Universidade de Aveiro Departamento de Engenharia Mecânica
2020

Pedro Manuel
Monteiro Limas

Desenvolvimento de um sistema SCADA para ajuda à manutenção na AviSabor

Universidade de Aveiro Departamento de Engenharia Mecânica
2020

Pedro Manuel
Monteiro Limas

Desenvolvimento de um sistema SCADA para ajuda à manutenção na AviSabor

Relatório de Estágio apresentado à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Mecânica, realizado sob orientação científica de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro.

Apoio financeiro de: Projeto Investigação do Centro de Tecnologia Mecânica e Automação (TEMA) com referência UIDB/00481/2020 e UIDP/00481/2020; e Projeto Centro de Tecnologia Mecânica e Automação (TEMA) Infraestrutura de Investigação com a referência CENTRO-01-0145-FEDER-022083.

O júri / The jury

Presidente / President

Prof. Doutor António Gil D'Orey de Andrade Campos
Professor Auxiliar c/ Agregação da Universidade de Aveiro

Vogais / Committee

Prof. Doutor José Paulo Santos
Professor Auxiliar da Universidade de Aveiro (Orientador)

Prof. Doutor Pedro Nicolau Faria da Fonseca
Professor Auxiliar da Universidade do Aveiro (Arguente)

Agradecimentos / Acknowledgements

Em primeiro lugar, quero demonstrar os meus agradecimentos ao Professor José Paulo Santos pela orientação prestada ao longo do desenvolvimento desta dissertação.

A concretização deste projeto deve-se a um problema proposto pela AviSabor que, independentemente dos acontecimentos do presente ano, sempre se mostrou interessada e disponível para me ajudar naquilo que fosse necessário. Um agradecimento especial ao Engº Hugo Loureiro, ao Engº Carlos Sousa e ao Engº Gabriel Esteves, que se mostraram sempre prestáveis com qualquer problema que tivesse.

Um obrigado especial ao Professor Abílio Borges pela disponibilidade, pelo material, por toda a ajuda prestada e porque, dadas as circunstâncias, não hesitou em oferecer-me um espaço na Atena onde eu pudesse fazer os testes necessários.

Sei que quando me lembrar da universidade há, inevitavelmente, algumas pessoas que marcaram e ajudaram a traçar o meu percurso académico. Aos comparsas Francisca Matos, Patrícia Assis, Fabrício Souza e Rui Martins, que embora tenho entrado a meio, entrou com força. A estes quero deixar um grande obrigado, por tornarem alguns dias pesados em autênticas nuvens e votos de um brilhante futuro.

Deixo um fortíssimo e especial abraço de agradecimento ao Rúben Costa que me deu "carry" nas mais variadas ocasiões e por ser a prova de que o trabalho árduo compensa. Por todas as horas de conversa, trabalho e amizade e por isso, "Estamos aí, brother".

Ao Miguel Arroz só tenho que agradecer por todas as horas em que foi um puro "craque" e pela descomplicação de todos os problemas técnicos que foram surgindo.

Um obrigado aos meus Vieiras, que são facilmente um dos melhores grupos aí na via. Não é que tenham tido especial contributo para a tese, mas um gajo ama-vos na mesma.

Aos "goodfellas" André João, Miguel Curado, Rafael Teles por serem quem são. Dificilmente se arranja desta massa noutro lado.

Sem querer particularizar, mas com particular destaque aos Andrés, Joões, Migueis, Pedros, e Ruis desta vida. "Let's get it".

Finalizando com um "brinde" à família.

Pelo que viste e pelo que eu não sei se viste, pelo que fizeste e talvez pelo que ainda fazes, qualquer texto é insuficiente. Um grande obrigado Paizão, "O puro dos puros".

Se uma é trabalhoso, duas é obra. Imagine-se três. À Manuela, à Rita e à minha Mãe. Um hino a estas mulheres que acompanharam de perto todo o percurso, que viram a minha evolução nos últimos cinco anos (e não só), que travaram algumas lutas comigo e fizeram de mim o homem que sou. Estou-vos grato.

Palavras-chave Sistema SCADA, Modbus-RTU-RS485, Servidor, Variador de Frequência, Microcontrolador, Página *Web*, Manutenção Preventiva

Resumo Em empresas que operem na indústria agro-pecuária, a água é um dos mais recursos mais solicitados pelas várias etapas do processo produtivo. A AviSabor, S.A é uma empresa responsável pelo abate, desmanche e expedição de frangos para todo o país. Nesta fábrica ocorrem frequentemente anomalias nos equipamentos responsáveis pelo bombeamento de água para o interior da unidade fabril. Como não existe supervisão, sempre que algum destes equipamentos atinge a falha, a equipa de manutenção é obrigada a deslocar-se até ao edifício onde estes se encontram para averiguar o problema e encontrar uma solução. A conjugação destes fatores constituem custos para a empresa que podem ser evitados com a implementação de um sistema de supervisão que possa ser acedido remotamente. Assim, por forma a permitir uma manutenção mais eficaz e prevenir avarias destes equipamentos, o estágio proposto em colaboração com a AviSabor, S.A, tem como objetivo o desenvolvimento de um sistema SCADA que viabilize a monitorização do sistema de bombeamento de água. Já que na empresa o funcionamento dos variadores de frequência que controlam o funcionamento das bombas é regulado por autómatos que os ligam e desligam conforme a necessidade de água no processo produtivo, implementou-se em laboratório um sistema que viabilize a monitorização e controlo de variadores de frequência e dos autómatos associados, a sua integração com bases de dados MySQL e a apresentação dos dados recolhidos (valores de frequência, intensidade de corrente e tensão) numa página *web*, recorrendo ao ambiente de programação Node-RED. A implementação de um sistema deste género contribui para uma manutenção preventiva eficaz visto que quando alguma anomalia é detetada, é enviado um alerta via *e-mail* para que a equipa de manutenção possa atuar rapidamente e arranjar ou substituir o equipamento danificado.

Keywords SCADA System, Modbus-RTU-RS485, Server, Frequency Inverter, Microncontroller, Webpage, Preventive Maintenance

Abstract In companies operating in the agribusiness industry, water is one of the most requested resources by the various stages of the production process. AviSabor, S.A is a company responsible for poultry slaughter, cutting and dispatch throughout the country. In this factory, there are frequent anomalies in the equipment responsible for pumping water into the plant. As there is no supervision, whenever there's a faulty equipment, the maintenance team is obliged to go to the building where they are located to examine the problem and find a solution. The combination of these factors constitutes costs for the company that can be avoided by implementing a remote supervisory system. Thus, to allow more effective maintenance and prevent damage to these equipment, the proposed internship in collaboration with AviSabor, S.A, aims to develop a SCADA system that enables the monitoring of the water pumping system. Since the operation of the frequency inverters which control the pumps is regulated by PLCs that turn them on and off according to the need for water in the production process, a system has been implemented in laboratory to enable the monitoring and control of frequency inverters and associated PLCs, their integration with MySQL databases and the presentation of the collected data (frequency, current intensity and voltage) on a dedicated webpage, using Node-RED programming environment. The implementation of such a system contributes to an effective preventive maintenance since when an anomaly is detected, an alarm is sent via e-mail so that the maintenance team can act on it quickly and fix or replace damaged equipment.

Índice

1 Introdução **1**
1.1 Enquadramento 1
1.2 Objetivos 1
1.3 Organização do Documento 2

2 Caraterização da Empresa **3**
2.1 Descrição e História 3
2.2 Processo Produtivo 4
2.2.1 Receção das Aves Vivas 5
2.2.2 Abate 5
2.2.3 Sangria 5
2.2.4 Escaldão e Depenadoras 5
2.2.5 Evisceração 5
2.2.6 Túnel de Refrigeração 6
2.2.7 Calibragem 6
2.2.8 Desmancha 6
2.3 Caraterização do Problema 6

3 Revisão bibliográfica de Sistemas SCADA e Tecnologias IoT **11**
3.1 Outras Dissertações 11
3.1.1 Monitorização e controlo remoto de sistemas de rega agrícola . . . 11
3.1.2 Proposta de uma solução SCADA para a Manutenção Preditiva . . 13
3.2 Sistemas SCADA 14
3.2.1 Soluções Comerciais 16
3.3 PLC - *Programmable Logic Controller* 17
3.4 Variadores de Frequência 18
3.4.1 Conceitos Técnicos Utilizados 19
3.5 Microcontroloadores 19
3.6 IoT - *Internet Of Things* 19
3.6.1 Arquitetura IoT 20
3.6.2 Plataformas IoT 20
3.7 *Softwares* e Linguagens de Desenvolvimento *web* 24

4 Solução Proposta **25**
4.1 Descrição da Solução 25
4.2 Abordagem Inicial para o Sistema de Aquisição de Dados 27

4.3 Sistema de Aquisição de Dados (Frequência, Intensidade de Corrente, Tensão) . . . 28
4.3.1 Comunicação - Modbus-RTU-RS-485 [1] . . . 29
4.3.2 Comunicação com o Servidor . . . 34
4.4 Servidor (Raspberry Pi) . . . 34
4.4.1 *Broker* MQTT . . . 34
4.4.2 Node-RED . . . 35
4.4.3 Repositório de Dados - MySQL . . . 36
4.5 Interface do Utilizador para Controlo e Supervisão . . . 38
4.5.1 Comunicação entre a Interface e Servidor Apache . . . 38
4.5.2 Página *web* . . . 39
4.5.3 Registo de Eventos e Alarmes . . . 39

5 Implementação **41**
5.1 Parametrização do Variador de Frequência . . . 41
5.1.1 Configuração dos Terminais . . . 42
5.2 Sistema de Aquisição de Dados (Frequência, Intensidade de Corrente, Tensão) . . . 43
5.2.1 Módulo IoT - ESP32 . . . 44
5.2.2 Comunicação entre o Sistema de Aquisição de Dados (FIT) e o Variador de Frequência . . . 45
5.2.3 Comunicação entre o Sistema de Aquisição de Dados e o Servidor (Raspberry Pi) . . . 50
5.2.4 Desenvolvimento da placa PCB . . . 51
5.3 Servidor (Raspberry Pi) . . . 52
5.3.1 Módulo Raspberry Pi 3B . . . 52
5.3.2 Comunicação entre Servidor e Sistema de Aquisição de Dados . . . 53
5.3.3 Repositório de Dados (Adquiridos pelo Sistema de Aquisição e Limites Selecionados) . . . 53
5.4 Interface Gráfica de Supervisão e Controlo do Variador de Frequência (Página *web* SCADA) . . . 55
5.4.1 Comunicação entre Servidor e Interface . . . 55
5.4.2 Página *web* SCADA . . . 55
5.5 Modo Automático de Regulação de Frequência . . . 58
5.5.1 Funcionamento . . . 60
5.5.2 Comunicação do Módulo de Aquisição de Entradas Analógicas (Módulo AI/DO) . . . 60
5.5.3 Sistema de Aquisição de Dados (FIT) . . . 63
5.5.4 Ativação do Modo Automático de Regulação da Frequência . . . 63
5.5.5 Domínio de IP's . . . 64

6 Análise de desempenho **65**
6.1 Desempenho do Sistema SCADA . . . 65
6.1.1 Sistema de Aquisição de Dados (Frequência, Intensidade de Corrente, Tensão) . . . 65
6.1.2 Servidor Local (Raspberry Pi) . . . 65
6.1.3 Página *web* . . . 66

6.1.4 Modo Automático de Regulação de Frequência 69
6.1.5 Modo Remoto vs Modo Automático de Regulação de Frequência . 69

7 Conclusões e Trabalhos Futuros **71**
7.1 Conclusões 71
7.2 Trabalhos Futuros 74

A Desenvolvimento **81**
A.1 Configuração do Variador 81
A.2 Esquema Elétricos 83
A.2.1 Diagrama de Componentes 83
A.2.2 Esquema Elétrico da PCB 84
A.3 Alterações no Node-RED 85
A.4 Alterações das definições do *mail* 87
A.5 Diagramas de Sequência 89
A.6 Registo de Alarmes 91
A.7 Fotografias do Sistema na *breadboard*. 94
A.8 Opções de Microcontroladores Implementáveis 95
A.9 Código de programação - ESP32 96
A.9.1 Configuração da Rede Wi-Fi 96
A.9.2 Configuração do *broker* MQTT 96

Lista de Figuras

2.1 Fluxograma do processo produtivo 4
2.2 Planta síntese da AviSabor 7
2.3 Sala das bombas e os resevatórios de água 7
2.4 (a) Variador de Frequência - Delta C2000, (b) PLC Siemens S7-1200 . . . 8

3.1 Arquitetura proposta na dissertação 12
3.2 Esboço do funcionamento da solução implementada 13
3.3 Representação da Solução proposta 13
3.4 Arquitetura geral da solução proposta 14
3.5 Imagem ilustrativa das camadas que compõem um sistema SCADA segundo o modelo ISA95 15
3.6 Imagem ilustrativa do *software* Siemens Simatic WinCC 16
3.7 Imagem ilustrativa do *Software* Movicon 17
3.8 Exemplo de um PLC da Siemens 18
3.9 Imagem representativa de uma gama de variadores de frequência Delta . . 18
3.10 Módulo ESP32 19
3.11 Arquitetura de uma implementação IoT 20
3.12 Arquitetura do serviço Microsoft Azure IoT 21
3.13 Arquitetura do serviço Siemens Mindsphere 22
3.14 Arquitetura do serviço Bosch Suite IoT 23

4.1 Representação esquemática da solução proposta 26
4.2 *Hardware* da empresa: (a) PLC Siemens S7-1200, (b) Switch Siemens Scalance XB005, (c) Siemens KTP400, (d) Variador de Frequência - Delta C2000 27
4.3 Arquitetura conceptual do Sistema de Aquisição de Dados 29
4.4 Comunicação entre o microcontrolador e o variador 29
4.5 Relação entre o comprimento do cabo e a velocidade de comunicação RS-485 30
4.6 Tensões das linhas A e B no envio de um *byte* 31
4.7 Estrutura de uma resposta ao envio de uma mensagem série 31
4.8 Exemplo de leitura de um registo 31
4.9 Exemplo de escrita de valor de frequência num registo 32
4.10 Resistências de terminação, *pull-up* e *pull-down* 32
4.11 Arquitetura sonceptual do Servidor 34
4.12 Diagrama de comunicação de um *broker* **MQTT** 35
4.13 Programação visual usando o Node-RED 36
4.14 Exemplo da organização e estrutura de tabelas 37

4.15 Exemplo de *design* para uma página web para aplicações SCADA 39
4.16 Exemplo de uma tabela de registo de alarmes em Excel 39

5.1 (a) Terminais do variador, (b) Esquema da alimentação dos terminais do variador 42
5.2 Exemplo de leitura de um registo 43
5.3 Ligação Modbus-RTU-RS-485 entre o conector do Sistema de Aquisição de Dados e o conector PU do variador de frequência 43
5.4 Ligações do Sistema de Aquisição de Dados na *breadboard* 44
5.5 Disposição dos pinos do ESP32 45
5.6 (a) Conversor de níveis TTL para RS-485, (b) Disposição dos pinos do conector do Olimex MOD-RS-485 (UEXT) 46
5.7 Fluxograma de aquisição de dados e escrita nos registos do variador 48
5.8 Fluxograma correspondente à leitura de registos do variador 49
5.9 Modelo 3D do protótipo desenvolvido em placa **PCB** 51
5.10 Várias vistas da PCB 51
5.11 Raspberry Pi Model 3B+ 52
5.12 Diagrama de comunicação entre o Sistema de Aquisição de Dados, o Servidor e o *broker* MQTT 53
5.13 Código Node-RED para inserir valores na base de dados MySQL 54
5.14 Caixa de controlo dos modos de funcionamento do variador de frequência 55
5.15 Estados de funcionamento do varidor 56
5.16 Disposição dos gráficos de Frequência e Tensão (em função do tempo) no separador "Vista Comprimida" na página *web* 56
5.17 Exemplo de alarme acompanho por um ***popup*** 57
5.18 *Flows* do Node-RED e o exemplo de um alarme recebido via *e-mail* 57
5.19 Arquitetura do Modo Automático de Regulação de Frequência segundo a norma ISA95 58
5.20 *Hardware* do Modo Automático: (a) ADAM 6017, (b) PLC Siemens ET200s, (c) Variador de Frequência Mitsubishi D720S, (d) Sistema de Aquisição de Dados, (e) Motor Trifásico, (f) Potenciómetro, (g) *Router* TP-LINK . . 59
5.21 Gráfico da aplicação *web* do Módulo de Aquisição I/O 61
5.22 Tensão das entradas analógicas do Módulo de Aquisição de Entradas Analógicas 61
5.23 *Flow* em Node-RED para aquisição do valor analógico 62
5.24 *Flow* de publicação do valor da frequência de comando num tópico MQTT 62
5.25 Fluxograma do programa *ladder* responsável pelo Ciclo Automático 63
5.26 Caixa do Modo Automático na página *web* 64

6.1 Nós de comunicação com a base de dados 66
6.2 Vários separadores presentes na interface 67
6.3 *Gauges* associados a cada variável 67
6.4 Exemplo da aba de "Controlo" da página *web* 68
6.5 Conteúdo do separador de alarmes 69

7.1 Esquema representativo do funcionamento de um *chip* LoRa 74

A.1 ***Pinout*** do conector **PU** (igual ao RJ45) do variador de frequência. 81

A.2 Conector de três pinos da PCB do Sistema de Aquisição de Dados. 82
A.3 Variador de Frequência utilizado (Mitsubishi FR-D720S). 82
A.4 Ligação dos fios condutores entre o variador e o adaptador. 82
A.5 Diagrama de componentes. 83
A.6 Esquema de ligações da placa de circuito impresso. 84
A.7 Alteração das credenciais da base de dados. 85
A.8 Alteração do *mail*. 86
A.9 Alteração dos dados do *broker* MQTT. 86
A.10 1º passo: Gerir a sua Conta Google. 87
A.11 2º passo: Opção de Segurança. 87
A.12 3º passo: Ativar o acesso em Acesso a apps menos seguras. 88
A.13 4º passo: Ativar a opção Permitir aplicações menos seguras. 88
A.14 Diagrama de sequência de controlo e monitorização. 90
A.15 1º passo - abrir o ficheiro .txt em Excel. 91
A.16 2º passo - Várias opções dos carateres delimitadores de colunas. 91
A.17 3º passo - Escolher a "vírgula"como carater delimitador. 92
A.18 4º passo - Escolher o formato de cada coluna. 92
A.19 5º passo - Exemplo de uma tabela em Excel. 92
A.20 Guardar o ficheiro num formato .xlsx. 93
A.21 Sistema de Aquisição de Dados desenvolvido na placa de prototipagem (*breadboard*) e na **PCB**. 94
A.22 Opções de Microcontroladores Implementáveis: (a) ESP32, (b) Arduino + *Ethernet Shield*. 95

Lista de Tabelas

2.1 Especificações Técnicas das bombas 9

4.1 Potencial da linha com e sem atividade 30
4.2 Funções do *Quality of Service* 35
4.3 Credenciais MySQL 36
4.4 Exemplos de queries MySQL 37
4.5 Camadas do Modelo OSI do protocolo TCP/IP 38

5.1 Especificações Técnicas do variador Mitsubishi FR-D720S 41
5.2 Configuração dos terminais do variador 42
5.3 Especificações do ESP32 e Arduino Uno + Ethernet Shield 45
5.4 Designação e descrição dos pinos do conector 46
5.5 Especificações do modelo do Raspberry Pi 52

6.1 Tabelas criadas na base de dados 66
6.2 Tabelas de comparação dos modos 70

Lista de Acrónimos

AC Corrente Alternada (em Inglês, Alternate Current)

ADC Analog-to-Digital Converter

AI Entrada Analógica (em Inglês, Analog Input)

AO Saída Analógica (em Inglês, Analog Output)

API Application Programming Interface

AP Access Point

ASCII Código Padrão Americano para Intercâmbio de Informações (em inglês, American Standard Code for Information Interchange)

CPS Cyber-Physical System

CRC Cyclic Redundancy Check

CSS Cascading Style Sheets

DC Corrente Contínua (em Inglês, Direct Current)

DI Entrada Digital (em Inglês, Digital Input)

DO Saída Digital (em Inglês, Digital Output)

EIA Electronics Industry Alliance

ERP Planeamento de Recursos Empresariais (em Inglês, Enterprise Resource Planning)

FTP File Transfer Protocol

GND Malha de terra (em Inglês, Ground)

GSM Groupe Special Mobile

HMI Human Machine Interface

HTML Hypertext Markup Language

HTTPS Hyper Text Transfer Protocol Secure

HTTP Protocolo de Transferência de Hipertexto (em inglês, HyperText Transfer Protocol)

IDE Integrated Development Environment

IEEE Instituto de Engenheiros Eletricistas e Eletrônicos (em inglês, Institute of Electrical and Electronic Engineers)

IoS Internet das Serviços (em Inglês, Internet of Services)

IoT Internet das Coisas (em Inglês, Internet of Things)

IP Internet Protocol

IT Tecnologia da Informação Technology (em Inglês, Information Technology)

JS Javascript

LAN Local Area Network

MCU Microcontroller Unit

MQTT Message Queuing Telemetry Transport

Mbps Unidade de transmissão de dados Megabits por segundo

OBDC Open Database Connectivity

OPC Open Process Control

OT Tecnologia de Operação Technology (em Inglês, Operational Technology)

OTA Over The Air

PCB Printed Circuit Board

PHP Personal Home Page

PLC Programmable Logic Controller

QoS Quality of Service

RAM Random Access Memory

RTU Remote Terminal Unit

SA Sociedade Anónima

SCADA Supervisory Control and Data Acquisition

SCTP Stream Control Transmission Protocol

SMTP Simple Mail Transfer Protocol

SMT Surface-Mount Technology

SQL Structured Query Language

SSID Service Set Identifier

STA Station

TCP Transmission Control Protocol

THT Through-Hole Technology

TIA Totally Integrated Automation

TTL Transistor–Transistor Logic

UART Universal Asynchronous Receiver-Transmitter

UDP User Datagram Protocol

USB Universal Serial Bus

WAN Wide Area Network

Capítulo 1

Introdução

1.1 Enquadramento

A Indústria 4.0 promove a ligação entre objetos físicos como sensores, dispositivos e recursos empresariais à *internet*. Baseia-se em sistemas ciber-físicos *(Cyber-Physical Systems - CPS)*, que não são mais do que maquinaria e equipamentos integrados e interligados num sistema. Geralmente, os CPS são percebidos como a capacidade de interação entre o mundo virtual e os elementos físicos, sendo fundamentais para dotar um sistema de inteligência e capacidade de decisão e dessa forma permitir, por exemplo, uma manutenção eficaz [2]. A manutenção é uma atividade crítica que tem lugar no chão de fábrica. Quando esta não é devidamente realizada, quer por falta de qualificação ou falhas técnicas, tem impacto tanto no desempenho do processo produtivo como no produto acabado. A imprevisibilidade criada pela falta de manutenção regular dos equipamentos pode culminar paragens de linha não programadas assim como em consequências potencialmente perigosas como acidentes e falta de segurança dos trabalhadores [3].

A monitorização em tempo real tem assumido um papel fundamental na indústria, nomeadamente para supervisão e controlo interno da qualidade dos processos de fabrico para, desta forma, garantir a qualidade do produto final. Como na AviSabor não existe supervisão dos equipamentos que fazem parte do sistema de bombeamento de água para o interior da unidade fabril, este projeto visa o desenvolvimento de um sistema baseado num microcontrolador capaz de adquirir dados relevantes - frequência, intensidade de corrente - para dessa forma avaliar em tempo real o estado destes equipamentos.

Apesar de já haver alguns sistemas concebidos exclusivamente para efeitos SCADA, estes revelam-se muitas vezes limitados e com problemas de integração, devido à existência de diversos protocolos de comunicação entre os sensores, variadores de frequência, autómatos, bases de dados e outros.

1.2 Objetivos

Para que a AviSabor, S.A consiga cumprir todos os objetivos a que se propõe anualmente para com os seus clientes, é de extrema importância que o departamento de Engenharia e de Qualidade faça um acompanhamento e uma análise constante das necessidades das linhas de produção. Um acompanhamento permanente permite reagir rapidamente na prevenção e correcção dos mais diversos tipos de falhas, desde má ges-

tão de *stock*, falhas na maquinaria (devido à falta de manutenção ou antiguidade dos equipamentos utilizados), utilização inapropriada dos recursos da empresa, entre outros. Desta forma, evita-se a subida dos custos de manutenção e a qualidade do produto é assegurada.

Adotando uma perspetiva técnica, pode-se constatar que com equipamentos danificados ou no fim do ciclo de vida, pode, em casos extremos, dar-se a paragem das linhas de produção para que a equipa de manutenção possa intervir no problema, o que culmina em custos extraordinários e desnecessários para a empresa. Este projeto insere-se no âmbito da dissertação do presente autor em ambiente industrial, para apoio à monitorização do processo de bombeamento de água nas instalações da AviSabor. O principal objetivo é o desenvolvimento de um sistema SCADA de equipamentos associados ao bombeamento de água para as linhas de produção como é o caso dos variadores de frequência (que estão ligados a bombas e motores para regular a sua velocidade, explicado com maior detalhe na Secção 3.4) e bombas que possa ser acedido remotamente. A consulta e controlo em tempo real de parâmetros como : tensão debitada, intensidade de corrente, frequência de comando, permite uma supervisão por parte dos técnicos de operação no seu posto de trabalho através de um monitor informativo ou um dispositivo semelhante com a mesma finalidade. Dados os pressupostos supramencionados, os objetivos para este projeto são então:

- Desenvolvimento de um protótipo que faça a recolha de alguns dos parâmetros supramencionados;
- Desenvolvimento de uma página *web* (*dashboard*) onde seja possível monitorizar os dados recolhidos;
- Sistema de alarmes por *e-mail* e *popup* automático para aviso dos colaboradores.

1.3 Organização do Documento

Este documento encontra-se dividido em seis capítulos:

- No Capítulo 1 é introduzido o tema e a motivação do projeto, assim como os seus objetivos;
- No Capítulo 2 é feita uma breve descrição do processo produtivo da empresa e a caraterização do problema;
- No Capítulo 3 é feita uma revisão bibliográfica de alguns sistemas SCADA e de tecnologias IoT;
- No Capítulo 4 é apresentada a solução proposta;
- No Capítulo 5 é apresentada uma proposta de implementação da solução;
- No Capítulo 6 é realizada uma análise de desempenho do sistema desenvolvido;
- No Capítulo 7 são apresentadas as conclusões que se retiraram da conclusão deste trabalho, assim como propostas de possíveis trabalhos futuros.

Capítulo 2

Caraterização da Empresa

2.1 Descrição e História

Fundada a 19 de maio de 1980, com a denominação de Hilário & Gomes, Lda, tinha como objetivo a produção, abate e comercialização avícola e similares. A 29 de setembro de 1987, o pacto social da empresa foi parcialmente alterado, quer na denominação da firma, quantidade e valor das quotas, quer no capital social e gerência da empresa. A sociedade adotou a denominação de Hilário H. M. Santos & Filhos, Produção e abate de aves, Lda. A 2 de Fevereiro de 2001, a sociedade então por quotas foi transformada em sociedade anónima, passando a usar a atual denominação Hilário Santos & Filhos, S. A. com o capital de um milhão e setenta e cinco mil euros. Em novembro de 2006, 51% do capital social foi adquirido pela Lusiaves S.A., sendo os restantes 49% adquiridos em novembro de 2007, passando a fazer parte do Grupo Lusiaves. Em abril de 2014 alterou a sua designação para AviSabor - Indústria Agroalimentar, S.A. Em janeiro de 2015 finalizou as obras de aumento do centro de abate, o que permitiu aumentar cerca de três vezes o número de frangos abatidos para 12.000 frangos/hora. Foi ainda instalada uma linha de desmancha automática que permite desmanchar até 6.000 frangos/hora.

2.2 Processo Produtivo

O processo produtivo da empresa é ilustrado pelo fluxograma da Figura 2.1, bem como os recursos associados a cada etapa.

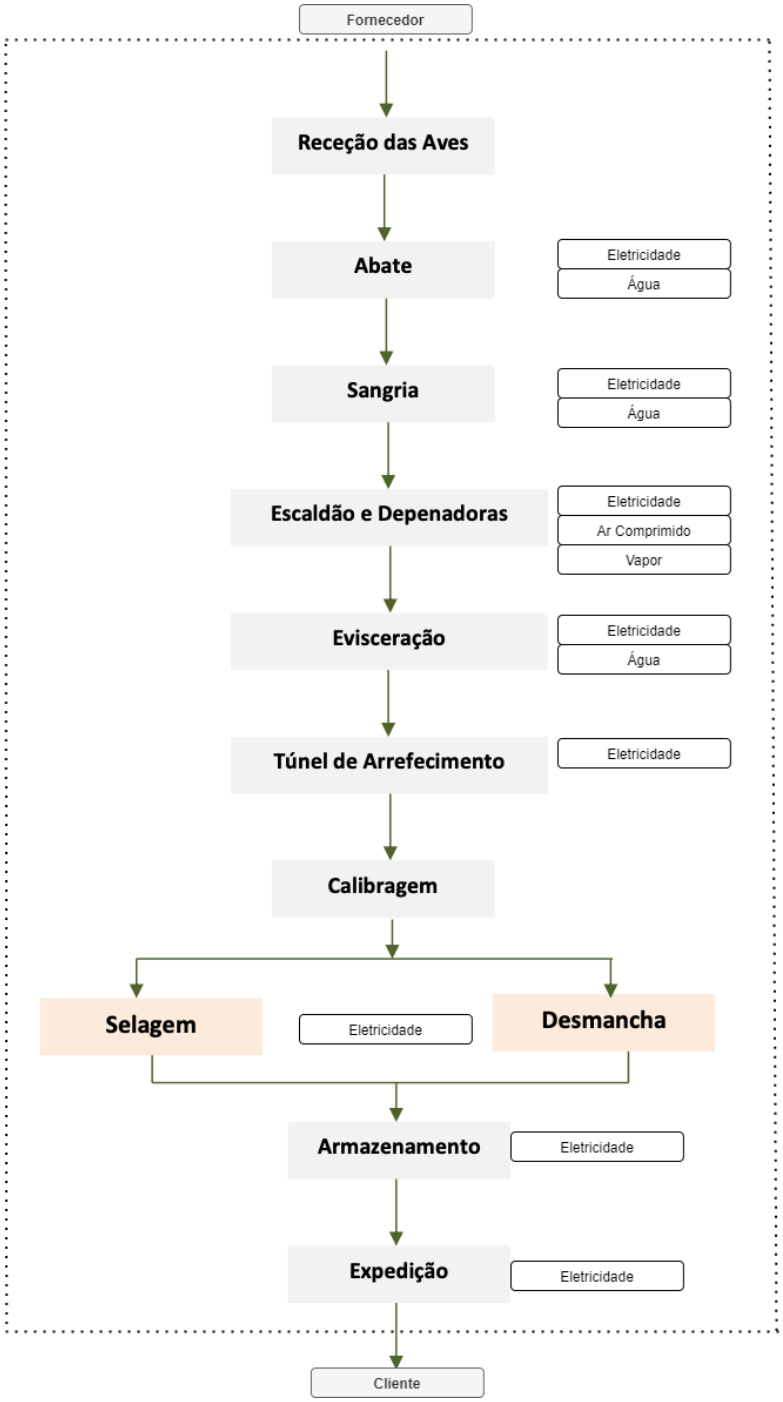


Figura 2.1: Fluxograma do processo produtivo.

2.2.1 Receção das Aves Vivas

As aves vivas são rececionadas em contentores no cais do frango vivo, que por sua vez, são descarregados e colocados na linha de transporte para posterior retirada das gavetas.

2.2.2 Abate

No setor do abate, as aves vivas são retiradas das gavetas e penduradas na linha de transporte aéreo de abate. Ao mesmo tempo os contentores e as gavetas são higienizados.

2.2.3 Sangria

De seguida, as aves são encaminhadas para a eletrocussão, que provoca a insensibilização das aves por corrente através da água, passando pelo corte das carótidas, e sangria completa do frango.

2.2.4 Escaldão e Depenadoras

Após o processo de sangria, as aves mortas são escaldadas através da imersão em tanques de água quente (49-53^{o}C), que se encontram em constante agitação. Posteriormente, são retiradas as penas por meio mecânico, nas depenadoras, as cabeças e as patas.

As patas, por sua vez, são escaldadas de modo a permitir a remoção da pele através de borrachas, e as patas selecionadas são mergulhas em água fria para provocar uma redução de temperatura mais rápida. Após o respetivo armazenamento seguem para a câmara de refrigeração (0-4^{o}C).

2.2.5 Evisceração

Antes da retirada das vísceras, é realizado um furo vertical junto à cloaca, e um corte ao longo do abdómen de forma a facilitar a evisceração. De seguida, são removidas automaticamente as vísceras para pratos que acompanham a respetiva carcaça. Após a limpeza e seleção das vísceras (fígados, corações e moelas), são mergulhadas em água fria (4-12^{o}C) para provocar uma redução térmica mais rápida, colocadas em caixas e armazenadas na câmara de refrigeração (0-4^{o}C). As tripas são encaminhas para alinha dos subprodutos. Posteriormente, os pescoços são removidos, mergulhados em água fria (4-12^{o}C), lavados, e armazenados na câmara de refrigeração. Por último, e para finalizar a fase do abate, as peles dos pescoços do frango são removidas da carcaça, e os pulmões aspirados do seu interior, seguindo para os subprodutos. As carcaças são lavadas, por dentro e por fora, antes da sua transferência aérea automática para a linha do túnel de refrigeração. Após o término da fase de abate, o frango é encaminhado para a calibragem, após o túnel de refrigeração.

2.2.6 Túnel de Refrigeração

O frango é encaminhado para o túnel de refrigeração para arrefecimento e estabilização da temperatura, permanecendo cerca de 150min a uma temperatura entre 0 e 2^{o}C.

2.2.7 Calibragem

Na calibradora o frango é classificado como Classe A e B, conforme software da *Meyn*. O frango classificado como A, é automaticamente selado, acondicionado e armazenado na câmara de armazenamento (0-4^{o}C), enquanto que, o da Classe B segue para a Desmancha.

2.2.8 Desmancha

O frango classificado como da Classe B, é transferido automaticamente da linha da calibradora para a linha da desmancha, onde as asas são automaticamente cortadas, assim como o peito. A pele é retirada e o peito é desossado. O peito é selecionado por observação visual (remoção do sangue, gordura e osso). Da limpeza do peito resulta a produção de aparas e *trimmings* que são posteriormente comercializados. O peito passa de seguida pelo Raio-X para que se possa detetar a presença de osso. Segue-se o corte automático da carcaça e da perna (com desossa da coxa).

Por fim, o produto é acondicionado em caixas, pesado e identificado sendo posteriormente armazenado na câmara de armazenamento (4-10^{o}C). De referir que todas as caixas e paletes são devidamente higienizadas, e posteriormente armazenadas de forma a serem reutilizadas.

2.3 Caraterização do Problema

Na indústria agropecuária, um dos recursos mais importantes é a água. Como tal, é fundamental fazer uma gestão eficiente da mesma, por forma a evitar desperdícios.

Como se pode ver pela planta da AviSabor representada na Figura 2.2, a sala das bombas da Figura 2.3, é o local no qual existem os equipamentos responsáveis pelo bombeamento de água com base nos quais se desenvolveu o sistema pretendido.

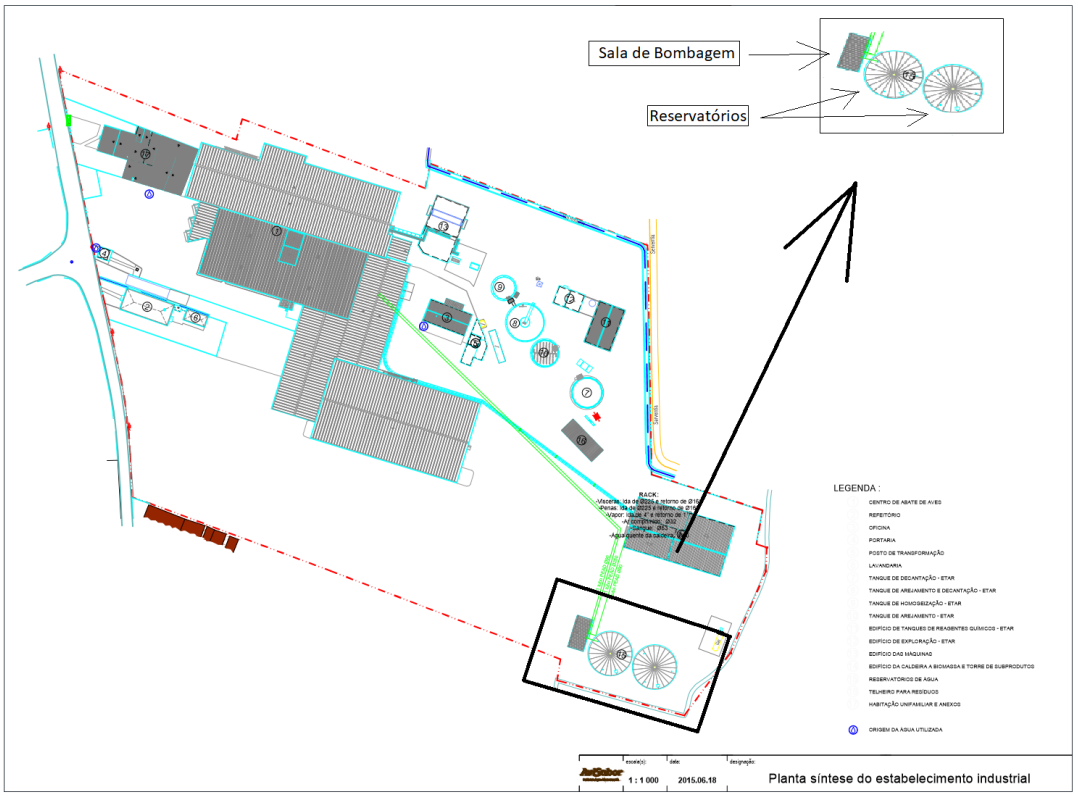


Figura 2.2: Planta síntese da AviSabor.

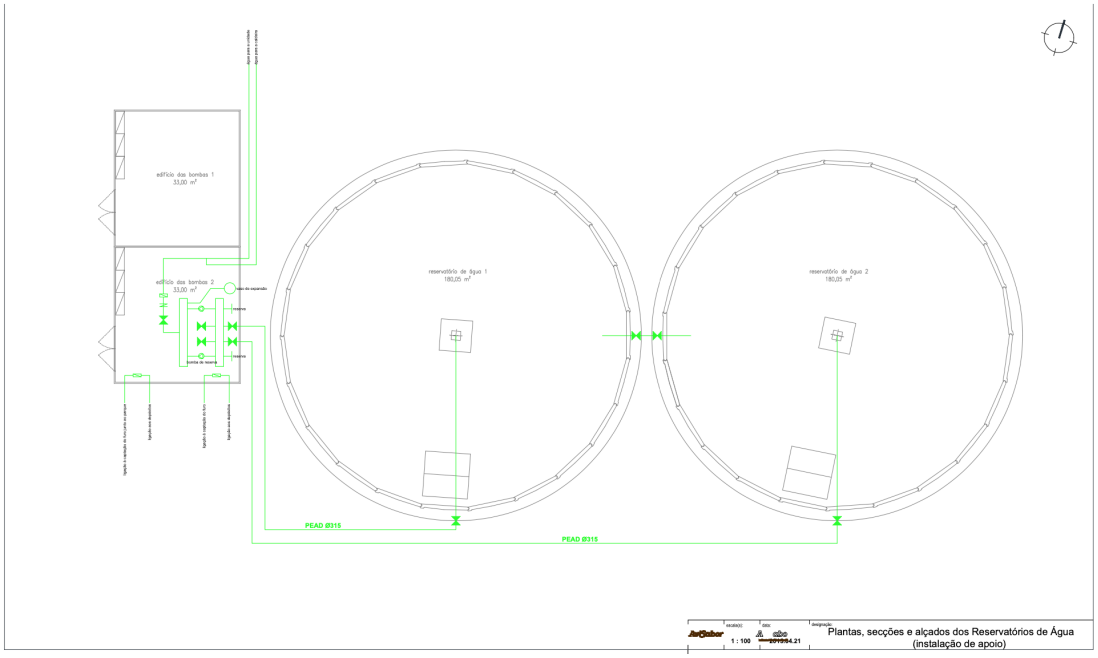


Figura 2.3: Sala das bombas (à esquerda na Figura) e os resevatórios de água (à direita na Figura).

Nesta sala existe um sistema de bombagem que funciona da seguinte forma: a água presente em dois reservatórios (com capacidade para um milhão de litros cada) é puxada por bombas ligadas a variadores de frequência da marca Delta (sirva de exemplo a Imagem (a) da Figura 2.4), ligados, por sua vez, a PLC's Siemens S7-1200 (observe-se a Imagem (b) da mesma Figura) que através da leitura dos sensores de nível ligados às entradas analógicas do autómato, ativa ou desativa as bombas.

(a) (b)

Figura 2.4: (a) Variador de Frequência - Delta C2000, (b) PLC Siemens S7-1200.

Em suma, as condições deste projeto são:

- Nº de bombas: 3;
- Nº de variadores de frequência: 3;
- Nº de autómatos: 2.

Com a consulta da Tabela 2.1 pretende-se dar a conhecer com maior detalhe as especificações técnicas das bombas com que a empresa trabalha e dessa forma entender melhor as caraterísticas do problema em análise.

Tabela 2.1: Especificações Técnicas das bombas.

	Bomba 1	**Bomba 2**	**Bomba 3**
Marca	EFAFLU	EFAFLU	LOWARA
Potência [kW]	11	15	18,5
Frequência [Hz]	50	50	60
Rotações Máximas [rpm]	1450	1800	3650
Tensão [V]	400	400	480

Neste sentido, a AviSabor pretende que a equipa de manutenção tenha acesso remoto a alguns dos parâmetros registados pelo *hardware* desta sala: intensidade de corrente, potência debitada e frequência (no caso dos variadores) e pressão, recorrendo para isso a uma interface gráfica (*dashboard*). Esta deve fazer uma exibição estruturada e organizada dos dados que se pretendem monitorizar ao utilizador.

Capítulo 3

Revisão bibliográfica de Sistemas SCADA e Tecnologias IoT

Este capítulo é fundamental para o desenvolvimento do presente projeto, pois assenta numa pesquisa detalhada sobre a tecnologias de supervisão existentes atualmente no mercado. Também recorre a outras dissertações desenvolvidas nesta área, dando oportunidade ao autor da presente dissertação de conhecer e aprender metodologias e abordagens distintas e dessa forma enriquecer o projeto.

3.1 Outras Dissertações

3.1.1 Monitorização e controlo remoto de sistemas de rega agrícola

Nesta dissertação de mestrado [4] foi desenvolvido um Sistema de Monitorização e Controlo de Rega. Esta dissertação foi motivada pelo facto de não existirem sistemas de rega automáticos deste género de baixo custo, eficientes e que não exijam mão-de-obra especializada. A sua dissertação consiste num sistema dotado de possibilidade de monitorização e controlo remoto de um sistema de irrigação. Como se pode observar na Figura 3.1, o trabalho desenvolvido era composto por um dispositivo *Master*, que desempenha um papel de servidor encarregue de guardar informações relativas aos horários de rega, criação páginas *web* e gestão da base de dados, definindo os valores máximos e mínimos das variáveis de rega a controlar. O sistema de retransmissão de dados é efetuado pelo *Master* de cada campo de cultivo que funciona, também, como um ponto de ligação à *internet* (*Access Point*).

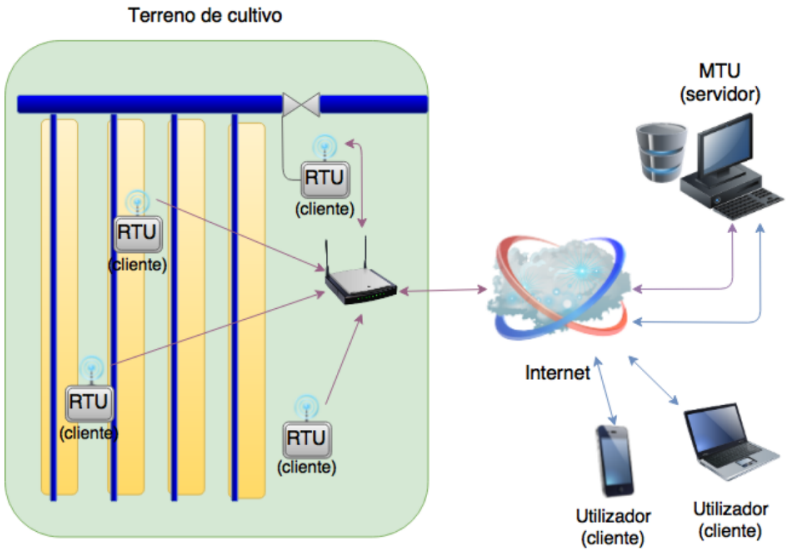


Figura 3.1: Arquitetura proposta na dissertação [4].

Por fim, o sistema é ainda composto por módulos de sensores (temperatura e humidade) e atuadores (para atuar as válvulas de rega) responsáveis pela aquisição contínua de dados e atuação em válvulas de rega, respetivamente. Os protocolos de comunicação implementados foram os seguintes:

- Modbus RTU TCP/IP ou impulsos elétricos entre os sensores e os autómatos de cada campo (*Slave*);
- HTTP entre os autómatos de cada campo e o servidor (*Master*);
- ODBC - um padrão que define um conjunto de interfaces para comunicação com bases de dados recorrendo a linguagens como C++, Java, Visual Basic [5] - entre o servidor e a base de dados.

A Figura 3.2 mostra o resultado final da arquitetura do sistema implementado. No terreno os módulos sensores foram instalados em locais em que os agricultores necessitavam de recolher informação sobre a humidade e temperatura do ar e do solo. O módulo atuador foi instalado na linha de rega principal. Periodicamente os módulos camunicavam com a unidade mestre de forma a atualizar o estado do terreno e verificar se é necessário iniciar um ciclo de rega. Por outro lado, a partir de qualquer localização com acesso à *internet*, o agricultor pode observar o estado de um terreno à escolha e alterar o seu horário de rega.

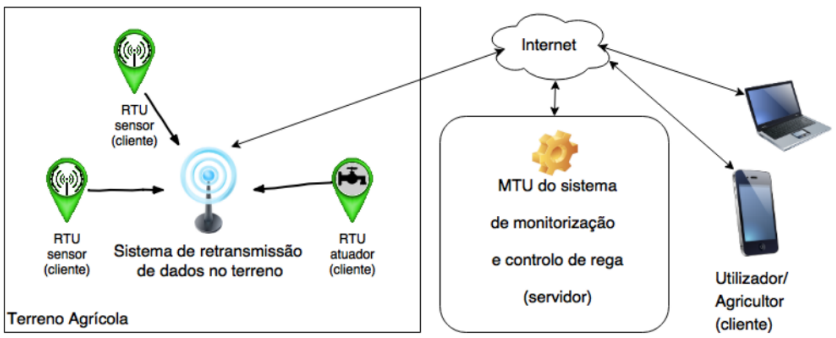


Figura 3.2: Esboço do funcionamento da solução implementada [4].

3.1.2 Proposta de uma solução SCADA para a Manutenção Preditiva

Nesta dissertação de mestrado [3], foi desenvolvido um Sistema de Monitorização SCADA para Manutenção Preditiva.

O autor desta dissertação pretendia com ela interligar os níveis de gestão mencionados na Figura 3.3, para facilitar o acesso à informação de redes industriais a partir do "chão de fábrica" (*Shop Floor*), nomeadamente sensores, atuadores, HMI, passando pelos dispositivos de controlo e supervisão até à gestão da produção. Esta solução pretendia integrar os respetivos níveis com o intuito de tornar a manutenção mais eficaz e com capacidade de previsão de falhas nos equipamentos.

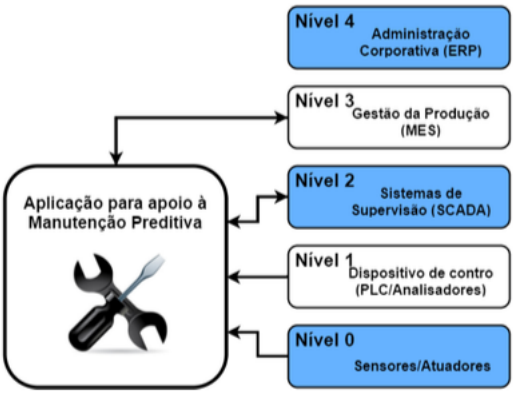


Figura 3.3: Representação da Solução proposta [3].

A Figura 3.4 é um esquema representativo da arquitetura da solução implementada.

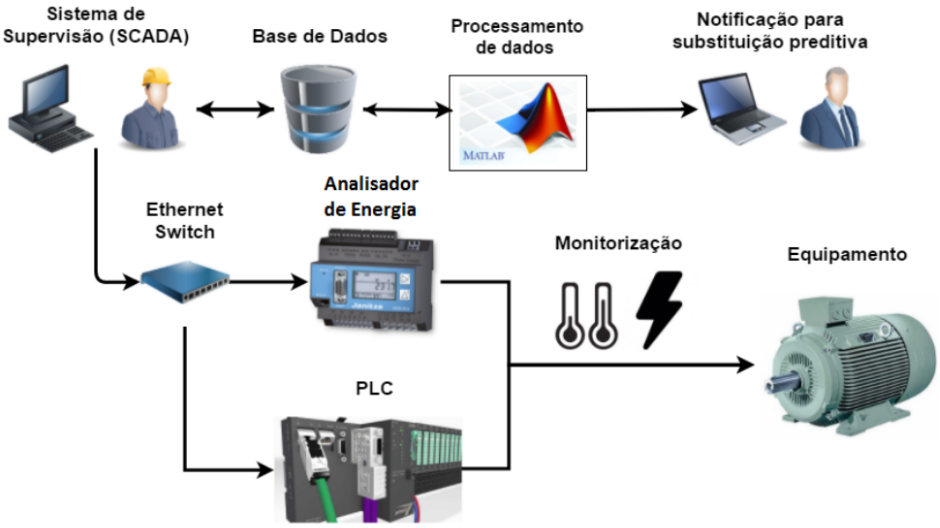


Figura 3.4: Arquitetura geral da solução proposta [3].

O trabalho começou com o desenvolvimento de pequenas aplicações SCADA recorrendo ao *software* Movicon que permite comunicações usando diversos protocolos de comunicação como TCP/IP, Siemens S7 TCP, IEC870, IEC850, Ethernet/IP entre outros. Seguiu-se a escolha do equipamento, um motor elétrico assíncrono trifásico para a realização dos testes de manutenção: testes térmicos (através de sensores de temperatura ligados ao autómato *VIPA Slio 015*) e elétricos (através do analisador de energia *Janitza UMG 604* da Figura 3.4), onde se adquiriram dados como intensidade de corrente e tensão, para serem tratados por um algoritmo em Matlab baseado em Redes Neuronais e fazer assim a previsão do comportamento do motor.

3.2 Sistemas SCADA

Os sistemas SCADA (acrónimo para *Supervisory Control and Data Acquisition*) são utilizados para monitorizar e controlar diversos equipamentos com diferentes fins: gestão de matéria-prima, controlo de estrangulamentos, comunicações, desperdícios, entre outros. A lógica de um sistema consiste em reunir informação, transferindo-a para o repositório principal de uma empresa, por exemplo o seu ERP - *software* de gestão de processos de negócio que gere e integra as os recursos financeiros, cadeias de abastecimento, operações, relatórios, fabrico e recursos humanos de uma empresa [6] - fazendo a sua análise e controlo, mostrando a informação recolhida de uma forma estruturada e de simples interpretação. No início, estes sistemas permitiam informar periodicamente o estado do processo industrial, monitorizando sinais representativos de medidas e estados de dispositivos, através de um painel de lâmpadas e indicadores [7].

Com a perceção da sua utilidade e o seu contínuo progresso, estes sistemas começaram a abranger aplicações cada vez mais diversificadas, estando presentes em diversas áreas como: a indústria de celulose, a indústria petrolífera, a indústria têxtil, a indústria metalúrgica, a indústria automóvel, a indústria electrónica, entre outras [8]. A Figura 3.5 mostra os níveis de processo de um sistema de integração de processos. No nível cinco

encontra-se a administração dos recursos da empresa, nomeadamente a gestão de vendas e gestão financeira. No nível 4 encontra-se o planeamento e o controlo da produção, realizado pela gestão da cadeia logística. Ambos os níveis dependem das informações recolhidas e armazenadas no repositório de dados do sistema SCADA, que estão relacionadas com os processos inerentes aos níveis um e dois, compostos por elementos de controlo automático das atividades da empresa, como PLC's, máquinas e atuadores e os dispositvos de entrada de sinais, como sensores [9]. Estes sistemas possibilitam a integração entre homem e máquina, permitindo ao utilizador o acesso a todas as informações do processo, assim como a possibilidade de paragens de emergência e rearme do equipamento.

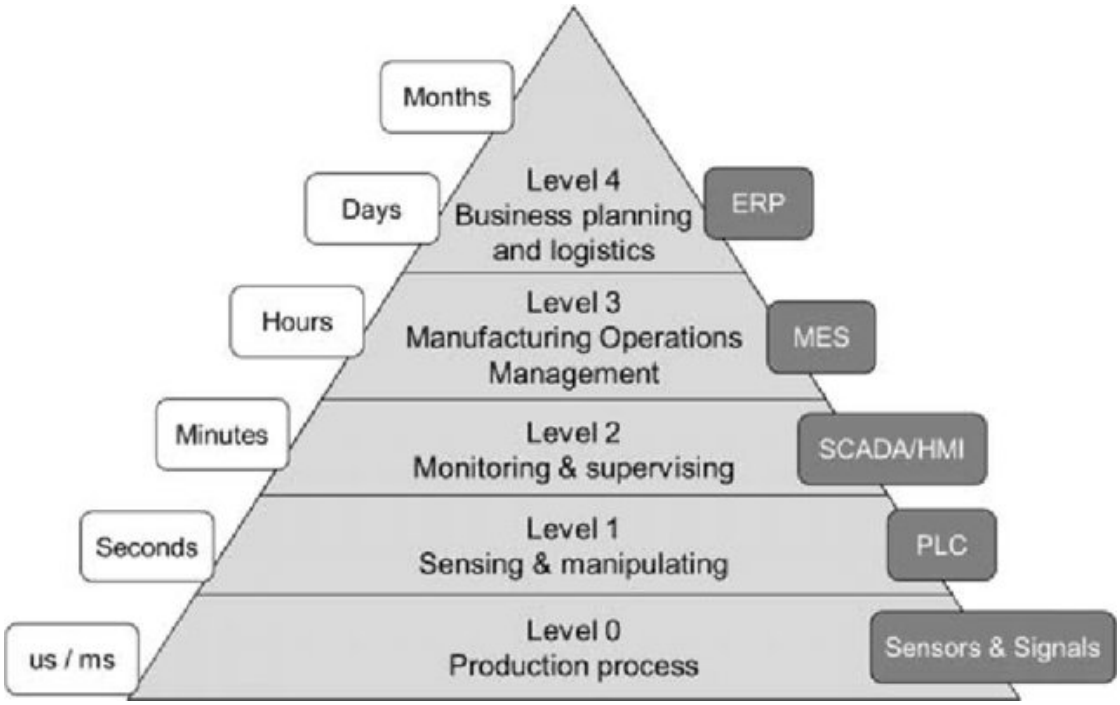


Figura 3.5: Imagem ilustrativa das camadas que compõem um sistema SCADA segundo o modelo ISA95 [10].

Os sistemas SCADA melhoram a eficiência de qualquer processo onde sejam aplicados, disponibilizando em tempo real o estado de qualquer processo, por meio de um conjunto de gráficos e relatórios, que permitem a tomada de decisões apropriadas, quer automaticamente, quer por iniciativa do operador, contribuindo desta forma para um aumento da segurança nos processos e diminuição de custos [8]. São o topo de uma arquitetura que se desenvolve desde os processos propriamente ditos até ao utilizador em qualquer lugar, sendo compostos por diversas partes que incluem normalmente *hardware* de entrada e de saída (dispositivos I/O), controladores, redes de comunicação, interface gráfica, comunicações, equipamentos e *software*.

3.2.1 Soluções Comerciais

Atualmente, existem diversos *softwares* de supervisão que se caraterizam pela vasta componente gráfica e recursos disponíveis. Grande parte destes contêm uma biblioteca de protocolos de comunicação para que seja possível realizar controlo de qualquer tipo de equipamentos existentes. Os *softwares* mais conhecidos são:

- **Siemens Simatic WinCC**: Simatic WinCC é uma aplicação desenvolvida pela Siemens para visualização de sistemas, com funções para controlo e automatização de processos. Com o Simatic WinCC são criados processos de visualização perfeitos com funcionalidades completas em controlo e monitorização para todos os segmentos da indústria. A Figura 3.6 mostra um exemplo da aparência de um programa concebido neste *software*.

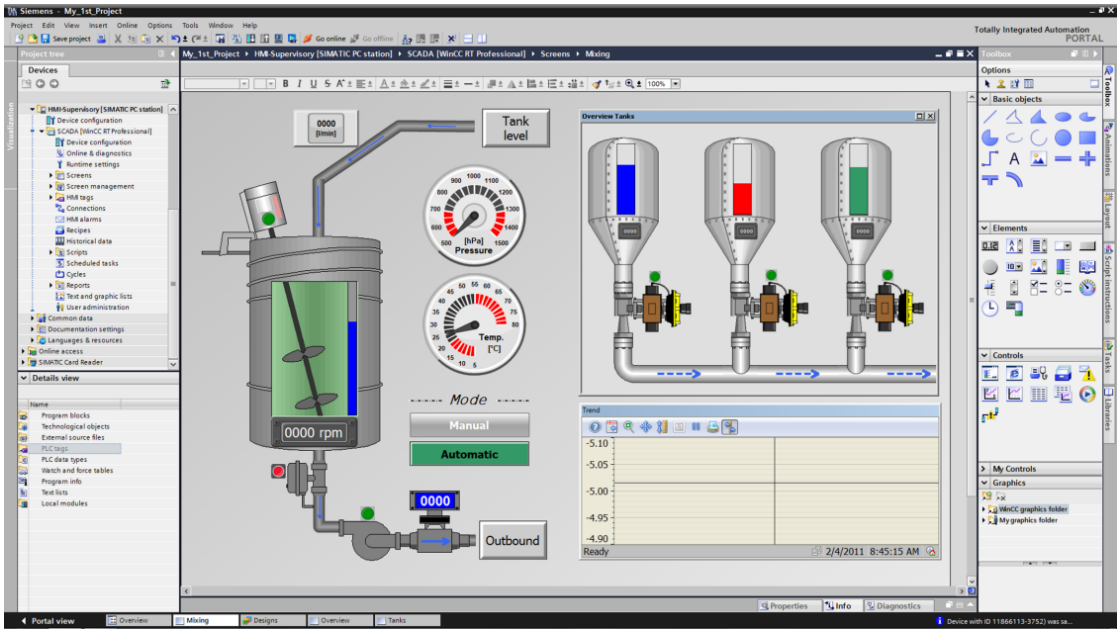


Figura 3.6: Imagem ilustrativa do *software* Siemens Simatic WinCC [11].

- **Movicon**: *Monitoring, Vision and Control* é um *software* concebido para criar interfaces. O Movicon, desenvolvido pela Progea, permite uma fácil comunicação com o processo com o qual se pretende interagir. Os componentes utilizados na gestão do processo, tais como PLC's, sensores, atuadores, robôs, etc, constituem o sistema onde o Movicon está integrado, comunicando entre si seguindo as normas dos protocolos de comunicação específicos. O intuito desta aplicação consiste em monitorizar os processos, usando animações, *synoptic Windows*, ou controlar o processo através da utilização de *dialog boxes*, bem como uma vasta gama de outras funções que permitem um controlo completo do processo fabrico. A Figura 3.7 é o exemplo de um programa em execução desenvolvido no Movicon.

Figura 3.7: Imagem ilustrativa do *Software* Movicon [12].

3.3 PLC - *Programmable Logic Controller*

Um PLC (ou autómato) é um controlador preparado para desempenhar funções de controlo e automação de máquinas ou processos industriais, que funcionam a partir de um programa neles instalado. Com alimentação, processador, entradas e saídas analógicas ou digitais e sistema operativo, um PLC usa uma memória programável para armazenar instruções e para implementar funções de sequenciamento e aritmética para controlar máquinas e processos associados. Os autómatos adquirem continuamente os valores de entrada de vários dispositivos de entrada e ativa a(s) saída(s) correspondente, dependendo do programa em execução. É um aparelho concebido especificamente para funcionar em ambientes industriais exigentes: como temperaturas elevadas, humidade, peiras, etc. Desempenham um papel crucial no campo da automação e estão presentes em grande parte dos sitemas SCADA.

Figura 3.8: Exemplo de um PLC da Siemens [13].

3.4 Variadores de Frequência

Um variador de frequência [14] é um aparelho semelhante aos da Figura 3.9. Um variador controla a frequência da potência fornecida a um motor AC para controlar a velocidade de rotação (normalmente dada pelo número de revoluções por minuto - **RPM**) e também a aceleração do veio acoplado. Sem um variador, um motor AC operaria na sua velocidade nominal logo que se ligasse a sua fonte de alimentação. Assim, um variador tem como objetivo o ajuste da velocidade e aceleração de um motor AC.

Figura 3.9: Imagem representativa de uma gama de variadores de frequência Delta [15].

3.4.1 Conceitos Técnicos Utilizados

Esta secção pretende descrever alguns conceitos que são discutidos e utilizados posteriormente neste projeto.

- **Tensão de Saída**: a tensão aos terminais de um variador;
- **Corrente de Saída**: intensidade de corrente que flui aos terminais de um variador;
- **Frequência de Saída**: frequência da tensão aos terminais de um variador.

3.5 Microcontroloadores

Com a introdução do conceito da Indústria 4.0, veio a necessidade de comunicação entre sistemas de produção no chão de fábrica e o mundo virtual, uma lacuna que empresas como a Espressif Systems vieram preencher, em que os seus dispositivos se destacam pela sua velocidade e custos reduzidos. A Espressif Systems produz microcontroladores, idênticos aos da Figura 3.10, estes são pequenos dispositivos (idênticos aos Arduinos e também compatíveis com o *software* ArduinoIDE) mais baratos que os tradicionais PLC's, mais simples, flexíveis e versáteis, ideais para aplicações IoT. São dotados de capacidades de comunicação via Wi-Fi e *bluetooth* para que qualquer utilizador os possa utilizar para a concretização dos mais variados tipos de projetos.

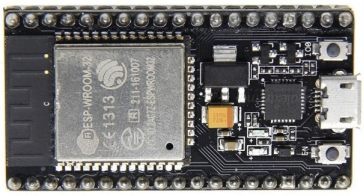


Figura 3.10: Módulo ESP32 [16].

3.6 IoT - *Internet Of Things*

A Internet das Coisas IoT está a transformar a maneira como as organizações contactam com os clientes, fornecedores, sócios e outros, representando uma mudança e oportunidade significativas em qualquer indústria. O avanço na tecnologia das comunicações abriu portas para a integração da tecnologia operacional tradicional (OT - *Operational Technology*) de uma empresa com o sistema de tecnologia da informação (IT - *Information Technology*) da mesma, contribuindo para uma maior flexibilidade face aos requisitos dos clientes [17].

Cada sistema IoT é diferente, no entanto, a arquitetura do sistema é transversal às suas diversas aplicações. Resumidamente, consiste num objeto ligado à rede que, via sensores, reúne informações específicas sobre o ambiente que o rodeia que são depois passadas aos *gateways* (que atuam como "intermediário" entre os dispositivos e redes com diferentes protocolos de comunicação). A fase seguinte é a aquisição, em que se reúne uma grande quantidade de informação não processada, convertendo-a num fluxo digital, filtrado e pré-processado, pronto a ser analisado. A terceira e última fase é

representada por *edge devices* (equipamentos que servem como ponto de entrada nas redes de comunicação de uma empresa) responsáveis por um processamento e amostragem de dados processados . Todos estes dados são posteriormente guardados numa base de dados que pode ser local (servidor local) ou numa *cloud*. A arquitetura é explicada com maior detalhe na Secção 3.6.1.

3.6.1 Arquitetura IoT

A tecnologia IoT possibilita a ligação de objetos do dia-a-dia, como: eletrodomésticos, carros, máquinas de café, termostatos, à *internet* através de dispositivos para o efeito, procurando desta forma uma simbiose entre pessoas, processos e objetos.

Através de computação de baixo custo, *cloud*, *big data*, análise e tecnologias móveis, os objetos físicos podem partilhar e armazenar informação com uma intervenção humana mínima. Esta realidade permite uma comunicação constante, onde os sistemas digitais podem registar, monitorizar e ajustar cada interação entre objetos ligados. Um exemplo de uma solução conceptual comum, esquematizada na Figura 3.11 é caracterizada por vários dispositivos que usam um *gateway* (atua como um ponto de acesso para sensores e atuadores e gere as comunicações inter-dispositivos e com uma rede externa e "traduz" de forma bidirecional os dados trocados) para comunicar entre si numa rede e com um servidor que execute uma plataforma IoT que ajuda a integrar todos os dados existentes [18].

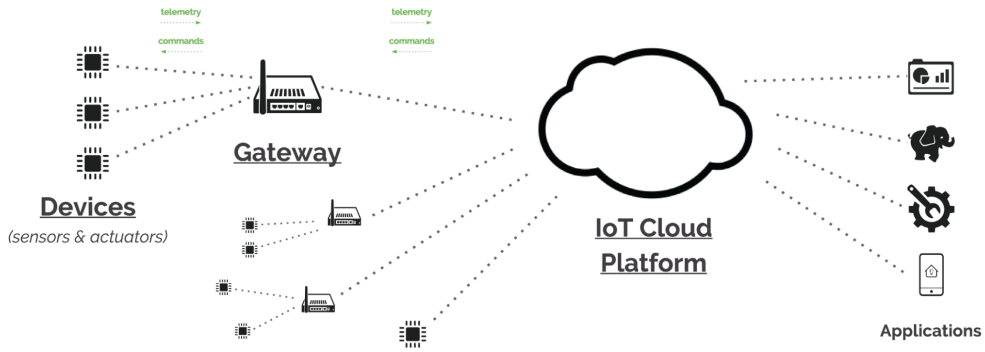


Figura 3.11: Arquitetura de uma implementação IoT [19].

3.6.2 Plataformas IoT

Uma plataforma IoT é uma tecnologia que permite a provisão, gestão e automação de dispositivos ligados dentro do universo da Internet das Coisas. Pode-se considerar um *middleware* no que toca à forma sobre como se liga os dispositivos remotos às aplicações do utilizador e gere todas as interações entre o *hardware* e as camadas da aplicação. Para programadores, uma plataforma IoT fornece um conjunto de recursos prontos a ser utilizados que aceleram o desenvolvimento de aplicações.

3.6.2.1 Microsoft Azure IoT Suite

A Microsoft Azure [20] é uma ferramenta de computação na *cloud* que fornece de serviços de análise, armazenamento e comunicações. Veja-se a Figura 3.12.

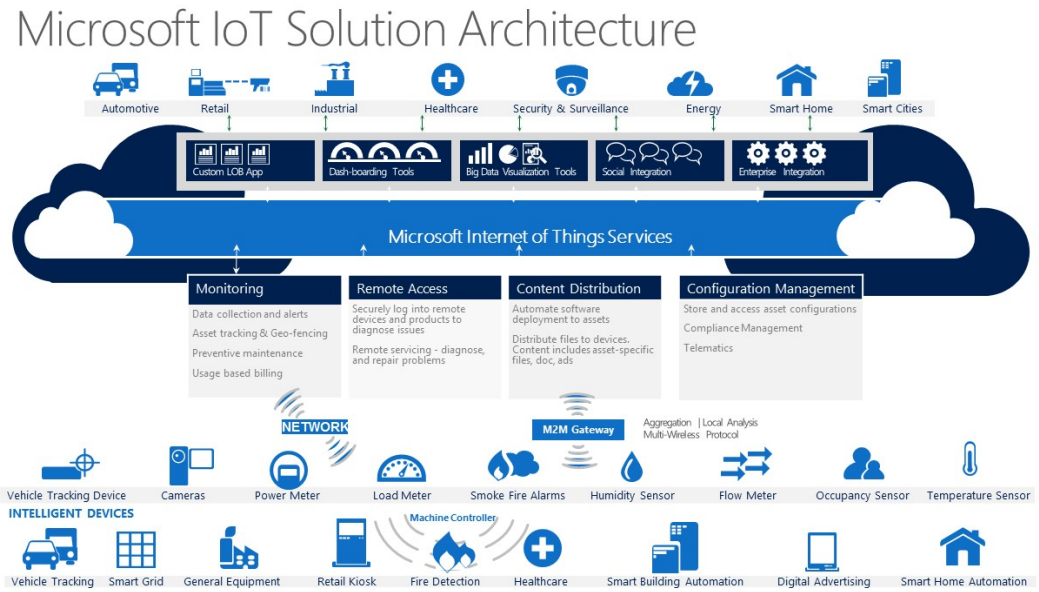


Figura 3.12: Arquitetura do serviço Microsoft Azure IoT [21].

Uma das soluções mais interessantes da Azure IoT Suite é a monitorização remota, que simplifica a monitorização dos dados obtidos pelos dispositivos, a pesquisa de características específicas nos dados adquiridos e a execução de processos. Com base nesta solução, são criadas instâncias dos seguintes serviços da Azure:

- **IoT Hub**: é a peça central de uma solução IoT da Azure. Serve como *cloud gateway* (serve de "intermediário" que traduz a informação trocada de forma bidirecional, não especifica nenhum protocolo de comunicação) para estabelecer a ligação com uma grande quantidade de dispositivos, assim como para processar essa quantidade de dados. Suporta protocolos de comunicação como o **HTTP** e o **MQTT**;
- **Stream Analytics**: é um serviço de processamento de dados em tempo real que deteta incongruências e arquiva dados dos dispositivos IoT. Os resultados podem ser partilhados com outros serviços, tais como *Event Hubs*, *Power BI* ou serviços de armazenamento;
- **Blob Storage**: é um local para armazenamento dos dados que os dispositivos recolhem e enviam para a *cloud*;
- **DocumentDB**: gere o estado dos dispositivos, tais como configurações, estado e propriedades de segurança;
- **Web App**: encarregue de alojar uma aplicação *web* que analisa o painel de dados do dispositivo. Configura e envia comandos para os dispositivos, cria e atualiza a lógica de negócios e executa ações orientadas a eventos, como o envio de alarmes quando determinados limites forem atingidos;
- **Logic App**: ajuda a integrar a solução IoT na infraestrutura existente e automatizar o processo de fluxo de trabalho. Este serviço permite que a equipa de

desenvolvimento projete fluxos de trabalho que se iniciam com um *trigger* e executam uma série de etapas.

3.6.2.2 Siemens MindSphere

O Siemens MindSphere [22] é uma plataforma IoT aberta e baseada na *cloud* da Siemens que interliga os seus produtos, fábricas, sistemas e máquinas, permitindo o aproveitamento dos dados gerados pela Internet das Coisas (IoT), veja-se a Figura 3.13. Este guarda todo o tipo de informação associada a uma indústria e disponibiliza-a em aplicações, as MindApps, dando a liberdade aos clientes de tomarem decisões baseadas em valores reais. Esta pode ser usada para otimização de produtos, no decorrer da produção, em processos de fabrico, entre outras aplicações que possam beneficiar dos dados recolhidos.

O MindSphere é um sistema usado de forma recorrente no ramo da automação industrial e na gestão da frota de automóveis.

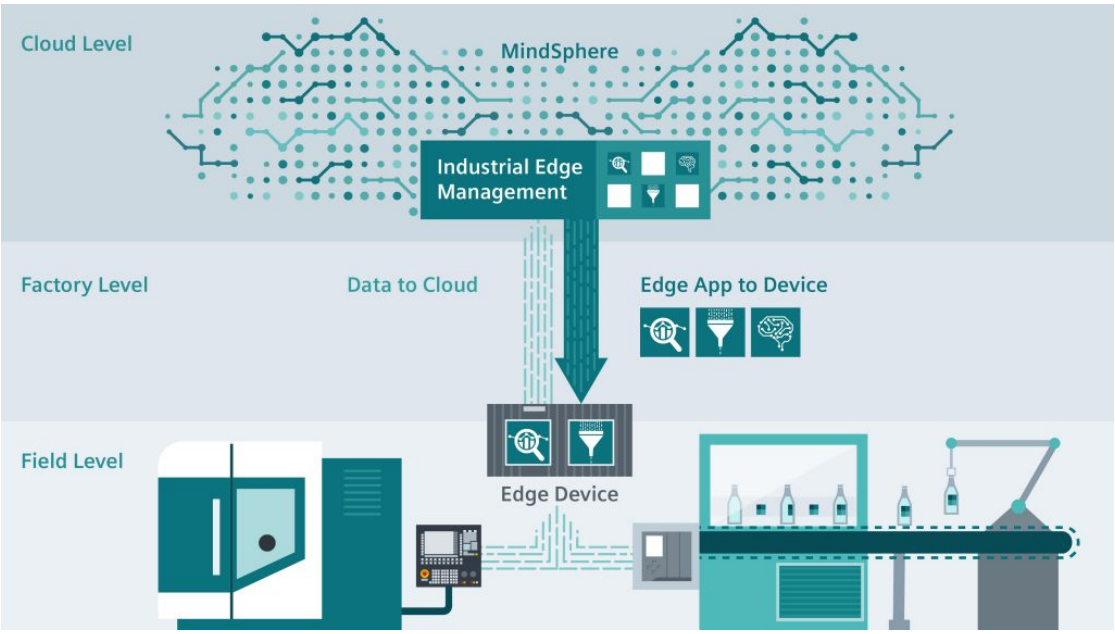


Figura 3.13: Arquitetura do serviço Siemens Mindsphere [23].

3.6.2.3 Bosch IoT Suite

A Bosch IoT Suite [24] é a plataforma IoT da Bosch baseada em ferramentas *open source*. O seu núcleo é construído sobre os projetos do Eclipse IoT, que fornecem a tecnologia necessária para criar uma plataforma de nível comercial. Observe-se a Figura 3.14, para se perceber as interações que ocorrem entre todos os dispositivos que operam no seu conjunto para formar o Bosch IoT Suite.

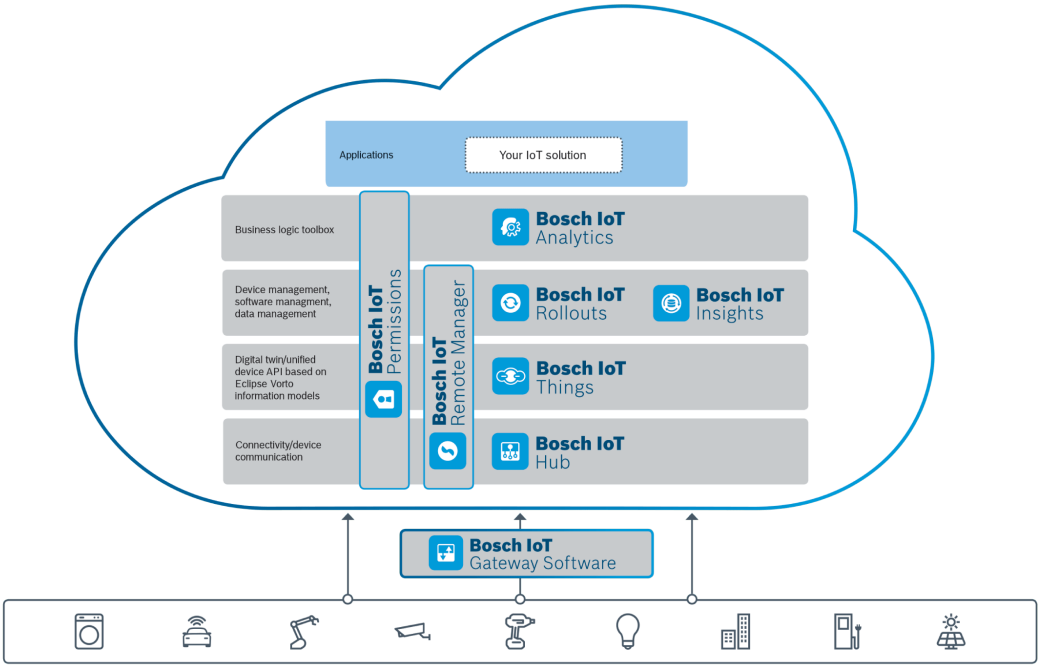


Figura 3.14: Arquitetura do serviço Bosch Suite IoT [25].

A partir desta plataforma a Bosch, os seus clientes e parceiros podem criar uma ampla variedade de soluções, serviços e projetos IoT, nomeadamente:

- **Eclipse Ditto**: é o projeto base onde os dispositivos IoT e os seus *digital-twins* se reúnem.
- **Eclipse hawkBit**: consiste numa solução para implementar atualizações de *software*. É a base do Bosch IoT Rollouts;
- **Eclipse Hono**: consiste em interfaces de serviço remoto às quais se pode ligar um grade número de dispositivos.
- **Eclipse Vorto**: permite descrever os recursos abstratos e a funcionalidade dos dispositivos como *Informations Models*. A Bosch IoT Suite usa este modelo para normalizar os dados adquiridos pelos dispositivos na utilização de API's.

Observe-se a Figura 3.14, para se perceber as interações que ocorrem entre todos os dispositivos que operam no seu conjunto para formar o Bosch IoT Suite. A partir desta plataforma a Bosch, os seus clientes e parceiros podem criar uma ampla variedade de soluções, serviços e projetos IoT.

3.7 *Softwares* e Linguagens de Desenvolvimento *web*

- **Arduino IDE**: é uma plataforma aberta (*open source*) que possui uma linguagem própria com base em C. Existem inúmeras bibliotecas livres que aceleram o desenvolvimento de produto para a resolução de problemas mundanos. Uma vez que é aberto a uma comunidade *online* C/C++, e disponibiliza ainda um ambiente integrado de desenvolvimento (IDE - *Integrated Development Environment*) para facilitar a programação de microcontroladores compatíveis [26].

- **CSS**: *(Cascading Style Sheets)*, é uma linguagem para adicionar um estilo (cores, tipo de letra, espaçamento) a documentos *web* [27].

- **HTML**: *HyperText Markup Language*, é a linguagem utilizada para o desenvolvimento de páginas *web*. Foi criada por Tim Berners-Lee e formalizada em 1990. É um documento composto por *tags* que é interpretado pelos navegadores web de forma a criar no ecrã uma página legível e organizada. Atualmente é utilizada a versão HTML5 que é comummente utilizada em conjunto com **Javascript** e **CSS** [28].

- **HTTP**: *Hypertext Transfer Protocol* é a base para a comunicação de dados, sendo usado desde 1990 na World-Wide Web. Este protocolo funciona com base no modelo de solicitação e resposta. Um cliente envia uma solicitação ao servidor, composta pelo tipo de solicitação, URI (*Uniform Resource Identifier*) e a versão do protocolo. A solicitação pode conter ainda informações do cliente e um corpo com conteúdo. Por sua vez, o servidor responde enviando uma *status line* composta pela versão do protocolo e o código de sucesso ou de erro. A resposta pode ainda ser constituída por informações do servidor e pelo conteúdo solicitado pelo cliente [29].

- **JavaScript**: é uma linguagem de programação interpretada estruturada de alto nível. Juntamente com HTML e CSS, o JavaScript é uma das três principais tecnologias da World Wide Web. Permite o desenvolvimento páginas da Web interativas e é assim uma parte fundamental de aplicações *web*. A grande maioria dos sites usa, e todos os principais navegadores têm um mecanismo JavaScript dedicado para a sua execução [30]. Depois de enviada a resposta a ligação é terminada.

- **phpMyAdmin** : é um aplicação *web* livre e de código aberto desenvolvido em PHP para administração de uma base de dados MySQL pela *internet*. A partir deste sistema é possível criar e remover bases de dados, criar, remover e alterar tabelas, inserir, remover e editar campos, executar códigos SQL e manipular campos chaves. O phpMyAdmin é muito utilizado por programadores que, frequentemente, necessitam manipular bases de dados [31].

Capítulo 4

Solução Proposta

4.1 Descrição da Solução

Neste capítulo é idealizada uma solução conceptual para o desafio proposto de monitorização e controlo que dê resposta às necessidades da equipa de manutenção da AviSabor, S.A.

A Figura 4.1 representa a arquitetura do funcionamento da solução conceptual. Mostra como os dispositivos interagem uns com outros, segundo que protocolos e com que finalidade. Começando pela camada inferior, o Sistema de Aquisição (Secção 5.3) extrai os dados de frequência, intensidade de corrente e tensão contidos em registos específicos do variador de frequência a partir do módulo IoT (ESP32). Estes dados seguem via **MQTT** para o servidor (representado na camada do meio), mais concretamente para o ambiente de programação Node-RED, no qual são processados e armazenados numa base de dados MySQL. O utilizador, veja-se a camada superior, tem acesso a estes dados através de uma página *web* alojada no Node-RED, desenvolvida com a *framework bootstrap vue*. A camada do meio é ainda composta por um Modo Automático de Regulação de Frequência na qual se pretende estudar o desempenho do sistema em funcionamento autónomo, recorrendo a um ***input*** que é o valor analógico de um potenciómetro (para simular um sensor de pressão). Cada parte do sistema é explicada com maior detalhe nas Secções que se seguem.

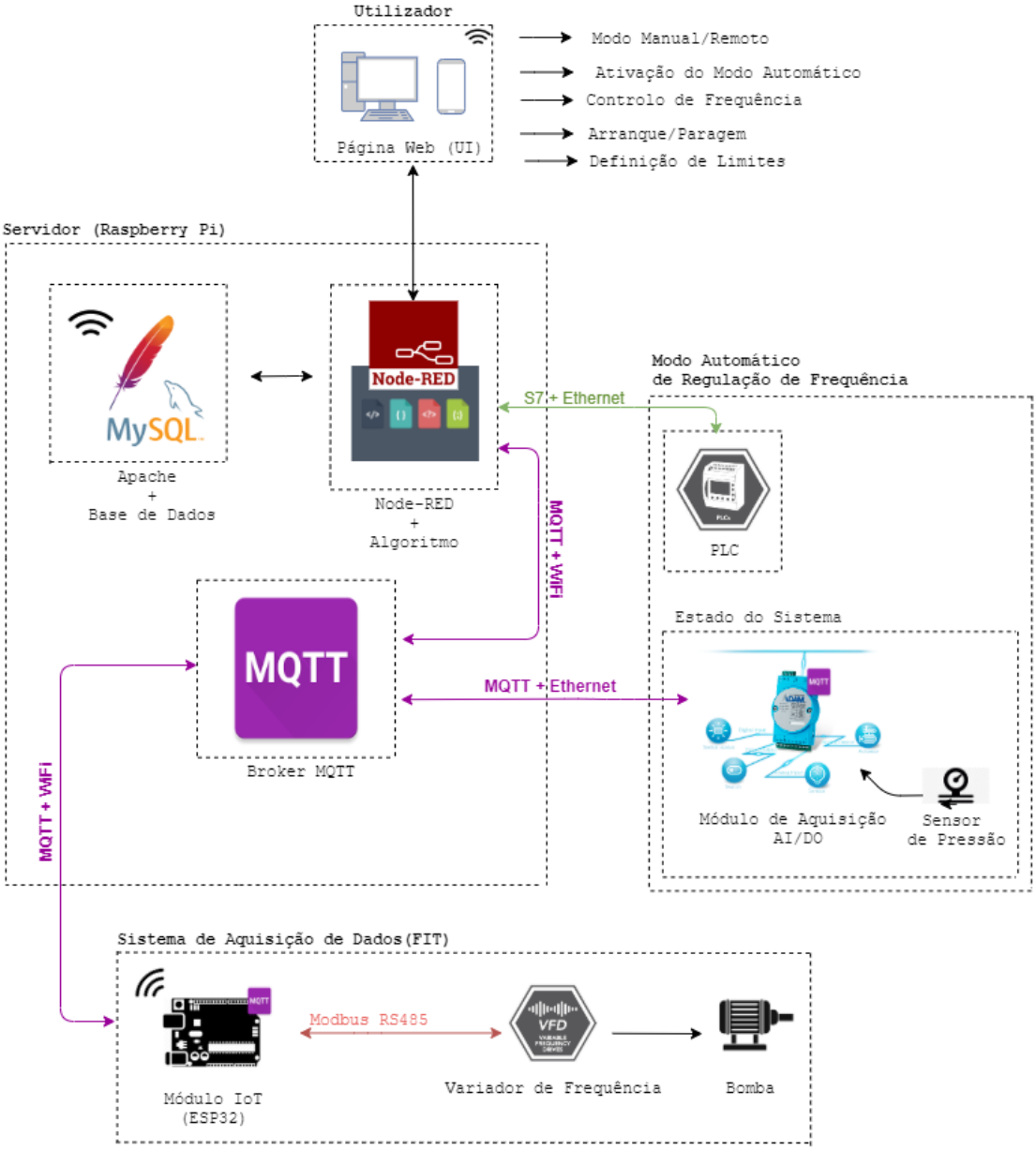


Figura 4.1: Representação esquemática da solução proposta.

4.2 Abordagem Inicial para o Sistema de Aquisição de Dados

Em ambiente industrial há uma conjunto de variáveis que podem interferir no funcionamento de qualquer aparelho e portanto no sistema do qual ele faz parte, uma supervisão e controlo permanente permitem uma manutenção mais eficaz, com menores tempos de reação no processo produtivo e facilita a resolução atempada de quaisquer problemas que possam surgir, *troubleshooting*.

A solução idealizada para este projeto assenta nos mesmos princípios da solução implementada na dissertação [4], que propõe um sistema de supervisão e controlo. No caso das bombas, dados como o caudal, a pressão de saída, o nível dos reservatórios, são dispostos num painel HMI da Siemens, ligado a um *switch Scalance XB005* que gere as comunicações entre um PLC S7-1200 da mesma marca e variadores de frequência da Delta.

Inicialmente, a solução passava pela tentativa de interseção dos sinais que chegam ao autómato usando o *software* ***Wireshark***, um programa que analisa o tráfego de rede, e o organiza por protocolos, permitindo controlar a entrada e saída de dados do computador, segundo diversos protocolos, ou da rede à qual o computador está ligado [32]. Outra solução seria o desenvolvimento de um programa para um microcontrolador e, ligando-o ao barramento de dispositivos, obter os dados pretendidos, armazená-los num servidor e mostrá-los numa página *web*. O *hardware* da empresa com que se pretendia trabalhar inicialmente seria:

1. PLC: Siemens S7-1200 - Imagem (a) da Figura 4.2;
2. *Switch*: Siemens Scalance XB005 - Imagem (b) da Figura 4.2;
3. HMI: Siemens KTP400 Basic - Imagem (c) da Figura 4.2;
4. Variador de Frequência: Delta C2000 - Imagem (d) da Figura 4.2.

(a) (b) (c) (d)

Figura 4.2: *Hardware* da empresa: (a) PLC Siemens S7-1200, (b) Switch Siemens Scalance XB005, (c) Siemens KTP400, (d) Variador de Frequência - Delta C2000.

No entanto, chegou-se à conclusão que para avançar com o projeto e se conseguir simular o problema proposto, dever-se-ia utilizar material idêntico disponível em labo-

ratório (explicado detalhadamente na Secção 5.5), pelo que se substituiu o equipamento supramencionado por:

1. PLC: Siemens ET200S - Imagem (b) da Figura 5.20;

2. *Router*: TP-LINK WR-841N - Imagem (g) da Figura 5.20;

3. Variador de Frequência: Mitsubishi FR-D720S - Imagem (c) da Figura 5.20.

Omitiu-se o painel HMI uma vez que este pode ser facilmente substituído por uma página *web online*.

A evolução tecnológica tem seguido uma tendência cada vez mais marcante, a IoT, cujo objetivo consiste em dotar os objetos mundanos de capacidade de processamento e comunicação. Os microcontroladores (μC) vieram responder a esta necessidade por serem dispositivos de pequenas dimensões com entradas e saídas digitais e com possibilidade de comunicação *wireless*.

Na Figura 4.1 da arquitetura da solução proposta está omitido o *router*. No entanto, todos os aparelhos estão interligados a uma rede local (criada pelo *router*) para que possam comunicar entre si, sempre que necessário, sem recorrer a ligações físicas. O sistema a desenvolver deverá apresentar capacidades para:

- aquisição e interpretação de mensagens série Modbus-RTU-RS-485 do variador de frequência do sistema de bombeamento;

- armazenamento de dados na *web* dos valores de frequência, intensidade de corrente, tensão e limites que fazem disparar os alarmes;

- mostrar os dados recolhidos numa interface gráfica multi-plataforma para o utilizador, baseada em linguagens *web* ;

- envio de alarmes via *e-mail* para os técnicos no caso de irregularidades no funcionamento dos dispositivos.

4.3 Sistema de Aquisição de Dados (Frequência, Intensidade de Corrente, Tensão)

O Sistema de Aquisição de Dados da Figura 4.3 deve ser capaz de comunicar com o variador de frequência por forma a obter os valores de frequência de comando, intensidade de corrente e tensão debitada armazenados nos registos internos do mesmo, enviando, por sua vez, esses dados para um servidor. Toda a programação deste sistema é escrita na memória do microcontrolador cujo funcionamento se rege por um algoritmo concebido para o efeito.

Figura 4.3: Arquitetura conceptual do Sistema de Aquisição de Dados.

4.3.1 Comunicação - Modbus-RTU-RS-485 [1]

A Figura 4.4 representa as interação física que ocorre a este nível.

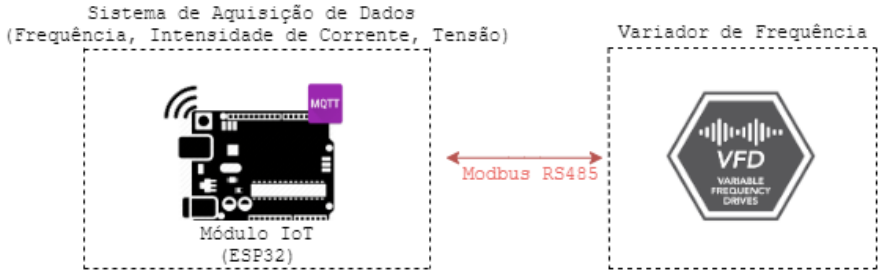


Figura 4.4: Comunicação entre o microcontrolador e o variador.

O protocolo Modbus foi proposto em 1979 pela Modicon. Carateriza-se por não definir um meio físico de transmissão da informação, mas sim a forma como a informação entre *Master* e *Slave* é trocada, ou seja, este protocolo define como as mensagens são trocadas pelos equipamentos ligados no barramento.

Este protocolo apresenta ainda três formas de envio de mensagens: RTU, ASCII e TCP. A nível industrial é muito comum a utilização do padrão RS-485 para definir a estrutura física das mensagens série. Neste projeto é utilizado somente Modbus-RTU, uma vez que é protocolo segundo o qual o variador comunica, veja-se a Figura 4.4.

4.3.1.1 Meio de transmissão

A norma RS-485 enuncia que devem ser utilizados, no mínimo, um par de fios entrelaçados para uma comunicação do tipo *Half-Duplex* ou dois pares para uma comunicação *Full-Duplex*. No tipo *Half-Duplex*, existe apenas um equipamento a enviar dados num determinado instante, enquanto que no segundo, *Full-Duplex* podem ser utilizados múltiplos equipamentos para enviar dados ao mesmo tempo. O tipo de comunicação mais frequente é a *Half-Duplex*, recorrendo a três fios (A, B e GND) que podem estar ligados até trinta e dois equipamentos num barramento. Esta norma estabelece que a velocidade de comunicação que pode ser usada pode atingir a taxa de transferência máxima de 10

Mbits/s para cerca de 15 m, sendo que o comprimento máximo de um barramento é 1200 m (4000 ft, seguindo a linha azul da Figura 4.5).

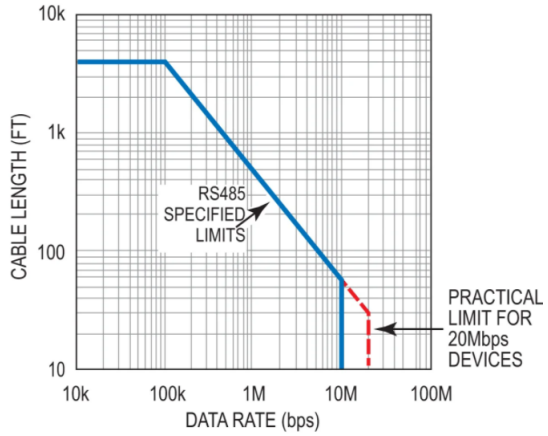


Figura 4.5: Relação entre o comprimento do cabo e a velocidade de comunicação RS-485 [33].

4.3.1.2 Tensões

Os equipamentos devem aplicar uma tensão positiva para enviar um *bit* com o valor lógico "0" e uma tensão negativa para enviar um *bit* com o valor lógico "1". O valor de tensão deve pertencer ao intervalo entre +1,5 e +6 Volt. Considera-se que um emissor envia um *bit* com o valor lógico "1" quando se aplica uma tensão no condutor U+ inferior à tensão que aplica no condutor U- pertencente ao intervalo entre -1,5 e -6 volt. Devido à atenuação de sinal, a diferença de potencial entre os condutores pode diminuir, no entanto, o seu módulo não deve ser inferior a 1,5V.

A norma RS-485 determina que a diferença entre a tensão do fio A (V_A) e a tensão do fio B (V_B) não deve exceder os 200 mV quando não existe transmissão de informação, para que o barramento possa tolerar ruído e o recetor ignorar o sinal.

4.3.1.3 Palavras Séries

Na Tabela 4.1 descrimina o potencial da linha de transmissão de dados nas várias situações que acontecem.

Tabela 4.1: Potencial da linha com e sem atividade.

Estado	**Potencial**
Idle	Não há atividade na linha, a diferença de potencial é de 200 mV.
Start Bit	A tensão do condutor U+ é inferior à tensão do condutor U-.
Bit a 1	A tensão do condutor U+ é inferior à tensão do condutor U-.
Bit a 0	A tensão do condutor U+ é superior à tensão do condutor U-.
Stop Bit	A tensão do condutor U+ é superior à tensão do condutor U-.

A Figura 4.6 mostra a diferença de potencial entre os condutores **A** e **B** no envio de uma mensagem série.

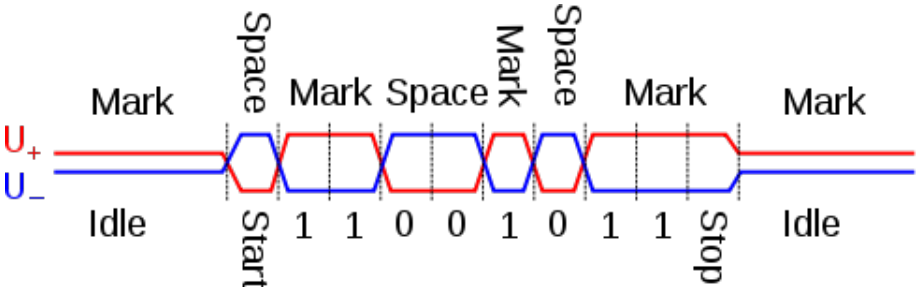


Figura 4.6: Tensões das linhas A e B no envio de um *byte* [34].

Na Figura 4.7 pode-se ver a estrutura das mensagens série trocadas: o primeiro *byte* corresponde à identificação do *slave* para o qual a mensagem é enviada, o segundo *byte* corresponde à função da mensagem (leitura - 03_H e escrita - 06_H) e os restantes contêm os dados a ser transmitidos e os últimos dois *bytes* correspondem ao *CRC* (*Cyclic Redundancy Check*) ou Verificação Cíclica de Redundância, para que os equipamentos saibam se receberam a mensagem sem erros. Este *CRC* é calculado usando os restantes *bytes* da mensagem.

Endereço do Slave	Função	Dados		CRC	
8 bits	8 bits	L 8 bits	H 8 bits	L 8 bits	H 8 bits

Figura 4.7: Estrutura de uma resposta ao envio de uma mensagem série.

Seguindo um exemplo do manual de um variador Mitsubishi, para ler o valor da frequência de comando (endereço do registo - $000D_H$ e somente esse registo - 0001_H) com endereço de *slave* 1, deverá ser enviado o conteúdo da seguinte figura:

Endereço do Slave	Função	Endereço do Registo		Valor		CRC	
H01	H03	H00	H0D	H00	H01	H15	HC9

Figura 4.8: Exemplo de leitura de um registo.

Seguindo outro exemplo do manual, para escrever um valor de 60 Hz (1770_H, valor hexadecimal) no registo 40014 (endereço do registo da frequência de comando - $000D_H$) no endereço de *slave* 5, seria enviado o conteúdo da Figura 4.9:

Endereço do Slave	Função	Endereço do Registo		Valor		CRC	
H05	H06	H00	H0D	H17	H70	H17	H99

Figura 4.9: Exemplo de escrita de valor de frequência num registo.

4.3.1.4 Resistências

Resistências de Terminação

Para reduzir a distorção do sinal transmitido são utilizadas resistências na extremidade dos dois condutores do par entrelaçado (R_c e R_b), em que cada resistência liga um condutor ao outro.

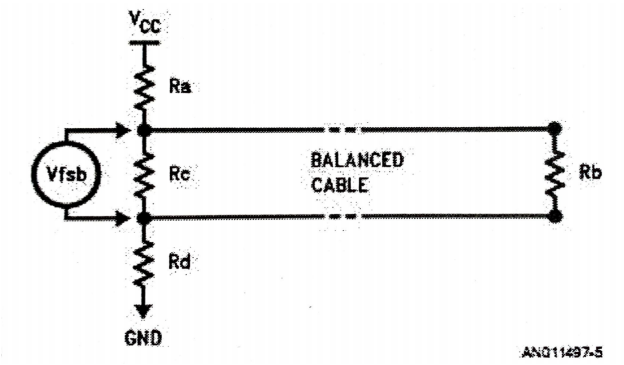


Figura 4.10: Resistências de terminação, *pull-up* e *pull-down* [1].

Resistências de *pull-up* e *pull-down*

Enquanto nenhum equipamento enviar dados os seus amplificadores operacionais de envio mantêm-se num estado de alta impedância, ou seja, tudo se passa como não estivessem eletricamente ligados ao par de condutores.

$$V_{\mathrm{Rc//Rb}} = I \times (R_{\mathrm{a}}//R_{\mathrm{b}}) = 200mV \tag{4.1}$$

De acordo com a Lei de Ohm, pode-se calcular a corrente que atravessa estas resistências. Considerando que todos os equipamentos estão num estado de alta impedância, e assumindo uma tensão de alimentação V_{cc} igual a 5V, para que a diferença de potencial seja inferior a 200mV as resistência R_a e R_b devem ter um valor superior a 727Ω, por exemplo, cada uma delas deve valer 820Ω.

A corrente que deve passar no paralelo $R_a//R_b$ para que a tensão aos seus terminais seja igual a 200mV, é dada pela expressão:

$$I = \frac{200}{R_{\mathrm{a}}//R_{\mathrm{b}}} = \frac{0,2}{60} = 3,3mA \tag{4.2}$$

Em que $R_a = R_b = 120\Omega$ e $R_a//R_b = 60\Omega$, através da seguinte expressão:

$$R_{eq} = \frac{1}{(\frac{1}{R_1}) + (\frac{1}{R_2}) + ...(\frac{1}{R_n})} \tag{4.3}$$

A tensão aos terminais da resistência *pull-up* V_{R_a} deve ser igual à tensão aos terminais da resistência de *pull-down* V_{R_d} e no seu conjunto a soma da tensão aos terminais das resistências de *pull-up* e *pull-down* e resistências de terminação tem de ser igual a V_{cc}. Veja-se as seguintes expressões:

$$V_{cc} = V_{R_a} + V_{R_b//R_c} + V_{R_d} \tag{4.4}$$

$$V_{R_a} = \frac{(V_{cc} + (V_{R_b//R_c})}{2} = \frac{5 - 0,2}{2} = 2,4V \tag{4.5}$$

Para que a tensão aos terminais da resistência de *pull-up* seja igual a 2,4V, quando percorrida por uma corrente 3,3mA a percorre, o seu valor tem que ser igual a:

$$R_a = \frac{U}{I} = \frac{2,4}{0,0033} = 727\Omega \tag{4.6}$$

4.3.1.5 Tempos

Tempo de Envio de um *byte*

Cada *byte* de uma mensagem ModBus demora um tempo conhecido a ser enviado que depende da taxa de transferência. Para além dos *bits* de dados, é enviado ainda um *start bit*, um *bit* de paridade um *stop bit*. Assim o tempo de envio será dado pela seguinte equação:

$$T_{byte} = \frac{n_{bits} + 3}{T_{Transferência}} \tag{4.7}$$

Tempo de Espera entre *bytes*

Todos os *bytes* da mensagem têm de ser enviados consecutivamente, e preferencialmente sem tempo de espera entre eles. Se o tempo decorrido entre o envio de um *byte* e o *byte* seguinte exceder 1.5 vezes o tempo de envio, essa mensagem é ignorada. Por exemplo, com uma taxa de transferência de 9600 bps, o tempo máximo de espera é 1,71 ms.

$$T_{\text{Espera entre mensagens}} \leq 1,5 \times T_{byte} \leq 1,71ms \tag{4.8}$$

Tempo mínimo de espera entre mensagens

O protocolo preconiza que deve existir um tempo de silêncio igual ou superior a 3.5 vezes o tempo de envio de um *byte* Modbus. Com uma taxa de transferência de 9600bps o tempo mínimo de espera entre mensagens é 4 ms.

$$T_{\text{Espera entre bytes}} > 3,5 \times T_{byte} > 4ms \tag{4.9}$$

4.3.2 Comunicação com o Servidor

A comunicação entre o Sistema de Aquisição de Dados e o Servidor é feita através de uma ligação Wi-Fi, ou física usando um cabo *ethernet*. Nas comunicações Wi-Fi (IEEE 802.11) os dispositivos que se queiram ligar à rede, designados *station* (STA), têm que se ligar a dispositivos que funcionem como *Access Point* (Pontos de Acesso), ou *routers*. Estes requerem credenciais de acesso, o Service Set Identifier (SSID) e a *password* do ponto de acesso ao qual se pretenda ligar.

4.4 Servidor (Raspberry Pi)

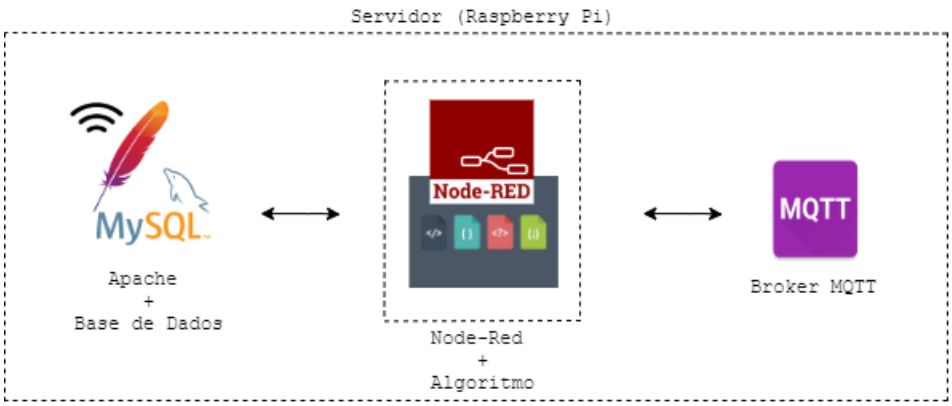


Figura 4.11: Arquitetura sonceptual do Servidor.

O servidor (Figura 4.11) deve receber os dados enviados pelo Sistema de Aquisição de Dados da Secção 4.3, de armazenar esses valores e os mostrar ao utilizador na página *web*, que servirá de *UI* (*User Interface* - Interface do Utilizador, descrita com maior detalhe na Secção 4.5). Deve ainda ser capaz de analisar e processar os dados recebidos e, quando necessário, enviar alarmes para os utilizadores via e-mail. Para que os utilizadores possam visualizar os dados na Interface do Utilizador, o servidor terá que conter os ficheiros necessários para a criação dessa interface. Dadas as condições supramencionadas, optou-se pela utilização um RaspberryPI 3 para servir este propósito, visto reunir um conjunto de caraterísticas que o tornam adequado para o efeito.

4.4.1 *Broker* MQTT

Por forma a tornar a comunicação com o Sistema de Aquisição possível, é utilizado um *broker* **MQTT**, ou seja, um intermediário que permite uma comunicação bidirecional entre dois ou mais dispositivos, através da subscrição de um ou mais tópicos.

O **MQTT**, acrónimo para *Message Queuing Telemetry Transport* é um protocolo de mensagens baseado na publicação/subscrição de tópicos via TCP/IP, desenvolvido para dispositivos com restrições ou redes de baixa largura de banda (*bandwidth*), alta latência e pouco seguros, ou seja, é o protocolo ideal para qualquer aplicação cuja comunicação se mostra difícil ou instável, sendo muito comum em aplicações IoT. Um *Broker* é o intermediário entre, no mínimo, dois dispositivos e baseia-se no protocolo, veja-se a Figura 5.12.

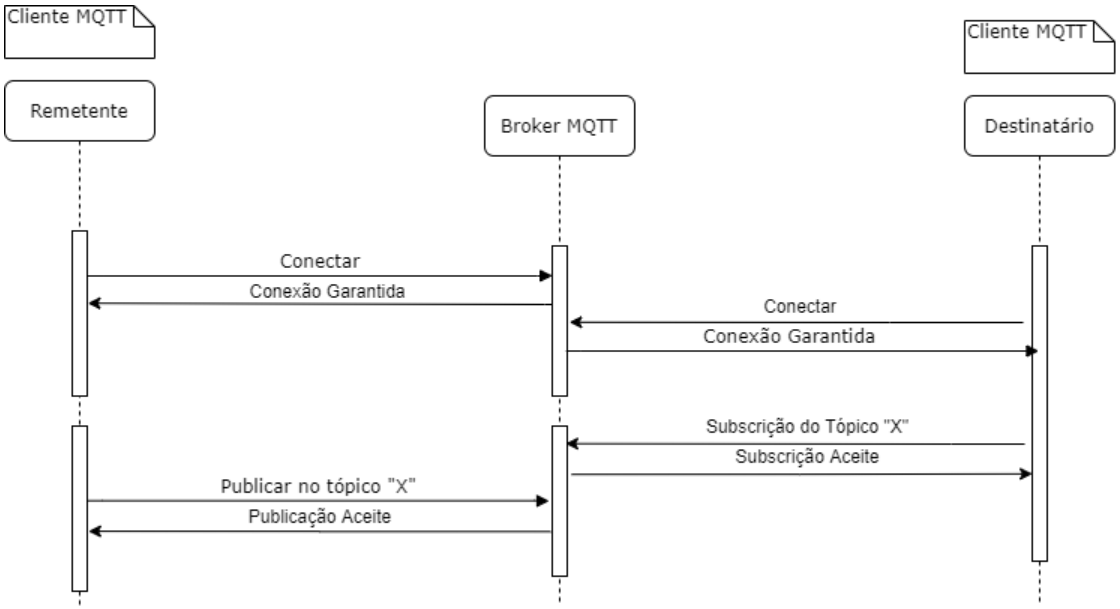


Figura 4.12: Diagrama de comunicação de um *broker* **MQTT**.

Com este protocolo é também possível definir o QoS (*Quality of Service*) para a troca de mensagens entre os clientes do intermediário (*broker* **MQTT**), ou seja, é possível definir o nível de garantia de entrega de uma determinada mensagem, havendo três níveis, como mostra a Tabela 4.2:

Tabela 4.2: Funções do *Quality of Service* .

QoS	**Cadência de Envio**	**Descrição**
0	No máximo uma vez	A mensagem é enviada apenas 1 vez e tanto o destinatário como o *broker* não confirmam a entrega.
1	No mínimo uma vez	A mensagem é enviada multiplas vezes até que haja confirmação de que a mensagem foi recebida.
2	Uma vez	O destinatário e o remetente executam um *handshake* para garantir que a mensagem chega ao destino uma só vez.

No caso deste Sistema, tanto o Sistema de Aquisição como o Servidor, são remetentes e destinatários no *broker* **MQTT**.

4.4.2 Node-RED

O Node-RED é uma ferramenta de programação visual, que pode ser acedida a partir de um *browser*, como se pode ver na Figura 4.13. Desenvolvida em Node.js, executa os fluxos ou *flows* em função de eventos. Foi concebida para implementação de aplicações IoT, sendo uma forma simples e rápida de teste. O Node.js pode ser designado como o ambiente de execução *Javascript* no lado do servidor (*back-end*), ao contrário de outros

softwares desenvolvidos em *Javascript* para o desenvolvimento de *websites* que têm como destino o lado do cliente (*client-side*). Através da instalação dos pacotes (*palletes* no Node-RED) adequados e recorrendo aos seus nós, facilmente se constrói um fluxo básico de informação, sem grandes bases de programação.

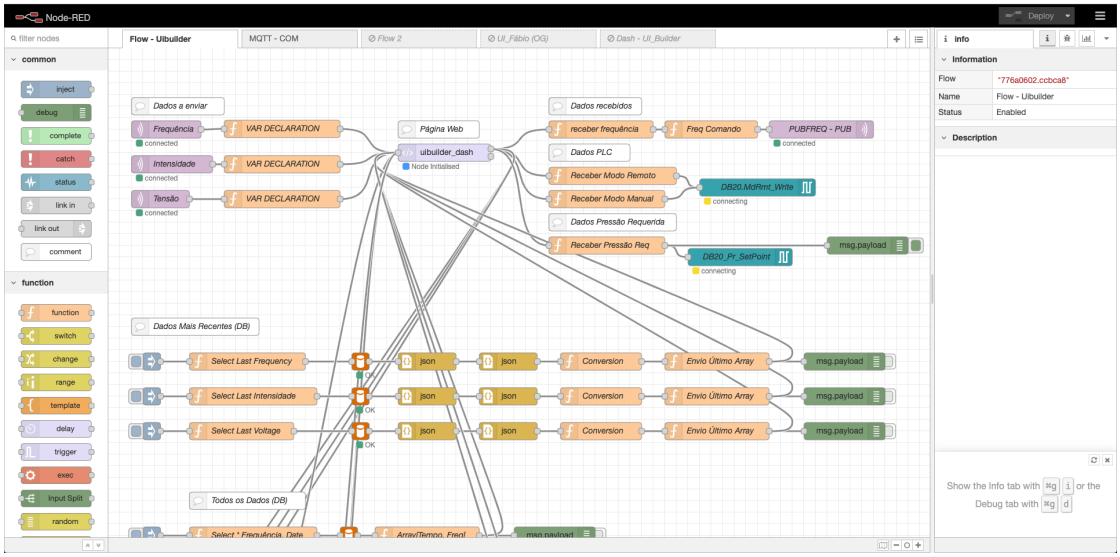


Figura 4.13: Programação visual usando o Node-RED.

4.4.3 Repositório de Dados - MySQL

O sistema de gestão de bases de dados MySQL é um sistema relacional, ou seja, dispõe os dados em tabelas (observe-se a coluna esquerda da Figura 4.14 a organização da informação em várias tabelas), as quais podem estar relacionados entre si, sendo essas relações responsáveis a estruturação dos dados. O MySQL permite o uso de credenciais para a segurança das bases de dados, sendo que o utilizador que queira aceder às tabelas e aos seus dados, terá que inserir essas credenciais, que podem ser:

Tabela 4.3: Credenciais MySQL .

Credenciais	**Descrição**	**Valor de Origem**
Username	Nome do utilizador	root
Password	Senha do utilizador	-

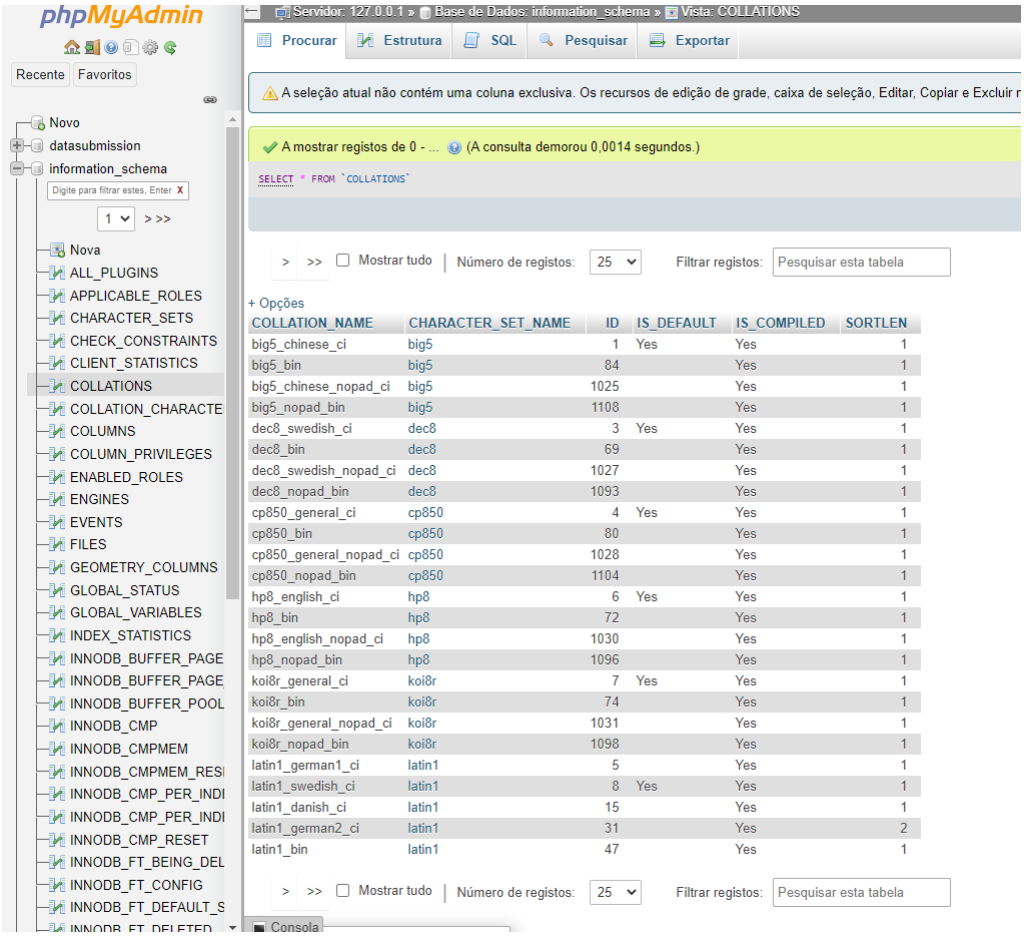


Figura 4.14: Exemplo da organização e estrutura de tabelas.

A linguagem SQL ou Linguagem de Consulta Estruturada é uma linguagem desenvolvida para criação, alteração e consulta de dados guardados em bases de dados relacionais. Esta linguagem usa *queries* específicas de acordo com a finalidade, como se pode ver nos três exemplos apresentados na Tabela 4.4. As *queries* aqui apresentadas são apenas exemplos, podendo conter mais conteúdo para refinar o seu objetivo, como por exemplo, para ordenar os dados numa determinada ordem.

Tabela 4.4: Exemplos de queries MySQL.

Função	**Query**
Adicionar Linha	INSERT INTO table (column1, column2) VALUES (value1, value2);
Alterar Valor	UPDATE table SET column1 = value1 WHERE condition';
Selecionar	SELECT column1, column2, ... FROM table;

4.5 Interface do Utilizador para Controlo e Supervisão

Para a criação de uma interface gráfica do utilizador, neste caso uma página *Web* são usados vários tipos de ficheiros que, servindo diferentes propósitos, contribuem para o mesmo fim. Por exemplo, para o caso do sistema desenvolvido um ficheiro **HTML** define a estrutura gráfica de uma página, um ficheiro **CSS** define detalhes visuais dos objetos da página, um ficheiro **Javascript** gere as interações entre os dados que chegam ou saiem da interface e o servidor.

4.5.1 Comunicação entre a Interface e Servidor Apache

Um servidor Apache é um servidor **HTTP** que recebe pedidos de clientes, normalmente *browsers* alojados em computadores e responde também com mensagens **HTTP**. Recorre ao protocolo **HTTP** para que um utilizador consiga remotamente aceder a documentos contidos nesse servidor, através da *internet*. A *internet* é um sistema global responsável pelo controlo da ligação entre computadores e servidores. O conjunto de protocolos **TCP/IP** usa outros protocolos para especificar como a informação é trocada. A parte **TCP** define como as aplicações podem criar canais de comunicação na rede, assim como a divisão da mensagem em blocos de informação de menor tamanho. A parte **IP** define o enderçamento e o trajeto desses blocos até chegarem ao destino correto [35]. Por exemplo, o protocolo o SMPT (*Simple Mail Transfer Protocol*) para envio e receção de *e-mails*, **FTP** (*File Transfer Protocol*) para a transferência de documentos e o **HTTP** para navegar. Segundo o Modelo OSI (*Open Systems Interconnection*) [36] o protocolo **TCP/IP** é uma estrutura conceptual utilizada para descrever as operações que ocorrem numa rede. As comunicações entre um sistema de computação são divididas em sete camadas diferentes: Física, Enlace, Rede, Transporte, Sessão, Apresentação e Aplicação, cada uma associada a um conjunto de protocolos específicos, veja-se a Tabela 4.5.

Tabela 4.5: Camadas do Modelo OSI do protocolo TCP/IP [36] .

Camada	**Protocolos Utilizados**
7 - Aplicação	HTTP, RTP, SMTP, FTP, SSH, Telnet, SIP, RDP, IRC, SNMP, NNTP, POP3, IMAP, DNS
6 - Apresentação	XDR, TLS
5 - Sessão	NetBIOS
4 - Transporte	NetBEUI, TCP, UDP, SCTP, DCCP, RIP
3 - Rede	IP (IPv4, IPv6), IPsec, ICMP, ARP, RARP, NAT
2 - Data Link	Ethernet, IEEE 802.1Q, HDLC, Token ring, FDDI, PPP, Switch, ATM
1 - Física	Modem, 802.11 Wi-Fi, RDIS, RS-485, RS-232, EIA-422, RS-449, Bluetooth, USB, 10BASE-T, 100BASE-TX, ISDN, SONET, DSL

4.5.2 Página *web*

Uma forma simples de construir uma interface consiste na criação de uma página *Web*. Esta permite que o utilizador aceda a essa interface em qualquer dispositivo desde que tenha acesso à *internet*. A imagem da Figura 4.15 é um exemplo de um *design* para uma página *web* SCADA do *software* "i4designer "da SPA [37].

Figura 4.15: Exemplo de *design* para uma página web para aplicações SCADA [37].

4.5.3 Registo de Eventos e Alarmes

Na Figura 4.16 pode-se ver um exemplo retirado deste projeto, no qual o Excel é utilizado para criar um registo de eventos e alarmes, assim como a data e hora em que estes foram acionados. Pretende-se criar um comando para a geração de um ficheiro texto (extensão **.txt**) que, sendo aberto com o Excel seguindo os passos da Secção A.6, permite a organização e estruturação dos vários registos por data e hora de acontecimento.

Modo Remoto ativado.	02/09/2020	11:23:24
O motor foi iniciado.	02/09/2020	11:24:12
Valor de frequência enviado.	02/09/2020	11:25:00
Frequência ultrapassou valor limite.	02/09/2020	11:26:20
O motor foi parado.	02/09/2020	11:30:02
Modo Automático ativado.	02/09/2020	11:35:24
Valor de tensão ultrapassou o limite.	02/09/2020	11:45:07

Figura 4.16: Exemplo de uma tabela de registo de alarmes em Excel.

Capítulo 5

Implementação

Neste capítulo é apresentada uma descrição mais pormenorizada da solução proposta para o Sistema de Monitorização e Controlo, estando esta dividida em duas secções: a primeira aborda a camada inferior do sistema, o Sistema de Aquisição de Dados e a parametrização do variador de frequência, a segunda referente à camada superior, tanto a interface como as interações entre sistema e o servidor.

Desta forma, é importante contextualizar o problema e referir o fluxo de informação real que ocorre nas instalações da AviSabor. A água entra continuamente nos reservatórios até o autómato dar a informação de nível máximo. Assim que este é atingido, as bombas param. A água vai entrando na unidade fabril conforme as necessidades, sendo que as máquinas controlam automaticamente o consumo de água e dando indicação para abrir ou fechar as válvulas para cortar o fluxo de água.

Para que se perceba melhor a dinâmica das comunicações do projeto desenvolvido, consulte-se o Diagrama de Sequência da Figura A.14 do Apêndice A.5, onde se encontram representados os vários passos desde o *input* de um valor de frequência na página *web* da Secção 5.4.2 até à alteração da frequência de comando da bomba.

5.1 Parametrização do Variador de Frequência

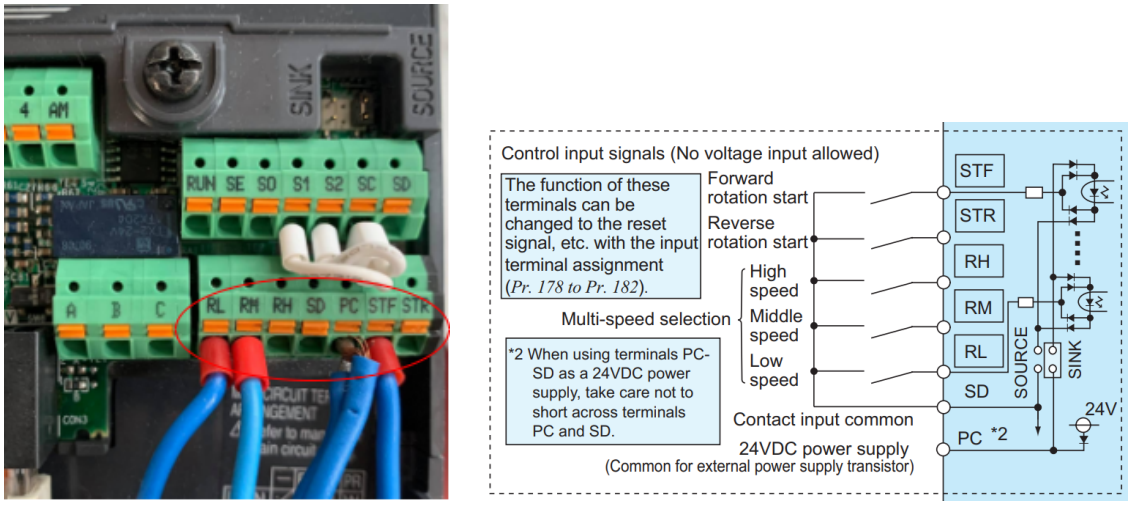
O variador FR-D720S da Mitsubishi está preparado para receber as mensagens seguindo a estrutura própria do protocolo Modbus RTU, sendo visto no barramento como um *Slave*. Após a consulta do *datasheet* para saber tanto os parâmetros de comunicação a configurar assim como um esquema das ligações necessárias para efetuar a comunicação podem ser consultados na Tabela 5.2 e na Figura A.6 do Apêndice A.

Tabela 5.1: Especificações Técnicas do variador Mitsubishi FR-D720S [38].

	Potência [kW]	**Alimentação (50/60 Hz) [V]**	**Frequência de Saída [Hz]**
Mitsubishi FR-D720S	0.1 - 2.2	Monofásica 200-240	0.2 - 400

5.1.1 Configuração dos Terminais

Neste projeto, pretende-se que exista a possibilidade de comunicação remota através da página *web* ou localmente, através do painel do variador de frequência. Como tal, os terminais do mesmo, mostrados na Figura 5.1 devem servir propósitos diferentes, que precisam de ser parametrizados usando o painel do variador. Na configuração utilizada, o terminal **PC** alimenta os restantes em função do objetivo pretendido: rotação horário ou anti-horária, tipo de comunicação, paragem, entre outros.



(a) (b)

Figura 5.1: (a) Terminais do variador, (b) Esquema da alimentação dos terminais do variador [39].

Veja-se a Tabela 5.2:

Tabela 5.2: Configuração dos terminais do variador.

Parâmetros	**Descrição**	**Valor**
79	Modo de Operação - Comutável	6
117	Número da Estação de Comunicação (*Slave*)	1
118	*Baud Rate* - Taxa de Transferência = 9600 bits/s	96
119	*Stop Bit* = 1	0
120	Paridade = Ímpar	1
123	Tempo de Espera	9999
124	Seleção CR/LF = Inexistente	0
178	Terminal STF - Rotação no sentido horário	60
179	Terminal STR - Rodação no sentido anti-horário	61
180	Terminal RL - Comuta entre controlo através do painel e externo	16
181	Terminal RM - Comuta entre controlo externo e remoto	66
549	Seleção do Protocolo - Modbus RTU	1

Os parâmetros intrínsecos ao protocolo de comunicação como: o endereço do (*Slave*), a taxa de transferência de informação (*baudrate*) e bit de paridade (*parity bit*), definiram-se seguindo a documentação do variador, conforme a Tabela 5.2 acima.

Após uma leitura cuidada da documentação, é possível construir a seguinte estrutura de mensagens, representada na Figura 5.2. Repare-se que o endereço do *Slave*, a *Function Code*, o *Number of Points Hi* e o *Number of Points Lo* são os mesmos para qualquer registo que se pretenda ler. Já parâmetros como (*Starting Adress Hi* e *Starting Adress Lo*) são únicos para cada registo, e por isso o *CRC* também será único.

Seguindo um exemplo do manual do variador, para ler o valor da frequência de comando de um equipamento com endereço de *slave* 1, deverá ser enviado a mensagem da Figura 5.2:

Endereço do Slave	Função	Endereço do Registo		Valor		CRC	
H01	H03	H00	H0D	H00	H01	H15	HC9

Figura 5.2: Exemplo de leitura de um registo.

A informação que este envia para o *Master* é codificada, para que seja possível enviar os valores com casas decimais. A codificação feita pelo variador necessita de uma posterior descodificação no ESP, para que o valor da variável a ler seja apresentada no formato decimal em vez do formato hexadecimal, observe-se o fluxograma da Figura 5.8.

5.2 Sistema de Aquisição de Dados (Frequência, Intensidade de Corrente, Tensão)

O Sistema de Aquisição de Dados tem como objetivo a recolha dos parâmetros contidos em certos registos internos (de frequência, intensidade de corrente e tensão) do variador de frequência, seguindo as normas do protocolo Modbus-RTU e ligando corretamente os terminais dos dois aparelhos, como ilustra a Figura 5.3. Utilizou-se um cabo *ethernet* descarnado para a comunicação entre o conector de três pinos da Figura A.2 do Apêndice A.1 e o conector PU da Figura A.1 do Apêndice A.1.

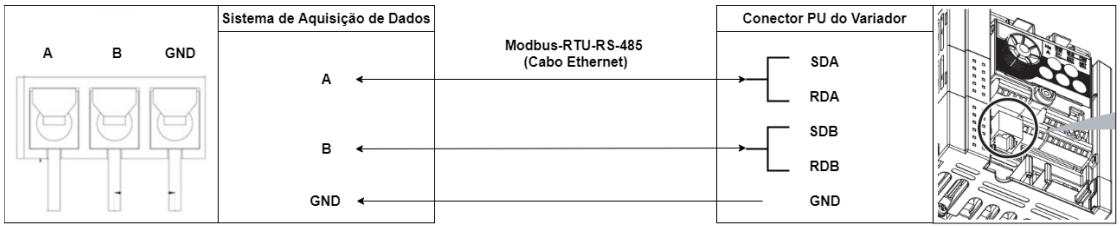


Figura 5.3: Ligação Modbus-RTU-RS-485 entre o conector do Sistema de Aquisição de Dados e o conector PU do variador de frequência.

Foi, inicialmente, desenvolvido numa placa de prototipagem de circuitos, a *breadboard*, como se pode pela ilustração da Figura 5.4 e nas imagens (a) e (b) da Figura A.21 do Apêndice A.7, passando de seguida para um protótipo numa Placa de Circuito Impresso (PCB), como se pode ver pelas Imagens (a) e (b) da Figura 5.10.

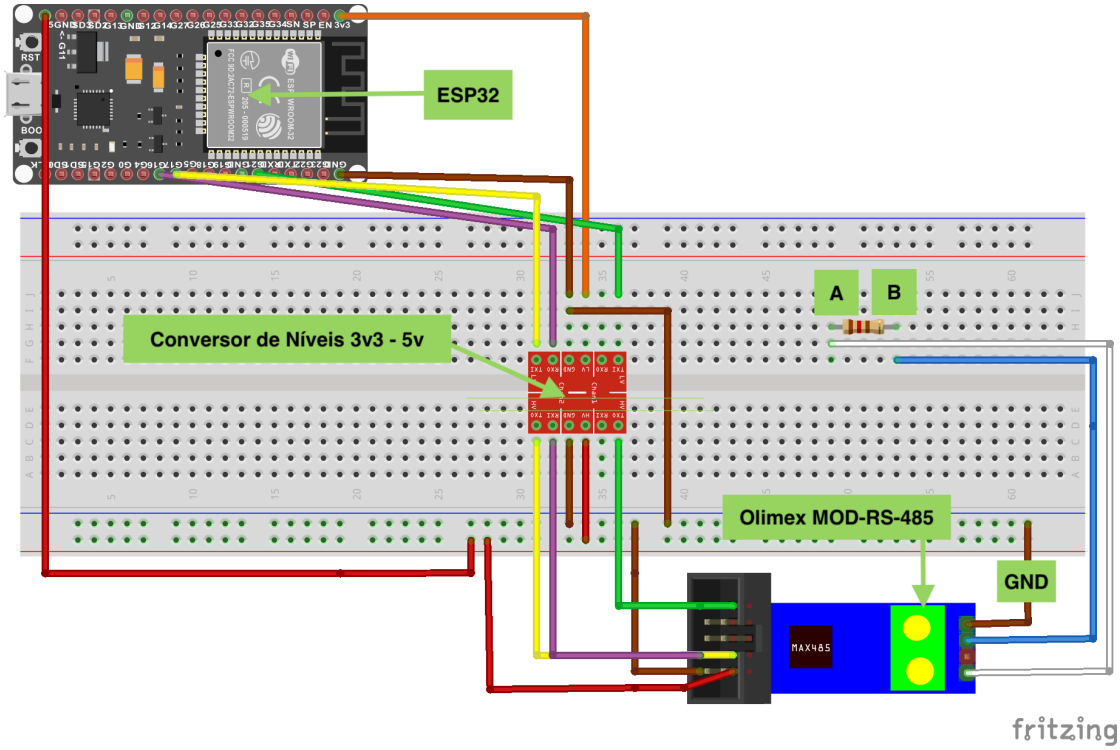


Figura 5.4: Ligações do Sistema de Aquisição de Dados na *breadboard*.

De seguida são apresentadas as implementações relativas ao Sistema de Aquisição de Dados, desde a comunicação com o variador e com o servidor, assim como o armazenamento de toda a informação útil. É também apresentado o microcontrolador escolhido para este projeto, com uma breve descrição das suas capacidades.

5.2.1 Módulo IoT - ESP32

O microcontrolador escolhido para o desenvolvimento deste projeto foi a placa NodeMCU-32S, baseada no módulo ESP-WROOM-32 da Expressif Systems. Na Figura 5.5 está representado o *pinout* dessa placa (referenciado na Secção 5.2.2 para descrever os pinos utilizados) e na Tabela 5.3 pode-se observar uma lista das especificações do ESP32 e o Arduino Uno R3, representados, respetivamente, na Imagem (a) e Imagem (b) da Figura A.22 do Apêndice A.8 com um *Ethernet Shield*, visto que este dispositivo era uma hipótese de implementação, dada a sua capacidade de comunicar através de uma ligação *ethernet*, que teria sido útil caso se quisesse um controlo maior sobre o autómato.

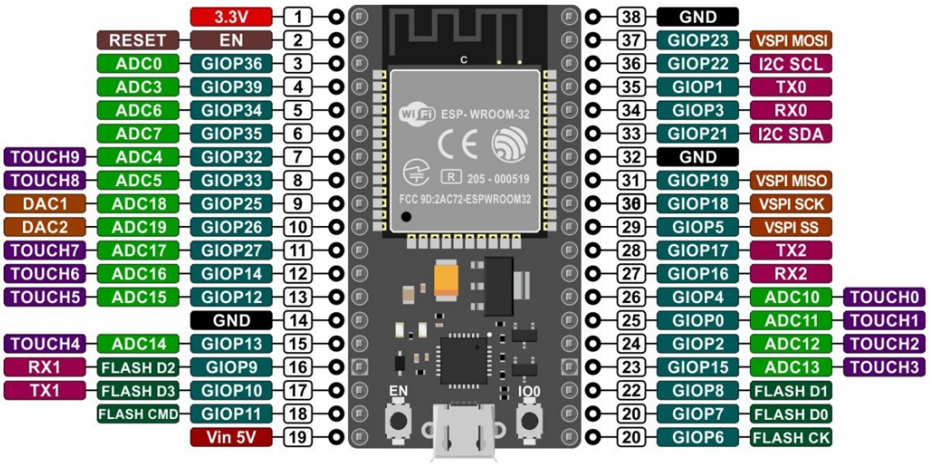


Figura 5.5: Disposição dos pinos do ESP32.

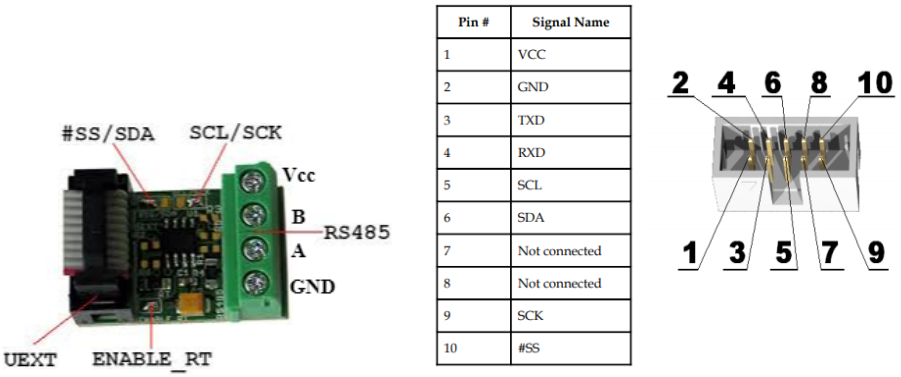
Tabela 5.3: Especificações do ESP32 e Arduino Uno + Ethernet Shield.

	ESP32	**Arduino UNO R3 + Ethernet Shield**
Cores (Núcleos)	2	1
Arquitetura	32 bits	8 bits
Clock (Frequência)	160 MHz (em cada núcleo)	16 MHz
Wi-Fi	Sim	Não
Ethernet	Não	Sim
Bluetooth	Em alguns casos	Não
RAM	512 KB	2KB
FLASH	16 MB	32 KB
GPIO	36 pinos	14 pinos
ADC	18 pinos	6 pinos
DAC	2 pinos	0 pinos
PWM	36 pinos	6 pinos
Interfaces	SPI/I2C/UART/I2S/CAN	SPI/I2C/UART
Com. Série	3 (Tx + Rx)	1 (Tx + Rx)
Tensão Pinos	3.3V	5V

5.2.2 Comunicação entre o Sistema de Aquisição de Dados (FIT) e o Variador de Frequência

O Sistema de Aquisição de Dados deverá ser capaz de ler e escrever os valores contidos nos registos internos do variador de frequência. Como mencionado na Secção 4.3.1.2, a tensão dos pinos do ESP32 é de apenas 3.3V. Assim, será necessário converter essa tensão com recurso a um conversor de níveis TTL (*Logic Level Shifter*), ou seja, de níveis lógicos (3.3V/0V para 5V/0V) para que seja possível interpretar as mensagens enviadas e

recebidas pelos equipamentos nos 5V de uma comunicação Modbus. O conversor Olimex MOD-RS485 representado na Imagem (a) da Figura 5.6 é utilizado para regular a tensão dos sinais **DE** (*Drive Output Enable*) e /**RE** (*Receiver Enable*), caso se queira enviar ou receber mensagens série para/do variador. Visto que o conversor controla a tensão desses sinais através do ajuste da tensão dos pinos #**SS** e **SCK** do conector, deve-se ajustar a tensão do sinal **DE** para 5V, ou seja, dos pino #**SS** e **SCK** para 5V (**preTransmission** da Apêndice A.9) e para 0v para o dispositivo ficar à "escuta" de mensagens de resposta (**postTransmission** da Apêndice A.9). Os pinos do conector estão representados na Imagem (b) da Figura 5.6.



(a) (b)

Figura 5.6: (a) Conversor de níveis TTL para RS-485, (b) Disposição dos pinos do conector do Olimex MOD-RS-485 (UEXT).

Veja-se a Tabela 5.4 com a descrição da função dos pinos da Imagem (b) da Figura 5.6 utilizados para este projeto:

Tabela 5.4: Designação e descrição dos pinos do conector.

Pino nº	**Descrição**
1 - V_{cc}	Alimentação da placa.
2 - Gnd	Ligação à malha de referência terra.
3 - Tx	Ligação ao **GPIO** Tx do microcontrolador.
4 - Rx	Ligação ao **GPIO** Rx do microcontrolador.
9 - SCK	Emissão ou receção de mensagens consoante o ajuste de tensão.
10 - SS	Emissão ou receção de mensagens consoante o ajuste de tensão.

Utilizou-se apenas uma resistência de terminação prevista pelo protocolo entre os terminais A e B do equipamento com o valor de 120Ω indicados pelo manual do variador. Na programação do microcontrolador é usado o comando *pinMode* que indica que o pino escolhido para a leitura e escrita funcionará como ***output***.

5.2.2.1 Modbus - RTU

Na leitura e escrita dos valores de frequência, tensão e intensidade de corrente, o Sistema de Aquisição de Dados seguirá a norma RS-485 do protocolo Modbus-RTU. Para o efeito utilizou-se um conversor de níveis lógicos TTL para níveis de tensões coincidentes com as da norma utilizada, um conversor Olimex MOD-RS485. Como os dispositivos que comunicam segundo este padrão têm níveis de tensão na ordem dos 5V, será necessário que o conversor seja alimentado com 5V e a sua comunicação de níveis lógicos TTL seja igualmente de 5V, ou seja, será necessário aplicar um conversor de tensões entre 3.3V e 5V para que o microcontrolador consiga comunicar com o variador sem erros.

Observe-se na Figura 5.4 as ligações entre os três dispositivos encontram-se representadas, sendo necessário utilizar duas bibliotecas na programação do microcontrolador: **ModbusMaster.h**[1] e **HardwareSerial.h** para a comunicação.

Para o uso da biblioteca **ModbusMaster.h** deve-se indicar o nome do objeto a ser usado, utilizando o comando **ModbusMaster**.É com este objeto que se indica à biblioteca o ID do *slave* e quais os pinos utilizados para a comunicação (**GPIO16** e **GPIO17** da Figura 5.5), bem como a taxa de transmissão de informação (*baudrate*). Usando os seguintes comandos e funções: abertura das portas série para comunicar (**Serial.begin()**), as funções para serem processadas antes e depois de uma transmissão de dados (**preTransmission** e **postTransmission**) por forma a a regular a tensão do pino de ativação da recepção e emissão (**GPIO21** da Figura 5.5) dá-se início à comunicação. O fluxograma da Figura 5.7 mostra a sequência de processos para escrever num registo do variador.

[1] As expressões a negrito fazem referência a bibliotecas, funções, comandos e tipos de variáveis do código que se encontra na Secção A.9.

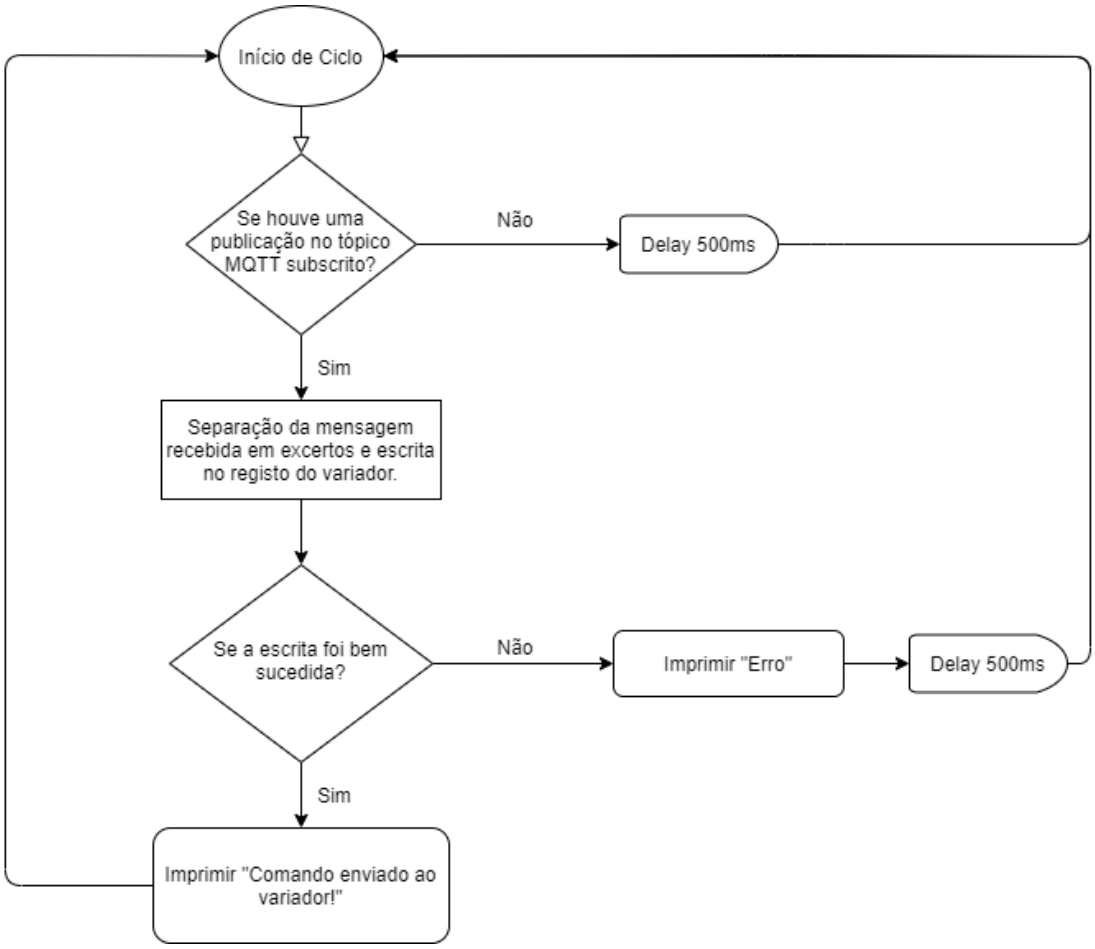


Figura 5.7: Fluxograma de aquisição de dados e escrita nos registos do variador.

Com a biblioteca **ModbusMaster.h**, para que o microcontrolador consiga escrever num registo de memória do variador usa-se o comando de escrita num registo - **writeSingleRegister** - onde se indica o endereço hexadecimal do registo. Este comando retorna um valor do tipo **uint8_t**, que no caso da comunicação ter sido bem sucedida, será igual a **ku8MBSuccess** e, consequentemente, alterará a frequência de comando do variador. Para a leitura de registos, utilizou-se a biblioteca **HardwareSerial.h**, que funciona da seguinte forma: para configurar o objeto e o tipo de comunicação, o procedimento é idêntico ao supramencionado para a biblioteca anterior. Utiliza-se o comando de **write()**, no qual se especifica um *array* com a mensagem série de leitura de um registo e cuja estrutura se rege pela do protocolo. Depois de enviada, o ESP32 deverá receber uma mensagem série proveniente do variador com a mesma estrutura, com a informação pretendida nos *bits* de dados do *frame* de resposta da Figura 4.7. Através do comando (**read()**), o algoritmo lê ciclicamente *byte* a *byte* o conteúdo da mensagem, e armazena-a num *buffer* temporário. A informação útil encontra-se na terceira e quarta posição desse *buffer* e após conversão do formato hexadecimal para *float*, o resultado pretendido é devolvido em formato decimal:

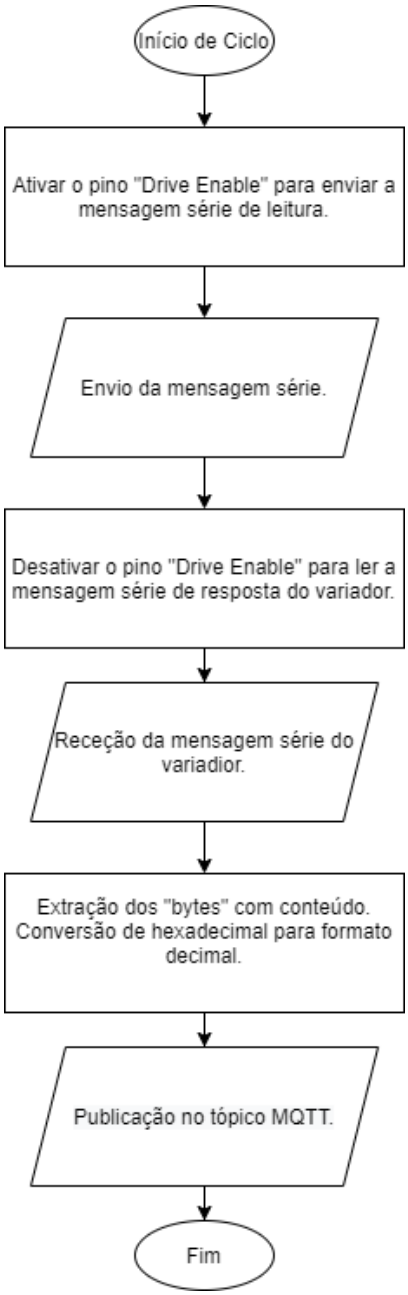


Figura 5.8: Fluxograma correspondente à leitura de registos do variador.

Para que a comunicação seja estabelecida, é necessário um par de fios entrelaçados que liguem os terminais **A** e **B** do conversor.

5.2.3 Comunicação entre o Sistema de Aquisição de Dados e o Servidor (Raspberry Pi)

5.2.3.1 WiFi & MQTT

O microcontrolador ESP32 tem uma placa Wi-Fi integrada, como se pode ver na Imagem (a) da Figura A.22 do Apêndice A.8, permitindo o desenvolvimento de projetos no âmbito da IoT, faltando ao utilizador configurar os parâmetros da rede à qual se tenciona ligar, (SSID *(Service Set Identifier)* e *Password*). Para tal, basta incluir a biblioteca **Wifi.h**.

Para a transmissão de dados, o Sistema de Aquisição de Dados (FIT) subscreve um tópico do *broker* **MQTT** para trocar mensagens e por isso torna-se necessário incluir a biblioteca **PubSubClient.h**. Esta biblioteca requer que se defina o tipo de comunicação criando um cliente se ligue a um IP específico (**WiFiClient** - linha 2) e um objeto do tipo **PubSubClient** - linha 3 para estabelecer comunicação com um *broker* **MQTT** com o cliente WiFi definido antes (**WiFiClient**), o IP do *broker* e a porta (linha 1 e 3).

Por forma a entender esta secção, segue-se um excerto do código utilizado na programação do microcontrolador:

```
1 const char* mqtt_server = "192.168.0.102";
2 WiFiClient espClient;
3 PubSubClient client(mqtt_server, 1883, callback, espClient);
```

onde a *callback* é uma função chamada sempre que se pretende publicar conteúdo em algum tópico do *broker*. Os tópicos do *broker* são subscritos utilizando o comando **subscribe** e nestes pode ser publicado conteúdo recorrendo ao comando **publish**, bastando definir o tópico e a mensagem.

5.2.4 Desenvolvimento da placa PCB

Na Figura 5.9 encontra-se representado o modelo tridimensional desenvolvido numa placa PCB.

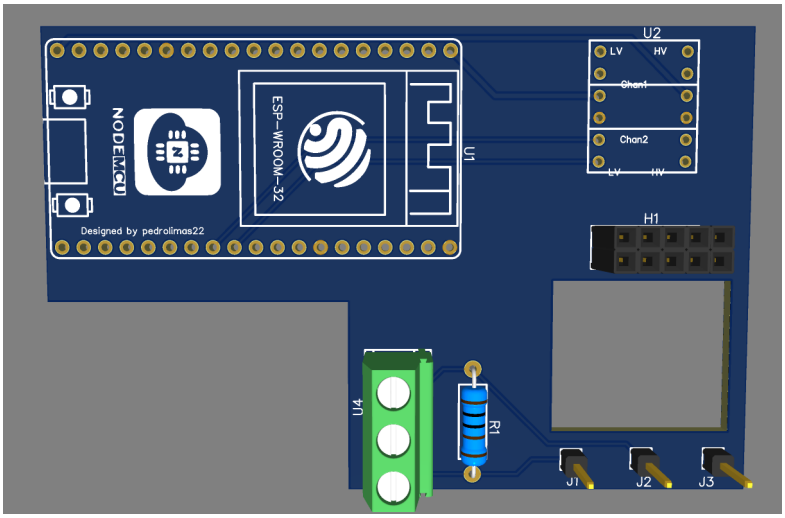
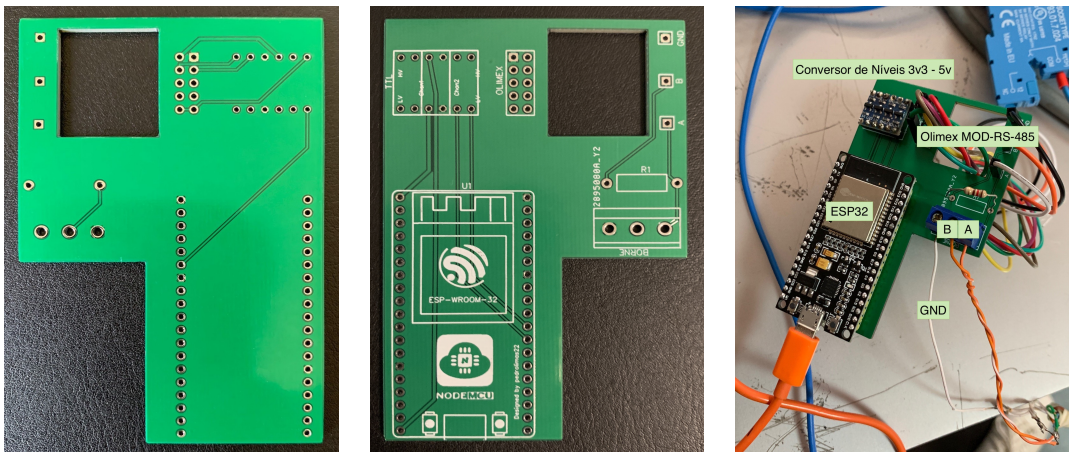


Figura 5.9: Modelo 3D do protótipo desenvolvido em placa **PCB**.

Para passar de uma placa *breadboard* para uma placa **PCB** é necessário desenhar as camadas que compõem a mesma, tornando-se para isso necessário a utilização de um *software* próprio. Para este projeto utilizou-se o **EasyEDA**, que contém várias bibliotecas de componentes e uma função de *auto-routing* que faz automaticamente o desenho do traçado de cobre na placa, seguindo o esquema elétrico que lhe serve de base. As imagens da Figura 5.10 seguintes mostram a **PCB** com e sem componentes.



(a) (b) (c)

Figura 5.10: PCB desenvolvida para o projeto: (a) Vista traseira da **PCB**, (b) Vista frontal da **PCB**, (c) Vista da **PCB** com os componentes.

5.3 Servidor (Raspberry Pi)

Nesta secção aborda-se detalhadamente as interações que ocorrem entre o servidor e o restante sistema, por forma a entender o seu comportamento.

5.3.1 Módulo Raspberry Pi 3B

O equipamento utilizado para simular o comportamento de um servidor foi um Raspberry Pi 3 Model B.

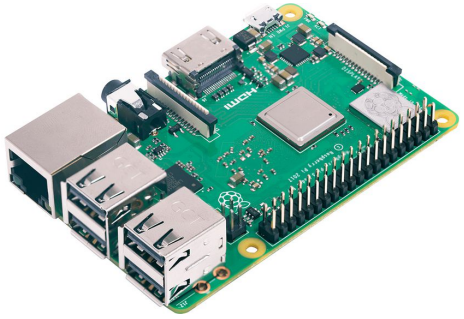


Figura 5.11: Raspberry Pi Model 3B+.

A principal função deste equipamento é oferecer uma alternativa prática de baixo custo para testar aplicações que não exijam elevado poder computacional. Apesar das pequenas dimensões e do seu aspeto pouco convencional, o Raspberry permite efetuar praticamente todas as tarefas que se realizam noutros computadores, como por exemplo navegar na Internet, criar textos ou reproduzir conteúdo multimédia. O Raspberry Pi Model 3 B apresenta especificações que o tornam numa escolha adequada, p ser consultadas na Tabela 5.5 [40]:

Tabela 5.5: Especificações do modelo do Raspberry Pi.

Componentes	**Especificações**
Processador	Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
Memória RAM	1GB LPDDR2 SDRAM
Conectividade *wireless*	2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
Conectividade c/ fios	Gigabit Ethernet over USB 2.0
Entradas	4 x Portas USB 2.0, 1 x HDMI
Alimentação	5.1V / 2.5A DC

5.3.2 Comunicação entre Servidor e Sistema de Aquisição de Dados

Já foi mencionado anteriormente que o objetivo do servidor consiste na comunicação bidirecional com o Sistema de Aquisição de Dados (FIT) da **Secção 5.3**. Este deve ser capaz de receber os valores enviados pelo variador, assim como de disponibilizar ao utilizador dados mais específicos de interesse para o mesmo, por exemplo, gráficos com todos os valores armazenados de qualquer registo, guardar os alarmes mais importantes num formato **CSV** ou **XML** compatíveis com Excel (procedimento explicado com maior detalhe na Secção A.6, do Apêncide A), o estado do variador, entre outros.

O *broker* utilizado para o desenvolvimento deste projeto foi o **Mosquitto** em que a ligação ao mesmo é feita através do Node-Red. Na Figura 5.12 pode-se ver um diagrama de sequência com o fluxo de informação e na Figura 5.13 o código em Node-RED necessário para subscrever e/ou publicar em tópicos **MQTT**. Na figura 5.12 exemplo pode-se observar a subscrição do tópico "Frequência"e a publicação do valor do *payload* no tópico "Frequência", sendo esta publicação concretizada quando o utilizador efetuar o *input* da frequência de comando.

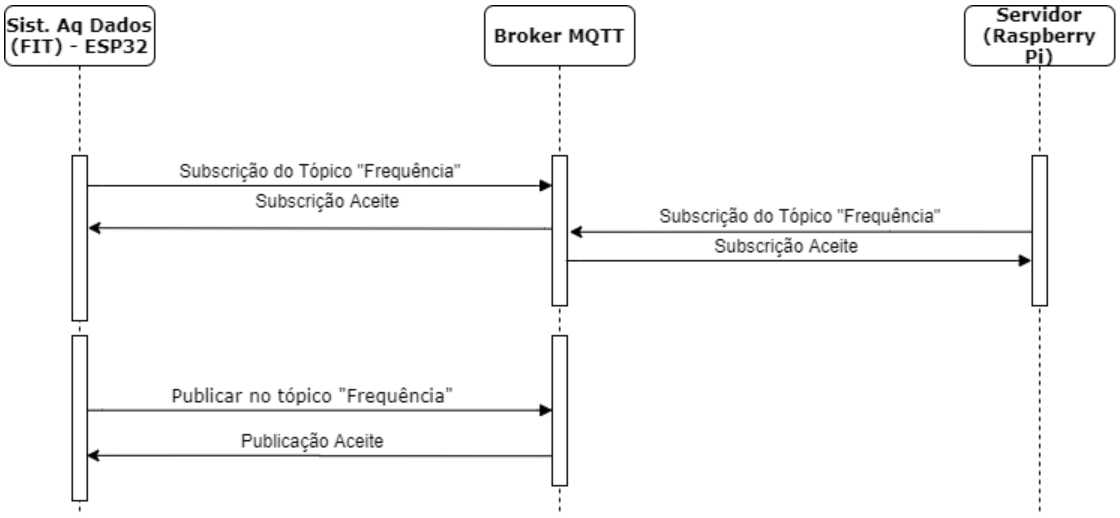


Figura 5.12: Diagrama de comunicação entre o Sistema de Aquisição de Dados, o Servidor e o *broker* MQTT.

5.3.3 Repositório de Dados (Adquiridos pelo Sistema de Aquisição e Limites Selecionados)

Para a concretização deste projeto, optou-se por uma base de dados MySQL.

Para instalar a base de dados MySQL no módulo do servidor, deve-se instalar o Apache pela linha de comandos, que criará um servidor **HTTP**. Todos os valores relevantes são enviados para a base através de nós MySQL do Node-RED.

Interpretando a Figura 5.13 de cima para baixo, os nós **MQTT** a roxo estão à escuta de mensagens publicadas nos tópicos que subscrevem: Frequência, Intensidade (de Corrente) e Tensão. O conteúdo dessas mensagens é convertido numa variável local (com o comando **move**) para serem utilizadas por outros nós desse *flow* (deixando de haver necessidade de ligar outros nós a esse para utilizar o seu conteúdo). O nó ***timestamp*** indica um intervalo temporal de ativação da sequência de nós que o seguem. O nó

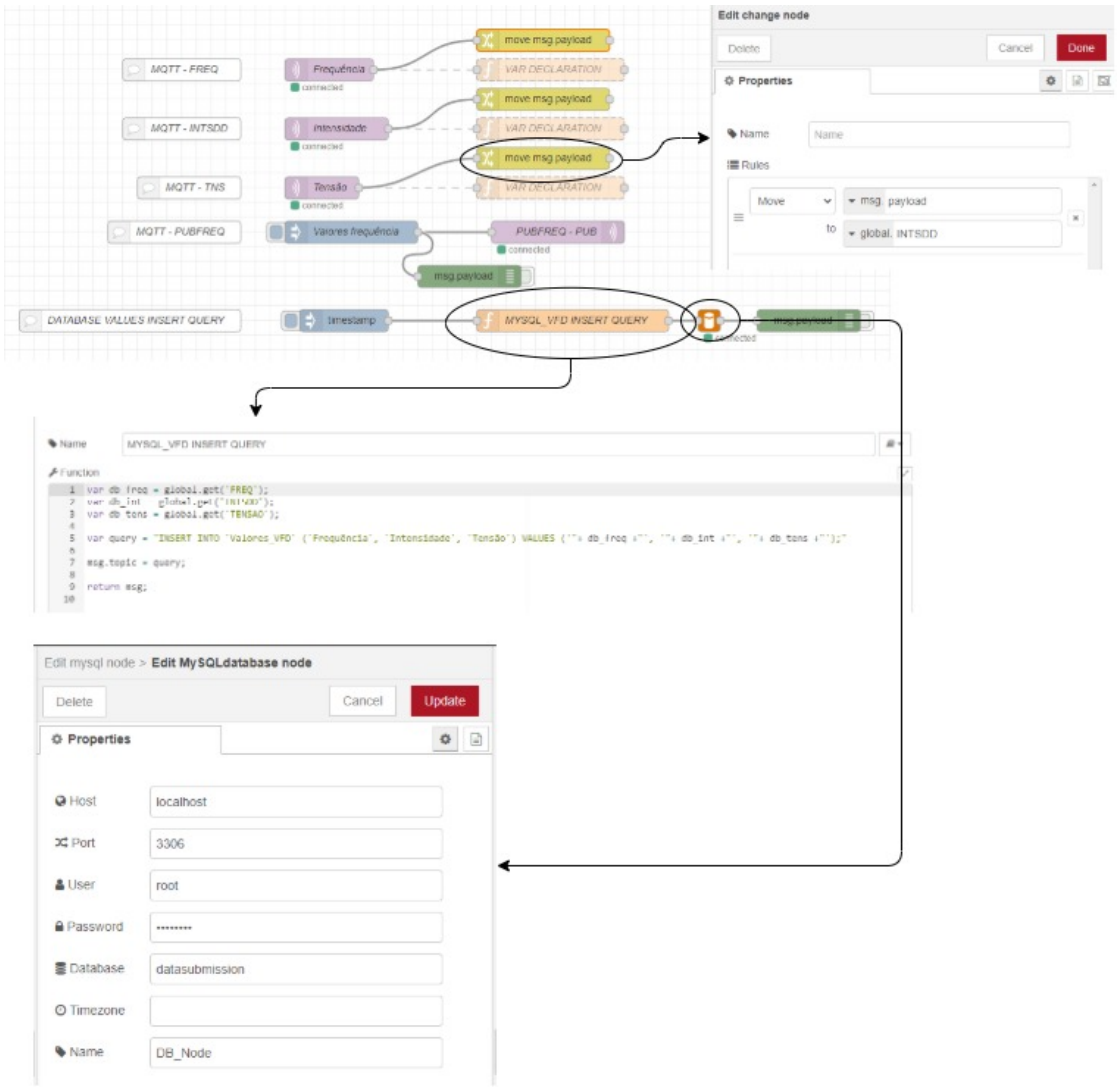


Figura 5.13: Código Node-RED para inserir valores na base de dados MySQL.

function adquire os valores globais supramencionados e seguindo uma **INSERT** *query* (Tabela 4.4), são inseridos os valores recolhidos numa tabela (nas colunas "Frequência", "Intensidade" e "Tensão" da tabela "Valores_VFD") configurada para os receber com o nó **MySQL**.

Este repositório foi criado para que os valores recolhidos pelo Sistema de Aquisição de Dados (FIT) da **Secção 5.2** e limites escolhidos pelo utilizador na Interface Gráfica da **Secção 5.4**, por forma a haver um registo estruturado de toda a informação recolhida na globalidade do sistema. Os dados armazenados são discutidos com maior detalhe na **Secção 6.1.2.**

5.4 Interface Gráfica de Supervisão e Controlo do Variador de Frequência (Página *web* SCADA)

Esta secção aborda a parte gráfica responsável por qualquer interação entre o utilizador e o sistema.

5.4.1 Comunicação entre Servidor e Interface

A interface foi desenvolvida para permitir flexibilidade na supervisão dos equipamentos, com o intuito de ser acedida em qualquer parte e a partir de qualquer equipamento com ligação à *internet*. Para que o servidor e a interface comuniquem corretamente entre si e os dados possam ser acedidos e amostrados, é necessário que o Apache e o Node-RED corram sempre que o módulo do servidor se encontre ligado. É necessário configurar estes *softwares* para iniciarem automaticamente com o ligar do módulo.

5.4.2 Página *web* SCADA

O desenvolvimento da página assenta na programação de ficheiros **HTML** e **CSS**, para a estrutura gráfica e disposição de objetos da mesma. Os eventos intrínsecos à página são moderados através de um ficheiro **JavaScript** que, em conjunto com os nós do Node-RED, acede à base de dados para retirar ou colocar dados e mostrar informação útil na página. Os ficheiros de criação estão alojados e divididos no nó *UIbuilder*, dando oportunidade e flexibilidade a eventuais alterações futuras.

5.4.2.1 Modo Manual e Modo Remoto de Controlo

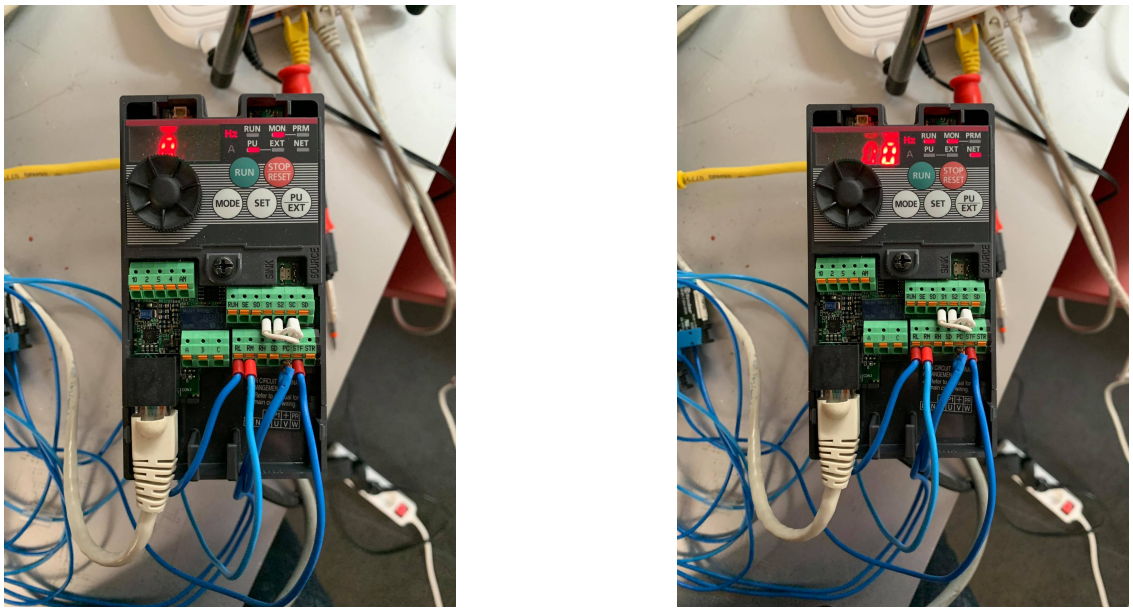
No separador "Controlo" existe a opção de escolher entre definir a frequência de comando através do painel do variador, ativando o Modo Manual, ou remotamente através da página *web*, ativando o Modo Remoto, como mostra a Figura 5.14.

Assim como o Modo Manual é utilizado para controlar os parâmetros do variador (frequência de comando, frequência limite, aceleração, alarmes, entre outros) através do painel embutido, o Modo Remoto permite que o parâmetros da frequência de controlo seja controlada externamente através da ligação RS-485 com o módulo IoT (ESP32).

A Imagem (a) da Figura 5.15 representa o variador no Modo Manual em que o LED **PU** está aceso. A Imagem (b) da mesma Figura representa o variador em Modo Remoto em que o LED **NET** está aceso.

Estes dois modos são alcançados através da parametrização do variador (já mencionada na Secção 5.1) e os estados são ativados consoante a energização dos terminais configurados para o efeito (veja-se a Tabela 5.2).

Figura 5.14: Caixa de controlo dos modos de funcionamento do variador de frequência.



(a) Variador em Modo Manual (b) Variador em Modo Remoto

Figura 5.15: Estados de funcionamento do varidor.

5.4.2.2 Gráficos de Frequência, Intensidade de Corrente e Tensão

Um dos principais objetivos da página *web* é a visualização simples e intuitiva dos valores recolhidos em tempo real. Para tal, a solução que melhor se adequava seria a presença de um medidor (***gauge*** na Figura 6.5) do Capítulo 5 acompanhado de um gráfico de monitorização em tempo real, que mostra o valor adquirido e a data/hora de aquisição, como mostra a Figura 5.16, onde está representada uma imagem dos gráficos de Frequência e Tensão em execução em tempo real:

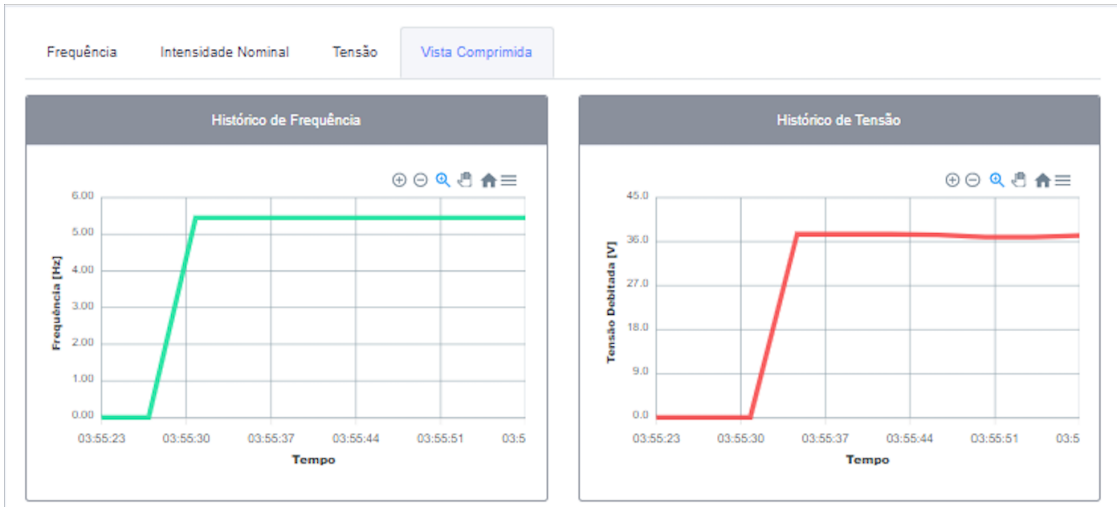


Figura 5.16: Disposição dos gráficos de Frequência e Tensão (em função do tempo) no separador "Vista Comprimida" na página *web*.

5.4.2.3 Alarmes de Limite Atingido

A interface criada faz um registo dos eventos mais importantes que nela ocorrem. Esses eventos são registados no separador "Alarmes" e a página oferece a possibilidade de descarregar o ficheiro com a extensão **.txt**. É igualmente responsável por alertar por *e-mail* os utilizadores da mesma, caso se ultrapassem os limites definidos.

As opções de controlo que a interface oferece são registadas no separador, e os alarmes que surgem são acompanhados de um *popup*, como na Figura 5.18 que implica que o utilizador carregue no botão **"OK"** para poder prosseguir, obrigando desta forma ao conhecimento do alarme.

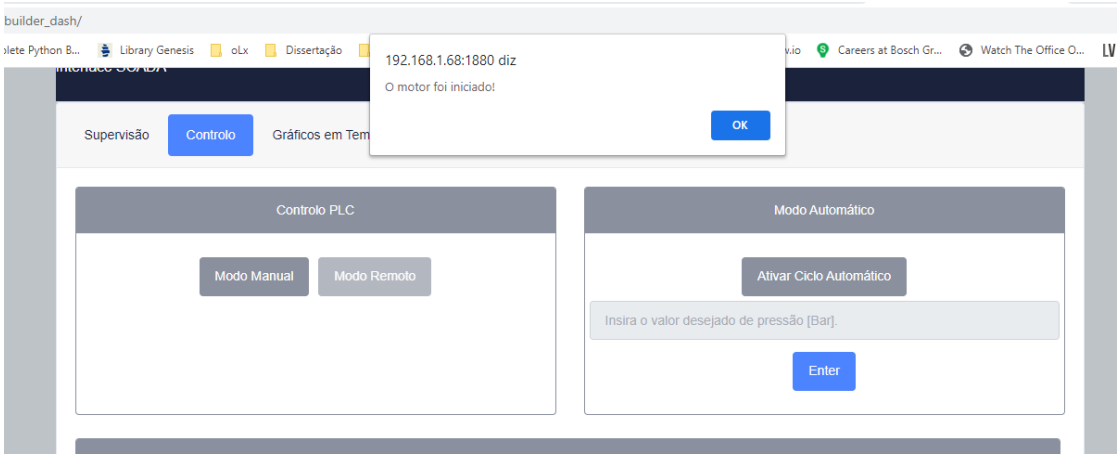


Figura 5.17: Exemplo de alarme acompanho por um ***popup***.

A interface está encarregue da gestão dos limites dos valores que disparam os alarmes, e, por meio do nó *email* do Node-RED enviam por *e-mail* os alertas de que os limites foram ultrapassados, de acordo com a Figura seguinte.

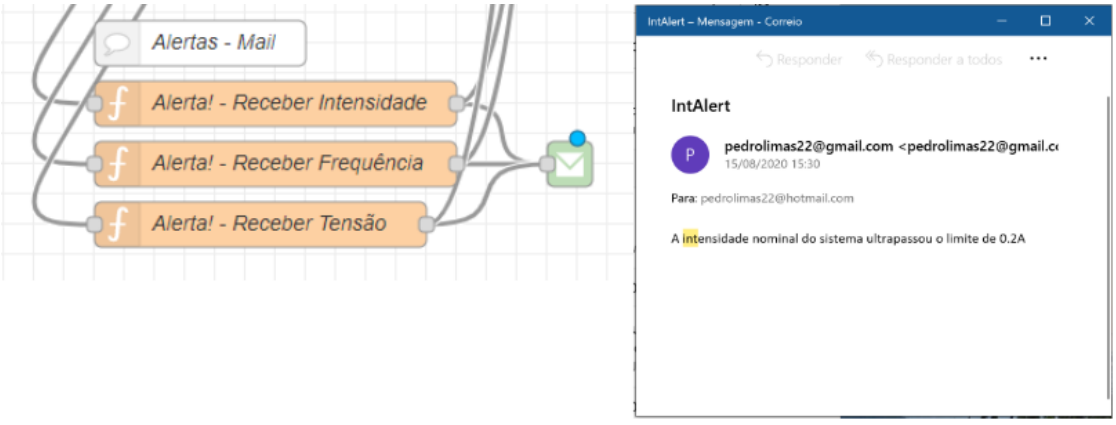


Figura 5.18: *Flows* do Node-RED e o exemplo de um alarme recebido via *e-mail*.

5.5 Modo Automático de Regulação de Frequência

Com o objetivo de mostrar a aplicabilidade prática deste projeto, desenvolveu-se um sistema automático de controlo e monitorização dos valores de frequência. Para o efeito, utilizou-se um módulo ADAM de aquisição de entradas analógicas e saídas digitais (**módulo AI/DO**) ADAM 6017, representado na Imagem (a) da Figura 5.20, para aquisição de sinais externos. A Figura 5.19 ilustra a arquitetura do funcionamento deste modo segundo o estilo da norma ISA95 [41].

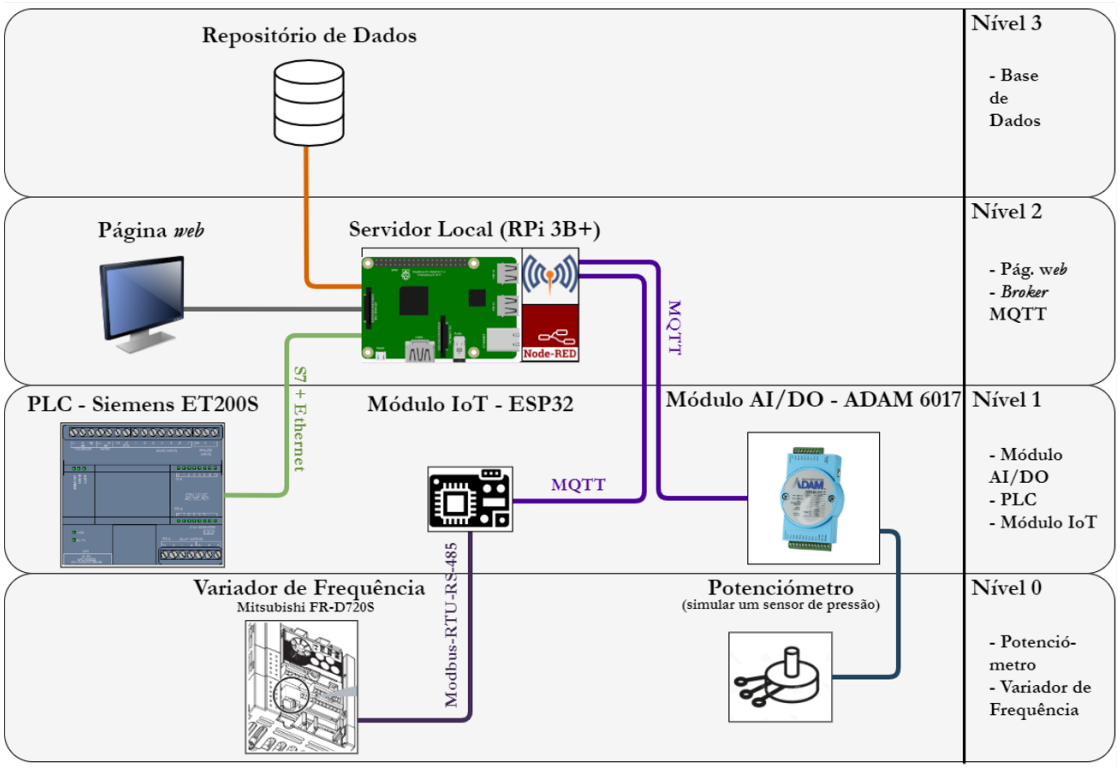
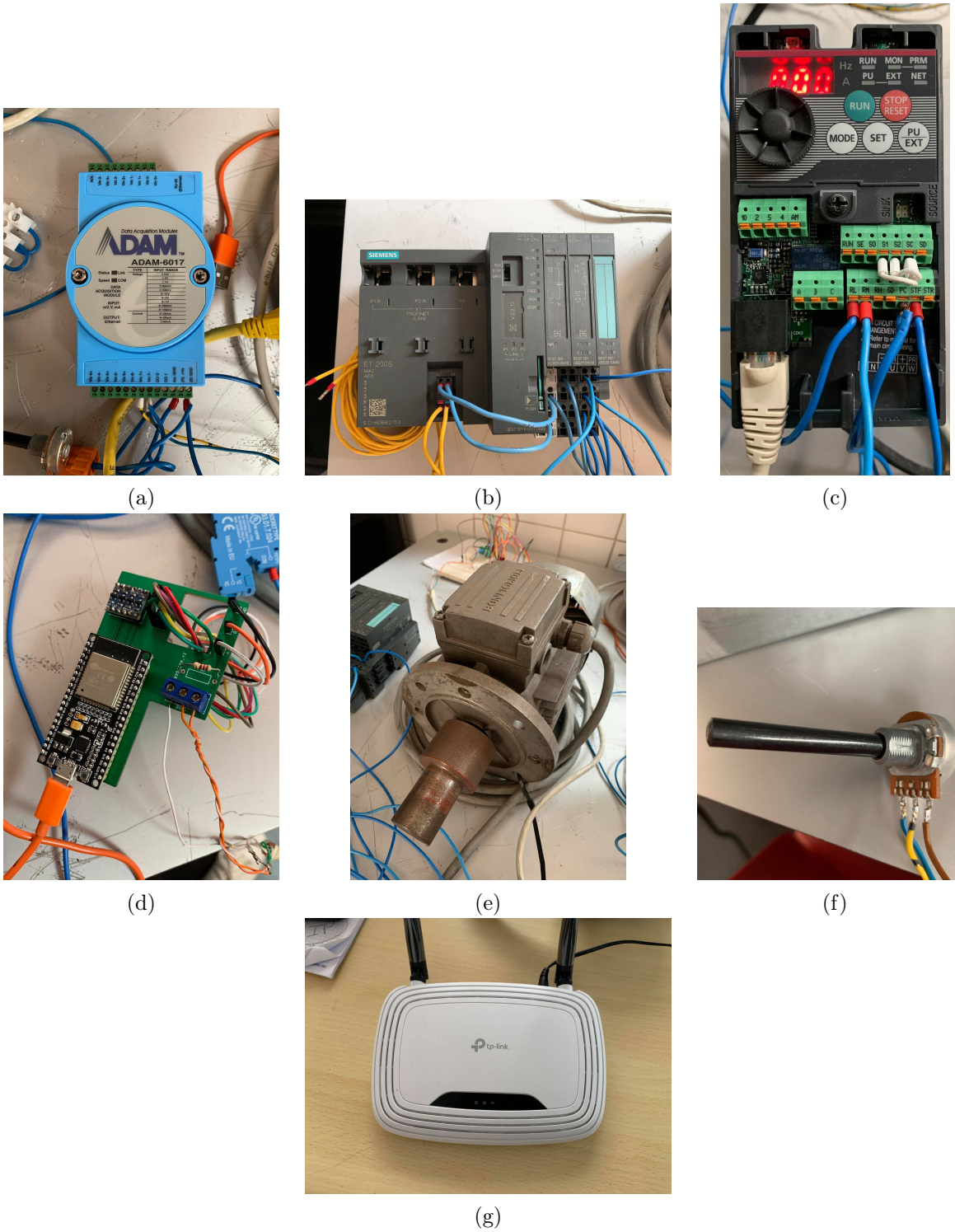


Figura 5.19: Arquitetura do Modo Automático de Regulação de Frequência segundo a norma ISA95 [41].

Para além do módulo de aquisição AI/DO (Imagem (a)), são necessários outros dispositivos, representados na Figura 5.20 para a realização da simulação:

- PLC Siemens ET200S - Imagem (b);
- Variador de Frequência Mitsubishi D720S - Imagem(c);
- Sistema de Aquisição de Dados- Imagem (d);
- Potenciómetro - Imagem (f) (para simular um sensor de pressão analógico) ;
- Motor Elétrico Trifásico - Imagem (e);
- *Router* TP-LINK WR-841N - Imagem (g).



(a) (b) (c)

(d) (e) (f)

(g)

Figura 5.20: *Hardware* do Modo Automático: (a) ADAM 6017, (b) PLC Siemens ET200s, (c) Variador de Frequência Mitsubishi D720S, (d) Sistema de Aquisição de Dados, (e) Motor Trifásico, (f) Potenciómetro, (g) *Router* TP-LINK.

5.5.1 Funcionamento

Nesta subsecção explicar-se-á o funcionamento do modo automático desenvolvido. Começando pela camada mais baixa, a aquisição do valor analógico debitado pelo potenciómetro, é utilizado para simular um sensor de pressão à saída de uma bomba. Ligando o potenciómetro ao módulo e regulando o mesmo, a resistência altera o valor da tensão adquirida, que é então processada pelo módulo de aquisição de entradas analógicas. O módulo de aquisição possui ainda a vantagem de poder comunicar via **MQTT**, facilitando assim o envio e receção de valores através do Node-RED.

5.5.2 Comunicação do Módulo de Aquisição de Entradas Analógicas (Módulo AI/DO)

A nível industrial, o controlo de aparelhos como variadores de frequência faz-se, geralmente, através da programação de autómatos, que são equipamentos dedicados a este tipo de aplicações. Desenvolveu-se um código em Node-RED e um programa *ladder* no TIA Portal numa tentativa de reproduzir as interações que acontecem num sistema real desde a aquisição de sinais até à reação do sistema face a esses ***inputs***.

A comunicação com este módulo é feita através da sua porta *ethernet* e necessita de uma configuração prévia que é feita no *software*: ***AdamApax.NET Utility***. O *software* confere flexibilidade ao módulo, no sentido em que permite que este comunique segundo o protocolo Modbus ou através de uma ligação **MQTT**, disponibilizando, ainda, a possibilidade de consultar o estado das suas entradas analógicas, assim como o das suas saídas digitais através de uma página *web*, representada nas Figuras 5.21 (a curva ascendente a rosa representa o valor da tensão em função do tempo da **AI-7**) e 5.22 (onde se podem consultar os valores analógicos exatos) que é acedida inserindo no *browser* o IP do módulo.

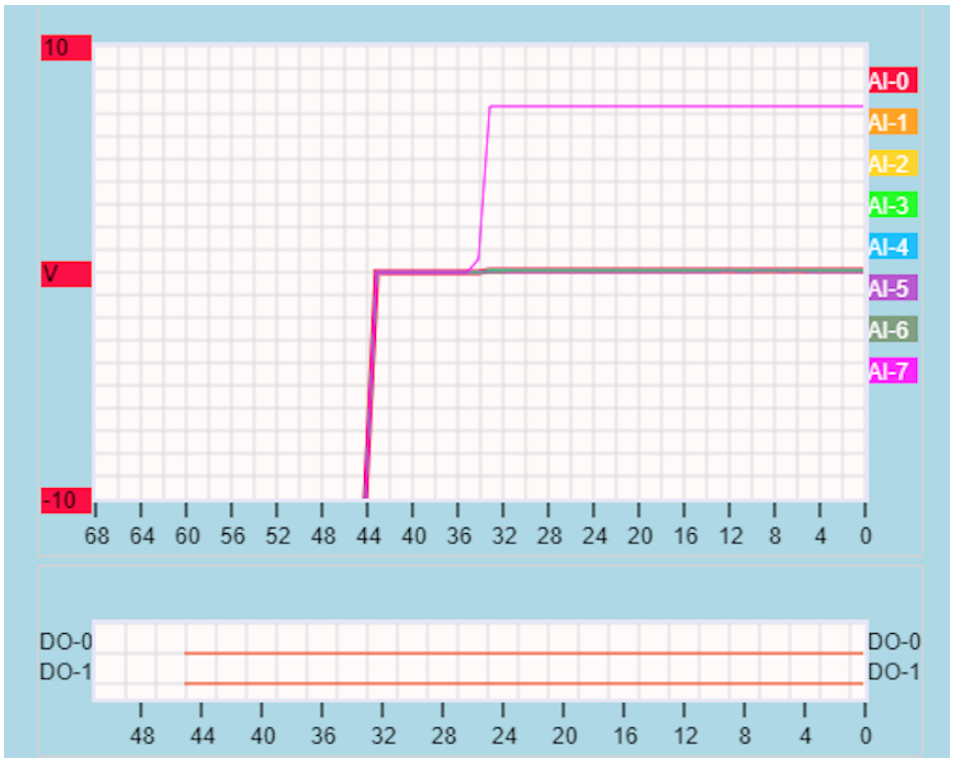


Figura 5.21: Gráfico da aplicação *web* do Módulo de Aquisição I/O.

Analog Input Values			
Polling 43 times...			
AI-0	AI-1	AI-2	AI-3
0.045 V	0.003 V	0.002 V	0.001 V
AI-4	AI-5	AI-6	AI-7
0.001 V	-0.000 V	0.001 V	7.280 V
Digital Output Values			

Figura 5.22: Tensão das entradas analógicas do Módulo de Aquisição de Entradas Analógicas.

Na configuração deve-se definir o IP do dispositivo, o IP do *broker* **MQTT** e aplicar as alterações para que seja estabelecida comunicação.

5.5.2.1 Comunicação com o Node-RED

Desenvolveu-se um *flow* que subscrevesse o tópico **MQTT** onde os valores são publicados pelo módulo ADAM já devidamente configurado. Analisando a Figura 5.24 entende-se, de forma muito intuitiva, a lógica associada.

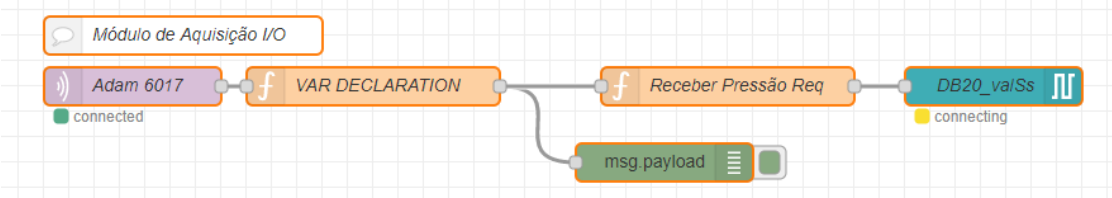


Figura 5.23: *Flow* em Node-RED para aquisição do valor analógico.

O valor analógico é adquirido subscrevendo o tópico **MQTT**. Este passa por um nó de funções onde é processado para ser convertido num valor adequado a esta aplicação, ficando, dessa forma, apto a ser escrito num *bit* de um *Datablock* do programa autómato.

5.5.2.2 Programação do Autómato no TIA-Portal

Recorrendo ao *software* **TIA-Portal** escreveu-se um programa *ladder* para fazer a gestão da frequência de comando do variador e, consequentemente, da velocidade do motor, em função de um ***input*** recebido, que é neste caso o valor da tensão analógica aos terminais do potenciómetro. O uso de um PLC para o controlo do valor da frequência de comando foi motivado por dois fatores: o controlo dos variadores de frequência na AviSabor é feito por autómatos e estudar a eficácia de um autómato num sistema automático neste projeto. Como já foi mencionado anteriormente, o Node-RED pode escrever ou ler valores contidos em *bits* de um autómato Siemens, através do nó **S7-Comm** disponibilizado.

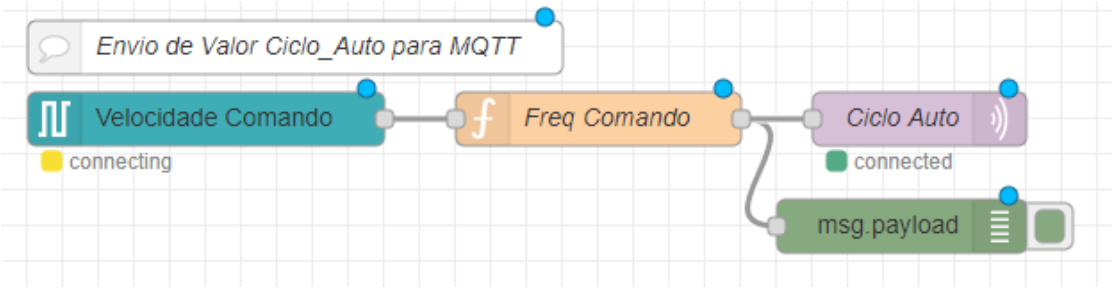


Figura 5.24: *Flow* de publicação do valor da frequência de comando num tópico MQTT.

Seguindo a lógica da Figura 5.24, consoante o valor da pressão à saída da bomba (simulada pelo valor analógico do potenciómetro), o autómato envia ao Node-RED um comando para aumentar ou diminuir a frequência de comando do variador para valores definidos pelo utilizador na interface. Este processa o valor enviado pelo autómato e usando um nó *function* para converter o valor recebido numa mensagem pronta a ser interpretada pelo variador, publicando-a num tópico **MQTT** exclusivo para este modo.

O ESP deverá estar igualmente subscrito a este tópico e, assim que enviada a mensagem, esta deverá ser convertida numa mensagem série e enviada ao variador, como se pode ver no fluxograma da Figura 5.25.

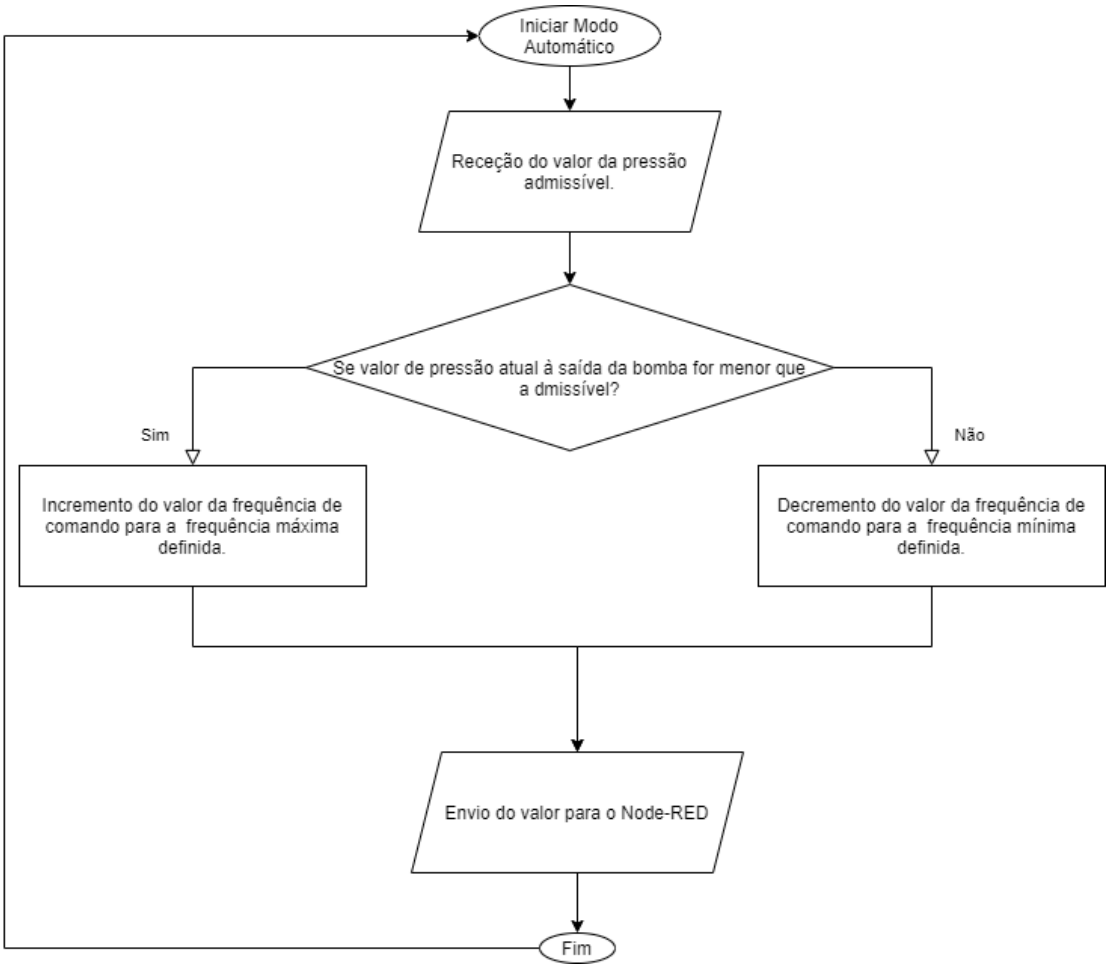


Figura 5.25: Fluxograma do programa *ladder* responsável pelo Ciclo Automático.

5.5.3 Sistema de Aquisição de Dados (FIT)

O seu funcionamento é igual ao já abordado na Secção 4.3, pelo que fica encarregue de estar à escuta de publicações no tópico **MQTT** onde o módulo publica. A mensagem recebida é então processada por um algoritmo que converte os dados extraídos numa mensagem série própria do protocolo Modbus-RTU, para ser interpretada pelo variador de frequência.

5.5.4 Ativação do Modo Automático de Regulação da Frequência

O Modo Automático é controlado a partir da caixa "Modo Automático" da página *web* da Figura 5.26, onde pode ser ativado ou desativado. Após a sua ativação, o utilizador deve definir o limite da pressão de saída admissível e os limites superior e inferior da frequência. O resto do processo é automático, pelo que o sistema reage em função do valor analógico adquirido pelo Módulo AI/DO, aumentando ou diminuindo a frequência de comando do variador de frequência.

Figura 5.26: Caixa do Modo Automático na página *web*.

5.5.5 Domínio de IP's

Em computação, uma rede de área local consiste numa rede de computadores utilizada para a interconexão entre equipamentos, cuja finalidade é a troca de dados. É um conjunto de *hardware* e *software* que permite a computadores individuais estabelecerem comunicação entre si, trocando informações e recursos [42].

Para a criação de uma rede local neste projeto, foi utilizado o ***router*** TP-LINK W841N (Imagem (g) da Figura 5.20). A rede local serviu para endereçar todos os equipamentos no mesmo domínio de rede e permitir desta forma que se fixassem os IP's para trocar mensagens com *hardware* específico do barramento.

Capítulo 6

Análise de desempenho

6.1 Desempenho do Sistema SCADA

Tendo em conta os requisitos definidos para a solução, este capítulo consiste numa análise crítica do desempenho da solução implementada por forma a verificar se a mesma cumpre os requisitos propostos.

6.1.1 Sistema de Aquisição de Dados (Frequência, Intensidade de Corrente, Tensão)

Relativamente ao desempenho do Sistema de Aquisição de Dados (FIT), dever-se-ia ter desenvolvido um método de comprovação de resultados para estudar a sua fiabilidade e robustez mediante a conceção de um mecanismo para gerar mensagens tanto no aparelho emissor como no recetor, para comprovar que as mensagens enviadas são efetivamente iguais às recebidas, assim como o tempo que demoram na transmissão para, dessa forma, verificar a fiabilidade do funcionamento do sistema desenvolvido. Há que destacar uma limitação que é a sua aplicabilidade noutro tipo de variadores. A programação do ESP foi realizada tendo por base a norma RS-485 do protocolo Modbus-RTU. Assim, qualquer dispositivo que se pretenda monitorizar ou controlar que comunique segundo as normas de outro protocolo, necessita do desenvolvimento de um algoritmo específico que siga as regras adequadas.

Os resultados obtidos com este Sistema de Aquisição foram os esperados, sendo que esteve a funcionar várias horas ininterruptamente e este manteve-se fiável em execução, cumprindo o seu propósito de escrita/leitura de registos do variador.

6.1.2 Servidor Local (Raspberry Pi)

O módulo utilizado para servir de servidor na realização deste projeto foi um Raspberry 3 Model B 5.3.1.

O aparelho cumpriu o seu papel ao desempenhar as tarefas impostas, funcionando tanto como um *broker* **MQTT** e como armazém de toda a informação. Todas as ligações estabelecidas pelo mesmo, mantiveram-se estáveis em toda a sua duração.

O nome e função das tabelas criadas na base de dados podem ser consultadas na Tabela 6.1 a seguir:

Tabela 6.1: Tabelas criadas na base de dados.

Nome	**Função**
Valores_VFD	Armazena os valores adquiridos de frequência, tensão e intensidade de corrente
LimCicloAuto	Armazena os valores limite de frequência máxima, mínima e pressão de *setpoint* definidos pelo utilizador
LimAlarmes	Armazena os valores limite de frequência, intensidade de corrente, e tensão que fazem disparar os alarmes definidos pelo utilizador

Para testar as funcionalidades do sistema, deixou-se o Sistema de Aquisição e os restantes aparelhos a funcionar durante 6 horas aproximadamente. A nível de leitura e de escrita na Base de Dados, o Node-RED não apresentou quaisquer falhas na comunicação, cumprindo sempre o seu papel de intermediário entre os dados registados ou a registar e a amostragem dos mesmos na página *web*.

Uma outra solução viável para guardar ou ler valores da base de dados, seria através do envio de pedidos **HTTP** ao servidor.

Na Figura 6.1 pode-se ver o fluxo responsável pela seleção dos valores de frequência, intensidade de corrente e tensão mais recentes da Tabela "Valores_VFD" do repositório de dados, seguido da conversão do respetivo valor para formato **JSON** e a criação de um *array* com esse valor e a data e hora do registo para ser reencaminhado para a página *web*.

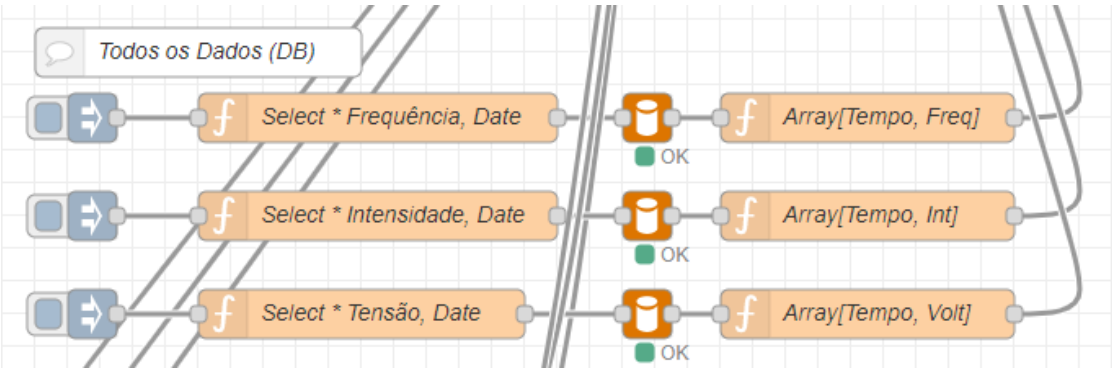


Figura 6.1: Nós de comunicação com a base de dados.

6.1.3 Página *web*

O resultado final da página *web* (interface) encontra-se representado na Figura 6.4. A Figura 6.2 mostra as várias abas acessíveis dentro da página, cada uma contendo informações diferentes mas relacionadas. Desta forma, a página encontra-se divida da seguinte forma:

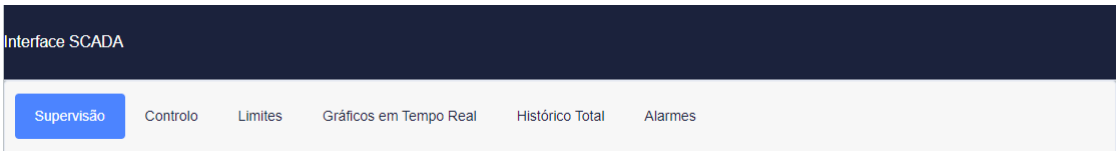


Figura 6.2: Vários separadores presentes na interface.

- **Supervisão** - Valores instantâneos, gráficos em tempo real e principais alarmes;
- **Controlo** - Possibilidade de controlar a frequência, escolher entre modo manual e remoto, ativar o Modo Automático e comandos básicos (arranque/paragem);
- **Limites** - Alteração dos valores limite que fazem disparar os alarmes;
- **Gráficos em Tempo Real** - Amostragem dos valores recolhidos instantaneamente;
- **Histórico Total** - Visualização de todos os registos da base de dados;
- **Alarmes de Limite Atingido** - Concentra os principais eventos e alarmes que vão ocorrendo.

6.1.3.1 Supervisão

O separador "Supervisão" consiste num sinóptico global onde são apresentados em *gauges* os valores que estão a ser adquiridos no momento, os gráficos em tempo real e uma caixa de alarmes onde são registados os principais eventos e alarmes. A atualização dos valores não é instantânea e demora cerca de 1.5-2 segundos a atualizar.

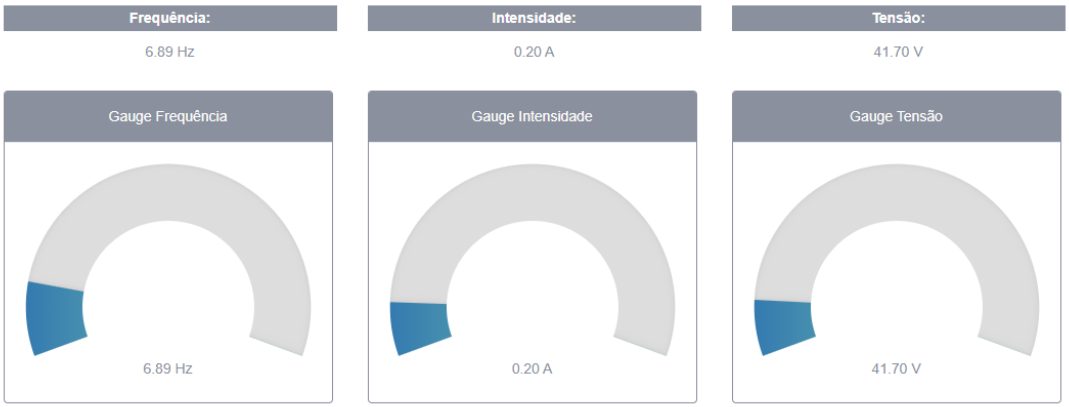


Figura 6.3: *Gauges* associados a cada variável.

6.1.3.2 Controlo

O separador "Controlo" é, como o nome indica, onde se controla o estado das bombas (maior ou menor velocidade) através da parametrização da frequência de comando do variador. Para isso, deve-se definir qual o modo de funcionamento do variador, manual ou remoto e de seguida, basta arrastar o *slider* ou escolher diretamente um valor. O tempo reação do sistema aos comandos enviados é praticamente nulo, pelo que o controlo sobre a execução do sistema se pode considerar instantâneo.

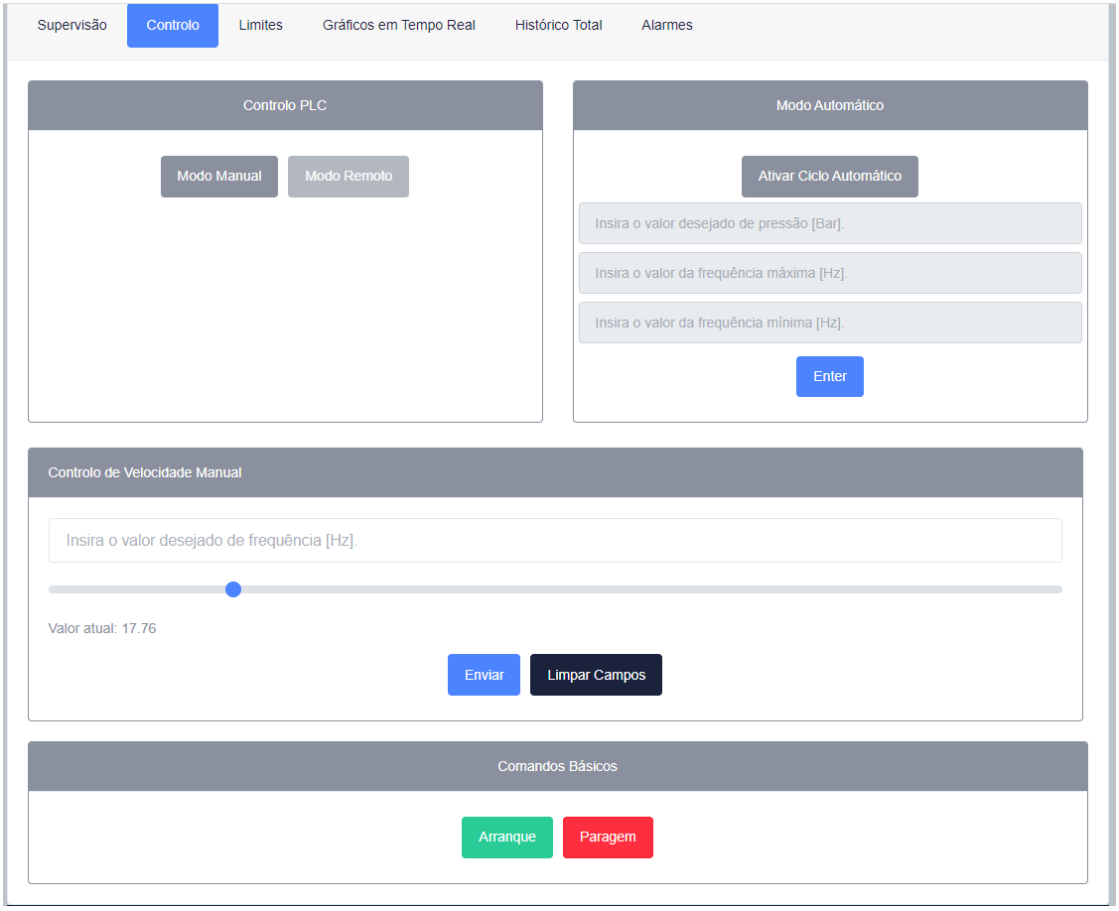


Figura 6.4: Exemplo da aba de "Controlo" da página *web*.

6.1.3.3 Gráficos

Tanto o separador "Gráficos em Tempo real" e "Histórico Total" têm a mesma função que é a amostragem dos valores recolhidos em gráficos. No entanto, o primeiro mostra os valores adquiridos no momento, e o segundo mostra todos os valores registados na base de dados. Os valores recolhidos são armazenados em intervalos de 3 segundos, pelo que o separador dos "Gráficos em Tempo real" atualiza os seus gráficos a cada 3 segundos com o último valor armazenado, que coincide com o último valor recolhido.

Poder-se-ia implementar um registo de valores com maior cadência, por exemplo, de 1 em 1 segundo. No entanto, um armazenamento e pedido de leitura de valores num intervalo de tempo tão curto pode gerar incongruências na amostragem da informação e

estes não serem representativos dos valores que estão a ser realmente adquiridos.

6.1.3.4 Alarmes de Limite Atingido

O separador "Alarmes" é exclusivamente dedicado ao registo dos principais eventos e alarmes que acontecem na *dashboard*. Este dá ainda a opção ao utilizador de fazer o *download* do registo desses eventos para serem consultados posteriormente. Em apêndice está uma demonstração simples para a conversão de um ficheiro *.txt* num ficheiro cuja extensão é compatível com Excel, para melhor organização da informação.

Figura 6.5: Conteúdo do separador de alarmes.

6.1.4 Modo Automático de Regulação de Frequência

A implementação deste modo serviu para teste da aplicabilidade e do funcionamento de todo o sistema.

6.1.4.1 Reação do Sistema

Os dados adquiridos são escritos num *bit* específico de um programa autómato desenvolvido para o efeito. Consoante os limites definidos pelo utilizador no separador "Limites" da interface, o sistema reage, aumentando ou diminuindo a frequência de comando. O tempo de reação ronda 4 segundos, pelo que não pode ser considerado rápido. No entanto, cumpre o seu propósito, regulando o valor da frequência de comando.

6.1.5 Modo Remoto vs Modo Automático de Regulação de Frequência

Esta secção tem por fim a comparação dos dois modos implementados: o Modo Remoto e o Modo Automático de Regulação de Frequência.

O Modo Remoto foi desenvolvido para permitir controlar a frequência de comando do variador a partir da página *web*, enquanto que o Modo Automático foi desenvolvido para testar a reação do sistema face a um ***input*** que foi o valor analógico da tensão de um potenciómetro.

Com o desenvolvimento destes dois modos, torna-se interessante perceber a diferença entre eles e as vantagens e as desvantagens que têm um relação ao outro. Para isso, consulte-se a Tabela 6.2:

Tabela 6.2: Tabelas de comparação dos modos.

Tempos de Resposta em função de eventos	**Modo Remoto**	**Modo Automático**
Arranque	0s	0s
Paragem	0s	0s
Alteração do valor da frequência de comando	0s	4s
Alarme	0s	0s
Alerta via *e-mail*	3s	3s

Por um lado, no Modo Remoto, detém-se maior controlo sobre o variador de frequência, dado que é possível regular a frequência de comando (até à frequência máxima de 400Hz) com total liberdade. Assim, é possível controlar a velocidade do veio de um motor ou o caudal à saída de uma bomba de forma instantânea, o que pode constituir uma vantagem em alguns projetos IoT. Como numa qualquer cadeia de produção se pretende, geralmente, automatizar a grande maioria dos processos, haver um modo que implique que haja alguém a operar um equipamento (neste caso, um variador de frequência) não é viável, pelo faz todo o sentido o desenvolvimento de um modo em que o utilizador apenas tenha que introduzir os parâmetros entre os quais o equipamento pode variar em função de um determinado ***input***. No entanto, há que destacar que o tempo de reação Modo Automático é significativamente superior ao Modo Remoto, refletindo-se no atraso do incremento/decremento de velocidade de uma bomba.

Capítulo 7

Conclusões e Trabalhos Futuros

7.1 Conclusões

Atualmente, com o avanço tecnológico e a introdução da indústria 4.0, o mercado exige ferramentas cada vez mais poderosas e precisas, que permitam grande flexibilidade e controlo sobre o processo produtivo ou parte dele, garantido o cumprimento de todas as fases e desta forma assegurar a qualidade do produto final.

A falta de supervisão dos equipamentos conduz muitas vezes a situações em que tanto o fluxo produtivo como os aparelhos que fazem parte de um processo ficam comprometidos, pelo que é cada vez mais importante que haja um acompanhamento constante do processo produtivo. Assim, a monitorização assume um papel fulcral em qualquer indústria, visto que permite um acompanhamento eficaz da matéria-prima enquanto esta é processada até se transformar no produto final, pronto a ser expedido.

Desta forma, em colaboração com a AviSabor, S.A, foi proposto o desenvolvimento de um Sistema SCADA para apoio à manutenção dos equipamentos associados ao sistema de bombeamento de água para o interior da instalação fabril. Este assentava nos dois pressupostos supramencionados, ou seja, supervisão e controlo dos variadores de frequência responsáveis pelo controlo das bombas que distribuem água pela empresa e dos autómatos que os controlam. Como estes equipamentos se encontram numa parte externa à fábrica, sempre que exista um problema relacionado com a captação de água (por falta de água nos reservatórios, no caso de haver uma bomba desferrada, se uma bomba queimar devido a picos de corrente ou de tensão), os técnicos de manutenção se tenham que deslocar até à sala para poder averiguar a situação e encontrar uma solução. Um dos focos seria então permitir a aquisição dos valores contidos nos registos destes variadores de frequência e enviá-los para uma parte central da empresa por forma a não haver deslocações desnecessárias. Outro foco consistiu em aplicar manutenção preventiva no processo produtivo no sentido em que, de acordo com os limites (de frequência, intensidade de corrente ou tensão) que se impusessem e os dados que estivessem a ser adquiridos, o protótipo envia alarmes por *e-mail* ao utilizador, por forma a notificá-lo de consumos anormais de corrente ou de tensão.

O estudo da eficiência energética do sistema de bombeamento teria sido de grande utilidade. Acedendo aos registos do variador, podem-se adquirir os valores de consumo energético que podiam ter conduzido o trabalho realizado para a otimização da eficiência energética do sistema de bombeamento. No entanto, como o problema surgiu de um pressuposto prático, com a falta de supervisão, o trabalho acabou por se focar na

automatização desse aspeto para solucionar apenas esse problema.

Apesar de pouco complexo, o sistema idealizado é funcional, sendo capaz de cumprir os objetivos propostos no início da concretização deste projeto. Foram desenvolvidas duas versões deste sistema: uma primeira, em que o Sistema de Aquisição de Dados assenta numa placa de teste (*breadboard*) e uma segunda, numa placa de circuito impresso (PCB). A versão desenvolvida na *breadboard* teve apenas o intuito de testar todas as capacidades do sistema em laboratório e, quando validados esses testes, desenvolveu-se uma nova versão assente numa PCB, que foi concebida pela flexibilidade que esta dá, podendo-se alterar os componentes facilmente.

Ao nível da implementação na empresa, o projeto ficou na fase de prova de conceito, ou seja, todos os testes foram realizados com equipamentos de substituição, diferentes dos utilizados pela AviSabor. Ainda que os protocolos de comunicação dos variadores da empresa e os de teste sejam iguais, a nível industrial, os autómatos e os variadores estão protegidos para assegurar as condições de operação e desta forma evitar alteração dos programas em execução ou até roubo de informação.

Relativamente aos resultados obtidos, foram realizados testes de precisão e fiabilidade das amostras recolhidas. Ao nível da monitorização, fez-se variar a frequência do comando sem qualquer critério, apenas com o intuito de testar a capacidade de aquisição de dados reais do protótipo desenvolvido. O Sistema de Aquisição de Dados mostrou-se robusto e a recolha de valores manteve-se precisa ao longo da duração dos testes, sendo os parâmetros coincidentes com os indicados pelo painel do variador. Ao nível do controlo, a resposta do sistema é instantânea, sendo que o tempo de reação do sistema face ao *input* de um novo valor de frequência de comando é nulo, o que o torna muito rápido e prático.

Seguindo esta linha de ideias, implementou-se um Modo Automático de Regulação de Frequência (abordado na Secção 5.5) para tentar aproximar o comportamento do sistema desenvolvido ao que aconteceria caso se adaptasse o mesmo num ambiente industrial. Utilizou-se o valor da tensão analógica aos terminais de um potenciómetro para simular o valor adquirido por um sensor de pressão à saída de uma bomba e fez-se variar manualmente o mesmo para estudar o comportamento do sistema. O lógica é simples: com o aumento da pressão de saída, diminui-se a frequência de comando para ajustar o valor da pressão, sendo o contrário igualmente válido, ou seja, com a diminuição do valor da pressão de saída, aumenta-se o valor da frequência de comando. Tudo isto acontece de forma automática graças ao programa *ladder* em execução no autómato , sendo apenas necessário fazer variar o valor analógico da tensão aos terminais do potenciómetro, que atua como um *input*.

O motor AC utilizado não tinha carga. Dessa forma, como se configurou o variador de frequência para que o motor tivesse um torque pequeno, reduziu-se em laboratório a velocidade do veio através do uso de materiais com elevada fricção encostados ao veio. Ora para a mesma frequência de comando, esta redução provoca picos de corrente e de tensão devido ao esforço do motor para fazer rodar o veio à velocidade suposta. Definindo limites inferiores ao valor máximo dos picos de qualquer um dos parâmetros, o alarme é lançado na página *web* e o utilizador recebe de imediato um *e-mail* que o avisa sobre o alcance deste limite, para que este tenha oportunidade de atuar sobre o equipamento em esforço.

Uma limitação deste projeto, é a sua aplicabilidade noutro tipo de variadores. A programação do microcontrolador (ESP32) foi realizada tendo por base a norma RS-485 inerente ao protocolo ModBus-RTU. Assim, qualquer dispositivo que se pretenda monito-

rizar ou controlar que comunique segundo outro protocolo, necessita do desenvolvimento de um algoritmo específico que siga as normas adequadas. Apesar da elevada taxa configurada para o envio e receção de mensagens, é importante que os equipamentos fiquem à escuta mais tempo do que o necessário teoricamente (*delay*) para que estes possam processá-las devidamente e reencaminhá-las para os destinatários certos, podendo, caso contrário, dar-se a um registo errado de dados.

Comparando com outras soluções abordadas no Capítulo 3, vê-se que existem ainda inúmeras otimizações que podiam ser aplicadas e tornar a solução atual numa mais prática e adaptável. O curto alcance de comunicação sem fios inter-dispositivos implica que todo os componentes deste sistema estejam dispostos perto uns dos outros. Apesar de um dos objetivos deste projeto ser o controlo remoto, este é apenas possível se todos eles estiverem ligados a uma rede que gire o IP de todos os equipamentos. Ainda que seja possível associar uma rede local à *internet*, no testes realizados na AviSabor isso foi difícil, visto que os variadores de frequência se encontram numa zona da empresa que as redes WiFi não conseguem alcançar e onde uma ligação GSM não é estável, dificultando a possibilidade de comunicação *online*.

O Node-RED foi a ferramenta que serviu de base a este projeto. Consiste num ambiente de desenvolvimento baseado em fluxos, ou *flows* para programação visual concebida para interligar equipamentos físicos (*hardware*) a Interfaces de Programação de Aplicações (as ***API's***) a serviços *online* [43]. Este reúne uma série de condições que o tornam numa ferramenta ótima para aplicações IoT dada a sua rápida e fácil implementação. A sua lógica é de muito fácil interpretação e construir uma sequência de operações consiste em arrastar nós (existem inúmeros pacotes - *palletes* - de nós para todo o tipo de aplicações: ligação a PLCs, ligação a bases de dados MySQL, criação de páginas *web* entre outros) e uni-los, não havendo necessidade de conhecimentos profundos de programação. Este *software* tem uma riqueza de bibliotecas enorme, tendo já sido utilizado para inúmeras aplicações que são armazenadas em repositórios como o GitHub e graças aos seus utilizadores e *developers* vai ficando mais cada vez mais refinado.

Quanto aos protocolos utilizados, note-se que as soluções abordadas no Capítulo 3 não fazem uso da ferramenta Node-RED nem do protocolo **MQTT** para a troca de informação, usando para isso ambientes de programação convencionais e o protocolo **HTTP**. O protocolo **MQTT** revelou-se uma ferramenta muito capaz e adequada a aplicações IoT, contendo três níveis de ***QoS*** (veja-se a Tabela 4.2), a opção *Last Will and Testament*, responsável por alertar os clientes sobre a desconexão de outro ao *broker*, e a opcção *Retained Messages*, na qual um cliente subscrito a um tópico recebe de imediato a última mensagem publicada, o que constitui uma vantagem relativamente ao protocolo **HTTP**, que recorre a mensagens mais pesadas e a cabeçalhos mais complexos para gerir as trocas de informação entre o cliente e o servidor.

O módulo utilizado para o servidor foi um Raspberry Pi 3B+. Ao longo de todos os testes realizados, o dispositivo mostrou um desempenho aceitável. No entanto, há que realçar que, a sua execução em períodos mais longos provocava um aumento de temperatura no dispositvo e, como este não é refrigerado, notaram-se algumas paragens na execução do Node-RED, chegando mesmo ao ponto de mandar o *software* abaixo e o módulo se reiniciar sozinho. Não foram ocorrências frequentes nem problemáticas, no entanto em ambiente industrial podem levar a perdas de informação que pode ser importante para a execução das tarefas no posto onde o aparelho se encontra implementado.

Concluindo, o sistema mostrou-se fiável na execução dos objetivos propostos em la-

boratório. No entanto, convém destacar que a sua aplicabilidade seria um ambiente industrial, que em muitos casos são ambientes com condições austeras (quer de ruído eletromagnético, poeiras, humidade, temperatura, entre outros) e, portanto, a sua fiabilidade não é um dado certo. Dever-se-ia ter realizado uma série de testes mais intensos as capacidades do Sistema de Aquisição de Dados sob stress, compondo à prova a sua verdadeira eficácia em ambiente industrial.

7.2 Trabalhos Futuros

Terminado este projeto mas mantendo um olhar crítico sobre o foi feito, percebe-se que a abordagem ao problema podia ter sido feita de outras maneiras. De seguida, far-se-ão algumas sugestões para complementar e otimizar o presente projeto:

- No Sistema de Aquisição de Dados, este pode ser mais compacto, deixando de lado os módulos separados e usando apenas os componentes estritamente necessários. O Sistema projetado faz uso do protocolo Modbus-RTU para receber valores dos registos do equipamento. Uma opção interessante seria a conceção de algoritmos para comunicar segundo as normas de outros protocolos, como por exemplo o protocolo DeviceNet, comum nos variadores da marca dinamarquesa Danfoss, utilizados também pela AviSabor.
- O Sistema de Aquisição de Dados liga-se a uma rede local que reencaminha toda a informação para o servidor. No entanto, para além de todo os aparelhos se encontrarem próximos uns dos outros, não se utilizou nenhum módulo que permitisse comunicação de longo alcance com outro microcontrolador com ligação à *internet* (ligação à WiFi da empresa, por exemplo). Um módulo LoRa é um dispositivo de comunicação sem fios que recorre à modulação de ondas de rádio para comunicações de longo alcance e que pode facilmente ser implementado para operar em conjunto com um ESP32. Na Figura 7.1 está representado um esquema simples do seu funcionamento, no qual um módulo de recolha de dados (*moisture sensor*), cujo funcionamento se assemelha ao deste projeto, reencaminha o a informação recolhida para uma página *web* alojada num servidor, graças ao *chip* LoRa.

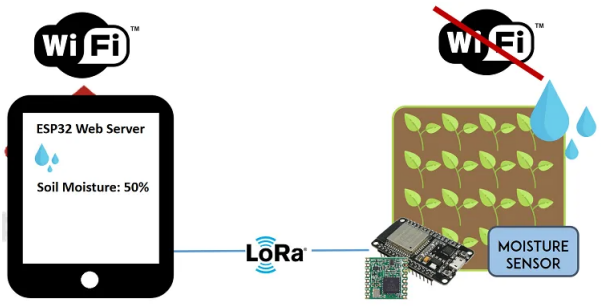


Figura 7.1: Esquema representativo do funcionamento de um *chip* LoRa.

- Uma adição interessante seria a implementação um algoritmo de *Machine Learning*. Como este projeto foi desenvolvido em laboratório, as condições são muito controladas. A ausência de fatores próprios de um ambiente industrial como ruído

eletromagnético, alcance das comunicações, entre outros, condicionaram o desempenho do sistema no sentido em que não houve falhas de troca de mensagens e discrepâncias nos valores adquiridos, e quando houve, foi devido a eventos muito esporádicos que despoletaram esses erros que deram origem a alguns dados sem expressão. Desta forma, não havendo um padrão de dados real, não se justifica a implementação de um algoritmo de *Machine Learning* neste projeto. No entanto, em ambiente real, se se quisesse efetuar uma manutenção preditiva nestes equipamentos, seria impreterível o desenvolvimento de um projeto que não recorresse a *Machine Learning*, visto que o intuito de uma manutenção deste tipo é prever a falha antes que esta ocorra. Assim, ao contrário da Manutenção Preventiva, que recorre às estatísticas do tempo útil de vida de um equipamento, a Manutenção Preditiva baseia-se na condição atual de qualquer equipamento para estimar a necessidade de manutenção.

- A interface gráfica desenvolvida resultou apenas numa implementação local. De facto, ainda que esta seja facilmente acedida através de uma ligação *wireless*, implica que o utilizador tenha que estar ligado à mesma rede que o servidor onde esta se encontra. Seria interessante alojar a página *web* e o repositório de dados numa *cloud* (como a **Microsoft Azure IoT Suite**) para que se possa aceder ao sistema SCADA de forma remota, a partir de qualquer rede WiFi, a qualquer momento.

- Na AviSabor existem equipamentos ao longo da linha de produção sem qualquer tipo de supervisão. Seria de maior utilidade ter mais sistemas SCADA a funcionar como o deste projeto, no sentido em que, havendo um mecanismo de envio de alarmes a partir de um aparelho específico, os técnicos de manutenção podiam atuar na fonte do problema. Situações deste género ocorrem frequentemente, em que os técnicos têm que andar à procura da avaria ao longo da linha.

Bibliografia

[1] J. P. O. Santos, "Apontamentos Teóricos de Automação II 2017-2018." Universidade de Aveiro.

[2] K. Sipsas, K. Alexopoulos, V. Xanthakis, and G. Chryssolouris, "Collaborative maintenance in flow-line manufacturing environments: An Industry 4.0 approach," 2016.

[3] A. F. F. de Caldas, "Proposta de uma solução SCADA para a manutenção preditiva," Master's thesis, Universidade de Aveiro, Novembro 2016.

[4] A. M. P. R. Rodrigues de Almeida, "Monitorização e controlo remoto de sistemas de rega agrícola," Master's thesis, Universidade de Aveiro, 2017.

[5] "ODBC," Maio 2020. Disponível em: `https://pt.wikipedia.org/w/index.php?title=ODBC&oldid=58201026`, Acedido em: 2020-11-26.

[6] M. Corporation, "O que é o ERP e porque precisa dele? | Microsoft Dynamics 365." Disponível em: `https://dynamics.microsoft.com/pt-pt/erp/what-is-erp/`, Acedido em: 2020-11-26.

[7] "SCADA," Outubro 2020. Disponível em `https://en.wikipedia.org/w/index.php?title=SCADA&oldid=981543773`, Acedido em: 2020-03-22.

[8] D. Silva e Costa, "Sistemas de Supervisão e Controlo Integrados," Master's thesis, Universidade de Aveiro, 2009.

[9] G. Tiburski, T. Moreira, and M. Misaghi, "Integração de Sistema Supervisório SCADA e ERP para controle de produção em tempo real,"

[10] M. Åkerman, *Implementing Shop Floor IT for Industry 4.0*. PhD thesis, 06 2018.

[11] "Simatic WinCC (TIA Portal) Runtime." Disponível em `https://mall.industry.siemens.com/mall/en/WW/Catalog/Products/10091384?tree=CatalogTree`, Acedido em: 2020-05-10.

[12] "Movicon Power HMI Win32/64/CE." Disponível em `https://www.pixsys.net/en/products/software-tools-complementary-fittings/movicon-scada`, Acedido em: 2020-05-10.

[13] "Simatic S7-1200 - Take Control of Communication." Disponível em `https://new.siemens.com/global/en/products/automation/systems/industrial/plc/s7-1200.html`, Acedido em: 2020-05-11.

[14] OMRON, *Technical Explanation for Inverters*, Junho 2012. Rev. 10.0.

[15] "Delta AC Drives." Disponível em: https://www.deltaacdrives.com, Acedido em: 2020-10-24.

[16] "NodeMCU ESP-32S." Disponível em https://docs.zerynth.com/latest/reference/boards/nodemcu_esp32/docs/, Acedido em: 2020-03-04.

[17] D. U. Press, "A Internet das Coisas (IoT – Internet of Things)." Disponível em https://www.cncs.gov.pt/a-internet-das-coisas-iot-internet-of-things/, Acedido em: 2020-07-07.

[18] E. Foundation, "White Papers | IoT development made simple - iot.eclipse.org."

[19] P. Ferrara, A. Mandal, A. Cortesi, and F. Spoto, *Cross Programming Language Taint Analysis for the IoT Ecosystem.* Abril 2019.

[20] "Microsoft Azure IoT Suite – Connecting Your Things to the Cloud | Blogue e Atualizações do Azure | Microsoft Azure." Disponível em https://azure.microsoft.com/pt-pt/blog/microsoft-azure-iot-suite-connecting-your-things-to-the-cloud/, Acedido em: 2020-07-07.

[21] R. Kang, "From Big Data to Business Intelligence - An Important Step Toward Gaining Advantages in the IoT Market," *Advantech*, Março 2015. Disponível em https://www.advantech.com/embedded-boards-design-in-services/embedded-iot/insight/0ed41d88-9c94-419d-93e4-b23b6325ca81, Acedido em: 2020-06-20.

[22] "MindSphere," Junho 2020. Disponível em https://en.wikipedia.org/w/index.php?title=MindSphere&oldid=963948256, Acedido em: 2020-05-05.

[23] "Industrial Edge from Siemens adds benefits from the cloud at the field level.." Disponível em https://press.siemens.com/global/en/pressrelease/industrial-edge-siemens-adds-benefits-cloud-field-level, Acedido em: 2020-05-04.

[24] "The core of the Bosch IoT Suite is based on open source." Disponível em https://developer.bosch-iot-suite.com/open-source/, Acedido em: 2020-07-07.

[25] "Bosch IoT Suite." Disponível em https://www.huaweicloud.com/en-us/solution/bosch-iot-suite/, Acedido em: 2020-06-20.

[26] "Arduino," Outubro 2020. Disponível em https://pt.wikipedia.org/w/index.php?title=JavaScript&oldid=59399666, Acedido em: 2020-05-05.

[27] "Cascading Style Sheets," Setembro 2020. Disponível em: https://pt.wikipedia.org/w/index.php?title=Cascading_Style_Sheets&oldid=59292010, Acedido em: 2020-11-02.

[28] "Html," Outubro 2020. Disponível em https://pt.wikipedia.org/w/index.php?title=HTML&oldid=59577601, Acedido em: 2020-05-05.

[29] "Hypertext Transfer Protocol," Agosto 2020. Disponível em https://pt.wikipedia.org/w/index.php?title=Hypertext_Transfer_Protocol&oldid=59099039, Acedido em: 2020-05-05.

[30] "Javascript," Setembro 2020. Disponível em https://pt.wikipedia.org/w/index.php?title=JavaScript&oldid=59399666, Acedido em: 2020-05-05.

[31] "phpmyadmin," Março 2020. Disponível em https://pt.wikipedia.org/w/index.php?title=PhpMyAdmin&oldid=57858625, Acedido em: 2020-06-21.

[32] "Wireshark," Outubro 2020. Disponível em https://pt.wikipedia.org/w/index.php?title=Wireshark&oldid=59490619, Acedido em: 2020-03-20.

[33] "RS-485 Cable: Critical to System Operation." Disponível em https://www.belden.com/blog/industrial-ethernet/rs-485-cable-critical-to-system-operation, Acedido em: 2020-09-11.

[34] "RS-485," Outubro 2020. Disponível em: https://en.wikipedia.org/w/index.php?title=RS-485&oldid=985848229, Acedido em: 2020-10-27.

[35] M. Kabir, "Introduction to TCP/IP," Maio 2020.

[36] "TCP/IP," Agosto 2020. Disponível em https://pt.wikipedia.org/w/index.php?title=TCP/IP&oldid=59201225, Acedido em: 2020-08-05.

[37] W. GmbH, "Your Own Web HMI/SCADA Solution in Just A Few Minutes," Junho 2019. Disponível em https://webfactory-i4.com/stories/your-own-web-hmi/scada-solution-in-just-a-few-minutes-sps-2019/, Acedido em: 2020-06-25.

[38] "Inverters - Mitsubishi Electric Factory Automation - EMEA." Disponível em: https://eu3a.mitsubishielectric.com/fa/en/products/drv/inv/local, Acedido em: 2020-10-24.

[39] S. Ec, "Frequency Inverter Instruction Manual," p. 504.

[40] "Raspberry Pi 3 Model B – Raspberry Pi Specs." Disponível em https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, Acedido em: 2020-06-10.

[41] "ISA95, Enterprise-Control System Integration- ISA." Disponível em: https://www.isa.org/standards-and-publications/isa-standards/isa-standards-committees/isa95, Acedido em: 2020-11-11.

[42] "Rede de área local," Setembro 2020. Disponível em: https://pt.wikipedia.org/w/index.php?title=Rede_de_%C3%A1rea_local&oldid=59477011, Acedido em: 2020-09-25.

[43] "Node-RED," Setembro 2020. Disponível em https://en.wikipedia.org/w/index.php?title=Node-RED&oldid=977745186, Acedido em: 2020-06-20.

Apêndice A

Desenvolvimento

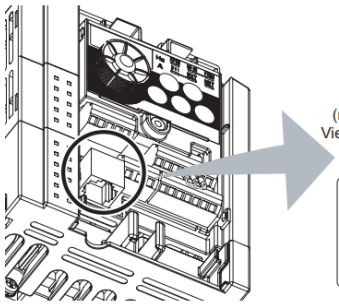
A.1 Configuração do Variador

Control circuit specifications

●RS-485 communication

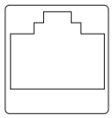
When the PU connector is connected with a personal, FA or other computer by a communication cable, a user program can run and monitor the inverter or read and write to parameters.
The protocol can be selected from Mitsubishi inverter and Modbus-RTU.

- **PU connector pin-outs**



Inverter
(receptacle side)
Viewed from bottom

8) to 1)



Pin Number	Name	Description
1)	SG	Earth (ground) (connected to terminal 5)
2)	—	Parameter unit power supply
3)	RDA	Inverter receive+
4)	SDB	Inverter send-
5)	SDA	Inverter send+
6)	RDB	Inverter receive-
7)	SG	Earth (ground) (connected to terminal 5)
8)	—	Parameter unit power supply

NOTE

- **Pins No. 2 and 8 provide power to the parameter unit. Do not use these pins for RS-485 communication.**
- **When making RS-485 communication with a combination of the FR-D700 series, FR-E500 series and FR-S500 series, incorrect connection of pins No.2 and 8 (parameter unit power supply) of the above PU connector may result in the inverter malfunction or failure.**
- **Do not connect the PU connector to the computer's LAN board, FAX modem socket or telephone modular connector. The product could be damaged due to differences in electrical specifications.**

Figura A.1: ***Pinout*** do conector **PU** (igual ao RJ45) do variador de frequência.

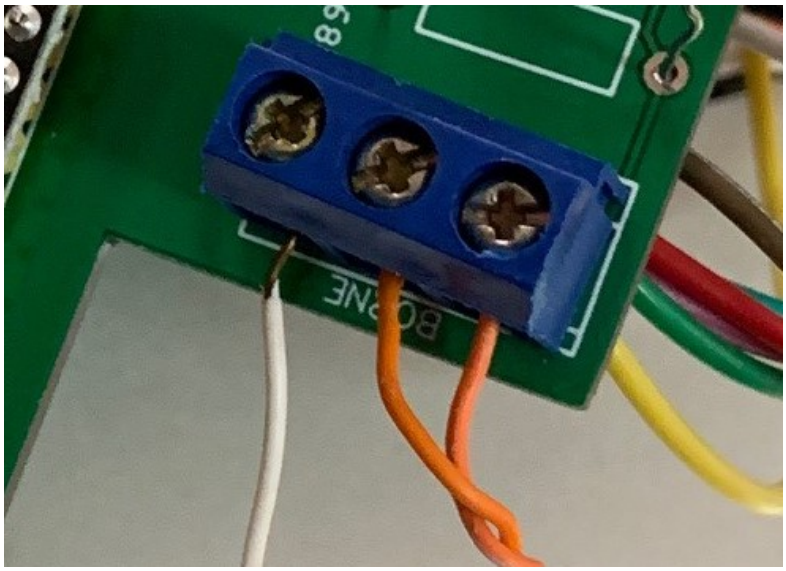


Figura A.2: Conector de três pinos da PCB do Sistema de Aquisição de Dados.

Figura A.3: Variador de Frequência utilizado (Mitsubishi FR-D720S).

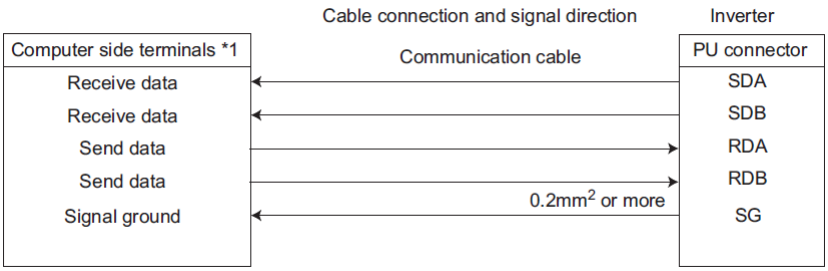


Figura A.4: Ligação dos fios condutores entre o variador e o adaptador.

A.2 Esquema Elétricos

A.2.1 Diagrama de Componentes

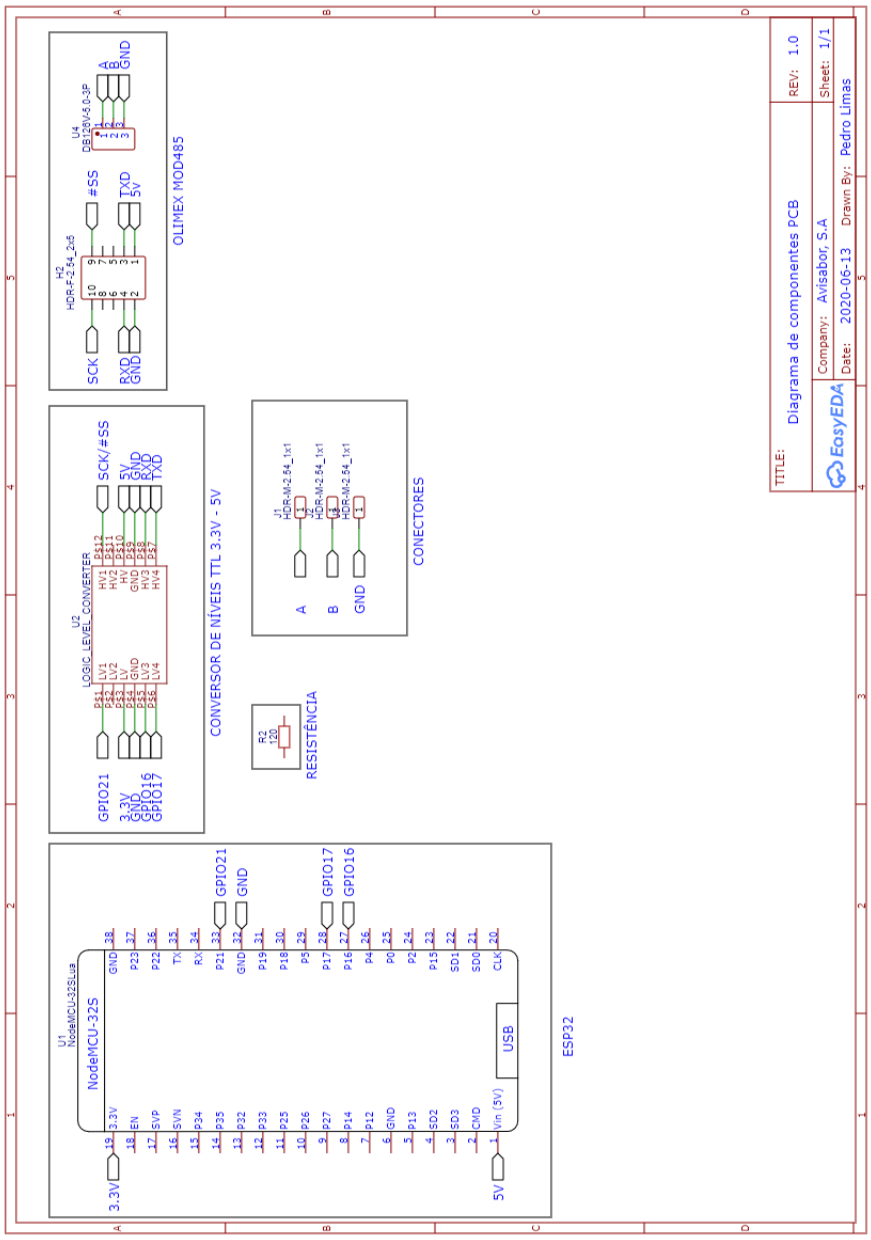


Figura A.5: Diagrama de componentes.

A.2.2 Esquema Elétrico da PCB

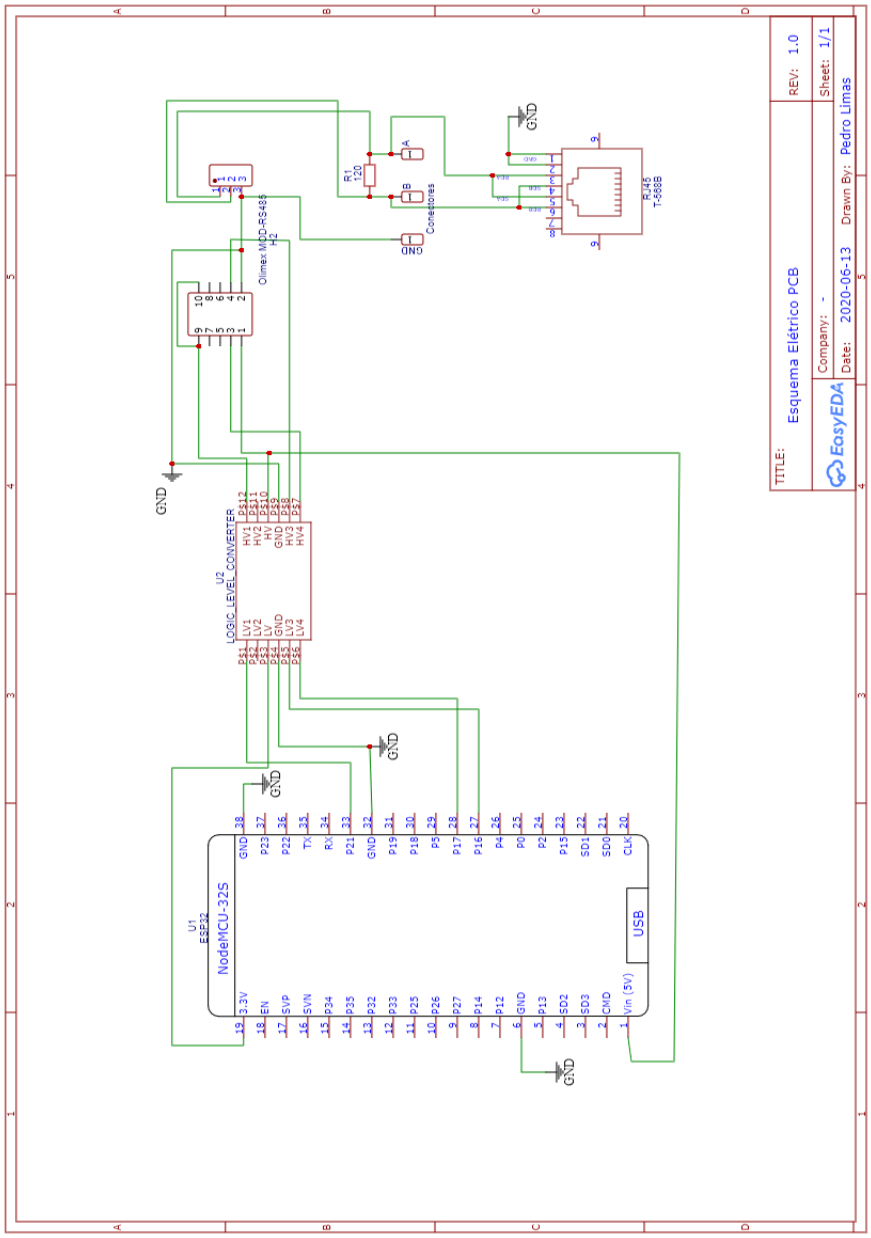


Figura A.6: Esquema de ligações da placa de circuito impresso.

A.3 Alterações no Node-RED

A.3.0.1 Atualização da base de dados

Caso seja necessário inserir ou atualizar as credenciais da base de dados, deve-se alterar os parâmetros "User", "Password"e "Database"da Figura A.9:

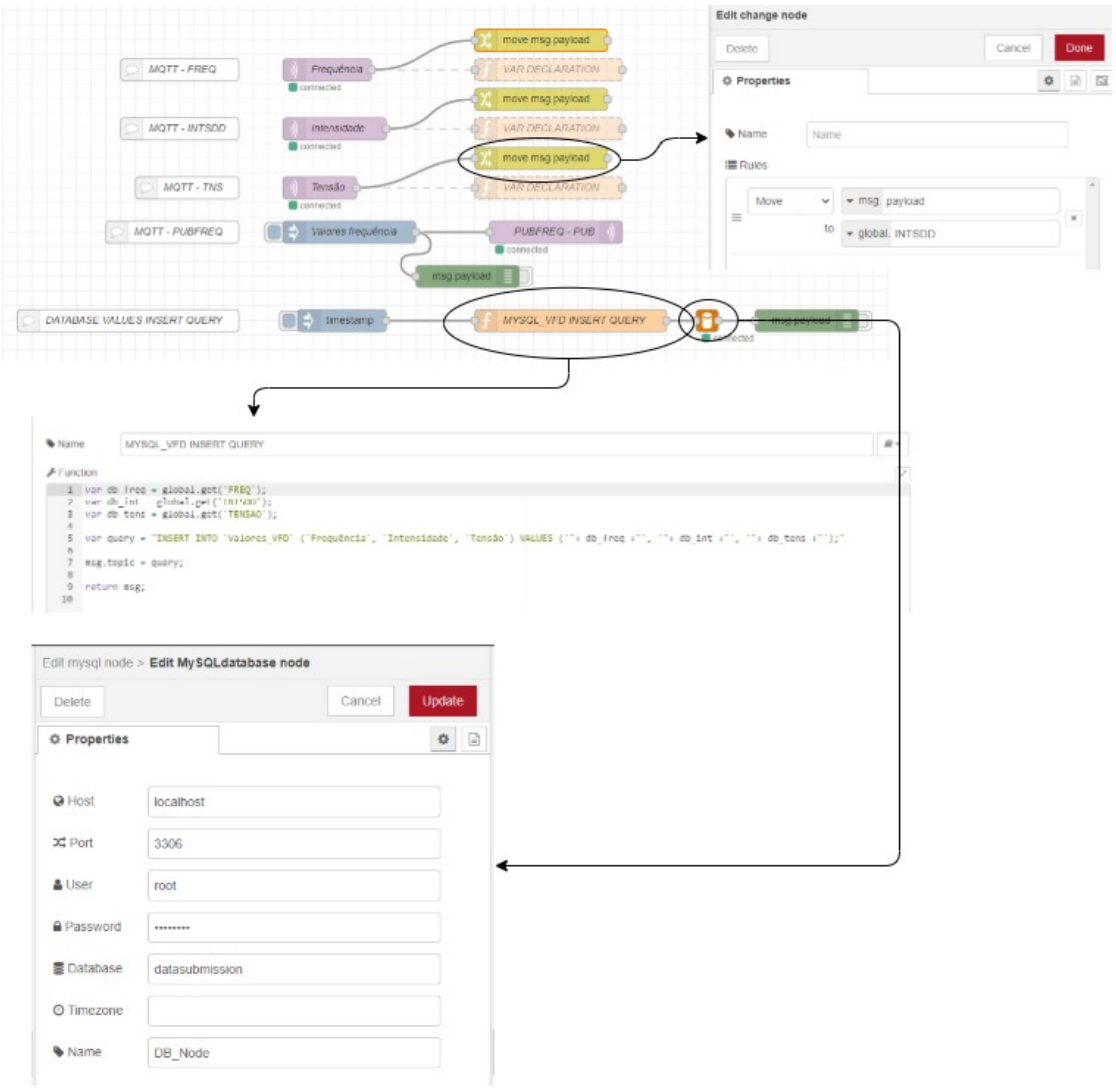


Figura A.7: Alteração das credenciais da base de dados.

A.3.0.2 Atualização do *e-mail*

Para mudar o *e-mail* de receção de alertas, altere-se os campos "To", "Userid"e "Password"da Figura a seguir:

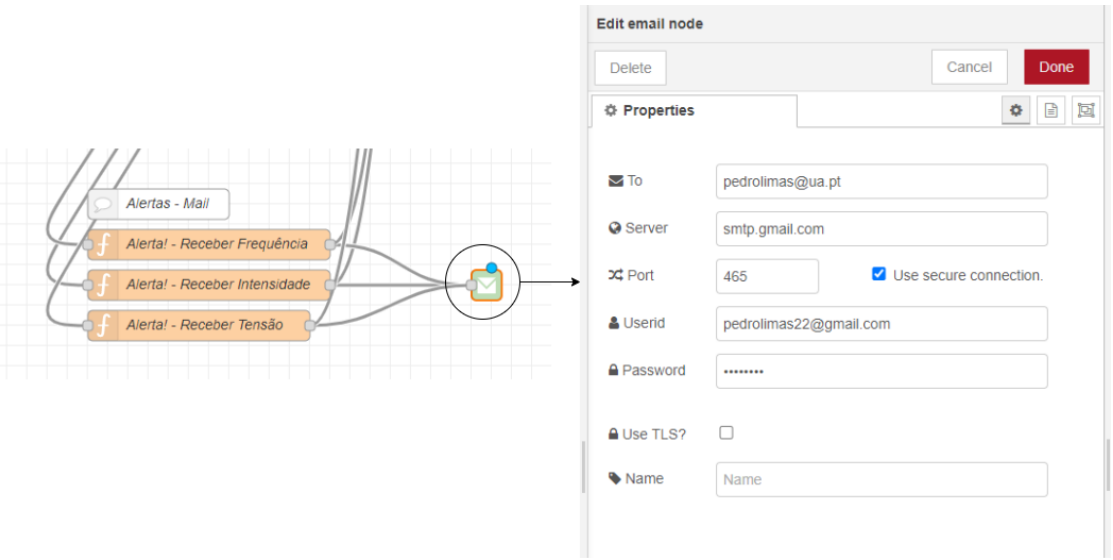


Figura A.8: Alteração do *mail*.

A.3.0.3 Atualização do *broker* MQTT

Para mudar o *e-mail* de receção de alertas, veja-se a seguir:

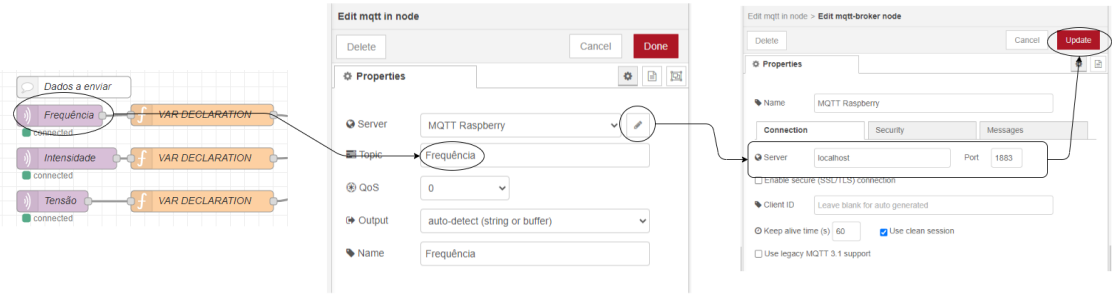


Figura A.9: Alteração dos dados do *broker* MQTT.

A.4 Alterações das definições do *mail*

O Node-RED recomenda a utilização de *mails* associados ao serviço Gmail, visto que este permite a utilização de aplicações de terceiros mediante o seguimento dos passos representados nas Figuras que se seguem:

Figura A.10: 1º passo: Gerir a sua Conta Google.

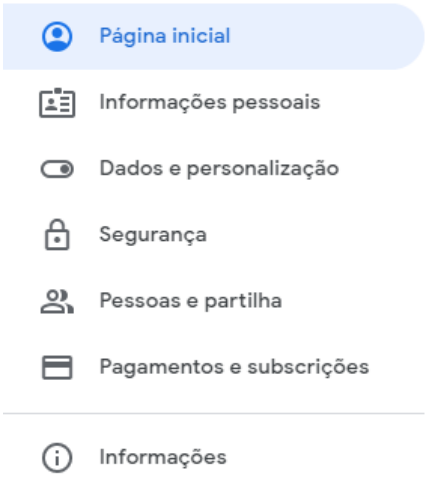


Figura A.11: 2º passo: Opção de Segurança.

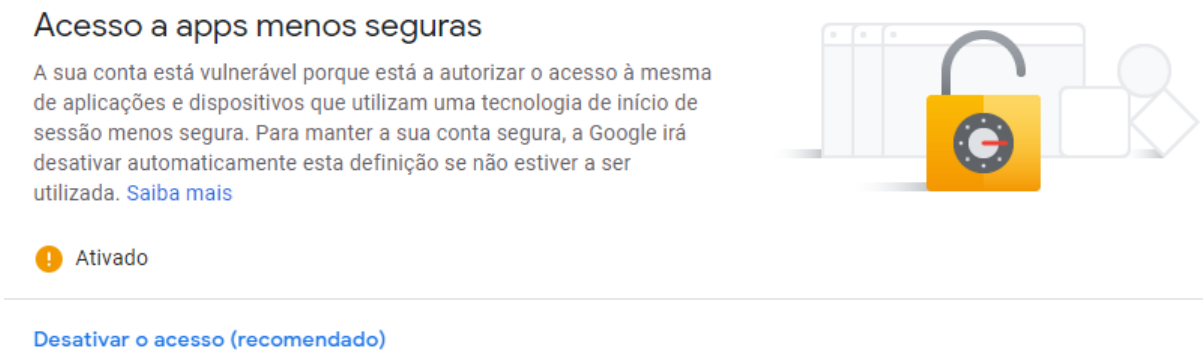


Figura A.12: 3º passo: Ativar o acesso em Acesso a apps menos seguras.

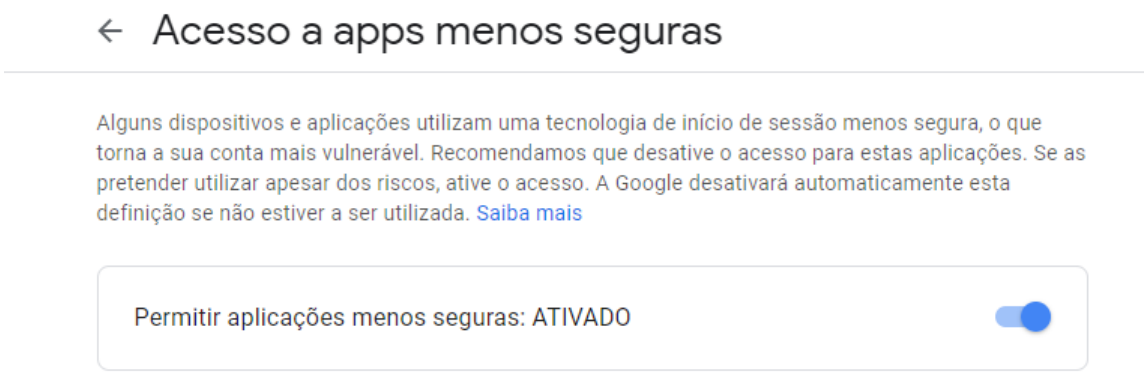


Figura A.13: 4º passo: Ativar a opção Permitir aplicações menos seguras.

A.5 Diagramas de Sequência

Esta imagem tem o intuito de explicar um exemplo de controlo e monitorização dos valores da frequência. Numa fase inicial, o utilizador escolhe através de um *slider* ou de um *input* em formato texto do valor da frequência. O Node-RED recebe e converte esse valor num formato específico para o microcontrolador programado. O microcontrolador, então subscrito a um tópico MQTT, adquire o valor publicado, converte-o e envia-o sob forma de uma mensagem série pronta a ser interpretada pelo variador. A bomba reage de imeadiato aumentando ou diminuindo a frequência de comando.

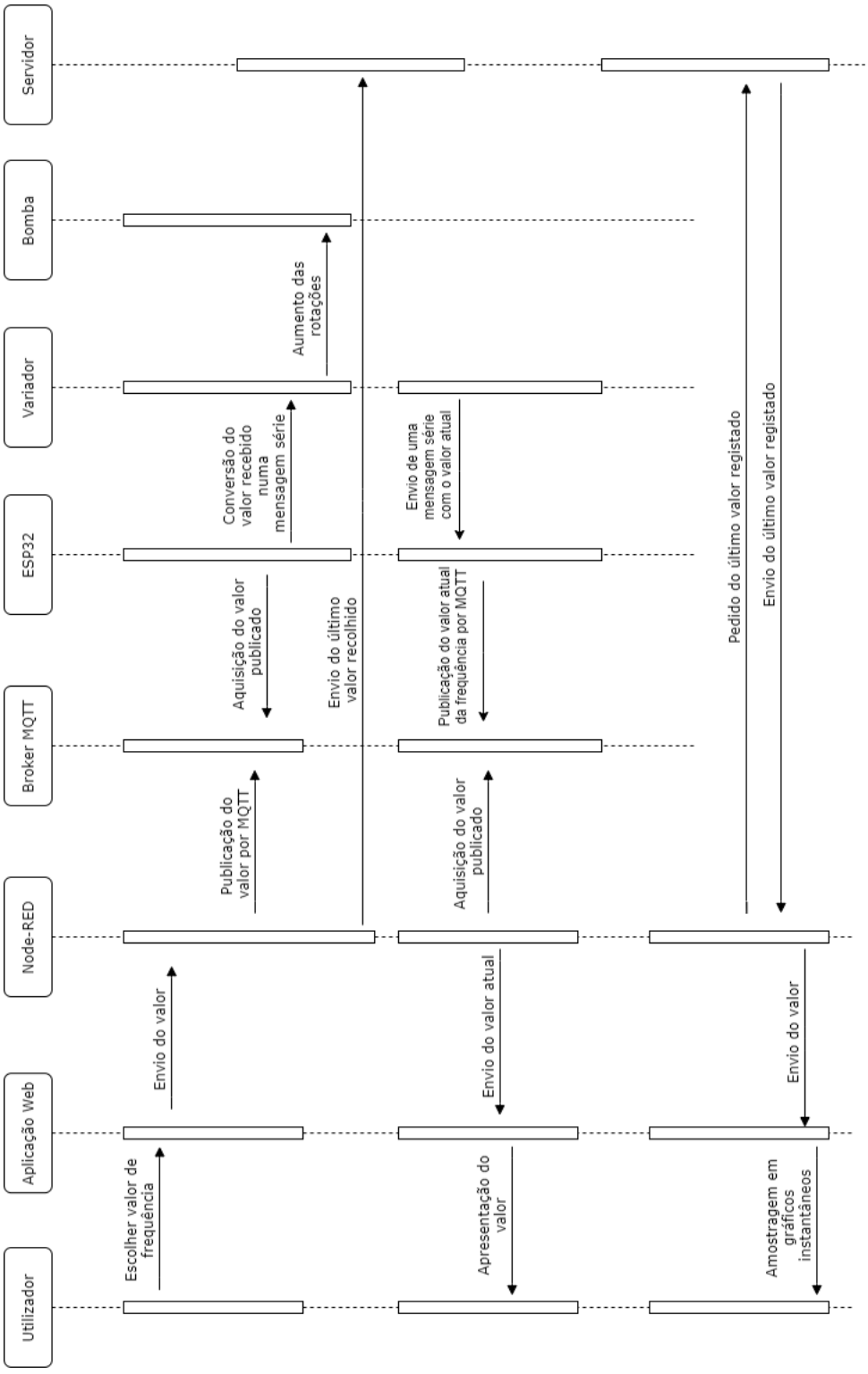


Figura A.14: Diagrama de sequência de controlo e monitorização.

A.6 Registo de Alarmes

Para uma melhor organização e para se poder armazenar os eventos e alarmes de interesse que ocorrem neste sistema, representa-se nas figuras seguintes uma sequência de imagens com os passos que se devem seguir para converter o ficheiro .txt que se faz *download* a partir da página *web*:

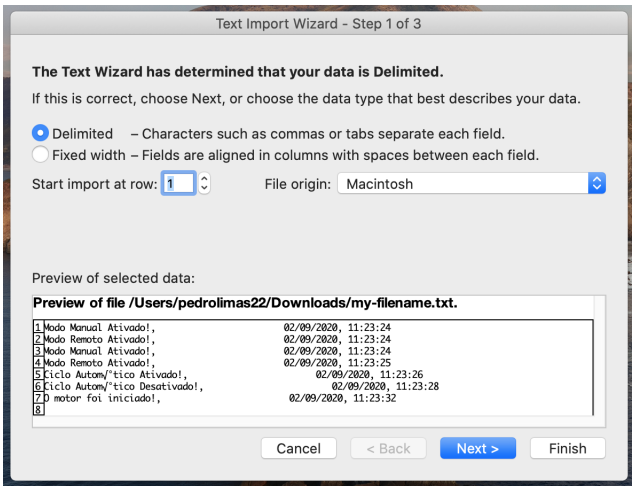


Figura A.15: 1º passo - abrir o ficheiro .txt em Excel.

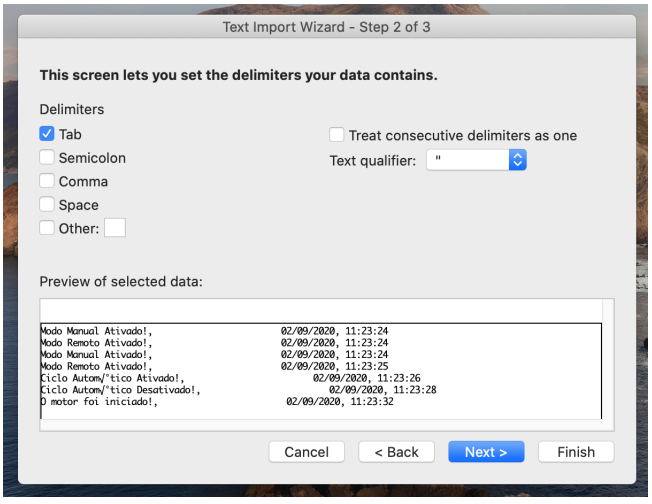


Figura A.16: 2º passo - Várias opções dos carateres delimitadores de colunas.

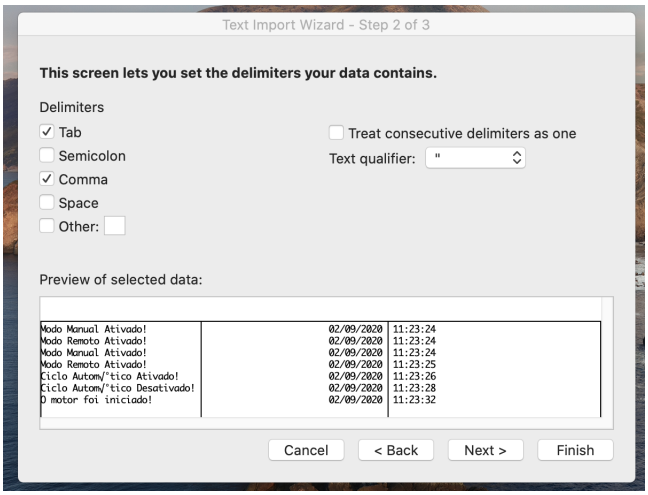


Figura A.17: 3º passo - Escolher a "vírgula"como carater delimitador.

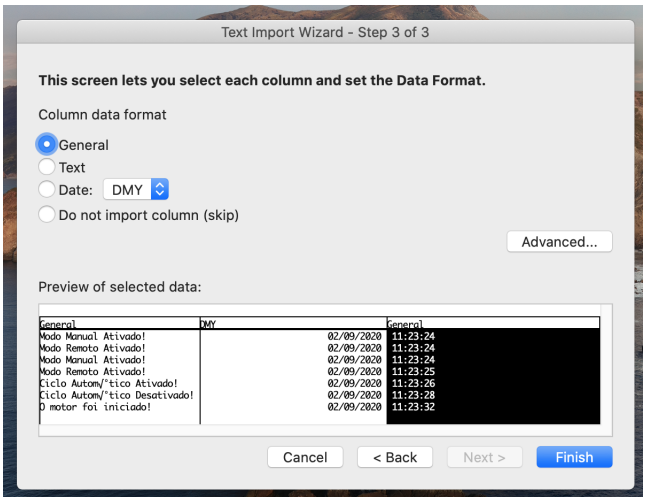


Figura A.18: 4º passo - Escolher o formato de cada coluna.

Modo Manual Ativado!	02/09/2020	11:23:24
Modo Remoto Ativado!	02/09/2020	11:23:24
Modo Manual Ativado!	02/09/2020	11:23:24
Modo Remoto Ativado!	02/09/2020	11:23:25
Ciclo Automático Ativado!	02/09/2020	11:23:26
Ciclo Automático Desativado!	02/09/2020	11:23:28
O motor foi iniciado!	02/09/2020	11:23:32

Figura A.19: 5º passo - Exemplo de uma tabela em Excel.

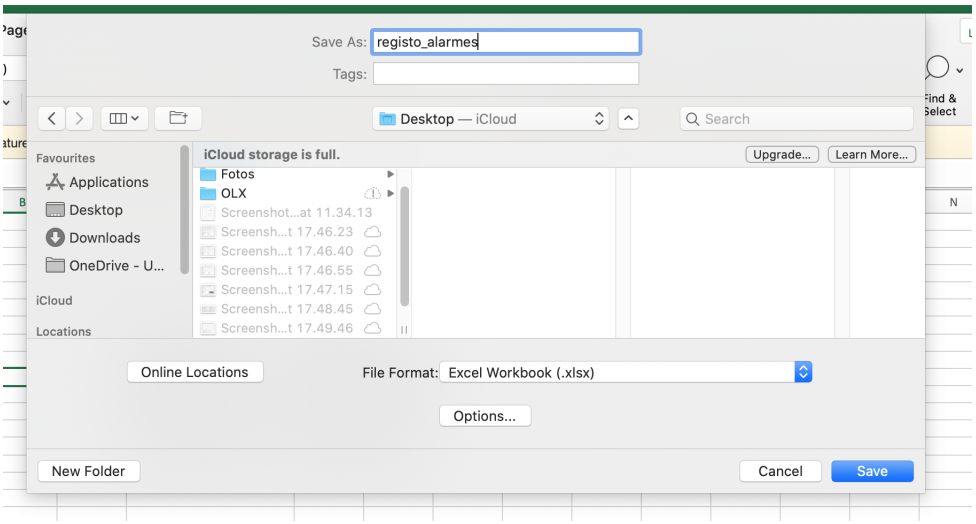
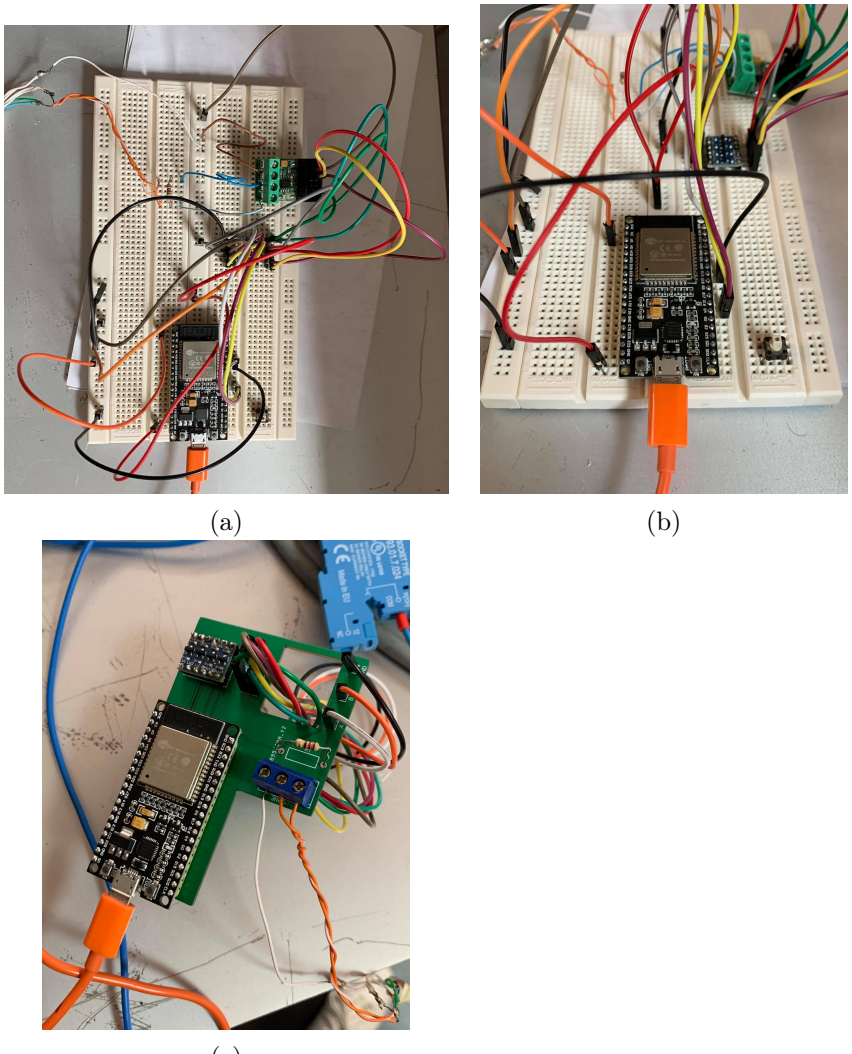


Figura A.20: Guardar o ficheiro num formato .xlsx.

A.7 Fotografias do Sistema na *breadboard*.

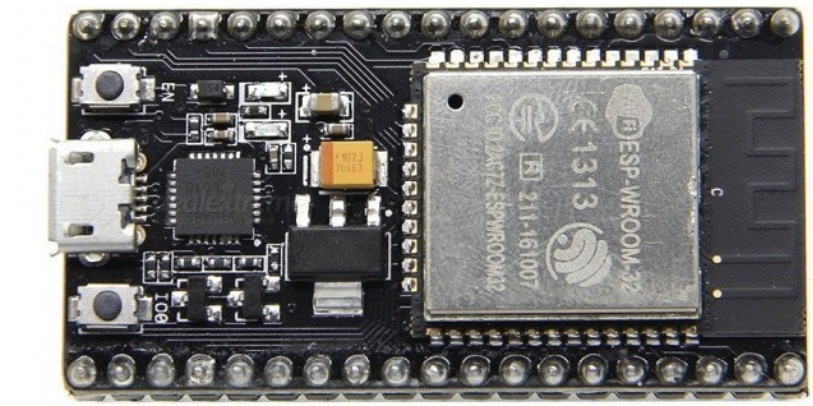


(a) (b)

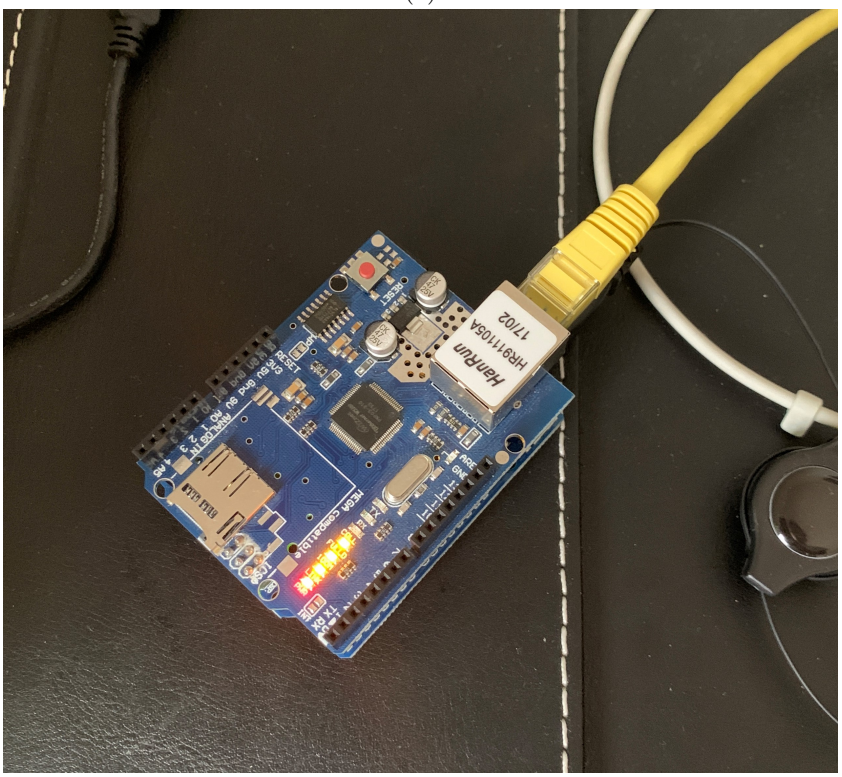
(c)

Figura A.21: Sistema de Aquisição de Dados desenvolvido na placa de prototipagem (*breadboard*) e na **PCB**.

A.8 Opções de Microcontroladores Implementáveis



(a)



(b)

Figura A.22: Opções de Microcontroladores Implementáveis: (a) ESP32, (b) Arduino + *Ethernet Shield*.

A.9 Código de programação - ESP32

A.9.1 Configuração da Rede Wi-Fi

Para configuração da rede à qual o microcontrolador se liga, deve-se alterar os parâmetros SSID e *password* para as credenciais da rede, nas linhas 29 e 30 do código, respetivamente.

A.9.2 Configuração do *broker* MQTT

Se se pretender usar outro IP para o *broker*, altera-se para o IP pretendido na linha 40.

A.9.2.1 Código de programação do Microcontrolador - ESP32

```
1 #include <HardwareSerial.h>
2 #include <ModbusMaster.h>
3 #include <PubSubClient.h>
4 #include <WiFi.h>
5 #include <Wire.h>
6
7 HardwareSerial MITSUBISHI( 2 ); // DECLARA  O DO OBJETO
    HARRDWARE SERIAL -> MITSUBISHI
8
9 // -----------------------[MODBUSMASTER - SETUP
    ]-------------------- //
10
11 ModbusMaster node; // DECLARA  O DO OBJETO ModbusMaster ->
    node (VFD)
12 ModbusMaster node2; // DECLARA  O DO OBJETO ModbusMaster ->
    node (VFD)
13
14 uint8_t deviceID = 1;
15 uint8_t resultMain;
16 String c, texto;
17 String ID, REGISTER, VALUE;
18
19
20 uint8_t deviceIDpf = 1;
21 uint8_t resultMainpf;
22 String  textopf;
23 String IDpf, REGISTERpf, VALUEpf;
24
25 // -----------------------[/MODBUSMASTER - SETUP
    ]-------------------- //
26
27 // SSID/PASSWORD DA REDE
28
29 const char* ssid = "Atena_IoT_LAN";
30 const char* password = "19739373";
31
32 void callback(char* topic, byte* payload, unsigned int length);
33
34 //PROCESSOS
35 TaskHandle_t Task0;
36 TaskHandle_t Task1;
37
38 // const char* mqtt_server = "192.168.137.125"; // BROKER'S IP
    ADDRESS
```

```
39 // const char* mqtt_server = "192.168.1.68"; // BROKER'S IP
     ADDRESS
40 const char* mqtt_server = "192.168.1.101"; // BROKER'S IP
     ADDRESS
41
42 WiFiClient espClient;
43 PubSubClient client(mqtt_server, 1883, callback, espClient);
44 long lastMsg = 0;
45 char Msg[50];
46 int value = 0;
47
48 #define RE_DE 21 // RE_DE PIN
49 #define TXD 17 // TX PIN
50 #define RXD 16 // RX PIN
51
52 #define BaudRate 9600 // BAUDRATE
53 #define Protocol SERIAL_8O1 // S RIE 8 BITS C/ 1 STOP BIT
54
55 void preTransmission() {
56   digitalWrite(RE_DE, HIGH);
57 }
58
59 void postTransmission() {
60   digitalWrite(RE_DE, LOW);
61 }
62
63 int bits_per_byte = 8;
64 int us_to_wait = (12 * bits_per_byte) * 1000000 / BaudRate; //
     = (1+8+1) bytes
65 int start_timer = 1000;
66 int end_timer = (6 * 11 * bits_per_byte) * 1000000 / BaudRate;
     // = 4x(1+9+1) bytes
67
68 byte MSGtoMITSUBISHI[8]; // BYTE - QUERY DE LEITURA
69 byte MSGtoMITSUBISHI2[8]; // BYTE - QUERY DE ESCRITA
70 byte MSGtoESP32[20]; // MENSAGEM ENVIADA AO ESP
71 byte MsgReceived[7]; // MENSAGEM RECEBIDA VARIADOR -> ESP
72
73 float f = 0.00;
74
75 //------------------------------------------- [FORMATO DE UMA
     MENSAGEM SERIAL]
     ---------------------------------------------- //
76
77 // Slave Address // Function Code // Byte Count //First
     Register Hi //First Register Lo //Error Check Lo (CRC) //
     Error Check Hi (CRC)
```

```
78
79 int start_index = 0;
80 int msgF = 0;
81 int msgI = 0;
82 int msgT = 0;
83 int location = 3;
84
85 void setup() {
86
87   Serial.begin(9600);
88   MITSUBISHI.begin(BaudRate, Protocol, RXD, TXD);
89   Serial2.begin(BaudRate, Protocol, RXD, TXD);
90
91   pinMode(RE_DE, OUTPUT);
92   digitalWrite(RE_DE, LOW);
93
94   node.preTransmission(preTransmission);
95   node.postTransmission(postTransmission);
96
97   node2.preTransmission(preTransmission);
98   node2.postTransmission(postTransmission);
99
100
101  setup_wifi();
102
103  // ----------------------[CABE ALHO DA QUERY DE LEITURA
       ]-------------------- //
104
105  //Slave Address - 0x01
106  MSGtoMITSUBISHI[0] = 0x01;
107  //Function Code - 0x03
108  MSGtoMITSUBISHI[1] = 0x03;
109  //Number of Points Hi - 0x00
110  MSGtoMITSUBISHI[4] = 0x00;
111  //Number of Points Lo - 0x01
112  MSGtoMITSUBISHI[5] = 0x01;
113
114  // ----------------------[/CABE ALHO DA QUERY DE LEITURA
       ]-------------------- //
115
116  // ----------------------[CABE ALHO DA QUERY DE ESCRITA
       ]--------------------- //
117
118  //Slave Address - 0x01
119  MSGtoMITSUBISHI2[0] = 0x01;
120  //Function Code - 0x06
121  MSGtoMITSUBISHI2[1] = 0x06;
```

```
122   //Number of Points Hi - 0x00
123   MSGtoMITSUBISHI2[2] = 0x00;
124   //Number of Points Lo - 0x0D
125   MSGtoMITSUBISHI2[3] = 0x0D;
126
127   // ----------------------[/CABE ALHO DA QUERY DE ESCRITA
        ]--------------------- //
128
129   delay(500);
130
131   xTaskCreatePinnedToCore(
132       Task1code, /* Function to implement the task */
133       "Task1", /* Name of the task */
134       10000,  /* Stack size in words */
135       NULL,  /* Task input parameter */
136       0,  /* Priority of the task */
137       &Task1,  /* Task handle. */
138       0); /* Core where the task should run */
139
140 }
141
142 void Task1code (void * parameter) {
143
144   for(;;) {
145
146      if (!client.connected())
147          reconnect();
148
149          client.loop();
150
151   }
152
153
154 }
155
156 void setup_wifi() {
157
158
159   delay(10);
160   // We start by connecting to a WiFi network
161   Serial.println();
162   Serial.print("Connecting␣to␣");
163   Serial.println(ssid);
164
165   WiFi.begin(ssid, password);
166
167   while (WiFi.status() != WL_CONNECTED) {
```

```
168     delay(500);
169     Serial.print(".");
170   }
171
172   Serial.println("");
173   Serial.println("WiFi connected");
174   Serial.println("IP address: ");
175   Serial.println(WiFi.localIP());
176
177 } // void setup_wifi
178
179 char* topiccp;
180
181
182 void callback(char* topic, byte* payload, unsigned int length)
      {
183
184   String messageTemp;
185
186   Serial.print("Message arrived in topic: ");
187   Serial.println(topic);
188
189
190   Serial.print("MESSAGE: ");
191
192   for (int i = 0; i < length; i++) {
193
194     Serial.print((char)payload[i]);
195     messageTemp += (char)payload[i];
196
197   }
198
199   textopf = messageTemp;
200   Serial.println();
201   Serial.println(textopf);
202
203
204   if (strcmp(topic, "esp/pubfreq") == 0)
205   {
206
207     Serial.println("AQUI");
208
209     IDpf = textopf.substring(2, 3);            // pega o
          n mero de ID enviado pela serial
210     //REGISTER = texto.substring(3, 7);      // pega o
          endere o da var avel que recebera a escrita ex: No inv
           delta ms300 a control word    2000H
```

```
211     VALUEpf = textopf.substring(3, 7);        // pega o valor q
            ser  escrito na vari vel Ex: o valor 0002H coloca o
           inv em RUN
212     deviceIDpf = 0;
213
214     switch (deviceIDpf) {
215
216       case 0:
217
218         node.begin(getInt(IDpf), Serial2);
219         resultMainpf = node.writeSingleRegister(0x000D, getInt(
               VALUEpf)); // para o inv delta a controlword    2000
               H = 8192 decimal   value = 0002 run, = 0001 stop
220
221         deviceIDpf = 1;
222         textopf = "";
223         IDpf = "";
224         //REGISTER = "";
225         VALUEpf = "";
226         textopf = "";
227
228         if (resultMainpf == node.ku8MBSuccess) {
229
230           Serial.println("O␣COMANDO␣FOI␣MANDADO␣AO␣VARIADOR.");
231
232         } else {
233
234           Serial.println("ERRO");
235           Serial.println("E S T ␣A␣PARAR␣NO␣1  ␣CASE.");
236
237         }
238
239         delay(45);
240
241   } // switch
242
243 } else if (strcmp(topic, "esp/cicloauto") == 0)  // if (strcmp)
      ... else if (strcmp)
244
245 {
246
247     Serial.println("AQUI2");
248
249     IDpf = textopf.substring(2, 3);            // pega o
          n mero de ID enviado pela serial
250     //REGISTER = texto.substring(3, 7);      // pega o
          endere o da var avel que recebera a escrita ex: No inv
```

```
        delta ms300 a control word    2000H
251     VALUEpf = textopf.substring(3, 7);        // pega o valor q
        ser  escrito na vari vel Ex: o valor 0002H coloca o
        inv em RUN
252     deviceIDpf = 0;
253
254     switch (deviceIDpf) {
255
256       case 0:
257
258         node.begin(getInt(IDpf), Serial2);
259         resultMainpf = node.writeSingleRegister(0x000D, getInt(
            VALUEpf)); // para o inv delta a controlword    2000
            H = 8192 decimal   value = 0002 run, = 0001 stop
260
261         deviceIDpf = 1;
262         textopf = "";
263         IDpf = "";
264         //REGISTER = "";
265         VALUEpf = "";
266         textopf = "";
267
268         if (resultMainpf == node.ku8MBSuccess) {
269
270           Serial.println("O␣COMANDO␣FOI␣MANDADO␣AO␣VARIADOR.");
271
272         } else {
273
274           Serial.println("ERRO");
275           Serial.println("E S T ␣A␣PARAR␣NO␣1  ␣CASE.");
276
277         }
278
279         delay(45);
280
281   } // switch
282
283 } // if topic == pubfreq// if topic == pubfreq
284
285 Serial.println();
286
287 Serial.println("----------------------");
288
289 messageTemp = "";
290
291
292 } // void callback
```

```
293
294
295 void reconnect() {
296
297   client.setServer(mqtt_server, 1883);
298   client.setCallback(callback);
299
300   // Loop until we’re reconnected
301   while (!client.connected()) {
302
303     Serial.print("Attempting␣MQTT␣connection...");
304     // Attempt to connect
305     if (client.connect("ESP32Client")) {
306
307       Serial.println("connected");
308
309       // MQTT TOPICS SUBSCRIPTION
310       client.subscribe("esp/pubfreq");
311       client.subscribe("esp/cicloauto");
312       client.subscribe("Intensidade");
313       client.subscribe("Frequ ncia");
314       client.subscribe("Tens o");
315
316     } else { // IN CASE THE BROKER CONNECTION FAILS
317
318       Serial.print("Failed␣connection,␣rc␣=␣");
319       Serial.print(client.state());
320       Serial.println("␣Trying␣again␣in␣5␣seconds");
321       // Wait 5 seconds before retrying
322       delay(1000);
323
324     }
325
326     delay(500);
327
328   } // while (!client.connected)
329
330 } // void reconnect
331
332 void loop() {
333
334
335   //---------------------------[LER FREQU NCIA
        ]-------------------------//
336   //Start Timer
337   delay(start_timer);
338
```

```
339   //Write Enable
340   digitalWrite(RE_DE, HIGH);
341   Serial.println("WRITE:␣01␣03␣00␣0D␣00␣01␣15␣C9");
342
343   MSGtoMITSUBISHI[2] = 0x00;
344   MSGtoMITSUBISHI[3] = 0x0D;
345   MSGtoMITSUBISHI[6] = 0x15;
346   MSGtoMITSUBISHI[7] = 0xC9;
347
348   MITSUBISHI.write(MSGtoMITSUBISHI, sizeof(MSGtoMITSUBISHI));
349   delayMicroseconds(us_to_wait);
350
351   //Read Enable
352   digitalWrite(RE_DE, LOW);
353   Serial.print("READ:");
354
355   //End Timer
356   delayMicroseconds(end_timer);
357   delay(500);
358
359   if (MITSUBISHI.available() >= sizeof(MsgReceived)) {
360     for (int i = 0; i < sizeof(MSGtoESP32); i++) {
361       MSGtoESP32[i] = MITSUBISHI.read();
362       if ((MSGtoESP32[i - 1] == MSGtoMITSUBISHI[0]) && (
              MSGtoESP32[i] == MSGtoMITSUBISHI[1])) {
363         start_index = i - 1;
364       }
365     }
366
367     for (int i = start_index; i < sizeof(MsgReceived); i++) {
368       MsgReceived[i - start_index] = MSGtoESP32[i];
369       Serial.print("␣");
370       int hex_value = MSGtoESP32[i];
371       if (hex_value < 0x10) {
372         Serial.print("0");
373       }
374       Serial.print(hex_value, HEX);
375     }
376     Serial.println();
377
378     msgF = 0;
379     msgF += MsgReceived[3];
380     msgF = msgF << 8;
381     msgF += MsgReceived[4];
382     delay(500);
383
384     float Frequency = ((float)msgF) / 100;
```

```
385
386     /**
387        Convert to char array to publish on the mqtt broker
388     */
389
390     // Convert the value to a char array
391     char freqString[8];
392     dtostrf(Frequency, 1, 2, freqString);
393     client.publish("Frequ ncia", freqString);
394
395     long now = millis();
396     if (now - lastMsg > 5000) {
397       lastMsg = now;
398     }
399
400     Serial.print("Frequ ncia: ");
401     Serial.print(Frequency, 2);
402     Serial.println(" Hz");
403     Serial.println("-----------------------------");
404
405   }
406
407   //--------------------------[/ LER FREQU NCIA
        ]-------------------------//
408
409   //--------------------------[LER INTENSIDADE
        ]-------------------------//
410
411   //Start Timer
412   delay(start_timer);
413
414   //Write Enable
415   digitalWrite(RE_DE, HIGH);
416   Serial.println("WRITE:  01 03 00 C9 00 01 54 34");
417
418   //Starting Address Hi - 0x00
419   MSGtoMITSUBISHI[2] = 0x00;
420   //Starting Address Lo - 0xC9
421   MSGtoMITSUBISHI[3] = 0xC9;
422   //Error Check Lo (CRC) - 0x54
423   MSGtoMITSUBISHI[6] = 0x54;
424   //Error Check Hi (CRC) - 0x34
425   MSGtoMITSUBISHI[7] = 0x34;
426
427   MITSUBISHI.write(MSGtoMITSUBISHI, sizeof(MSGtoMITSUBISHI));
428   delayMicroseconds(us_to_wait);
429
```

```
430   //Read Enable
431   digitalWrite(RE_DE, LOW);
432   Serial.print("READ:");
433
434   //End Timer
435   delayMicroseconds(end_timer);
436
437   if (MITSUBISHI.available() >= sizeof(MsgReceived)) {
438     for (int i = 0; i < sizeof(MSGtoESP32); i++) {
439       MSGtoESP32[i] = MITSUBISHI.read();
440       if ((MSGtoESP32[i - 1] == MSGtoMITSUBISHI[0]) && (
              MSGtoESP32[i] == MSGtoMITSUBISHI[1])) {
441         start_index = i - 1;
442       }
443     }
444
445     for (int i = start_index; i < sizeof(MsgReceived); i++) {
446       MsgReceived[i - start_index] = MSGtoESP32[i];
447       Serial.print("␣");
448       int hex_value = MSGtoESP32[i];
449       if (hex_value < 0x10) {
450         Serial.print("0");
451       }
452       Serial.print(hex_value, HEX);
453     }
454     Serial.println();
455
456     msgI = 0;
457     msgI += MsgReceived[3];
458     msgI = msgI << 8;
459     msgI += MsgReceived[4];
460     delay(500);
461
462
463     float Current = ((float)msgI) / 100;
464
465     char intensidadeString[8];
466     dtostrf(Current, 1, 2, intensidadeString);
467     client.publish("Intensidade", intensidadeString);
468
469
470     Serial.print("Intesidade:␣");
471     Serial.print(Current, 2);
472     Serial.println("␣A");
473     Serial.println("-----------------------------");
474   }
475
```

```
476   //--------------------------[/LER INTENSISDADE
          ]-------------------------//
477
478   //--------------------------[LER TENS O
          ]-------------------------//
479
480   //Start Timer
481   delay(start_timer);
482
483   //Write Enable
484   digitalWrite(RE_DE, HIGH);
485   Serial.println("WRITE:␣01␣03␣00␣CA␣00␣01␣A4␣34␣");
486
487   //Starting Address Hi - 0x00
488   MSGtoMITSUBISHI[2] = 0x00;
489   //Starting Address Lo - 0xCA
490   MSGtoMITSUBISHI[3] = 0xCA;
491   //Error Check Lo (CRC) - 0x15
492   MSGtoMITSUBISHI[6] = 0xA4;
493   //Error Check Hi (CRC) - 0xC9
494   MSGtoMITSUBISHI[7] = 0x34;
495
496   MITSUBISHI.write(MSGtoMITSUBISHI, sizeof(MSGtoMITSUBISHI));
497   delayMicroseconds(us_to_wait);
498
499   //Read Enable
500   digitalWrite(RE_DE, LOW);
501   Serial.print("READ:");
502
503   //End Timer
504   delayMicroseconds(end_timer);
505
506   if (MITSUBISHI.available() >= sizeof(MsgReceived)) {
507     for (int i = 0; i < sizeof(MSGtoESP32); i++) {
508       MSGtoESP32[i] = MITSUBISHI.read();
509       if ((MSGtoESP32[i - 1] == MSGtoMITSUBISHI[0]) && (
              MSGtoESP32[i] == MSGtoMITSUBISHI[1])) {
510         start_index = i - 1;
511       }
512     }
513
514     for (int i = start_index; i < sizeof(MsgReceived); i++) {
515       MsgReceived[i - start_index] = MSGtoESP32[i];
516       Serial.print("␣");
517       int hex_value = MSGtoESP32[i];
518       if (hex_value < 0x10) {
519         Serial.print("0");
```

```
520       }
521       Serial.print(hex_value, HEX);
522     }
523     Serial.println();
524
525     msgT = 0;
526     msgT += MsgReceived[3];
527     msgT = msgT << 8;
528     msgT += MsgReceived[4];
529     delay(500);
530
531     float Voltage = ((float)msgT) / 10;
532
533     char voltString[8];
534     dtostrf(Voltage, 1, 2, voltString);
535     client.publish("Tens o", voltString);
536     Serial.println(msgT);
537
538     Serial.print("Tens o:␣");
539     Serial.print(Voltage, 1);
540     Serial.println("␣V");
541     Serial.println("------------------------------");
542   }
543
544   //--------------------------[/LER TENS O
        ]----------------------- //
545
546
547   // --------------------[LEITURA DE MENSAGENS SERIAL DO
        REGISTO 0x000D]----------------------- //
548
549   //Start Timer
550   delay(start_timer);
551
552   // delay(500);
553
554
555 while (Serial.available() > 0) {
556
557         c = (char)Serial.read();  // READ FROM SERIAL MONITOR
558         texto += c;
559
560         //    if (c == ":")
561         //      break;
562         //    else
563         //      texto += c; // texto = texto + c;
564
```

```
565         ID = texto.substring(2, 3);             // pega o
               n mero de ID enviado pela serial
566         //REGISTER = texto.substring(3, 7);      // pega o
               endere o da var avel que recebera a escrita ex: No
                inv delta ms300 a control word    2000H
567         VALUE = texto.substring(3, 7);          // pega o valor q
                ser  escrito na vari vel Ex: o valor 0002H coloca
                o inv em RUN
568         deviceID = 0;
569
570   } // while Serial.Available
571
572
573     switch (deviceID) {
574
575       // Device ID imagin rio - Serve apenas para enviar
             comandos ao Inversor via RS485
576       case 0:
577
578         node.begin(getInt(ID), Serial2);
579         resultMain = node.writeSingleRegister(0x000D, getInt(
               VALUE)); // para o inv delta a controlword    2000H
               = 8192 decimal   value = 0002 run, = 0001 stop
580
581         deviceID = 1;
582         texto = "";
583         ID = "";
584         REGISTER = "";
585         VALUE = "";
586
587
588         if (resultMain == node.ku8MBSuccess) {
589
590           Serial.println("O comando foi mandado ao variador.");
591
592         } else {
593
594           Serial.println("ERRO");
595           Serial.println("Est  a parar no 1   case");
596         }
597
598         delay(500);
599         break;
600
601     } // switch
602
603
```

```
604 } //loop
605
606
607 // Fun  o que realiza a convers o de um caracter em ASCII
      para numero inteiro
608 int getInt(String texto) {
609
610   int temp = texto.length() + 1;
611   char buffer[temp];
612   texto.toCharArray(buffer, temp);
613   int num = atoi(buffer);
614   return num;
615
616 }
```