FACULDADE DE TECNOLOGIA DE SÃO BERNARDO DO CAMPO
“ADIB MOISÉS DIB”

ANDRÉ DE PAULA CAMPOS
KAUÊ COLOMBO DE OLIVEIRA
MARCELO SALES DE ABREU
VÍTOR EDUARDO GUEDES DE OLIVEIRA

APLICAÇÃO DE ANÁLISE DE DADOS EM MÁQUINAS INDUSTRIAIS COM EIXOS ROTATIVOS

São Bernardo do Campo – SP
Dezembro/2019

ANDRÉ DE PAULA CAMPOS
KAUÊ COLOMBO DE OLIVEIRA
MARCELO SALES DE ABREU
VÍTOR EDUARDO GUEDES DE OLIVEIRA

APLICAÇÃO DE ANÁLISE DE DADOS EM MÁQUINAS INDUSTRIAIS COM EIXOS ROTATIVOS

Trabalho de conclusão de curso apresentado à faculdade de tecnologia de São Bernardo do Campo “Adib Moisés Dib” como requisito parcial para a obtenção do título de tecnólogo em automação industrial.

Orientador: Prof. Me. Marcelo George Griese

São Bernardo do Campo – SP
Dezembro/2019

ANDRÉ DE PAULA CAMPOS
KAUÊ COLOMBO DE OLIVEIRA
MARCELO SALES DE ABREU
VÍTOR EDUARDO GUEDES DE OLIVEIRA

APLICAÇÃO DE ANÁLISE DE DADOS EM MÁQUINAS INDUSTRIAIS COM EIXOS ROTATIVOS

Trabalho de conclusão de curso apresentado à faculdade de tecnologia de São Bernardo do Campo “Adib Moisés Dib” como requisito parcial para a obtenção do título de tecnólogo em automação industrial.

Trabalho de Conclusão de Curso apresentado e aprovado em:_____/_____/2019

Banca Examinadora:

_____

Prof. Me. Marcelo George Griese, FATEC SBC – Orientador

_____

Prof. FATEC SBC – avaliador

_____

Prof. FATEC SBC – avaliador

Dedicamos esse trabalho de conclusão de curso aos nossos familiares, aos professores pelos ensinamentos, ao corpo administrativo pelo suporte dado durante todo o curso e aos colegas que nos acompanham desde o início do curso.

RESUMO

Este trabalho tem como objetivo efetuar a coleta e análise de informações e tomar decisões baseadas em um algoritmo de aprendizado de máquina, visando auxiliar na manutenção preditiva de equipamentos, processando os dados obtidos pelos sensores e comparando-os com os considerados ideais. Para testar os algoritmos de aprendizado foi construído uma giga de testes, considerando um motor trifásico com inversor de frequência, mancal para sustentação do eixo e uma polia, além do sistema Android. O protótipo desenvolvido simula uma máquina industrial com eixo rotativo, e faz a obtenção dos dados necessários através de sensores de vibração e corrente instalados em locais estratégicos, de modo a captar os sinais de campo durante o funcionamento da máquina. Estes sinais são enviados para o Arduino, que por sua vez, se comunica com o sistema Android, via Bluetooth, onde são armazenados em um banco de dados SQLite, para uso do algoritmo. Através do código, os dados obtidos são processados e são estabelecidos valores considerados ideais para cada variável. A partir disso, todos os valores medidos são comparados com o valor padrão e se for detectada alguma anomalia no sistema o manutentor é avisado através do aplicativo em seu celular. Desta forma, pode-se tomar ações estratégicas visando a manutenção ou troca dos componentes com tendência a quebra, consequentemente aumentando a disponibilidade da máquina e reduzindo os custos com manutenções corretivas. Os testes realizados comprovam o objetivo proposto, comparando as reações do sistema mediante situações ideais e anormais.

Palavras-chave: Análise de dados. Aprendizado de máquina. Indústria 4.0. Arduino. Manutenção preditiva.

ABSTRACT

This work aims to perform the collection and analysis of information and make decisions based on a machine learning algorithm, aiming to assist in predictive maintenance of equipment, processing the data obtained by the sensors and comparing them considered ideal. To test the learning algorithms, a test box was built, considering a three-phase motor with frequency inverter, shaft support bearing and a pulley, in addition to the Android system. The developed prototype simulates an industrial machine with rotary axis, and obtains the necessary data through vibration and current sensors installed in strategic places, in order to capture the field signals during the machine operation. These signals are sent to Arduino, which in turn communicates with the Android system via Bluetooth, where they are stored in a SQLite database for use by the algorithm. Through the code, the obtained data are processed and values considered ideal for each variable are established. From this, all measured values are compared with the default value and if any anomaly is detected in the system the maintainer is notified through the application on your mobile phone. In this way, strategic actions can be taken aiming at the maintenance or replacement of components with tendency to break, consequently increasing the availability of the machine and reducing the costs with corrective maintenance. The tests performed confirm the proposed objective, comparing the system reactions through ideal and abnormal situations.

Keywords: Data analysis. Machine learning. Industry 4.0. Arduino Predictive maintenance.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

APP	Aplicativo
CLP	Controlador Lógico Programável
CV	Cavalo Vapor
EUA	Estados Unidos da América
GPS	Sistema de Posicionamento Global – Global Positioning System
HZ	Hertz
IA	Inteligência Artificial
IDC	Internacional Data Corporation
IoT	Internet das Coisas – Internet of Things
LAN	Rede de Área Local – Local Area Network
LED	Diodo Emissor De Luz – Light Emitting Diode
MIT	Massachusetts Institute of Technology
mm	Milímetro
OPC	Plataforma Aberta de Comunicações – Open Platform Communications
UA	Arquitetura Unificada – Unified Architecture
VAC	Tensão Alternada – Volts Alternate Current
VCC	Tensão Contínua – Volts Continue Current
XML	Linguagem De Marcação Extensível – Extensible Markup Language

SUMÁRIO

INTRODUÇÃO

Com o avanço tecnológico na segunda metade do século XX, a indústria mudou o seu conceito de linha de produção, passando a produzir de forma automatizada. Entrando no século XXI tem-se uma nova etapa com a introdução da indústria 4.0 que começou a operar e analisar de forma remota equipamentos e processos de diferentes setores da empresa.

Antes da informatização da indústria já existiam processos sendo realizados por equipamentos automatizados tais como Controladores Lógicos Programáveis (CLP), braços robóticos, entre outros, porém a comunicação se limitava a troca de variáveis entre as máquinas para a automatização dos processos. Hoje, com o uso de sistemas inteligentes de leitura de dados e tomada de decisões, as empresas conseguem gerenciar diferentes etapas de seu processo produtivo de forma a diminuir possíveis desperdícios, paralisações e melhorar a qualidade do produto final.

O conceito 4.0 pode ser aplicado desde o controle de estoque até a gestão de recursos humanos, isso permite a integração de uma empresa como um todo. Pode ser dividido em três grandes etapas: coleta de dados, análise das informações obtidas e readequação dos processos conforme necessidade. A coleta de dados consiste na medição de parâmetros como: temperaturas, velocidade de rotação de motores, consumo de corrente e ociosidade. Já a etapa de readequação consiste inicialmente na análise das informações extraídas, por exemplo, caso o sistema detecte um aumento no consumo de corrente em determinado amperímetro, ele deve sinalizar que há uma anomalia naquele ponto. Sistemas mais complexos são capazes de identificar a causa do defeito e realizar ações para corrigi-las.

Diante do exposto, o objetivo do trabalho intitulado Aplicação de Análise de Dados em Máquinas Industriais com Eixos Rotativos é coletar dados de sensores, analisar e tomar decisões baseadas em um algoritmo de aprendizado de máquina para um sistema não supervisionado. Justifica-se que o projeto auxilia na manutenção preditiva de equipamentos, processando os dados obtidos pelos sensores e comparando-os com a média das cem primeiras amostras, definindo assim, as ações

que devem ser realizadas pelo manutentor caso haja alguma falha eminente. Para isso, o software é instalado em um smartphone que através da comunicação via Bluetooth, realiza a interação com o Arduino, que atua diretamente na coleta dos dados obtidos.

Desta forma, para o desenvolvimento do projeto, este é dividido em três capítulos:

Capítulo 1 – Fundamentação teórica: abordagem das teorias que dão sustentação ao desenvolvimento do projeto;

Capítulo 2 – Metodologia: descreve a trajetória a percorrer para a elaboração do projeto. Fornece métodos que são procedimentos amplos do raciocínio e técnicas que são procedimentos mais restritos que operacionalizam os métodos mediante emprego de instrumentos adequados;

Capítulo 3 – Desenvolvimento do projeto: descreve passo a passo a construção e o desenvolvimento do projeto intitulado Aplicação de Análise de Dados em Máquinas Industriais com Eixos Rotativos;

E finalmente, as Considerações finais: em linhas gerais, aborda-se o objetivo e justificativa feita na introdução, apontando, as relações existentes entre os fatos verificados e as teorias, conquistas alcançadas, pontos fortes e fracos do trabalho e possíveis sugestões para futuros trabalhos.

1 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo encontram-se as teorias que dão sustentação ao desenvolvimento do projeto intitulado Aplicação de Análise de Dados em Máquinas Industriais com Eixos Rotativos.

1.1 Breve histórico sobre as revoluções industriais

Coelho (2016) relata que a primeira Revolução Industrial aconteceu entre os anos de 1760 e 1840 na região da Inglaterra, substituindo atividades realizadas de forma artesanal pelo uso de máquinas e equipamentos, em sua grande maioria com a utilização da energia a vapor. Este movimento afetou os níveis sociais e econômicos, tirando do artesão a responsabilidade da procura pela matéria prima, controle do processo produtivo e comercialização do produto final, transferindo para um patrão e tornando-o um empregado.

A segunda Revolução Industrial surgiu na década de 1850 até o final da Segunda Guerra Mundial (1945). As principais evoluções foram percebidas nas áreas da indústria elétrica, química e do aço, este último, presente nos primeiros barcos acoplados por grandes motores a vapor, responsáveis por revolucionar o transporte de mercadorias. Durante este período surgiram também as primeiras linhas de produção serial, permitindo um maior volume de produção com um custo reduzido.

A partir da década de 1950 houve a disseminação do uso dos semicondutores, dos computadores e dos robôs nas linhas de produção, dando um grande passo na automação industrial. Também houve melhorias nos sistemas de comunicação e armazenamento de dados, além do surgimento da internet. Todos esses aspectos transformaram este período na Terceira Revolução Industrial, conhecida como a revolução digital.

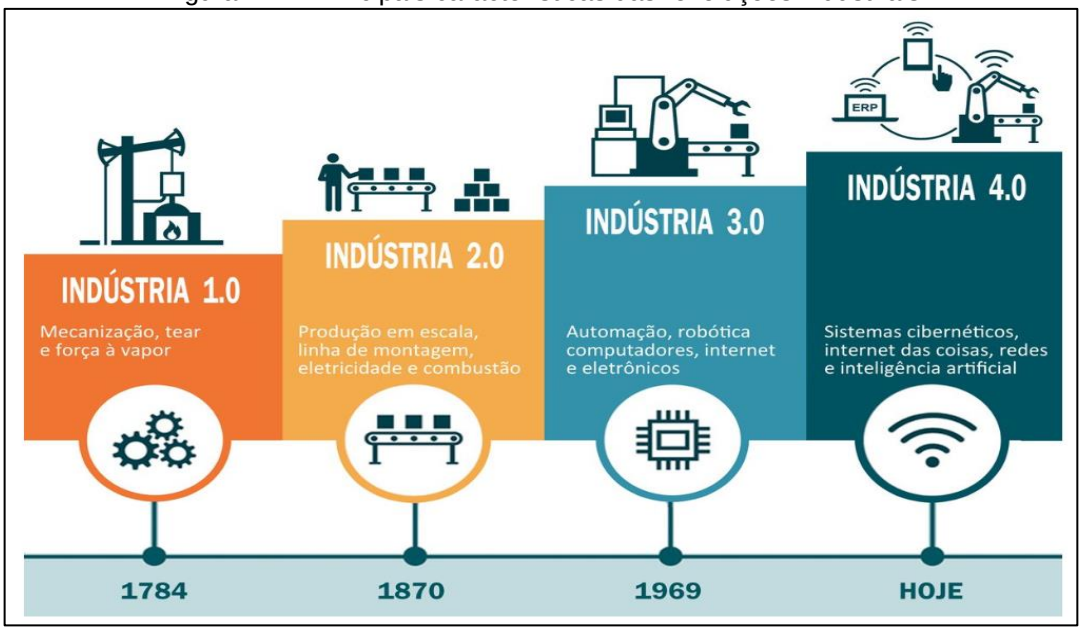
Em 2011, com a crescente necessidade de agilidade e flexibilidade de produção de bens e serviços, surgiu na feira industrial de Hannover, na Alemanha, um

novo conceito de revolução, chamada de 4ª Revolução Industrial, conhecida como Indústria 4.0. Esse novo modelo de fabricação visa a produção customizada em massa.

Ainda, Coelho (2016) enfatiza que na Industria 4.0 é preciso que as informações sejam trabalhadas com rapidez e de forma que possam ser obtidas de diferentes locais não importando a distância entre eles e por quantas vezes forem necessárias, ou seja, em tempo real. São implementados nos processos, dispositivos mais inteligentes, fazendo uso de sensores, monitorando em tempo real, para melhorar a capacidade de análise de dados, avaliar desempenho e corrigir possíveis problemas.

A Industria 4.0 tem parte de sua origem no MIT (Massachusetts Institute of Technology) no estado de Boston, EUA. Ela é usada para sintetizar ideias de que dispositivos físicos e virtuais estão conectados à Internet, cujo objetivo é permitir que as informações estejam disponíveis para uso, manipulação ou consulta por quaisquer equipamentos, softwares e redes. A Figura 1.1 destaca as principais características das revoluções industriais.

Figura 1.1 – Principais características das revoluções industriais



Fonte: www.anadi.com.br, 2018

1.2 Aprendizado de máquina

Segundo Hurwitz *et al*. (2018, p.4), “O *machine learning* é uma forma de Inteligência artificial (IA) que permite que um sistema aprenda a partir de dados e não através de programação explícita”.

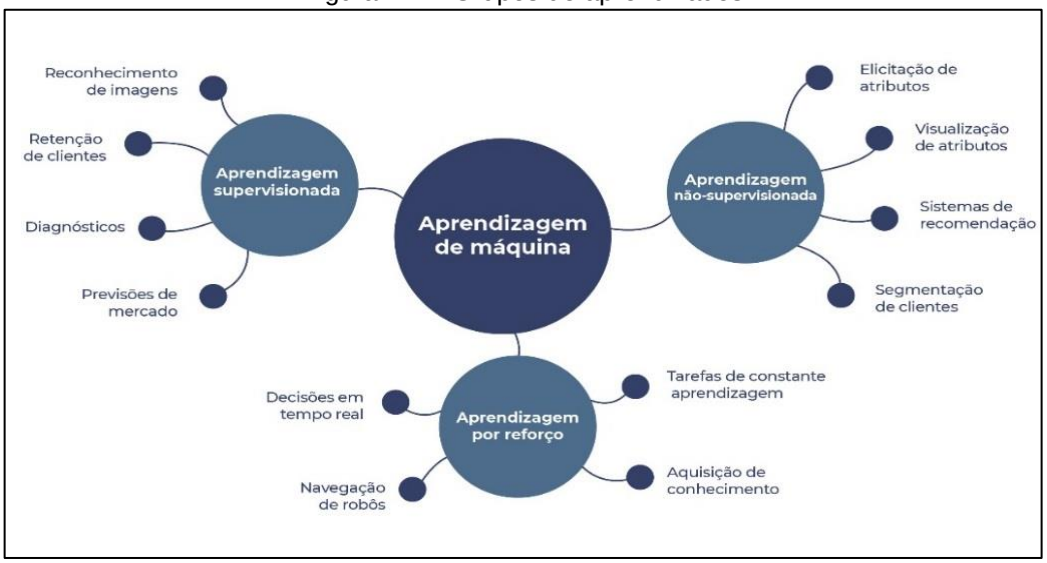
Uma outra forma de explicar o que é *machine learning* é dada por Li Hui (2019), “É um ramo da inteligência artificial baseado na ideia de que sistemas podem aprender com dados, identificar padrões e tomar decisões com o mínimo de intervenção humana”.

Hurwitz *et al*. (2018) destaca que a inteligência artificial usa uma variedade de algoritmos que coletam dados de forma iterativa com o intuito de produzir saídas cada vez mais precisas, como no caso das recomendações em sites, que oferecem aos seus usuários produtos semelhantes aos que foram pesquisados por ele anteriormente.

Uma parte importante desses sistemas são as suas bases de dados, por exemplo, um sistema que tenha como finalidade gerir a produção e estoque de produtos perecíveis com base nas previsões de venda, para ser preciso, deve conhecer previamente os padrões de venda e mercadorias em estoque em cenários diferentes como número de vendas por produto por mês. Quanto mais informações para serem comparadas melhor a precisão da saída.

Honda, Facure e Yaohao (2017) destacam que os algoritmos de *machine learning* são divididos em três grupos: aprendizado supervisionado, não supervisionado e por reforço, sendo que cada um é comumente utilizado para determinadas aplicações conforme ilustrado na Figura 1.2.

Figura 1.2 – Grupos de aprendizados



Fonte: www.isitics.com, 2019

1.2.1 Aprendizado supervisionado

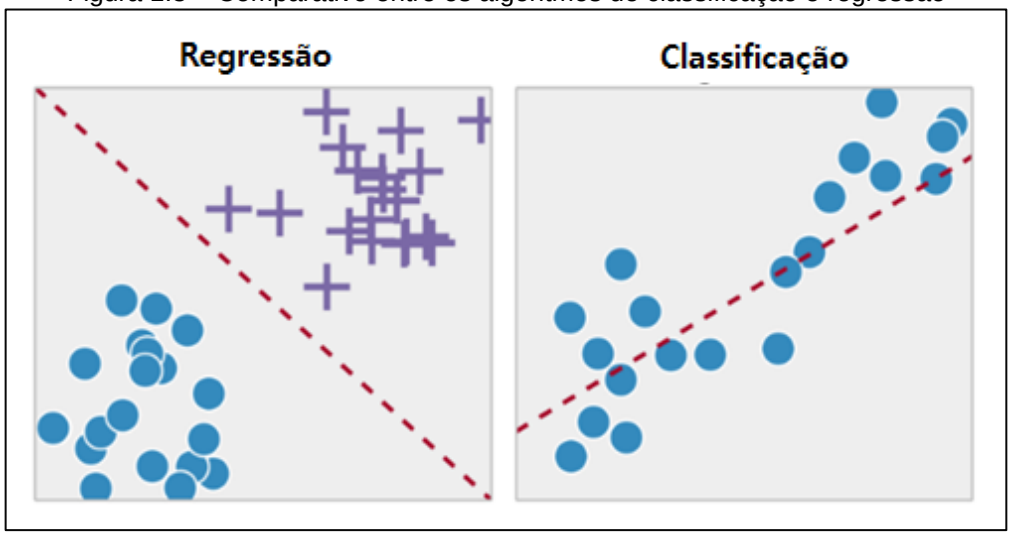
Barros (2016), explica que o aprendizado supervisionado ocorre quando há uma relação direta entre os dados coletados pelo sistema e a sua tomada de decisão. Um exemplo prático é a análise de crédito realizada pelo banco para fazer um empréstimo, é inserido no banco de dados da aplicação uma tabela que relaciona pessoas que não são consideradas passíveis de receber o crédito com suas características, tais como: idade, cidade, profissão, tempo de trabalho na carteira e últimos empréstimos. Com base nessas características quando inserido um novo nome na tabela, o algoritmo é capaz de prever o risco para a instituição de realizar essa transação. Neste tipo de aplicação é necessário conhecer quais são as respostas possíveis para a saída do sistema, no caso são duas: existe risco e não existe risco.

Os algoritmos de aprendizagem supervisionados são classificados em problemas de regressão e classificação. A regressão é a tarefa de prever uma quantidade contínua na saída, ou seja, sempre que a saída do algoritmo for um número real o algoritmo deve ser de regressão. Um exemplo de sistema de aprendizado por regressão é um aplicativo de um corretor imobiliário no qual são inseridos o bairro, a quantidade de quartos, tamanho e o preço que algumas casas

foram vendidas e com base nesses dados o algoritmo é capaz de prever qual o valor de venda ideal para uma nova casa que esteja a venda.

Os problemas de classificação de algoritmos têm como objetivo mapear variáveis de entrada e separá-las em categorias distintas. Esses algoritmos apontam que as entradas possuem uma ou mais classes de saídas formando um conjunto finito de possibilidades. A Figura 1.3 ilustra o comparativo entre as saídas geradas através de algoritmos de classificação e regressão.

Figura 1.3 – Comparativo entre os algoritmos de classificação e regressão



Fonte: SILVA, 2018, p.2

1.2.2 Aprendizado não supervisionado

Honda, Facure e Yaohao (2017) definem que o aprendizado não supervisionado não rotula em grupos as variáveis de entrada de forma que o próprio algoritmo detecta quais atributos são relevantes e/ou semelhantes e os agrupam. São utilizados em problemas onde não se sabe quais os resultados o sistema deve apresentar e o usuário não necessariamente sabe qual o efeito na saída de acordo com as mudanças nas variáveis de entrada. Como exemplo, um sistema que traça o perfil dos consumidores de um supermercado, ele é capaz de detectar a relação entre clientes que compram arroz e cerveja sem que isso seja programado, apenas com base em comparações ele cria relações entre esses consumidores aleatórios.

Este método de programação é utilizado em sistemas de manutenção preditiva. São inseridos no sistema os parâmetros dos sensores de quando o sistema está com alguma peça danificada e quando está operando de forma ideal, esse processo é chamado de treinamento.

O sistema treinado recebe os dados coletados pelos sensores de campo e caso haja uma anomalia em relação aos valores ideais, o algoritmo irá aplicar o método de similaridade que também é conhecido como técnica de "*nearest-neighbor*". Essa técnica começou a ser utilizada ainda nos anos de 1960, ainda que para outras finalidades.

Ponce e Alcaraz (2013) explicam que a técnica consiste em se calcular a distância entre cada dado de um conjunto de valores como (X_1, X_2...X_n), que representam os dados dos conjuntos já treinados, com um segundo grupo de valores (Y_1, Y_2...Y_n) que são os valores lidos pelos sensores. A distância entre esses dois pontos do conjunto pode ser calculada seguindo algumas técnicas como, por exemplo, o método da distância euclidiana, apresentada na equação 1.1.

$$d(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

Equação [1.1]

Onde p_n representa o elemento do primeiro grupo e q_n do segundo grupo de dados.

O valor obtido na equação representa a similaridade do estado do maquinário com o conjunto treinado, quanto mais próximo a 0 (zero), maior é a semelhança. Dessa forma ao submetermos as leituras dos sensores ao algoritmo em diferentes instantes, é possível observar se o estado atual do equipamento está tendendo a se igualar ao estado treinado como falha.

1.2.3 Aprendizado por reforço

Silva (2018) enfatiza que em um ambiente de aprendizado por reforço, o algoritmo de inteligência (agente) recebe uma indicação do estado atual do sistema (ambiente) que ele está interagindo. Com base nisso, ele toma uma decisão alterando o estado de sua saída, gerando uma alteração no ambiente. Em seguida, o agente recebe um valor de reforço, geralmente numérico e decimal, quanto melhor for o resultado apresentado pelo algoritmo maior é o valor do reforço. Dessa forma, o agente sempre toma ações que maximizem o valor da soma dos reforços recebidos.

Ressalta-se que como o sistema é não determinístico, uma mesma ação do agente pode apresentar diferentes alterações no ambiente, além disso, não existem entradas e saídas pré-inseridas no sistema para treiná-lo, de forma que ele precisa obter experiência para conhecer as ações, estados e recompensas possíveis.

Por exemplo, um sistema que elabora portfólios no mercado financeiro, tecnicamente as ações da bolsa de valores são imprevisíveis, o agente sugere a compra ou venda de ações e com base na rentabilidade ou não de sua sugestão um reforço é dado ao sistema. Dessa forma, um sistema experiente tende a ser assertivo por ter aprendido quais são as variáveis que fazem com que ele receba um reforço maior. A Figura 1.4 representa o ciclo de um sistema por reforço.

Figura 1.4 – Ciclo de um sistema por reforço

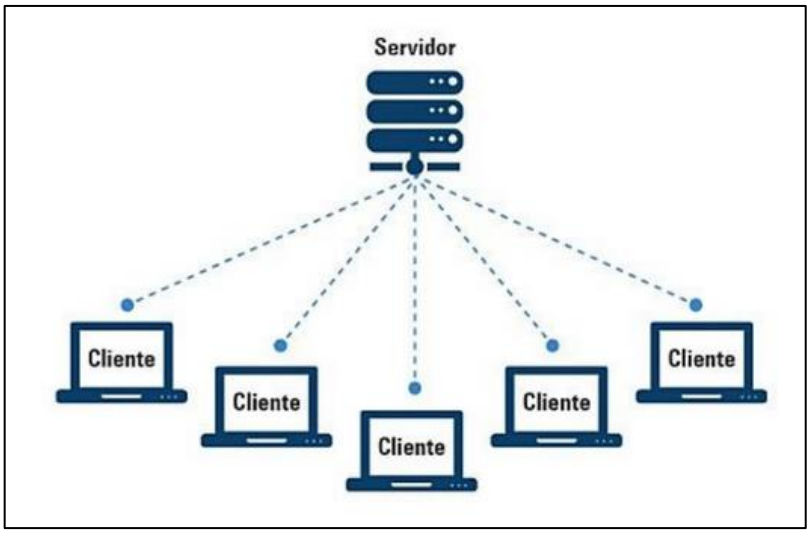
Fonte: DIAS JÚNIOR, 2012, p.26

1.3 Aplicativos móveis

Nonnenmacher (2012) destaca que os aplicativos móveis são softwares hospedados em sistemas operacionais de smartphones e tablets capazes de acessar conteúdo *online* e *offline*. Eles são desenvolvidos em plataformas móveis, sendo as principais o Android e o iOS. O Android é uma plataforma aberta e mais fácil de se utilizar por parte dos desenvolvedores, diferente do iOS que só pode ser vinculado aos dispositivos móveis de sua desenvolvedora, a Apple. Eles podem ser destinados para diferentes finalidades como a comunicação, diversão e a comodidade na realização de tarefas cotidianas.

Petcov (2018) aponta que os aplicativos são criados utilizando arquiteturas de programação e o modelo mais utilizado é o cliente/servidor. Neste tipo de configuração, os servidores são equipamentos que utilizam de seus recursos físicos como o processador, memória e disco rígido para processar os dados recebidos dos clientes e, posteriormente, devolvê-los aos mesmos, estabelecendo um canal de comunicação. A Figura 1.5 mostra o modelo de arquitetura cliente/servidor.

Figura 1.5 – Modelo cliente/servidor



Fonte: PETCOV, 2018, p.38

Iwai (2016) ressalta algumas vantagens dos aplicativos:

Fácil utilização: através da otimização de recursos presentes nos dispositivos, tornam a navegação e ações do usuário mais simples e ágeis;

Mobilidade: os aplicativos permitem o acesso a informações de maneira rápida e prática através dos celulares;

Fonte de dados: armazenam dados que podem ser utilizados na construção do perfil do usuário, gerando sugestões e tomadas de decisões.

A IDC - Internacional Data Corporation (2017), informa que a utilização de aplicativos vem despertando a atenção dos meios acadêmicos e empresariais. Estudos realizados apontam que até o ano de 2020, 40% das 3000 principais empresas da América Latina dependerão de produtos, serviços e experiências digitais. Mostra também que 2 em cada 5 empresas, possuem equipes dedicadas a promover a transformação digital.

Freitas (2017) destaca que na Indústria 4.0, que é caracterizada por fábricas inteligentes, onde os equipamentos, recursos e clientes estão interligados, é comum o investimento por parte de empresas em aplicativos capazes de promover a otimização de seus processos, com a redução de tempo, recursos e custos. Estes sistemas permitem o desenvolvimento de uma inteligência artificial baseada no conceito de *machine learning*, onde são observados padrões de cada processo, de modo a diminuir a ocorrência de futuros erros.

Pierson (2019) diz que no IoT (Internet das coisas - *Internet of things*), conceito que representa a conectividade entre equipamentos e máquinas à internet, os aplicativos são utilizados na conscientização contextual. Este processo consiste na captação de dados gerados por sensores e que são enviados a um micro controlador e transformados em uma visão ampla e detalhada do que acontece no ambiente.

Uma aplicação IoT é encontrada em sistemas de monitoramento remoto na logística, onde os materiais recebem transmissores ou etiquetas de radiofrequência que permitem sua rastreabilidade, detalhando sua movimentação durante toda a cadeia de fornecimento. Através de um aplicativo, os clientes podem acompanhar seus produtos em tempo real, tornando este serviço mais cômodo e confiável.

Novus (2017) destaca que o monitoramento remoto contribui de forma significativa para o aumento de produtividade e otimização dos custos de uma

empresa. Desta forma, foram desenvolvidos equipamentos que promovem a conexão entre os dispositivos e os aplicativos, através dos celulares. Essa comunicação é feita com protocolos que utilizam os serviços de dados móveis.

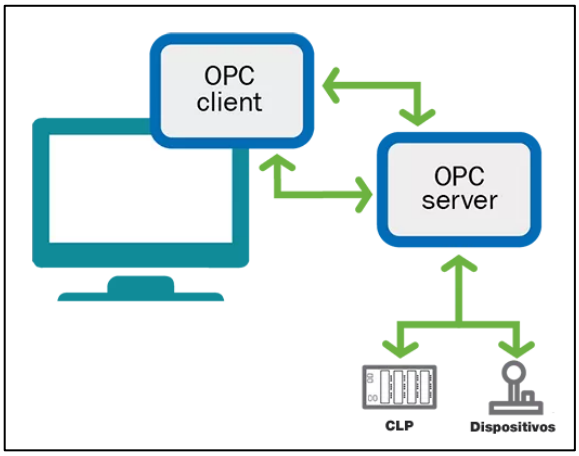
1.4 Interface de comunicação

Romano (2017) relata que a interface de comunicação OPC UA (*Open Platform Communications - Unifield Architecture* – Plataforma Aberta de Comunicação – Arquitetura Unificada), foi desenvolvida objetivando a troca de dados entre dispositivos que fazem uso de protocolos de comunicações distintos, podendo eliminar o uso de drivers proprietários.

O autor ainda ressalta que uma das vantagens do OPC UA é que não se limita ao uso de uma plataforma Windows para funcionar, podendo ser utilizado em sistemas como o Mac OS e Linux. Além de maior universalização, também trouxe maior segurança e um sistema extensivo que se utiliza de uma arquitetura multicamadas, garantindo a incorporação de inovações tecnológicas que ainda nem foram criadas.

Os OPC's, em geral, podem ser divididos em dois grandes grupos, sendo eles o cliente e o servidor. A Figura 1.6 ilustra a comunicação simplificada entre esses grupos.

Figura 1.6 – Comunicação simplificada do OPC



Fonte: www.logiquesistemas.com.br, 2017

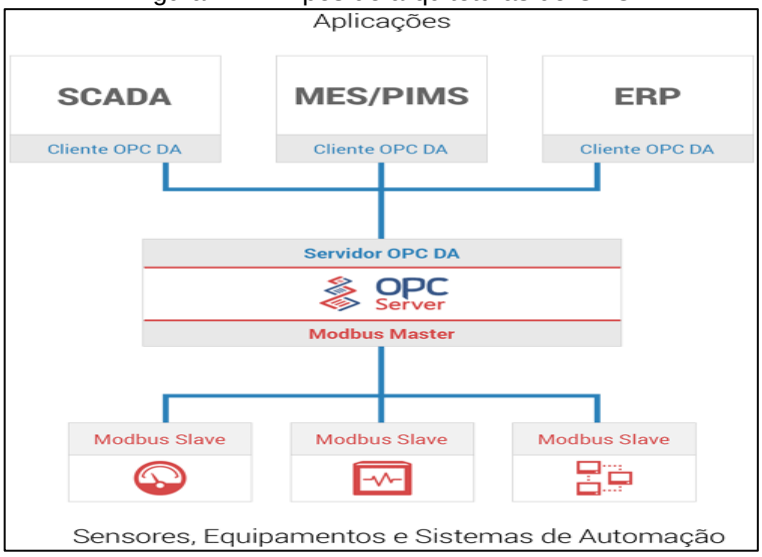
A seguir, apresentamos os modelos de OPC's:

OPC servidor – estabelece uma comunicação full-duplex entre o OPC e os dispositivos de campo que recebem e enviam dados;

OPC cliente – estabelece uma comunicação entre o OPC cliente e o OPC servidor, permitindo ao usuário final uma leitura e o envio de dados.

Venturelli (2018) enfatiza que o OPC UA "utiliza uma arquitetura robusta, com mecanismos de comunicação confiáveis, monitoramento de tempo configurável e detecção automática de falhas". A Figura 1.7 ilustra os tipos de arquiteturas do OPC.

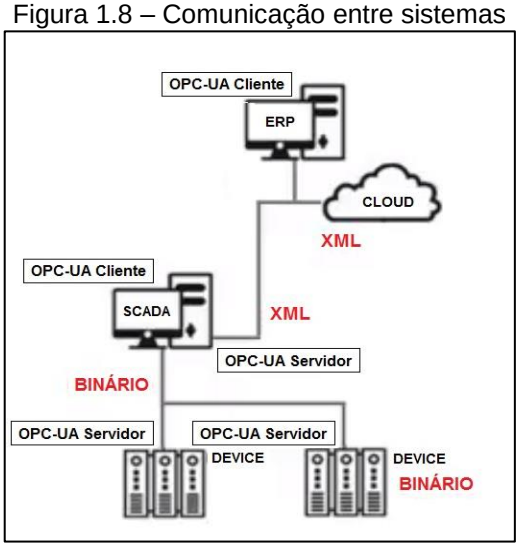
Figura 1.7 – Tipos de arquiteturas do OPC



Fonte: www.inforchannel.com.br, 2017

Quanto ao formato de mensagem, o OPC UA suporta dois tipos de comunicação, a Binária e a XML. A Figura 1.8 ilustra a comunicação entre os sistemas e o formato em que são transmitidas as mensagens.

Figura 1.8 – Comunicação entre sistemas



Fonte: www.automacaoindustrial.info, 2019

UA Binário é mais comumente utilizado entre máquinas, pois como o próprio nome diz, se utiliza de código binário para comunicação. Esse formato é considerado mais leve devido a sua linguagem ser pura, ou seja, nem o receptor nem o emissor precisam decodificar a mensagem, pois ambas trabalham com o sistema 0 e 1.

UA XML (*Extensible Markup Language*), de modo geral, podem ser encontradas no OPC Cliente, pois é considerada uma codificação de alto nível, ou seja, que pode ser interpretada pelo usuário final. Devido sua necessidade de codificar e descodificar o sinal esse formato se torna mais pesado em relação ao UA Binário que realiza uma transmissão direta, sem conversão.

1.5 Controlador lógico programável

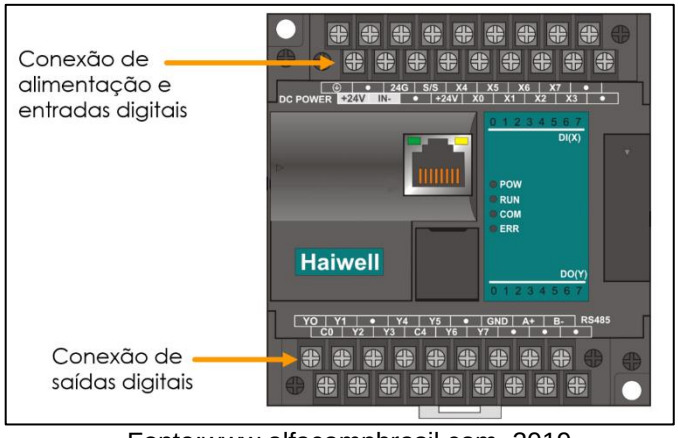
Sturaro (2009) destaca que o controlador lógico programável (CLP) é um equipamento semelhante a um computador, mas para uso industrial, que pode armazenar sequência de decisões e ações para controlar sistemas automatizados, mediante informações obtidas do meio físico ou virtual externo, ou seja, informações de um processo produtivo ou linha de produção, a Figura 1.9 exemplifica um CLP de pequeno porte CP1 da Omron.

Figura 1.9 – CLP de pequeno porte CP1 da Omron

Fonte: www.pyrotec.com.br, 2019

Os sinais são grandezas elétricas e físicas, que enviados pelos sensores de campo são coletados através dos módulos de entrada, podendo ser digitais ou analógicos. Os módulos de saída compõem a parte do hardware responsável por enviar sinais digitais ou analógicos, acionar outros equipamentos e cargas, dispositivos que desempenham um trabalho no processo. A Figura 1.10 ilustra as entradas e saídas de um CLP.

Figura 1.10 – Entradas e saídas de um CLP



Fonte:www.alfacompbrasil.com, 2019

A sequência de decisões que manipulam os dados com instruções de lógica, aritmética, temporização, contagem e comunicação, compõem o programa de aplicação, feito pelo usuário por meio de diferentes linguagens de programação, por

exemplo o Ladder, e são utilizadas para fazer o controle do processo produtivo desejado.

Gonçalves (2009) descreve que as características e tipos de entradas digitais variam de acordo com o modelo do CLP ou módulo de expansão, como a tensão elétrica do sinal de leitura pode ser 110 ou 220 V em corrente alternada. Tem-se ainda a possibilidade de 12, 24 ou 125 V em corrente contínua.

O funcionamento do controlador lógico programável (CLP) se dá através de entrada, controle e saída. As entradas são sensores capacitivos, indutivos, mecânicos, fotoelétricos ou geradores que passam as informações do sistema para o processamento, que é o computador em si. O Scan realiza a leitura do programa ininterruptamente mesmo que não haja mudança nas entradas e para que as saídas sejam sempre atualizadas. Esse processo é feito em tempo real para que a operação seja a mais precisa e segura possível. A Figura 1.11 ilustra a sequência do ciclo de Scan.

Figura 1.11 – Sequência do ciclo de Scan

Fonte: www2.pelotas.ifsul.edu.br, 2019

Azevedo (2018) disserta que com a crescente inovação tecnológica, incluindo a 4ª Revolução Industrial, o CLP acompanha o avanço, bem como os modos de comunicação com outros equipamentos, CLP’s ou sistemas virtuais para armazenamento e manipulação de dados. A comunicação pode ser feita por meio de

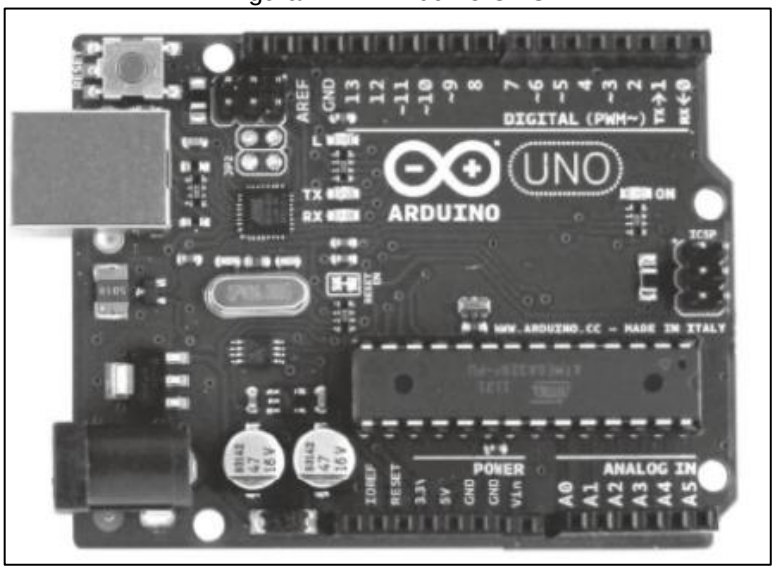
formas sem fio, através da internet e redes LAN. Com isso, é possível monitorar e gerenciar equipamentos ou processos remotamente. Além disso, as informações armazenadas ou extraídas fácil e remotamente viabilizam a análise e gerenciamento para uma manutenção preditiva eficaz.

1.6 Arduino

Nussey (2019) explica que o Arduino pode ser tanto um hardware quanto um software. Ele é uma placa eletrônica composta por alguns componentes, sendo o principal deles o microcontrolador, projetado para se comunicar através de entradas e saídas. O microcontrolador é um pequeno computador responsável por controlar outros aparelhos eletrônicos. Sua programação é feita através do *software* Arduino, onde são escritas linhas de códigos para ordenar que um LED acenda ou crie condições para que o mesmo seja feito, estabelecendo rotinas para o sistema em que se trabalha.

McRoberts (2011) destaca que tanto o *software* quanto o *hardware* do Arduino são de fonte aberta, ou seja, todos os recursos, códigos, projetos e esquemas podem ser utilizados livremente por qualquer um e para qualquer finalidade, estabelecendo uma grande comunidade de pessoas adeptas ao uso do Arduino, que compartilham suas experiências e tornam fáceis os acessos aos conteúdos referentes ao dispositivo. O *hardware* do Arduino pode receber o acoplamento de *shields*, denominação utilizada para placas eletrônicas contendo outros dispositivos, tais como *Bluetooth*, receptores GPS e *displays* de LCD. Existem diversos tipos de placas Arduino, sendo a mais comum delas o Arduino UNO, como mostra a Figura 1.12.

Figura 1.12 – Arduino UNO



Fonte: NUSSEY, 2019, p.18

O UNO é a versão 1.0 do *software* Arduino e é composto por um *chip* microcontrolador ATmega328P. Ele é conectado a outros componentes através de soquetes distribuídos ao redor da placa, que são divididos em pinos digitais, analógicos e de alimentação. Os pinos podem apresentar comportamentos de entradas ou saídas, recebendo ou emitindo sinais elétricos responsáveis pela comunicação do microcontrolador com os dispositivos nele conectados. O Arduino UNO pode ser alimentado por uma tensão de 5 V ou 3,3 V e sua programação se realiza por uma entrada USB, conectada ao computador onde o código de programação é compilado e transferido ao *hardwar*e através do *software* IDE (ambiente de desenvolvimento integrado).

2 METODOLOGIA

Neste capítulo encontram-se as diretrizes utilizadas para o desenvolvimento do projeto intitulado Aplicação de Análise de Dados em Máquinas Industriais com Eixos Rotativos. Trata-se de uma pesquisa aplicada que é desenvolvida nas dependências da FATEC São Bernardo do Campo e também nas residências dos integrantes do grupo.

Prodanov e Freitas (2013) enfatizam que a metodologia consiste no estudo, compreensão e avaliação de técnicas e métodos que possibilitam a construção do conhecimento, comprovando sua veracidade e aplicabilidade.

Ferrari (1974) destaca que o método científico é um instrumento utilizado para ordenar os pensamentos em sistemas e define procedimentos que levam a um objetivo preestabelecido. O conhecimento científico tem como principal característica sua verificabilidade, ou seja, é baseado em informações de fontes confiáveis que oferecem detalhes a respeito do tema proposto.

A redação do trabalho é realizada com o auxílio do Manual de Normalização de Projeto de Trabalho de Graduação da FATEC São Bernardo do Campo (2017) que tem como base as normas da ABNT. O trabalho é escrito em linguagem simples e clara, seguindo um raciocínio lógico e terminologia adequada.

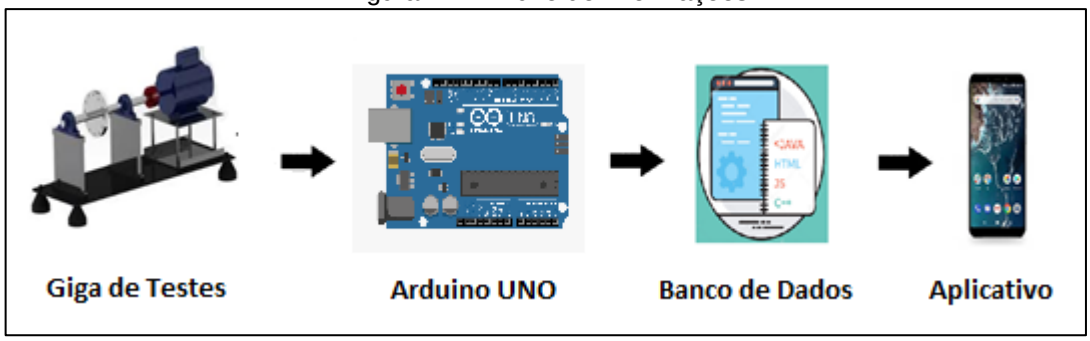
2.1 O tema-problema com justificativa e fluxograma do funcionamento

O tema-problema que se intitula Aplicação de Análise de Dados em Máquinas Industriais com Eixos Rotativos tem como objetivo a coleta de dados de sensores, análise e tomada de decisões com base em um algoritmo de aprendizado de máquina para um sistema não supervisionado. Tem como justificativa auxiliar na manutenção preditiva, processando os dados provenientes dos sensores e compara-os com os valores considerados ideais, obtidos através da média das cem primeiras amostras

coletadas, definindo as ações que devem ser realizadas pelo manutentor caso haja risco eminente de falha.

Para a construção do projeto utiliza-se um Arduino, um smartphone e uma giga de testes composta por um motor, eixo, mancal, polia e sensores. Na parte do *software* do aplicativo, todo o desenvolvimento dos algoritmos de programação é realizado no *software* Eclipse utilizando-se a linguagem de programação JAVA. A Figura 2.1 ilustra o fluxo de informações durante o processo.

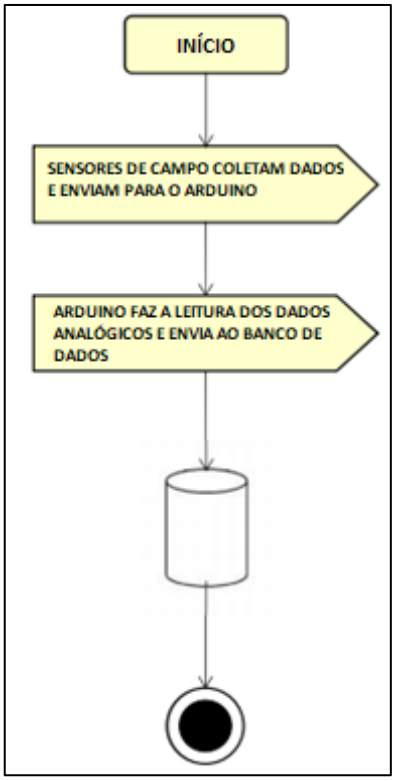
Figura 2.1 – Fluxo de informações



Fonte: Autoria própria, 2019

O fluxograma de funcionamento do projeto é divido em duas partes: a primeira retrata a coleta de dados e a segunda o processamento deles. Inicialmente com os sensores de campo enviando os dados obtidos para o Arduino, este com base na lógica de programação, realiza a leitura e tratamento dos dados analógicos que são transmitidos ao banco de dados. A Figura 2.2 ilustra o fluxograma de coleta de dados do projeto.

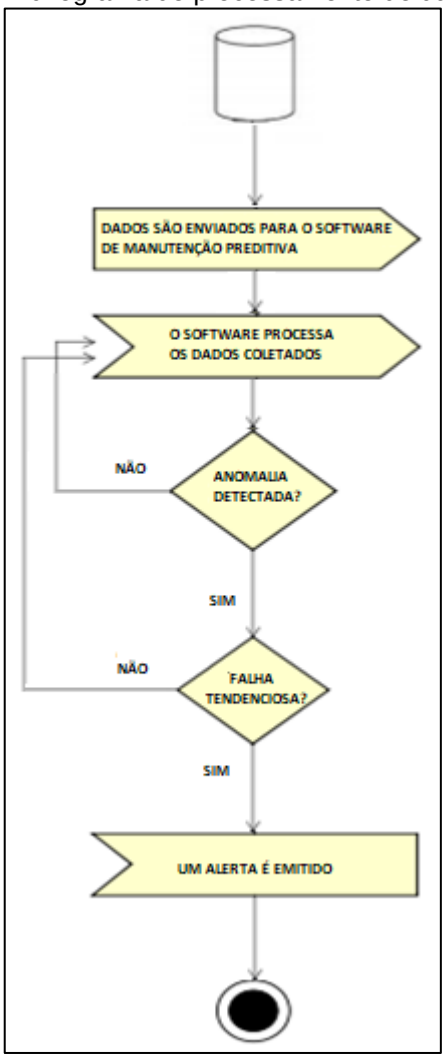
Figura 2.2 – Fluxograma de coleta de dados do projeto



Fonte: Autoria própria, 2019

A segunda parte do fluxograma é iniciada com a consulta do software ao banco de dados. O processamento dos dados obtidos se dá através da lógica de programação e, caso o programa identifique a presença de situações anormais no processo, um alerta é emitido, caso contrário, retorna a etapa de processamento e análise dos dados. A Figura 2.3 representa o fluxograma de processamento de dados do processo.

Figura 2.3 – Fluxograma de processamento de dados do projeto



Fonte: Autoria própria, 2019

2.2 Etapas teóricas e práticas para elaboração do projeto

Após delimitado o tema a ser desenvolvido e sua justificativa, é escolhido o professor orientador com base em sua área de especialização e a relação com o enfoque do trabalho, de modo a auxiliar na divisão das etapas do projeto, acompanhando e esclarecendo dúvidas em todo o desenvolvimento do mesmo.

Primeira etapa: reunião dos membros do grupo junto ao orientador a fim de definir as diretrizes para início das pesquisas. O orientador fez uma breve explanação a respeito do tema, indicando alguns materiais bibliográficos e marcou

obrigatoriamente um dia por semana para lhe apresentar o “andamento” das pesquisas.

Segunda etapa: realiza-se o levantamento bibliográfico na biblioteca da FATEC São Bernardo do Campo através de livros, artigos, dissertações, sites especializados, manuais e catálogos de empresas.

Terceira etapa: após leitura e releitura das bibliografias, faz-se a seleção daquelas que estão de acordo com o tema proposto. Constrói-se o Capítulo 1 – Fundamentação teórica e suas respectivas referências.

Quarta etapa: estudo da viabilidade econômica dos materiais para a construção do projeto. Pesquisas em sites e lojas especializadas. Aquisição dos materiais conforme Tabela 2.1.

Tabela 2.1 – materiais utilizados para a confecção do projeto

Componentes	**Quantidades**	**Valor (R$)**
Arduino Uno e proteção	1	55,00
Módulo Bluetooth HC-06	1	20,00
Fios conectores	20	10,00
Sensor de vibração SW-420	2	24,00
Sensor de corrente ACS712	1	22,00
Total		131,00

Fonte: Autoria própria, 2019

Quinta etapa: montagem do motor e do mancal na base de madeira, realizando a alimentação do motor trifásico e o acoplamento do eixo com a polia.

Sexta etapa: desenvolvimento do *software* para a placa do Arduino e fixação dos sensores de vibração e corrente na giga de testes. Inicia-se os ajustes dos *ranges* de funcionamento dos sensores e a coleta de dados entre a giga de testes e o Arduino.

Sétima etapa: é acrescentado a giga de testes um inversor de frequência para permitir a regulagem da velocidade de rotação do sistema.

Oitava etapa: realiza-se o desenvolvimento do algoritmo de inteligência artificial no *software* Eclipse, através da linguagem de programação *JAVA*.

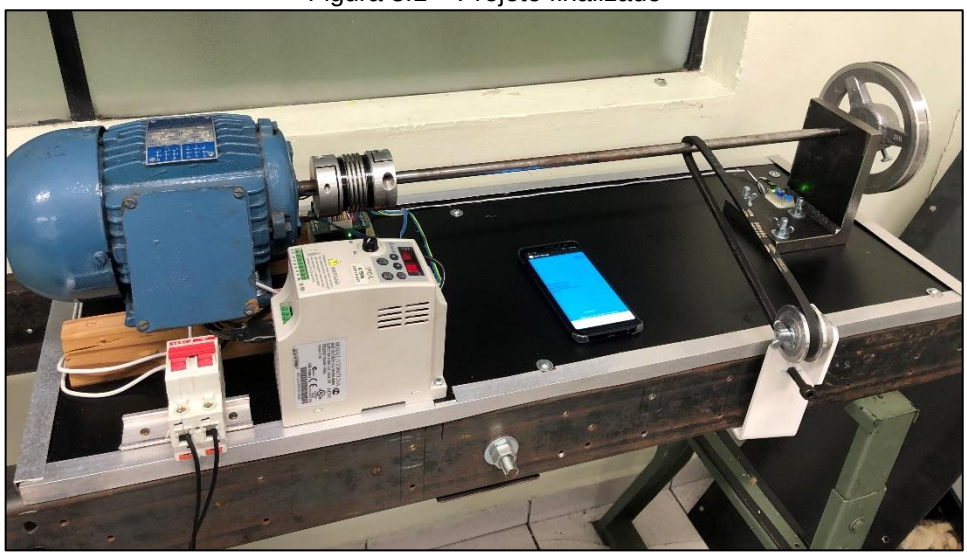
Nona etapa: integração das partes para realização dos testes finais. Armazena-se os resultados obtidos para análise do funcionamento do sistema.

Décima etapa: concretizado as etapas do projeto, faz-se as considerações finais e o resumo, juntamente com o *abstract*.

3 DESENVOLVIMENTO DO PROJETO

Neste capítulo encontra-se passo a passo o desenvolvimento e construção do projeto intitulado Aplicação de Análise de Dados em Máquinas Industriais com Eixos Rotativos. Para melhor entendimento e visualização do projeto, a Figura 3.1 ilustra-o finalizado.

Figura 3.1 – Projeto finalizado



Fonte: Autoria própria, 2019

O intuito do projeto é realizar a análise dos dados obtidos através de sensores acoplados a uma giga de teste, composta por um motor, eixo, mancal e polia, responsáveis por simular os componentes de máquinas operatrizes.

Os dados coletados são transferidos para o Arduino e posteriormente para o smartphone conectado via *Bluetooth*. Lá existe um aplicativo móvel desenvolvido através dos conceitos de *machine learning,* onde o software compara os valores obtidos dos sensores com valores ideais definidos pelo próprio algoritmo, apontando tendências de quebra dos componentes, permitindo a realização das manutenções preditivas.

O desenvolvimento e construção do projeto estão alicerçados nos seguintes tópicos:

- Montagem mecânica da giga de testes;
- Desenvolvimento do programa para a placa Arduino;
- Desenvolvimento do algoritmo de manutenção preditiva;
- Resultados obtidos;
- Obstáculos e soluções.

3.1 Montagem da estrutura mecânica da giga de testes

Após o levantamento e compra dos materiais necessários para a confecção da giga de testes, inicia-se a construção da mesma, que tem como objetivo simular os componentes e movimentações de máquinas operatrizes com eixos rotativos, permitindo assim, a captação das vibrações do sistema e a variação de corrente.

Na primeira parte aborda-se a montagem da estrutura mecânica. Usa-se uma base de madeira de 890 mm de comprimento, 330 mm de largura e 15 mm de espessura. Nela é fixado o motor de indução trifásico de 0,5 CV a 60 Hz, através de quatro parafusos M8 com porcas sextavadas. O motor é responsável pela movimentação dos componentes do sistema.

No eixo do motor existe um acoplamento de alumínio, que permite a transmissão do movimento ao segundo eixo. Ele é feito de aço ABNT 1020 trefilado, possuindo 12 mm de diâmetro e 600 mm de comprimento, apoiado por um mancal de aço ABNT 1015 com 15 mm de espessura, com uma bucha grafitada auto lubrificante que facilita a rotação e reduz o atrito. A Figura 3.2 ilustra o início da montagem mecânica da giga de testes.

Figura 3.2 – Início da montagem mecânica da giga de testes

Fonte: Autoria própria, 2019

No eixo acoplado ao motor é fixada uma polia de alumínio com diâmetro de 200 mm, simulando um componente de máquinas operatrizes. O desbalanceamento do sistema é realizado através do afrouxamento dos parafusos do mancal, desta forma, torna-se possível a coleta de dados através de sensores de vibração conectados a placa Arduino.

Na placa base, fixa-se o inversor de frequência, responsável pelo controle da frequência de alimentação que chega ao motor, permitindo o ajuste da velocidade de rotação do mesmo.

Uma vez realizada a montagem dos componentes, inicia-se a fixação da giga de testes em uma base de aço, visando a estabilização de toda a estrutura, ponto que afeta diretamente a captação das vibrações pelos sensores, desta forma, elimina-se falsas leituras. A Figura 3.3 ilustra a fixação da giga de testes na base de aço.

Figura 3.3 – Fixação da giga de testes na base de aço

Fonte: Autoria própria, 2019

Para frear o eixo do motor, utiliza-se uma correia de borracha fixada a um suporte na estrutura de aço e quando tensionada, promove um atrito no eixo, diminuindo sua rotação. Desta forma, torna-se possível simular o atrito ocorrido nos rolamentos de uma máquina rotativa, causados por desgastes ou má lubrificação. Esta diferença ocasiona o aumento dos valores de corrente consumidos pelo motor e é captada por sensores instalados em série com a alimentação do mesmo.

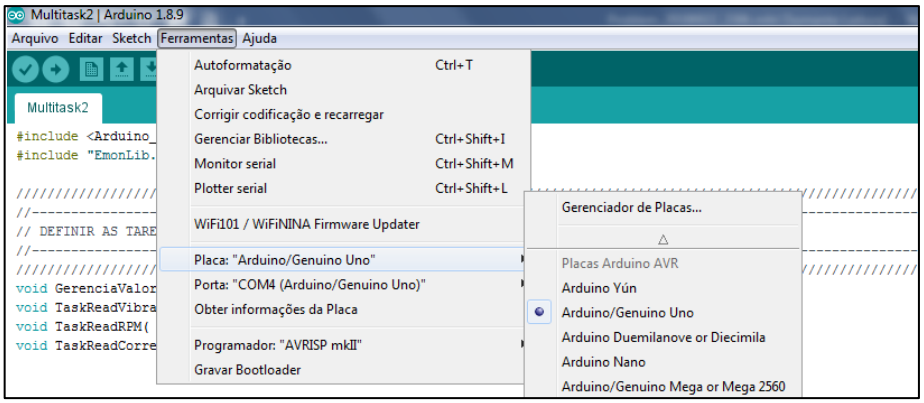
3.2 Desenvolvimento do programa para a placa Arduino

Depois da montagem da estrutura mecânica dá-se início ao desenvolvimento do programa para a placa Arduino. Por questões de viabilidade econômica, o uso do CLP e do OPC no projeto é substituído pela placa Arduino modelo UNO. Para sua configuração, primeiramente fez-se o *download* do *software* Arduino IDE e a instalação do mesmo.

Instalado o programa, a placa do Arduino UNO é conectada ao computador via cabo USB. Para o início da programação é necessário acessar o ícone do programa gerado na área de trabalho. Com o ambiente de desenvolvimento aberto, segue-se o

diretório Ferramentas > Placa > Arduino/ Genuino Uno, para a seleção do modelo do Arduino em que se deseja trabalhar. A Figura 3.4 ilustra a seleção do modelo no ambiente de desenvolvimento.

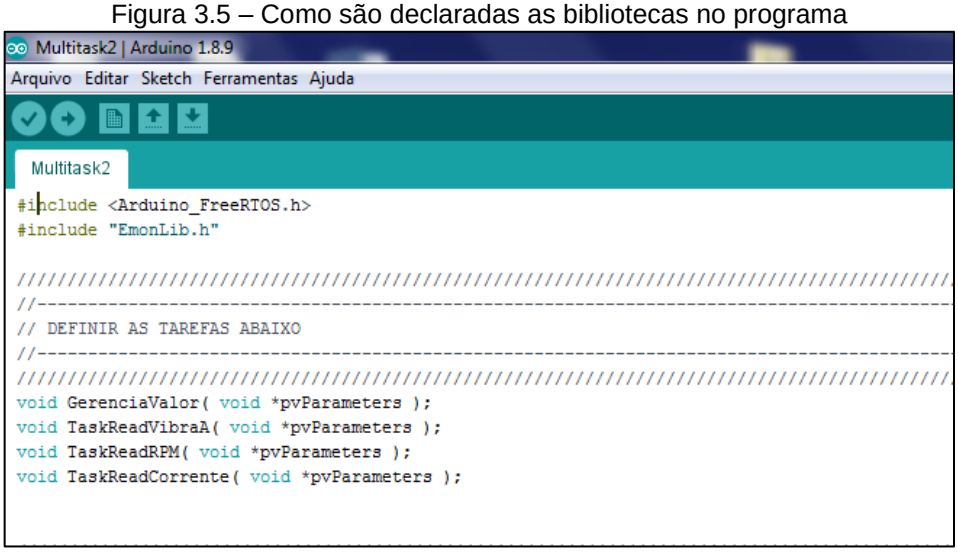
Figura 3.4 – Seleção do modelo no ambiente de desenvolvimento



Fonte: Autoria própria, 2019

Dando sequência, inicia-se a programação selecionando as bibliotecas, que são rotinas de auxílio ao programa principal com funcionalidades pré-programadas, necessitando apenas da adequação para o uso no projeto. Utiliza-se no programa a biblioteca Arduino_FreeRTOS.h, um sistema operacional de tempo real responsável por gerenciar múltiplas tarefas, baseando-se em critérios como: prioridade, criticidade e sequência de execução. Também se utiliza a biblioteca EmonLib.h que tem por finalidade o ajuste dos *ranges* dos sensores e transdutores, captando valores obtidos, realizando cálculos e retornando resultados. A Figura 3.5 ilustra como são declaradas as bibliotecas no programa.

Figura 3.5 – Como são declaradas as bibliotecas no programa



Fonte: Autoria própria, 2019

Para a leitura da corrente que circula pelo sistema é utilizado o sensor de corrente ACS712. Este sensor possibilita a leitura de valores de corrente contínua e corrente alternada. Seu funcionamento se dá através da utilização do efeito *hall* para identificar o campo magnético gerado pela passagem da corrente. A medição dos valores de corrente permite a identificação do aumento da mesma, gerada por falhas no sistema motriz como o desbalanceamento da carga e atritos que impedem que o motor atinja os valores nominais de rotação. Se permanecer por muito tempo, trabalhando com valores acima do especificado, a vida útil dos componentes é reduzida.

No código de programação são declarados os valores de calibração do sensor para a medição da corrente, os pinos analógicos em que ele está conectado e os valores de ruído a serem descartados no resultado final. A Figura 3.6 ilustra o bloco de programação para a parametrização do sensor de corrente.

Figura 3.6 – Bloco de programação para a parametrização do sensor de corrente

```
  #define CORRENTE_CAL 5.40 //VALOR DE CALIBRAÇÃO (DEVE SER
AJUSTADO EM PARALELO COM UM MULTÍMETRO MEDINDO A CORRENTE DA
CARGA)

  const int pinoSensor = A2; //PINO ANALÓGICO EM QUE O SENSOR
ESTÁ CONECTADO

  float ruido = 0.20; //RUÍDO PRODUZIDO NA SAÍDA DO SENSOR
(DEVE SER AJUSTADO COM A CARGA DESLIGADA APÓS CARREGAMENTO DO
CÓDIGO NO ARDUINO

  EnergyMonitor emon1; //CRIA UMA INSTÂNCIA

  emon1.current(pinoSensor, CORRENTE_CAL); //PASSA PARA A
FUNÇÃO OS PARÂMETROS (PINO ANALÓGIO / VALOR DE CALIBRAÇÃO)

  emon1.calcVI(20,100); //FUNÇÃO DE CÁLCULO (20 SEMICICLOS /
TEMPO LIMITE PARA FAZER A MEDIÇÃO)
  CorrenteAtual = emon1.Irms; //VARIÁVEL RECEBE O VALOR DE
CORRENTE RMS OBTIDO
  CorrenteAtual = CorrenteAtual-ruido; //VARIÁVEL RECEBE O
VALOR RESULTANTE DA CORRENTE RMS MENOS O RUÍDO

  if(CorrenteAtual < 0){ //SE O VALOR DA VARIÁVEL FOR MENOR
QUE 0, FAZ
      CorrenteAtual = 0; //VARIÁVEL RECEBE 0
```

Fonte: Autoria própria, 2019

O sensor de vibração SW-420 tem a finalidade de detectar as vibrações existentes na giga de testes, ocasionadas por componentes que não estão trabalhando em condições ideais. A sensibilidade do componente é definida por um trimpot, que possibilita o ajuste da precisão dos valores de vibração obtidos. No código de programação, são definidos os pinos que são utilizados como entrada e saída pelo sensor, além dos parâmetros padrões utilizados em comparação com os valores medidos. A Figura 3.7 mostra o bloco de programação para a parametrização do sensor de vibração.

Figura 3.7 – Bloco de programação para a parametrização do sensor de vibração

```
  for (;;)
  {
   TempoInicioA=millis();
   while(millis()-TempoInicioA<=1000){//enquanto a diferença
entre o tempo atual e o tempo de início for menor que
1000ms/1s ele continua dentro do loop para leitura

     if(digitalRead(pinoSensorA) == HIGH)// lê a porta
digital 7 e verfica a se o valor é alto
      {
     acumuladorA=acumuladorA+1;
      }

     if(digitalRead(pinoSensorB) == HIGH)// lê a porta
digital 7 e verfica a se o valor é alto
      {
     acumuladorB=acumuladorB+1;
      }

    }//fecha o while

   vibracaoA=acumuladorA;
   vibracaoB=acumuladorB;
   acumuladorA=0;
   acumuladorB=0;
```

Fonte: Autoria própria, 2019

A programação completa encontra-se no Apendice A.

3.3 Desenvolvimento do algoritmo de manutenção preditiva

As manutenções preventivas e preditivas têm o mesmo objetivo: evitar que o equipamento fique parado por quebra de componentes, necessitando da corretiva. A preventiva, indica a troca do componente mediante o tempo de uso informado pelo fabricante, diferente da preditiva, que analisa dados através de sensores e equipamentos especializados que monitoram a máquina, possibilitando a real avaliação das condições da peça. Este tipo de ação permite que a vida útil do componente seja mais bem aproveitada, desta forma, elimina-se o desperdício com trocas desnecessárias e consequentemente a redução de custos para a empresa. O algoritmo tem o intuito de sugerir ao manutentor o melhor momento para a substituição do componente comprometido, permitindo que a parada do equipamento ocorra de forma planejada, minimizando os impactos causados na linha produtiva. Para que isso

ocorra, é necessário realizar uma coleta de dados, através de sensores instalados na giga de testes, no caso, foram utilizados 2 de vibração e 1 de corrente.

Os valores obtidos pelos sensores, são enviados pelo Arduino, via Bluetooth, para um dispositivo Android, onde é instalado o aplicativo de manutenção preditiva. O software é construído através da plataforma IDE Eclipse, utilizando as bibliotecas de integração JAVA / Android. A biblioteca Arduino.*Bluetooth*, é responsável pela ativação, conexão e leitura dos dados enviados de forma serial via *Bluetooth*. Para o armazenamento das leituras, desenvolve-se um banco de dados SQLite, interno ao dispositivo móvel. Todos os valores coletados são persistentes, assim como os cálculos realizados.

Com os dados coletados, aplica-se o método de similaridade, também conhecido como técnica de "*nearest-neighbor*". Essa técnica consiste em calcular a distância entre cada dado de um conjunto de valores como (p1, p2, ..., pn) com um segundo grupo de valores (q1, q2, ..., qn). No projeto, “p1” e “q1” representam a leitura anterior e a atual de um sensor, respectivamente, assim como “p2” e “q2” de outro sensor e assim sucessivamente. A distância entre esses dois pontos do conjunto pode ser calculada utilizando o método da distância euclidiana, apresentada na Fundamentação teórica, conforme equação 3.1

$$d(P,Q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$$

Equação [3.1]

O cálculo é realizado internamente no sistema e seus resultados numéricos não são explícitos ao usuário. Essa distância representa a similaridade da leitura dos sensores com o valor ideal, este por sua vez, é definido pelo algoritmo realizando a média das primeiras 100 amostras coletadas. Portanto, quanto mais próximo a 0 for o resultado da equação da distância, maior é a semelhança.

Para obter as amostras iniciais, é necessário que o equipamento opere em sua condição considerada ideal. Com as 100 amostras, observa-se a amplitude máxima dos valores medidos referentes a cada sensor, e calcula-se o maior coeficiente

euclidiano que será admitido pelo sistema, este coeficiente é utilizado para limitar o cálculo de tendência de regressão linear simples.

Após definidos os valores ideais, o sistema continua coletando dados a cada 2 segundos e comparando-os com o padrão. Este valor de distância euclidiana é aplicado efetivamente no cálculo de tendência de regressão linear simples. Para simular uma falha, os parafusos que travam o mancal da máquina são soltos progressivamente. Os 10 primeiros valores obtidos durante esse processo de coleta de dados foram transcritos conforme Tabela 3.1.

Tabela 3.1 – Dez valores obtidos durante o processo de coleta de dados

Leitura	Sensor vibração 1	Sensor vibração 2	Sensor de corrente (A)	Coeficiente Euclidiano
1	4	0	1,516	11,1803
2	4	0	1,413	103
3	37	0	1,472	67,6018
4	15	0	1,485	25,5539
5	15	0	1,516	31
6	0	0	1,417	100,13
7	0	0	1,412	5
8	13	0	1,331	82,0366
9	13	0	1,378	47
10	13	0	1,360	18

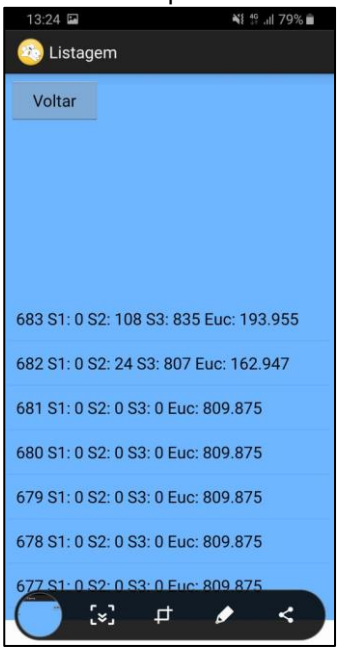
Fonte: Autoria própria, 2019

Quanto mais frouxos estão os parafusos, maior a trepidação lida pelo sensor de vibração posicionado nesse mancal, porém, quando são novamente apertados, os valores voltam a se estabilizar. O código é capaz de desprezar variações repentinas que não representam uma possível falha. Caso o sistema detecte uma aproximação aos valores máximos, de forma tendenciosa, é emitido um alerta para o manutentor, indicando qual dos sensores está detectando a anomalia.

O aplicativo é dividido em duas telas, a primeira possui um comparativo dos valores lidos entre cada um dos sensores, além de um botão exclusivo para que o operador possa redefinir os valores considerados ideais pelo sistema, iniciando a coleta de outras 100 amostras de dados. Esta ação se faz necessária mediante o desgaste natural dos componentes, permitindo que o manutencista identifique a

variação dos valores padrões após um período de tempo. A Figura 3.8 ilustra a interface da primeira tela do aplicativo.

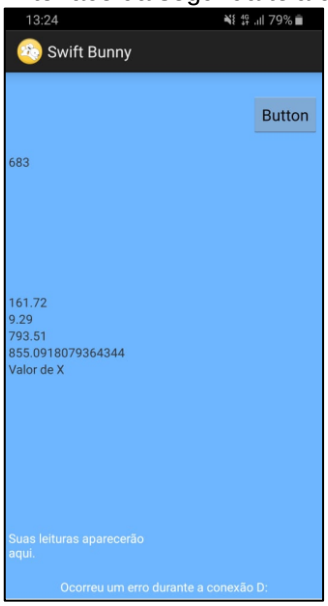
Figura 3.8 – Interface da primeira tela do aplicativo



Fonte: Autoria própria, 2019

A segunda tela do aplicativo é responsável por exibir ao manutencista os valores padrões identificados pelo algoritmo e também por exibir o alerta de falha do equipamento, indicando qual dos sensores está lendo os valores fora dos padrões estabelecido, indicando a necessidade de verificação das condições de trabalho do equipamento e se necessário uma troca do mesmo. A Figura 3.9 apresenta a interface da segunda tela do aplicativo.

Figura 3.9 – Interface da segunda tela do aplicativo



Fonte: Autoria própria, 2019

A programação completa do aplicativo Android encontra-se no Apendice B.

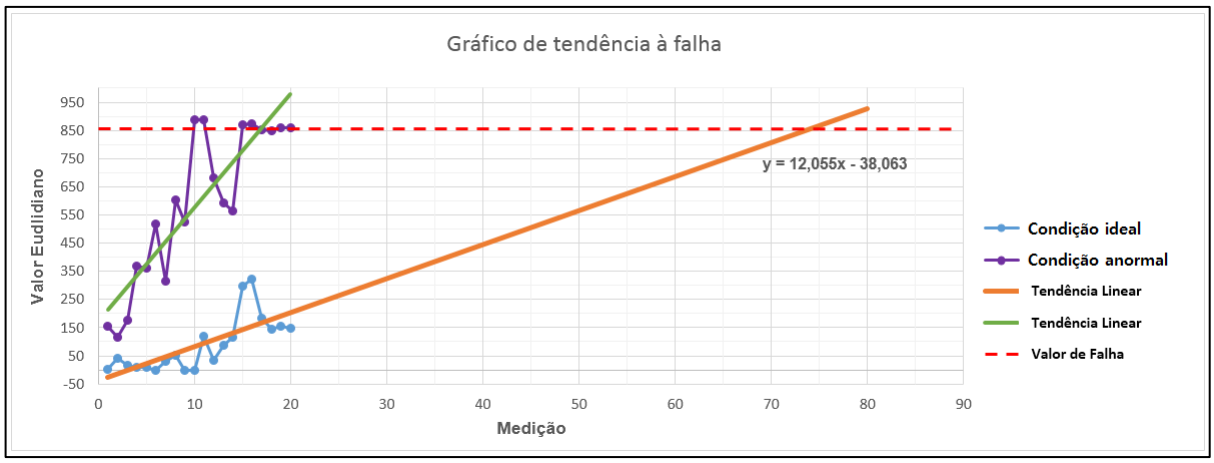
3.4 Resultados obtidos

Depois de concluídas as etapas de montagem mecânica da giga de testes, desenvolvimento do programa para a placa Arduino e desenvolvimento do algoritmo de manutenção preditiva, pode-se avaliar os resultados obtidos, validando sua funcionalidade. Para isso, divide-se a análise em duas partes, comparando primeiramente valores de corrente e posteriormente de vibração. Para simular uma situação de aumento de corrente, o freio mecânico é tensionado, provocando uma oposição ao funcionamento natural do motor. Desta forma, é realizado um comparativo entre os valores em estado ideal e com o freio atuante.

Para simular a variação dos sinais de vibrações, são medidos inicialmente os valores com a máquina em perfeito estado de funcionamento. Posteriormente, os parafusos dos mancais que sustentam o eixo são soltos parcialmente de forma a gerar trepidações excessivas no mesmo. O comparativo é realizado entre os valores de vibrações em estado normal de funcionamento do sistema e quando está trabalhando em situações anormais, como o afrouxamento dos parafusos do mancal.

Os valores ideais definidos pelo algoritmo, são aplicados na fórmula de distância euclidiana. O coeficiente resultante de cada ciclo de leitura é aplicado no cálculo de tendência de regressão linear. O mesmo procedimento é realizado para os valores em condições anormais de funcionamento. Sendo assim, obtém-se o comparativo de ambas as situações, destacando o momento em que o sistema emite o alerta para o manutentor, conforme ilustra o Gráfico 3.1.

Gráfico 3.1 – Comparativo entre os valores em situação normal e anormal de funcionamento



Fonte: Autoria própria, 2019

Os alertas são divididos em 3 categorias, sendo a primeira delas um aviso para que a manutenção seja agendada, a segunda exige a manutenção imediata do sistema pois o risco de quebra é eminente e a terceira indica que o sistema já ultrapassou seus limites de funcionamento, exigindo uma manutenção corretiva. Portanto, após a realização dos testes é possível dizer que o objetivo proposto foi atingido, pois o algoritmo é capaz de identificar situações anormais de funcionamento da máquina e informar os responsáveis, permitindo que ações sejam realizadas, inibindo os danos causados no processo.

3.5 Obstáculos e soluções

Durante as etapas de desenvolvimento e construção do projeto deparou-se com alguns obstáculos, que foram solucionados através de conhecimentos adquiridos nas pesquisas, experiência profissional dos integrantes do grupo e consulta a professores.

Obstáculo 1: fez-se necessária a troca do CLP integrado ao OPC em virtude da inviabilidade econômica e dificuldades em encontrar componentes. O uso do CLP e do OPC é interessante do ponto de vista industrial e sua aplicação futura no projeto não está descartada.

Solução: tal mudança ocasionou a troca pelo *software*/*hardware* Arduino, pois o mesmo manteve a mesma premissa do projeto inicial.

Obstáculo 2: durante os testes de funcionamento do motor com o eixo e a polia acoplada, notou-se a necessidade de reduzir a rotação do mesmo, que se encontrava ligado em triângulo e causava vibrações excessivas em toda a giga, fato que impedia a captação de dados pelos sensores.

Solução: incorporou-se à giga de testes, um inversor de frequência modelo VFD-L, que possibilita a redução da frequência de alimentação que chega até o motor, consequentemente permite a redução das rotações do mesmo.

Obstáculo 3: mesmo com a redução das rotações do motor, fez-se necessário estabilizar a giga de testes, pois quando era ligada sob uma superfície irregular, gerava valores inconstantes nos sensores de vibrações. Tal situação interfere diretamente na programação do *software* de inteligência artificial utilizado no projeto.

Solução: implementou-se uma base de aço à giga de testes, permitindo a estabilização da mesma e consequentemente a obtenção de medidas de vibrações mais regulares.

CONSIDERAÇÕES FINAIS

O trabalho de Análise de Dados em Máquinas Industriais com Eixos Rotativos tem como objetivo coletar informações de sensores inerentes ao processo, analisar e tomar decisões baseadas em um algoritmo de aprendizado de máquina para um sistema não supervisionado.

O Arduino recebe os dados coletados pelos sensores instalados na giga de testes, processando-os através do algoritmo de manutenção preditiva, capaz de identificar tendências de quebra dos componentes mediante valores obtidos que estão fora da condição considerada ideal.

As teorias pesquisadas foram de suma importância para os integrantes do grupo conhecerem melhor o funcionamento de aplicações de análise de dados na área da automação industrial. Essas teorias deram sustentação ao desenvolvimento do projeto, principalmente aquelas que dizem respeito as revoluções industriais, em específico os conceitos da indústria 4.0, como o de análise de dados e aprendizado de máquina não supervisionado. As análises feitas utilizando tais conceitos, resultam em ações que permitem uma maior disponibilidade dos equipamentos.

Os métodos e técnicas obtidas pela metodologia científica deram suporte para organizar e planejar as etapas que direcionam as diretrizes para o desenvolvimento do projeto, no tocante a organização, direcionamento e suporte para a concretização do objetivo proposto.

O objetivo encontra-se alcançado e comprova-se através dos resultados obtidos com simulações de falha dos componentes como aumento dos valores de corrente e vibrações excessivas no sistema. Analisando os dados, o algoritmo é capaz de notificar o manutentor através do aplicativo instalado em seu celular, indicando o risco eminente de quebra.

Como vantagem destaca-se o monitoramento, em tempo real, do equipamento através do aplicativo Android e a possibilidade de planejar ações de reparo e troca de

componentes danificados. Por consequência desses fatores, há o aumento da disponibilidade da máquina e a redução de custos com manutenções corretivas. Como ponto forte destaca-se a possibilidade de instalação do sistema de análise de dados em máquinas já existentes em um processo produtivo.

É importante ressaltar que, o sistema de análise de dados não monitora todas as variáveis presentes no dispositivo, podendo ocorrer a indisponibilidade do mesmo por fatores externos aos analisados neste projeto, fato que pode ser destacado como um ponto negativo.

Para melhorias em futuros trabalhos, há sugestões como o sistema passar a analisar mais variáveis de um equipamento, como as oscilações das rotações e mudanças de temperatura no motor, para isso, basta o acréscimo dos sensores correspondentes e a inclusão dos mesmos na lógica de programação. Além disso, pode ser utilizado o CLP e o OPC em substituição ao Arduino, dando mais robustez ao processo, principalmente em ambientes industriais.

REFERÊNCIAS

AZEVEDO, Hugo César Diniz. **Controlador lógico programável aplicado à Indústria 4.0.** 2018. 50 f. Dissertação de Mestrado - Curso de Programa de Pós-graduação em Ciência, Tecnologia e Inovação, Universidade Federal do Rio Grande do Norte, Natal: UFRGN, 2018.

BARROS, Pedro. **Aprendizagem de máquina:** supervisionada ou não supervisionada?. 2016. disponível em: <https://medium.com/opensanca/aprendizagem-de-maquina-supervisionada-ou-n%c3%a3o-supervisionada-7d01f78cd80a>. Acesso em: 13 mai. 2019.

COELHO, Pedro Miguel Nogueira. **Rumo à indústria 4.0.** 2016. 62 f. Dissertação de Mestrado - Curso de Engenharia Mecânica. Departamento de Engenharia Mecânica da Faculdade de Ciências e Tecnologia Universidade de Coimbra. Coimbra: UC, 2016.

DIAS JÚNIOR, Eugênio Passeli Ferreira. **Aprendizado por reforço sobre o problema de revisitação de páginas web.** 2012. 73 f. Dissertação de Mestrado - Curso de Informática. Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro. Rio de Janeiro: PUC, 2012.

FERRARI, Alfonso Trujillo. **Metodologia da ciência**. 3. ed. Rio de Janeiro: Kennedy,1974.

FREITAS, Arnold de Araujo. **A internet das coisas e seus efeitos na indústria 4.0.** 2017. 57 f. TCC (Graduação). Curso de Tecnologia em Sistemas, Centro Técnológico, Universidade Federal Fluminense, Niterói: UFF,2017.

GONÇALVES, Cassiano. **Estudo entre fabricantes de controladores lógico programáveis para uso em aplicações industriais.** 71 f. Cap. 2. Monografia - Curso de Especialização em Automação Industrial. Universidade Tecnológica Federal do Paraná. Curitiba: UTFP, 2009.

HONDA, Hugo; FACURE, Matheus; YAOHAO, Peng. **Os três tipos de aprendizado de máquina.** 2017. Disponível em: <https://lamfo-unb.github.io/2017/07/27/tres-tipos-am/>. Acesso em: 27 jul. 2017.

HURWITZ, Judith; KIRSCH, Daniel. **Machine learning for dummies.** Hoboken: John Wiley & Sons, Inc., 2018. 75 p.

INTERNACIONAL DATA CORPORATION – IDC. 2017. **El amanecer digital:** actual reto para las empresas de México y Latinoamérica, Disponível em: <http://br.idclatin.com/releases/news.aspx?id=2132>. Acesso em: 24 mar. 2019.

IWAI, Tomohiko. **Como conquistar mais clientes com aplicativos mobile.** Tihost, 2016. 7 p.

LI, Hui. **Machine Learning:** O que é e qual sua importância?. 2019. Disponível em: <https://www.sas.com/pt_br/insights/analytics/machine-learning.html#machine-learning-workingsOperaStable\Shell\Open\Command>. Acesso em: 28 mar. 2019.

MCROBERTS, Michael. **Arduino Básico.** São Paulo: Novatec, 2011. 453 p.

MANUAL DE NORMALIZAÇÃO DE PROJETO DE TRABALHO DE GRADUAÇÃO – FATEC SBCAMPO**. Material didático para utilização nos projetos de trabalho de graduação, dos cursos de Tecnologia em Automação Industrial e Informática para Negócios**. São Bernardo do Campo: Fatec, 2017.

NONNENMACHER, Renata Favretto. **Estudo do comportamento do consumidor de aplicativos móveis.** 2012. 70 f. TCC (Graduação). Curso de Administração, Ciências Administrativas. Universidade Federal do Rio Grande do Sul, Porto Alegre: UFRGS, 2012.

NOVUS. 2017. Disponível em: < https://www.novus.com.br/site/default.asp?TroncoID=053663&SecaoID=273506&SubsecaoID=0&Template=../artigosnoticias/user_exibir.asp&ID=934427&Idioma=55>. Acesso em: 12 abr. 2019.

NUSSEY, John. **Arduino: Para Leigos.** 2. ed. Rio de Janeiro: Alta Books, 2019. 400 p.

PETCOV, Rodrigo. **Aplicativos em nuvem.** São Paulo: Senac, 2018. 110 p.

PIERSON, Lillian. **Data Science para leigos.** 2. ed. Orlando: Alta Books, 2019.

PRODANOV, Cleber Cristiano ; FREITAS, Ernani Cesar de. **Metodologia do trabalho científico :** Métodos e técnicas da pesquisa do trabalho acadêmico**.** 2. ed. Nova Hamburgo: Universidade Feevale, 2013. 276 p.

PONCE, Rodrigo Villanueva; ALCARAZ, Jorge Luis García. **Evaluación de tecnología utilizando topsis en presencia de multi-colinealidad en atributos: ¿por qué usar distancia de mahalanobis?** 2013. Disponível em: <http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=s0120-62302013000200003&lng=en&nrm=iso>. Acesso em: 23 maio 2019.

ROMANO, Matheus. **Guia definitivo:** entenda tudo sobre o que é OPC UA e como isso pode impactar a sua indústria. 2017. Disponível em: <www.logiquesistemas.com.br/blog/opc-ua/>. Acesso em: 20 mar. 2019.

SILVA, Josenildo Costa da. **Algoritmos de aprendizagem de máquina:** qual deles escolher?: Um guia rápido sobre vários algoritmos em diferentes tarefas de aprendizagem. 2018. Disponível em: <https://medium.com/machina-sapiens/algoritmos-de-aprendizagem-de-m%C3%A1quina-qual-deles-escolher-67040ad68737>. Acesso em: 13 mai. 2019.

STURARO, André Zelioli. **Automação de bombas dosadoras com auxílio de controlador lógico programável.** 58 f. Cap. 2. Monografia - Curso de Engenharia Mecânica – Automação e Sistemas. Universidade São Francisco. Campinas: USF, 2009.

VENTURELLI, Márcio. **OPC UA na automação industrial**. 2018. Disponível em: <https://www.automacaoindustrial.info/opc-ua-na-automacao-industrial/>. Acesso em: 20 mar. 2019.

APENDICE A – CÓDIGO DE PROGRAMAÇÃO DO ARDUINO UNO

```
#include<Arduino_FreeRTOS.h>
#include<EmonLib.h>
voidGerenciaValor(void*pvParameters);
voidTaskReadVibraA(void*pvParameters);
voidTaskReadRPM(void*pvParameters);
voidTaskReadCorrente(void*pvParameters);
unsignedlongvibracaoA;
unsignedlongvibracaoB;
unsignedlongCorrenteAtual;
unsignedintrpm;
voidsetup(){
Serial.begin(9600);
xTaskCreate(
GerenciaValor
,(constportCHAR*)
,128
,1);
xTaskCreate(
TaskReadVibraA
,(constportCHAR*)
,128
,1);
xTaskCreate(
TaskReadRPM
,(constportCHAR*)
,128
,1);
xTaskCreate(
TaskReadCorrente
,(constportCHAR*)
,128
,1);
}
voidloop()
voidGerenciaValor(void*pvParameters)
{
(void)pvParameters;
longLastMillis;
longVarInicio=9999999;
intTempoAmostragem=5000;
intTIntervalo=133;//umtickdelaysão(15ms),ousejaTInterval
ox15=5010ms
for(;;)//Loopinfinitoutilizando'for(;;)'
{
vTaskDelay(TIntervalo/2);
Serial.print(VarInicio);Serial.println("");
vTaskDelay(TIntervalo/2);
Serial.print(vibracaoA);Serial.println("");
vTaskDelay(TIntervalo);
Serial.print(vibracaoB);Serial.println("");
vTaskDelay(TIntervalo);
Serial.print(CorrenteAtual);Serial.println("");
Serial.println();
}
}
voidTaskReadVibraA(void*pvParameters)
{
(void)pvParameters;
#definepinoSensorA2
#definepinoSensorB4
pinMode(pinoSensorA,INPUT);
pinMode(pinoSensorB,INPUT);
unsignedlongTempoInicioA;
unsignedlongTempoInicioB;
unsignedlongacumuladorA=0;
unsignedlongacumuladorB=0;
for(;;)
{
TempoInicioA=millis();
while(millis()-
TempoInicioA<=1000){//enquantoadiferençaentreotempoat
ualeotempodeinícioformenorque1000ms/1selecontinuaden
trodoloopparaleitura
if(digitalRead(pinoSensorA)==HIGH)//lêaportadigitaleverfic
aaseovaloréalto
{
acumuladorA=acumuladorA+1;
}
}
vibracaoA=acumuladorA;
acumuladorA=0;
TempoInicioB=millis();
while(millis()-
TempoInicioB<=1000){//enquantoadiferençaentreotempoat
ualeotempodeinícioformenorque1000ms/1selecontinuaden
trodoloopparaleitura
if(digitalRead(pinoSensorB)==HIGH)//lêaportadigitaleverfic
aaseovaloréalto
{
acumuladorB=acumuladorB+1;
}
}
vibracaoB=acumuladorB;
acumuladorB=0;
}
}
voidTaskReadCorrente(void*pvParameters)//Thisisatask.
{
(void)pvParameters;
#defineCORRENTE_CAL5.00
constintpinoSensor=A2;
doubleCorrenteTemp;
floatruido=0.20;
EnergyMonitoremon1;
emon1.current(pinoSensor,CORRENTE_CAL);
for(;;)//Loopinfinitoutilizando'for(;;)'
{
emon1.calcVI(20,100);
CorrenteTemp=emon1.Irms;
CorrenteTemp=CorrenteTemp-ruido;
if(CorrenteTemp<0){
CorrenteTemp=0;
}
CorrenteTemp=CorrenteTemp*1000;
CorrenteAtual=CorrenteTemp;
CorrenteTemp=0;
}
}
voidTaskReadRPM(void*pvParameters)
{
(void)pvParameters;
intpino_D0=7;
pinMode(pino_D0,INPUT);
unsignedlongtimeold;
unsignedlongTempoInicioC;
unsignedlongAcumuladorC;
unsignedintpulsos_por_volta=1;
timeold=0;
AcumuladorC=0;
for(;;)
{
TempoInicioC=millis();
while(millis()-TempoInicioC<=5000){
if(digitalRead(pino_D0)==HIGH)
{
AcumuladorC++;
while(digitalRead(pino_D0)==HIGH){}
}
}
rpm=5000*AcumuladorC/416;
timeold=millis();
AcumuladorC=0;}
```

APENDICE B – CÓDIGO DE PROGRAMAÇÃO DO APLICATIVO ANDROID

```
package br.com.kauecolombo.swiftbunny;

import android.app.activity;
import android.bluetooth.bluetoothadapter;
import android.content.context;
import android.content.intent;
import android.os.bundle;
import android.os.handler;
import android.os.message;
import android.text.editable;
import android.text.textwatcher;
import android.view.menuitem;
import android.view.view;
import android.widget.button;
import android.widget.edittext;
import android.widget.textview;
import
br.com.kauecolombo.swiftbunny.contextodados.contatosc
ursor;
import
br.com.kauecolombo.swiftbunny.contextopadroes.contatos
cursor2;

public class telalogin extends activity {

   /* definição dos objetos que serão usados na activity
principal
      statusmessage mostrará mensagens de status sobre
a conexão
      countermessage mostrará o valor do contador como
recebido do arduino
      connect é a thread de gerenciamento da conexão
bluetooth
    */
   static textview statusmessage;
   static textview countermessage, textview01, textview02,
textview03, textview04, textview05,  show;
   static textview mostra;
   static edittext edittext1;
   static button button1;
   bluetooth connect;
            int count = 1;
            double afericaosensor1;
            double afericaosensor2;
            double afericaosensor3;
            double afericaosensor11;
            double afericaosensor12;
            double afericaosensor13;
            double euclidianomax;
            double afericaogeral = 0.000000;
            double resto;
            double x1=0.000;
            double x2=0.000;
            int varredura = 0, sincronia = 0;
            long posicao, posicaopadrao;
            double padrao1 = 0.00, padrao2 = 0.00,
padrao3 = 0.00, padraogeral = 0.00;

   @override
   protected void oncreate(bundle savedinstancestate) {
      super.oncreate(savedinstancestate);
      setcontentview(r.layout.activity_tela_login);
      statusmessage = (textview)
findviewbyid(r.id.statusmessage);
      countermessage = (textview)
findviewbyid(r.id.countermessage);
      textview01 = (textview) findviewbyid(r.id.textview01);
      textview02 = (textview) findviewbyid(r.id.textview02);
      textview03 = (textview) findviewbyid(r.id.textview03);
      textview04 = (textview) findviewbyid(r.id.textview04);
      textview05 = (textview) findviewbyid(r.id.textview05);
      show = (textview) findviewbyid(r.id.show);


    mostra = (textview) findviewbyid(r.id.textview1);
      button1 = (button) findviewbyid(r.id.button1);

      contextodados db = new contextodados(this);
                  mostra.settext(string.valueof(posicao
= db.ultimo().getlong(0)));
                  db.close();

                  if (posicao >= 100) {
                        contextopadroes db1 =
new contextopadroes(this);
                        contatoscursor2 cursor =
db1.retornarcontatos(contatoscursor2.ordenarpor.nomede
crescente);
            textview01.settext(cursor.getpadraosensor1().to
string());
            textview02.settext(cursor.getpadraosensor2().to
string());
            textview03.settext(cursor.getpadraosensor3().to
string());
            textview04.settext(cursor.getpadraogeral().tostri
ng());
                        }
                  button1.setonclicklistener(new
button.onclicklistener() {
                        @override
                        public void onclick(view v) {
                              intent sensores =
new intent(telalogin.this, listagem.class);

            startactivity(sensores);

            }
                  });
            countermessage.addtextchangedlistener(new
textwatcher()
            {
               @override
               public void ontextchanged(charsequence s,
int start, int before, int count)
               {

               }
               public void beforetextchanged(charsequence
s, int start, int count, int aft)
               {
               }

               @override
               public void aftertextchanged(editable s)
               {
                        if
(countermessage.gettext().length() != 0 &&
(countermessage.gettext().length())==7) {
                  sincronia = 1;
                        }
                        else if
(countermessage.gettext().length() != 0) {
                              if (posicao
<=100)
                              salvarcadastro();
                                    else

            salvarcadastro100();
                        }
               }
            });
```

```
    bluetoothadapter btadapter =
bluetoothadapter.getdefaultadapter();
    if (btadapter == null) {
        statusmessage.settext("que pena! hardware
bluetooth não está funcionando :(");
    } else {
        statusmessage.settext("ótimo! hardware bluetooth
está funcionando :)");
    }

    btadapter.enable();

    connect = new bluetooth("98:d3:32:30:f7:1f");
    connect.start();

    try {
        thread.sleep(1000);
    } catch (exception e) {
        e.printstacktrace();
    }
}

public boolean onoptionsitemselected(menuitem item) {
    int id = item.getitemid();

    //noinspection simplifiableifstatement
    if (id == r.id.action_settings) {
        return true;
    }

    return super.onoptionsitemselected(item);
}

public static handler handler = new handler() {

    public void handlemessage(message msg) {
        bundle bundle = msg.getdata();
        byte[] data = bundle.getbytearray("data");
        string datastring= new string(data);
        if(datastring.equals("---n"))
            statusmessage.settext("ocorreu um erro durante a
conexão d:");

            statusmessage.settext("conectado :d");
        else {

            /* se a mensagem não for um código de status,
               então ela deve ser tratada pelo aplicativo
               como uma mensagem vinda diretamente do
outro
               lado da conexão. nesse caso, simplesmente
               atualizamos o valor contido no textview do
               contador.
             */
            countermessage.settext(datastring);


        }

    }
};

    public void salvarcadastro()
    {
                        if (count == 1 && sincronia == 1) {
                        afericaosensor1 =
double.parsedouble(countermessage.gettext().tostring());
                        count = count+1;
                        show.settext("vibração 1:" +
afericaosensor1);}

              else if (count == 2) {
                        afericaosensor2 =
double.parsedouble(countermessage.gettext().tostring());
                        count = count+1;
                        show.settext("vibração 2:" +
afericaosensor2);}

            else if (count == 3 && varredura == 0 &&
countermessage.gettext().tostring() != "9999999") {
                        afericaosensor3 =
double.parsedouble(countermessage.gettext().tostring());
                        afericaogeral = 0.0;
                        show.settext("corrente:" +
afericaosensor3);

                                contextodados db = new
contextodados(this);
                                posicao =
db.inserircontato(afericaosensor1, afericaosensor2,
afericaosensor3, afericaogeral);
                                salvarpadrao(this);
                        db.close();
                        count = count+1;

                        }

            else if (count == 3 && varredura == 1) {
                        afericaosensor3 =
double.parsedouble(countermessage.gettext().tostring());
                        afericaogeral = (double)
math.sqrt(math.pow(

            (afericaosensor1 - afericaosensor11), 2)
                                                    +
math.pow((afericaosensor2 - afericaosensor12), 2)
                                                    +
math.pow((afericaosensor3 - afericaosensor13), 2));

                                contextodados db = new
contextodados(this);
                                posicao =
db.inserircontato(afericaosensor1, afericaosensor2,
afericaosensor3, afericaogeral);
                                salvarpadrao(this);
                        db.close();
                        count = count+1;
                        if (posicao>120)
                                tendencia(this);
                        listagem li = new listagem();
                        li.obtersensor1(this);
                        show.settext("corrente:" +
afericaosensor3);}

            else if (count == 4) {
            afericaosensor11 =
double.parsedouble(countermessage.gettext().tostring());
            count = count+1;
            show.settext("vibração 1:" + afericaosensor11);}

            else if (count == 5) {
            afericaosensor12 =
double.parsedouble(countermessage.gettext().tostring());
            count = count+1;
            show.settext("vibração 2:" + afericaosensor12);}

            else if (count == 6) {
            afericaosensor13 =
double.parsedouble(countermessage.gettext().tostring());
            show.settext("corrente:" + afericaosensor13);
            count = 1;
                        varredura =1;
                        afericaogeral = (double)
math.sqrt(math.pow(

            (afericaosensor11 - afericaosensor1), 2)
                                                    +
math.pow((afericaosensor12 - afericaosensor2), 2)
                                                    +
math.pow((afericaosensor13 - afericaosensor3), 2));

                                contextodados db = new
contextodados(this);
```

```
                                listagem li = new
listagem();
                        posicao =
db.inserircontato(afericaosensor11, afericaosensor12,
afericaosensor13, afericaogeral);
                        li.obtersensor1(this);
                        salvarpadrao(this);
                        db.close();
    }
    }

    public void salvarcadastro100()
    {
                        if (count == 1 && sincronia == 1) {
                        afericaosensor1 =
double.parsedouble(countermessage.gettext().tostring());
                        count = count+1;
                        show.settext("vibração 1:" +
afericaosensor1);}

                else if (count == 2) {
                        afericaosensor2 =
double.parsedouble(countermessage.gettext().tostring());
                        count = count+1;
                        show.settext("vibração 2:" +
afericaosensor2);}

                else if (count == 3) {
                        afericaosensor3 =
double.parsedouble(countermessage.gettext().tostring());
                        afericaogeral = (double)
math.sqrt(math.pow(

                (afericaosensor1 -
double.parsedouble(textview01.gettext().tostring())), 2)
                                                                +
math.pow((afericaosensor2 -
double.parsedouble(textview02.gettext().tostring())), 2)
                                                                +
math.pow((afericaosensor3 -
double.parsedouble(textview03.gettext().tostring())), 2));

                                contextodados db = new
contextodados(this);
                                posicao =
db.inserircontato(afericaosensor1, afericaosensor2,
afericaosensor3, afericaogeral);
                                salvarpadrao(this);
                        db.close();
                        count = 1;

                        if (posicao>=120)
                                tendencia(this);

                        //listagem li = new listagem();
                        //li.obtersensor1(this);
                        show.settext("corrente:" +
afericaosensor3);
                        count =1;}
    }


    public void tendencia(context c){
                contextodados db = new contextodados(c);
                contextopadroes db1 = new contextopadroes(c);

                double exiyi = 0.00000000;
                double sxy = 0.00000000;
                double sxx = 0.00000000;
                double beta =0.000000;
                double alfa = 0.000000;

                int i=1;
                int linha = 20;

                int n = 20;
                double yi= 0.00000;
                int xi = 210;
                double ximedia = 10.5;
                int xixi = 2870;

                contatoscursor cursor =
db.retornarcontatos(contatoscursor.ordenarpor.nomecresc
ente);
                        cursor.movetofirst();
                        do {
                                yi = cursor.getdouble(4) +
yi;
                                exiyi =
linha*(cursor.getdouble(4)) + exiyi;
                                i = i+1;
                                linha = linha-1;
                                cursor.movetonext();
                        } while (i<=20);

                        cursor.close();

                double yimedia = yi/20;
                sxy = exiyi - n*ximedia*yimedia;
                sxx = xixi - n* (math.pow((ximedia), 2));

                beta = sxy/sxx;
                alfa = yimedia - beta*ximedia;

                x1 = yimedia;

                contatoscursor2 cursor2 =
db1.retornarcontatos(contatoscursor2.ordenarpor.nomecre
scente);
                        cursor2.movetofirst();
                x1 = ((cursor2.getdouble(4)) - alfa) / beta;

                if (x1>=0.000) {

                        double ultimossensores1 = 0.0000;
                        double mediaultimossensores1 =
0.0000;
                        int i1 =1;

                contatoscursor cursorsensor1 =
db.retornarcontatos(contatoscursor.ordenarpor.nomecresc
ente);
                        cursorsensor1.movetofirst();
                        do {
                                ultimossensores1 =
cursorsensor1.getafericaosensor1() + ultimossensores1;

                cursorsensor1.movetonext();
                                i1=i1+1;
                        } while (i1<=20);

                        mediaultimossensores1 =
ultimossensores1/20;

                        double ultimossensores2 = 0.0000;
                        double mediaultimossensores2 =
0.0000;
                        int i2 =1;
                        contatoscursor cursorsensor2 =
db.retornarcontatos(contatoscursor.ordenarpor.nomecresc
ente);
                        cursorsensor2.movetofirst();
                        do {
                                ultimossensores2 =
cursorsensor2.getafericaosensor2() + ultimossensores2;

                cursorsensor2.movetonext();
                                i2=i2+1;
                        } while (i2<=20);

                        mediaultimossensores2 =
ultimossensores2/20;
```

```
                        double ultimossensores3 = 0.0000;
                        double mediaultimossensores3 =
0.0000;
                        int i3 =1;
                        contatoscursor cursorsensor3 =
db.retornarcontatos(contatoscursor.ordenarpor.nomecresc
ente);
                        cursorsensor3.movetofirst();
                        do {
                                ultimossensores3 =
cursorsensor3.getafericaosensor3() + ultimossensores3;

            cursorsensor3.movetonext();
                                i3=i3+1;
                        } while (i3<=20);

                        mediaultimossensores3 =
ultimossensores3/20;

                        double errosensor1 = 0.000;
                        double errosensor2 = 0.000;
                        double errosensor3 = 0.000;
                        contatoscursor2 cursorpadrao =
db1.retornarcontatos(contatoscursor2.ordenarpor.nomede
crescente);
                        cursorpadrao.movetofirst();

                        errosensor1 =
(mediaultimossensores1*100) /
cursorpadrao.getpadraosensor1();
                        errosensor2 =
(mediaultimossensores2*100) /
cursorpadrao.getpadraosensor2();
                        errosensor3 =
(mediaultimossensores3*100) /
cursorpadrao.getpadraosensor3();

                        if (errosensor1 >= 150)
                                textview05.settext("motor
esta vibrando demais! correção necessária!");

                        if (errosensor2 >= 150)

            textview05.settext("problema no mancal!
correção necessária!");

                        if (errosensor3 >= 150)

            textview05.settext("corrente excessiva! correção
necessária!");

                        textview05.settext("s1:
"+errosensor1.tostring() + "  s2:    " +errosensor2.tostring()
+ "  s3:    " +errosensor3.tostring());
                        //x1 = x1-20;
                        db1.close();
                        db.close();
                        }
            else
                        tendencia2(this);
    }

    public void tendencia2(context c){
            contextodados db = new contextodados(c);
            contextopadroes db1 = new contextopadroes(c);

            double exiyi = 0.00000000;
            double sxy = 0.00000000;
            double sxx = 0.00000000;
            double beta =0.000000;
            double alfa = 0.000000;

            int i=1;
            int linha = 20;
            int n = 20;
            double yi= 0.00000;
            int xi = 210;
            double ximedia = 10.5;
            int xixi = 2870;

            contatoscursor cursor =
db.retornarcontatos(contatoscursor.ordenarpor.nomecresc
ente);
                        cursor.movetofirst();
                        do {
                                yi = -(cursor.getdouble(4))
+ yi;
                                exiyi = linha*-
(cursor.getdouble(4)) + exiyi;
                                i = i+1;
                                linha = linha-1;
                                cursor.movetonext();
                        } while (i<=20);

                        cursor.close();

            double yimedia = yi/20;
            sxy = exiyi - n*ximedia*yimedia;
            sxx = xixi - n* (math.pow((ximedia), 2));

            beta = sxy/sxx;
            alfa = yimedia - beta*ximedia;

            x2 = yimedia;

            contatoscursor2 cursor2 =
db1.retornarcontatos(contatoscursor2.ordenarpor.nomecre
scente);
                        cursor2.movetofirst();
            x2 = ((cursor2.getdouble(4)) - alfa) / beta;
            x2 = x2-20;

            textview05.settext("x2: " + x2.tostring() + "x1: " +
x1.tostring());
    }

    public void salvarpadrao(context c)
    {
            contextodados db = new contextodados(c);
        contatoscursor cursor =
db.retornarcontatos(contatoscursor.ordenarpor.nomedecre
scente);
        contextopadroes db1 = new contextopadroes(c);

        for( int i=0; i <cursor.getcount(); i++)
        {
            cursor.movetoposition(i);
            imprimirlinha(cursor.getafericaosensor1(),
cursor.getafericaosensor2(), cursor.getafericaosensor3(),
cursor.getafericaogeral());
        }

           if (posicao==100) {
            padrao1 =
(double.parsedouble(db.med1().getstring(0)))/100;
            padrao2 =
(double.parsedouble(db.med2().getstring(0)))/100;
            padrao3 =
(double.parsedouble(db.med3().getstring(0)))/100;

            euclidianomax = (double) math.sqrt(math.pow(

            (db.min1().getdouble(0)) -
(db.max1().getdouble(0)), 2)
                                                +
math.pow((db.min2().getdouble(0)) -
(db.max2().getdouble(0)), 2)
```

```
                                                +
math.pow((db.min3().getdouble(0)) -
(db.max3().getdouble(0)), 2));


            posicaopadrao = db1.inserircontato(padrao1,
padrao2, padrao3, euclidianomax);

            textview01.settext(padrao1.tostring());
            textview02.settext(padrao2.tostring());
            textview03.settext(padrao3.tostring());
            textview04.settext(euclidianomax.tostring());

      }
   }

   public void imprimirlinha(double afericaosensor1, double
afericaosensor2, double afericaosensor3, double
afericaogeral)
   {


            if(countermessage.gettext().tostring().equalsign
orecase("nenhum contato cadastrado."))
                        countermessage.settext("");
   }
   }

contextodados
package br.com.kauecolombo.swiftbunny;

import android.content.contentvalues;
import android.content.context;
import android.database.cursor;
import android.database.sqlexception;
import android.database.sqlite.sqlitecursor;
import android.database.sqlite.sqlitecursordriver;
import android.database.sqlite.sqlitedatabase;
import android.database.sqlite.sqliteopenhelper;
import android.database.sqlite.sqlitequery;
import android.util.log;

public class contextodados extends sqliteopenhelper {

            private static final string nome_bd = "afericao";
            private static final int versao_bd = 1;
            private static final string log_tag = "afericao";
            private final context contexto;

            public contextodados(context context) {
                        super(context, nome_bd, null,
versao_bd);
                        this.contexto = context;
                        }

            @override
            public void oncreate(sqlitedatabase db)
            {
                        string[] sql =
contexto.getstring(r.string.contextodados_oncreate).split("\
n");
                        db.begintransaction();

                        try
                        {
                                    // cria a tabela e testa os
dados
                                    executarcomandossql(db,
sql);

            db.settransactionsuccessful();
                        }
                        catch (sqlexception e)
                        {
                                    log.e("erro ao criar as
tabelas e testar os dados", e.tostring());
                        }
                        finally
                        {
                                    db.endtransaction();
                        }
            }

            @override
            public void onupgrade(sqlitedatabase db, int
oldversion, int newversion)
            {
                        log.w(log_tag, "atualizando a base de
dados da versã£o " + oldversion + " para " + newversion +
", que destruirã¡ todos os dados antigos");
                        string[] sql =
contexto.getstring(r.string.contextodados_onupgrade).split
("\n");
                        db.begintransaction();

                        try
                        {
                                    executarcomandossql(db,
sql);

            db.settransactionsuccessful();
                        }
                        catch (sqlexception e)
                        {
                                    log.e("erro ao atualizar as
tabelas e testar os dados", e.tostring());
                                    throw e;
                        }
                        finally
                        {
                                    db.endtransaction();
                        }
                        oncreate(db);
            }

            private void
executarcomandossql(sqlitedatabase db, string[] sql)
            {
                        for( string s : sql )
                                    if (s.trim().length()>0)

                                                db.execsql(s);
            }

            public contatoscursor
retornarcontatos(contatoscursor.ordenarpor ordenarpor)
            {
                        string sql = contatoscursor.consulta +
(ordenarpor ==
contatoscursor.ordenarpor.nomedecrescente ? "asc" :
"desc");
                        sqlitedatabase bd =
getreadabledatabase();
                        contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
                        cc.movetofirst();
                        return cc;
            }

            public contatoscursor max1()
            {
                        string sql =
contatoscursor.consultamax1;
                        sqlitedatabase bd =
getreadabledatabase();
                        contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
                        cc.movetolast();
                        return cc;
            }
```

```
	public contatoscursor min1()
	{
		string sql =
contatoscursor.consultamin1;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor max2()
	{
		string sql =
contatoscursor.consultamax2;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor min2()
	{
		string sql =
contatoscursor.consultamin2;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor max3()
	{
		string sql =
contatoscursor.consultamax3;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor min3()
	{
		string sql =
contatoscursor.consultamin3;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor ultimo()
	{
		string sql =
contatoscursor.consultaultimo;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor med1()
	{
		string sql =
contatoscursor.consultamed1;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor med2()
	{
		string sql =
contatoscursor.consultamed2;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor med3()
	{
		string sql =
contatoscursor.consultamed3;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}

	public contatoscursor med4()
	{
		string sql =
contatoscursor.consultamed3;
		sqlitedatabase bd =
getreadabledatabase();
		contatoscursor cc = (contatoscursor)
bd.rawquerywithfactory(new contatoscursor.factory(), sql,
null, null);
		cc.movetolast();
		return cc;
	}


	public long inserircontato(double
afericaosensor1, double afericaosensor2, double
afericaosensor3, double afericaogeral)
	{
		sqlitedatabase db =
getwritabledatabase();

		try
		{
			contentvalues initialvalues
= new contentvalues();

	initialvalues.put("afericaosensor1",
afericaosensor1);

	initialvalues.put("afericaosensor2",
afericaosensor2);

	initialvalues.put("afericaosensor3",
afericaosensor3);
```

```
		initialvalues.put("afericaogeral", afericaogeral);
				return db.insert("afericao",
null, initialvalues);
			}
			finally
			{
				db.close();
			}
		}


		public static class contatoscursor extends
sqlitecursor
		{
			public static enum ordenarpor{
				nomecrescente,
				nomedecrescente
			}

			private static final string consulta =
"select * from afericao order by id ";

			private static final string
consultamax1 = "select max (afericaosensor1) from
afericao";

			private static final string consultamin1
= "select min (afericaosensor1) from afericao";

			private static final string
consultamax2 = "select max (afericaosensor2) from
afericao";

			private static final string consultamin2
= "select min (afericaosensor2) from afericao";

			private static final string
consultamax3 = "select max (afericaosensor3) from
afericao";

			private static final string consultamin3
= "select min (afericaosensor3) from afericao";

			private static final string
consultamed1 = "select sum (afericaosensor1) from
afericao";

			private static final string
consultamed2 = "select sum (afericaosensor2) from
afericao";

			private static final string
consultamed3 = "select sum (afericaosensor3) from
afericao";

			private static final string
consultaultimo2 = "select from afericao where id = (select
max(id) from afericao)";

			private static final string
consultaultimo = "select max (id) from afericao";


			private contatoscursor(sqlitedatabase
db, sqlitecursordriver driver, string edittable, sqlitequery
query)
			{
				super(db, driver, edittable,
query);
			}

			private static class factory implements
sqlitedatabase.cursorfactory
			{
				@override
					public cursor
newcursor(sqlitedatabase db, sqlitecursordriver driver,
string edittable, sqlitequery query)
					{
						return new
contatoscursor(db, driver, edittable, query);
					}
			}

			public double getafericaosensor1()
			{
				return
getdouble(getcolumnindexorthrow("afericaosensor1"));
			}

			public double getafericaosensor2()
			{
				return
getdouble(getcolumnindexorthrow("afericaosensor2"));
			}

			public double getafericaosensor3()
			{
				return
getdouble(getcolumnindexorthrow("afericaosensor3"));
			}

			public double getafericaogeral()
			{
				return
getdouble(getcolumnindexorthrow("afericaogeral"));
			}
		}
}
listagem
package br.com.kauecolombo.swiftbunny;
import java.util.arraylist;
import java.util.list;
import android.app.activity;
import android.content.context;
import android.content.intent;
import android.os.bundle;
import android.view.view;
import android.widget.arrayadapter;
import android.widget.button;
import android.widget.listview;
import
br.com.kauecolombo.swiftbunny.contextodados.contatosc
ursor;
public class listagem extends activity{

	public static list<string> sensores1 = new
arraylist<string>();
	private listview valor1;
	static button button1;
protected void oncreate(bundle savedinstancestate) {
		super.oncreate(savedinstancestate);
		setcontentview(r.layout.valores_sensores);

			button1 = (button)
findviewbyid(r.id.btnvoltar);


	button1.setonclicklistener(new
button.onclicklistener() {

	@override
public
void onclick(view v) {

	intent voltar = new intent(listagem.this,
telalogin.class);

	startactivity(voltar);
}
	});
```

```
                valor1 = (listview)
findviewbyid(r.id.lvsensor1);
                        valor1.setadapter(new
arrayadapter<string>(this,

        android.r.layout.simple_list_item_1,
obtersensor1(this)));

}

            public list<string> obtersensor1(context c) {
                sensores1.clear();
                        contextodados db = new
contextodados(c);
                contatoscursor cursor =
db.retornarcontatos(contatoscursor.ordenarpor.nomecresc
ente);
                        cursor.movetofirst();
                        do {
                                string momento
= cursor.getstring(0) + " s1: " + cursor.getstring(1) + " s2: "
+ cursor.getstring(2) + " s3: "  + cursor.getstring(3) + " euc:
"  + cursor.getstring(4);

        sensores1.add(momento);
                        } while
(cursor.movetonext());
                        cursor.close();
                        db.close();
                        return sensores1;
                }
}
contexto padrões
package br.com.kauecolombo.swiftbunny;
import android.content.contentvalues;
import android.content.context;
import android.database.cursor;
import android.database.sqlexception;
import android.database.sqlite.sqlitecursor;
import android.database.sqlite.sqlitecursordriver;
import android.database.sqlite.sqlitedatabase;
import android.database.sqlite.sqliteopenhelper;
import android.database.sqlite.sqlitequery;
import android.util.log;
import
br.com.kauecolombo.swiftbunny.contextodados.contatosc
ursor;

public class contextopadroes extends sqliteopenhelper {
        private final context contexto;

        public contextopadroes(context context) {
                super(context, nome_bd, null,
versao_bd);
                this.contexto = context;
                }

        @override
        public void oncreate(sqlitedatabase db)
        {
                string[] sql =
contexto.getstring(r.string.contextopadroes_oncreate).split
("\n");
                db.begintransaction();

                try
                {
                        // cria a tabela e testa os
dados
                        executarcomandossql(db,
sql);

        db.settransactionsuccessful();
                }
                catch (sqlexception e)
                {
                        log.e("erro ao criar as
tabelas e testar os dados", e.tostring());
                }
                finally
                {
                        db.endtransaction();
                }
        }
        public void onupgrade(sqlitedatabase db, int
oldversion, int newversion)
        {
                log.w(log_tag, "atualizando a base de
dados da versã£o " + oldversion + " para " + newversion +
", que destruirã¡ todos os dados antigos");
                string[] sql =
contexto.getstring(r.string.contextopadroes_onupgrade).sp
lit("\n");
                db.begintransaction();

                try
                {
                        executarcomandossql(db,
sql);

        db.settransactionsuccessful();
                }
                catch (sqlexception e)
                {
                        log.e("erro ao atualizar as
tabelas e testar os dados", e.tostring());
                        throw e;
                }
                finally
                {
                        db.endtransaction();
                }
                oncreate(db);
        }

        {
                for( string s : sql )
                        if (s.trim().length()>0)

                                db.execsql(s);
        }

        /** retorna um contatoscursor ordenado
        * @param critã©rio de ordenaã§ã£o
        */
        public contatoscursor2
retornarcontatos(contatoscursor2.ordenarpor ordenarpor)
        {
                string sql = contatoscursor2.consulta
+ (ordenarpor ==
contatoscursor2.ordenarpor.nomecrescente ? "asc" :
"desc");
                sqlitedatabase bd =
getreadabledatabase();
                contatoscursor2 cc =
(contatoscursor2) bd.rawquerywithfactory(new
contatoscursor2.factory(), sql, null, null);
                cc.movetofirst();
                return cc;
        }
        public long inserircontato(double
padraosensor1, double padraosensor2, double
padraosensor3, double padraogeral)
        {
                sqlitedatabase db =
getreadabledatabase();

                try
                {
                        contentvalues initialvalues
= new contentvalues();
```

```
		initialvalues.put("padraosensor1",
padraosensor1);

		initialvalues.put("padraosensor2",
padraosensor2);

		initialvalues.put("padraosensor3",
padraosensor3);

		initialvalues.put("padraogeral", padraogeral);
				return db.insert("padroes",
null, initialvalues);
			}
			finally
			{
				db.close();
			}
		}

		public static class contatoscursor2 extends
sqlitecursor
		{
			public static enum ordenarpor{
				nomecrescente,
				nomedecrescente
			}

			private static final string consulta =
"select * from padroes order by id ";

			private
contatoscursor2(sqlitedatabase db, sqlitecursordriver
driver, string edittable, sqlitequery query)
			{
				super(db, driver, edittable,
query);
			}

			private static class factory implements
sqlitedatabase.cursorfactory
			{
				@override
				public cursor
newcursor(sqlitedatabase db, sqlitecursordriver driver,
string edittable, sqlitequery query)
				{
					return new
contatoscursor2(db, driver, edittable, query);
				}
			}

			public double getpadraosensor1()
			{
				return
getdouble(getcolumnindexorthrow("padraosensor1"));
			}

			public double getpadraosensor2()
			{
				return
getdouble(getcolumnindexorthrow("padraosensor2"));
			}

			public double getpadraosensor3()
			{
				return
getdouble(getcolumnindexorthrow("padraosensor3"));
			}

			public double getpadraogeral()
			{
				return
getdouble(getcolumnindexorthrow("padraogeral"));
			}
		}
}
bluetooth
package br.com.kauecolombo.swiftbunny;

import java.io.ioexception;
import java.io.inputstream;
import java.io.outputstream;
import java.util.arrays;
import java.util.uuid;

import android.bluetooth.bluetoothadapter;
import android.bluetooth.bluetoothdevice;
import android.bluetooth.bluetoothserversocket;
import android.bluetooth.bluetoothsocket;
import android.os.bundle;
import android.os.message;

public class bluetooth extends thread {

   bluetoothsocket btsocket = null;
   bluetoothserversocket btserversocket = null;
   inputstream input = null;
   outputstream output = null;
   string btdevaddress = null;
   string myuuid = "00001101-0000-1000-8000-
00805f9b34fb";
   boolean server;
   boolean running = false;
   public bluetooth() {

      this.server = true;
   }
   public bluetooth(string btdevaddress) {

      this.server = false;
      this.btdevaddress = btdevaddress;
   }
   public void run() {

      this.running = true;
      bluetoothadapter btadapter =
bluetoothadapter.getdefaultadapter();

      /*  determina que ações executar dependendo se a
thread está configurada
      para atuar como servidor ou cliente.
       */
      if(this.server) {
         try {

            /*  cria um socket de servidor bluetooth.
               permanece em estado de espera até que
algum cliente
            estabeleça uma conexão.
             */
            btserversocket =
btadapter.listenusingrfcommwithservicerecord("super
counter", uuid.fromstring(myuuid));
            btsocket = btserversocket.accept();

            if(btsocket != null) {

               btserversocket.close();
            }

         } catch (ioexception e) {

            e.printstacktrace();
            tomainactivity("---n".getbytes());
         }

      } else {
         try {
            bluetoothdevice btdevice =
btadapter.getremotedevice(btdevaddress);
```

```
            btsocket =
btdevice.createrfcommsockettoservicerecord(uuid.fromstri
ng(myuuid));

            if (btsocket != null) {
                btsocket.connect();
            }

        } catch (ioexception e) {

            e.printstacktrace();
            tomainactivity("---n".getbytes());
        }
     }

     if(btsocket != null) {

        this.isconnected = true;
        tomainactivity("---s".getbytes());
         try {

            /*  obtem referências para os fluxos de entrada e
saída do
            socket bluetooth.
             */
            input = btsocket.getinputstream();
            output = btsocket.getoutputstream();
            while(running) {
                byte[] buffer = new byte[1024];
                int bytes;
                int bytesread = -1;
                 do {
                    bytes = input.read(buffer, bytesread+1, 1);
                    bytesread+=bytes;
                } while(buffer[bytesread] != '\n');

                tomainactivity(arrays.copyofrange(buffer, 0,
bytesread-1));

            }

        } catch (ioexception e) {
            e.printstacktrace();
            tomainactivity("---n".getbytes());
            this.isconnected = false;
        }
     }

  }
  private void tomainactivity(byte[] data) {

     message message = new message();
     bundle bundle = new bundle();
     bundle.putbytearray("data", data);
     message.setdata(bundle);
     telalogin.handler.sendmessage(message);
  }
  public void write(byte[] data) {

     if(output != null) {
        try {

            /*  transmite a mensagem.
             */
            output.write(data);

        } catch (ioexception e) {
            e.printstacktrace();
        }
     } else {

        /*  envia à activity principal um código de erro
durante a conexão.
         */
        tomainactivity("---n".getbytes());
     }
  }
  public void cancel() {

     try {

        running = false;
        this.isconnected = false;
        btserversocket.close();
        btsocket.close();

     } catch (ioexception e) {
        e.printstacktrace();
     }
     running = false;
     this.isconnected = false;
  }

  public boolean isconnected() {
     return this.isconnected;
  }
}
```

montagem tela inicial

```
<relativelayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#ff6eb6ff"
   android:paddingbottom="4dp"
   android:paddingleft="4dp"
   android:paddingright="4dp"
   android:paddingtop="4dp"

   <textview
      android:id="@+id/countermessage"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_centerhorizontal="true"
      android:layout_centervertical="true"
      android:text=""

android:textappearance="?android:attr/textappearancelarg
e"
      android:textcolor="#ffffffff"
      android:textsize="14sp"
      android:visibility = "invisible" />

   <textview
      android:id="@+id/statusmessage"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignparentbottom="true"
      android:layout_centerhorizontal="true"
      android:text="hello there!"
      android:textcolor="#ffffffff" />

   <button
      android:id="@+id/button1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignparentright="true"
      android:layout_alignparenttop="true"
      android:layout_margintop="20dp"
      android:text="button" />

   <textview
      android:id="@+id/textview1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignparentleft="true"
      android:layout_below="@+id/button1"
      android:layout_margintop="20dp"
```

```
      android:text="posição do banco" />

   <textview
      android:id="@+id/textview02"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@+id/countermessage"
      android:layout_alignleft="@+id/textview01"
      android:text="valor padrão sensor2" />

   <textview
      android:id="@+id/textview03"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignleft="@+id/textview02"
      android:layout_centervertical="true"
      android:text="valor padrão sensor3" />

   <textview
      android:id="@+id/textview04"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignleft="@+id/textview03"
      android:layout_below="@+id/countermessage"
      android:text="valor padrão euclidiano max" />

   <textview
      android:id="@+id/textview01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@+id/textview02"
      android:layout_alignleft="@+id/textview1"
      android:text="valor padrão sensor1" />

   <textview
      android:id="@+id/show"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_above="@+id/statusmessage"
      android:layout_alignright="@+id/countermessage"
      android:layout_marginbottom="20dp"
      android:text="suas leituras aparecerão aqui."

android:textappearance="?android:attr/textappearancelarg
e"
      android:textcolor="#ffffffff"
      android:textsize="14sp" />

   <textview
      android:id="@+id/textview05"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_alignleft="@+id/textview04"
      android:layout_below="@+id/textview04"
      android:text="valor de x" />

</relativelayout>
```

montagem tela leituras

```
<relativelayout
xmlns:android="http://schemas.android.com/apk/res/androi
d"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:background="#ff6eb6ff"
   android:paddingbottom="4dp"
   android:paddingleft="4dp"
   android:paddingright="4dp"
   android:paddingtop="4dp">

   <button
      android:id="@+id/btnvoltar"
      android:layout_width="100dp"
      android:layout_height="50dp"
      android:text="voltar" />

      <listview
         android:id="@+id/lvsensor1"
         android:layout_width="350dp"
         android:layout_height="350dp"
         android:layout_alignparentbottom="true"
         android:layout_toleftof="@+id/lvsensor2"
         android:textsize="7sp" >
      </listview>

</relativelayout>
```