isec
Engenharia

MESTRADO EM ENGENHARIA ELETROMECÂNICA

DEFINITIVO

Aplicação Web de Diagnóstico e Análise de falhas em MIT e Marcadores Visuais

Autor

Hiuram Rodrigues Antunes

Orientadores

Doutor Inácio de Sousa Adelino da Fonseca

Doutor José Manuel Torres Farinha

Coimbra, dezembro 2021

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

DEPARTAMENTO DE ENGENHARIA ELETROMECÂNICA

Aplicação Web de Diagnóstico e Análise de falhas em MIT e Marcadores Visuais

Relatório de Trabalho de Projeto para a obtenção do grau de Mestre em Engenharia Eletromecânica

Especialização em Instalações e Equipamentos em Edifícios

Autor

Hiuram Rodrigues Antunes

Orientadores

Doutor Inácio de Sousa Adelino da Fonseca

Doutor José Manuel Torres Farinha

INSTITUTO POLITÉCNICO DE COIMBRA

INSTITUTO SUPERIOR DE ENGENHARIA DE COIMBRA

Coimbra, dezembro 2021

Agradecimentos

Em primeiro lugar, agradeço aos professores, Doutor Inácio de Sousa Adelino Fonseca e Doutor José Manuel Torres Farinha, que no papel de orientadores, demonstraram constante disponibilidade, apoio, incentivo e confiança.

Aos meus colegas e amigos que me acompanharam durante todo este percurso académico.

Um grande agradecimento à minha família por todo apoio concedido. Em especial, dedico o meu agradecimento à minha mãe pelo incentivo e pela dedicação que me proporcionou durante toda esta fase e também ao meu cunhado, Luciano da Silva, pelos conselhos e incentivo demonstrados desde sempre.

Agradeço ainda à comunidade do Instituto Superior de Engenharia de Coimbra (ISEC) pela oportunidade de formação e experiência e, pelo conhecimento adquirido durante estes anos.

Resumo

Um dos grandes grupos de motores de corrente alternada (AC) é designado por motores de indução (também designados motores assíncronos). São motores que possuem características próprias que os distinguem de outros motores do setor industrial. Em termos de consumo, estima-se que estas máquinas apresentam um consumo de 40% da energia. Neste contexto, apesar destes motores possuírem uma manutenção reduzida, estes estão em funcionamento contínuo ou possuem tempo de paragem reduzido, em condições reais. Assim sendo, podem apresentar algumas avarias no decorrer do seu serviço. Posto isto, têm-se desenvolvido, no âmbito de manutenção preditiva, estratégias para a monitorização contínua destas máquinas. Assim, pretende-se com a realização deste projeto, a implementação de um sistema que permita analisar e diagnosticar as falhas em motores de indução trifásico, integrando este sistema numa aplicação *Web* de modo a monitorizar a condição destes motores.

Palavras Chave: Motor de Indução Trifásico, Análise e Diagnóstico de Falhas, Manutenção Preditiva, Aplicação Web.

Abstract

One of the major groups of alternating current (AC) motors is known as induction motors (also called asynchronous motors). These are motors that have their own characteristics which distinguish them from others in the industrial sector. In terms of consumption, it is estimated that these machines consume 40% of energy. Despite of these engines own a low maintenance, in real conditions, they are in continuous operation or have reduced downtime. Therefore, they may present some faults during their service. Thus, it has been developed as part of predictive maintenance strategies for continuous monitoring of these machines. Therefore, in this project it is intended to implement a system that allows faults analysis and diagnosis in three-phase induction motors. A Web application it's also integrated in order to monitor the condition of these motors.

Key Words: Three-Phase Induction Motor, Faults Analysis and Diagnosis , Web Application, Predictive Maintenance.

O futuro constrói-se no presente, com esforço, dedicação, perseverança e sobretudo com alegria e generosidade.

Dedico este trabalho à minha família.

Índice

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Acrónimos

ADC	*Analog-to-Digital Converter*
API	*Application Programming Interface*
CA	Corrente Alternada
DFT	*Discrete Fourier Transform*
CRUD	*Create, Read, Update and Delete*
DOM	*Document Object Model*
EPRI	*Electric Power Research Institute*
EPVA	*Extended Park's Vector Approach*
f.e.m	Força Eletromotriz
FFT	*Fast Fourier Transform*
GPIO	*General Purpose Input/Output*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IEEE	*Institute of Electrical and Electronics Engineers*
IoT	*Internet of Things*
JSON	*JavaScript Object Notation*
JSX	*JavaScript Syntax Extension*
JWT	*JSON Web Token*
MCSA	*Motor Current Signature Analysis*
MIT	Motor de Indução Trifásico
MQTT	*Message Queuing Telemetry Transport*
MVC	*Model View Controller*
NPM	*Node Package Manager*
ORM	*Object-relational mapping*
PVA	*Park vector analysis*
SPA	*Single Page Application*
STFT	*Short Time Fourier Transform*
SQL	*Structured Query Language*
TW	Transformada de *Wavelet*

REST *Representational State Transfer*

Capítulo 1

Introdução

Nos dias de hoje, os motores de indução a nível industrial apresentam algumas características que os distinguem das outras máquinas. Estas características podem ser vistas desde o ponto de vista de robustez, baixo custo, facilidade de manutenção, alta flexibilidade e capacidade de trabalho em ambientes agressivos. Consequentemente, a manutenção destas máquinas, em ambientes industriais, tem merecido especial atenção devido ao seu funcionamento contínuo ou tempo de paragem reduzido. No entanto, têm-se desenvolvido novas estratégias de manutenção e supervisão para um bom funcionamento destas máquinas. Porém, a sua manutenção periódica ou mesmo a utilização de métodos inovadores nem sempre é suficiente para suprimir algumas das avarias que estas apresentam. Neste sentido, a aplicação de métodos não invasivos tende a ser uma mais valia para acompanhar a condição de monitorização destas máquinas. Uma vez que permite uma manutenção mais adequada e eficaz, de forma a evitar interrupções de processos durante o funcionamento e custos de manutenção desnecessários. A estratégia de manutenção mais simples, e por isso, a mais utilizada, é a manutenção corretiva, que consiste na atuação após a ocorrência da falha, normalmente substituindo o equipamento com falha. Em segundo plano, é utilizada a manutenção preventiva, que consiste na avaliação periódica dos equipamentos mesmo na ausência de falhas. Este tipo de manutenção baseia-se no tempo de vida e no tipo de utilização de cada equipamento. Por último, recorre-se à manutenção preditiva, ou seja, à utilização de técnicas de análise e diagnóstico que têm como objetivo a deteção de falhas num estado embrionário.

1.1 Enquadramento

O presente tema enquadra-se na área da Manutenção Preditiva, sendo neste caso na área particular de análise de falhas em motores de indução trifásico. O tema é de especial importância pois permite de uma forma prévia determinar possíveis falhas em motores antes de acontecerem. A relevância deve-se ao facto de ser um dos elementos mais usados na indústria para transmissão de movimento rotativo.

1.2 Objectivos

O objetivo principal é desenvolver uma aplicação capaz de acompanhar a condição de MITs, promovendo assim, o diagnóstico e análise de falhas destes motores.

- Desenvolver uma aplicação *online* que seja responsável pelo acompanhamento das condições dos motores de indução, através de informações dos seus principais parâmetros vinda do pré-processamento realizado no *hardware*;
- Aplicar as técnicas de diagnósticos de falhas para análises de diferentes tipos de falhas existentes nos motores de indução;
- Estudar diferentes tipos de *Framework* JavaScript para o desenvolvimento do *software*;
- Aplicar algoritmos matemáticos para a aquisição e o pré-processamento de sinais;
- Estudar diferentes tipos de marcadores visuais aplicáveis aos motores elétricos.

O objetivo geral deste projeto é desenvolver um sistema de diagnóstico e análise de motores de MITs. Além disso, numa segunda parte aplicar sistemas inteligentes para processamento de imagem, relativamente a marcadores visuais.

1.3 Motivação e contexto

A motivação para este trabalho nasceu da possibilidade de com uma forma barata e simples implementar um conjunto de trabalhos práticos para a leccionação de aulas no Instituto Superior de Engenharia de Coimbra. Para além deste objetivo, importante, outro mais importante ainda, era juntar num pacote de *software* os

diversos algoritmos associados ao tema que têm sido estudados na comunidade científica, mas de difícil acesso ou ainda não estavam disponíveis num só pacote. Outro objetivo era obter conhecimento na área de manutenção preditiva e nas ferramentas actualmente mais utilizadas.

Atualmente no setor industrial é possível encontrar fabricantes que possuem soluções completas que permitem a monitorização da condição dos motores de indução. Assim, destaca-se a tecnologia *ABB Ability Smart Sensor* da empresa ABB. Estes sensores são instalados sob a carcaça do motor e, permitem capturar dados de vibração, temperatura e outros parâmetros de forma a reduzir o consumo de energia e aumentar em cerca de 30% o tempo de vida útil do motor. Após a recolha dos dados, é possível salvar ou enviar para o servidor (baseado em *cloud*) através de comunicação via bluetooth [1]. Estes dados estarão posteriormente disponíveis na aplicação ou no *smartphone* em função de cada cliente.

Figura 1.1: *ABB Ability Smart Sensor* [1]

Existem ainda, equipamentos portáteis com bastantes funcionalidades utilizados na análise e monitorização nos MITs. Destes equipamentos, destacam-se o *Dynamic Motor Analyser* - Baker EXP40000 do fabricante SKF e o analisador da série 1770 da Fluke.

O *Dynamic Motor Analyser* - Baker EXP4000, representado na Figura 1.2a, é projetado para a utilização local e permite a análise de dados do motor em pleno funcionamento. É um equipamento poderoso destinado à monitorização de motores de indução e utilizado na área de manutenção preditiva. Este permite analisar

parâmetros de qualidade de energia de alimentação, eficiência, corrente, condição do rotor. Integram também um *software* que utiliza algoritmos avançados que permite detectar problemas como: sobrecarga térmica, problemas mecânicos, desequilíbrios mecânicos, barras de rotor quebradas, entre outros. [2].

O analisador da serie 1770, da Fluke, representada na Figura 1.2b, é um outro equipamento concebido para a análise da qualidade de energia e potência em motores monofásicos e trifásicos. Este equipamento disponibiliza medições automáticas e permite facilmente determinar os parâmetros de tensão, corrente e harmónicos.

(a) SKF *Dynamic Motor Analyser [2]*

(b) Analisador Fluke 1770 [3]

Figura 1.2: Ferramentas de análise da condição de motores de indução

1.4 Organização do documento

Para além do capítulo introdutório, o projeto encontra-se organizado pelos seguintes capítulos:

- no Capítulo 2 faz-se uma introdução ao MIT e restantes aspectos relacionados;
- no Capítulo 3 apresentam-se as principais técnicas de diagnóstico de falhas e de processamento de sinal;
- no Capítulo 4 apresentam-se as tecnologias modernas de desenvolvimento de aplicações *Web*;
- no Capítulo 5 desenvolvem-se a formalização dos requisitos do Sistema;
- no Capítulo 7, finalmente, apresentam-se as principais conclusões e possibilidades de trabalho futuro.

Capítulo 2

Motor de Indução trifásico

Neste capítulo abordam-se os conceitos fundamentais do motor de indução trifásico (MIT). Assim, é descrito ao longo das próximas secções, o modo de funcionamento, seus componentes e as suas características funcionais. Na secção 2.5 apresentam-se ainda, sumariamente, as principais falhas que ocorrem nos MIT, nomeadamente as falhas de natureza mecânica e elétrica.

2.1 Introdução

O MIT (Figura 2.1) ou simplesmente motor assíncrona trifásico, é a máquina elétrica mais robusta e a mais utilizada na maioria dos sistemas eletromecânicos. Estes motores possuem características próprias e exclusivas, distinguindo-se de outras máquinas elétricas, visto que apresentam baixo custo, reduzida manutenção e uma eficiência elevada.

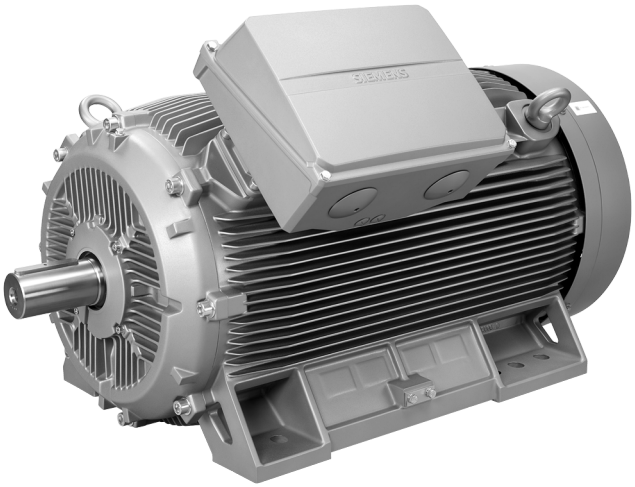


Figura 2.1: Motor de indução trifásico [4].

2.2 Constituição

O MIT é essencialmente constituído por dois componentes básicos, o estator e o rotor, sendo uma parte estática e outra móvel, respetivamente. O espaço de ar que separa e impede o contato entre o estator e o rotor é denominado de entreferro. Os elementos indispensáveis para o bom funcionamento de um MIT são representados na Figura 2.2.

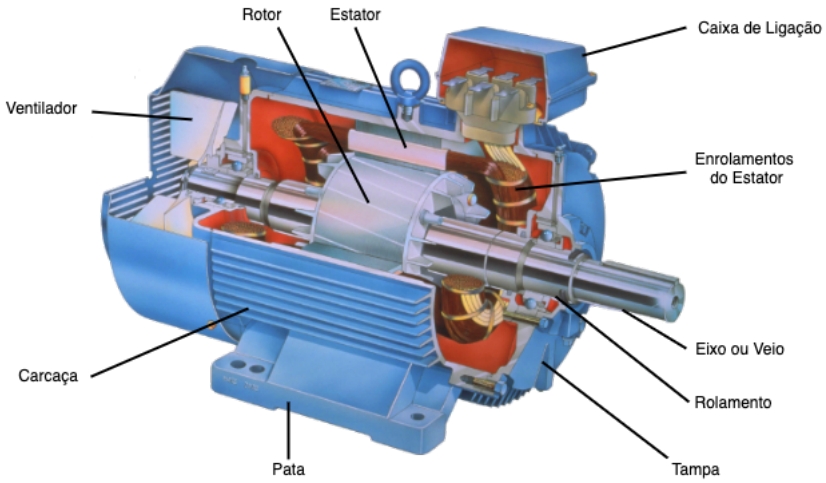


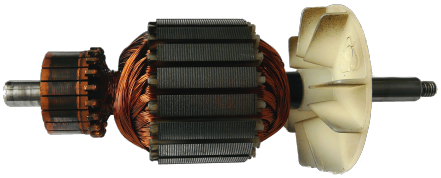
Figura 2.2: Elementos constituintes do MIT [5].

O estator, parte estática do motor, é constituído pelo empilhamento de chapas finas de aço magnético laminado tratadas termicamente de forma a reduzir ao mínimo as perdas por correntes *Foucault* (também designadas de correntes parasitas) e histerese. As chapas apresentam uma ranhura interna onde estão alojados os enrolamentos.

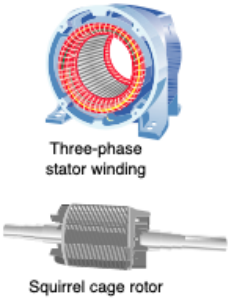
O rotor, parte móvel do motor, é também composto por chapas finas empilhadas de aço magnético laminado tratadas termicamente e isoladas com verniz entre si.

Atendendo ao tipo de rotor, o MIT pode ser classificado em dois tipos: Rotor de gaiola de esquilo e rotor bobinado, como apresentado na Figura 2.3. O mais comum é o rotor de gaiola de esquilo. Neste tipo, o enrolamento consiste em barras de cobre ou alumínio, encaixadas nas ranhuras do rotor e ligadas entre si em curto-circuito por anéis condutores [6]. Por outro lado, o rotor bobinado ou enrolado é menos comum e, só utilizado em aplicações especiais, onde é necessário a alteração de características no rotor, nomeadamente na resistência. O enrolamento, neste tipo de rotor é estruturado de forma semelhante ao do estator, contendo o mesmo número

de pólos.



(a) Rotor Bobinado



(b) Rotor de gaiola de esquilo

Figura 2.3: Tipo de rotores [5]

2.3 Princípio de Funcionamento

No MIT, o estator encontra-se ligado à rede de alimentação de corrente alternada (CA) e, é constituída por três enrolamentos desfasados entre si por um ângulo de $2\pi/3$ rad (120°). Por outro lado, no rotor não é aplicada qualquer tensão ou conexão externa, no entanto, as correntes são induzidas a partir dos enrolamentos do estator.

Todavia, quando os enrolamentos do estator são percorridos por uma corrente elétrica nas bobinas em cada uma das fases, gera-se um campo magnético girante, que tende a girar no entreferro à velocidade de sincronismo, como representada pela Equação 2.1.

$$n_s = \frac{60 \times f}{p}[rpm] \tag{2.1}$$

Em que, **f** é frequência de alimentação da rede (Hz) e **p** o número de pares de pólos.

Assim, atendendo à *Lei de Faraday*, quando o campo magnético atravessa o rotor, provoca uma variação no fluxo magnético do mesmo, gerando uma força electromotriz induzida (f.e.m). Por sua vez de acordo com a *Lei de Lenz* ou Lei da Auto-Indução, a corrente induzida no circuito fechado, possui um sentido tal que, pelas suas ações magnéticas tende sempre a opor-se à causa que lhe deu origem.

O escorregamento ou deslizamento (s) é determinado pela razão da velocidade de rotação síncrona n_s do campo e a velocidade do rotor e, é obtido pela Equação 2.2.

$$s = \frac{(n_s - n)}{n_s} \times 100 \tag{2.2}$$

A velocidade do rotor pode ainda ser obtida em função do escorregamento, como indica a Equação 2.3.

$$n = (1 - s)n_s \tag{2.3}$$

A velocidade do rotor é ligeiramente inferior à velocidade do campo magnético girante. Esta diferença depende dos binários de carga no veio do motor. Daí ser considerada uma máquina assíncrona, visto que a velocidade de rotação do motor e a frequência das grandezas elétricas de alimentação não têm uma relação constante.

A frequência induzida no rotor, f_2 é obtida pelo produto entre escorregamento (s) e a frequência da rede (f), conforme a Equação 2.4.

$$f_2 = s \times f \tag{2.4}$$

2.4 Características Funcionais

Nos motores elétricos, o rendimento nunca atinge o valor de 100% uma vez que durante o seu funcionamento ocorrem perdas. No entanto quanto mais próximo da unidade, maior é a eficiência do motor. O rendimento é determinado pela razão entre a energia utilizada e a energia fornecida e, calculado de acordo com a Equação 2.5.

$$n = \frac{P_{mec}}{P_{abs}} \times 100\% \tag{2.5}$$

Em que, P_{mec} é a potência mecânica fornecida pelo motor ao veio e P_{abs} a potência absorvida da rede de alimentação.

Assim, nos MITs, as principais perdas que ocorrem classificam-se em:

- **Perdas eléctricas**, ou perdas por efeito de joule nos condutores do estator e do rotor. Normalmente, estas ocorrem devido a circulação da corrente elétrica nos condutores dos enrolamentos que originam dissipação do calor;

- **Perdas mecânicas** que ocorrem devido ao atrito nos rolamentos e na ventilação. Estas são influenciadas pela tipo de rolamentos, velocidade do rotor, a força exercida no rolamento, entre outros;

- **Perdas magnéticas** surgem devido ao efeito de histerese e às correntes de *Foucault* nas lâminas ferromagnéticas do estator e do rotor.

Para além das perdas anteriormente mencionadas, podem surgir ainda perdas adicionais ou residuais, que não são contabilizadas pela soma das potências das perdas anteriores.

A potência elétrica (P_{ele}) absorvida pelo motor é obtida pela soma das potências das três fases e dada pela Equação 2.6.

$$P_{ele} = 3 \times P_f = 3 \times V_f \times I_f \times \cos(\psi) \tag{2.6}$$

Onde, P_f é a potência da fase e é obtida pelo produto da tensão (V_f) e da corrente (I_f) por fase.

2.5 Análise de Falhas em MIT

Os motores de indução são altamente fiáveis, mas também estão sujeitos à ocorrência de um vasto número de falhas indesejáveis, que por sua vez perturbam a simetria do motor [7]. Na generalidade, as falhas em MITs, podem ser classificados em duas grandes categorias:

- Falhas de origem interna, que ocorrem devido a causas internas da própria máquina. Estas falhas podem ainda ser subdivididas em falhas mecânicas ou elétricas;
- Falhas externas quando surgem derivadas a problemas externos à máquina, tais como causas elétricas, mecânicas ou ambientais.

Assim sendo, a Figura 2.4 e a Figura 2.5 mostram resumidamente, as falhas de origem interna e externa, bem como as suas principais causas.

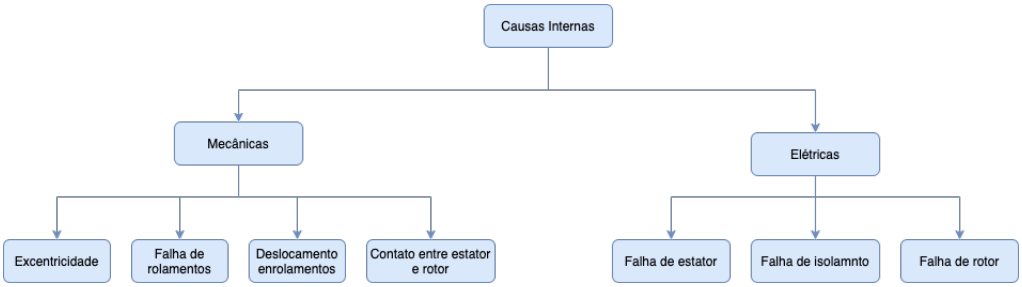


Figura 2.4: Falhas internas, adaptado de [8].

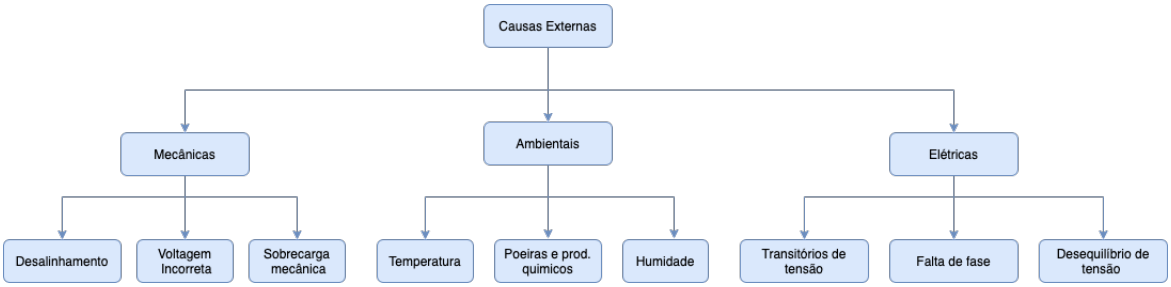


Figura 2.5: Falhas externas, adaptado de [8].

De um modo geral, são vários os estudos realizados até ao momento que investigam a distribuição das falhas em MITs. Na Tabela 2.1, apresentam-se os resultados obtidos pelos estudos realizados pelo *Institute of Electrical and Electronics Engineers* (IEEE) e pelo *Electric Power Research Institute* (EPRI). O estudo elaborado pelo IEEE, foi realizado em diversos motores de indução, em ambiente industrial. Assim, de uma forma geral, e de acordo com a Tabela 2.1, as falhas que ocorrem mais frequentemente são aquelas que surgem nos rolamentos e no estator.

Estudo	**Rolamento (%)**	**Estator (%)**	**Rotor (%)**	**Outras (%)**
IEEE	42	28	8	22
EPRI	41	36	9	14

Tabela 2.1: Percentagem de avarias que surgem nas diversas partes do motor [9].

Nas subseções seguintes, são descritos os tipos mais comuns de falhas nos MITs.

2.5.1 Falhas no Estator

A maioria das falhas que ocorrem no estator derivam de problemas que surgem no enrolamento do estator, devido a curto-circuitos. No entanto, podem também surgir problemas no núcleo de ferromagnético, sendo pouco frequentes.

Avarias no enrolamento do estator

Normalmente, as avarias associadas aos enrolamentos do estator são devidas às falhas ou deterioração no isolamento dos enrolamentos, mais especificamente entre as espiras das bobinas. Assim sendo, as possíveis avarias associados ao estator podem ser divididas nos seguintes tipos [9]:

- Curto-circuito entre duas espiras da mesma fase;
- Curto-circuito entre duas bobinas da mesma fase;
- Curto-circuito entre espiras de duas fases;
- Curto-circuito entre espiras das três fases;
- Curto-circuito entre os condutores do enrolamento e o núcleo do estator;

- Falha de circuito aberto quando um dos enrolamento parte.

Na Figura 2.6 ilustra-se os diversos tipos de avarias anteriormente descritos.

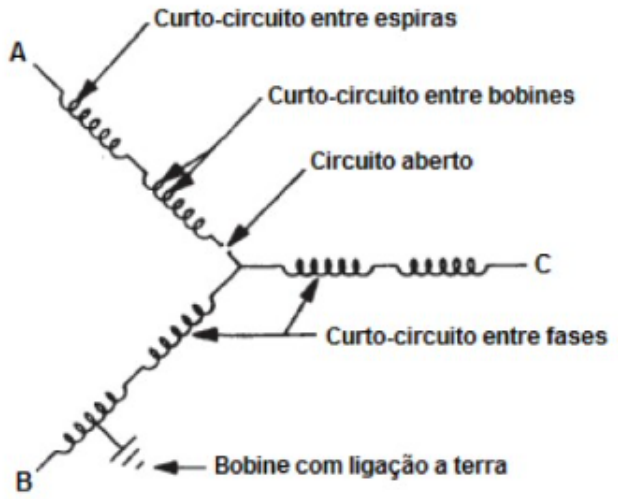


Figura 2.6: Tipos de avarias no enrolamento do estator em ligação em estrela [9].

2.5.2 Falhas no Rotor

As principais falhas do rotor podem ser de natureza elétrica, nomeadamente barras partidas, ou mecânica como é o caso da excentricidade do rotor. Em seguida, apresentam-se em resumo, as principais falhas relativas ao rotor.

Barras de rotor partidas

As falhas de barras partidas ocorrem maioritariamente em motores de grande potência, em especial, nos motores com rotor de gaiola de esquilo (principalmente na região junto aos anéis terminais) e, em condições pouco favoráveis, devido aos seguintes fatores [9, 10]:

- Anomalias na fabricação;
- Sobreaquecimento e oscilações de temperatura;
- *Stress* mecânico devido a avarias nos rolamentos;
- Arranques frequentes no motor a altas voltagens;
- Corrosão do metal da barra do rotor.

Na Figura 2.7 apresenta-se um exemplo de um rotor com uma e duas barras partidas.

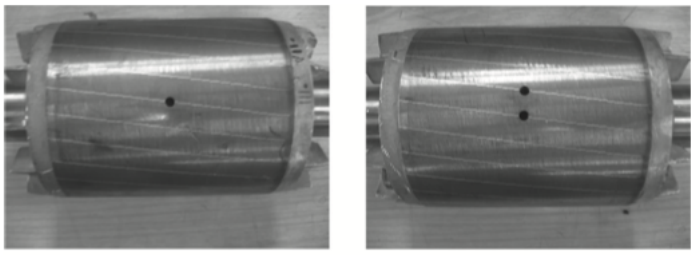


Figura 2.7: Exemplo de Rotor com uma e duas barras partidas [11].

Excentricidade no rotor

A excentricidade caracteriza-se por uma distância não uniforme entre o rotor e a parte interna do estator. Entretanto, as avarias mais comuns podem ter origem em deformações no veio, desgaste nos rolamentos, nas chumaceiras, núcleo do estator oval, entre outros.

A excentricidade pode ser classificada em dois tipos: estática e dinâmica, como apresentado na Figura 2.8. Para além disso, à combinação dos dois tipos de excentricidade dá-se o nome de excentricidade mista.

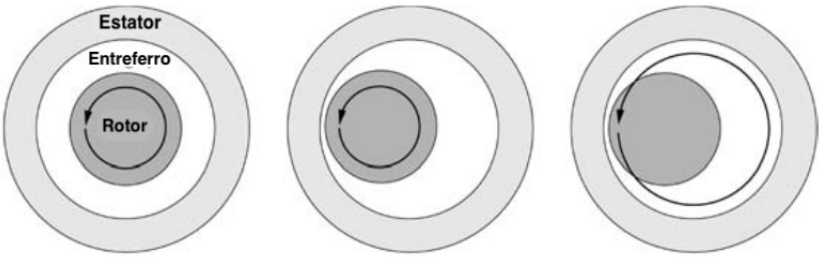


Figura 2.8: Concêntrico e tipos de excentricidade (estática e dinâmica).

Num motor ideal, sem defeitos, o centro de rotação do estator e do rotor estão perfeitamente alinhados, ou seja, o centro de rotação do rotor é o mesmo que o do estator. Se o centro geométrico do rotor se encontrar deslocado com o centro de rotação designa-se por excentricidade dinâmica. Por outro lado, se o centro de rotação coincidir com o centro geométrico do rotor e este se encontrar deslocado relativamente ao centro geométrico do perímetro interno do estator, dá-se o nome de excentricidade estática. A excentricidade é responsável por provocar ruído, vibrações e um desequilíbrio na força de atracão magnética.

Em máquinas elétricas, em condições reais, é impossível não existir excentricidade. Pelo que, os fabricantes recomendam um nível de excentricidade máximo entre 5-10%.

2.5.3 Avarias nos Rolamentos

O rolamento é um dos principais componentes encontrado em MIT. Este é utilizado para suportar o rotor e tem como principal função reduzir o atrito ou fricção de deslizamento entre as superfícies de contato. É constituído por dois anéis, um interior e um exterior e por um conjunto de esferas ou rolos cilíndricos fixos numa armadura lubrificada como ilustrado na Figura 2.9.

Figura 2.9: Rolamento [5].

De acordo com a Tabela 2.1, a maior parte das avarias em MITs, são as avarias mecânicas nos rolamentos, apresentando uma percentagem de aproximadamente 42%. Qualquer defeito no rolamento traduz-se num movimento anormal no MIT [7, 9].

As causas mais comuns que conduzem ao aparecimento de falhas nos rolamentos são:

- Cargas e temperaturas elevadas;
- Ajustes apertados;
- Fadiga;
- Lubrificação inadequada;
- Corrosão e contaminação;
- Desalinhamento dos mancais.

Algumas das consequências que advém destas falhas são o aparecimento de vibrações elevadas, desgaste, aumento do atrito e o sobreaquecimento.

2.5.4 Vibrações

As vibrações fazem parte de qualquer motor e surgem devido a forças dinâmicas, nomeadamente, desequilíbrios, desalinhamento, desgaste, entre outros. Estas são consequência de uma resposta periódica ou oscilatória. Aos ciclos de vibração denominam-se de frequência. Desta forma, entende-se por vibrações qualquer "movimento repetitivo, mesmo em altas frequências, com amplitudes baixas e comportamento irregular e aleatório."[12]
As avarias mais communs que resultam de altos níveis de vibrações em MIT são [13]:

- Desequilíbrio nos componentes rotativos;
- Desalinhamento nos rolamentos;
- Danos em rolamentos e engrenagem;
- Defeitos nas correntes e correias de transmissão;
- Variações do binário;
- Ressonância.

2.6 Conclusão

Este Capítulo de forma resumida apresentou as principais metodologias de análise de suporte matemático do MIT, bem como uma introdução resumida ao mesmo.

Capítulo 3

Técnicas de processamento de sinal e Diagnóstico de falhas

Na secção 2.5 do capítulo 2, foram descritos os principais tipos de avarias que podem surgir em MITs. Para que se obtenham bons resultados é importante a deteção precoce das mesmas. Assim sendo, neste capítulo, abordam-se os diferentes tipos de técnicas de processamento de sinal que são imprescindíveis no diagnóstico de avarias e, na secção 3.2, descrevem-se as técnicas utilizadas para a identificação de falhas.

3.1 Técnicas de Processamento de Sinal

As técnicas de processamento de sinal são frequentemente utilizadas para filtrar, compactar ou extrair informações de um sinal. Resumidamente, o processo inicia-se com a medição de um sinal analógico por parte de um computador ou de um processador de sinal que transforma o sinal analógico num sinal digital.

A análise dos sinais adquiridos pode ser realizada em três domínios: domínio de tempo, domínio de frequência e domínio de tempo-frequência.

3.1.1 Domínio do Tempo

A técnica de análise no domínio de tempo, é geralmente utilizada no processamento de sinais. Apesar desta técnica permitir a identificação de falhas e avaliar o seu processo de evolução, não permite identificar a natureza e a localização das mesmas, pelo que deve ser complementada por outros métodos de análise.

Neste domínio destaca-se uma técnica de processamento de sinais, a transformada

de *Hilbert*. No entanto, para a análise da vibração, são utilizadas outras técnicas no domínio do tempo e baseadas na amplitude do sinal, sendo estas, o cálculo do valor eficaz ou valor RMS, do valor pico, do valor pico a pico, entre outras. No caso do valor RMS, existem valores tabelados, como vai ser demonstrado na Figura **??**, da subsecção 3.2.3 .

Transformada de *Hilbert*

A transformada de *Hilbert* é um método eficiente para o processamento de sinais de amplitude e fase. Uma vez que permite segregar a componente essencial do sinal das restantes componentes, facilitando assim, a detecção das avarias.
A transformada de *Hilbert* $H[s(t)]$ de um sinal s(t), é definida pela Equação 3.1 [9].

$$H[s(t)] = \frac{1}{\pi} \int_{-\infty}^{+\infty} \frac{s(\tau)}{(t-\tau)} d\tau \tag{3.1}$$

Utilizando o teorema do valor médio, a Equação 3.1 pode ser simplificada pela expressão da Equação 3.2:

$$H[s(t)] = \frac{1}{\pi t} \otimes s(t) \tag{3.2}$$

3.1.2 Domínio da Frequência

A análise no domínio frequência, também designada por análise espetral, baseia-se na transformação de um sinal no domínio do tempo em um sinal no domínio da frequência, tendo como principal objetivo a identificação das harmónicas que compõem o sinal. Ou seja, por este método, os sinais de amplitude de frequência maiores traduzem-se em sinais mais significativos. Neste domínio as técnicas que assumem maior relevância são a Transformada Rápida de *Fourier* (FFT) e a Técnica de Envelope.

Transformada Rápida de *Fourier* (FFT)

Apesar da FFT ser uma técnica muito utilizada, a sua eficácia está apenas associada a sinais estacionários ou regimes permanentes de funcionamento, pelo que esta necessita de uma grande amostragem de sinal e consequentemente requer um grande intervalo de tempo, para além da existência de interferências, a nível do ruído, presentes em qualquer ambiente industrial, durante a obtenção dos sinais. A

Transformada Rápida de *Fourier* é uma transformada utilizada para a conversão de um sinal no domínio do tempo para o domínio da frequência, sem perda de informação e vice-versa [9].

A FFT é um algoritmo eficaz para o cálculo da Transformada Discreta de *Fourier* DFT e a sua inversa e é definida pela expressão da Equação 3.3 [12]:

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-j\frac{2\pi kn}{N}}, k = 0, 1, ..., N-1 \tag{3.3}$$

Em que, **n** é o índice do vetor discreto do sinal x(n) no domínio do tempo, **k** é o índice do vetor discreto do sinal X(k) no domínio da frequência e **N** é a quantidade total de amostras.

Segundo *Cooley* e *Tukey* é possível dividir a DFT em duas partes mais pequenas, em que N é dado por uma potência de 2, ou seja, $N = 2^m$ e assim obter-se a expressão da FFT [9]:

$$X(k) = \sum_{m=0}^{N/2-1} x(2m) e^{-j\frac{2\pi km}{N/2}} + \sum_{m=0}^{N/2-1} x(2m+1) e^{-j\frac{2\pi km}{N/2}} e^{-j\frac{2\pi k}{N}} \tag{3.4}$$

Em que **N/2**, deve-se à divisão do número de amostras em duas partes, cada uma com metade do número de pontos de amostra.

Técnica do Envelope

A análise de envelope, também designada por análise de ressonância em alta frequência ou desmodulação de ressonância, trata-se de uma técnica eficiente para o processamento de sinais e identificação de avarias em rolamentos de corpos salientes, engrenagens, entre outros. [13].

Nesta técnica, é aplicado um filtro que permite a passagem de altas frequências mas bloqueia a passagem das frequências mais baixas, também designado por filtro passa-banda. Este tipo de análise é realizada em três etapas: primeiro é feita a filtragem do sinal através do filtro passa-banda; de seguida é realizada a extração do envelope, com o recurso à Transformada de *Hilbert* e por fim é feita a extração do espetro de frequência do sinal envelopado aplicando a FFT.

3.1.3 Domínio do Tempo-Frequência

Na análise no domínio Tempo-Frequência, os sinais são representados num único plano e em dois domínios em simultâneo, o tempo e a frequência [11]. De entre as ferramentas matemáticas mais disseminadas, no domínio de Tempo-Frequência, destacam-se a Transformada de *Wavelet* (TW) e a Transformada de *Fourier* de Curta Duração (STFT). Ambas as técnicas podem ser usadas para a análise de sinais não estacionários.

Transformada de *Fourier* de Curta Duração (STFT)

A STFT é a técnica tradicional de análise no domínio tempo-frequência. Esta técnica deriva da aplicação em blocos ou janelas da Transformada de *Fourier* [14]. A STFT, fornece uma resolução constante em todas as frequências.

A STFT de um sinal $\mathbf{x(t)}$, no domínio do tempo, é dada pela Equação 3.5.

$$STFT_x^w(t,f) = \int_{-\infty}^{+\infty} x(t)w(t-\tau)e^{-j2\pi ft}d\tau \tag{3.5}$$

Em que o sinal de STFT é resultante da multiplicação do FT do sinal por uma função de janela $w(t-\tau)$.

Transformada de *Wavelet*

A TW é considerada uma alternativa à STFT para a análise de sinais não estacionários. Para além de ser uma das ferramentas mais avançada e eficiente utilizada no processamento de sinais. Esta técnica foi desenvolvida para suprimir os problemas de resolução da STFT uma vez que este método utiliza a técnica de multi-resolução, em que frequências diferentes são analisadas com resoluções diferentes, o que difere da STFT. Para além disso, a TW, na representação no domínio tempo-frequência de um sinal, utiliza uma janela escalonável [14, 11].

O reconhecimento de padrões através da análise multi-resolução, faz com que o método baseado na TW, seja adequado para o diagnóstico de falhas em MITs.

A Equação 3.6 descreve a fórmula básica de TW.

$$W(d,\tau) = \int_{-\infty}^{+\infty} x(t)\frac{1}{\sqrt{d}}\psi\left(\frac{t-\tau}{d}\right))d\tau \tag{3.6}$$

Em que, τ representa a translação, ou seja, a localização da janela e d representa a escala, ou seja, o parâmetro de frequência.

Integrando a zero uma *Wavelet*, obtém-se uma *Wavelet* mãe. Esta *Wavelet* permite gerar outras funções de janela e traduz-se pela Equação 3.7.

$$\psi_{d,t}(t) = \frac{1}{\sqrt{d}}\psi\left(\frac{t-\tau}{d}\right) \tag{3.7}$$

Em que o fator $\frac{1}{\sqrt{d}}$ é utilizado para garantir o princípio da conservação de energia.

É ainda possível encontrar duas implementações da TW, uma quando esta é aplicada de forma contínua, denominada Transformada Contínua de *Wavelet* CWT e outra quando esta é aplicada na sua forma discreta, representada pela Transformada Discreta de *Wavelet* DWT.

3.2 Técnicas de identificação de Falhas

O monitoramento de condições consiste na avaliação do estado de saúde de uma planta industrial e do seu conjunto de máquinas. A detecção precoce de falhas é uma etapa fundamental para a manutenção preventiva e para o correto funcionamento das máquinas, reduzindo assim, o seu tempo de paragem. Para aumentar a produtividade, confiabilidade e a segurança das máquinas é essencial o monitoramento de condições. Além disso, este processo é também muito importante uma vez que permite [7]:

- Redução dos custos de manutenção;
- Determinação de falhas;
- Aumento da confiablidade das máquinas e seus componentes;
- Otimizar a mão de obra e as peças suplentes;
- Maximizar o desempenho;

- Aumentar a precisão da previsão da falha.

Na generalidade, o monitorimento das condições de operação do motor, pode ser efetuado por meio de três principais etapas, as quais são, aquisição de dados, processamento e análise de sinal. Na etapa de aquisição de dados, são mensuradas, através de sensores ou transdutores, as grandezas elétricas e mecânicas, nomeadamente, o fluxo magnético, a corrente/tensão e a vibração. De seguida é feito o processamento dos sinais medidos, avaliando a presença ou não de avarias. Por último, realiza-se o monitoramento que consiste na análise da assinatura de condição da máquina, para identificar a presença de uma possível falha.

Neste contexto, existem várias técnicas de identificação de falhas em MIT. Estas podem dividir-se em duas categorias (técnicas invasivas e técnicas não invasivas), tal como ilustrado na Figura 3.1.

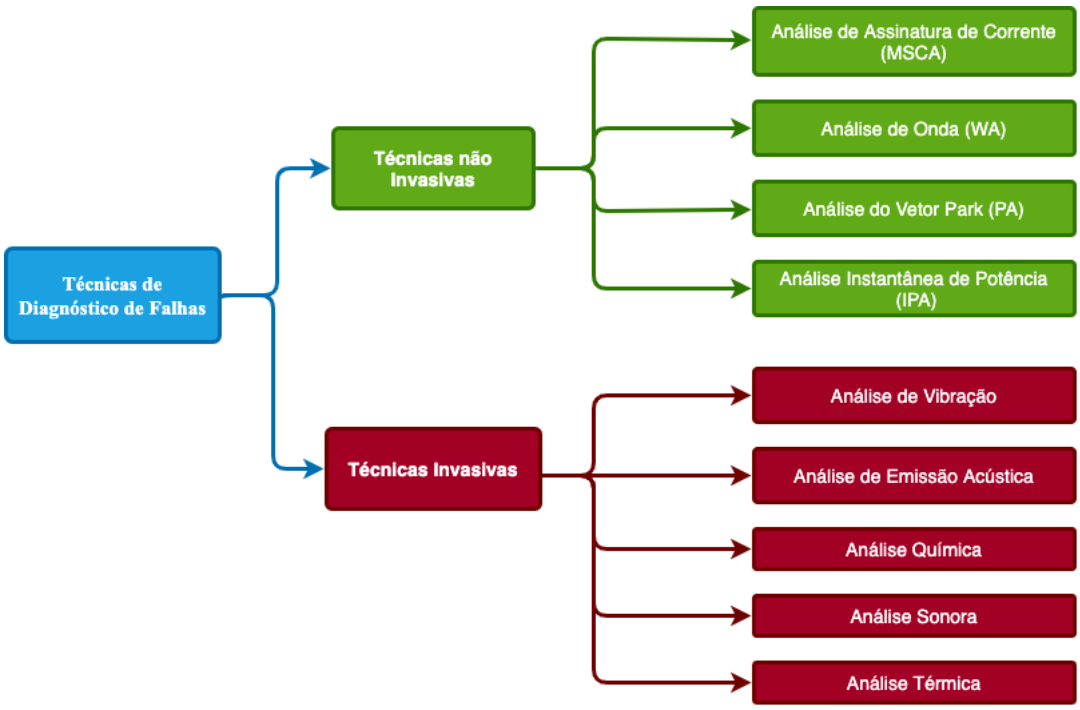


Figura 3.1: Técnicas de diagnóstico de falhas em MIT [5].

De um modo geral, nas próximas secções apenas serão abordadas as técnicas nas quais este projeto tem foco principal.

3.2.1 *Motor Current Signature Analisys* (MCSA)

O MCSA, é uma das técnicas mais utilizada na identificação de falhas em motores de indução. Além de ser uma técnica não invasiva, esta é ainda uma técnica de deteção online, permitindo o monitoramento durante o funcionamento da máquina. O MCSA baseia-se apenas na análise do sinal de corrente do motor e, usualmente, pode detectar os seguintes problemas [15]:

- Barras do rotor partidas;
- Avarias nos Rolamentos;
- Excentricidade dinâmica do rotor;
- Curto-circuito nos enrolamentos do estator.

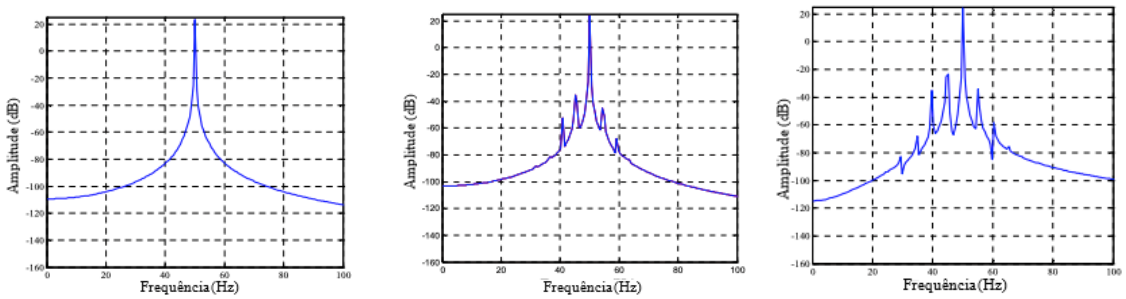
O MCSA apresenta várias vantagens, nomeadamente uma curta duração no diagnóstico, fácil aplicação em motores de várias potências e informar à cerca do estado de degradação.

A técnica MCSA analisa o espetro da corrente produzido pela corrente no enrolamento do estator, no qual são identificadas a amplitude e frequência de cada componente que constitui o sinal. Na presença de uma avaria no padrão da assinatura de corrente, surgem assimetrias, nas quais são identificadas as harmónicas, numa frequência e amplitude diferentes da harmónica fundamental. Essas harmónicas são processadas por meio de transformada rápida de *Fourier* (FFT).

Numa situação em que o rotor se apresente saudável (Figura3.2a) verifica-se que a assinatura de corrente do estator apresenta um comportamento sinusoidal com uma única harmónica de frequência 50 HZ. No entanto, quando o rotor possui barras partidas surgem componentes espectrais de bandas laterais junto da componente fundamental, como ilustram as Figuras 3.2b e 3.2c.

A frequência de falha, dada pelas componentes espetrais de bandas laterais, é determinada pela Equação 3.8, onde s representa o escorregamento e f_s a frequência fundamental. Com esta equação é possível identificar e detetar a presença de barras partidas no rotor, que resultam em componentes da corrente induzidas nas bobines do estator.

$$f_{BP} = f_s(1 \pm 2s) \tag{3.8}$$



(a) Rotor saudável (b) Rotor 1 barra partida (c) Rotor 2 barras partidas

Figura 3.2: Espetro de frequência de barras partidas, adaptado de [16].

A Tabela 3.1, representa os limites de severidade e recomendações na análise da condição do rotor. Assim, quanto maior a diferença entre a frequência fundamental e as frequência de bandas laterais, maior será a estimativa de barras partidas. Assim, para uma diferença de amplitudes superior a 60dB, considera-se a condição do rotor saudável.

Condição	**Diferença entre amplitudes [dB]**	**Condição da gaiola do rotor**	**Ação a executar**
1	>60	Excelente	Nenhuma
2	54-60	Boa	Nenhuma
3	48-54	Moderada	Analisar tendência
4	42-48	Barra rachada em desenvolvimento ou juntas com alta resistência	Diminuir o intervalo de testes e analisar tendência
5	36-42	Duas barras possivelmente rachadas ou juntas com elevada resistência	Confirmar com a análise de circuito do motor
6	30-36	Múltiplas barras ou anéis de curto circuito rachadas ou quebradas	Rebobinar
7	<30	Danos severos na gaiola do rotor	Rebobinar ou substituir

Tabela 3.1: Limites de severidade e respetivos diagnósticos de condição do rotor, adaptado de [17].

3.2.2 *Extended Park's Vector Approach* (EPVA)

O EPVA, baseia-se na transformação de Park em que, após a obtenção simultânea do sinal de todas as correntes é aplicada a transformada de park, na qual um MIT é transformado num sistema bifásico equivalente, representado por um sistema de

eixos ortogonais d e q. Após a transformada de park é calculado o modúlo do vetor, das tensões, fluxos e correntes, e por fim o seu espetro. As componentes do vetor de Park, i_d e i_q, são obtidas pelos valores das correntes das três fases do motor i_a, i_b e i_c, como representadas nas Equações 3.9 e 3.10, respetivamente.

$$i_d = \sqrt{\frac{2}{3}i_a} - \sqrt{\frac{1}{6}i_b} - \sqrt{\frac{1}{6}i_c} \tag{3.9}$$

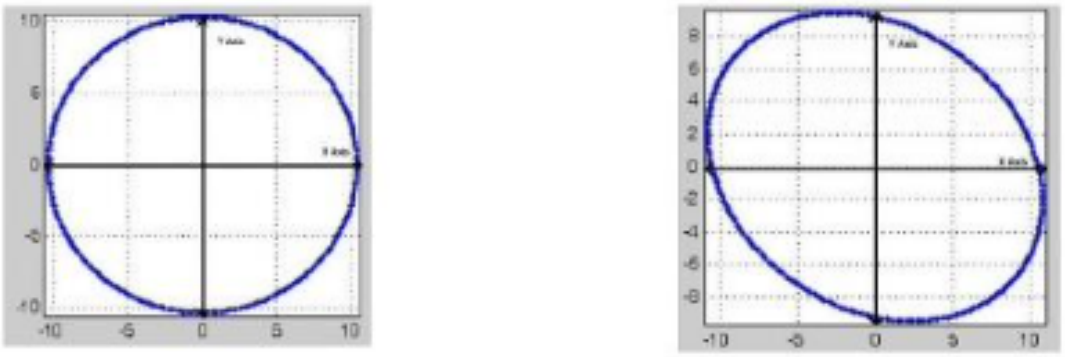
$$i_q = \frac{1}{\sqrt{2}}i_a - \frac{1}{\sqrt{2}}i_c \tag{3.10}$$

O EPVA, é uma técnica utilizada no diagnóstico das seguintes avarias:

- Falhas estacionárias no rotor;
- Tensões de alimentação desequilibradas;
- Falhas nas espiras do estator;
- Desalinhamento mecânico.

Apesar de permitir o diagnóstico das avarias, esta técnica é mais utilizada para avaliar a severidade das mesmas.

Assim, o padrão da curva do Vector de Park de um motor sem avarias é uma circunferência centrada na origem das suas coordenadas (0,0) (Figura 3.3a) e no caso do motor com avarias o circulo é distorcido para a forma de uma elipse, tal como representado na Figura 3.3b.



(a) Motor saudável (b) Motor com avarias

Figura 3.3: Representação do vetor de *park* em motores, adptado de [10]

3.2.3 Análise de vibração

O monitoramento pela análise de vibração é a técnica invasiva mais eficaz, para detectar as avarias de natureza mecânica no MIT. As avarias mais comuns diagnosticadas por esta análise já foram descritas na seção 2.5.4 do capítulo 2.

Na análise de vibração, recorre-se essencialmente aos sensores instalados no motor. Esta baseia-se na medição de três grandezas, tais como, deslocamento, velocidade e aceleração. Além disso, para a análise de vibração, são utilizadas maioritariamente a aceleração e a velocidade em gamas de frequência baixa [9, 12]. Normalmente, nos motores com características semelhantes, tipicamente possuem velocidade constante. Ou seja, com o aumento da frequência de vibração, espera-se que o deslocamento diminua e a aceleração aumente, pelo que com o aumento da frequência tenha que se progredir sucessivamente de um dispositivo de descolamento para um transdutor de velocidade e, por fim um acelerômetro. Após a medição das grandezas, é efetuado o processamento com o recurso à transformada rápida de *Fourier* FFT e, obtido o espetro de frequência. Um motor saudável produz um espectro de frequência característico ou de referência. Qualquer vibração altera o espetro de frequência, que por sua vez, ao comparar-se com o espetro de referência é possível detetar e diagnosticar as avarias. O valor eficaz do nível de vibração é medido na largura de banda, usualmente utilizada, compreendida na faixa de frequências 0,01–1 kHz ou 0,01–10 kHz.

Apesar desta técnica ser simples e das mais utilizadas no diagnóstico de falhas nos MITs, esta apresenta algumas limitações tais como, interferência da vibração do meio industrial, eficácia limitada e necessita de sensores de custo elevado.

Para avaliar a condição de funcionamento das máquinas, utilizam-se as normas ISO 10816 e 20816. Estas especificam os limites de severidade de vibração em máquinas de pequenas e grandes potências.

Segundo a norma ISO 10816-3, as máquinas são classificadas de acordo com o tipo, a potência ou a altura de veio e a rigidez da estrutura de suporte, deste modo são divididas em quatro grupos [18]:

- **Grupo I**: Máquinas com potência superior a 300 kW; máquinas elétricas com altura de veio H 315 mm;
- **Grupo II**: Máquinas com potência entre 15 e 300 kW; máquinas elétricas com altura(H) de veio entre 160 e 315 mm;

- **Grupo III**: Bombas com impulsor multivuluta com accionamento separado com potência superior a 15 kW;
- **Grupo IV**: Bombas com impulsor multivuluta com accionamento integrado e com potência superior a 15 kW.

Na Tabela 3.2 representa-se a classificação e os limites de severidade de vibração de acordo com a velocidade RMS e o respetivo grupo/classe.

Severidade da Velocidade RMS	Limites da Gama de Velocidade e Classes de Máquinas			
[mm/s]	Classe I	Classe II	Classe III	Classe IV
0,28	BOM	BOM	BOM	BOM
0,45				
0,71				
1,12	SATISFATÓRIO			
1,8		SATISFATÓRIO		
2,8	SATISFATÓRIO (Alerta)		SATISFATÓRIO	
4,5		SATISFATÓRIO (Alerta)		SATISFATÓRIO
7,1	INACEITÁVEL (Perigo)		SATISFATÓRIO (Alerta)	
11,2		INACEITÁVEL (Perigo)		SATISFATÓRIO (Alerta)
18			INACEITÁVEL (Perigo)	
28				INACEITÁVEL (Perigo)
45				

Tabela 3.2: Limites de severidade de vibração, adaptado de [18]

3.3 Conclusão

Este Capítulo, apresenta de uma forma resumida as principais técnicas de processamento de sinal, bem como as técnicas de identificação de falhas mais comuns em MITs.

Capítulo 4

Desenvolvimento de aplicações *Web*

Este capítulo descreve, sucintamente os conceitos gerais e tecnologias modernas usadas para o desenvolvimento de aplicações *Web*, dando ênfase à linguagem JavaScript e uma das suas bibliotecas bastante difundida o ReactJS. Também serão abordados conceitos chaves que auxiliam na implementação do cliente e servidor.

4.1 JavaScript

JavaScript é uma linguagem de programação *web*, de alto nível, interpretada e de tipagem fraca [19]. É das linguagens mais utilizadas [20], na maioria dos *web sites* e navegadores modernos. Com a introdução do *NodeJs* houve a possibilidade da ascensão de criação de serviços *back-end* em JavaScript.

Até então, o JavaScript segue os padrões e as normas estabelecidas pelo comité, *European Computer Manufacturers Association* (ECMA).

4.2 ReactJS

ReactJS é uma biblioteca JavaScript, desenvolvida em colaboração entre o Facebook e o Instagram, utilizada para a criação de *user interfaces (UI)* [21]. É uma biblioteca de código aberto e destina-se a aplicações e *sites* de página única (SPA). Uma das características desta biblioteca é tornar o desenvolvimento escalável, interactivo e de fácil manutenção. No padrão *Model-View-Controller*, ReactJS é considerado o *View* e orientado a componentes.

Outro conceito chave do ReactJS é o virtual *Document Object Model* (DOM). O virtual DOM é altamente eficiente em estruturas declarativas. Em Reactjs todas as alterações de UI são realizadas em primeiro lugar no virtual DOM e, em seguida

estas alterações são comparadas e aplicadas no DOM real. A sincronização do virtual DOM com DOM real é realizada pela biblioteca ReactDOM.

Todavia, na maioria das páginas web modernas, a interação é feita directamente no DOM real, com a consequência de que o desempenho e o carregamento de páginas HTML fica mais lento.

A Figura 4.1 representa um exemplo de renderização de elemento ReactJS e seus descendentes (*children*) no virtual DOM através da função render do ReactDOM.

```
import ReactDOM from "react-dom";

function Welcome() {
  return <h1>Benvindo ao ISEC</h1>
}

ReactDOM.render(<Welcome/>,document.getElementById("app"));
```

Figura 4.1: Exemplo de renderização de elemento ReactJS no DOM

Para a criação de componentes UI, o ReactJS utiliza a extensão JSX, uma sintaxe JavaScript baseada em tags semelhantes ao código HTML. Esta sintaxe é recomendada, mas não é um requisito obrigatório.

4.2.1 MUI

O MUI que significa *Material to build User Interface (UI)* é o *framework* de UI mais popular no desenvolvimento de aplicações em ReactJS. É um projeto de código aberto, licenciado sob a licença MIT, que implementa as especificações do *Material Design da Google* [22].

O MUI segue os padrões do ReactJS e fornece uma biblioteca bastante robusta, personalizável e dispõe de componentes básicos aos avançados para construir UIs. Na *core* do pacote estão disponíveis os principais componentes de UI, como por exemplo: *Layout*, *Inputs*, *Navigation*, entre outros. Por defeito, os componentes de MUI são estilizados por meio de estilos, *css-in-js*, umas das técnicas modernas de estilização de componentes.

Na Figura 4.2 é apresentado um exemplo de um botão reutilizável, exposto pelo MUI. Este consiste na alteração dinâmica do *design* do componente através da

propriedade/atributo cor. Além desta propriedade existem outras que podem ser alteradas/personalizadas, como por exemplo: a variante, o icon, entre outros.

(a) Botão com cor primário

(b) Botão com cor sucesso

Figura 4.2: Exemplo de um botão, com alteração da propriedade cor

O MUI permite adaptar os layouts em diversos dispositivos, sejam eles *Desktops* ou *smartphones*. Nas Figuras 4.3 e 4.4 apresentam-se exemplos de disposição de componentes em dispositivos de tamanho médio e grande (*Desktop*) e de tamanho pequeno (*smartphones*), respetivamente. Neste caso recorrendo ao componente *Grid*, é possivel fazer-se esta adaptação.

Figura 4.3: Exemplo de *layout* responsivo em dispositivos médios e grandes.

Figura 4.4: Exemplo de *layout* responsivo adaptável aos dispositivos pequenos.

4.3 NodeJS

O NodeJS é um ambiente de desenvolvimento, ou simplesmente, uma plataforma de execução e interpretação de códigos *JavaScript*, criada sob o mecanismo V8 do Google Chrome. Utiliza-se um modelo Entrada/Saída não bloqueante, assíncrono e orientado a eventos. Além disso, é multiplataforma, de código aberto (*open-source*) e de único *thread* [23]. Embora só se possa realizar apenas um *thread* de cada vez, o NodeJS é muito eficiente e bastante rápido a tratar de múltiplas solicitações em simultâneo, devido à sua natureza orientada a eventos. No entanto, escrever uma API apenas utilizando NodeJS é difícil e desnecessário. Atualmente, existem vários *Frameworks* para servidor em NodeJS, sendo o mais popular o ExpressJS.

4.4 GraphQL

GraphQL é uma linguagem de consulta para APIs [24], criada pelo Facebook, em 2012, e lançada publicamente em 2015. Além disso, o GraphQL oferece um serviço *runtime* do lado do servidor para executar consultas de dados, utilizando o tipo de sistema (*type system*) definida no API.

Ultimamente, GraphQL é considerada a tecnologia mais popular para a construção e manipulação de APIs. As principais vantagens que este API possui são: apresentar uma sintaxe amigável e intuitiva, ser declarativa e flexível e também, permitir uma fácil comunicação entre o cliente e os serviços *back-end*.

Em geral, GraphQL possui três estruturas principais: *query, mutation e subscription*. Aquando das operações CRUD, a leitura é realizada pela *query* (método GET do HTTP). Por sua vez, as mutações equivalentes às operações criar, atualizar e apagar, realizadas pelos métodos *CREATE, UPDATE e DELETE* do HTTP, respetivamente.

A Figura 4.5 representa a arquitetura de comunicação e serviços GraphQL.

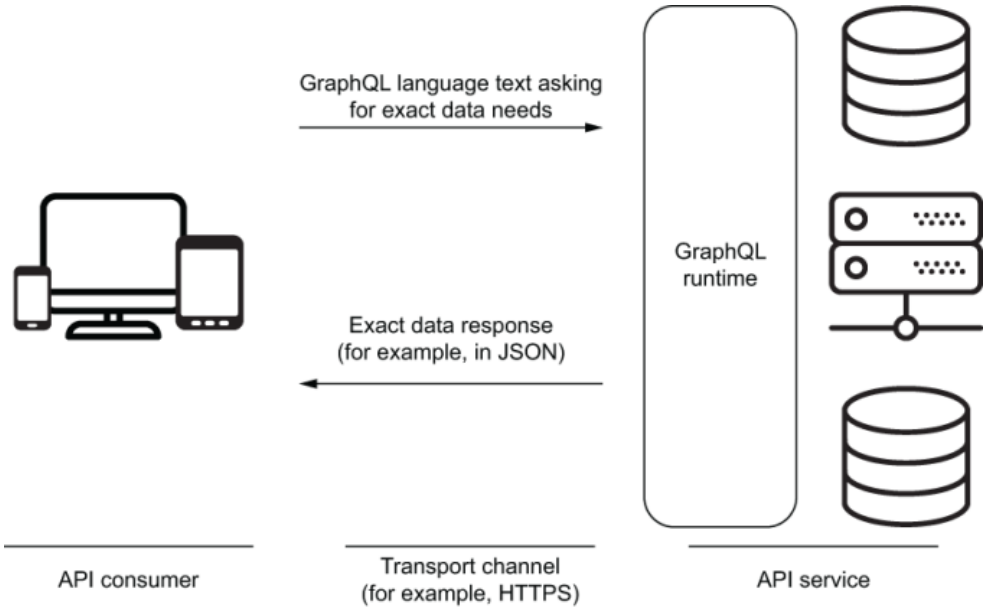


Figura 4.5: Exemplo de arquitetura de GraphQL [4].

4.4.1 *Schema Definition Language - SDL*

O *Schema Definition Language* (SDL) é a forma mais concisa de especificar esquemas GraphQL. A sintaxe é uma das especificações do GraphQL que permite definir a estrutura de dados de uma API. Na Figura 4.6 é apresentado um exemplo de definição de esquema GraphQL. No esquema do tipo (*type*) *Team*, são definidos os atributos ou ainda outro tipo de objeto que o definem. Os atributos podem ainda ser definidos como obrigatórios, quando se faz o uso do ponto de exclamação (!). Podem ainda, definir-se atributos como sendo outro tipo de objecto, como por exemplo, neste caso, o atributo *techs* retorna uma lista tipo *Tecnhician*.

```
type Team {
  cod:ID
  name:String!
  owner:Technician
  techs:[Technician!]
}
```

Figura 4.6: Exemplo de SDL.

É de realçar ainda, que os atributos podem ou não ter argumento(s). Considera-se que o objeto tipo *Team* deverá retornar à lista de *technician* definido pelo cliente. Neste caso define-se como argumento no atributo *techs*, o limite, que é responsável

por definir a lista fixa de técnicos a retornar, exemplo (techs(limite:Int):[Technician]).

4.4.2 Consultas com GraphQL

O GraphQL permite uma manipulação de dados bastante flexível e permite ao cliente definir quais os atributos necessários e retornados por cada tipo de objeto. As consultas, bem como as mutações são definidas como o tipo de objeto e pelas palavras chaves *Query* e *Mutation* respetivamente. Na Figura 4.7 apresenta-se um exemplo de consulta GraphQL.

```
query {
  getTeams {
    cod
    name
    techs {
      userName
    }
  }
}
```

Figura 4.7: Exemplo de consulta com GraphQL.

4.4.3 *Mutations* com GraphQL

O tipo *Mutation* permite realizar operações *CRUD*, criar, apagar e atualizar. É assim designado por causar modificações. Tal como as consultas, este é definido pela palavra chave *Mutation*. A Figura 4.8, ilustra a mutação para a criação de uma nova *Team*, designada pela *createTeam*. Neste caso, o argumento ou a entrada, *name* é obrigatório.

```
type Mutation {
  createTeam(name:String!, owner:String):Team
}
```

Figura 4.8: Exemplo de mutação com GraphQL.

Em casos avançados da utilização da API, o GraphQL possui o tipo de objecto *Input*, para definir as entradas de dados, como mostra a Figura 4.9.

```
input TeamInput {
  name:String!
  owner:String
}

type Mutation {
  createTeam(input:TeamInput!):Team
}
```

Figura 4.9: Exemplo de mutação GraphQL.

4.4.4 Apollo Server

O Apollo Server é uma biblioteca de código aberto que permite criar serviços GraphQL em JavaScript. Assim sendo compatível com especificações do GraphQL.

Das principais vantagens da utilização do Apollo Server destacam-se as seguintes:

- Excelente Ecossistema: O Apollo Server oferece além de serviços GraphQL, oferece também a compatibilidade com arquitectura REST;
- Documentação: Existe atualmente bastante informação sobre Apollo Server. A sua biblioteca está em constante actualização e bem documentada;
- Fácil configuração e compatibilidade: O Apollo Server permite a criação de um servidor compatível com qualquer cliente GraphQL.

4.5 Protocolos de comunicação

4.5.1 Protocolo HTTP

O *Hypertext Transfer Protocol* (HTTP) é um protocolo de comunicação da camada de aplicação do modelo *Open System Interconnection* (OSI) para sistemas de informação distribuídos e colaborativos [25]. Este protocolo é a base para a comunicação de dados via *World Wide Web* (WWW), permitindo transferir e visualizar páginas *Web* em formato de *Hyper Text Markup Language* (HTML).

O protocolo HTTP utiliza o método *request-response* baseado na arquitetura cliente/servidor, sendo cliente o *Web Browser* e o servidor *Web Server*.

O protocolo HTTP define um conjunto de métodos de requisição que indicam a ação a ser realizada no recurso especifico, sendo os principais métodos de requisição os seguintes:

- **GET** - é o método mais comum e é utilizado para recuperar dados de um servidor através de um URL sem causar qualquer alteração;
- **POST** - o método POST é utilizado para enviar/submeter dados para o servidor, causando mudança no seu estado;
- **PUT** - semelhante ao método POST, este permite modificar diretamente um recurso existente ou criar-lo novamente;
- **DELETE** - é o método que permite apagar um recurso ou conjunto de recursos;
- **OPTIONS** - representa os métodos HTTP que estão disponíveis para um determinado URL;
- **HEAD** - é idêntico ao método GET, sendo este utilizado para obter parte de dados por meio do cabeçalho da resposta. Frequentemente utilizado para testar links de hipertexto quanto à validade, acessibilidade e modificações recentes [26].

Para além dos anteriores descritos, existem também os métodos CONNECT, TRACE e PATCH.

4.5.2 Protocolo MQTT

O protocolo *Message Queuning Telemetry Transport* (MQTT) foi inventado em 1999 por Andy Stanford-Clark (IBM) e Arlen Nipper (Eurotech) [27]. Trata-se de um protocolo de transporte de mensagens entre cliente e servidor através do modelo *publish/subscribe*. É extremamente simples, flexível e leve, projetado para dispositivos restritos e redes de baixa largura de banda e alta latência, nomeadamente no contexto de comunicação *Machine-to-machine* (M2M) e *Internet of Things* (IoT) [27, 28].

O MQTT utiliza o protocolo TCP/IP (*Transmission Control Protocol/Internet Protocol*) ou qualquer protocolo que fornecer conexões bidireccionais, sem que haja

quaisquer perdas de informações. Naturalmente, para a conexão criptografada e mensagens sensíveis é utilizado o protocolo de segurança *Transport Layer Security* (TLS), também suportado pelo MQTT.

Em geral, o protocolo MQTT define dois tipos de entidades de rede, o servidor (denominado de *Broker*) e os clientes que podem ser dispositivos, sensores ou qualquer sistema que possa interagir com o *Broker* e receber mensagens, como representado na Figura 4.10. O *Broker* MQTT é um servidor que serve de elo entre clientes e, é responsável pela receção e filtragem de mensagens e, posteriormente o envio destas mensagens para os respetivos clientes.

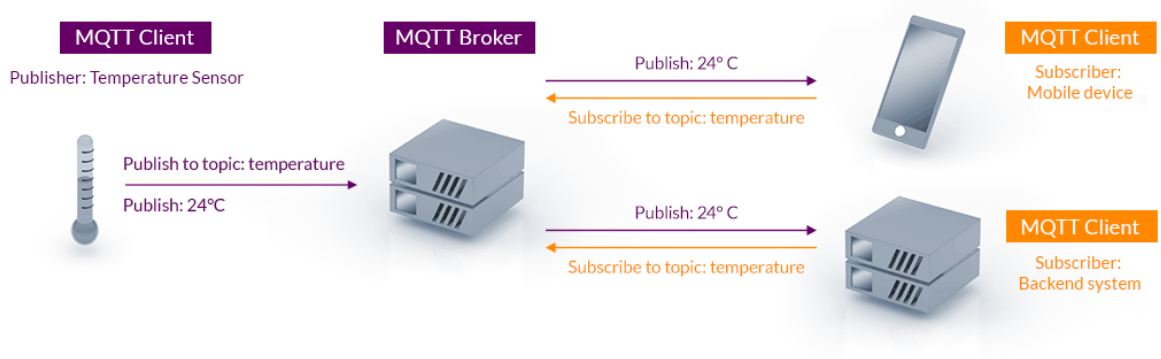


Figura 4.10: Arquitetura *publish/subscribe* MQTT [27]

No esquema representado na Figura 4.10, os dois clientes MQTT (*Smartphone e Backend System*) após subscreverem ao tópico *"Temperature"* recebem todas as mensagens publicadas pelo cliente *Temperature Sensor* através do *Broker*.

No protocolo MQTT, as mensagens enviadas são associados a um tópico e a um *Quatity of Service* (QoS) que permite garantir a entrega de uma determinada mensagem ao cliente. No entanto, existem 3 níveis de *Quatity of Service*, tais como:

- **QoS 0** - a mensagem é enviada uma única vez sem qualquer confirmação do *Broker*. Também conhecida pelo modo de envio *Fire and Forget*;
- **QoS 1** - este nível garante que a mensagem seja entregue pelo menos uma vez, sendo necessário a sua confirmação pelo *Broker*. Este modo de comunicação, persistente, designa-se por *PUBACK*;
- **QoS 2** - este nível é considerado o mais seguro do MQTT, mas também o mais lento. Neste nível, a mensagem será recebida apenas uma vez pelo

receptor. Utiliza pelo menos dois pares de transmissões entre o cliente e o *Broker*(denomindo de *handshake*).

4.6 Conclusão

Este Capítulo apresentou as tecnologias modernas de desenvolvimento de aplicações *Web*. De uma forma mais detalhada são apresentadas a tecnologia de *FrontEnd - ReactJS* e a metodologia de troca de informação assente em *GraphQL*. Foram também descritos os protocolos de comunicação HTTP, bem como, o protocolo utilizado em IoT denominado por MQTT.

Capítulo 5

Arquitetura do Sistema

Nos primeiros capítulos fez-se um estudo teórico do MIT, abordando-se os fundamentos teóricos, as principais avarias que podem ocorrer nestes motores, bem como as condições de diagnóstico de falhas nestas máquinas. Por outro lado, no capítulo anterior abordaram-se os principais conceitos e ferramentas de desenvolvimento de *software*. Assim, este capítulo encontra-se dividido em duas seções. Na seção 5.1 apresentam-se as ferramentas seleccionadas para a implementação do *hardware* e na seção 5.2 estão apresentadas as tecnologias e ferramentas utilizadas para a implementação da aplicação.

5.1 *Hardware*

5.1.1 Raspberry Pi

O Raspberry Pi, representado na Figura 5.1, é um computador de placa única *single-board computer*, de tamanho reduzido, desenvolvido no Reino Unido pela Fundação Raspberry Pi [29]. Originalmente, desenvolvido com o objectivo de promover o ensino da ciência de computação básica ao nível escolar. Atualmente, é notável a sua utilização nas diversas áreas, desde aplicações industriais, *Internet of Things* (IoT) para automação residencial, entre outros.

Raspberry Pi ligado aos periféricos (monitor, rato e teclado) executa qualquer tipo de ação de forma semelhante a outros computadores.

Destacam-se as principais vantagens da sua utilização tais como: apresentar baixo custo, portabilidade e muitos recursos que permite ao utilizador configurá-lo de diversas maneiras.

Figura 5.1: Representação de Raspbery Pi modelo 3B.

De uma forma geral, o Raspberry Pi é essencialmente constituído por duas partes: o *hardware* e o *software*, utiliza como sistema operativo o Raspberry Pi OS, baseado em distribuição Debian e otimizado para o *hardware* Raspberry Pi.

5.1.2 Módulo sensor de tensão

A medição da tensão do motor será realizada através de um sensor de tensão CA, representada na Figura 5.2. Este sensor apresenta elevada precisão de medida, fácil montagem e é aplicado essencialmente para equipamentos eléctricos e em medições de energia eléctrica. As especificações deste sensor encontram-se resumidas na Tabela 5.1.

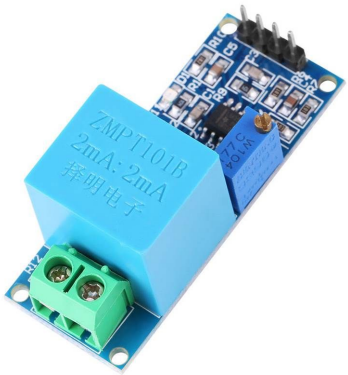


Figura 5.2: Sensor de tensão CA.

Corrente primário e secundário	2mA
Temperatura de operação	-40ºC a +130ºC
Frequência de operação	50/60Hz
Nível dielétrica	3000VAC/min
Linearidade	$\leq 0.1\%$
Resistência de isolamento	$> 100M\Omega$

Tabela 5.1: Especificações do sensor de tensão

5.1.3 Módulo sensor de corrente

Para a medição da corrente consumida no motor foi adquirido um transformador de corrente trifásico, como representado na Figura 5.3. Trata-se de um módulo não invasivo, de alta rigidez dielétrica e, destina-se para a medição de corrente nas três fases do MIT. Na tabela 5.2 encontram-se ilustradas as principais caraterísticas técnicas.

Figura 5.3: Sensor de corrente toroidal

Material	Nonocristalino
Temperatura de operação	-35ºC a +45ºC
Frequência de operação	50/60Hz
Rigidez dielétrica	3.5KV 50Hz 1min
Linearidade	$\leq 0.2\%$

Tabela 5.2: Caraterísticas técnicas do módulo de corrente

5.2 Metodologias de desenvolvimento da aplicação

Nesta secção é feita a análise e o planeamento para o desenvolvimento da aplicação, quer no lado do cliente *(front-end)* quer no lado do servidor *(back-end)*. Para o desenvolvimento da aplicação foi escolhida a linguagem de programação JavaScript, visto esta poder ser utilizada em ambas as partes da aplicação.

No contexto de análise, fez-se um levantamento dos principais requisitos da aplicação, com o objectivo de a adequar no âmbito do projeto. Assim, os principais requisitos da aplicação a desenvolver foram os seguintes:

- Registo e autenticação na aplicação através do *email* ou nome;
- Edição de informações de utilizador;
- Visualização de *gadgets* principais (disponibilidade de motores ou sensores, canais de placa de aquisição disponíveis, entre outros);
- Criação/Edição de motor e visualização individual de cada motor permitindo a análise das suas grandezas eléctricas ao longo do tempo, de tal forma que se possa realizar o diagnóstico;
- Configuração/comunicação com os serviços *web* ou API. No caso do Raspberry Pi, a comunicação é realizada através do protocolo MQTT ou outro similar;
- Realização de monitorização em tempo real ou a partir de ficheiros que possuem os dados simulados.

Nas secções 5.2.1 e 5.2.2 são abordadas as tecnologias e/ou as dependências, de preferências *open-source*, selecionadas para o desenvolvimento de *front-end* e *back-end*, respectivamente, de tal forma a colmatar e validar os requisitos anteriormente apresentados.

5.2.1 *Front-end*

A aplicação *web (front-end)* foi desenvolvido recorrendo à tecnologia ReactJS. Toda a interface gráfica foi implementada com auxilio da biblioteca MUI. Para comunicação com o *hardware* (Rasperry Pi) através do protocolo MQTT, escolheu-se o cliente MQTT para JavaScript (MQTT.js).

A tradução da aplicação web foi realizado pelo *framework* de internacionalização *i18next* com integração à ReactJS. Este *framework* permite a configuração da lógica de tradução para diversos idiomas e suporta a extensão JSON. Além disso, possui *plugin* de deteção automaticamente de idioma.

Os pedidos HTTP da aplicação *web* com o servidor são realizados pela API GraphQL. Esta configuração é implementada pelo *Apollo client*.

Em seguida são descritas as tecnologias difundidas que mais de adequam na integração com a biblioteca ReactJS.

Apollo Client

O *Apollo Client* é um cliente GraphQL padrão de código aberto para aplicações *Web, iOS e Android*. Trata-se de uma biblioteca JavaScript voltada para ReactJS auxiliando na comunicação HTTP e *Websocket* através da API GraphQL.

ApexCharts

O ApexCharts é uma biblioteca JavaScript de gráficos modernos e interativos para as páginas *Web* (Figura 5.4) [30]. Tem integração com os principais *frameworks* ou bibliotecas JavaScript, nomeadamente ReactJS, Vue e Angular. Esta biblioteca é flexível, versátil e de fácil configuração.

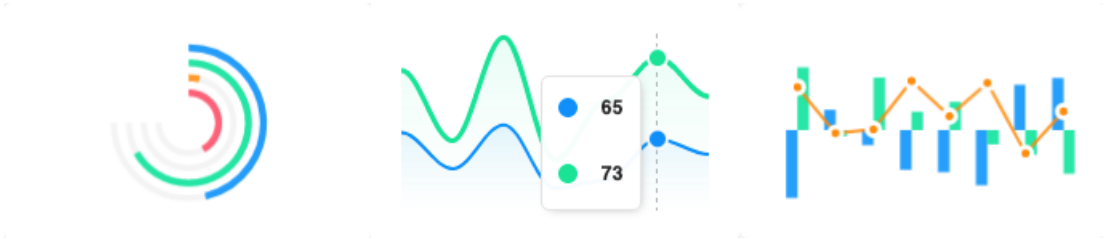


Figura 5.4: Exemplo de gráficos ApexCharts

5.2.2 *Back-end*

O servidor da aplicação web é implementado na plataforma NodeJS, recorrendo-se às tecnologias Express e *Apollo Server*. **Estas duas tecnologias são responsáveis pela a criação do servidor GraphQL, de modo a permitir a comunicação com a aplicação web através da API GraphQL**. A integração do GraphQL com o Express é conseguida através da *midleware apollo-server-express*.

A Figura 5.5, apresenta a configuração básica do servidor GraphQL com a integração das duas tecnologias (*Apollo Server* e *Express*). O servidor recebe os pedidos num único *endpoint* (/graphQL) e, estes são tratados internamente pelo *Apollo Server*.

```
//Importar módulos
import express from "express";
import { ApolloServer } from "apollo-server-express";

//Configuração do express.js
const app = express();

const serverSetup = new ApolloServer({
  payground, //Ferramenta graphql de teste
  typeDefs, // Definições de esquema GraphQL
  resolvers,// Funções/métodos para esquema fields
  context, //Função de partilha de contexto com os resolvers
  //outras opções de configurações do apollo server
})

//Integração com o express.js
serverSetup.applyMiddleware({ app, path: "/graphql" });

//Outras lógicas ...
```

Figura 5.5: Exemplo de configuração de servidor GraphQL.

Para a manipulação e armazenamento de dados em banco de dados SQL *(Structured Query Language)* escolheu-se a biblioteca do *Sequelize*. Trata-se de um ORM *(Object-relational mapping)* para plataforma NodeJS baseado em *promise* e, suporta os dialetos PostgresSQL, MySQL, MariaDB, SQLite e MSSQL. Para o projeto optou-se pela base de dados PostgreSQL. A configuração com o *Sequelize* é realizado por duas dependências, *pg-hstore e pg*.

Outras dependências não menos importantes escolhidas na implementação do servidor foram: *jsonwebtoken* - gerar, verificar e descodificar JWT; *dotenv*-configuração das variáveis ambiente (env); entre outras.

5.3 Base de Dados

A base de dados escolhida para o armazenamento de dados foi o PostgreSQL. Optou-se por esta base de dados, uma vez que, é de código aberto sem quaisquer custos de licenciamento, simples e flexível.

Atendendo aos requisitos do sistema e a todas as informações necessárias a serem guardadas na base de dados, foram criadas um conjunto de tabelas com os seus respetivos atributos. Na Figura 5.6, está representado o diagrama Entidade-Relacionamento da estrutura lógica da base de dados.

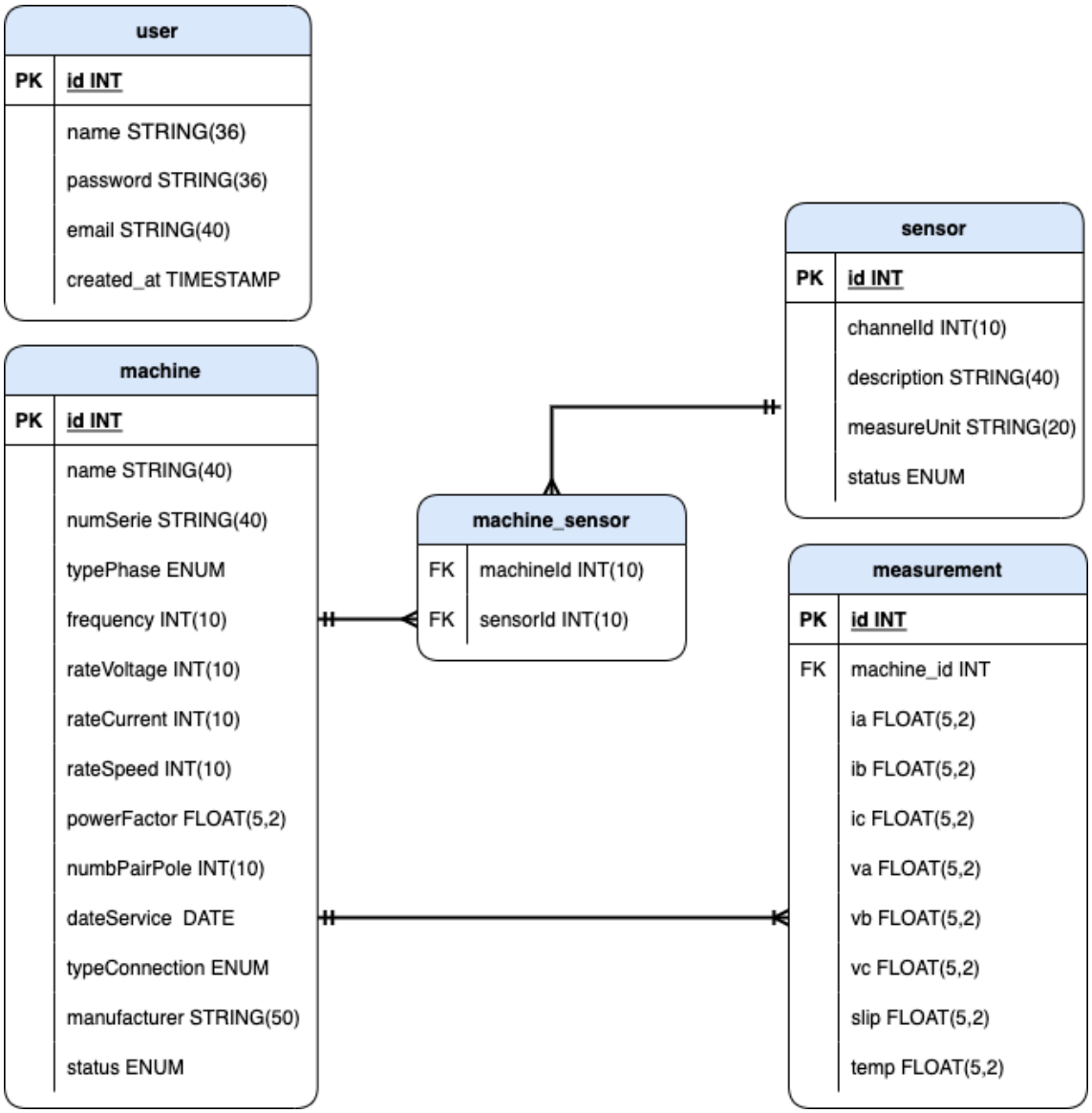


Figura 5.6: Diagrama Entidade-Relacionamento.

5.4 Conclusão

Este Capítulo apresentou os requisitos necessários à construção do sistema, quer do ponto de vista de *software*, quer do ponto de vista de *hardware*. Além disto fez-se uma descrição dos módulos de sensores adotados e apresentou-se a estrutura de base de dados a utilizar.

Capítulo 6

Conceção e implementação

O presente projecto tem como principal objectivo a conceção e implementação de um sistema que permite o diagnóstico de falhas em MITs. Neste capítulo é apresentada toda a implementação do *hardware* e da aplicação de forma a validar a sua aplicabilidade no âmbito do projeto.

6.1 Marcadores Visuais

Os marcadores visuais são uma solução utilizada na área de Tecnologia de informação destinados para a codificação de informações nos domínios da robótica, virtual e da realidade aumentada. Atualmente estes são fortemente utilizados, por exemplo, na manipulação de imagens, em códigos de barras, QR Codes, entre outros.

No âmbito do projeto, utilizou-se a biblioteca de visão computacional denominada por OpenCV para implementar um sistema marcador visual que permite o reconhecimento de dígitos. Assim, a ideia da utilização deste marcador surgiu da necessidade de efetuar o reconhecimento de marcas presentes em motores (dígitos ou objetos), afim de poder verificar e identificar cada motor em análise. Na Figura 6.1 apresentam-se os resultados obtidos num dos testes. Os restantes resultados encontram-se em Anexo B.

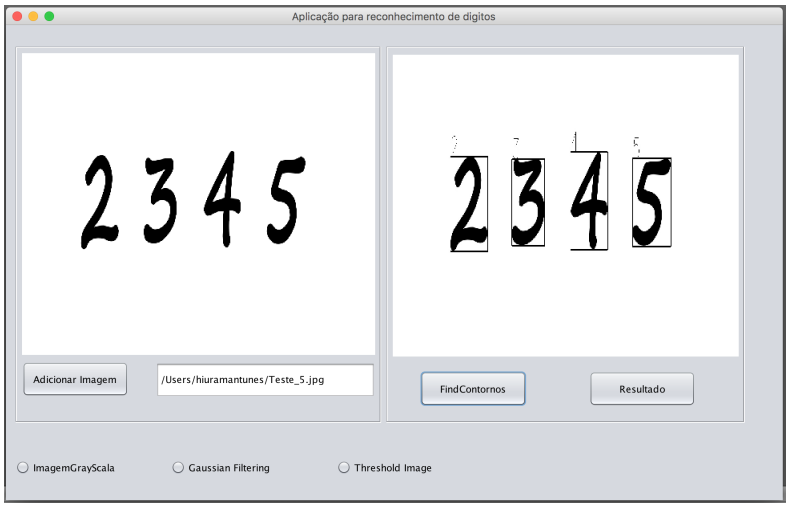


Figura 6.1: Aplicação de Reconhecimento de dígitos.

6.2 Descrição geral do sistema

Na Figura 6.2 está exemplificada a visão geral da arquitetura do sistema de diagnóstico de falhas em motores de indução. Este, encontra-se dividido em três partes, o *frontend*, o *backend* e a tecnologia IoT, alojado no Raspberry. A conectividade entre o *frontend* e o Raspberry Pi é realizada através do protocolo MQTT (*subscribe/publish*).

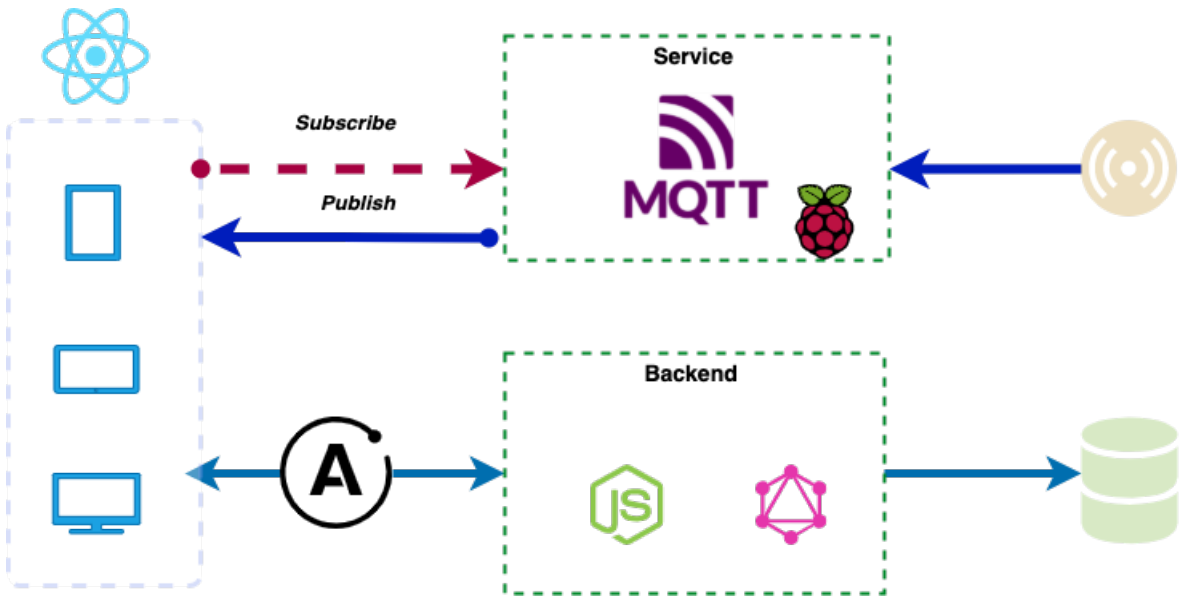


Figura 6.2: Arquitetura do sistema.

Tal arquitetura está definida para que sejam facilmente integrados novos componentes ao sistema de forma a permitir uma maior usabilidade, flexibilidade e escalabilidade.

6.3 Sistema de Aquisição de dados

6.3.1 Especificações Gerais

Na Figura 6.3 apresenta-se o sistema de aquisição de dados composto por uma placa ADC conectada aos pinos GPIO *(General Purpose Input/Output)* da interface de *hardware* Raspberry pi através de conectores *jumpers*. Na placa de aquisição estão expostos dois módulos ADC de aquisição de dados, tendo cada módulo 8 canais com uma resolução de 16 bits.

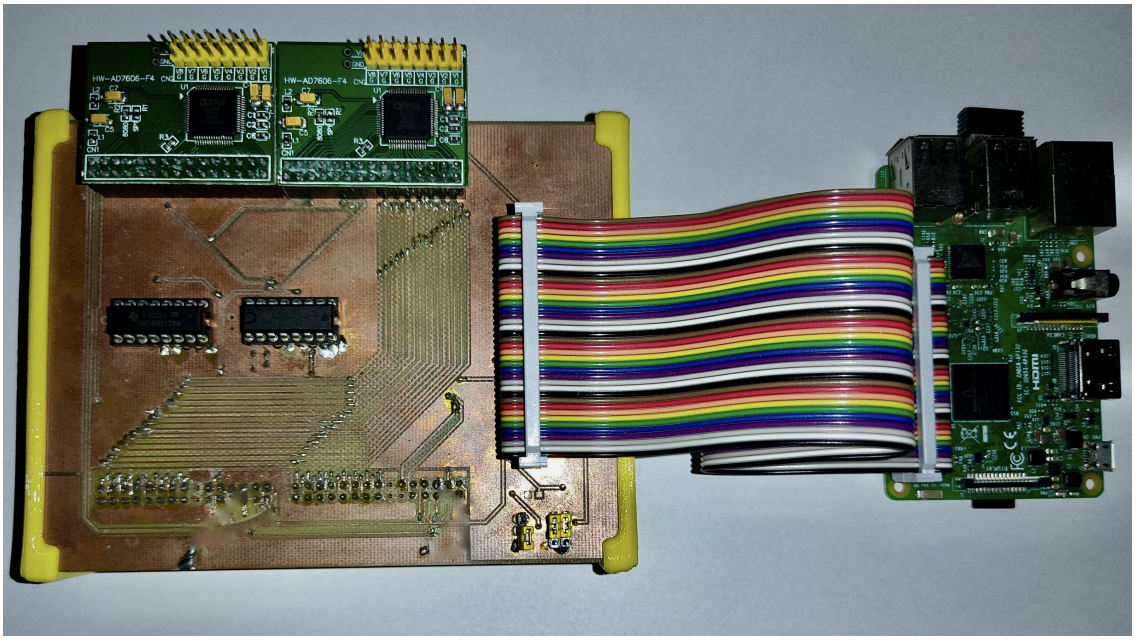


Figura 6.3: Módulo de aquisição de dados.

6.3.2 Unidade de processamento

Como ilustrado na Figura 6.3, o Raspberry Pi está encarregue de todo o processamento ao nível do *hardware*. Dos principais processos realizados neste *hardware*, destacam-se os seguintes:

- **Inicialização/Execução do ADC**: Escrita em linguagem de programação C, permite a aquisição de dados provenientes dos sensores ou dispositivos conectados aos canais do módulo ADC;

- **Processamento de sinais**: Neste processo utiliza-se a linguagem Python e nele encontra-se toda a lógica de processamento de sinais, nomeadamente a

análise de sinais no domínio de frequência através de FFT, a determinação ou cálculo dos parâmetros do motor entre outros;

- **Broker Mosquitto**: Trata-se de um servidor MQTT de código aberto, responsável pela comunicação entre o *hardware* e o *frontend* atráves do modelo *publish/subscribe*.

6.3.3 Broker MQTT

O Broker MQTT é configurado no Raspberry Pi e possibilita a troca de mensagens entre este e a aplicação *web*. O broker serve de intermediário na comunicação entre os sensores e os clientes que se designa de *publisher* e *subcriber*, respetivamente. Além disto, é também responsável por gerir todas as publicações e subscrições do protocolo MQTT.

Para habilitar a execução do Broker Mosquitto em *background* é necessário inicializar o seu serviço através do comando *"systemctl start mosquitto"*. Após a configuração do Broker, qualquer cliente poderá se ligar ao mesmo.

A conexão da aplicação *web* desenvolvida é feita como mostra a Figura 6.4. Esta ação é realizada na página *Dashboard* da aplicação.

Figura 6.4: Conexão com o serviço Broker Mosquitto.

6.4 Aplicação Web

A aplicação *Web* desenvolvida é constituída pelos módulos principais, sendo eles: **autenticação, página inicial, monitorização e dispositivos**. Nas próximas subsecções estão descritos sumariamente estes módulos. No entanto, no Anexo A estarão também disponíveis para visualização os demais.

A aplicação *web* tem suporte de comunicação com os serviços ou API através de dois protocolos, sendo eles o HTTP e o MQTT. A comunicação pelo protocolo MQTT permite a conexão com o *hardware* para a realização de tarefas de *publish/subscrive* pelo que permite monitorização dos motores em tempo real. A comunicação entre a aplicação e o servidor GraphQL é realizada através do protocolo HTTP.

6.4.1 Módulo de autenticação

A Figura 6.5 apresenta a página de autenticação que permite o acesso à aplicação. Para iniciar a sessão na plataforma é necessário preencher os campos "email"e "palavra-passe". Esta página permite também aceder à página de registo de utilizador. Por ser um projeto académico não existe nenhum tipo de permissão atribuído aos utilizadores.

Figura 6.5: Página de Autenticação.

Neste módulo, a autenticação atende aos seguintes passos:

- No lado do cliente são verificados e validados os campos preenchidos (email e palavra-passe) antes do envio para o *back-end*. Após a validação, é enviada uma requisição para o *back-end* para que este possa efetuar a autenticação;

- Caso o *back-end* verifique que os dados submetidos estejam corretos é gerado e enviado um *JSON Web Token*. Este, contém no seu *payload* dados referentes à autenticação do utilizador, bem como *token* e o tempo de expiração. O *token* é adicionado no cabeçalho do HTTP na forma de *Autorization* atráves do *Apollo Client* nas futuras requisições.

6.4.2 Página *Dashboard*

De acordo com a Figura 6.6, na Página *Dashboard* ilustram-se de um modo resumido, as informações relativas aos dispositivos, sensores, canais do ADC e o estado de configuração do serviço Broker instalado no *hardware* Raspberry Pi. Para além disto, é também possível estabelecer a ligação da aplicação com o serviço *Broker* através do botão "Criar nova Conexão". Este, por sua vez, abre um *popup* de conexão do serviço.

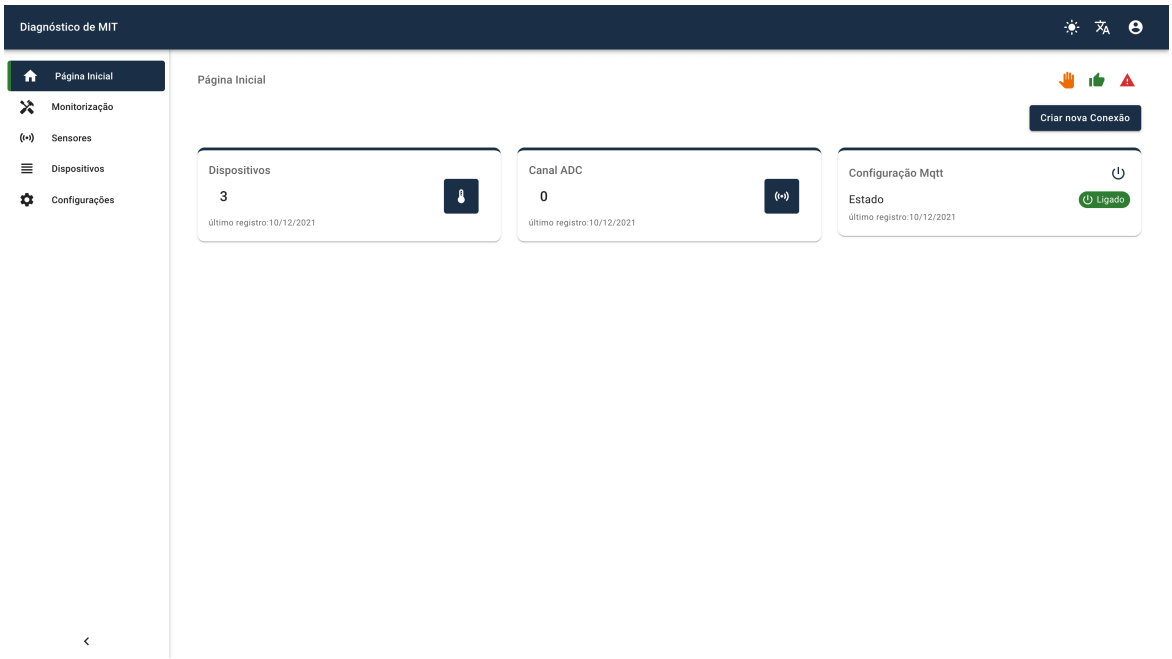


Figura 6.6: Vista *Dashboard*

6.4.3 Página de Monitorização

Esta página permite a análise e o diagnóstico de motores propriamente dita. A monitorização do motor pode ser realizada em tempo real ou através da importação de um ficheiro com os dados previamente adquiridos. Consoante a seleção anterior é possível analisar as grandezas elétricas do motor em causa, nomeadamente as correntes, as tensões, o escorregamento e as condições do estator e do rotor. Para além disso é também possível visualizar detalhadamente o comportamento do motor através os seguintes gráficos:

- Gráfico de correntes do estator no domínio de tempo;
- Gráfico do Espetro de frequência da corrente numa das fases;
- Gráfico de Análise do vetor de Park

Para a análise em tempo real é ainda necessário ao utilizador o preenchimento dos campos "Selecionar Motor", "Número Amostra"e "Frequência Amostragem"de tal forma que, a análise seja feita em conformidade. Quando estes campos forem devidamente preenchidos e através do botão "Aplicar", este publica para o tópico "service/paramSetup"do serviço Broker através do protocolo MQTT. Em seguida, são subscritos para o mesmo tópico os dados processados pelo *hardware*. Na Figura 6.7 é possível visualizar os dados referentes à análise previamente realizada.

Figura 6.7: Vista de Monitorização em tema escuro.

6.4.4 Página Dispositivos

A página "Dispositivos"permite ao utilizador verificar a lista de motores registados, analisar e acompanhar detalhes individuais de cada motor e ainda registar ou editar um novo motor, como ilustram as Figuras 6.8, 6.9 e 6.10. Tem ainda a funcionalidade de *search* permitindo a procura de motores registados na base de dados.

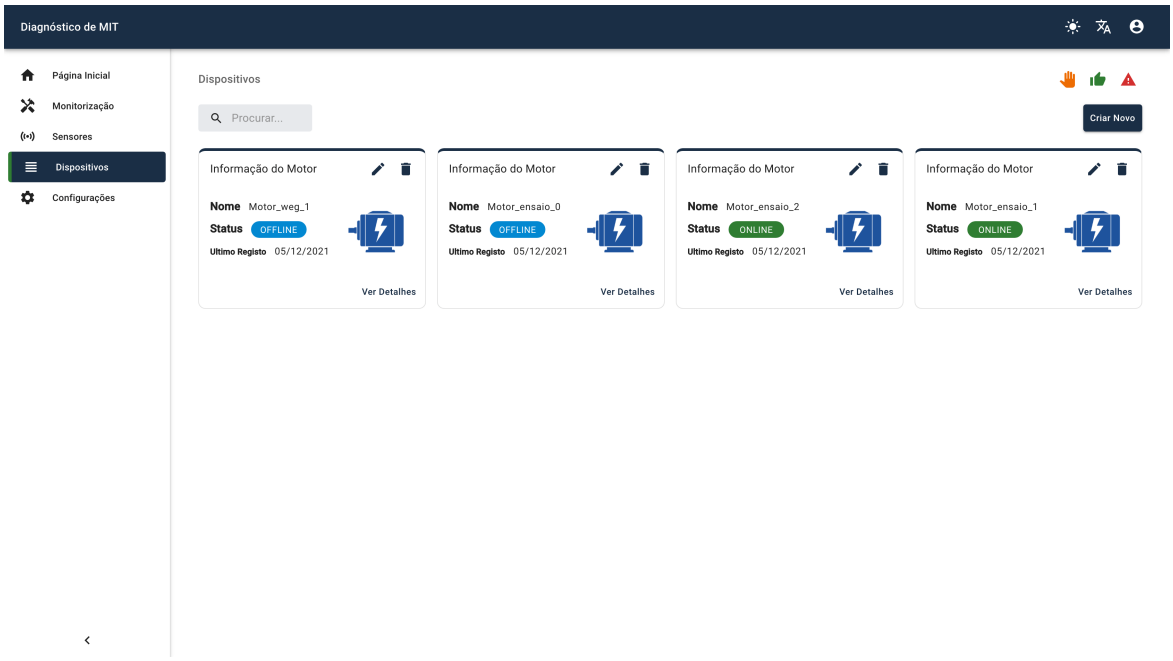


Figura 6.8: Vista de lista de motores registados

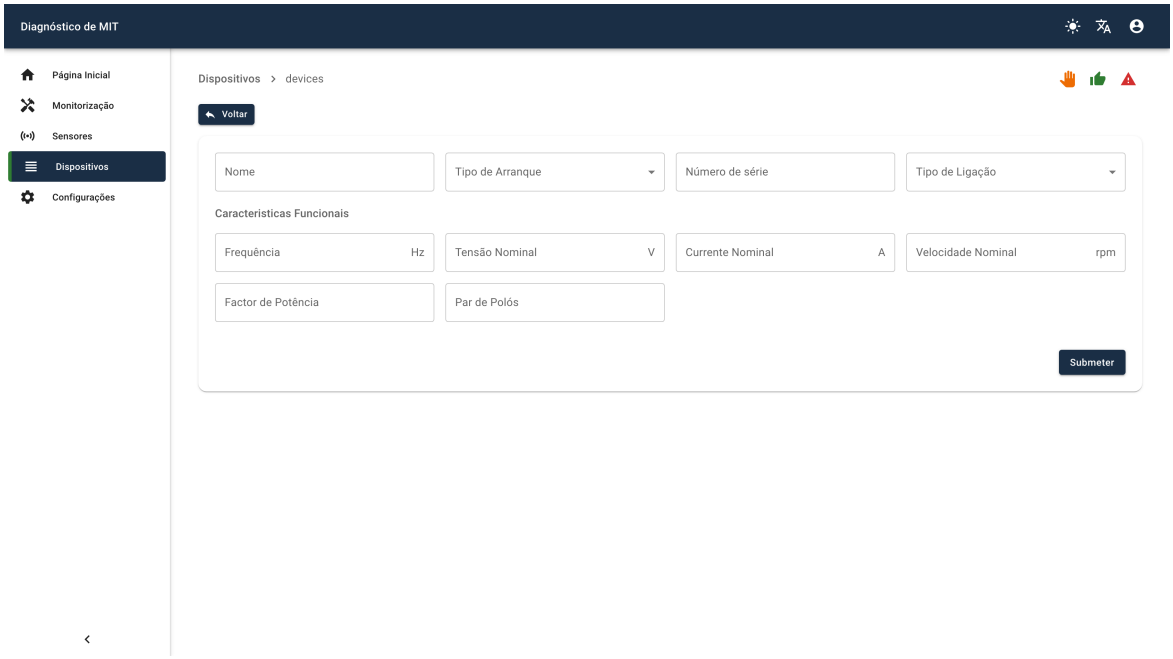


Figura 6.9: Vista de criação de novo motor

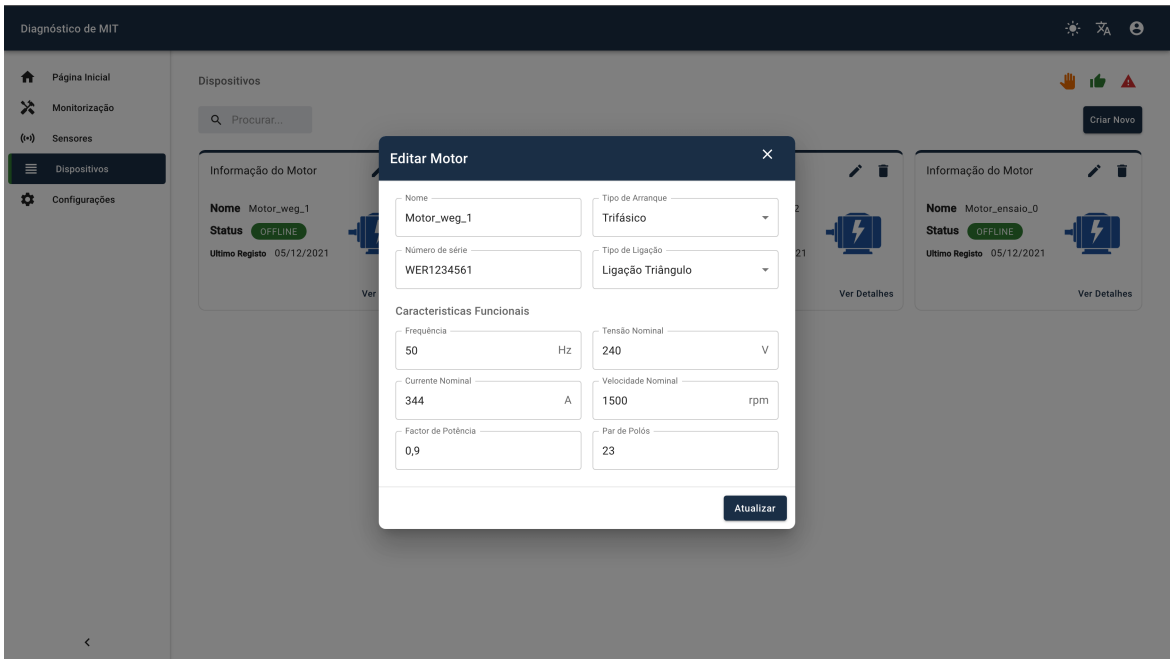


Figura 6.10: Vista de edição do motor

6.5 *Back-end*

No *back-end* está toda a lógica de configuração e conexão da base de dados. Esta também permite a interação com a aplicação *web* e é responsável pelo manuseamento de ficheiros ou imagens alojados na pasta *public*.

A configuração da base de dados é feita atendendo ao diagrama de entidades de modelos representado na seção 5.3. A definição de cada um dos modelos da base de dados está definida na pasta "database/models"do servidor.

Inicialmente é feita a configuração do módulo *Sequelize*, onde é definido o nome da base de dados (database), o utilizador(username) e a palavra-passe(password). É ainda necessário definir as propriedades do *host*, bem como o dialeto a utilizar e outras opções consoante as necessidades. Na Figura 6.11 ilustra-se um exemplo de configuração possível. As propriedades anteriormente descritas, são obrigatórias para proceder à conexão de qualquer base de dados utilizando o módulo *Sequelize*.

```
const sequelize = new Sequelize("database", "username", "password",
{
  host:"localhost",
  dialect:"postgres" //'mysql' | 'mariadb' | 'mssql' | 'postgres'
  //outras opções
});
```

Figura 6.11: Configuração da base de dados com o *Sequelize*

6.6 Conclusão

Este capítulo apresentou todo o processo de desenvolvimento do sistema, desde a aplicação *web* ao servidor/serviço. Foram descritas as funcionalidades da aplicação e dos módulos que o compõem, bem como as funcionalidades do sistema de aquisição de dados, nomeadamente da configuração do serviço Broker Mosquitto (elo de ligação entre clientes e *subscribes*).

Capítulo 7

Conclusões e Trabalho Futuro

O presente documento ilustra o trabalho realizado na construção de um sistema de análise de falhas em MIT para análise preditiva de eventuais futuras avarias.

O sistema é constituído por duas peças de tecnologia:

- *hardware* - sistema barato de aquisição de dados recorrendo a um conjunto de dois ADCs de 8 entradas em paralelo, o que permite adquirir 16 variáveis de forma síncrona a uma frequência máxima de 200 MSPS. Para o efeito foi desenvolvida uma placa que permite a gestão dos sinais (realizada noutro trabalho), que se encontra interligada ao Raspberry Pi;

- *software* - no caso do *software* podemos dividir em três partes importantes:

  - *software* relacionado com a aquisição: desenvolvido em baixo nível na línguagem C, permite controlar a aquisição dos sinais de forma adequada, de acordo com as solicitações externas;

  - algoritmos inteligentes: desenvolvimento em *Python* dos algoritmos existentes e descritos nos diversos trabalhos científicos, relacionados com a análise de avarias em MIT (parte do trabalho realizado noutro trabalho de Mestrado, tendo sido melhorado e adicionadas alterações e algoritmos);

  - *interface Web*: desenvolvido na totalidade, no presente trabalho, para permitir a interligação entre as diferentes vertentes do sistema.

O sistema de aquisição está operacional e pode ser utilizado em laboratório, havendo no entanto a necessidade de o acondicionar numa caixa para melhor proteção do mesmo e do Raspberry Pi. Há ainda a necessidade de colocar fichas para todas

as conexões necessárias. Salienta-se que o sistema de aquisição custa 14 Euros (um ADC de 8 canais) + 18 Euros (um TI de 3 canais) + 39 Euros (um Raspberry Pi).

Os principais algoritmos estudados baseiam-se na aquisição da corrente. Os métodos de deteção das falhas trabalham em frequência ou tempo-frequência, tendo alcançado um bom desempenho, embora já conhecido pelos relatos da comunidade científica através de teses de doutoramento, de mestrado e de artigos científicos.

Finalmente o sistema *Web* foi desenvolvido recorrendo às tecnologias mais emergentes na área e portanto mais modernas, tentando que o mesmo possa ser adaptável a diferentes tipos de equipamentos e não só ao MIT. A ideia inicial era este *software* ler a configuração do equipamento através do sistema de aquisição incorporado na máquina (Raspberry PI) e apresentar todas as opções em função dos sinais programados para serem adquiridos. Este objetivo foi conseguido, sendo que os componentes disponibilizados no *interface Web* são obtidos externamente.

No geral o aluno considera que cumpriu com os objetivos propostos, sendo de salientar que eram objetivos complexos e de grau de dificuldade elevada.

Em termos de trabalhos futuros propõe-se eventualmente a adição de sensores de vibração nos motores para permitir a utilização desta informação no âmbito dos algoritmos desenvolvidos ou que futuramente o sejam.

Bibliografia

[1] ABB Ability. *ABB Ability Smart Sensor*. URL: https://new.abb.com/motors-generators/pt/servicos/servicos-avancados/smart-sensor. [Acesso: 2021].

[2] Megger. *Baker EXP4000*. URL: https://megger.com/dynamic-motor-analyzer-baker-exp4000. [Acesso: 2021].

[3] Fluke. *Fluke 1770*. URL: https://www.fluke.com/pt-pt/produto/testes-eletricos/qualidade-de-energia/1773-1775-1777. [Acesso: 2021].

[4] Siemens. URL: https://www.automation.siemens.com/bilddb/search.aspx?lang=en. [Acesso: 2020].

[5] Gratispng. URL: https://www.gratispng.com. [Acesso: 2020].

[6] Stephen D. Umans. *Máquinas Elétricas de Fitzgerald e Kingsley*. 7ª Edição. AMGH Editora Ltda., 2014.

[7] Muhammad Irfan Nordin Saad e Rosdiazli Ibrahim. *Condition Monitoring and Faults Diagnosis of Induction Motors. Eletrical Signatre Analysis*. Taylor & Francis Group, LLC, 2019. Cap. 2.

[8] Yogesh R. Patni e Prof. M.M.Ansari. "Review paper on Stator and Rotor Fault Diagnosis Of 3-Phase Induction Motors". Em: 05 (2018).

[9] Madhuchhanda Mitra Subrata Karmakar Surajit Chattopadhyay e Samarjit Sengupta. *Induction Motor Fault Diagnosis. Approach trought Current Signature Analisys*. Springer, 2016.

[10] Sourav Pradhan Partha Sarathee Bhowmik e mangal Prakash. "Fault Diagnostic and Monitoring Methods of Induction Motor:A Review". Em: 1 (2013).

[11] Vahid Ghorbanian Jawad Faiz e Gojko Joksimović. *Faults Diagnosis of Induction Motors*. The Institution of Engineering e Technology, 2017.

[12] Hosameldin Ahmed e Asoke K. Nandi. *Condition Monitoring with Vibration Signals*. John Wiley & Sons Ltd, 2020.

[13] Paresh Girdhar e Associates. *Practical Machinery Vibration Analysis and Predictive Maintenance*. Newnes, 2004.

[14] Nabamita Banerjee Roy e Kesab Bhattacharya. *Application of Signal Processing Tools and Artificial Neural Network in Diagnosis of Power System Faults*. CRC Press, 2021.

[15] *Induction Motors – Modelling and Control*. InTech, 2012.

[16] L.CHRIFI ALAOUI Samir BOUSLIMANI Said DRID e M.Ouriagli. "An Extended Park's Vector Approuach to Detect Broken Bars Faults In Induction Motor". Em: (2015).

[17] MPT. *Modern Pumping Today*. URL: https://modernpumpingtoday.com. [Acesso: 2021].

[18] DMC. *Medição de vibrações*. URL: https://www.dmc.pt/medicao-de-vibracoes. [Acesso: 2021].

[19] W3schools. URL: https://www.w3schools.com/js/. [Acesso: 2021].

[20] Stack Exchange. *2020 Developer Survey*. URL: https://insights.stackoverflow.com/survey/2020. [Acesso: 2021].

[21] ReactJS. *A JavaScript library for building user interfaces*. URL: https://reactjs.org/docs. [Acesso: 2021].

[22] Material Ui. *React UI framework*. URL: https://material-ui.com. [Acesso: 2021].

[23] NodeJS. *Documentation*. URL: https://nodejs.org/en/docs. [Acesso: 2021].

[24] GraphQL. *A query language for your API*. URL: https://graphql.org/learn. [Acesso: 2021].

[25] IETF Datatracker. *Hypertext Transfer Protocol*. URL: https://datatracker.ietf.org/doc/html/rfc2616. [Acesso: 2021].

[26] RFC. *Hypertext TRansfer Protocol —HTTP/1.1*. URL: https://www.rfc-editor.org/rfc/rfc2616.pdf. [Acesso: 2021].

[27] MQTT. *The Standard for IoT Messaging*. URL: https://mqtt.org. [Acesso: 2021].

[28] OASIS Standard. *MQTT Version 5.0*. URL: http://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html. [Acesso: 2021].

[29] Raspberry Pi. *Raspberry Pi Foundation*. URL: https://www.raspberrypi.org. [Acesso: 2021].

[30] APEXCHARTS. *ApexCharts*. URL: https://apexcharts.com. [Acesso: 2021].

Apêndice A

Aplicação *web*

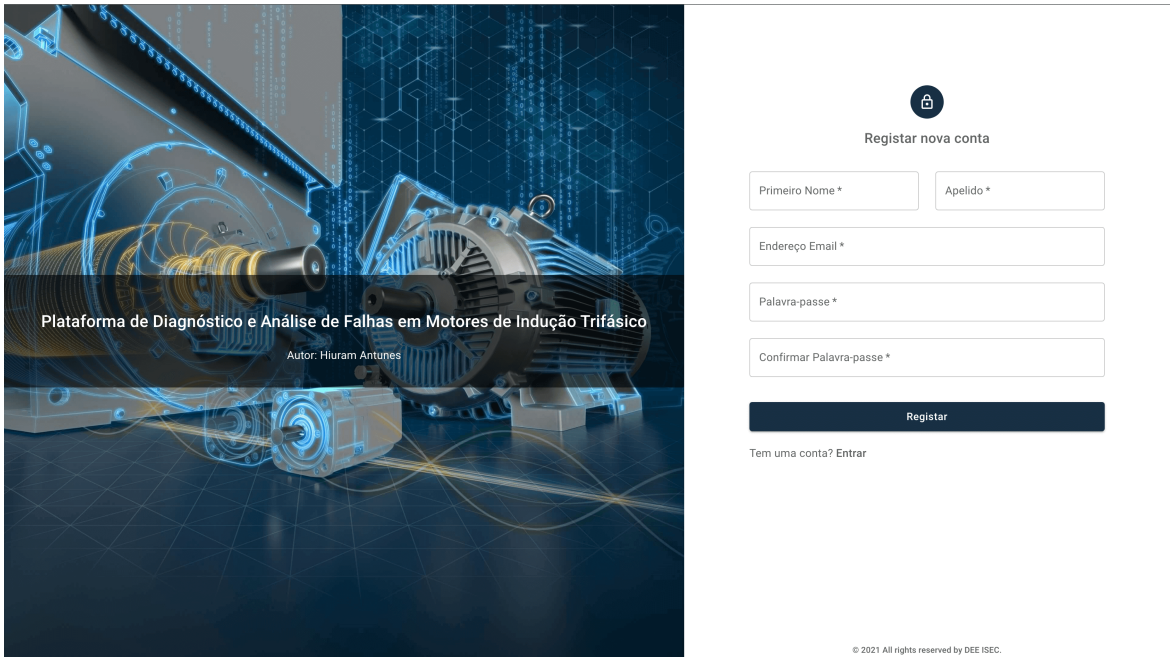
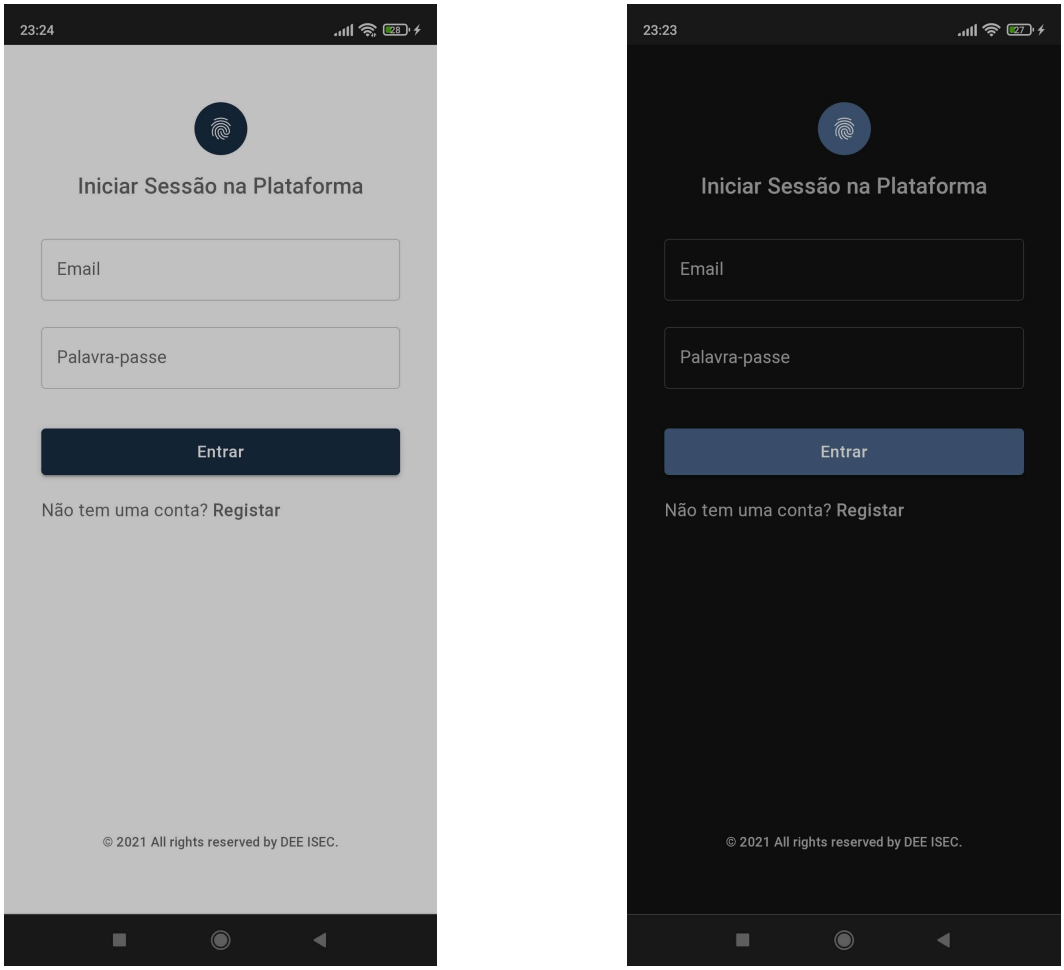


Figura A.1: Página de Registo.



(a) Tema claro *(Light)* (b) Tema escuro *(Dark)*

Figura A.2: Disposição de aplicação em *smartphone*

Apêndice B

Marcadores Visuais

Figura B.1: Resultados Obtidos (Opencv e JavaFx)

Figura B.2: Classificador em Python e OpenCV.