de rede. Outro cliente Docker é o Docker Compose, que permite trabalhar com aplicativos que consistem em um conjunto de containers(DOCKER, 2020).

Neste projeto o Docker foi utilizado para padronizar e agilizar os processos de: tratamento do vídeo, armazenamento dos dados e criação dos dashboards. Representando respectivamente os containers de OpenCV e Tesseract, PostgreSQL e Grafana. A adoção desta estrutura facilitou e agilizou o entendimento quanto a replicação do nosso serviço em outros servidores.

2.6 PostgreSQL

O PostgreSQL é um dos bancos de dados relacional mais antigo criado ainda na década 90, no entanto, extremamente poderoso. Por ser antigo e possuir licenciamento livre ele se tornou robusto e amplamente utilizado pela comunidade de TI. O Sistema Gerenciador de Banco de Dados (SGBD) PostgreSQL, já possui uma sólida reputação no mercado devido às constantes atualizações geradas ocasionalmente pelo PostgreSQL Global Development Group.

Atualmente já se encontra na versão 13 atualizado trimestralmente e apoiado por recursos de algumas empresas parceiras e doações o PostgreSQL possui diversos pontos positivos, como por exemplo: o seu acesso é totalmente gratuito, possibilidade de criar extensões, além das principais linguagens de programação possuírem bibliotecas que facilitam a integração de escrita e leitura neste banco de dados.

2.7 Grafana

Chegamos então no Grafana que é uma aplicação de criação de dashboards e acompanhamento de dados em tempo real. Possui a capacidade de integração com diversos banco de dados como: Prometheus, Graphite, OpenTSDB, Elasticsearch,

PostgreSQL, MySQL e também com aplicações em Cloud como Google Cloud Monitoring, CloudWatch e Azure Monitor.

Figura 5 – Exemplo de tela do Grafana com integração com um Zabbix Server.

Fonte: http://zabbixbrasil.org/?p=1674.

Criado a sua primeira versão em 2014 pela Torkel Ödegaard, o grafana inicialmente não tinha suporte para banco de dados relacionais como por exemplo o PostgreSQL, foi criado com o princípio de atender apenas a banco de dados de série temporal, no entanto, logo após o seu lançamento e o aumento da sua popularidade, a empresa decidiu também criar a possibilidade esses outros tipos de bancos de dados.

Possui também plugins que podem ser desenvolvidos e inseridos na aplicação facilitando o desenvolvimento de painéis complexos de acordo com a necessidade de cada negócio.

Por se tratar de uma aplicação Open Source o Grafana possui uma ampla comunidade espalhada pelo mundo que auxilia no desenvolvimento destes plugins e o cumulativo nestes conhecimentos.

Apesar da grande quantidade de recursos disponíveis, a estrutura do Grafana é leve e flexível, já que permite organizar dados vindos de diversas fontes em um mesmo

dashboard. Além disso, a plataforma é de fácil instalação, e todas as integrações com bancos de dados e outras ferramentas são feitas por meio de plugins. Todos os plugins podem ser instalados e gerenciados de forma simples na ferramenta, e há muitas opções disponíveis(OPEN SERVICES, 2019).

2.8 ZeroMQ

Nomeado amigavelmente pela empresa que a desenvolveu como: *Intelligent socket library for messaging*, ele é utilizado para a transferência de mensagens baseado na estrutura de soquetes, no entanto, como é informado anteriormente pela palavra *Intelligent Socket* ele se comporta em um formato uniforme onde pode se absorver diversos tipos de padrões de comunicação como: Request-Reply, Publisher-Subscriber, Push-Pull, entre outros.

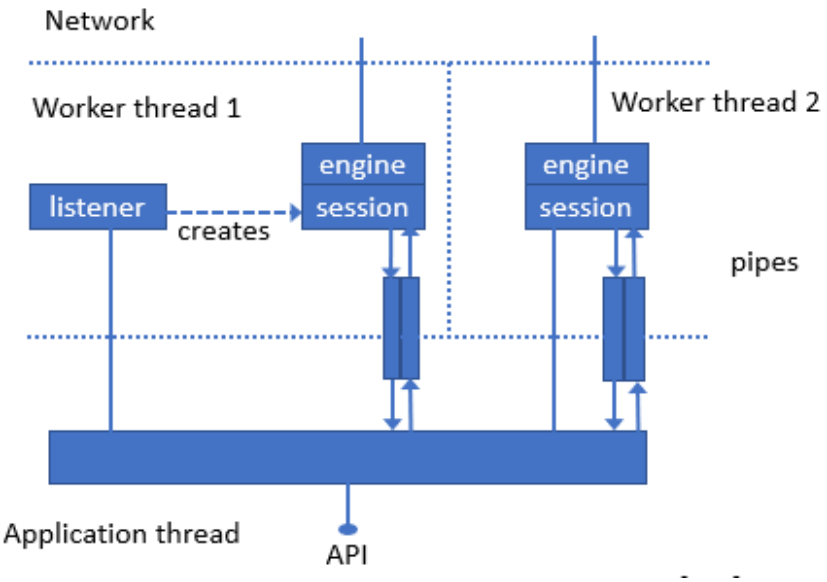
Possui a capacidade de fornecer integração entre entidades com mais de uma linguagem de programação como por exemplo: a máquina 1 pode estar utilizando Python como sua linguagem nativa e pode se comunicar através do ZeroMQ com uma máquina 2 que utiliza C++.

ZeroMQ é uma biblioteca de mensagens de rede assíncrona conhecida por seu alto desempenho. Resumindo, o ZMQ permite enviar mensagens (dados binários, dados serializados, strings simples, etc.) pela rede por meio de vários métodos como TCP ou multicast, bem como entre processos(PALADINO, 2020).

ZeroMQ fornece uma série de APIs de linguagem que rodam na maioria dos sistemas operacionais e permite que você se comunique perfeitamente entre todos os tipos de programas. Ele também fornece uma coleção de padrões, como solicitação-resposta e publicação-assinatura, que ajudam você a criar e estruturar sua rede(PALADINO, 2020).

Atualmente a biblioteca do ZeroMQ tem suporte para mais de 30 linguagens de programação e possui um licenciamento Open Source, fazendo desta ferramenta uma das mais difundidas atualmente quando o assunto se trata de um tráfego rápido de mensagens.

Figura 6 – Arquitetura utilizada do ZeroMQ.



Fonte: https://www.educba.com/zeromq-vs-rabbitmq/.

A sua estrutura é dividida em:

- Listener: Fica escutando as informações recebidas por TCP para que seja gerado um objeto para tratar.
- Session: É o objeto gerado pelo listener que faz conexão com o soquete do ZeroMQ.
- Engine: É a parte do objeto que faz comunicação direta com a rede.
- Pipes: O Pipe lida com a interação interna de mensagens, gerando uma fila de transmissão rápida de mensagens de forma rápida pelas threads.

2.9 MULTITHREAD

Desta forma, as threads podem ser entendidas como fluxos independentes de execução pertencentes a um mesmo processo, que requerem menos recursos de controle por parte do sistema operacional. Assim, as threads são o que consideramos processos

leves (lightweight processes) e constituem uma unidade básica de utilização do processador(JANDL, 2004).

E ainda segundo Jandl, Sistemas computacionais que oferecem suporte para as threads são usualmente conhecidos como sistemas multithreading. Em sistemas não dotados de suporte a threads nativamente, o uso de bibliotecas de extensão permite a utilização de pseudo-threads. Exemplos de bibliotecas de suporte threads de usuário são o PThreads do POSIX ou o C-threads do sistema operacional Mach.

O conceito de Multithread é bastante necessário para entendermos a capacidade de processamento e transmissão dos pacotes de vídeo neste projeto. Basicamente, a execução do multithread é onde o sistema operacional possui a habilidade de executar diversas solicitações de aplicações simultaneamente, ao contrário, de um sistema que não possui esta competência ele ficaria travado na execução da thread principal.

2.10 IMAGEZMQ

Assim como os demais, o ImageZMQ é uma biblioteca que é utilizada para transmissão de imagens ou vídeos, baseado no ZeroMQ ou ZMQ falado anteriormente, ele possui uma característica importantíssima que é alta disponibilidade para transmissão de dados.

Para realização da transmissão é necessário uma implementação curta com poucas linhas de código tanto no emissor quanto no receptor e tudo isso baseado em Python onde é derivado da biblioteca do OpenCV.

Através deste tipo de comunicação é possível enviar vídeos ou imagens compactadas utilizando o protocolo ZMQ para aliviar e reduzir o tráfego na rede, além de permitir que o servidor central que recebe as imagens consiga tratar de forma simultânea vários remetentes ao mesmo tempo.

Em resumo, o ImageZMQ é um mecanismo de transporte de imagem fácil de usar para uma rede de processamento de imagem distribuída. Por exemplo, uma rede de uma dúzia de Raspberry Pis com câmeras podem enviar imagens para um computador central mais poderoso. A Raspberry Pi realiza captura de imagem e processamento de

imagem simples, como inversão, desfoque e detecção de movimento. Em seguida, as imagens são passadas via imagezmq para o computador central para processamento de imagem mais complexo, como marcação de imagem, extração de texto, reconhecimento de recursos, etc(BASS, 2020).

Por utilizar o OpenCV como meio principal do tratamento para imagens o ImageZMQ se torna extremamente versátil para utilização devido a sua ampla documentação gerada por sua comunidade Open Source.

Aliando o ZMQ, o Image ZMQ e o OpenCV é possível conseguir um alto desempenho que é um fator crucial para aplicações que tratam da saúde humana ou qualquer outro tipo de dado crítico que não pode sofrer congestionamento de tráfego na rede e é necessário uma baixa latência na transmissão.

Figura 7 – Exemplo de Código para o servidor principal.

```
# execute este programa no Mac para exibir fluxos de imagem de vários RPis
import cv2
import imagezmq
image_hub = imagezmq . ImageHub ()
enquanto True : # mostra imagens transmitidas até Ctrl-C
    rpi_name , image = image_hub . recv_image ()
    cv2 . imshow ( rpi_name , image ) janela # 1 para cada RPi
    cv2 . waitKey( 1 )
    image_hub . send_reply ( b 'OK' )
```

Fonte: (BASS, 2020).

Figura 8 – Exemplo de Código utilizado em um raspberry.

```
# execute este programa em cada RPi para enviar um fluxo de imagem rotulado
import socket
import time
from imutils.video import VideoStream
import imagezmq

sender = imagezmq . ImageSender ( connect_to = 'tcp: // jeff-macbook: 5555' )

rpi_name = socket . gethostname () # enviar RPi hostname com cada imagem
picam = VideoStream (usePiCamera = True ) . start ()
vezes . sleep ( 2.0 ) # permite que o sensor da câmera aqueça
enquanto True : # envia imagens como stream até Ctrl-C
    image = picam . leia ()
    remetente . send_image ( rpi_name , imagem )
```

Fonte: (BASS, 2020).

Como forma de exemplo prático, o seu código modelo é extremamente enxuto para o seu receptor(servidor central) e o emissor(raspberry pi) conforme são mostrados a seguir nas figuras 7 e 8.

3 DESENVOLVIMENTO

3.1 SOLUÇÃO PROPOSTA

Esse projeto consiste no desenvolvimento de uma plataforma de análise e acompanhamento de leitos de Unidades de Tratamento Intensiva (UTI) em tempo real, para monitorar individualmente os parâmetros e registrar os dados de um paciente de forma segura e rápida.

Imaginando um contexto onde os leitos são isolados por quarto, o hospital possui vários andares e a equipe de enfermeiros e médicos estão distribuídos por andares. Se houver a redução do corpo de profissionais em alguns desses andares ou até mesmo por imprudência da equipe, os pacientes podem estar correndo um grave risco de morte em casos da falta de uma prestação de socorro rápida.

Para isso, o projeto visa aumentar a eficiência na identificação de algum problema de saúde dos pacientes utilizando o monitoramento em tempo real, por meio da flexibilidade de notificar os profissionais de saúde independente do local que eles estejam, sendo necessário apenas ter um supervisório que valide e alarme essas informações.

Já existem soluções que são de propriedade intelectual de algumas empresas mas que os seus dashboards centralizados exibem apenas equipamentos da sua própria marca, o que não é a realidade da maioria dos hospitais públicos que não possuem um sistema Philips PIIC IX(produto Philips para essa solução), por exemplo. Pois, grande parte dos equipamentos adquiridos são concebidos por meio de licitações onde os valores propostos variam de modelo e de fabricante, inviabilizando as soluções concorrentes.

Para tal foram implementados tanto hardware quanto software neste projeto a ponto de integrar as diversas possibilidades apresentadas atualmente nas UTIs e em outros tipos de leitos, ou seja, caso o equipamento possua alguma interface de vídeo que possa ser transmitida que é o caso dos monitores cardíacos que possuem uma saída

VGA ou HDMI é possível interpretar esses dados, transmitir, armazenar e criar dashboards customizáveis.

Devido a restrição de prazo de 4 meses para implantação e desenvolvimento do projeto foi necessário a divisão em duas partes: 1 - Transmissão dos dados com uma alta disponibilidade e 2 - Tratamento do vídeo, Armazenamento dos dados e Implantação do Dashboard. Este documento trata da parte “2” onde é será exposto a seguir os tópicos para resolução do problema. Para um entendimento mais visual, favor consultar a figura 13 onde é melhor exposto a divisão das atividades. A seguir mostraremos os componentes de forma individual para que seja entendido em sua totalidade a construção deste projeto.

3.2 DOCKER

Por se tratar de um recurso de infraestrutura e a sua aptidão para escalabilidade, o docker se tornou um dos principais aliados para este projeto a ponto de desprezar o tempo de configuração e download de diversas bibliotecas para o correto funcionamento da aplicação. Inicialmente foram realizados os downloads de diversos containers e integrados entre eles por interfaces virtuais de rede disponíveis nas máquinas locais.

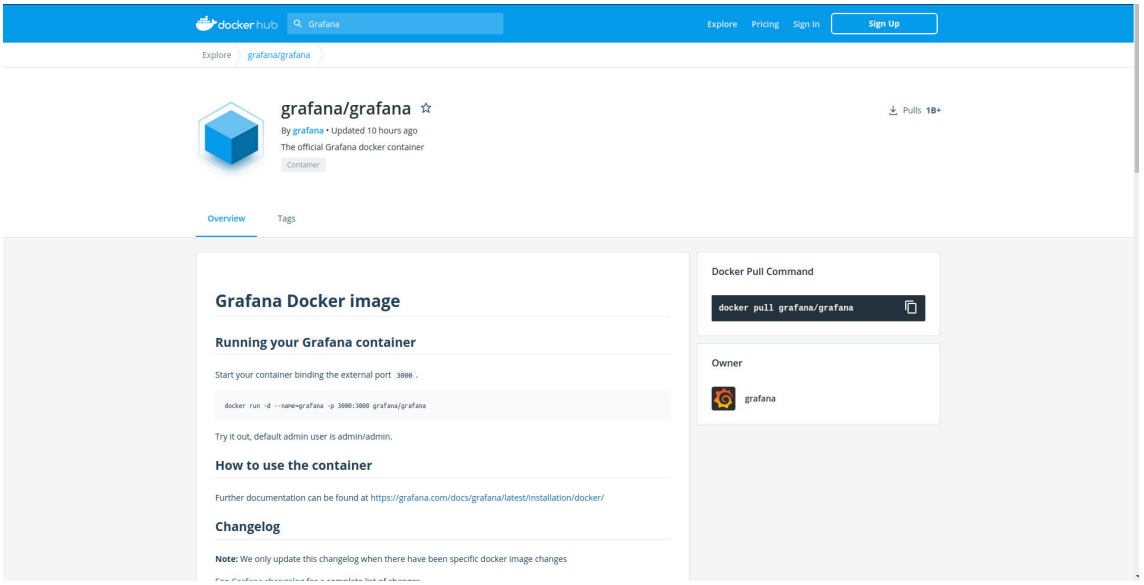
Containers utilizados:

- Python 3 e Tesseract;
- PostgreSQL - versão 9.6.19;
- Grafana - versão 7.1.5;

Todos esses containers estão disponibilizados gratuitamente com todas as informações necessárias para configuração no https://hub.docker.com/, conforme exibido na figura 9.

Caso não seja encontrado um container com as versões ou as aplicações necessárias é possível desenvolver um novo container e subir uma versão no Docker Hub que é o repositório desses dados para que fique acessível para toda a comunidade.

Figura 9 – Repositório do Docker.



Fonte: https://hub.docker.com/.

Para configuração do container do postgres por exemplo, foi possível ser configurado apenas realizando este comando:

```
docker run -d --name postgres \
    -p 5432:5432 \
    -e POSTGRES_PASSWORD=postgres \
    -v /var/lib/postgresql/data:/var/lib/postgresql/data \
    -d postgres
```

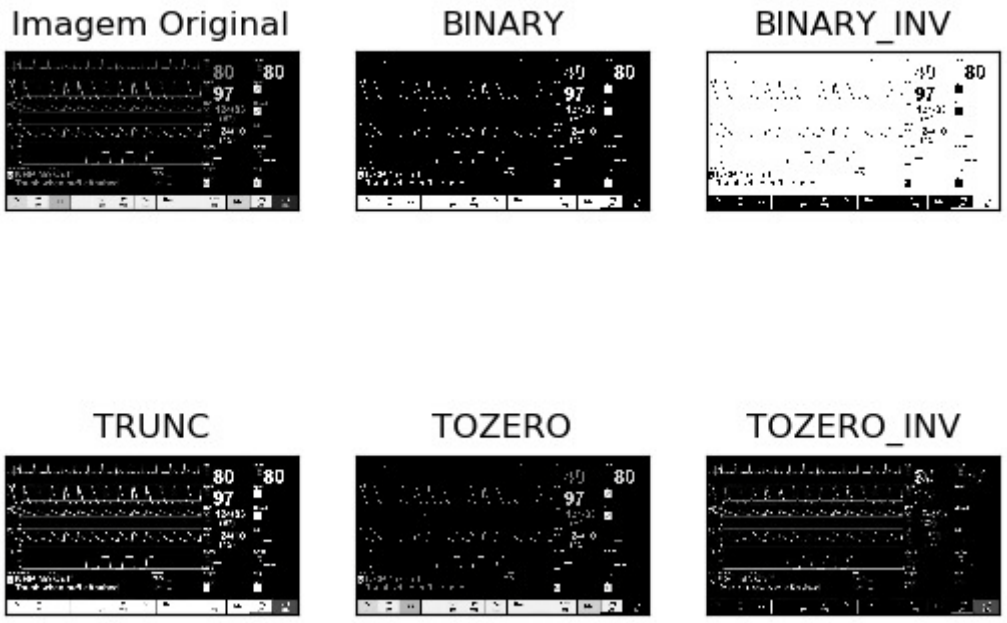
Os demais containers seguem nesta mesma linha de raciocínio facilitando a reprodução do sistema e a agilidade na implantação.

3.3 OpenCV e Tesseract

O OpenCV e o Tesseract trabalham de forma conjunta no mesmo código extraindo informações dos dados conforme estruturado. O Tesseract é uma biblioteca que pode ser baixada e já utilizada para extração dos dados, no entanto, ela possui algumas deficiências no tratamento de todos tipos de textos. Para evitar o treinamento

da biblioteca com alguns formatos de vídeos diferentes (provenientes dos monitores cardíacos, por exemplo) a solução foi realizar o tratamento da imagem em formato de frames e descaracterizar as cores de alguns pixels para que a correta leitura dos textos fosse possível.

Figura 10 – Descaracterização da imagem.



Fonte: o próprio autor.

Conforme a figura 10, foi necessário primeiramente converter o frame para o tom de cinza e após isso realizar a tratativa dos limiares de cores. Utilizando a função cv2.threshold nativa do OpenCV, é possível estabelecer diferentes tons de branco ou preto para captar na imagem, ou seja, podemos variar a cor do pixel de 0 a 255. Estabelecendo o máximo e o mínimo podemos ignorar ou visualizar determinadas informações ao utilizar esta função. Além disso neste projeto foi necessário utilizar variações do threshold que são:

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV
- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO

- cv2.THRESH_TOZERO_INV

Estas funções possuem parâmetros já predeterminados para tratamento da imagem conforme pode ser visto figura 10 ele omite ou destaca alguns pontos da imagem conforme a cor presente no pixel.

Além de definir os extremos dos pixels podemos também definir a posição a ser tratada, logo, após definirmos todos esses pontos que são: o tipo de thresh, os limiares e a posição da imagem desejada nós enviamos aquela nova imagem para ser tratada pelo Tesseract para que ele possa realizar a extração do texto e então atribuir a uma variável.

Na figura 11 e 12 podemos ver o código onde é feito este tratamento individual das cores e a definição do local da imagem onde é disponibilizado cada um dos dados dos sensores respectivamente.

Figura 11 – Tratamento das cores.

```
27          ret,thresh1 = cv2.threshold(frame, limiar, maximo, cv2.THRESH_BINARY)
28          ret,thresh2 = cv2.threshold(frame, limiar, maximo, cv2.THRESH_BINARY_INV)
29          ret,thresh3 = cv2.threshold(frame, limiar, maximo, cv2.THRESH_TRUNC)
30          ret,thresh4 = cv2.threshold(frame, limiar, maximo, cv2.THRESH_TOZERO)
31          ret,thresh5 = cv2.threshold(frame, limiar, maximo, cv2.THRESH_TOZERO_INV)
```

Fonte: o próprio autor.

Figura 12 – Captura dos dados determinando o quadro ideal na imagem.

```
33      freq_cardiaca = pytesseract.image_to_string(gray[25:57,350:400], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789/');
34      pulso = pytesseract.image_to_string(thresh2[20:60,440:480], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789/');
35      oximetria = pytesseract.image_to_string(thresh2[60:95,355:400], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789/');
36      pressao_invasiva = pytesseract.image_to_string(thresh2[100:118,357:411], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789/');
37      media_pressao_invasiva = pytesseract.image_to_string(thresh2[117:135,370:387], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789');
38      pressao_arterial = pytesseract.image_to_string(thresh2[140:158,361:411], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789/');
39      media_pressao_arterial = pytesseract.image_to_string(thresh2[155:173,370:388], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789/');
40      pressao_parcial_co2 = pytesseract.image_to_string(thresh2[183:206,353:383], config='--psm 10 --oem 3 -c tessedit_char_whitelist=0123456789/');
41
```

Fonte: o próprio autor.

3.4 POSTGRESQL

Para integração do PostgreSQL ao código principal desenvolvido em Python, não houve plena dificuldade, pois existem diversas bibliotecas que fazem integração com os mais variados bancos de dados.

O PostgreSQL oferece os mesmo comandos baseados em SQL como: SELECT, CREATE, DROP, etc, que facilitam ainda mais a sua reprodução e o seu desenvolvimento.

Foi realizada a criação de apenas uma tabela no banco. Quando se trata apenas da interpretação dos dados dos equipamentos provenientes da UTI poderemos absorver essas informações e inseri-las em uma única linha da tabela, para que seja realizado um insert único.

Uma tabela única para armazenamento dos dados foi criada utilizando o seguinte comando:

```
CREATE TABLE dados_uti( freq_cardiaca INT, pulsacao INT, oximetria INT, pressao_invasiva
VARCHAR, media_pressao_invasiva INT, pressao_arterial VARCHAR,
media_pressao_arterial INT, pressao_parcial_co2 INT, date TIMESTAMP NULL DEFAULT
CURRENT_TIMESTAMP);
```

Através deste comando é possível informar que as colunas criadas irão receber apenas valores inteiros em sua totalidade e apenas a coluna do *date* irá ser setado automaticamente pelo banco de dados ao receber um insert através da aplicação criada.

3.5 GRAFANA

No modelo de dashboard desenvolvido no grafana são informados os seguintes pontos: frequência cardíaca, pulsação, oximetria, média da pressão invasiva e a média da pressão arterial pulmonar.

Não foram selecionados todos os valores que um monitor cardíaco pode emitir, pois durante o projeto não foi possível obter alguns desses equipamentos para filtrar a sua imagem bem como os sensores e a sua utilização.

A imagem final do dashboard encontra-se posteriormente na figura 13.

4 RESULTADOS

Todos os resultados foram obtidos por meio de cenários simulados, onde era possível obter exatamente os mesmos dados e não era necessário uma intervenção ou disponibilização de equipamentos hospitalares.

4.1 METODOLOGIA

Devido ao contexto da pandemia não foi possível realizar os testes em ambientes hospitalares garantindo a reprodução do cenário ideal.

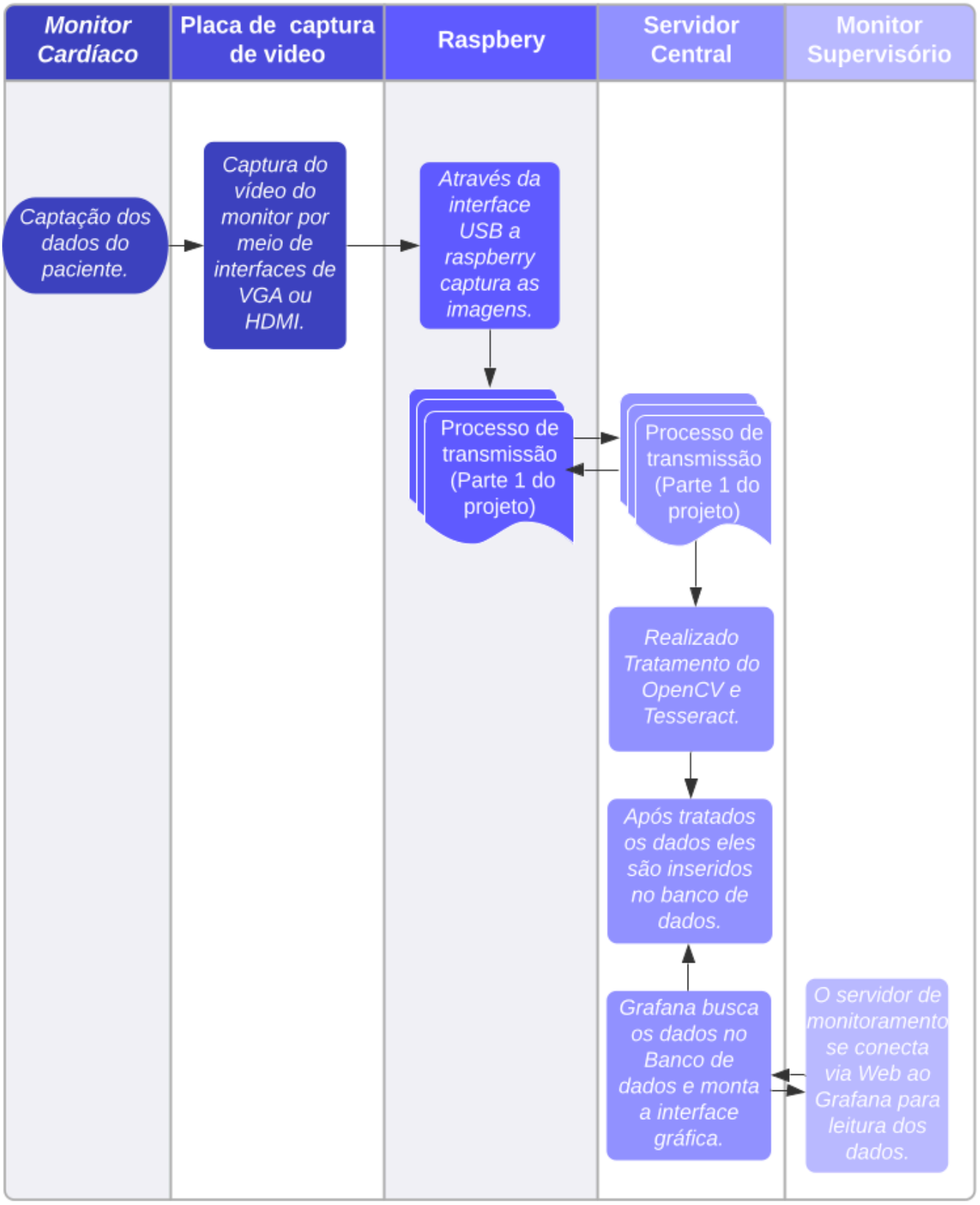
Todavia, a metodologia utilizada foi no modo: **pesquisa aplicada**. Ou seja, este projeto visa gerar artefatos para a construção e o incentivo do desenvolvimento de novas tecnologias aplicadas ao setor hospitalar utilizando *big data* e o processamento de imagens.

Foi reproduzido em um ambiente controlado e simulado por meio de máquinas virtuais a fim de gerar dados idênticos aos criados em um ambiente de produção. Simulando desta forma os dados exibidos em um monitor cardíaco e transmitidos para processamento por meio de uma raspberry.

4.1.1 CENÁRIO E EXECUÇÃO DOS TESTES

O cenário utilizado para realizar os testes estava de acordo com a figura 13, no entanto, o Monitor cardíaco foi substituído por um notebook onde foi possível simular a tela de um equipamento em pleno funcionamento. Uma placa de captura de vídeo foi utilizada para todos os dados mostrados na interface VGA do notebook fossem mostradas exatamentes iguais com uma qualidade de até 1080p (o “p” se refere ao conceito de “*progressive scan*”, ou seja, todas as 1080 linhas são atualizadas simultaneamente tornando a sua reprodução mais suave) e variações de 25, 29.95, 30, 50, 59.94 e 60 frames por segundo.

Figura 13 – Cenário utilizado para testes.



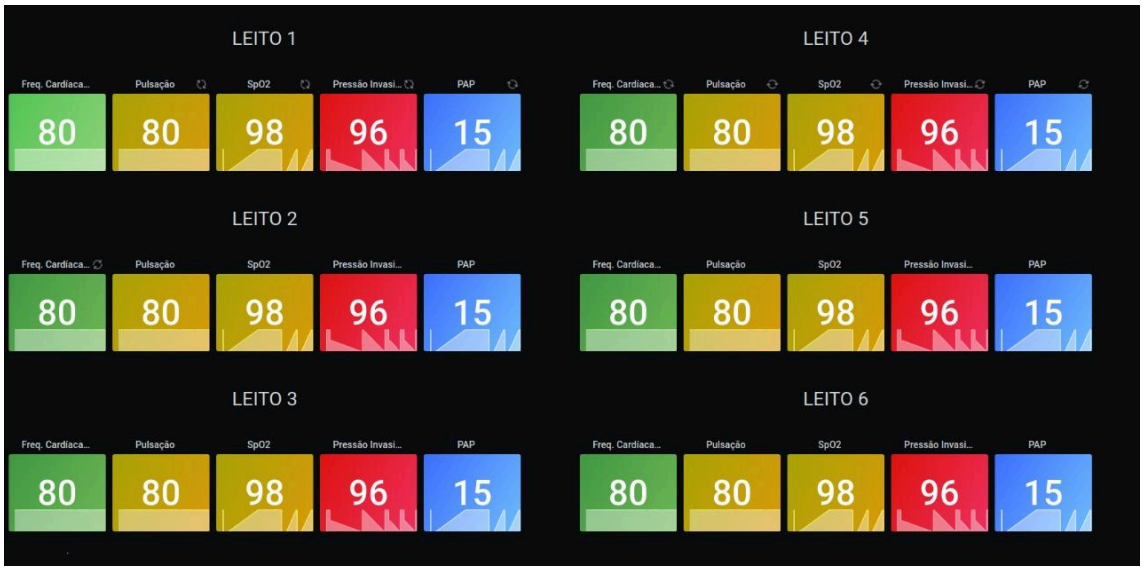
Fonte: o próprio autor.

Utilizamos um vídeo de demonstração do equipamento que variava diversos parâmetros durante a sua execução, garantindo desta forma a qualidade dos testes e o registro correto das informações. Além disso todos os dados são tratados por frames e inseridos no banco de dados a uma taxa de 0,03 segundos por inserção considerando 30 fps e o grafana realizando a atualização da tela a cada dois segundos tornando a variação de valor do equipamento para com o servidor supervisório perceptível.

4.2 RESULTADOS OBTIDOS

Como resultados do fluxo apresentado na figura 14, o grafana conseguiu renderizar os valores apresentados a partir de um vídeo reproduzido em um notebook como alternativa a um monitor cardíaco. Segue abaixo a imagem do nosso dashboard constituindo as informações de modo centralizado e encerrando o ciclo do projeto de transmissão, armazenamento e exibição dos dados.

Figura 14 – Modelo final de Dashboard.



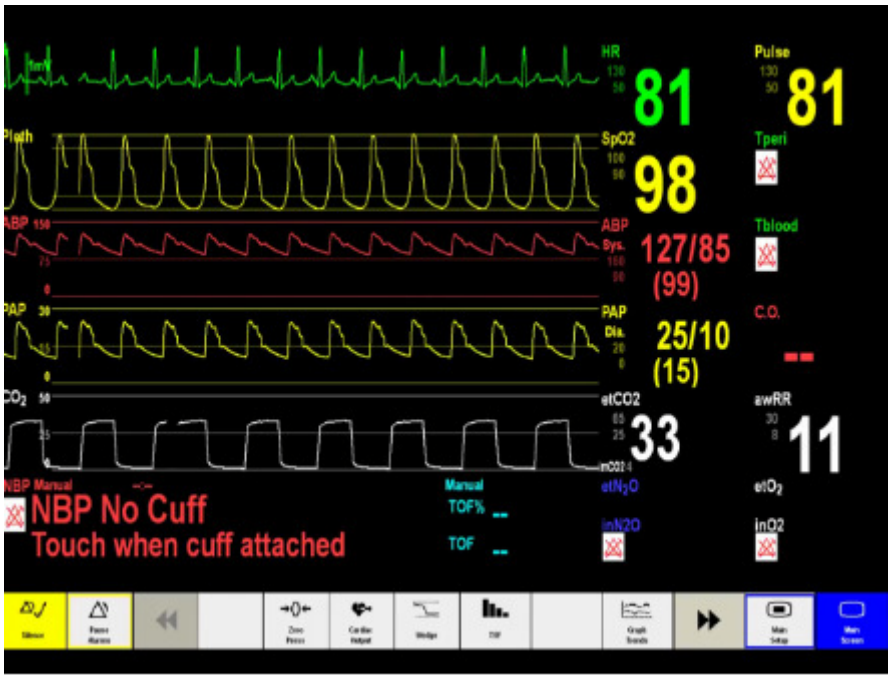
Fonte: o próprio autor.

Neste modelo de tela são informados os seguintes pontos: frequência cardíaca, pulsação, oximetria, média da pressão invasiva e a média da pressão arterial pulmonar.

Tais dados são extremamente importantes para uma análise completa do estado atual do paciente de forma local ou remota.

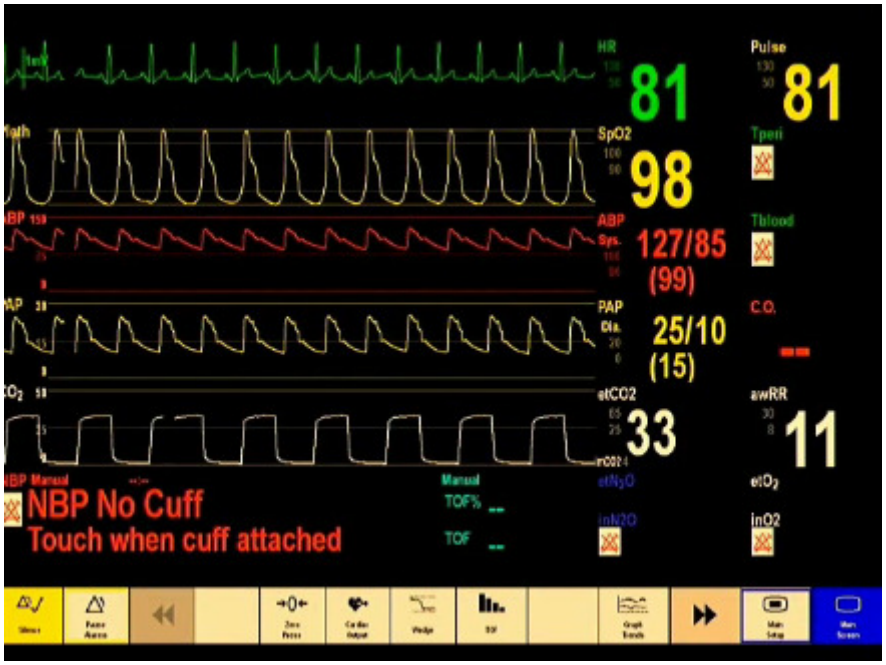
O frame que foi utilizado para simular o monitor cardíaco e o recebido pelo servidor principal após a transferência pela raspberry pi, são exibidos nas figuras 15 e 16. Para análise podemos ver uma leve variação das cores gerado pelo ruído da transmissão e pelo tratamento das imagens por meio do opencv. Os dados sobre o processo utilizado para a transferência destes frames são exibidos na parte 1 deste projeto disponibilizado pelo graduando Sérgio Madruga Sales Filho.

Figura 15 - Frame enviado através da máquina onde foi simulado o monitor cardíaco.



Fonte: o próprio autor.

Figura 16 - Frame recebido pelo servidor para retirada dos dados desejados.



Fonte: o próprio autor.

5 CONSIDERAÇÕES FINAIS

Após diversos testes realizados em máquinas virtuais e em máquinas presentes em nuvem para fidelizar a qualidade do projeto, consideramos viável a criação e produção deste produto como um item que sucederia a melhora do atendimento hospitalar em meio aos problemas críticos de saúde presentes no mundo atual.

Para um ambiente de produção de software seria necessário a alteração em alguns requisitos deste projeto como alteração do produtor de dashboard que possui uma taxa de atualização a cada 2 segundos. A alteração do modelo da raspberry PI 3 para uma do modelo PI 4 e a migração dos serviços de tratamento de imagem que sairia do servidor central e seria realizado pelo *endpoint*, ou seja, seria feito na própria raspberry.

Sendo realizado tais ajustes, este projeto possuiria um excelente custo benefício e atenderia a proposta inicial da ideia, facilitando e adaptando os hospitais a esse novo cenário de tecnologia inclusiva e centralizadora de dados poderíamos tornar este produto um ponto de integração entre hospitais com baixa renda, tecnologia do mercado de biomédica e a utilização de big data.

Referências

THOMAS, R. **Access to health information and support** : a public highway or a private road? Journal of the American Medical Association, v. 280, n. 15, p. 1371-1374, 1998.

MARIA, Z. M. **Tecnologia em Saúde:** da abordagem teórica à construção e aplicação no cenário do cuidado. 1ª edição. Fortaleza: Ed. UECE, 2016. http://www.uece.br/eduece/dmdocuments/Ebook%20-%20Tecnologia%20em%20Saude%20-%20EBOOK.pdf. Acesso em: 16 Set. de 2020.

BRITO, A. **Processamento digital de imagens utilizando o OpenCV**, 2015. https://agostinhobritojr.github.io/tutorial/pdi/. Acesso em: 24 ago. de 2020.

OMS, **Big data in global health: improving health in low- and middle-income countries**, 2015. https://www.scielosp.org/article/bwho/2015.v93n3/203-208/ . Acesso em: 02/12/2020.

OLIVEIRA, **Controle de custos assistenciais na saúde suplementar utilizando big data e analytics para prever comportamentos e antecipar cuidados aos beneficiários**, 2020. https://revistas.face.ufmg.br/index.php/rahis/article/view/6266/3238 . Acesso em 02 /12/2020.

MANCINI, L. **COVID-19 e o impacto no sistema de saúde no Brasil**, 2020. https://bunzlsaude.com.br/blog/dicas-curiosidades/covid-19-e-o-impacto-no-sistema-de-saude-do-brasil/ . Acesso em 03/12/2020.

SARAIVA CAS. **Fatores físicos-ambientais e organizacionais em uma unidade de terapia intensiva neonatal: implicações para a saúde do recém-nascido.** – Trabalho de conclusão do curso de mestrado profissionalizante em Engenharia. Universidade Federal do Rio Grande do Sul. Porto Alegre, 2004.

CAMPOS, M. **Análise do planejamento das unidades intensivas**, 2020.

https://interfisio.com.br/analise-do-planejamento-das-unidades-de-terapia-intensiva/ . Acesso 03/12/2020.

RODRIGUES, E. **Implantação de monitor multiparamétrico com acesso remoto em leito de UTI para cirurgia cardíaca**, 2019. http://tede.bc.uepb.edu.br/jspui/bitstream/tede/3628/2/PDF%20-%20Eder%20Rodrigues%20Ara%C3%BAjo.pdf . Acesso em 03/12/2020.

JOG, Y. et al. **Internet of Things as a Solution enabler in health sector**. International Journal of Bio-Science and Bio-Technology, v. 7, n. 2, p. 9-24, 2015.

MOREIRA, C. **Sistema de controle de apresentação por meio de raspberry pi**, 2019. https://repositorio.ufu.br/bitstream/123456789/26247/4/SistemaControleApresenta%C3%A7%C3%A3o.pdf . Acesso 03/12/2020.

PONCIANO, D. **Estudo da Biblioteca OpenCV**, 2009. http://monografias.poli.ufrj.br/monografias/monopoli10001999.pdf . Acesso 03/12/2020.

MARENGONI, M. **Introdução à visão computacional usando OpenCV**, 2010. https://www.researchgate.net/publication/41799461_Tutorial_Introducao_a_Visao_Computacional_usando_OpenCV . Acesso em 04/12/2020.

ALECRIM, E. **O que é virtualização e para que serve,** 2012. http://www.infowester.com/virtualizacao.php . Acesso em: 04/12/2020.

BURGER, Thomas. **The Advantages of Using Virtualization Technology in the Enterprise**. https://software.intel.com/en-us/articles/the-advantages-of-using-virtualization-technology-in-the-enterprise . Acesso em: 04/12/2020.

REVISTA VINCCI. **Virtualização e Docker**, 2017. http://revistavincci.satc.edu.br/ojs/index.php/Revista-Vincci/article/view/89/51 . Acesso em 04/12/2020.

SANTOS, B. P. **Internet das Coisas: da Teoria à Prática**, 2019. https://homepages.dcc.ufmg.br/~mmvieira/cc/papers/internet-das-coisas.pdf . Acesso em 05/12/2020.

BRAZILIAN JOURNAL OF HEALTH REVIEW. **Sistema Embarcado para transferência de sinais vitais usando o padrão**, 2020.

https://www.brazilianjournals.com/index.php/BJHR/article/viewFile/6550/5777 . Acesso em 05/12/2020.

DOCS DOCKER. **Docker Overview**, 2020. https://docs.docker.com/ . Acesso em 05/12/2020.

OPEN SERVICES. **O que é e como funciona o Grafana, entenda aqui!** https://www.opservices.com.br/grafana/ . Acesso em 05/12/2020.

PALADINO, V. **A brief introduction to ZeroMQ**, 2020. https://intelligentproduct.solutions/technical-software/introduction-to-zeromq/ . Acesso em 05/12/2020.

BASS, J. **Transporting OpenCV images via ZMQ**, 2020. https://pypi.org/project/imagezmq/ . Acesso em 05/12/2020.

JANDL, P. **Notas sobre sistemas operacionais**, 2004. https://docente.ifrn.edu.br/rodrigotertulino/livros/notas-sobre-sistemas-operacionais . Acesso em 06/12/2020.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus João Pessoa - Código INEP: 25096850
Av. Primeiro de Maio, 720, Jaguaribe, CEP 58015-435, João Pessoa (PB)
CNPJ: 10.783.898/0002-56 - Telefone: (83) 3612.1200

Documento Digitalizado Ostensivo (Público)

TCC c/ Ficha

Assunto:	TCC c/ Ficha
Assinado por:	Eron Vieira
Tipo do Documento:	Relatório
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Eron Almeida Vieira da Silva**, **ALUNO (20142610326) DE BACHARELADO EM ENGENHARIA ELÉTRICA - JOÃO PESSOA**, em 29/04/2025 10:15:47.

Este documento foi armazenado no SUAP em 29/04/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1474433
Código de Autenticação: b52ca06d25