Instituto Politécnico de Tomar

Escola Superior de Tecnologia de Tomar

David Sérgio de Oliveira Peneireiro

RenPAD

Relatório de Estágio

Orientado por:

Prof. Doutor Paulo Coelho – Instituto Politécnico de Tomar

Eng. Carlos Maia – Renova FPA S.A.

Relatório de Estágio

apresentado ao Instituto Politécnico de Tomar

para cumprimento dos requisitos necessários

à obtenção do grau de Mestre

em Controlo e Eletrónica Industrial

.

Resumo

Mediante a necessidade de monitorização e supervisão da temperatura em determinados ambientes fabris, nomeadamente nas salas de servidores, e após aturada pesquisa no mercado por dispositivos que cumprissem com os requisitos de precisão e funcionamento requeridos, resolveu a Renova S.A., recorrendo ao seu departamento de Eletrónica, desenvolver um produto de raiz para monitorizar e supervisionar a temperatura que não somente respeitasse e cumprisse com os requisitos e modo de funcionamento pretendidos, mas também que fosse um produto de fácil evolução e o mais abrangente possível, conforme vem sendo característica de todos os produtos por si desenvolvidos.

Assim nasceu o projeto RenPAD, que pode ser dividido em três grandes etapas distintas de desenvolvimento:

- Uma placa de aquisição de dados com a marca Renova Eletrónica (*hardware*+*software*), com comunicação *Ethernet* para recolha dos dados e atualização do *software* e que disponibiliza um conjunto alargado de possibilidades de conexão, com oito entradas analógicas com conversor analógico/digital de 12*bit's* (onde se podem ligar, por exemplo, sensores LM60 ou outros), oito entradas digitais, oito saídas digitais, dois sensores digitais de temperatura e humidade SHT75 e um barramento de oito sensores de temperatura TMP75. Todas as entradas poderão atuar diversos tipos de alarmes autonomamente e a configuração é realizada com recursos a comandos *TCP-IP*.
- *Software* de configuração e monitorização das várias placas RenPAD.
- Serviço de recolha de dados das várias placas RenPAD e seu armazenamento.

Palavras-chave: RCM3700, Sensores, *DataLogger*, c#, DynamicC, MicroC/OS-II, microprocessador, multi-threading, Serviço Windows

Abstract

Upon the need for monitoring and supervising the temperature in certain manufacturing environments, particularly in server rooms, and after thorough research on the market for devices that comply with the requirements of precision and operation modes, Renova SA has decided, through its Department of Electronics, to develop a new product that could not only fulfill and respect the requirements of temperature monitoring and supervising and its operation modes required, but also would be easy to change, upgradable and as comprehensive as possible product, as it is characteristic of all products there developed.

Thus was born the RenPAD project, which can be divided into three main distinct stages of development:

- A data acquisition board with Renova Eletrónica brand (hardware + software), with *Ethernet* communication for data collection and software upgrade, and providing a wide range of connection possibilities with eight analog inputs by use of 12bit's analog/digital converter (which can connect, for example, LM60 sensors or others), eight digital inputs, eight digital outputs, two temperature and humidity sensors SHT75 and a bus of eight temperature sensors TMP75. All inputs can actuate autonomously various kinds of alarms and its configuration is done using TCP - IP commands.
- Configuration and monitoring multiple RenPAD devices software.
- Service of data collection and storage from the various RenPAD devices.

Keywords: RCM3700, Sensors, DataLogger, c#, DynamicC, MicroC/OS-II, microprocessor, multi-threading, Windows Service

Agradecimentos

A todos os que me incentivaram, ajudaram e “esclareceram” nos momentos mais críticos e desmotivantes.

À Renova S.A. e ao meu orientador de estágio, Eng. Carlos Maia, que me têm dado todo o apoio nas diversas iniciativas que tomo.

Ao meu orientador de estágio, do IPT, Prof. Doutor Paulo Coelho não apenas pelo apoio prestado mas também pelas longas horas passadas a ler as inúmeras versões desta tese, para que esta assumisse o rigor e a qualidade apresentada.

À minha esposa que, com muito sacrifício e força de vontade, me ajudou tornar possível esta caminhada.

Aos meus filhos, pelas insubstituíveis horas perdidas.

Índice

Índice de Figuras

Índice de Tabelas

Lista de Abreviaturas e Siglas

AC – *Alternate current*

ACK - *Acknowledge*

ADC - *Analog-to-Digital Converter*

ASCII - *American Standard Code for Information Interchange*

DC – *Direct Current* (Corrente contínua)

DHCP – *Dynamic Host Configuration Protocol*

ECB – *Event Control Block*

E/S – Entrada / Saída

GND – *Ground* (plano de massa em esquemas eletrónicos)

HAL – *Hot Air (Solder) Leveling*

I²C – *Inter-Integrated Circuit*

IDE – *Integrated Development Environment*

ISR – *Interrupt Service* Request

LSB – *Less Significant Byte*

Mosfet - *Metal Oxide Semiconductor Field Effect Transistor*

MSB – *Most Significant Byte*

NACK – *Negative Acknowledge*

OTA – *Over The Air*

OTP – *One Time Programmable*

Ram – *Random access memory*

RH – *Relative Humidity*

ROHS – *Restriction of Certain Hazardous Substances*

Rom – *Read only memory*

RS-232 – *Recommended Standard 232*.

RTOS – *Real Time Operating System*

RX – *Receive/Receiver/*Reception

SPI – *Serial Peripheral Interface*

STR – Sistema de Tempo Real

TCP-IP – *Transmission Control Protocol – Internet Protocol*

TRDN – Terminal de Recolha de Dados Novo

TX – *Transmit/ Transmitter/T*ransmission

UPD – *User Datagram Protocol*

VAIF – Valor de Alarme Inferior

VASP – Valor de Alarme Superior

V_{BE} – *Base-Emitter Voltage*

V_{CC} – *Colector-Colector Voltage* ou *Voltage at the Comon Collector*

V_{CE} – *Collector-Emitter Voltage*

V_{DD} – *Drain-Drain Voltage*

V_{EB} – *Emmiter-Base Voltage*

VHIF – Valor de Histerese Inferior

VHSP – Valor de Histerese Superior

1 – Entidade Acolhedora

1.1 – Apresentação da Empresa

A RENOVA – Fábrica de Papel do Almonda, S.A., é uma empresa portuguesa de capital privado, constituída em 1939, com sede em Zibreira, concelho de Torres Novas, que desenvolve a sua atividade na produção e comercialização de produtos de consumo, com as seguintes classificações de atividade:

- Fabricação de artigos de papel para uso doméstico e sanitário (CAE: 17220; NACE 17.22);

- Fabricação de papel e cartão (exceto canelado) (CAE 17120: NACE 12.12).

Labora num regime de funcionamento em operação contínua em três turnos. No final de 2013 o número de colaboradores rondava os 587. [1]

É uma empresa/marca, onde o ambiente, a segurança, a qualidade e a inovação estão nas suas preocupações.

No que respeita à política ambiental, a Renova está bem posicionada em relação às suas congéneres a nível europeu. Em 1999, a Renova foi a primeira empresa do seu sector de atividade a obter a certificação Ambiental, de acordo com o referencial ISO14001, e em 2004, a certificação EMAS (Sistema de Ecogestão e Auditoria da União Europeia).

A segurança das operações e o "bem estar "dos trabalhadores nos locais de trabalho, levou a Renova em 2004, a obter uma certificação de Segurança e Saúde no trabalho, segundo a norma OHSAS 18001. Também em 2004, a Renova recebe o certificado de Gestão da Qualidade, ISO 9001:2000, ISO 17025.

Em 2007, surgem para a Renova novos desafios que a levaram a obter mais duas certificações: uma para Segurança Alimentar, de acordo com o referencial BRC/IoP, e outra em Sistema de Investigação, Desenvolvimento e Inovação, de acordo com a NP 4457 (2007, relativo à Gestão da Investigação, Desenvolvimento e Inovação).

No contexto económico e social, a Renova é atualmente em Portugal líder de mercado em todos os produtos de papel *tissue*. Em Espanha é líder no segmento dos guardanapos e também está presente na França, Bélgica e Luxemburgo. A sua faturação em 2008 foi de 130 milhões de euros. [4]

1.2 – Localização da Empresa

A Renova possui duas unidades industriais, uma situada na nascente do Rio Almonda (designada por Fábrica 1), e outra a cerca de dois quilómetros de distância deste local (designada por Fábrica 2).

Situada no centro do país, com morada em:

Renova - Fábrica de Papel do Almonda, S.A, Zibreira

2354-001 Torres Novas – Portugal

a Renova tem ainda escritórios em Espanha, França, Luxemburgo, Bélgica e Canadá exportando os seus produtos para mais de 140 países.

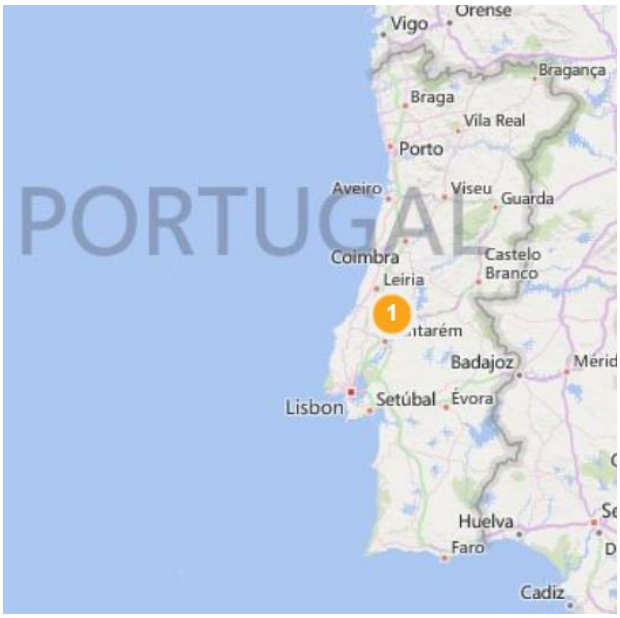


Figura 1: Localização da Empresa

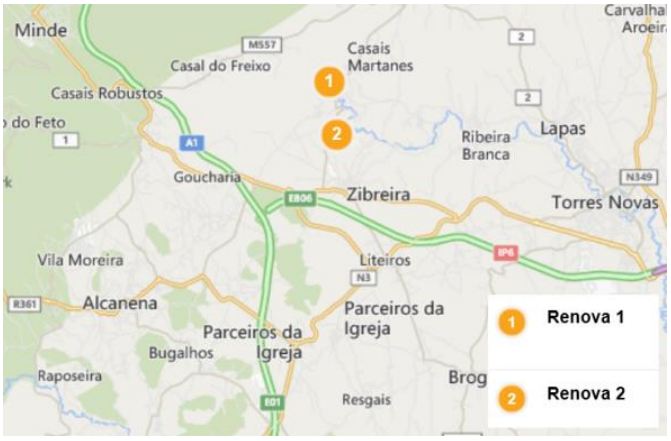


Figura 2: Localização detalhada da Empresa

Figura 3: Vista aérea Fábrica 1

Figura 4: Vista aérea Fábrica 2

1.3 – História da Empresa

Dedicando-se numa fase inicial à produção de papel de embalagem, escrita e impressão, alguns anos mais tarde a Renova entrou numa área que viria a condicionar todo o seu futuro: os produtos de papel de uso doméstico e sanitário (*tissue*).

A crescente especialização na fabricação do papel *tissue* e os grandes investimentos em infraestruturas culminaram na construção de uma nova fábrica, o que ampliou substancialmente o alcance da Renova e dos seus produtos. Esta altura é também o início de uma filosofia de total envolvência com a natureza.

A qualidade Renova abriu-lhe as portas para o mundo. Num ambiente cada vez mais competitivo, uma profusão de mensagens distintas começa a proliferar no mercado: é o início do *Marketing*. A Renova compreendeu que a consciência das diferenças individuais cresceu nesta altura, tal como as possíveis escolhas dos cidadãos.

É dado o primeiro passo para a internacionalização com a criação da Renova España, S.A. Para fazer face a esta estratégia de internacionalização são feitos grandes investimentos em infraestruturas, equipamentos e tecnologia.

Numa altura em que era usual, mesmo publicitariamente, anunciar a “bondade” dos produtos não reciclados, a Renova decidiu investir numa unidade de Reciclagem (Divisão de Reciclagem).

Para além dos interesses ecológicos e económico o investimento também corresponde a uma integração industrial a montante, importante vantagem estratégica para a empresa.

Ao aumento da consciência ambiental na Europa e no Mundo, a Renova responde posicionando-se na fila da frente e tornando-se numa referência europeia, indo mais longe que a legislação e manifestando a sua preocupação e respeito pela preservação dos recursos naturais do nosso planeta.

Com a definição de uma estratégia verdadeiramente ibérica e com a modernização da organização a Renova decide-se a elevar o Bem – Estar à categoria máxima: Para um novo Bem – Estar do corpo do espírito e dos sentidos.

A Renova prossegue a sua estratégia de internacionalização com a entrada no mercado Francês. Dando continuidade à sua estratégia de aposta em mercados inovadores, dinâmicos e competitivos, a Renova lança-se passados alguns anos no mercado Belga.

Em 2005 a Renova faz um lançamento inesperado: o primeiro papel higiénico preto do mundo, no Salão "*Maison&Object*", em Paris.

Embora a intenção inicial fosse afirmar a diferença e carácter inovador num produto icónico, ao condensar num só objeto várias características chave da personalidade da marca Renova (qualidade, segurança, perfume, cor, embalagem), a fama do produto motivou um subsequente alargamento de gama e, sobretudo, uma rápida expansão da notoriedade da marca no mundo, com impacto positivo nos negócios internacionais da empresa.

Em 2007 a Renova empenhou-se a fundo no lançamento de uma gama alargada de produtos amigos do ambiente – Renovagreen. Produtos fabricados a partir de materiais reciclados e produzidos sem concessões para com o ambiente.

Uma maior aproximação aos cidadãos também tem sido a tónica dos tempos recentes, seja pela via da aposta na presença nas redes sociais e no negócio *online*, seja pelo investimento na capacidade de fabricar produtos personalizados.

A maior aproximação aos cidadãos está também patente na criatividade com que a Renova tem diversificado os seus canais comerciais, apostando nas "Boutiques Renova"

inseridas em espaços comerciais de sucesso seguindo o princípio "*Pop Up Store*" ou até através da presença em espaços como o museu do Louvre em Paris. [1]

Referência a algumas das datas mais importantes:

- 2014 - Lançamento da nova geração de produtos de papel colorido, Renova Red Label, em todas as categorias, desde o papel higiénico, rolos de cozinha, aos lenços de bolso.
- 2013 - Entrada na maior cadeia de supermercados do mundo, a *Wallmart*.
- 2012 - Lançamento da linha de guardanapos de mesa personalizados. Com vinte guardanapos por pacote, cuja face e verso de cada guardanapo pode ser personalizados individualmente com qualquer motivo à escolha do cliente.
- 2009 - Lançamento da gama Renova design, uma linha de produtos com temas diversos e bastante coloridos.
- 2008 - Entrada em funcionamento da loja *on-line*.
- 2007 - Lançamento da gama de produtos ecológicos "Renova Green" e do despenseiro de papel Renova GoldBox: uma peça de joalharia fabricada à mão, em ouro de 24 Quilates, cujo logotipo Renova sobressai com recurso à incrustação de 148 diamantes, concebida para celebrar a venda de um milhão de rolos da gama Black. A Renova alarga as suas vendas com mais de 50 mercados internacionais, com 40% a 50% das vendas fora do seu mercado doméstico. Neste ano a Renova obteve também mais duas certificações: uma para Segurança Alimentar, de acordo com o referencial BRC/IoP, e outra em Sistema de Investigação, Desenvolvimento e Inovação, de acordo com a NP 4457 (2007, relativo à Gestão da Investigação, Desenvolvimento e Inovação).
- 2006 - Pela primeira vez, um produto Renova é alvo de crítica pela maioria dos jornais e revistas de moda em todo o mundo.
- 2005 - Depois do lançamento de Renova Black, o primeiro papel higiénico preto de sempre, novos canais comerciais de todo o mundo mostraram interesse no produto.
- 2004 - Renova Bélgica inicia suas operações na Bélgica e em Lisboa. Neste ano a Renova também obtém a certificação de Segurança e Saúde no Trabalho OHSAS 18001, a certificação EMAS (Sistema de Ecogestão e Auditoria da União Europeia)

e as certificações de gestão de qualidade segundo as normas ISO 9001:2000 e ISO 17025.

- 2003 - Lançamento da primeira linha de papel higiénico humedecido.
- 2002 - Renova França inicia operações em França, utilizando o inovador papel higiênico "Fresh & Clean", como principal motor de penetração no mercado.
- 1999 - Concedida a certificação ISO 14001, para cumprimento da legislação ambiental. A Renova SA torna-se na primeira empresa com tal certificação em Espanha e Portugal.
- 1998 - Lançamento do primeiro papel higiênico no mundo integrando micro-gotículas de creme hidratante ("Renova Fresh & Clean").
- 1995 - Alteração da política de *branding* da empresa, com a marca "Renova" a surgir associada a todos os produtos produzidos.
- 1990 - Renova España SA inicia operações em Espanha.
- 1989 - Lançamento da primeira gama completa de produtos associados ("Renova Class").
- 1979 - Inauguração da Fábrica 2.
- 1970 - Lançamento da primeira linha de produtos de higiene feminina ("Reglex").
- 1961 - Modificação a estrutura de negócios: O papel de escritório deixa de ser o core-business da empresa, sendo substituído pelo papel descartável para uso doméstico e sanitário.
- 1958 - A empresa lança o rolo de papel higiénico "Renova Super", o produto mais vendido de sempre.
- 1950 - Aquisição de uma linha de alta tensão dedicada, complementando a principal fonte de energia da fábrica: o rio.
- 1943 - Um novo grupo de acionistas toma conta da empresa. Renova - Fábrica de Papel do Almonda, SA nasce.
- 1939 - Fundação da empresa privada Fábrica de Papel do Almonda Lda.
- 1818 - David Ardisson escolheu a marca Renova como a marca d'água para a primeira folha de papel fabricado nas margens do rio Almonda, Torres Novas.

[2][4]

1.4 – Portfólio de Produtos

A Renova produz uma vasta gama de produtos na área do papel e papel *tissue*.

Desde o papel Vegetal ao papel de embalagem, passando por papel de escrita e impressão, toalhas de mesa, papel de uso doméstico e higiénico… a variedade é imensa. Existe no entanto uma panóplia de produtos que considero icónicos do ponto de vista de evolução tecnológica e que contribuíram grandemente para o carisma que a marca Renova transmite, e que passo a apresentar de forma sucinta.

1.4.1 – Renova Super

Lançado em 1958, é o produto mais vendido de sempre da marca.

Figura 5: Renova Super [3]

1.4.2 – Renova Fresh & Clean

Lançado em 1998, o Renova Fresh & Clean é o primeiro papel higiénico perfumado no mundo que integra micro-gotículas de creme hidratante na sua matriz.

Figura 6: Renova Fresh & Clean [3]

1.4.3 – Renova Black

Lançado em 2005, este foi o primeiro papel higiénico preto de sempre. A sua criação foi considerada um arrojo: um produto genial em termos de criativos. Alterou a perspetiva de um produto usualmente envergonhado passando a considerar-se um artigo de decoração requintado.

É o embaixador mais importante da Renova, sendo criticado em todas as revistas da especialidade e moda do mundo inteiro, elevando o valor da marca Renova fazendo-a reconhecida e apetecida em países de todos os pontos do mundo.

Depois do seu lançamento surgiram mais algumas variações de cores distintas e vibrantes que colmataram a gama Renova Black.

Também do ponto de vista técnico, este papel e o seu processo de fabrico são considerados revolucionários e a sua criação foi um enorme desafio, uma vez que não se trata apena de tingir o papel com uma cor: trata-se de lhe dar uma cor vibrante, que não debote, que respeite a pele do consumidor, e cujo processo de fabrico respeite as restritivas normas ambientais a que a renova se compromete, bem como as que estão definidas pela certificação ambiental ISO14001.

Figura 7: Renova Black – Colors [3]

1.4.4 – Renova Green

Lançada em 2007, esta é uma gama de produtos ecológicos em que tanto o papel como a embalagem são constituídos de material 100% reciclado.

Esta gama de produtos foi concebida, e transmite a o consumidor mais interessado, a preocupação ambiental da empresa e a sua tentativa de diminuir a pegada ecológica de cada um.

Figura 8: Renova Green [3]

1.4.5 – Renova Design

Mais um produto concebido com vista a alterar o paradigma do papel *tissue*. Desta vez com desenhos apelativos, tornando um artigo banal num artigo decorativo.

Figura 9: Renova Design [3]

1.4.5 – RenovaPrinte

Papel de impressão 100% reciclado de excelente qualidade, brancura e lisura. Boa opacidade permitindo impressões de grande qualidade.

Figura 10: RenovaPrinte [3]

1.5 – Funcionamento da Empresa – Processo Produtivo

O processo produtivo na RENOVA encontra repartido por cinco divisões:

- Divisão de Reciclagem (DIRE), onde o papel velho é transformado em fibra nas condições de utilização;
- Divisão de Fabricação (DIFA), onde se processa a fabricação da folha de papel;
- Divisão de Transformação (DITA), onde o papel é transformado num conjunto diversificado de produtos de utilização doméstica, sanitária e industrial;
- Divisão de Produtos Sanitários (DISA), sector especializado na produção de proteções sanitárias femininas;
- Divisão de Produtos Personalizados (DIPE), onde são produzidos os produtos personalizados (produtos totalmente impressos a várias cores).

A Fábrica 1 possui uma parte da Divisão de Fabricação (uma máquina de papel *tissue* e duas de papel de impressão e escrita 100% reciclado e de papel *craft*). Nesta unidade encontra-se também a DISA e DIPE. A Fábrica 2 integra a Divisão de Reciclagem, parte da Divisão de Fabricação (duas máquinas de produção de *tissue*) e a Divisão de Transformação. A figura 11 ilustra o processo produtivo da Renova. [4]

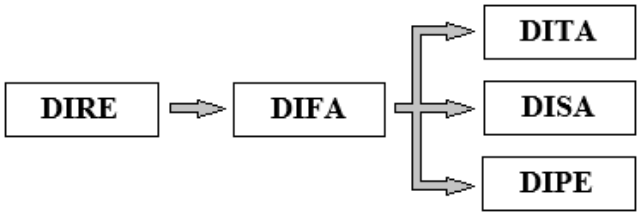


Figura 11: Fluxo Produtivo [4]

1.5.1 – Divisão de Reciclagem – DIRE

O objetivo da DIRE é obter fibras recicladas de elevada qualidade, partindo de "papéis velhos" selecionados. A reciclagem consiste em retirar do "papel velho" toda a matéria não fibrosa – aditivos, cargas, tintas e outros contaminantes resultantes da utilização do papel. Esta matéria é eliminada por rejeição sequencial utilizando conjuntamente quarto processos distintos: hidrociclonagem, crivagem, lavagem e flutuação. Os referidos processos baseiam-se nas propriedades físico-químicas que diferenciam os contaminantes das fibras, como sejam a forma e tamanho, a densidade e a afinidade electroestática. São também realizadas duas etapas de branqueamento (oxidativo e redutor) que permitem aumentar e uniformizar a brancura da pasta reciclada. [4]

A Divisão de Reciclagem ocupa-se ainda da gestão do tratamento de efluentes da Fábrica 2 (ETAR2) e da gestão do Aterro Controlado de Resíduos Industriais (ACR).

Com uma capacidade de 35000 ton/ano, a DIRE, é responsável pela produção de cerca de 50% da matéria-prima fibrosa, utilizada na Renova, para produção de papel.

Figura 12: Reciclagem [5]

Figura 13: Tanque Biológico – ETAR2 [6]

1.5.2 – Divisão de Fabricação – DIFA

A DIFA utiliza como matérias-primas principais a pasta reciclada (fornecida pela DIRE) e pasta virgem (adquirida no exterior), tem como objetivo a produção de papel da Renova.

Como referido anteriormente, a Fábrica 1 possui uma parte da DIFA (uma máquina de papel *tissue* e duas de papel de impressão e escrita 100% reciclado e de papel *craft*). A DIFA da Fábrica 2 dispõe de duas máquinas de produção de papel *tissue*.

O papel *tissue* é um papel de toque muito suave, flexível, de alto grau de maciez e absorção, propriedades conseguidas através de um processo de formação específico, que assenta basicamente na escolha criteriosa de componentes e na formação de microondulações paralelas e em direção transversal à linha de produção (crepe). As ondulações são formadas por ação de uma lâmina colocada estrategicamente de modo a retirar o papel do cilindro secador com o efeito desejado.

Este tipo de papel tem usos diversos, e embora predomine o uso doméstico e sanitário (lenços, guardanapos, toalhas e papel higiénico), podem também ser usados para embalagem e em filtros, dada a sua maciez e permeabilidade. A sua fabricação pode acontecer em qualquer das duas máquinas, podendo dividir-se em quatro etapas: preparação da pasta, formação da folha, secagem e formação do crepe.

Por preparação da pasta designa-se todo o conjunto de operações que antecedem a folha e que começam na desintegração da pasta seca e acabam na entrada da máquina. A etapa de formação da folha tem início na caixa de chegada da máquina, que deposita numa teia esgotante um jato de suspensão contínuo e de secção retangular, que ao longo do seu percurso vai eliminando a água através de força centrífuga, vácuo, prensagem e secagem. A fabricação termina com a bobinagem simples, ou de várias folhas para bobinas de tamanho normalizado. [4]

A DIFA é composta por Áreas, agrupando linhas de produção, que são geridas por um Gestor de Área que reporta ao respetivo Gestor de Serviços.

Destas, a Área 1 encontra-se especializada na produção de papéis de impressão, escrita e embalagem, enquanto as restantes (Área 4, Área 5 e Área 6) estão especializadas na produção de papel *tissue*.

Também da responsabilidade da DIFA está a gestão do tratamento de efluentes da Fábrica 1 (ETAR1).

Figura 14: Máquina de Papel [7]

1.5.3 – Divisões de Transformação – DITA, DISA, DIPE

Utilizando como matéria-prima principal o papel produzido pela DIFA, as Divisões de Transformação (DITA, DISA e DIPE) têm como objetivo a produção de "transformados" de papel, principalmente produtos fabricados com base em papel *tissue*.

A atividade das Divisões de Transformação está atualmente dividida em três vetores convergentes, aqui designados por áreas de transformação:

- Dobras
- Rolos Multiusos
- Rolos de papel Higiénico

Todas as linhas destas Divisões recebem o papel em bobinas normalizadas, de acordo com o tipo de produto e de linha, fornecidas diretamente pela Divisão de Fabricação e, parte proveniente do armazém automático de papel. As linhas de produção são muito flexíveis, podendo produzir-se mais do que um produto final em cada uma, com apenas algumas alterações mecânicas de fluxo e/ou matéria-prima.

Genericamente, todas as máquinas produtivas destas divisões são compostas por uma bobinadora ou dobradora e uma embaladora. Se o produto produzido numa determinada linha for ensacado, esta ainda terá uma ensacadora e um robô antropomórfico

como paletizador. Se o produto for encaixotado, a linha terá ainda uma encartonadora (coloca o produto dentro de caixas de cartão), sendo as caixas posteriormente encaminhadas, através de tapetes rolantes, até uma paletização central. [4]

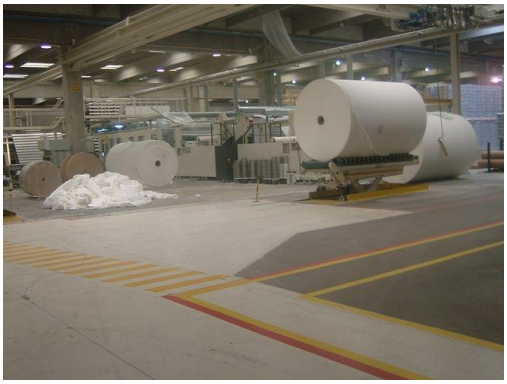


Figura 15: Área de Fabricação [5]

Figura 16: Área de Fabricação [6]

1.5.4 – Métodos de Manutenção

Por norma a manutenção dos equipamentos tem dois níveis:

- Manutenção Preventiva: Existe uma rotina mensal de verificação de uma *check-list* (ordem de trabalho lançada automaticamente pelo sistema de gestão SAP) para avaliar os pontos potencialmente críticos. Com base nos resultados dessa avaliação são realizados trabalhos imediatos de manutenção, ou poderá ficar apenas o registo para que o trabalho seja planeado ou realizado na próxima paragem de verificação (dependente do setor ou peça da máquina, se existem ou não peças de substituição, do tempo previsto de intervenção e da gravidade da situação)

  Da mesma forma, existe uma empresa externa, a quem está entregue um plano periódico de lubrificação.

- Manutenção Corretiva: Ocorre quando uma máquina pára devido a uma avaria. São chamados os serviços de manutenção (mecânica ou elétrica ou ambas), é feito um registo de consignação da linha (para que este fique "oficialmente parada") e os trabalhos são desenvolvidos até a avaria estar solucionada. No final é efetuado o registo de "desconsignação" para "entregar" de novo a máquina aos operadores criando assim um procedimento de segurança que visa salvaguardar tanto a linha como os operadores a ela consignados.

1.6 – Mundo Renova

A Renova FPA SA não é apenas uma empresa de papel. Apesar do seu "*core business*" estar focado na fabricação e transformação de papel *tissue*, existe uma gama imensa de produtos distintos do papel, pertencentes ao universo Renova. Dentre estes vou apenas sublinhar os que mais se evidenciam.

1.6.1 – Pró-mineral – Águas dos Açores

São comercializadas pela Pro-mineral duas marcas distintas de água, provenientes das captações nas povoações de Vale das Furnas e Serra do Trigo, na freguesia de Furnas na ilha vulcânica de São Miguel no arquipélago dos Açores, e que colocam no mercado açoriano cerca de um milhão de litros de água por ano, tendo já sido comercializada em Portugal continental, França e E.U.A. Uma é do tipo água de nascente, a Água Gloria Patri e a outra do tipo gasosa mineral, a Água Magnificat.

1.6.1.1 – Magnificat

"Água jovem", suavemente gasosa, encontra-se no interior rochoso da ilha de S. Miguel. Magnificat é a expressão mais sublime de uma das maiores hidrópoles do mundo... [8]

Apesar de ser captada em ambiente vulcânico possui valores de mineralização muito altos (252mg/l); esta é uma água jovem e fresca com a particularidade de possuir um pH de apenas 4,9. [9]

Esta água foi distinguida com o segundo lugar na categoria de águas naturalmente gasosas no decorrer da Termatália 2008, prova internacional de águas termais, evento anual promovido pela Feira Internacional de Turismo Termal, em Ourense – Espanha, que tem conquistado um estatuto de referência mundial para profissionais do ramo, com participantes de todos os continentes. [10]

Figura 17: Água Magnificat-Açores [8]

1.6.1.2 – Gloria Patri

Esta é uma água lisa, de mineralização débil e um pH de 6,95. Nasce na Serra do Trigo na ilha de São Miguel e conhecem-se as suas propriedades desde o descobrimento do arquipélago dos Açores.

Também esta água foi distinguida com o terceiro lugar na categoria de água de mineralização débil na Termatália 2010 onde estiveram a concurso mais de 15 águas. [10][11]

Figura 18: Água Gloria Patri [8]

1.6.2 – Artigos de higiene íntima feminina - Renova First Silk Sensation

Pensos ultrafinos com elevada capacidade de absorção, com cobertura superior acetinada, que permanece seca para maior conforto; elevada capacidade de absorção, graças à tecnologia de partículas superabsorventes, que permite reter o fluxo e odores nas camadas mais profundas do penso. [12]

Figura 19: Renova First Silk Sensation [12]

1.6.3 – Sistemas de gestão de tempos e controlo de acessos - Renova Eletrónica

A Renova Eletrónica, apesar de ser um departamento da Renova, que tem como responsabilidade desenvolver e manter soluções de *hardware* e *software* para a Renova, tem uma vertente comercial separada do "*core business*" da Renova. Na verdade, algumas das soluções desenvolvidas por este departamento para consumo interno da Renova, são

colocadas à disposição de clientes por um canal comercial próprio e relativamente independente. Desta forma existe uma rentabilização dos recursos alocados a determinado desenvolvimento para consumo interno.

Os expoentes máximos deste aproveitamento e rentabilização do investimento em conhecimento, pesquisa e desenvolvimento para colmatar necessidades internas são as soluções de controlo de ponto e assiduidade e de controlo de acessos. Apesar de existirem outros produtos comercializados pela Renova Eletrónica, a pedido de empresas que de uma forma ou de outra tomam conhecimento do que de melhor aqui se faz, estes não tem expressão face às soluções mencionadas. Assim, tal como o *core business* da Renova é o papel *tissue*, o *core business* da Renova Eletrónica são as soluções supra mencionadas de controlo de ponto e de acessos, tendo atualmente uma carteira com mais de 1200 clientes distintos, comercializando as suas soluções não apenas em todo o território nacional, mas exportando também para os mercados Angolano e Moçambicano.

Foi neste departamento, constituído por uma competente equipa multidisciplinar e multifacetada, sediado na Fábrica 2 da Renova, que decorreu o meu estágio, tendo a solução desenvolvida sido concebida para suprimir necessidades internas, mas contemplando todas características inerentes e indispensáveis para, caso exista procura externa e oportunidade de mercado, poder ser comercializada sem necessidade de alterações ou adaptações.

1.6.3.1 – História da Renova Eletrónica

Com vista a colmatar necessidades internas existentes na Renova, o departamento de manutenção elétrica iniciou na década de 80 o desenvolvimento de equipamentos.

Em meados do ano de 1988, aquando o interesse de aquisição de relógios de ponto eletrónicos para uso interno, com características e modos de funcionamento particulares, constatou a Renova que o mercado dos Pontógrafos Eletrónicos, por se encontrar ainda numa fase muito embrionária, não oferecia soluções comerciais que colmatassem todos os requisitos e funcionalidades pretendidas. Decidiu então, por intermédio do seu departamento de manutenção elétrica, criar um equipamento que correspondesse às suas espectativas e exigências internas.

Passados cerca de dois anos, e mediante solicitação de algumas empresas da zona que, de uma forma ou outra, tomaram conhecimento do desenvolvimento destes equipamentos e das suas funcionalidades, e dado que existiam e tinham sido testados na empresa durante algum tempo com resultados muito positivos, iniciou a Renova a sua comercialização de forma regular.

Dada a qualidade e as características únicas que estes equipamentos ofereciam, rapidamente aumentaram as solicitações externas, engrossando a carteira de clientes.

Surgiu assim a necessidade de criar um departamento autónomo, responsável por esta nova área de negócio, capaz de dar resposta à comercialização dos equipamentos, assistência técnica, desenvolvimento de *software* adequado e adaptado, formação, entre outros. Assim nasce a Renova Eletrónica.

Com a evolução dos produtos cresceu também uma equipa técnica coesa e especializada, capaz de fazer face às mais variadas responsabilidades associadas ao desenvolvimento, instalação e formação destes sistemas.

Em 2003 foi realizada uma parceria com a SAP (passando o Módulo de Salários da solução de *software WebRenWin* a integrar dados diretamente com o *SAP Business One*), que veio trazer a este departamento uma postura mais dinâmica e sólida, com perspetivas de crescimento a médio e longo prazo.

Neste momento, e passados 26 anos da sua criação, a Renova Eletrónica continua a ser uma referência no mercado, conhecida pelas suas soluções inovadoras e abrangentes.

Figura 20: Publicidade aos primeiros equipamentos desenvolvidos e comercializados pela Renova Eletrónica

1.6.3.2 – Principais produtos comercializados

Entre os produtos desenvolvidos e comercializados pela Renova Eletrónica destacam-se os seguintes:

- TRDNv5 – Terminal de recolha de dados com as seguintes características:

    - Quatro entradas multifuncionais, configuráveis uma a uma, para servirem como leitor principal (funcionamento conjunto com as teclas), leitor controlador de acessos (podendo realizar marcações sem necessidade de premir qualquer tecla, com opção de atuar um relé), leitor de dados através de dispositivos externos (tais como *Scannet, etc*.), podendo ainda, em casos particulares utilizar-se uma das portas para criar uma mini-rede de terminais (com um *Master* e vários *Slaves* onde apenas o *Master* está ligado à LAN). Todas as portas podem funcionar com leitores magnéticos, de aproximação ou de impressão digital
    - Quatro relés para uso diversificado (sirene, controlo de acessos, *etc*) atuados a partir de uma tecla, leitor ou ainda automaticamente a partir de uma tabela de horários configuráveis pelo administrador (para atuar por exemplo uma sirene);
    - Teclado com 5 teclas de função totalmente configuráveis (como tecla de entrada, saída, consulta de dados, justificações… necessita de introdução de senha, qual o relé a atuar *etc*.) e 12 teclas de entrada de dados para introdução de dados (justificações, senha, *etc*.).
    - Mostrador de 20x4 caracteres, retro-iluminado e *beep* para *interface* com o utilizador
    - Bateria interna recarregável que, em caso de falha de energia mantém o terminal ativo por mais de quatro horas, com aviso ao processador para um *Power-Off* controlado caso a sua carga atinja o mínimo aceitável
    - Bateria de *Backup* para evitar a perca de dados caso a bateria de alimentação esgote. Quando à plena carga, esta bateria assegura uma proteção contra perda de dados durante cerca de quatro anos.

- 512Kb de memória Flash para a aplicação e tabelas de configuração e, 512Kb de memória Ram não volátil (por meio da bateria de *backup*) para armazenamento de dados, que, dependendo do espaço ocupado, inerente à aplicação, pode armazenar dados a uma taxa de 4 marcações*100 empregados por dia durante cerca de um mês
- Placa de rede de 10Mbps
- Integração total com o *software* de Gestão WebRenWin
- Este terminal tem como principais funcionalidades:

    - Registo de tempos dos funcionários (marcações de entrada/saída)
    - Consultas de saldos nos terminais pelos funcionários
    - Marcação de saídas em serviço
    - Marcações especiais (refeições, outras)
    - Marcações de imputações de tempos a obras
    - Controlo de acessos com acionamento de trincos elétricos ou ligação a barreiras de parques de estacionamento
    - Entre outras

Figura 21: TRDNv5

- WebRenWin

  - O WebRenWin é um programa de controlo de ponto em ambiente cliente-servidor, baseado em *Intranet*, e com uma base de dados relacional SQL.
  - Existem diversos módulos que podem ser adquiridos em conjunto com o WebRenWin, o que permite que a aplicação seja adequada às necessidades de cada cliente.

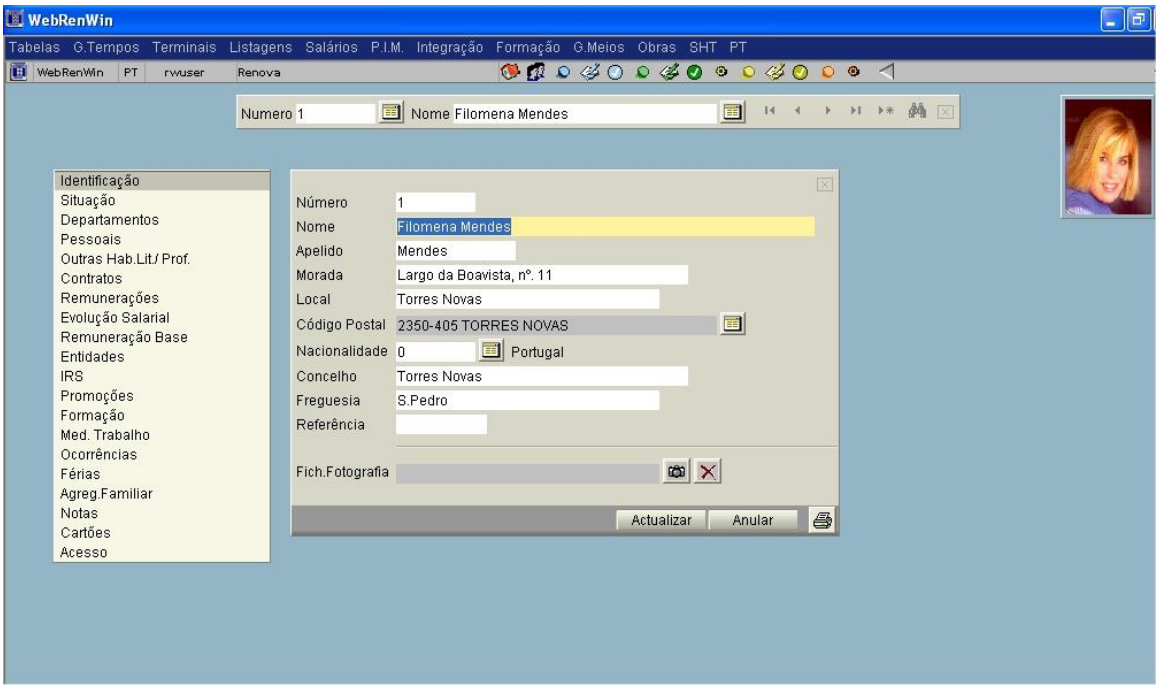


Figura 22: WenRenWin [13]

  - Módulos Disponíveis**:**

    - OBRAS - Através deste módulo qualquer empresa poderá obter dados dos tempos efetuados em cada obra. São ainda possíveis de gerir toda uma série de fluxos como operações, ordens de fabrico, máquinas e linhas de produção por exemplo. Esta contagem de tempos é feita através da digitação nos terminais de ponto ou através

da leitura de códigos de barra. Existe também a possibilidade de orçamentar e saber a qualquer momento os respetivos desvios.

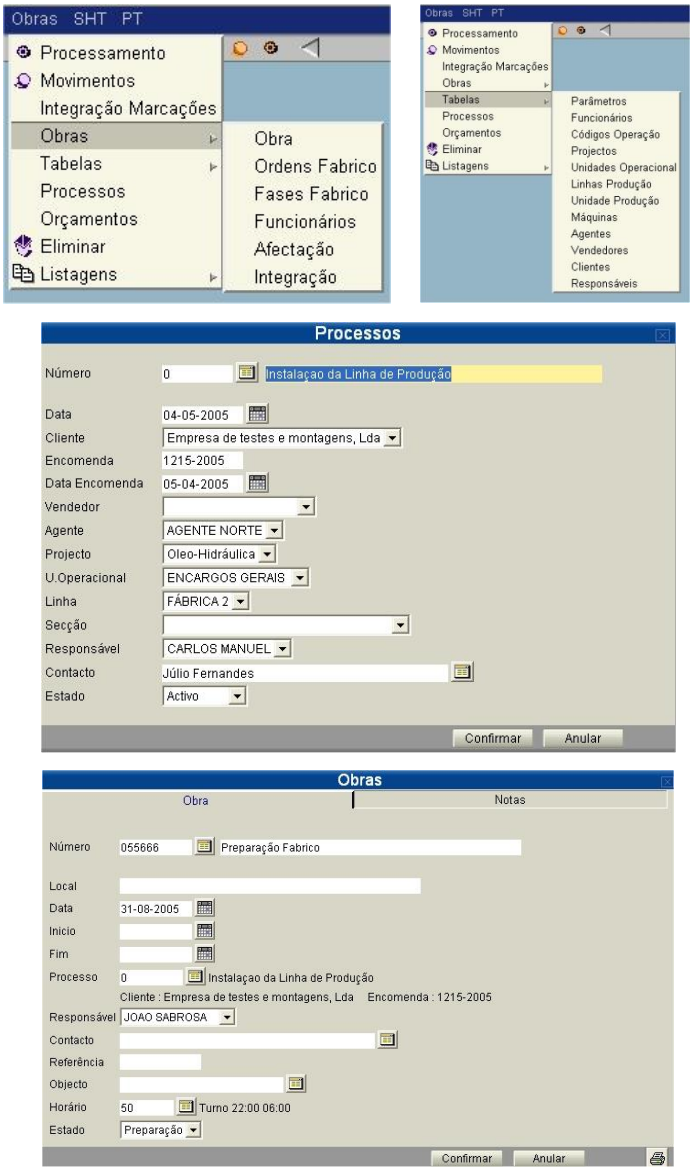


Figura 23: WebRenWin - Módulo de Obras [13]

- FORMAÇÃO - O módulo de formação, tal como o próprio nome indica, destina-se a planear todas as ações de formação a decorrer em determinado ano. Permite a criação de ações com os respetivos módulos, a criação de turmas para cada uma das ações ou módulos, criação de horários, critérios de avaliação, bem como o controlo de presenças. Através deste planeamento é possível emitir vários relatórios para análise.

Figura 24: WebRenWin - Módulo de Formação [13]

- HIGIENE E SEGURANÇA NO TRABALHO - Com este módulo pretende-se permitir uma gestão eficaz e prática dos SHST (Serviços de Higiene e Segurança no Trabalho). Torna-se assim possível gerir diversos aspetos relacionados com os acidentes de trabalho.

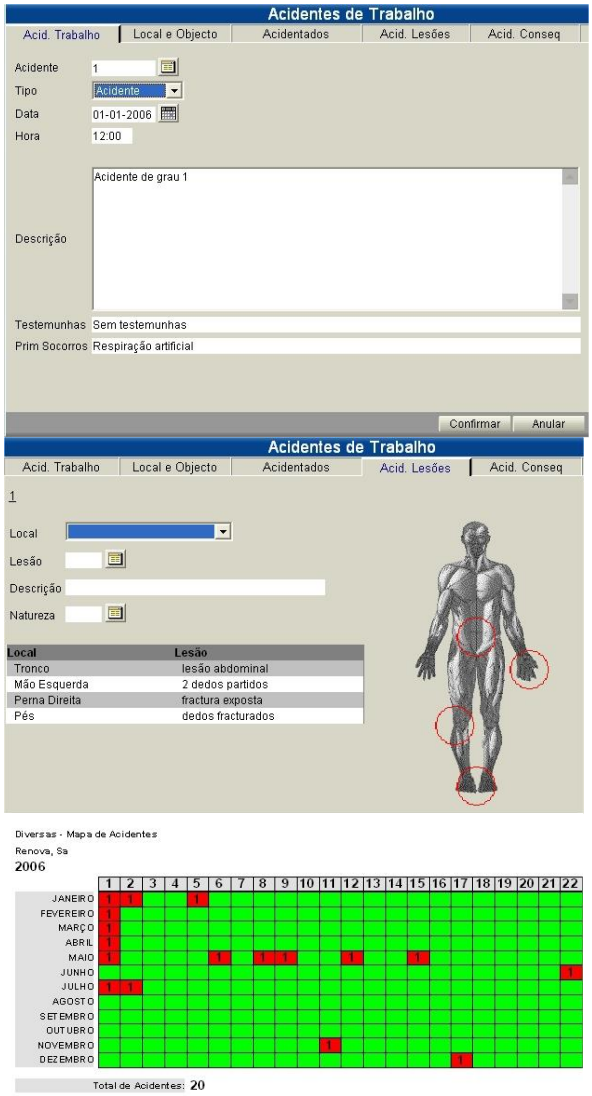


Figura 25: WebRenWin - Módulo de higiene e segurança no trabalho [13]

- SALÁRIOS - Com este módulo torna-se possível tratar todas as obrigações fiscais e legais exigidas no processamento de salários. O utilizador tem a possibilidade de ele próprio, com toda a facilidade, processar as remunerações dos funcionários da empresa.

  Principais funcionalidades:

  - Processamento de salários para trabalhadores dependentes e independentes.
  - Permite fazer vários processamentos diferentes para o mesmo período de referência.

- Grande flexibilidade e produtividade no registo das alterações mensais - faltas, horas extra, remunerações e descontos.
- Anulação e repetição de processamentos.
- Permite a escolha do modelo de recibo a utilizar.
- Mapas legais (Seg. Social, IRS, Mapa Seguro, Sindicato, Relatório Único, Modelo 10, entre outros).
- Envio de recibos por correio eletrónico.
- Alertas de Fins de Contrato, Fins de Períodos Experimentais e Término de Prazo de Avisos Prévios.
- Gestão do cadastro de pessoal.
- Contabilização dos movimentos processados, classes, por distribuição e provisões de custos

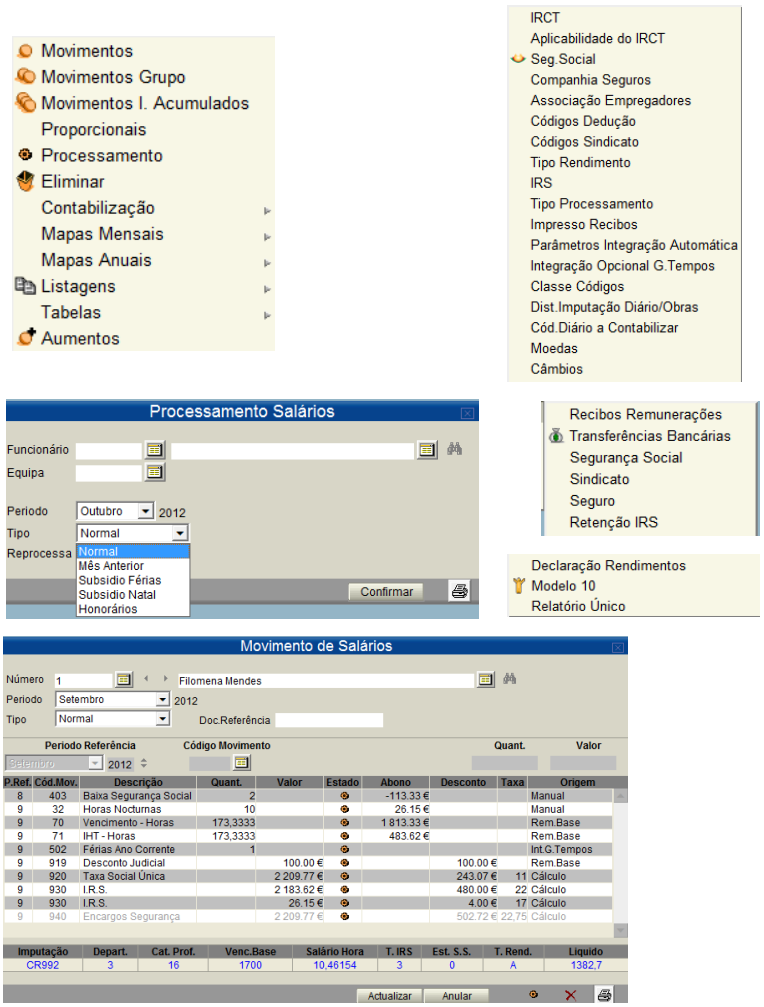


Figura 26: WebRenWin – Módulo de Salários [13]

- GESTÃO DE MEIOS – Este módulo permite de forma fácil controlar todo o tipo de equipamentos entregues a cada funcionário, tais como: fardamento, equipamento de proteção individual, telemóveis, ferramentas…
- PROCESSAMENTO INDIVIDUAL DE MARCAÇÕES – Este módulo gere marcações individualmente, tratando-as de forma isolada. É utilizado por exemplo, para marcação de refeições.
- Existem ainda outros módulos específicos de integração com outros *softwares*, recolhas automáticas, de gráficos…

- O WebRenWin é uma poderosa ferramenta que ajuda as empresas a administrar e avaliar dados, bem como a simplificar/desburocratizar os processos.
  A aplicação de Gestão de Tempos é altamente parametrizável, e a sua operacionalidade consiste no relacionamento entre todas as tabelas, por forma a permitir o correto funcionamento da aplicação.
  O *software* foi desenvolvido em Base de Dados *SQL Server* funcionando em Ambiente WINDOWS, dotado de um amigável *interface web*.
  [13]

2 – Projeto RenPAD - Conceito

A suportar todo o trafego de informação gerada pela enorme estrutura apresentada no capítulo anterior, tanto em termos de dados de fabricação, de vendas, de marketing, controlo de qualidade, gestão energética, gestão ambiental, *etc*… existe um grande investimento em meios informáticos, cujo coração assenta em duas grandes salas de servidores.

Figura 27: Uma das salas de servidores da Renova.

Cada vez mais as últimas gerações de *hardware* baseado em *racks*, como servidores *Web*, equipamentos de telecomunicações, *hardware* de interligação de redes, *hardware* de armazenamento e salvaguarda de informação, *etc*, continuam a colocar mais capacidade de processamento em espaços físicos menores, permitindo que um número superior de equipamentos sejam instalados em compartimentos de *rack* típico. Consequentemente, as densidades de potência dos equipamentos instalados em *rack* crescem a taxas alarmantes.

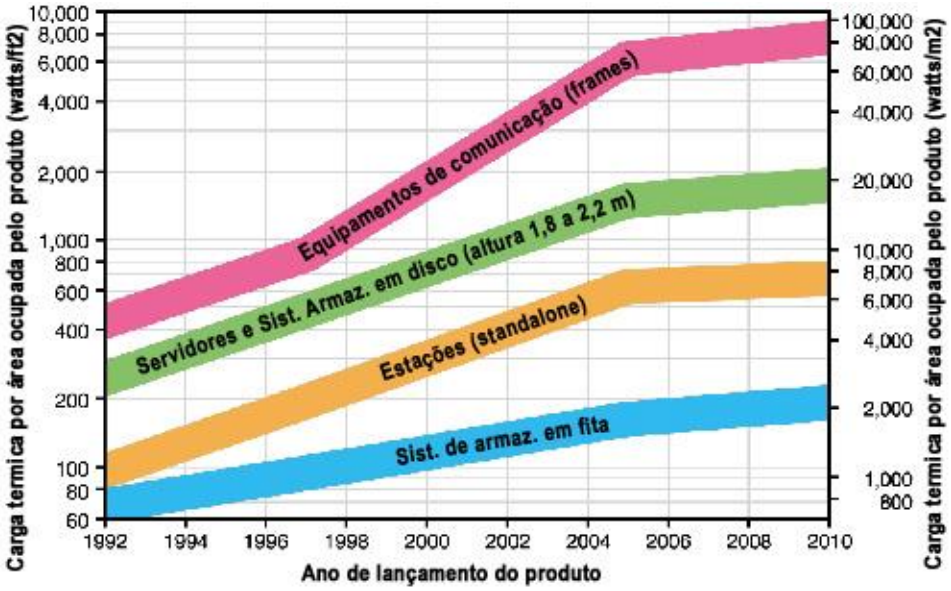


Figura 28: Evolução das cargas térmicas por área ocupada pelo *hardware* em função do seu ano de lançamento. [14]

Neste tipo de instalações, uma das principais causas de paralisação (*Downtime*) são as falhas de *hardware* provocadas por aquecimento excessivo. De forma simples, quando os servidores aquecem acima da temperatura de funcionamento recomendada pelo fabricante, desligam-se automáticamente, implicando grandes riscos. Por exemplo, caso o equipamento de *firewall* sobreaqueça e se autodesligue, a rede interna poderá ficar exposta ao mundo.

Mas o sobreaquecimento da sala de servidores não é a única ameaça à integridade dos sistemas. Também o subaquecimento, derivado de um sistema de climatização mal regulado, é problemático. Caso a humidade relativa da sala seja elevada, se a temperatura cair abaixo de um certo nível, essa humidade irá condensar criando água no interior dos equipamentos. Também se a humidade relativa for baixa em demasia, abaixo dos 40%HR, com o acumular das cargas elétricas (por exemplo nas pás de ventoinhas de refrigeração em fricção com o ar seco) poderão ocorrer mais fácilmente fenómenos de eletricidade estática, podendo inclusivamente provocar faíscas, colocando também desta forma os equipamentos em risco.

A climatização de precisão engloba pois, ar condicionado, ventilação e controlo de humidade para equipamentos de TI. Para garantir a disponibilidade máxima dos equipamentos de TI, o edificio onde estes estão instalados terá que garantir que estes

funcionem dentro dos limites climatéricos aceitáveis. É necessária não só capacidade para climatizar, mas também aumentar o fluxo de ar dentro dos *racks* e redirecionar o ar quente para longe dos dispositivos. Mais especificamente, para garantir o nível óptimo de disponibilidade do sistema, o ar precisa atingir os equipamentos na quantidade, temperatura e humidade corretas.

Assim, não somente pela integridade e longevidade fisica dos equipamentos que por si só apresentam um grande impacto no orçamento de uma empresa, mas sobretudo pela garantia de disponibilidade que estes pressupõem em todo o funcionamento fabril que deles depende, o controlo de climatização de uma sala de servidores é um fator de extrema atenção. [14]

As duas salas de servidores da Renova têm integrado sistemas de condicionadores de precisão, que controlam a qualidade e pureza do ar, mantendo-o limpo a uma temperatura em torno dos 23ºC e uma humidade relativa de 45 a 60%HR.

Figura 29: Condicionador de precisão e pormenor da consola.

2.1 – Formulação do problema

Apesar dos condicionadores de precisão instalados nas salas de servidores não evidenciarem anomalias claras de funcionamento, perante a ocorrência de alguns incidentes, normalmente associados a problemas de climatização das salas, decidiu a Renova encontrar forma de aferir cabalmente o correto funcionamento, ou não, dos mesmos. Para esse efeito foi realizada uma pesquisa generalizada no mercado por soluções comerciais que medissem e registassem com rigor as variáveis associadas à correta climatização das salas de servidores: temperatura e humidade.

Paralelamente, e uma vez que se iria realizar um investimento em equipamentos deste tipo, perante a necessidade de não só monitorizar mas também controlar estas variáveis climatéricas em determinadas etapas no processo fabril, decidiu-se a aquisição de um equipamento que, com pouca adaptação servisse para realizar ambas as funções nas linhas de fabricação.

Existem no mercado uma vastíssima gama de *datalogger's* (equipamentos de monitorização e registo de dados) capazes de realizar a recolha e armazenamento dos dados pretendidos, com bastante rigor e precisão, alguns deles com opção de descarregar a informação para computador para que possam ser analisados e trabalhados de forma mais ampla. No entanto, nenhuma das soluções estudadas permitia qualquer tipo de programação por forma a interferir no processo de modo a controlar essas variáveis, conforme requisito do parágrafo anterior. Para realizar esse tipo de controlo seria necessário desenvolver algum equipamento que interpretasse e processasse os dados do *datalogger* adquirido. Nestes moldes, optou então a Renova por desenvolver um equipamento de raiz que suprimisse completamente as suas necessidades, com a vantagem de, ao ser desenvolvido no seu departamento de eletrónica, ser facilmente configurável, adaptável e expansível tendo como limitações apenas as impostas pela seleção de componentes.

2.2 – Assunções de desenvolvimento.

Para a execução deste projeto, foi seguida a mesma metodologia e procedimentos utilizados pelo departamento de Eletrónica da Renova em todos os outros projetos. Assim, a escolha dos componentes globais tais como: microprocessador, fonte de alimentação, bateria…, o desenho de alguns circuitos eletrónicos, a programação do microcontrolador e o desenvolvimento dos programas de monitorização e do serviço de recolha de dados são os mesmos, ou partilham características semelhantes aos utilizados em projetos anteriores, existindo apenas adaptações próprias que visam ajustar o trabalho existente à realidade deste novo projeto, ou alterações que tragam alguma melhoria ao funcionamento ou desenvolvimento do projeto e suas aplicações. Desta forma diminui-se a necessidade de investimento em novos componentes, espaço de armazenamento, rapidez de desenvolvimento, rapidez na formação de utilizadores e técnicos que utilizaram o produto entre outros.

2.3 – Conceção do projeto.

O projeto foi diferenciado em duas componentes distintas e complementares: o *hardware*, e o *software*.

Entende-se por *hardware* o conjunto constituído por: placa de circuito impresso, placas de suporte dos vários componentes, caixas e programa do microcontrolador embebido na solução.

Já *software*, consta do programa de computador denominado Monitorizador RenPAD, concebido para realizar recolhas dos dados obtidos através da placa de aquisição de dados RenPAD e monitorizar a sua evolução longo do tempo.

O *software* pode ainda ser subdividido em três âmbitos distintos: a Base de Dados RenPAD, o serviço de monitorização RenPADSrv e o programa de configurações e análise com o mesmo nome dado ao programa global: Monitorizador RenPAD.

Todas estas componentes “afetam-se” entre si segundo o diagrama apresentado na figura seguinte:

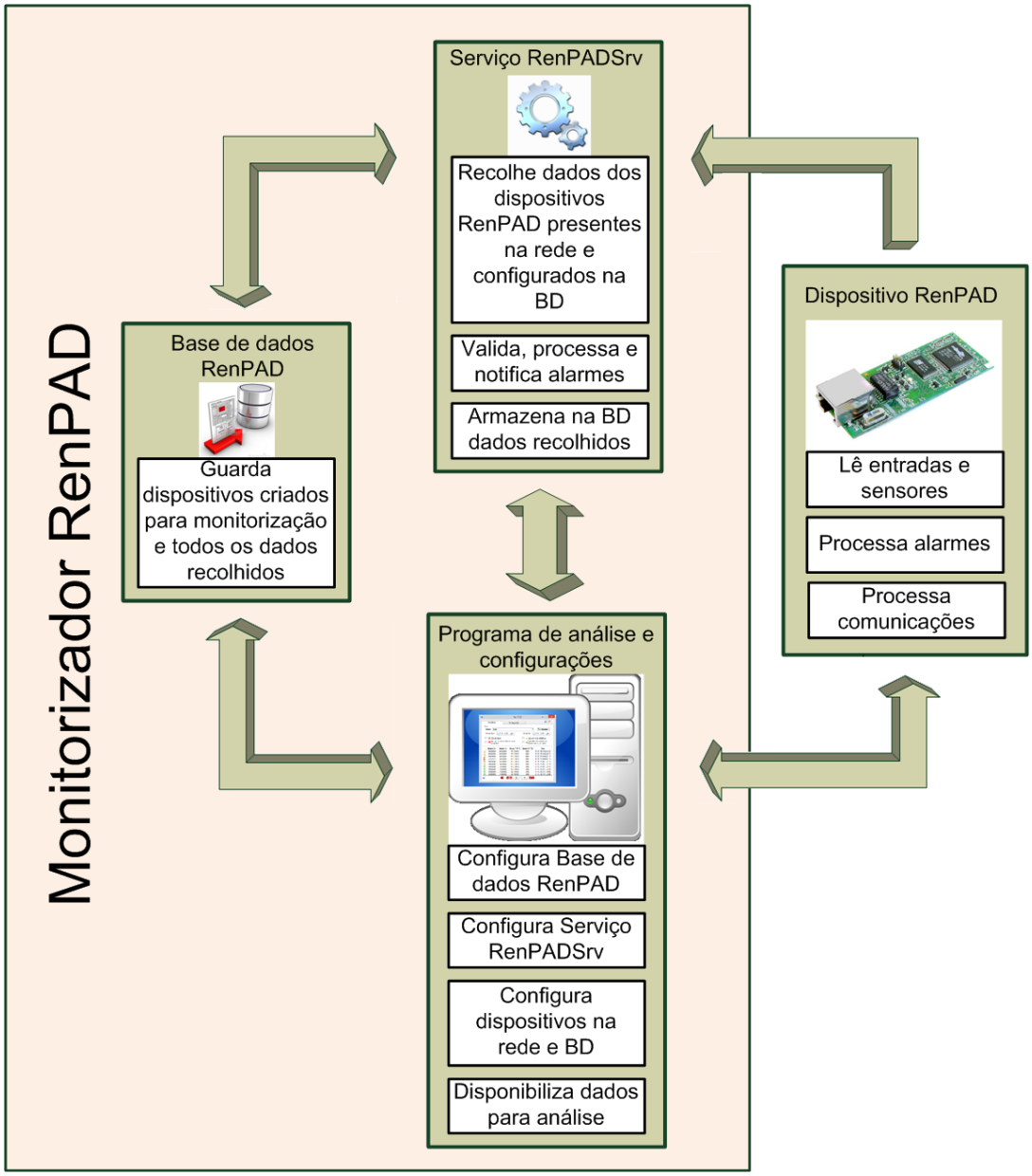


Figura 30: Diagrama de afetação de processos RenPAD

3 – Desenvolvimento do *Hardware* – Dispositivo RenPAD

No contexto deste projeto, e como já referido, entende-se por *hardware* o conjunto constituído por: placa de circuito impresso, placas de suporte dos vários componentes, caixas e programa do microcontrolador embebido na solução. Assim, quando se fala no desenvolvimento do *hardware*, trata-se efetivamente de conceber, desenhar, fabricar e implementar não somente a solução de *hardware* puro, como todas as placas de circuito impresso e componentes, mas também o desenvolvimento do programa a carregar no microcontrolador.

Originalmente, o produto pretendido deveria apenas fazer leituras de temperatura e humidade com razoável precisão. No entanto, e tendo em conta os componentes disponíveis que iriam implementar a solução global e o seu enorme potencial, com intuito de tornar o produto desenvolvido o mais robusto, imune, e escalável possível, adicionamos algumas especificações às características base.

Assim, o produto desenvolvido é neste momento uma ferramenta muito completa, que implementa as seguintes características:

- Comunicação para configuração, leitura de dados e atualização do *software* embebido por rede *Ethernet*;
- 8 Entradas analógicas com conversor ADC de 12 *bit's*;
- 8 Entradas digitais;
- Todas as entradas possibilitam a configuração de diversos tipos de alarme…
- 2 Conetores de ligação a sensores de temperatura e humidade SHT75;
- 1 Bus de ligação para 8 sensores de temperatura TMP75;
- 2 Portas série (Tx/Rx) para utilização futura;
- 8 Saídas digitais de 5 ou 12V, configuráveis para sinalização de alarmes ou outros eventos de interação com dispositivos externos;
- Bateria de funcionamento em caso de falha de energia;

3.1 – A escolha dos componentes

Como referido anteriormente, um dos fatores que influenciou a escolha dos componentes no desenvolvimento dos projetos, foi a sua utilização em outros produtos

desenvolvidos e fabricados pela Renova Eletrónica. Não é com surpresa que utilizamos, por exemplo, o mesmo microcontrolador nas diversas aplicações desenvolvidas. Ainda assim, na escolha de novos componentes tiveram-se em conta todos os critérios necessários ao desenvolvimento de um produto fiável e durável e o mais rentável possível, tais como: preço, precisão e resolução, disponibilidade no mercado e facilidade de aquisição, entre outros.

Depois de aturada pesquisa do mercado e do processo de seleção, foram selecionados os seguintes componentes principais para o produto final:

3.1.1 – Módulo Microprocessador Rabbit RCM3700, seu ambiente de programação Dynamic C e o sistema de tempo real embebido MicroC/OS-II.

Na gama dos microcontroladores/microcontroladores embebidos, este é um dos mais poderosos que conheço, não só pelas opções de *hardware* que disponibiliza como também pelo *software* de programação que o acompanha – o Dynamic C, e que implementa o *kernell* de tempo real MicroC/OS-II , também designado por MicroC/OS-2 ou µC/OS-II.

Este módulo, apesar do seu reduzido tamanho, conta com um microprocessador Rabbit 3000 a 22.1MHZ, um relógio de tempo real, um módulo *Ethernet* 10Base-T, 512Kb de memória Ram (mantida por uma pilha externa), 512Kb de memória Flash-Rom para armazenamento de dados e do programa e 33 entradas/saídas digitais podendo algumas delas ser transformadas em, por exemplo, portas série (num total de 4). O módulo é ligado à placa de circuito impresso recorrendo a uma ficha *header* em pente de 2x20 pinos e tem um desenho extremamente pequeno em comparação com outros módulos semelhantes.

Figura 31: Módulo RCM3700 [15]

RCM3700 RabbitCore Specifications			
Features	RCM3700	RCM3710	RCM3720
Microprocessor	Low-EMI Rabbit 3000®at 22.1 MHz		
Ethernet Port	10Base-T interface, RJ-45, 2 LEDs		
Flash Memory	512K	256K	512K
SRAM	512K	128K	256K
Serial Flash Memory	1Mbyte		
Backup Battery	Connection for user-supplied backup battery (to support RTC and SRAM)		
General-Purpose I/O	33 parallel digital I/0 lines: • 31 configurable I/O • 2 fixed outputs		
Additional I/O	Reset		
Auxiliary I/O Bus	Can be configured for 8 data lines and 5 address lines (shared with parallel I/O lines), plus I/O read/write		
Serial Ports	Four 3.3 V CMOS-compatible ports configurable as: • 4 asynchronous serial ports (with IrDA) or • 3 clocked serial ports (SPI) plus 1 HDLC (with IrDA) or • 1 clocked serial port (SPI) plus 2 HDLC serial ports (with IrDA)		
Serial Rate	Maximum asynchronous baud rate = CLK/8		
Slave Interface	A slave port allows the RCM3700 to be used as an intelligent peripheral device slaved to a master processor, which may be a Rabbit 3000 or another type of processor		
Real-Time Clock	Yes		
Timers	Ten 8-bit timers (6 cascadable, 3 reserved for internal peripherals), one 10-bit timer with 2 match registers		
Watchdog/Supervisor	Yes		
Pulse-Width Modulators	4 PWM output channels with 10-bit free-running counter and priority interrupts		
Input Capture/ Quadrature Decoder	2-channel input capture can be used to time input signals from various port pins • 1 quadrature decoder unit accepts inputs from external incremental encoder modules or • 1 quadrature decoder unit shared with 2 PWM channels		
Power	4.75–5.25 V DC 100 mA @ 22.1 MHz, 5 V; 78 mA @ 11.05 MHz, 5 V		
Operating Temperature	–40°C to +70°C		
Humidity	5% to 95%, non-condensing		
Connectors	One 2 x 20, 0.1" pitch		
Board Size	1.20" × 2.95" × 0.89" (30 mm × 75 mm × 23 mm)		

Tabela 1: Quadro de especificações do módulo RCM3700 [15]

Podemos visualizar esquemáticamente os subsistemas componentes do módulo RCM3700 descritos no parágrafo anterior na figura seguinte:

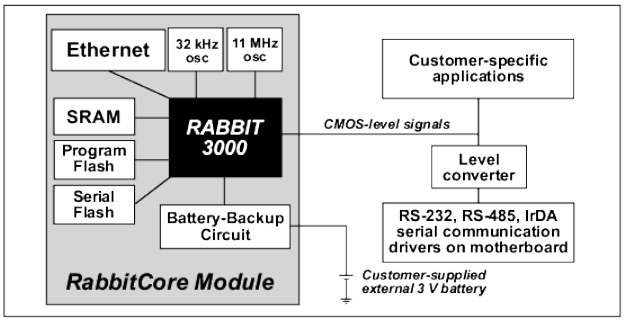


Figura 32: Subsistemas componentes do módulo RCM3700 [16]

Já a figura seguinte representa a função de cada porto do microprocessador Rabbit 3000 no módulo RCM3700. Podemos verificar por exemplo, que a porta A do microcontrolador Rabbit 3000 é, no módulo RCM3700, implementada pelos portos PA0 a PA7 enquanto que a porta C, que integra as portas série C e D, é definida no módulo RCM3700 pelos portos PC0, PC2, PC1 E PC3.

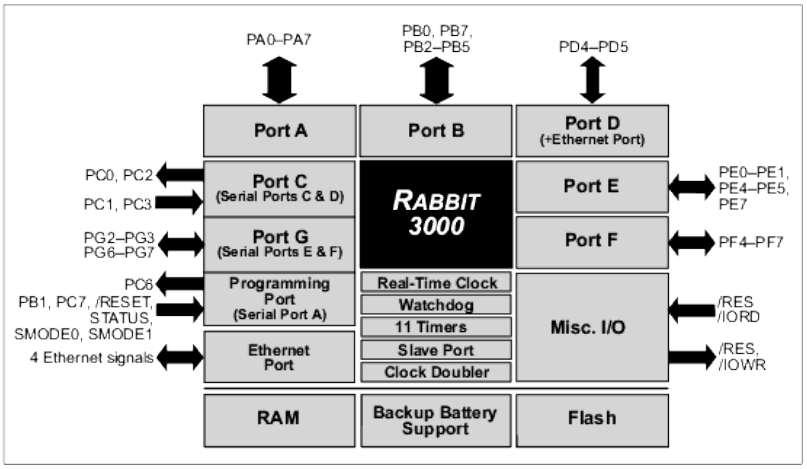


Figura 33: Funções dos portos do microprocessador Rabbit 3000 no módulo RCM3700 [16]

Como referido anteriormente, o módulo RCM3700 é fisicamente conectado por meio de uma ficha *header* em pente de 2x20 pinos com o seguinte *pinout*:

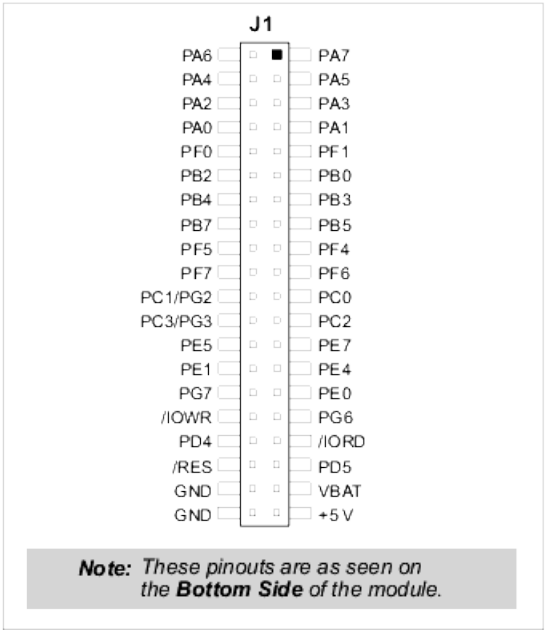


Figura 34: *Pinout* do módulo RCM3700 [16]

A descrição pormenorizada da função de cada um destes pinos, suas funções alternativas e função assumida por defeito são apresentadas na tabela seguinte:

Pino		Nome	Uso predefinido	Uso alternativo	Notas
Header J1	1-8	PA[7:0]	Parallel I/O	External data bus (ID0-ID7) Slave port data bus (SD0-SD7)	External Data Bus
	9	PF1	Input/Output	QD1A CLKC	
	10	PF0	Input/Output	QD1B CLKD	
	11	PB0	Input/Output	CLKB	
	12	PB2	Input/Output	IA0 /SWR	External Address 0 Slave port write
	13	PB3	Input/Output	IA1 /SRD	External Address 1 Slave port read
	14	PB4	Input/Output	IA2 SA0	External Address 2 Slave Port Address 0
	15	PB5	Input/Output	IA3 SA1	External Address 3 Slave Port Address 1
	16	PB7	Input/Output	IA5 /SLAVEATTN	External Address 5 Slave Port Attention
	17	PF4	Input/Output	AQD1B PWM0	
	18	PF5	Input/Output	AQD1A PWM1	
	19	PF6	Input/Output	AQD2B PWM2	
	20	PF7	Input/Output	AQD2A PWM3	
	21	PC0	Output	TXD	Serial Port D
	22	PC1/PG2	Input/Output	RXD/TXF	Serial Port D Serial Port F
	23	PC2	Output	TXC	Serial Port C

Pino		Nome	Uso predefinido	Uso alternativo	Notas
Header J1	24	PC3/PG3	Input/Output	RXC/RXF	Serial Port C Serial Port F
	25	PE7	Input/Output	I7 /SCS	External Address 7 Slave Port Chip Select
	26	PE5	Input/Output	I5 INT1B	
	27	PE4	Input/Output	I4 INT0B	
	28	PE1	Input/Output	I1 INT1A	I/O Strobe 1 Interrupt 1A
	29	PE0	Input/Output	I0 INT0A	I/O Strobe 0 Interrupt 0A
	30	PG7	Input/Output	RXE	Serial Port E
	31	PG6	Input/Output	TXE	
	32	/IOWR	Output		External write strobe
	33	/IORD	Input		External read strobe
	34	PD4	Input/Output	ATXB	Alternate Serial Port B
	35	PD5	Input/Output	ARXB	
	36	/RES	Reset output	Reset input	Reset output from Reset Generator
	37	VBAT			
	38	GND			
	39	+5 V			
	40	GND			

Tabela 2: Descrição do *pinout* do módulo RCM3700 [16]

Como referido anteriormente, também o *software* de desenvolvimento e programação para esta gama de microprocessadores, o Dynamic C, é uma poderosa ferramenta, sendo mais uma das grandes vantagem deste módulo em relação a outros de fabricantes concorrentes.

Figura 35: Logotipo Dynamic C

O IDE (ambiente integrado de desenvolvimento) apesar de simples e intuitivo, implementa praticamente todas as funções presentes nos melhores ambientes de desenvolvimento. Funções como pesquisa avançada, de auto-indexação, de ajuda integrada, *etc*… estão presentes e facilmente acessíveis.

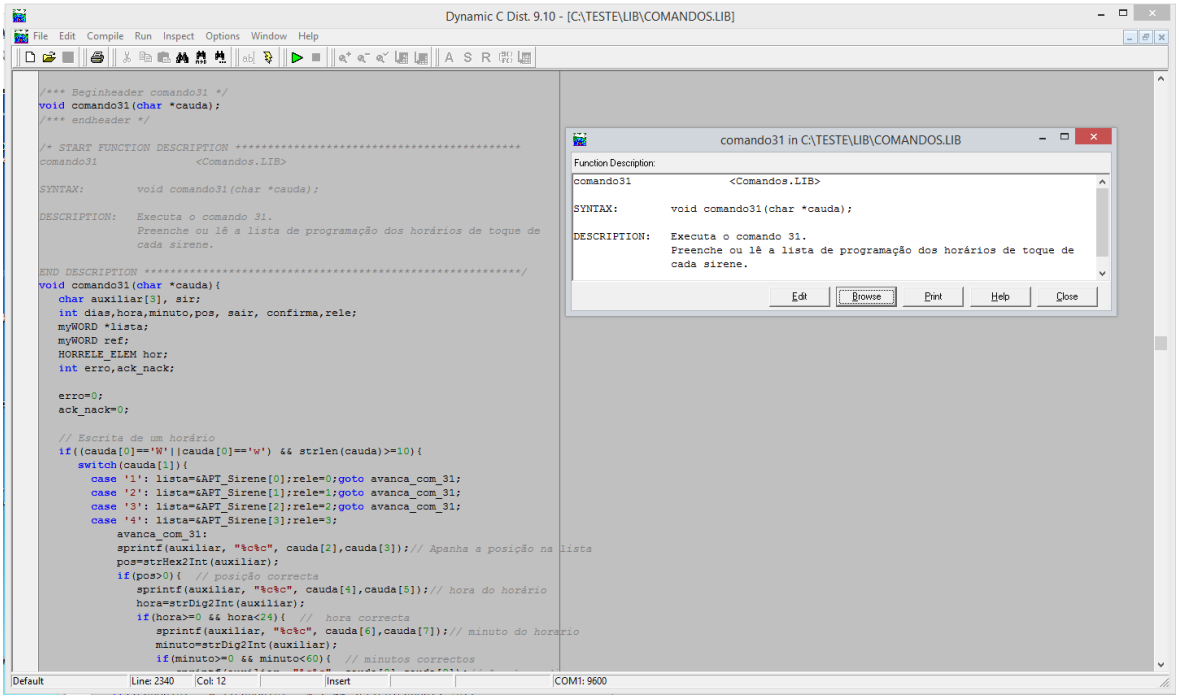


Figura 36: Ambiente de desenvolvimento do IDE Dynamic C, com janela de ajuda integrada (ctrl+H).

É no entanto pela integração do *kernel* de tempo real MicroC/OS-II como base da sua conceção, que o Dynamic C se destaca da concorrência tornando-o numa ferramenta completa para desenvolvimento fácil e rápido de soluções de tempo real com recurso a *multithreading*, semáforos, *mailboxes*, prioridade de eventos, entre outros.

Um Sistema Operativo de Tempo Real (RTOS da sigla anglo-saxónica *Real Time Operating System*) é um sistema operativo desenhado e vocacionado para a execução de múltiplas tarefas (*multithreads*) em que o tempo de resposta a um evento (externo ou interno) está pré-definido. O não cumprimento de uma tarefa, dentro do prazo esperado,

caracteriza uma falha do sistema. Outra característica dos sistemas de tempo real (STR) é a sua interação com o meio ao redor. O STR tem que reagir, dentro de um prazo pré-definido, a um estímulo do meio.

O MicroC/OS-II, propriedade da empresa *Micrium Inc* que recentemente lançou uma nova versão deste RTOS, o MicroC/OS-III, tem como principais características as seguintes:

- O MicroC C/OS-II vem com o código-fonte ANSI C;
- Possui escabilidade entre 5 a 24 kbytes;
- Portabilidade;
- Preempção;
- Multi-tarefa;
- Semáforos;

Por ser um sistema operativo de tempo real o MicroC /OS-II não possui uma gestão de processos muito complexa como outros sistemas operativos a aplicar em sistemas de comutação mais poderosos, podendo mesmo considerar-se muito primitiva. O sistema operativo funciona como o único processo mantido em si. Porém o MicroC/OS-II pode gerir até 64 tarefas (*threads*), sendo que 8 destas são utilizadas pelo próprio sistema operativo para assegurar a sua execução, a fim de manter o funcionamento e execução das restantes.

A cada tarefa é associada uma prioridade de execução para definir, em caso de sobreposição de pedidos, qual a tarefa a ser atendida em primeiro lugar. Neste sistema operativo, o menor valor de prioridade corresponde a uma maior prioridade de execução. Para o MicroC/OS-II o número de prioridade da tarefa é também um identificador para esta.

No sistema operativo MicroC/OS–II as tarefas podem estar num de cinco estados (*Task States*): Dormente (*Dormant*), Pronto (*Ready*), Espera (*Waiting*), Execução (*Running*) e Exceção (ISR – *Interrupt Service Request*), existindo ferramentas de criação, remoção, suspensão e retoma de tarefas bem como alteração do valor da sua prioridade, invocadas nos seguintes moldes:

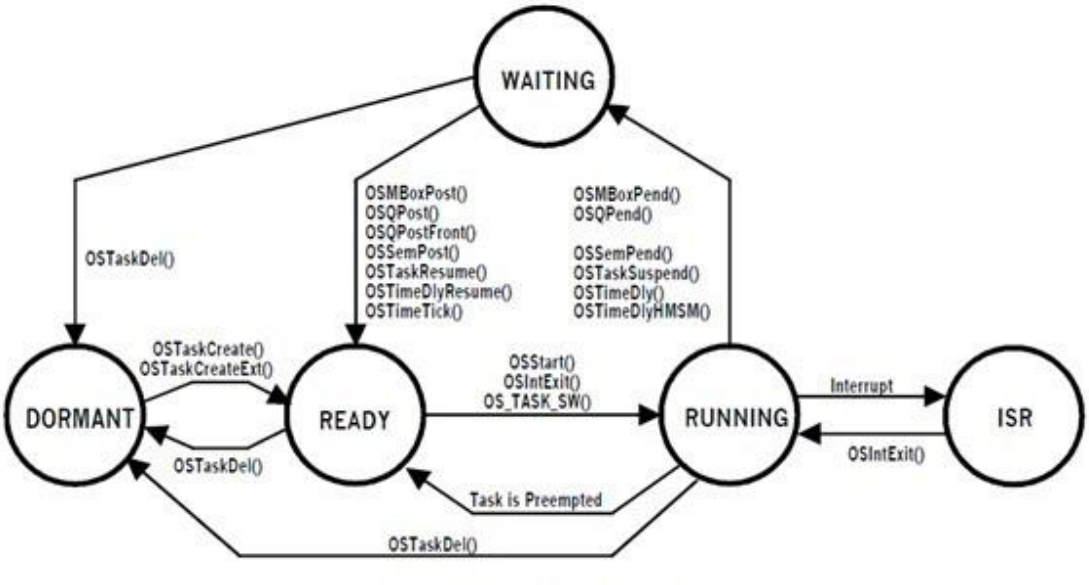


Figura 37: Esquema de transições entre os diversos estados das tarefas no MicroC/OS-II [17]

- Diz-se que uma tarefa se encontra no estado Dormente quando reside na memória (ROM ou RAM) mas ainda não está disponível para o MicroC/OS-II, ou seja, é carregado um evento para a memória, mas a tarefa associada a esse evento (e por ele originada) ainda está para ser criada. Essa tarefa passa para o estado de Pronto pela chamada dos métodos "OSTaskCreate()" ou "OSTaskCreateExt()".
- Quando as tarefas estão prontas para serem executadas, todas elas estão no estado de Pronto. Logo que o processador esteja disponível, é selecionada para execução a tarefa de maior prioridade que se encontre nesse estado.
- Quando ocorre um pedido de execução de uma tarefa de maior prioridade, a tarefa em execução (de menor prioridade) é colocada novamente no estado de Pronto, e é trocada pela tarefa de maior prioridade que passa ao estado de Execução. A esta troca dá-se o nome de preempção.
- Caso uma tarefa em execução termine completamente, é colocada novamente no estado de Dormente através da invocação do método "OSTaskDel()".
- Quando uma tarefa necessita de outra para continuar a executar é colocada no estado de Espera até que a tarefa pela qual aguarda execute completamente. Quando isso ocorrer, a tarefa que estava no estado de Espera volta para o estado de Pronta, ficando novamente a aguardar pela sua vez de ser executada.

- Quando ocorre uma exceção ou interrupção, o estado ISR ganha o controlo do processador. A tarefa que está no estado de Execução nesse momento é interrompida e colocada no estado Pronto, e o processador passa a executar as tarefas que se encontram no estado de ISR. Findo este processo, o processador volta a atender as tarefas que estão no estado de Pronto iniciando na de maior prioridade. A tarefa que se encontrava anteriormente no estado de execução e que foi interrompida, terá que aguardar até ter a prioridade mais elevada para terminar a sua execução.

Para comunicação e troca de dados entre tarefas (*Intertask Communication*), o MicroC/OS-II fornece três tipos de mecanismos com intuito de proteger os dados compartilhados. São eles: semáforos, correio de mensagens (*message mailbox*) e fila de mensagens (*message queue*).

O mecanismo de comunicação é despoletado a partir de sinais enviados por uma tarefa (*thread1*) ou por um ISR para outra tarefa (*thread2*), através de um objeto do *Kernel* chamado *Event Control Block* (ECB). Isto faz com que a *thread2*, que foi bloqueada, espere até que a outra que enviou o sinal termine sua execução ou liberte esse sinal para poder executar novamente.

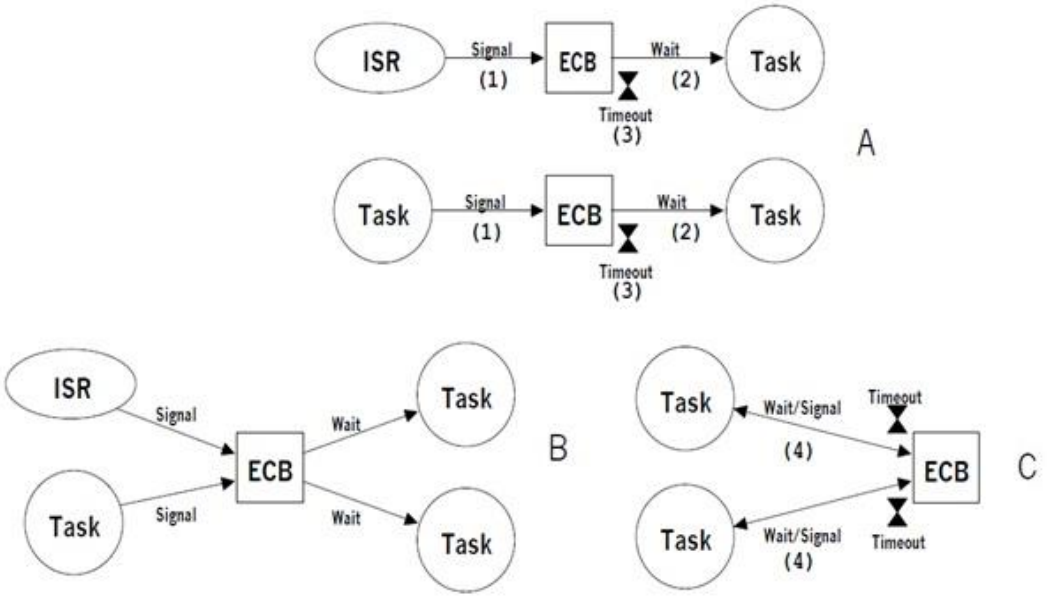


Figura 38: Opções de uso do objeto ECB (*Event Control Block*) [17]

Destes mecanismos de comunicação, os semáforos são os mais utilizados (a titulo de exemplo, no *software* desenvolvido no âmbito deste projeto foi o único ECB utilizado). Quando o ECB é um semáforo, ambas as tarefas irão esperar e sinalizar o ECB para evitar que ambas entrem na sua região crítica em simultâneo.

Tal como o gestor de tarefas, também o gestor de memória do MicroC/OS-II é algo básico e muito primitivo. Deve-se isto ao facto deste sistema operativo não possuir uma grande capacidade de memória principal e secundária, por limitações físicas dos dispositivos que o implementam, o que o impede de recorrer a características como memória virtual e *swap*.

As aplicações que correm no MicroC/OS-II podem utilizar a memória dinâmica para alocar e desalocar, através dos comandos "malloc()" e "free()" respetivamente, usados pelo compilador ANSI C. No entanto a utilização destas funções podem deixar bastante fragmentação na memória em sistemas de tempo real.

Todos os blocos de memória têm o mesmo tamanho e as partições contém números integrais de blocos. As operações de alocação e devolução ("desalocação") desses blocos à memória global são feitas em períodos de tempo constantes, chamando-se por isso operações determinísticas, cujo consumo global de tempo pode ser rigorosamente determinado e tido em conta na programação das tarefas que os implementam.

Já no que toca às partições, o número de blocos que as implementam pode variar consoante a necessidade das aplicações que as criam. Isso pode ser visto nas figuras abaixo. Este tipo de memória não está sujeito a fragmentação, e um bloco de memória especifico deve retornar sempre à partição de onde veio.

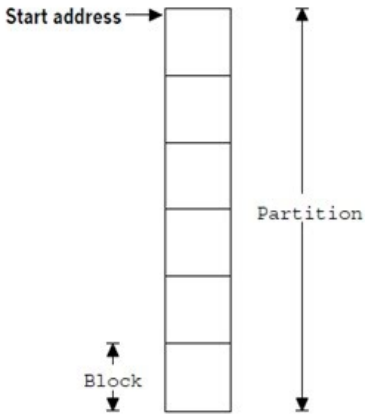


Figura 39: Partição de memória no MicroC/OS-II [17]

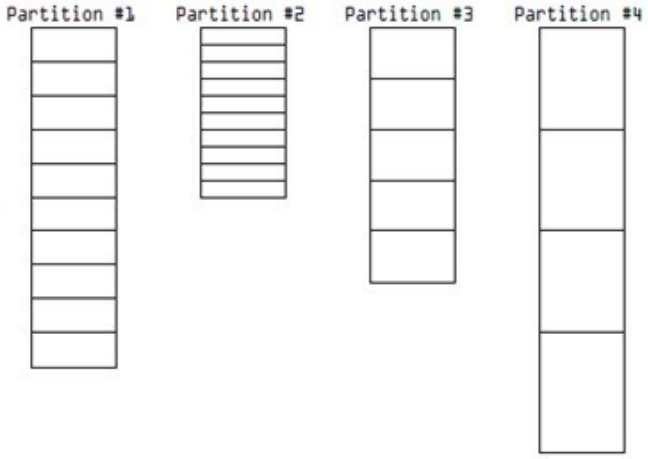


Figura 40: Múltiplas partições de memória no MicroC/OS-II [17]

Relativamente aos dispositivos de Entrada/Saída (E/S), estes podem comunicar com o MicroC/OS-II, através de interrupções. Quando uma tarefa está em execução e ocorre um evento de E/S é gerada uma interrupção, e como já foi mencionado anteriormente, numa situação destas, a tarefa que está no estado de Execução é colocada no estado de Pronto até que o evento que originou essa interrupção, e que possui maior prioridade, termine sua execução. [17]

3.1.2 – Conversor analógico-digital MCP3208

Este ADC disponibiliza a conversão de 8 entradas analógicas em digitais de forma série (compatível com o protocolo SPI - *Serial Peripheral Interface*), com uma resolução de 12 *bit's* e uma taxa máxima de amostragem de 100Ksps (kilo amostragens por segundo). Pode ser alimentado recorrendo a uma tensão entre 2.7 e 5.5V e implementa a tecnologia *Low Power* CMOS que permite consumos extraordinariamente baixos, de 500ηA a 400µA entre o regime de standby e consumo máximo. Suporta ainda uma referência externa, que implementamos com o componente de referência de tensão de precisão MCP1525 de 2,5V, para compensação das variações de temperatura externas para que esta não influencie no resultado final da conversão.

Nas figuras seguintes são mostradas as características mais imperantes deste sensor: *Pinout*, diagrama de blocos e temporizações da *interface* série

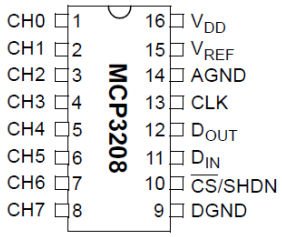


Figura 41: *Pinout* MCP3208 PDIP *Package* [18]

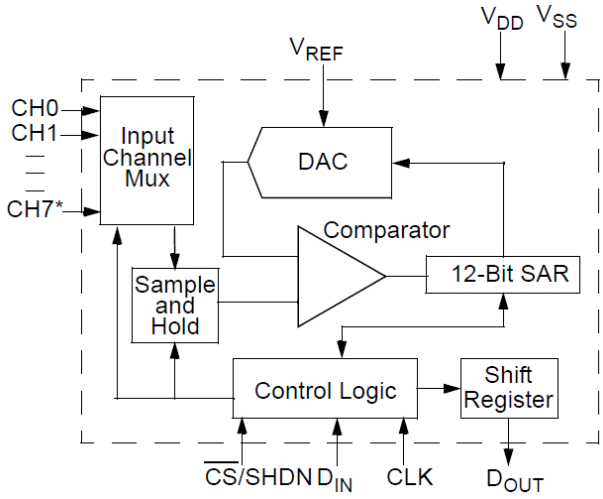


Figura 42: Diagrama de blocos do MCP3208 [18]

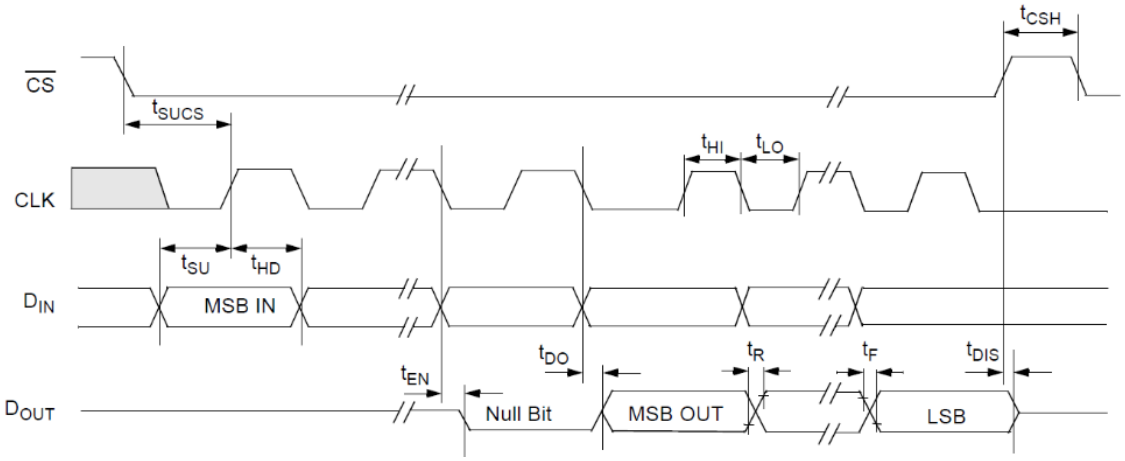


Figura 43: Temporizações da *Interface* Série [18]

Como referido anteriormente, a comunicação com o dispositivo MCP3208 é realizada por meio de um canal série, compatível com a norma *interface* SPI e tem o seguinte modo de funcionamento (ver Tabela 3 e Figura 44):

- A comunicação é inicializada colocando a linha CS no estado baixo.

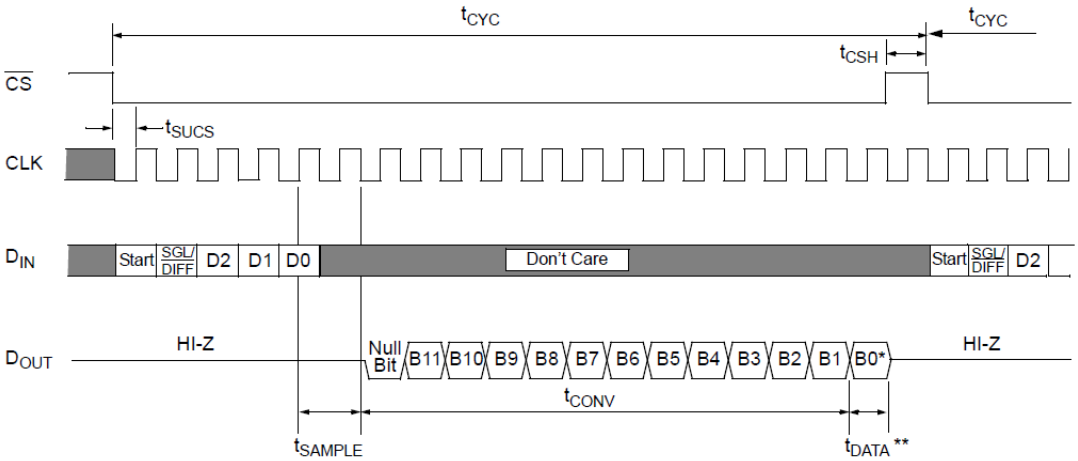
- Caso o dispositivo tenha sido inicializado com a linha CS no estado baixo, esta deve ser colocada no estado alto e recolocada no estado baixo para que o início da comunicação tenha sucesso.
- O primeiro sinal de *clock* recebido com a linha CS no estado baixo e a linha DIN no estado alto, constitui um *start bit*.
- Seguidamente, é iniciada a programação do dispositivo para determinar se a conversão é realizada no modo *single-ended* (valor simples da entrada) ou no modo *de differential input* (valor diferencial entre duas entradas). Isto é conseguido à custa do envio do *bit* SGL/DIFF que quando colocado no nível alto seleciona o modo *single-ended* e quando colocado no nível baixo seleciona o modo *differential input*.
- Os três *bit's* seguintes (D0, D1 e D2) fazem a seleção da entrada que se pretende ler, conforme se pode visualizar na Tabela 3.
- O processo de conversão da entrada analógica selecionada, para o valor digital correspondente é iniciado durante o bordo ascendente do quarto pulso de *clock* após a recepção do *start bit*, e termina durante o bordo descendente do quinto pulso de *clock*.

  Uma vez que D0 é um *bit* entrada, é necessário mais este pulso de *clock*, criando um compasso de espera para que o dispositivo tenha tempo de completar a amostra e durante este período, o valor de DIN não tem significado.

  Este período de conversão é designado por t_{Sample} na Figura 44.
- No bordo descendente do pulso de *clock* seguinte, o dispositivo coloca à saída um *bit null*, de nível baixo, de modo a preparar a saída para a transmissão do valor da conversão.
- Durante os seguintes 12 pulsos de clock é transmitido para a saída o valor digital obtido na conversão da entrada selecionada, sendo enviado primeiro o Byte mais significativo (MSB) seguido do Byte menos significativo (LSB). Os dados são transmitidos para a saída durante o bordo descendente do pulso de *clock*.
- Por limitações físicas do dispositivo, para assegurar o seu correto funcionamento a frequência de *clock* deve manter-se abaixo dos 10KHz.

Control Bit Selections				Input Configuration	Channel Selection
Single /Diff	D2	D1	D0		
1	0	0	0	single-ended	CH0
1	0	0	1	single-ended	CH1
1	0	1	0	single-ended	CH2
1	0	1	1	single-ended	CH3
1	1	0	0	single-ended	CH4
1	1	0	1	single-ended	CH5
1	1	1	0	single-ended	CH6
1	1	1	1	single-ended	CH7
0	0	0	0	differential	CH0 = IN+ CH1 = IN-
0	0	0	1	differential	CH0 = IN- CH1 = IN+
0	0	1	0	differential	CH2 = IN+ CH3 = IN-
0	0	1	1	differential	CH2 = IN- CH3 = IN+
0	1	0	0	differential	CH4 = IN+ CH5 = IN-
0	1	0	1	differential	CH4 = IN- CH5 = IN+
0	1	1	0	differential	CH6 = IN+ CH7 = IN-
0	1	1	1	differential	CH6 = IN- CH7 = IN+

Tabela 3: Configuração do modo de funcionamento do MCP3208 [18]



* After completing the data transfer, if further clocks are applied with $\overline{CS}$ low, the A/D converter will output LSB first data, followed by zeros indefinitely.

** t_{DATA}: during this time, the bias current and the comparator power down while the reference input becomes a high impedance node, leaving the CLK running to clock out the LSB-first data or zeros.

Figura 44: Esquema de comunicações para o MCP3208 [18]

A função criada para implementar o mecanismo de comunicação para a leitura de uma entrada analógica deste ADC pode ser consultada integralmente no Anexo 1 onde,

pela leitura dos comentários (texto de cor verde) se pode acompanhar o desenvolvimento de cada diagrama.

Para redução do ruido de acoplamento resultante da sua parte digital que poderia contaminar a amostragem, o MCP3208 apresenta os circuitos analógicos e digitais internos fisicamente separados, estando apenas conectados por uma resistência com valor entre 5 e 10Ω sendo disponibilizadas duas massas separadas: a massa do bloco analógico (AGND) e a massa do bloco digital (DGND).

Estas massas devem ser ligadas ao plano de massa do circuito, conforme se apresenta na figura seguinte:

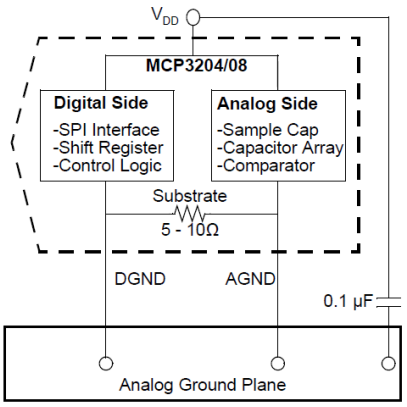


Figura 45: Separação das massas DGND de AGND para redução de ruido por efeito de acoplamento. [18]

A equação de conversão do valor analógico para digital é representada da seguinte forma: [18]

$$Digital\ Output\ Code = \frac{4096 \times V_{in}}{V_{ref}} \qquad (1)$$

Com:

V_{in} – Valor de tensão da entrada analógica.

V_{ref} – Valor da tensão de referência.

3.1.3 – Sensor de temperatura LM60

Um dos sensores de temperatura inicialmente utilizado, o LM60 pode operar com tensões entre os 2.7 e os 10V e tem um comportamento relativamente linear de 6.25mV/ºC para temperaturas entre os -40 e os 125ºC. A sua tolerância relativamente alta (±2ºC a

25ºC), aliada à tolerância de conversão do ADC MCP3208, fizeram com que as leituras de alguns dos LM60 utilizados (com tolerâncias piores) tivessem um erro maior do que o previamente definido e ditassem a sua substituição/complemento com sensores mais fiáveis e precisos como os TMP75 e os SHT75.

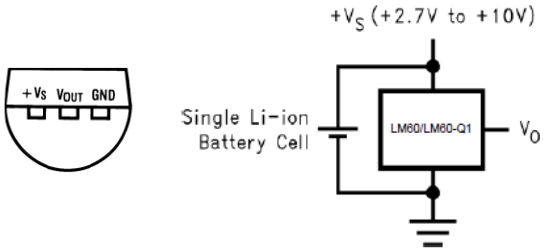


Figura 46: *Pinout* LM60 TO92 *Package* e esquema típico de ligação. [19]

V_O = (+6.25 mV/°C × T°C) + 424 mV

Temperature (T)	Typical VO
+125°C	+1205 mV
+100°C	+1049 mV
+25°C	+580 mV
0°C	+424 mV
–25°C	+268 mV
–40°C	+174 mV

Tabela 4: Fórmula de conversão da tensão Vo em Temperatura e tabela de valores típicos. [19]

As fórmulas de conversão utilizadas para este sensor são as seguintes: [18][19]

$$T = \frac{\frac{ValorLido \times 2.5}{4096} - 0{,}424}{6{,}25 \times 10^{-3}} \ (℃) \qquad (2)$$

$$ValorLido = 694{,}6816 + 10{,}24 \cdot T \ \ (Decimal) \qquad (3)$$

Por exemplo: Se ValorLido = 0x03A9 = 937 decimal.

$$T = \frac{\frac{937 \cdot 2{,}5}{4096} - 0{,}424}{6{,}25 \cdot 10^{-3}} = 23.66 \ ℃$$

A título de exemplo, seguidamente é apresentado o código da função do programa de monitorização RenPAD que calcula o valor da conversão do valor analógico, fornecido por um sensor LM60, em digital através do ADC MCP3208 (ver fórmula 1 da secção anterior).

```
/// <summary>
/// Converte o valor presente na entrada de um sensor analógico em valor de
/// temperatura segundo a fórmula do sensor LM60
/// </summary>
/// <param name="valorAIn">Valor a converter.</param>
/// <returns>Valor convertido.</returns>
public static float ConverterHex2GrausAIn(int valorAIn)
{
    return (float)((((valorAIn * 2.5) / 4096) - 0.424) / 0.00625);
}
```

3.1.4 – Sensor de temperatura TMP75

Este sensor tem especificações parecidas com o LM60 (comportamento relativamente linear de 6.25mV/ºC para uma gama de temperaturas de -40 a 125ºC, alimentação de 2.7 a 5.5V, ADC interno com uma resolução de 9 a 12 *bit's* selecionáveis pelo utilizador…), sendo a tolerância de 1.5ºC à temperatura ambiente de 25ºC e com o conversor ADC interno (reduzindo erros de conversão), a tolerância global deste sensor é inferior à que o LM60 apresenta. Além disso, em testes experimentais realizados com diversos sensores em diversas aplicações diferentes, verificou-se também que a uniformidade dos valores de temperatura obtidos entre diversos sensores ligados no mesmo ponto, é muito superior à dos sensores LM60 a funcionar nas mesmas condições, dando-nos por isso uma maior confiança no resultado apresentado.

Estes sensores utilizam um protocolo de comunicações proprietário (*Two-Wire interface SMBus-compatible*), implementado no microcontrolador por meio de *software*, e permite a ligação de até oito sensores, endereçáveis um a um, no mesmo barramento de comunicações.

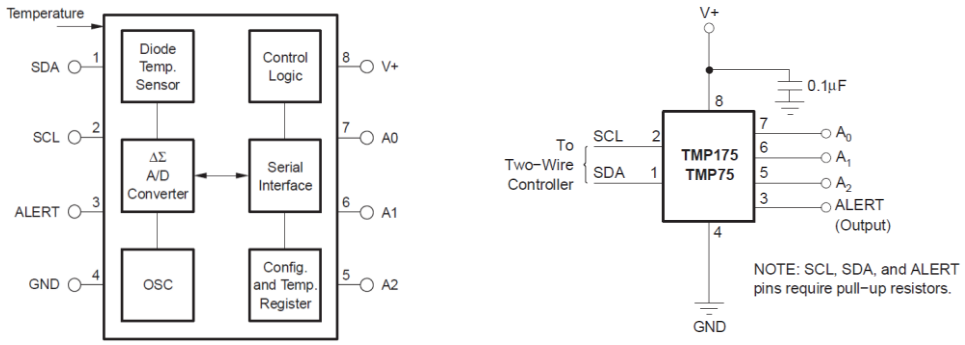


Figura 47: *Pinout* TMP75 SO-2 *Package* e esquema típico de ligação. [20]

Estes sensores não necessitam de nenhum componente para assegurar o seu funcionamento no entanto, e como podemos verificar na Figura 47, é recomendado a ligação de um condensador de filtragem de alimentação de 0,1µF.

Para a ligação destes sensores ao barramento de comunicação, e uma vez que o objetivo é espalhá-los por vários pontos distintos em cada sala de servidores por forma a termos uma ideia mais abrangente da distribuição de temperatura pela sala, foi desenhada uma placa de circuito impresso. Nessa placa apenas constam, para além de uma ficha para ligação ao barramento de comunicação e à alimentação do circuito, o condensador de filtragem anteriormente mencionado e os pinos em forma de *shunt*, que definem o endereço de cada sensor, um regulador de tensão que transforma a tensão de alimentação presente no bus (de 5V ou de 12V conforme a seleção) para a alimentação de 3,3V que irá alimentar efetivamente o sensor.

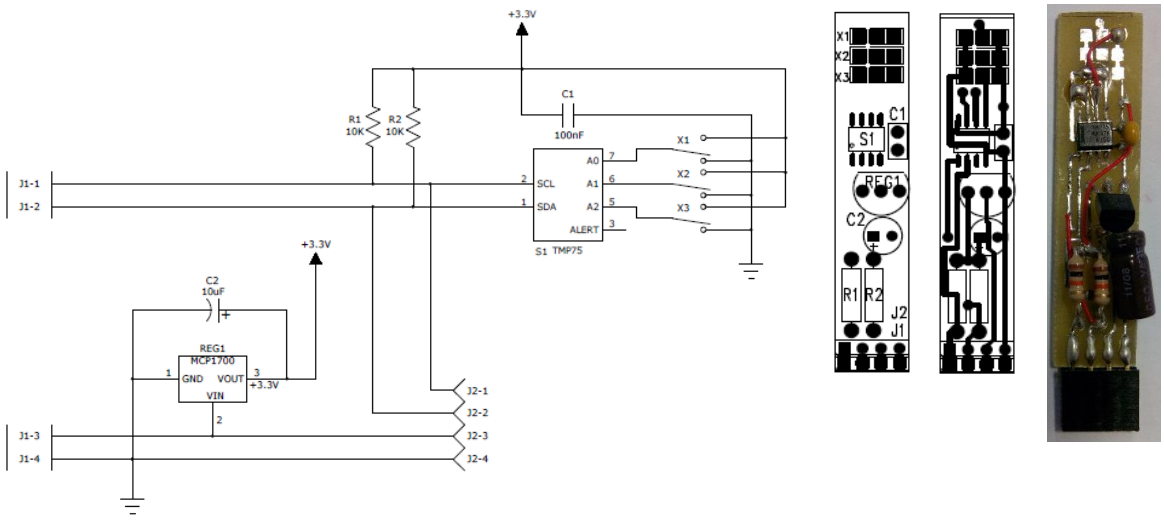


Figura 48: Placa de circuito impresso para sensor TMP75: Esquema de ligações, e placa de circuito impresso.

Para diferenciar cada um dos oito sensores que podem ser ligados ao barramento de comunicação, é necessário atribuir a cada um deles um endereço distinto dos outros. Esse endereço é definido pelo estado das linhas A0, A1 e A2 segundo a seguinte tabela:

A0	A1	A2	Endereço
0	0	0	1
1	0	0	2
0	1	0	3
1	1	0	4
0	0	1	5
1	0	1	6
0	1	1	7
1	1	1	8

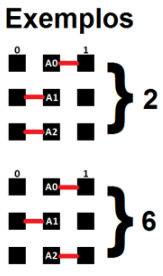


Tabela 5: Tabela de configuração de endereços do sensor TMP75 e exemplos concretos da realização dos *shunt's* na placa de circuito impresso da Figura 48. [20]

Devido a sua multifuncionalidade, este sensor está dotado de quatro registos internos: três de configuração e um de armazenamento conforme se pode verificar na figura seguinte:

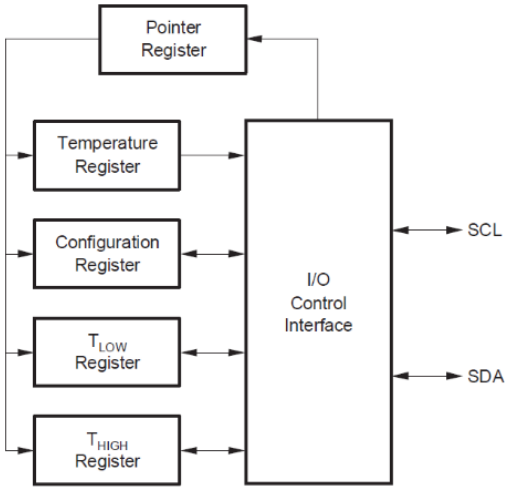


Figura 49: Estrutura de registos internos do sensor TMP75. [20]

- Registo de configuração: onde é guardada a configuração de funcionamento e modos de operação do sensor.

BYTE	D7	D6	D5	D4	D3	D2	D1	D0
1	OS	R1	R0	F1	F0	POL	TM	SD

  - *Shutdown Mode* (SD)
  - *Thermostat Mode* (TM)
  - *Polarity* (POL)
  - *Fault Queue* (F1/F0)
  - *Converter Resolution* (R1/R0)
  - *One-Shot* (OS)

Tabela 6: Registo de configuração dos modos de operação do sensor TMP75. [20]

De todos os modos de operação, apenas irei descrever o que nos interessa configurar para o nosso funcionamento, que são os *bit's* de configuração da resolução de leitura: R0 e R1. Estes *bit's* serão os únicos a ser enviados com o valor lógico 1 para configurar a resolução máxima de leitura do sensor, ou seja: 12 *bit's* conforme se pode validar na tabela seguinte:

R1	R0	RESOLUTION	CONVERSION TIME (typical)
0	0	9 Bits (0.5°C)	27.5ms
0	1	10 Bits (0.25°C)	55ms
1	0	11 Bits (0.125°C)	110ms
1	1	12 Bits (0.0625°C)	220ms

Tabela 7: Tabela de configuração da resolução da leitura do sensor TMP75. [20]

- Registo de temperatura: onde é armazenado o valor da última conversão de temperatura com sucesso. Notar que antes da ocorrência de qualquer conversão, durante o ato de ligação o dispositivo por exemplo, este registo apresenta o valor zero por defeito.

D7	D6	D5	D4	D3	D2	D1	D0
T11	T10	T9	T8	T7	T6	T5	T4
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
T3	T2	T1	T0	0	0	0	0

Tabela 8: Registos internos de temperatura do sensor TMP75. [20]

- Registos T_{Low} e T_{Hight}: Registo de configuração da temperatura mínima e máxima de disparo de um evento de alarme, quando o sensor está configurado para funcionar nesse modo. Para o nosso modo atual de funcionamento, estes registos não têm interesse.

Estes registos estão acessíveis através da manipulação de um apontador de registos (*Pointer Register*) durante a comunicação segundo a seguinte tabela:

P7	P6	P5	P4	P3	P2	P1	P0
0	0	0	0	0	0	Register Bits	

P1	P0	REGISTER
0	0	Temperature Register (READ Only)
0	1	Configuration Register (READ/WRITE)
1	0	T_{LOW} Register (READ/WRITE)
1	1	T_{HIGH} Register (READ/WRITE)

Tabela 9: Apontador de registos do sensor TMP75. [20]

Uma vez que podemos comunicar com o sensor TMP75 tanto para configurarmos o modo de funcionamento, como para lermos o valor da última conversão de temperatura armazenado, existem dois diagramas de comunicação distintos: um para leitura e outro para escrita de dados.

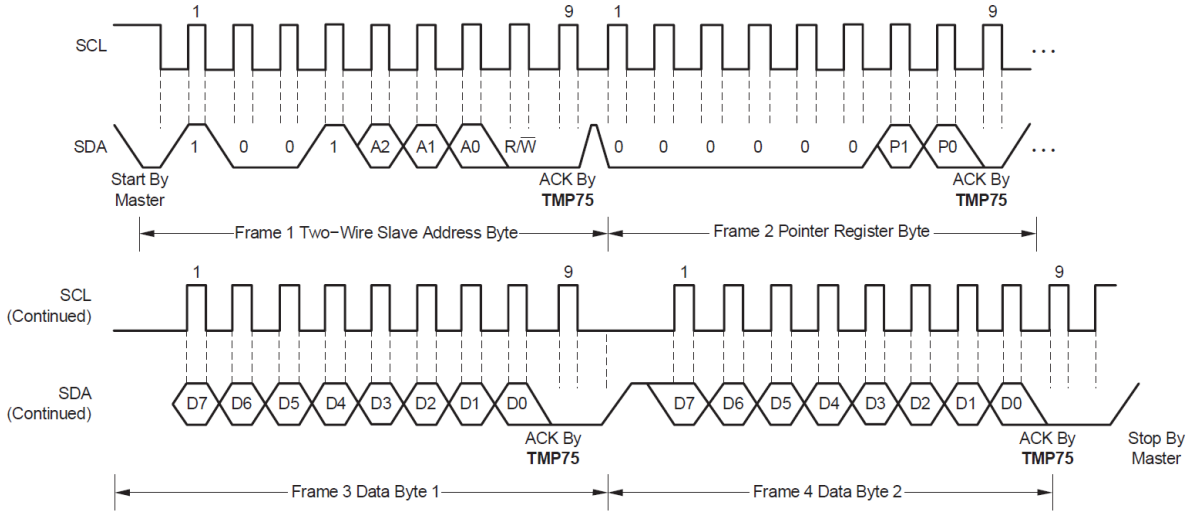


Figura 50: Diagrama de escrita de dados do sensor TMP75. [20]

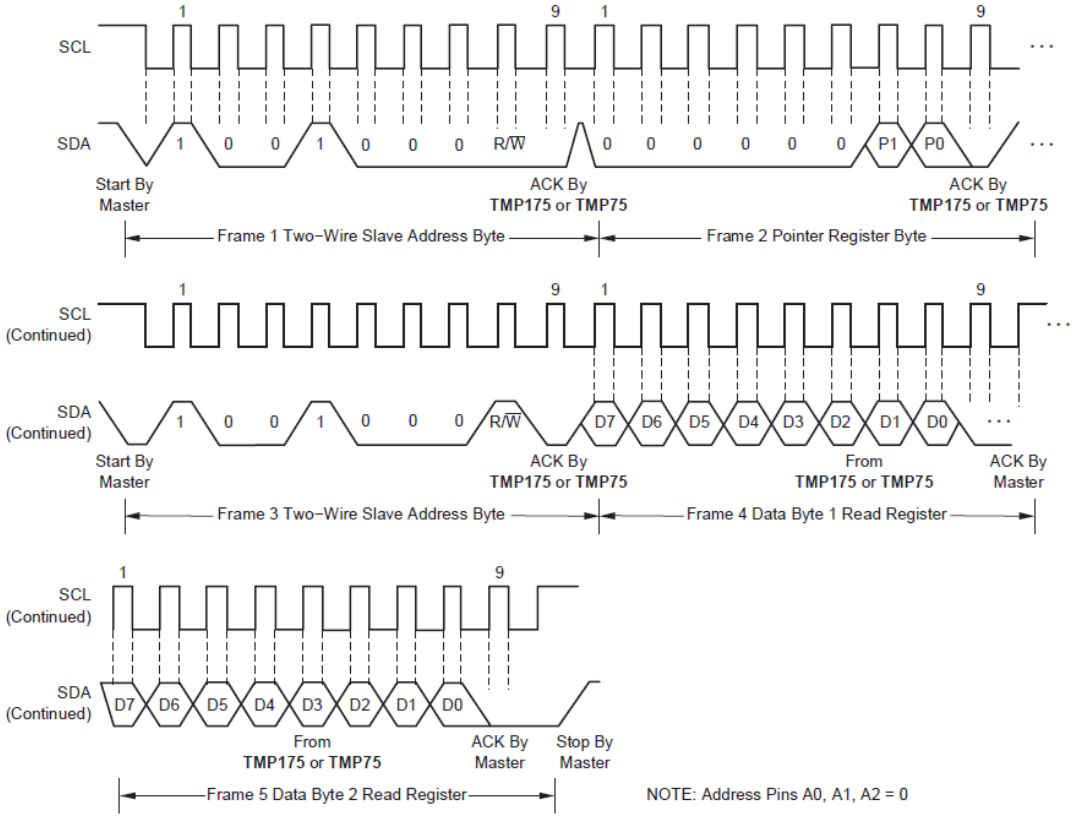


Figura 51: Diagrama de leitura de dados do sensor TMP75. [20]

Para o funcionamento desejado do sensor, foi criada a função apresentada integralmente no Anexo 2, que é chamada a cada 5 segundos e que implementa os diagramas apresentados nas Figura 50 e Figura 51. Pelos comentários (texto a cor verde), pode acompanhar-se o desenvolvimento de cada diagrama.

Uma vez que a fórmula de cálculo para determinar a temperatura devolvida neste sensor não é linear temos: [20]

➔ Se *ValorLido* estiver entre 0x0000 e 0x07FF => Temperatura positiva.

$$T = \frac{ValorLido}{4} \cdot 0{,}25\ (℃) \qquad (4)$$

$$Valor\ Lido = \frac{4 \cdot T}{0{,}25}\ (\text{Decimal}) \qquad (5)$$

Por exemplo: Valor Lido = 0x017A = 378 decimal.

$$T = \frac{378}{4} \cdot 0.25 = \ 23{,}62\ ℃$$

➔ Se *ValorLido* estiver entre 0x0800 e 0x0FFF => Temperatura negativa.

$$T = \frac{(ValorLido - 4096)}{4} \cdot 0{,}25\ (℃) \qquad (6)$$

$$Valor\ Lido = \frac{4 \cdot T}{0{,}25} + 4096\ (\text{Decimal}) \qquad (7)$$

Por exemplo: Valor Lido = 0x0D90 = 3472 decimal.

$$T = \frac{3472 - 4096}{4} \cdot 0.25 = \ -39\ ℃$$

Dado que este sensor apenas admite temperaturas ente -40℃ a 125℃ o valor lido apenas pode tomar valores nos seguintes intervalos:

0xD80 ---------- 0xFFF : (de -40℃ a -0.06℃)

0X000 ---------- 0x7D0 : (de 0℃ a 125℃)

A título de exemplo, seguidamente é apresentado o código da função do programa de monitorização RenPAD que calcula o valor de temperatura obtida pela conversão do valor fornecido por um sensor TMP75, e uma tabela com exemplos de valores de temperatura e respetiva conversão.

```
/// <summary>
/// Converte o valor presente na entrada de um sensor TMP75 (Hexadecimal) em valor
/// de temperatura segundo a fórmula do sensor.
/// </summary>
/// <param name="valorXXX75">Valor a converter.</param>
/// <returns>
/// Valor convertido. Se o valor de entrada for -1 devido a um erro retorna
/// 0xD80 correspondente a -40ºC
///</returns>
public static decimal ConverterHex2GrausTMP75(int valorTMP75)
{
    if (valorTMP75==-1)
    {
        return 0xD80;
    }
    else if (valorTMP75<=0x7FF)
    {
        return (decimal)(valorTMP75 * 0.0625);
    }
    else
    {
        return (decimal)((valorTMP75 - 4096) * 0.0625);
    }
}
```

TEMPERATURE (°C)	DIGITAL OUTPUT (BINARY)	HEX
128	0111 1111 1111	7FF
127.9375	0111 1111 1111	7FF
100	0110 0100 0000	640
80	0101 0000 0000	500
75	0100 1011 0000	4B0
50	0011 0010 0000	320
25	0001 1001 0000	190
0.25	0000 0000 0100	004
0	0000 0000 0000	000
−0.25	1111 1111 1100	FFC
−25	1110 0111 0000	E70
−55	1100 1001 0000	C90

Tabela 10: Exemplo de valores de conversão de temperatura para sensores TMP75. [20]

3.1.5 – Sensor de temperatura e humidade SHT75

Fabricado pela *Sensirion*, estes dispositivos incorporam um sensor de temperatura e um sensor de humidade no mesmo componente.

Alimentados com uma tensão entre 2.4 e 5.5V, e com uma resolução de 12 *bit's* para os valores de humidade e 14 *bit's* para os valores da temperatura, e uma precisão de ±1.8%RH e de ±0.3ºC a 25º estes são os sensores de temperatura mais precisos, e mais dispendiosos, que temos incorporados no produto neste momento.

Cada unidade destes sensores é calibrada individualmente numa câmara húmida de precisão e o valor de calibração armazenado numa memória OTP (*One Time Programable Memory*) para garantir o máximo de fiabilidade possível. Não obstante, caso o sensor opere numa gama de valores fora da recomendada, especialmente para humidades acima dos 80%RH, os valores medidos podem sofrer deturpações como pode ser visualizado na figura seguinte:

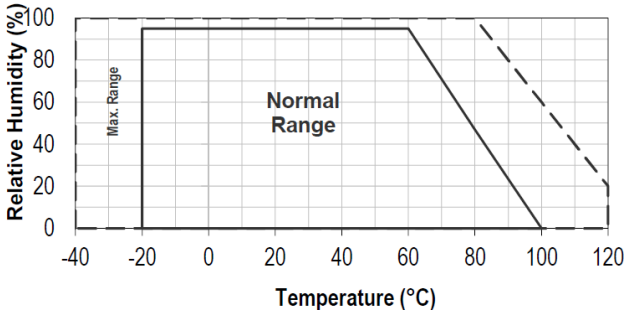
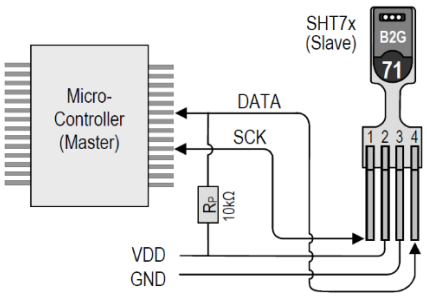


Figura 52:Condições de funcionamento recomendadas para os sensores SHT75. [21]

Este fenómeno é apenas temporário, e os valores regularizam após algum tempo.

A figura seguinte apresenta o *pinout* do sensor e o esquema de ligações recomendado:



Pin	Name	Comment
1	SCK	Serial Clock, input only
2	VDD	Source Voltage
3	GND	Ground
4	DATA	Serial Data, bidirectiona

Figura 53: *Pinout* e esquema de ligações típico do sensor SHT75. [21]

Também para este sensor foi criada uma placa de circuito impresso, semelhante à do sensor TMP75 apresentada na secção anterior, onde foi incluído, além da resistência de *pull-up* entre os pinos de VDD e DATA como recomendado na figura anterior, um regulador de tensão que transforma a tensão de alimentação fornecida à placa (5 ou 12V conforme a seleção) em 3,3V e um condensador de filtragem:

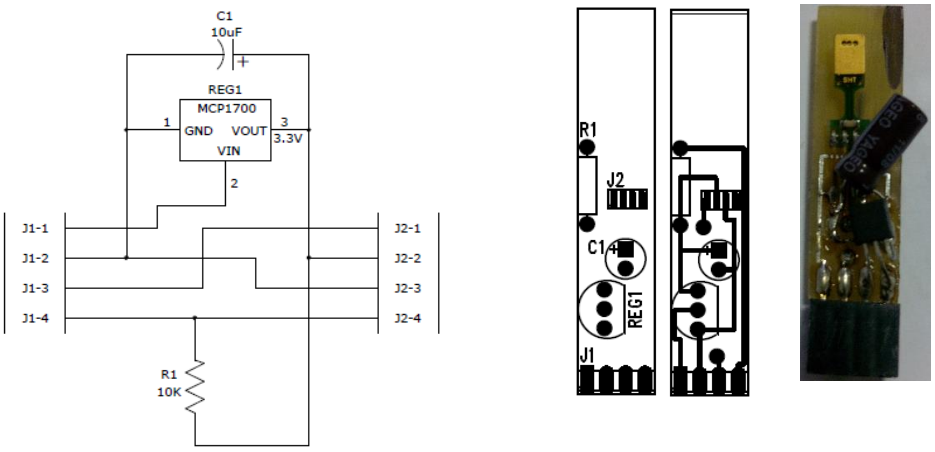


Figura 54: Placa de circuito impresso para sensor SHT75: Esquema de ligações, e placa de circuito impresso.

Da mesma forma que os sensores TMP75, também estes sensores utilizam um protocolo de comunicações proprietário apesar de compatível com I^2C. A implementação do protocolo foi implementado no microcontrolador por meio de *software*.

O protocolo de comunicações é implementado com recurso a duas linhas: SCK e DATA.

- SCK (*Serial Clock Input*) – Utilizado para sincronizar a comunicação entre o microprocessador e o sensor.
- DATA (*Serial DATA*) – Este pino (*tri-state*) é utilizado para transmitir dados de e para o sensor.

Para enviar um comando para o sensor, o sinal DATA é válido no bordo ascendente de SCK e deverá manter-se estável enquanto SCK permanece no nível lógico alto. Após a ocorrência do bordo descendente de SCK o valor de DATA pode ser alterado. Para uma maior segurança deve-se aguardar algum tempo (T_{SU} e T_{HO}) antes e após os bordos ascendentes e descendente de SCK respetivamente, para que as linhas estabilizem.

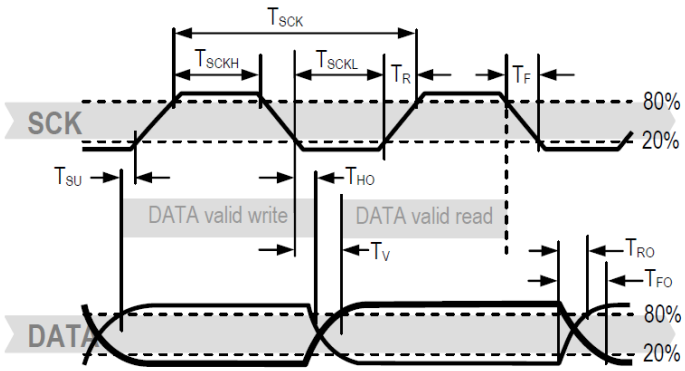


Figura 55: Temporizações na comunicação com o sensor SHT75. [21]

Para a leitura dos dados do sensor, a linha DATA tem dados válidos durante o período T_V, depois de SCK descer para o nível baixo e mantendo-se válidos até que ocorra novamente o próximo bordo descendente de SCK.

O sensor está dotado de um registo de estado (*Status Register*) que armazena as configurações e modos de funcionamento do sensor. Apesar de poder ser alterado, na nossa solução são apenas utilizados os valores configurados por defeito por serem os ideais ao funcionamento pretendido:

Bit	Type	Description	Default	
7		reserved	0	
6	R	End of Battery (low voltage detection) '0' for VDD > 2.47 '1' for VDD < 2.47	X	No default value, bit is only updated after a measurement
5		reserved	0	
4		reserved	0	
3		For Testing only, do not use	0	
2	R/W	Heater	0	off
1	R/W	no reload from OTP	0	reload
0	R/W	'1' = 8bit RH / 12bit Temp. resolution '0' = 12bit RH / 14bit Temp. resolution	0	12bit RH 14bit Temp.

Tabela 11: *Status Register* do sensor SHT75. [20]

O protocolo de comunicações implementa as seguintes fases:

- Para iniciar uma transmissão, é necessário emitir uma sequência de início. Esta é iniciada colocando a linha DATA no nível lógico baixo, enquanto SCK se mantém no nível lógico alto, seguido por uma descida

do nível lógico de SCK, elevando depois o nível da linha DATA enquanto SCK se mantém no nível lógico alto como podemos observar na figura seguinte:

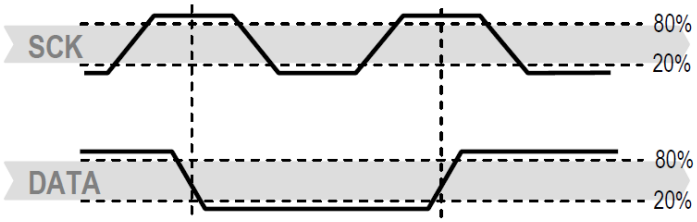


Figura 56: Sequência de "Transmition Start" das comunicações do sensor SHT75. [21]

- Subsequentemente deve enviar-se a sequência de três *bit's* correspondentes ao endereço do sensor. Atualmente, o único valor de endereço suportado para estes sensores é '000'.
- Seguidamente devem enviar-se os cinco *bit's* do comando conforme a tabela seguinte:

Command	Code
Reserved	0000x
Measure Temperature	**00011**
Measure Relative Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values. Wait minimum 11 ms before next command	**11110**

Tabela 12: Lista de comandos de comunicação do sensor SHT75. [21]

- O SHT75 indica a correta receção da cadeia de comando enviada, por meio de um ACK (*acknowledge*) *bit* que consiste na colocação da linha DATA no nível lógico baixo após o bordo descendente do oitavo pulso de SCK. A linha DATA é libertada (e assume o nível lógico alto) após o bordo descendente do nono pulso de SCK.
- Após a emissão de um comando de medição do valor de humidade relativa ou de temperatura ('00000101' ou '00000011'), o controlador terá que aguardar que a medição termine durante um período nunca inferior a 320ms (no nosso caso, em que assumimos o pior cenário).

- Para sinalizar o fim da medição, o sensor coloca a linha de DATA no nível lógico baixo e entra num estado de espera (*Idle*).
- O microprocessador deve aguardar por este sinal de *Data Ready* antes de reiniciar a linha de SCK para ler os dados. O resultado da medida fica armazenado até que seja lido.
- Aquando do pedido dos dados pelo microcontrolador, serão enviados os dois bytes de dados relativos à medição (no contexto do projeto não está a ser utilizado o byte de controlo *CRC checksum*), com o byte mais significativo (*MSB*) a ser transmitido em primeiro lugar. O microcontrolador tem de sinalizar a correta receção de cada byte através de um *ACK*, colocando a linha DATA no nível lógico baixo.
- A comunicação termina depois da leitura do segundo byte de dados referente à medição (*LSB*), colocando a linha DATA no nível lógico alto (escapando assim o envio do *ACK*). Findo este processo, o sensor entra em modo *Sleep*.

Na figura seguinte podemos visualizar toda a sequência de comunicação atrás descrita. Para o exemplo foi considerada uma medição da humidade relativa com o valor em binário "0000 0100 0011 0001" = 1073 em decimal = 35.50%RH (sem compensação de temperatura).

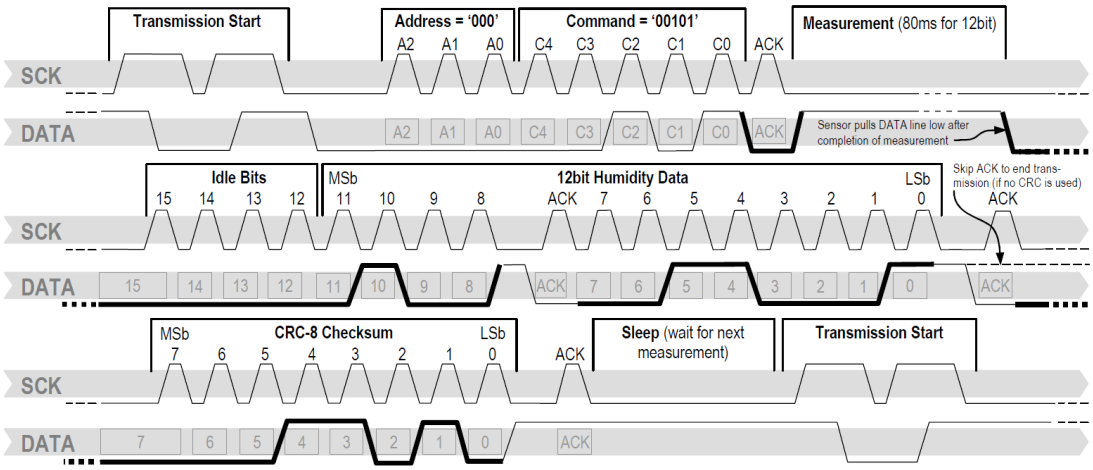


Figura 57: Exemplo do diagrama de comunicações do sensor SHT75. [21]

O dispositivo também implementa uma forma de *reset* do interface série de comunicações, para restabelecimento em caso de falha. Isto é conseguido mantendo a linha DATA no nível lógico alto e pulsando a linha SCK nove ou mais vezes conforme mostrado na figura seguinte:

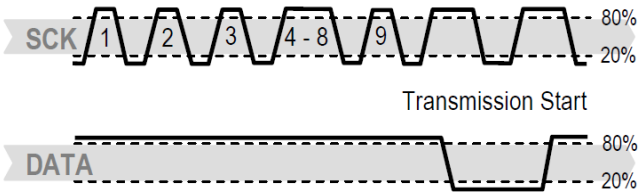


Figura 58: Esquema de *reset* à *interface* série de comunicações do sensor SHT75. [21]

A função criada para leitura dos dados de cada sensor SHT75, é apresentada integralmente no Anexo 3, e que implementa os diagramas apresentados nas Figura 57 e Figura 58. Pelos comentários (texto a cor verde), pode acompanhar-se o desenvolvimento de cada diagrama.

A obtenção do valor da temperatura a partir da conversão do valor devolvido pelo sensor é muito linear e obedece à seguinte fórmula: [21]

$$T = d_1 + d_2 \times SO_T \qquad (8)$$

Com:

SO_T – Valor de temperatura obtido do sensor.

d_1 e d_2 – Coeficientes de conversão de temperatura segundo a tabela seguinte:

VDD	d_1 (°C)	d_1 (°F)
5V	-40.1	-40.2
4V	-39.8	-39.6
3.5V	-39.7	-39.5
3V	-39.6	-39.3
2.5V	-39.4	-38.9

SO_T	d_2 (°C)	d_2 (°F)
14bit	0.01	0.018
12bit	0.04	0.072

Tabela 13: Coeficientes de conversão de temperatura para sensor SHT75. [21]

Assim, e uma vez que a tensão de alimentação do circuito VDD é de 3,3V e a resolução de trabalho do sensor para a temperatura é de 14 *bit's*, d_1 assume o valor correspondente aos 3,5V (o mais próximo na tabela dos de 3,3V de funcionamento) de 39.7ºC e um d_2 de 0,01ºC teremos:

$$T = 0{,}01 \cdot ValorLido - 39{,}7\ (℃) \qquad (9)$$

$$ValorLido = \frac{T+39.7}{0{,}01}\ (Decimal) \qquad (10)$$

Por exemplo: Valor Lido = 0x1814 = 6164 decimal.

$$T = 0{,}01 \cdot 6164 - 39{,}7 = 21{,}94\ ℃$$

A título de exemplo, seguidamente é apresentado o código da função do programa de monitorização RenPAD que calcula o valor da temperatura obtida pela conversão do valor fornecido por um sensor SHT75.

```
/// <summary>
/// Converte o valor presente na entrada de um sensor SHT75Tmp (Hexadecimal) em
/// valor de temperatura segundo a fórmula do sensor.
/// </summary>
/// <param name="valorSHT75Tmp">Valor a converter.</param>
/// <returns>Valor convertido.</returns>
public static decimal ConverterHex2GrausSHT75Tmp(int valorSHT75Tmp)
{
    decimal valorEnviar;
    //if (valorSHT75Tmp>0x3FFF)
    //{
    //    valorSHT75Tmp = 0x3FFF;
    //}
    //else if (valorSHT75Tmp <0)
    //{
    //    valorSHT75Tmp = 0;
    //}
    valorEnviar = (decimal)(valorSHT75Tmp * 0.01 - 39.7);

    if (valorEnviar > (decimal)124.13)
    {
        valorEnviar = (decimal)124.13;
    }
    else if (valorEnviar < (decimal)-39.7)
    {
        valorEnviar = (decimal)39.7;
    }
    return valorEnviar;
}
```

Já a obtenção do valor da humidade a partir da conversão do valor devolvido pelo sensor não é tão linear, existindo mesmo duas fórmulas de cálculo para esta conversão.

A fórmula mais complexa, à qual chamo de fórmula compensada, deve ser utilizada sempre que os valores de temperatura puderem diferir significativamente dos 25ºC, em que esse valor da temperatura obtido previamente pelo sensor é utilizado para a compensação da referida não-linearidade do sensor: [21]

$$RH_{true} = (T_{°C} - 25) \times (t_1 + t_2 \times SO_{RH}) + RH_{linear} \quad (11)$$

Com:

$T_{°C}$ – Temperatura real obtida numa medição anterior e calculada pela fórmula (9).

SO_{RH} – Valor da humidade obtido do sensor.

RH_{linear} – Valor de humidade relativa, obtido através da conversão do valor lido do sensor (SO_{RH}) pela fórmula (15).

t_1 e t_2 – Coeficientes de compensação de temperatura segundo a tabela seguinte:

SO_{RH}	t_1	t_2
12 bit	0.01	0.00008
8 bit	0.01	0.00128

Tabela 14: Coeficientes de compensação de temperatura para obtenção da humidade relativa pela fórmula compensada do sensor SHT75. [21]

Assim teremos as seguintes fórmulas compensadas de conversão do valor lido em percentagem de humidade relativa:

$$RH_{True} = \left(T_{Actual\ (℃)} - 25\right) \cdot (0{,}01 + 0{,}00008 \cdot ValorLido) + RH_{Linear}\ (\%RH) \quad (12)$$

$$ValorLido = \frac{-b+\sqrt{b^2-4ac}}{2a}\ (Decimal) \quad (13)$$

$$com: \begin{cases} a = -1{,}5955 \cdot 10^{-6} \\ b = 0{,}0367 + 0{,}00008 \cdot \left(T_{Actual\ (℃)} - 25\right) \\ c = -2{,}0468 + 0{,}01 \cdot \left(T_{Actual\ (℃)} - 25\right) - RH_{True} \end{cases}$$

Dada a maior complexidade na obtenção dos valores pela fórmula anterior, e uma vez que a nossa gama de temperaturas nunca varia muito dos 25ºC, utilizamos neste projeto a fórmula simples (não compensada) de conversão dos valores obtidos do sensor

em humidade relativa. Esta fórmula, nas condições de temperatura referidas, apresenta uma linearidade elevada que pode ser observada na figura seguinte:

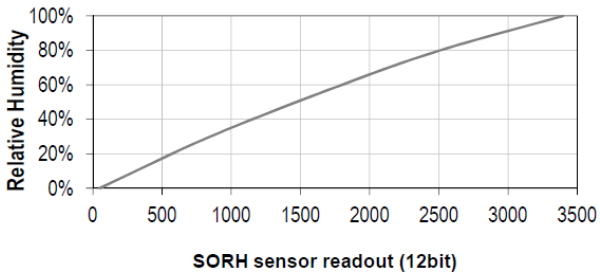


Figura 59: Curva de conversão do valor obtido do sensor SHT75 (SO_{RH}) em humidade relativa. [21]

A fórmula simplificada de conversão dos valores lidos do sensor SHT75 para humidade relativa, apresenta a seguinte forma: [21]

$$RH_{linear} = c_1 + c_2 \times SO_{RH} + c_3 \times SO_{RH}^2 \ (\%RH) \quad (14)$$

Com:

SO_{RH} – Valor da humidade obtido do sensor.

c_1, c_2 e c_3 - Coeficientes de compensação de humidade segundo a tabela seguinte:

SO_{RH}	c_1	c_2	c_3
12 bit	-2.0468	0.0367	-1.5955E-6
8 bit	-2.0468	0.5872	-4.0845E-4

Tabela 15: Coeficientes de compensação de humidade do sensor SHT75. [21]

Uma vez que a resolução de trabalho do sensor para a humidade é de 12 *bit's*, os valores dos coeficientes de humidade selecionados são:

c_1 = -2,0468, c_2 = 0.0367 e c_3 = -1,5955E-6

E com estes coeficientes, a fórmula assume a seguinte configuração:

$$RH_{Linear} = -2{,}0468 + 0{,}0367 \cdot ValorLido - 1{,}5955 \cdot 10^{-6} \cdot ValorLido^2 \ (\%RH) \quad (15)$$

$$ValorLido = \frac{0.0367 - \sqrt{0{,}0367^2 - 4 \cdot \left(-1{,}5955 \cdot 10^{-6} \cdot (-2{,}0468 - RH_{Linear})\right)}}{3{,}191 \cdot 10^{-6}} \ (Decimal) \quad (16)$$

Por exemplo: Se ValorLido = 0x043A = 1082 decimal.

$$RH_{Linear} = -2{,}0468 + 0{,}0367 \cdot 1082 - 1{,}5955 \cdot 10^{-6} \cdot 1082^2 = 35{,}79\ \%$$

A título de exemplo, podemos observar a função criada no programa de monitorização RenPAD para implementar esta conversão, apresentada a seguir.

```
/// <summary>
/// Converte o valor presente na entrada de um sensor SHT75Hum (Hexadecimal) em
/// valor de temperatura segundo a fórmula do sensor.
/// </summary>
/// <param name="valorSHT75Hum">Valor a converter.</param>
/// <returns>Valor convertido.</returns>
public static decimal ConverterHex2RHLinearSHT75Hum(int valorSHT75Hum)
{
    decimal valorEnviar = 0;

    //if (valorSHT75Hum > 0x0CA4)
    //{
    //    valorSHT75Hum = 0x0CA4;
    //}
    //else if (valorSHT75Hum < 0x38)
    //{
    //    valorSHT75Hum = 0x38;
    //}

    valorEnviar=(decimal)(-2.0468+(0.0367*valorSHT75Hum)-(1.5955*
                                Math.Pow(10, 6)*valorSHT75Hum*valorSHT75Hum));

    if (valorEnviar < 0)
    {
        valorEnviar = 0;
    }
    else if (valorEnviar > 100)
    {
        valorEnviar = 100;
    }

    return valorEnviar;
}
```

Uma vez que, como referido anteriormente, este sensor funciona a 14 *bit's* para medição da temperatura e 12 *bit's* para medição da humidade, segundo as fórmulas de conversão e os limites físicos, os valores mínimos e máximos admitidos são os seguintes:

- Intervalo admissível para temperatura:

  0x0000----------0x3FFF : (de -39,7℃ a 124,13℃)

- Intervalo admissível para humidade:

  0x038----------0xCA4 : (de 0% a 100%)

3.1.6 – Fonte de alimentação comutada 230VAC-15VDC

Neste projeto é utilizada uma fonte de alimentação de 230V AC para 15V DC.

A fonte selecionada inicialmente foi uma fonte comutada *Cicon* CFM1003S que admite tensões de entrada entre 85 e 264VAC e fornece uma tensão de saída de 15VDC com uma corrente máxima de 670mA e um rendimento de 76%. As especificações podem ser observadas na tabela seguinte:

MODEL	OUTPUT VOLTAGE	MAX. LOAD	MIN. LOAD	RIPPLE & NOISE	VOLTAGE ACCURACY	LINE REGULATION	LOAD REGULATION	%EFF
CFM1003S	15 V	670mA	0A	1%	± 1%	± 0.5%	± 1%	76% Typ.

Tabela 16: Especificações técnicas da fonte comutada CFM1003S. [22]

Apesar da escolha inicial recair sobre esta referência devido ao seu mais atrativo custo, esta fonte pode ser trocada em qualquer altura por um modelo equivalente de qualquer outro fabricante que garanta uma tensão de saída superior a 12V considerada a tensão mínima admissível de funcionamento do circuito.

Figura 60: *Cicon* CFM1003S.

3.1.7 – Conversor DC-DC – 15VDC-5VDC

A escolha do conversor DC-DC de 15V para 5V recaiu sobre o componente SI8050S.

Este conversor distingue-se da concorrência, nomeadamente do LM7805, pela sua maior eficiência e poder de dissipação, além de uma regulação mais precisa da tensão de saída.

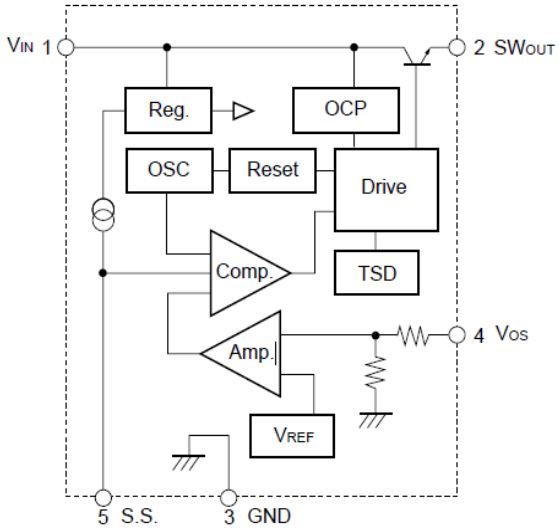


Figura 61: Conversor DC-DC SI8050S e seu diagrama de blocos. [23]

Parameter	Symbol	Ratings					Unit
		SI-8033S	SI-8050S	SI-8090S	SI-8120S	SI-8150S	
DC Input Voltage Range	V_{IN}	5.5 to 28	7 to 40	12 to 40	15 to 40	18 to 40	V
Output Current Range	Io	0 to 3.0					A
Operating Junction Temperature Range	T_{jop}	–30 to +125					°C

Tabela 17: Parâmetros recomendados de funcionamento. [23]

Com uma tensão de entrada admissível entre os 7V e 40V, uma tensão de saída regulada entre os 4,8V e 5,2V, uma corrente de saída de 3A e um rendimento máximo entre 79 a 91%, este conversor necessita de mais quatro componentes que garantem o seu adequado funcionamento, dispostos como se pode ver na figura seguinte. A disposição destes componentes na placa de circuito impresso, e as áreas de ocupação mencionadas na Figura 63 são um fator de extrema importância na determinação do rendimento global do conversor.

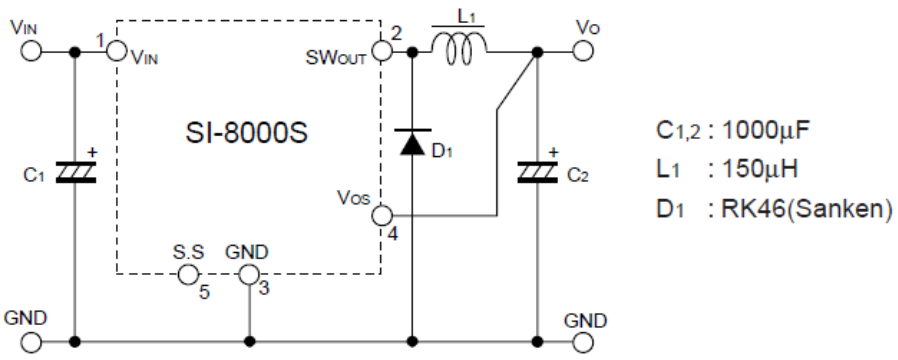


Figura 62: Componentes necessários para o adequado funcionamento do conversor SI8050S [23]

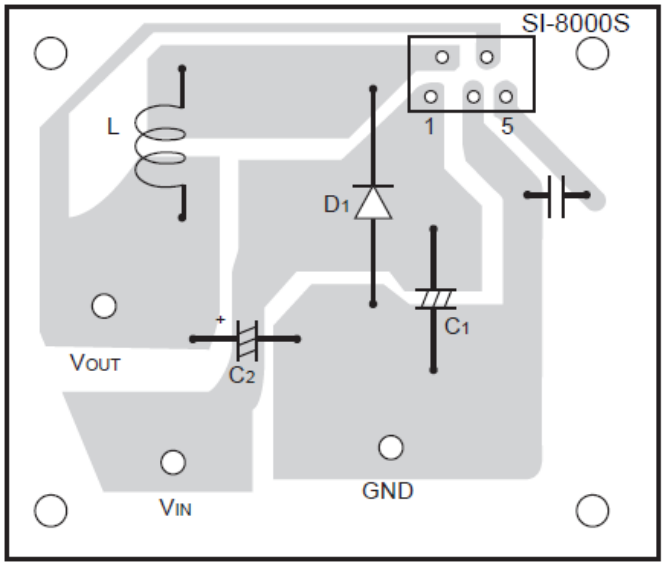


Figura 63: Exemplo da correta distribuição dos componentes necessários para o adequado funcionamento do conversor SI8050S numa placa de circuito impresso. [23]

3.1.8 – Analisador da tensão de alimentação DS1231

O DS1231 é um componente que dispõe de um mecanismo interno de supervisão da tensão de alimentação do microcontrolador. Caso esta ultrapasse uma tolerância predefinida, esse evento é sinalizado indicando ao processador que a sua tensão de alimentação está com problemas. Uma vez que a placa de aquisição de dados RenPAD dispõe de uma bateria auxiliar, que garante o funcionamento do equipamento durante falhas de energia, este componente é utilizado para avaliar o estado da descarga dessa bateria, e logo que o seu valor baixe da predefinição de 7,5V, é enviado um sinal para o microprocessador indicando que este dispõe de pouco tempo de operação, para que termine

as tarefas críticas e se desligue após todos os processos estarem devidamente terminados, eliminando assim a possibilidade de uma eventual corrupção de dados devido à falta abrupta de energia.

Este dispositivo garante ainda um *reset* "limpo" ao microcontrolador, eliminando a possibilidade de transitórios ocorridos no botão de pressão, e garantindo o tempo mínimo de estabilização sem tensão necessário à sua correta e completa reinicialização.

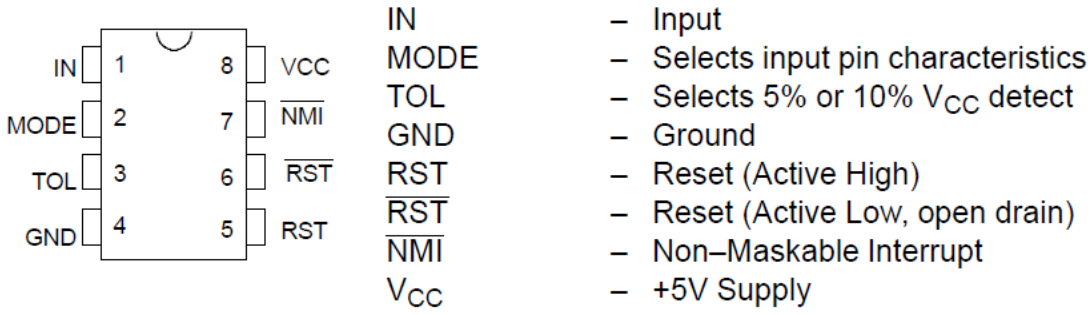


Figura 64: *Pinout* DS1231 DIP8 *Package*. [24]

Na figura seguinte podemos observar o modo de funcionamento do DS1231 implementado para este projeto:

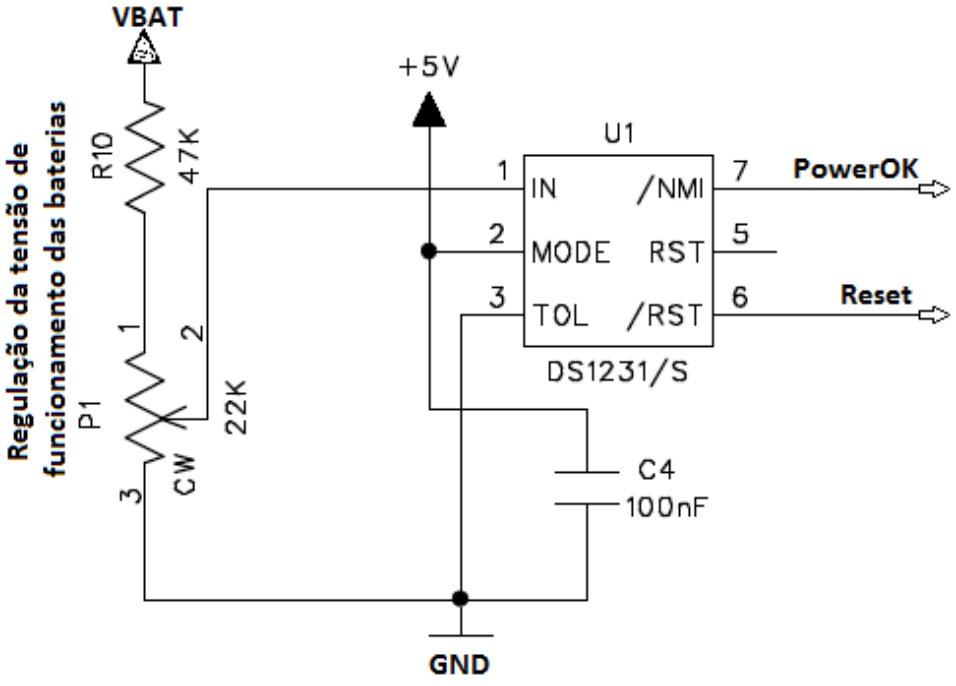


Figura 65: Modo de funcionamento do DS1231.

Através da resistência R10 e do Potenciómetro P1 é regulada uma tensão em IN que faz disparar a interrupção interna do DS1231 sinalizada em /NMI sempre que a tensão

da bateria baixe da tolerância de funcionamento de 7,5V, irá avisar o microcontrolador que a bateria se encontram perto do limite mínimo de funcionamento.

O controlo do sinal de *reset* do microcontrolador é realizado sempre que a linha TOL cai abaixo dos 10% de VCC, como no caso de uma falha completa de energia. Nesse momento, a linha /RST (e subsequentemente o *reset* do microprocessador) fica "agarrada" sendo liberada somente após a completa estabilização do valor de tensão na linha dos 5V.

NOTA: O valor predefinido de tolerância da tensão da bateria mencionado de 7,5V apenas é válido para o pack de baterias de sete células de NI-CD de 1,2V cada, considerado adequado para o funcionamento da placa de aquisição de dados RenPAD. Caso se utilizem outros packs de baterias, determinado por um modo de funcionamento ou especificações mais exigentes de trabalho, este valor de referência terá que ser alterado atendendo ao novo limite mínimo de tensão suportado pelo pack sem que este entre em rotura.

3.1.9 - Referência de tensão de precisão MCP1525

Este componente assegura a calibração do conversor analógico-digital MCP3208 regulando a sua tensão de saída consoante a temperatura ambiente e injetando essa tensão em VREF do MCP3208.

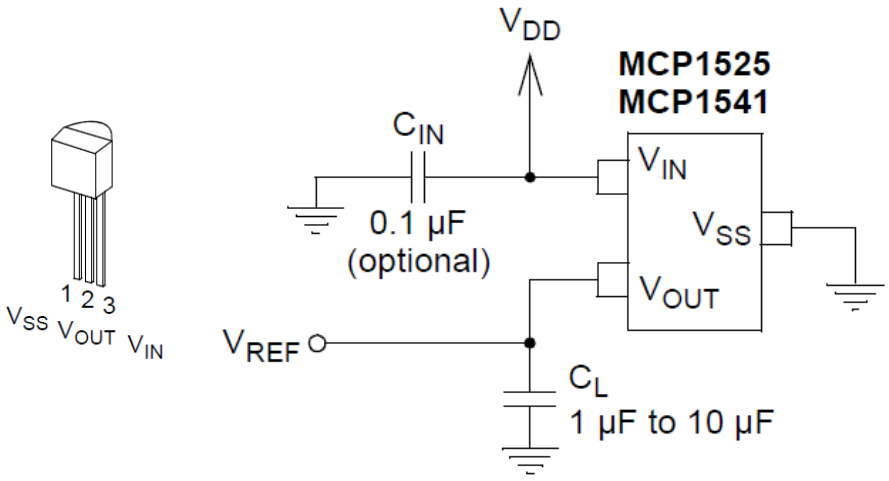


Figura 66: *Pinout* da referência de tensão de precisão MCP1525 e esquema típico de montagem. [25]

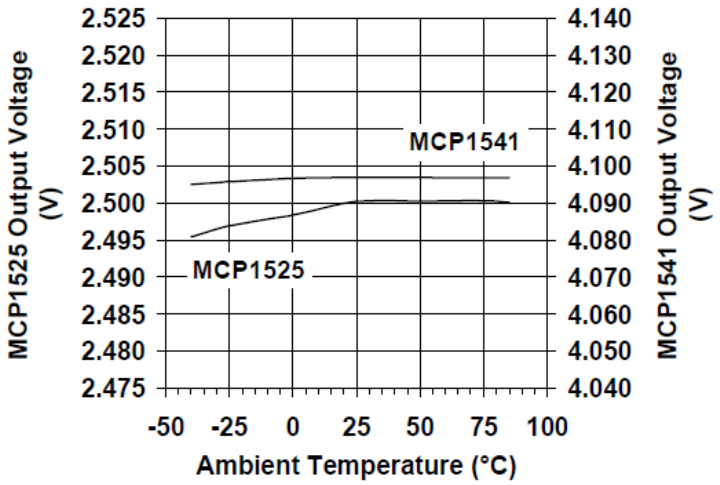


Figura 67: Tensão de saída *versus* temperatura ambiente da referência de tensão de precisão MCP1525. [25]

3.1.10 – *Mosfet* de canal P (SI9435BDY)

Este *mosfet* de canal P é utilizado no circuito como interruptor digital. Quando tem a gate polarizada com uma tensão de 0V, está à condução. Caso a tensão na gate suba para 5V o *mosfet* entra ao corte desligando a alimentação do circuito.

Este componente apresenta o seguinte *pinout*:

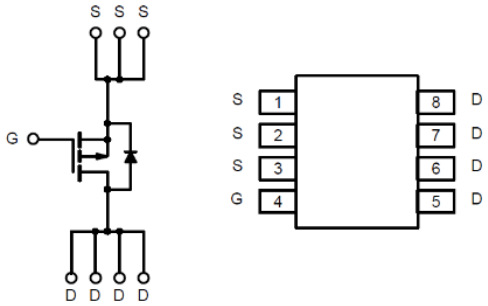


Figura 68: *Pinout* do *mosfet* SI9435BDY SO8 *Package*. [26]

E as suas especificações são as seguintes:

V_{DS} (V)	$r_{DS(on)}$ (Ω)	I_D (A)
–30	0.055 @ V_{GS} = –10 V	±5.1
	0.07 @ V_{GS} = –6 V	±4.6
	0.105 @ V_{GS} = –4.5 V	±3.6

Tabela 18: Especificações do *mosfet* de canal P SI9435BDY. [26]

3.2 – Descrição das fichas externas

Na tabela seguinte é apresentada a descrição de cada ficha de ligação a dispositivos externos, denominadas neste projeto por fichas externas, e a função de cada pino da respetiva ficha.

Ficha	Pino	Descrição
F1 - Entradas analógicas	1	+5V/+12V
	2	**AIN0**
	3	AGND
	4	ND
	5	+5V/+12V
	6	**AIN1**
	7	AGND
	8	ND
	9	+5V/+12V
	10	**AIN2**
	11	AGND
	12	ND
	13	+5V/+12V
	14	**AIN3**
	15	AGND
	16	ND
	17	+5V/+12V
	18	**AIN4**
	19	AGND
	20	ND
	21	+5V/+12V
	22	**AIN5**
	23	AGND
	24	ND
	15	+5V/+12V
	16	**AIN6**
	27	AGND
	28	ND
	29	+5V/+12V
	30	**AIN7**
	31	AGND

Ficha	Pino	Descrição
F2 - Entradas digitais	1	+5V/+12V
	2	**DIN0**
	3	GND
	4	ND
	5	+5V/+12V
	6	**DIN1**
	7	GND
	8	ND
	9	+5V/+12V
	10	**DIN2**
	11	GND
	12	ND
	13	+5V/+12V
	14	**DIN3**
	15	GND
	16	ND
	17	+5V/+12V
	18	**DIN4**
	19	GND
	20	ND
	21	+5V/+12V
	22	**DIN5**
	23	GND
	24	ND
	25	+5V/+12V
	26	**DIN6**
	27	GND
	28	ND
	29	+5V/+12V
	30	**DIN7**
	31	GND

Ficha	Pino	Descrição
F3 - Saidas digitais	1	**DOUT0 +**
	2	**DOUT0 -**
	3	**DOUT1 +**
	4	**DOUT1 -**
	5	ND
	6	**DOUT2 +**
	7	**DOUT2 -**
	8	**DOUT3 +**
	9	**DOUT3 -**
	10	ND
	11	**DOUT4 +**
	12	**DOUT4 -**
	13	**DOUT5 +**
	14	**DOUT5 -**
	15	ND
	16	**DOUT6 +**
	17	**DOUT6 -**
	18	**DOUT7 +**
	19	**DOUT7 -**
F4 - Outros	1	+5V/+12V
	2	GND
	3	SHT75 1 - CLK
	4	SHT75 1 - DATA
	5	ND
	6	PC.0 - TXD
	7	PC.1 - RXD
	8	PC.2 - TXC
	9	PC.3 - RXC
	10	ND
	11	+5V/+12V
	12	GND
	13	SHT75 2 - CLK
	14	SHT75 2 - DATA
	15	ND
	16	TMP75 CLK
	17	TMP75 DATA
	18	+5V/+12V
	19	GND

Tabela 19: Descrição das fixas externas da placa de aquisição de dados RenPAD.

3.3 – Descrição das funções dos portos do microprocessador Rabbit 3000

Nas tabelas seguintes é apresentada a relação de cada pino das fichas externas com o respetivo porto do microprocessador Rabbit 3000 onde este está conectado, e o modo de configuração desse porto. A primeira tabela apresenta essa relação por ficha, a segunda apresenta essa relação por porta.

Descrição por ficha			
Relacionado com a ficha	**Porta**	**Descrição**	**Sentido a definir no microprocessador**
F1 - Entradas analógicas	PB.3	MCP3208 - CS / SHDN	OUT
	PB.4	MCP3208 - CLK	OUT
	PB.5	MCP3208 - Din	OUT
	PB.7	MCP3208 - Dout	IN
F2 - Entradas digitais	PE.0	DIN0	IN
	PE.1	DIN1	IN
	PE.4	DIN2	IN
	PE.5	DIN3	IN
	PE.7	DIN4	IN
	PF.0	DIN5	IN
	PF.1	DIN6	IN
	PF.4	DIN7	IN
F3 - Saídas digitais	PA.0	DOUT0	OUT
	PA.1	DOUT1	OUT
	PA.2	DOUT2	OUT
	PA.3	DOUT3	OUT
	PA.4	DOUT4	OUT
	PA.5	DOUT5	OUT
	PA.6	DOUT6	OUT
	PA.7	DOUT7	OUT
F4 - Outras	PB.0	SHT75 1 - CLK	OUT
	PB.2	SHT75 1 - DATA	IN/OUT
	PC.0	A definir - TXD	?OUT
	PC.1	A definir - RXD	?IN
	PC.2	A definir - TXC	?OUT
	PC.3	A definir - RXC	?IN
	PD.4	SHT75 2 - CLK	OUT
	PD.5	SHT75 2 - DATA	IN/OUT
	PG.6	TMP75 CLK	OUT
	PG.7	TMP75 DATA	IN/OUT
Utilitários	PF.5	PowerLineOff	IN
	PF.6	PowerOk	IN
	PF.7	PowerOn	OUT

Tabela 20: Descrição por ficha das funções nos portos do microprocessador Rabbit 3000.

Descrição por porto			
Porta	**Sentido a definir no microprocessador**	**Descrição**	**Relacionado com a ficha**
PA.0	OUT	DOUT0	F4
PA.1	OUT	DOUT1	F3
PA.2	OUT	DOUT2	F3
PA.3	OUT	DOUT3	F3
PA.4	OUT	DOUT4	F3
PA.5	OUT	DOUT5	F3
PA.6	OUT	DOUT6	F3
PA.7	OUT	DOUT7	F3
PB.0	OUT	SHT75 1 - CLK	F4
PB.2	IN/OUT	SHT75 1 - DATA	F3
PB.3	OUT	MCP3208 - CS / SHDN	F1
PB.4	OUT	MCP3208 - CLK	F1
PB.5	OUT	MCP3208 - DIn	F1
PB.7	IN	MCP3208 - DOut	F1
PC.0	?OUT	A definir - TXD	F4
PC.1	?IN	A definir - RXD	F4
PC.2	?OUT	A definir - TXC	F4
PC.3	?IN	A definir - RXC	F4
PD.4	OUT	SHT75 2 - CLK	F4
PD.5	IN/OUT	SHT75 2 - DATA	F4
PE.0	IN	DIN0	F2
PE.1	IN	DIN1	F2
PE.4	IN	DIN2	F2
PE.5	IN	DIN3	F2
PE.7	IN	DIN4	F2
PF.0	IN	DIN5	F2
PF.1	IN	DIN6	F2
PF.4	IN	DIN7	F2
PF.5	IN	PowerLineOff	Utilitários
PF.6	IN	PowerOk	Utilitários
PF.7	OUT	PowerOn	Utilitários
PG.6	OUT	TMP75 CLK	F4
PG.7	IN/OUT	TMP75 DATA	F4

Tabela 21: Descrição por porta das funções nos portos do microprocessador Rabbit 3000.

3.4 – Informações adicionais – Descrição das Linhas, *Led's* e *Jumper's*

Na tabela seguinte são apresentadas todas as linhas, *led's* e *jumper's* constantes nos esquemas do capítulo 2.5. São também descritos os seus modos de funcionamento e as configurações possíveis para cada linha.

PowerLineOff	Indica o estado da linha de alimentação (220V).	0	Linha Ok
		1	Linha desligada
PowerOk	Indica quando a tensão de alimentação do circuito está abaixo do limiar (útil para detetar quando a tensão da bateria está baixa demais obrigando a desligar o circuito).	0	Necessário desligar o circuito o mais rápido possível.
		1	Tensão acima do valor mínimo.
PowerOn	Linha que permite ao CPU ligar ou desligar o circuito. Para, por exemplo, desligar o circuito em caso de se atingir a tensão mínima da bateria (ver capítulo 2.1.8).	0	Desligar o circuito
		1	Manter o circuito em funcionamento
Led Verde - D5	Indicação visual de bateria carregada.	Muito brilhante - Bateria completamente carregada	
		Pouco brilhante - Bateria em carga.	
Led Amarelo - D4	Indicação de circuito ligado	Aceso - Circuito ligado.	
		Apagado - Circuito desligado.	
Led Vermelho - D6	Indicação de tensão abaixo do limiar mínimo.	Aceso - Tensão de alimentação com valor abaixo do limiar mínimo.	
		Apagado - Tensão de alimentação com valor normal.	
Led Vermelho - D9	Indicação de circuito desligado devido a picos de tensão na linha de VCC.	Aceso - Circuito desligado devido à ocorrência de picos em VCC.	
		Apagado - Circuito em funcionamento normal.	
X1	Permite desligar a alimentação efetuada pela bateria.	Quando retirado a baterias fica desconectada do circuito	
X2	Permite medir a tensão aos terminais da bateria ou servir de equalizador das tensões de referência entre o pc e o circuito.	Medir a tensão entre o pino 1 e 2 ou ligar a massa do PC ao pino 1 de X2.	
X3	Permite medir a tensão de alimentação de 12v ou ligar uma ponta lógica.	---	
X4	Permite indicar ao circuito se este deve ser mandado desligar pelo CPU ou por *hardware* quando a tensão de alimentação se aproximar do limiar mínimo.	1-2: Desligar por *hardware* assim que NMI desça ao valor lógico 0	
		2 -3: Desligar por *software* (por PowerOn) - permite desligar o circuito apenas no momento pretendido.	
X5	Divide a massa da parte analógica da massa da parte digital. Deve permanecer sempre ligado.	---	
X6..X32	Permite ligar os periféricos externos à alimentação de 12V ou 5V.	1-2: Ligação à alimentação de 5V.	
		2-3: Ligação à alimentação de 12V.	

Tabela 22: Informações adicionais – Descrição das linhas, *Led's* e *Jumper's*

3.5 – Esquemas gerais da placa de circuito impresso

Nesta secção é apresentado o esquemático da placa de circuito impresso desenvolvida.

Para uma mais fácil leitura e compreensão dos esquemas idealizados, estes foram divididos em seis secções: alimentação, ligações ao microcontrolador, entradas digitais, entradas analógicas, saídas digitais e fichas, onde cada secção é individualmente descrita.

Os *layout's* da placa de circuito impresso também estão disponíveis e podem ser consultados no anexo 4.

3.5.1 - Alimentação

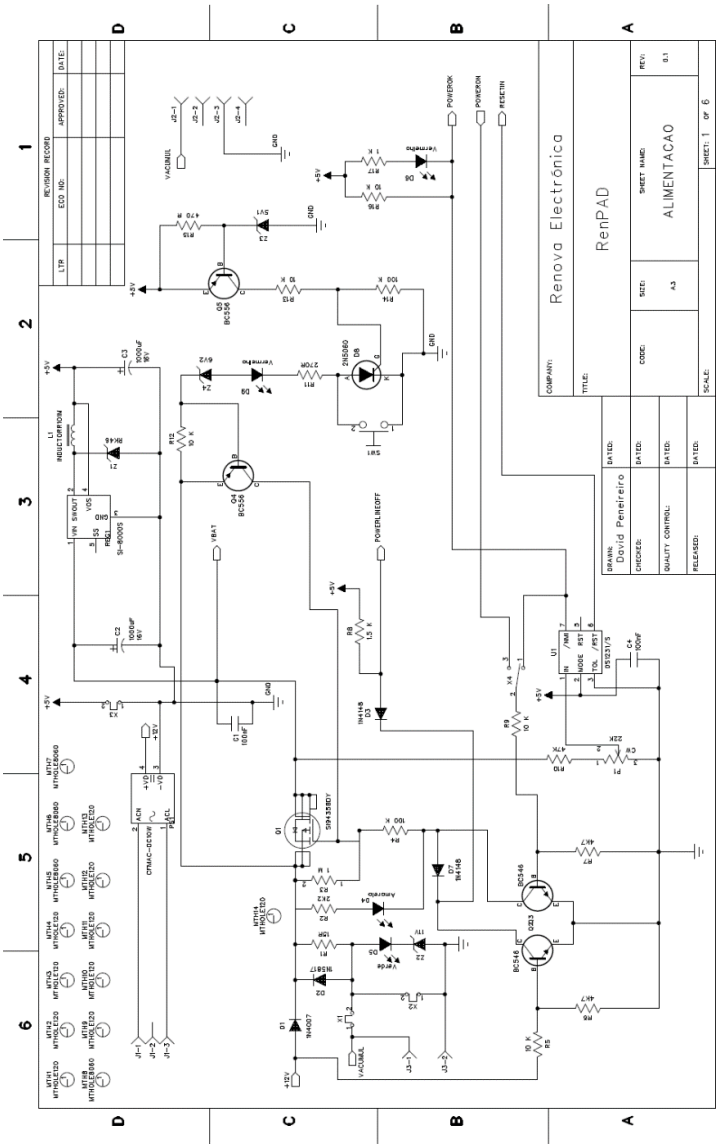


Figura 69: Esquema geral da secção de alimentação.

Na figura anterior é apresentada a secção do esquema, designada por secção de alimentação, que controla todo o processo relacionado com a alimentação de energia ao dispositivo desde a conversão dos 230VAC nas várias tensões DC de serviço, ao controlo de *reset's* e picos de alimentação na linha dos 5VDC, à sinalização de eventos anómalos nas linhas, carregamento da bateria associada à placa entre outros.

Uma vez que esta é a secção com esquemas mais complexos e facilmente divisíveis em subsecções, são descritas nas secções seguintes cada subsecção do esquema de modo a facilitar a sua leitura e compreensão.

3.5.1.1 – Conversão 230VAC – 12VDC

Esta conversão de tensão é obtida de forma direta por meio da fonte de alimentação comutada PS1 (ver secção 3.1.6):

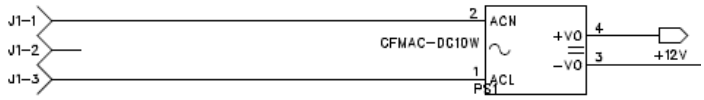


Figura 70: Fonte comutada 220VAC-12VDC.

3.5.1.2 – Carregador da Bateria

Os 12V obtidos na conversão anterior são a entrada do circuito de baixa tensão até ao interruptor digital (ver secção 3.1.10) e são utilizados para carregar a bateria depois de passar pelo díodo de proteção contra polaridade inversa D1.

O carregamento da bateria é controlado pela resistência R1 e o díodo de *zener* Z2. Estes componentes são calculados para que a corrente de carga da bateria seja geralmente muito baixa, com intuito de prolongar a longevidade da mesma. Geralmente este processo apenas terá inconveniente caso exista uma ocorrência muito regular de falhas de energia, ou esta falhe por grandes períodos de tempo, que levem à completa descarga da bateria. Caso isso ocorra, devido ao baixo poder de carregamento da mesma num curto espaço de tempo, o dispositivo terá que permanecer ligado à energia durante um período longo para que a bateria recarregue o suficiente para garantir o correto funcionamento do equipamento durante uma nova falha de energia.

Por norma, a placa de aquisição de dados RenPAD está equipada com uma bateria constituída por um *pack* de sete pilhas NI-CD de 1,2V. Os valores definidos no esquema

têm por base este *pack* de pilhas. Caso se opte por uma configuração de pilhas diferentes, ou por baterias de qualquer outro tipo (chumbo por exemplo), devem recalcular-se os valores de R1 e Z2 por forma a limitar a corrente de carga da bateria conforme as novas especificações. Teremos também que ter em atenção que alguns tipos de pilhas recarregáveis atualmente no mercado não são compatíveis com este processo de carga ou mesmo que carreguem e funcionem, o seu período de vida útil poderá ser largamente comprometido.

O processo de carga da bateria apresentado implica algumas perdas em R1 e especialmente em Z2. Note-se que Z2 estará à condução sempre que a tensão de alimentação esteja presente, pois a tensão aos seus pinos será sempre superior a 11V. Apesar de ser essa a sua função, limitando desta forma a tensão de carga VACUMUL, este processo básico de regulação de tensão tem inerentes as referidas perdas "significativas". Quanto mais carregada estiver a bateria, maior será a corrente escoada em Z2, mais o *led* D5 brilha e maiores serão as perdas neste circuito. Esta é no entanto uma desvantagem assumida, dada a simplicidade e fiabilidade deste circuito que compensa largamente o investimento em circuitos específicos de controlo de carga de baterias com rendimentos superiores, mas com custos muito mais consideráveis.

Já durante processo de descarga da bateria, quando falha a tensão de alimentação e o circuito passa a funcionar apoiado na energia fornecida por esta, a corrente é escoada através do díodo D2 evitando passar em R1 e uma vez que a tensão deverá ser inferior tensão de condução de Z2 este manter-se-á ao corte, limitando assim as perdas de descarga.

A bateria tanto pode ser ligada na ficha J2 como na ficha J3.

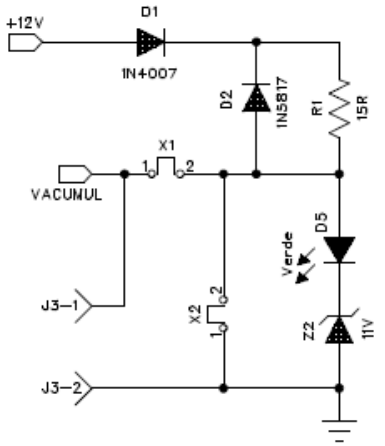


Figura 71: Carregador da bateria.

3.5.1.3 – Interruptor digital

O que designo por interruptor digital não é mais do que um *mosfet* SI9435GDY de canal P (Q1), colocado em série no circuito de alimentação de 12V que, quando tem a *gate* polarizada com 0V está à condução, garantindo alimentação aos restantes circuitos. Caso a tensão na *gate* suba, este entra ao corte e deixa de conduzir, cortando assim a alimentação aos restantes circuitos. Ver como é feito o controlo da *gate* nas secções seguintes.

A tensão da *source* é a tensão de 12V fornecida tanto pelo conversor AC-DC (designada por 12V), como pela bateria (designada por VACUMUL) quando a primeira não está presente e a tensão do *dreno* (designada no circuito por VBAT) é a utilizada para alimentação de todo o restante circuito.

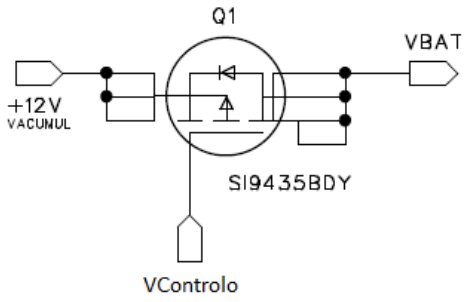


Figura 72: Interruptor Digital

3.5.1.4 – Conversão 12VDC – 5VDC

Esta conversão de tensão é obtida de forma direta por meio do conversor DC-DC REG 1 (ver secção 3.1.7) a partir da tensão de funcionamento VBAT:

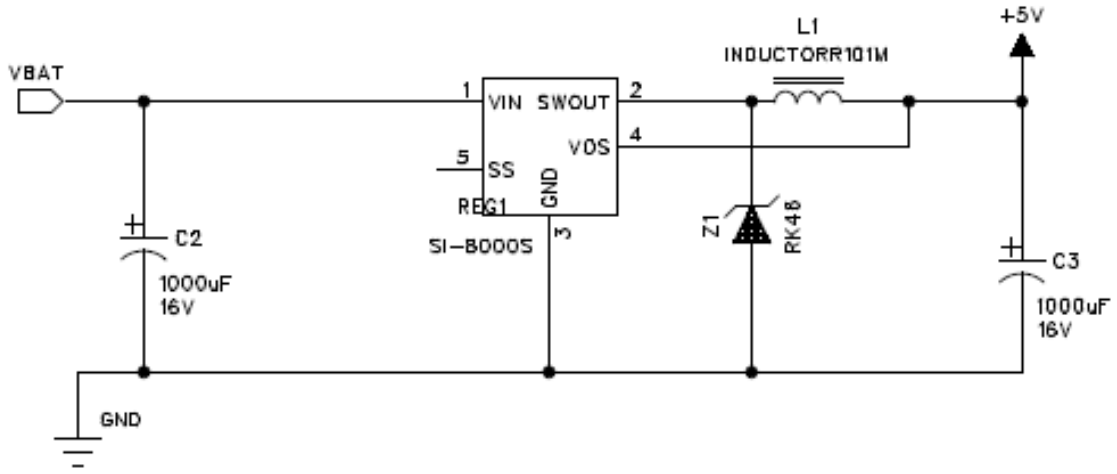


Figura 73: Conversão 12VDC-5VDC

3.5.1.5 – Tensão de controlo do interruptor digital

Existem no esquema quatro secções distintas de circuitos de controlo do interruptor digital: O circuito de deteção de picos de tensão na linha de 5V, o circuito de r*eset*, o circuito de controlo de presença de energia na rede e o circuito de controlo de tensão baixa controlado pelo microcontrolador ou diretamente por *hardware*.

Qualquer um destes circuitos que force o desligar da alimentação pela presença de tensão na *gate* do *mosfet* ganha prioridade sobre todos os outros circuitos de controlo que estejam a forçar essa tensão em sentido contrário.

3.5.1.5.1 – Controlo do interruptor digital por circuito de controlo de presença de energia na rede

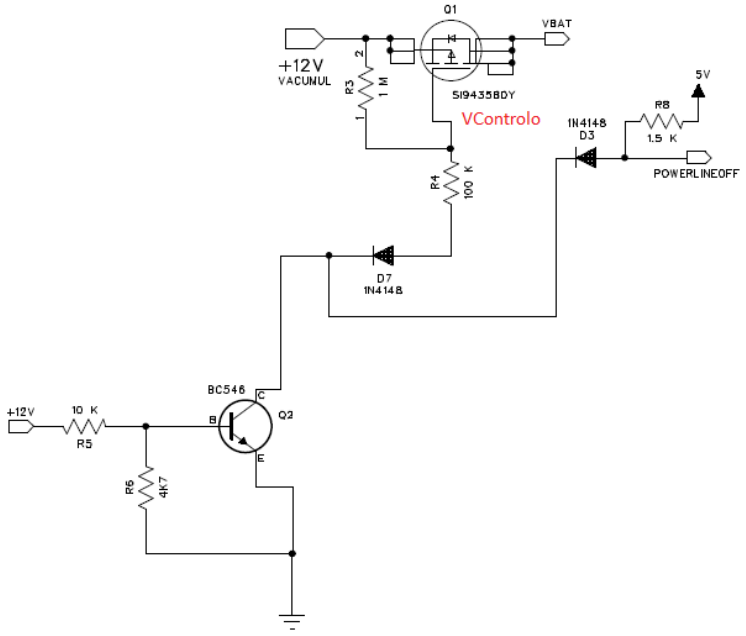


Figura 74: Controlo do interruptor digital por circuito de controlo de presença de energia na rede.

Este circuito é o responsável pela "ligação" do interruptor eletrónico quando o circuito é alimentado. No ato de ligação da placa de aquisição de dados RenPAD à energia elétrica, o transístor Q2 recebe 12V na sua base e satura passando a comporta-se como um curto-circuito entre o coletor e o emissor, colocando dessa forma um valor perto de 0V (depende de R3, R4, D7, VCE_{Q2} e tensão na *source* do *mosfet*) na *gate* do *mosfet* Q1 provocando a sua passagem à condução e permitindo dessa forma a passagem de corrente que alimentará todo o circuito adjacente.

Quando deixa de haver tensão de alimentação, o transístor Q2 passa a estar ao corte por deixar de haver tensão de polarização na sua base. Neste cenário seria espectável que o sistema se deligasse, no entanto logo que ocorra a falha de energia a bateria passa automaticamente e de forma "invisível" do regime de carga para o regime de alimentação, e o interruptor digital mantém-se "ligado" porque o sistema descrito na secção seguinte assim o preserva.

3.5.1.5.2 – Controlo do interruptor digital por circuito de controlo de tensão baixa controlado pelo microcontrolador ou diretamente por hardware

Este controlo é realizado por meio do circuito integrado DS1231 (ver descrição na secção 3.1.8).

Como anteriormente mencionado, o circuito integrado DS1231 está constantemente a analisar a tensão de alimentação geral do circuito VBAT e caso esta baixe de um determinado valor predefinido, esse evento é sinalizado por meio do pino /NMI. Além disso, este circuito integrado também assegura, através dos seus pinos RST e /RST a correta reinicialização do microcontrolador caso a tensão desça abaixo de 10% de VCC.

Com o circuito desenvolvido, podemos controlar o interruptor digital por *hardware*, através do uso direto do sinal /NMI, ou pelo microcontrolador através da linha POWERON. Esta seleção é feita com recurso à configuração do *jumper* X4.

Na figura seguinte vemos como este sistema foi implementado:

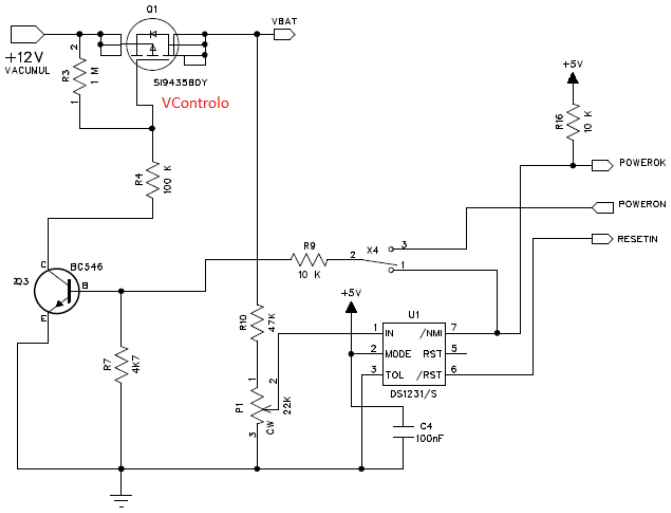


Figura 75: Controlo do interruptor digital por circuito de controlo de tensão baixa controlado pelo microcontrolador ou diretamente por *hardware*

Podemos observar que, à semelhança do esquema apresentado na secção anterior, se o transístor Q3 tiver na base um sinal de 5V que o sature, este irá impor na *gate* do *mosfet* uma tensão perto dos 0V que o fará manter-se no estado "ligado".

Caso a tensão na base do transístor desça para os 0V, este passa ao corte fazendo com que a tensão na *gate* do *mosfet* esteja perto da sua tensão na *source* (apenas difere devido à queda de tensão na resistência R3), muito acima dos 0V forçando a sua passagem ao corte (estado de "desligado"), cortando a alimentação a todo o circuito adjacente.

3.5.1.5.3 – Tensão de controlo do interruptor digital por circuito de reset

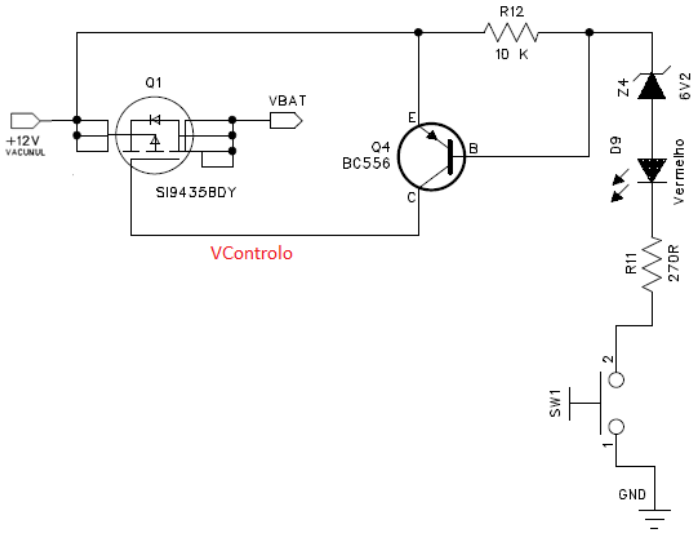


Figura 76: Circuito de *reset*.

Uma vez que o botão de pressão SW1 se encontra geralmente no estado aberto, existe apenas uma corrente residual a fluir pela resistência R12, o que faz com que a tensão presente na base e no emissor do transístor Q4 sejam praticamente iguais fazendo que não ocorra polarização e este se mantenha ao corte. Nesse estado a tensão de controlo do interruptor digital não sofre qualquer interferência deste circuito e é controlada apenas por qualquer um dos circuitos já descritos anteriormente.

Aquando da ocorrência do acionamento do botão de pressão SW1 (ordem de *reset*), o circuito é fechado e passa a fluir corrente pelo troço R12-Z4-R11-SW1 para a massa. O díodo de *zener* Z4 impõe uma tensão na base do transístor Q4 de 6,2V fazendo com que este sature completamente e garantindo uma tensão VEB segura para que não queime. Neste cenário de saturação, a junção emissor-coletor do transístor comporta-se como um

curto-circuito impondo na *gate* do *mosfet* uma tensão próxima da sua tensão de *source* (muito superior a 0V), fazendo com que este passe ao corte (estado "desligado"), retirando a alimentação a todos os circuitos adjacentes.

Quando o analisador da tensão de alimentação DS1231 verifica que VBAT desceu para 0V, passa a controlar o funcionamento do microcontrolador assegurando a sua permanência no estado de *reset*, libertando-o apenas quando todos os sinais de tensão estejam devidamente estabilizados. Este modo de funcionamento ocorre sempre que VBAT desce abaixo dos 10% da sua VCC.

3.5.1.5.4 – Tensão de controlo do interruptor digital por circuito de deteção de picos de tensão na linha de 5V

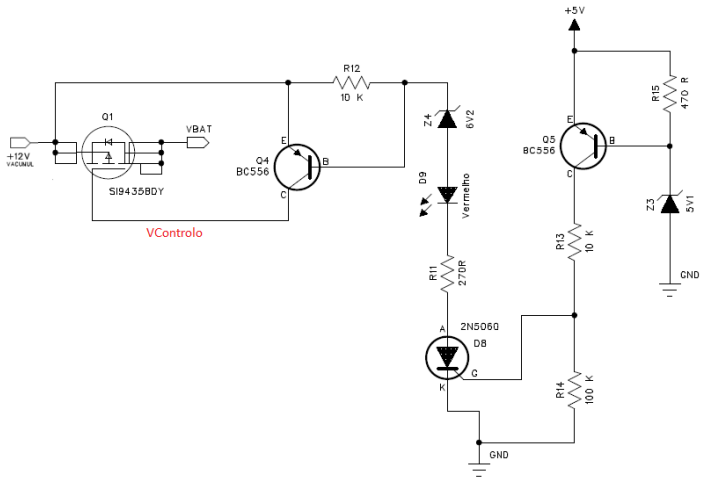


Figura 77: Circuito de deteção de picos de tensão na linha de 5V.

Na fase de arranque, após a ligação da placa de aquisição de dados RenPAD à energia elétrica, e durante o regime normal de funcionamento do equipamento o tiríston 2N5080 está ao corte, assemelhando-se o seu funcionamento ao de um *switch* aberto.

Quando o tiríston é polarizado por meio de uma tensão aplicada na sua *gate*, este entra no regime de condução, assemelhando-se o seu funcionamento a um *switch* fechado.

Partindo deste pressuposto, e sabendo que a *gate* deste tiríston recebe tensão somente quando o transístor Q5 satura, e que isso apenas acontece quando o valor na linha de alimentação de 5V é tal que, associado ao valor de tensão imposto pelo *zener* Z3 suba acima dos 5,8V todas as observações sobre o funcionamento do circuito da secção anterior

são válidas para explicar o funcionamento deste circuito e o controlo dos picos de tensão na linha de 5V é efetivo para picos superiores a 5,8V.

Como este circuito de proteção é alimentado pela tensão presente na *source* do *mosfet*, caso o tirístor arme ficará constantemente armado, e para desarmá-lo é sempre necessário que a tensão aos seus terminais desapareça completamente. Significa isto que, caso este circuito de proteção atue, enquanto não for pressionado o botão de *reset* o equipamento fica completamente desligado.

3.5.2 – Ligações ao microcontrolador

Na figura seguinte é apresentada a secção de esquema designada por CPU, onde são definidas todas as ligações aos pinos do módulo microprocessador RCM3700.

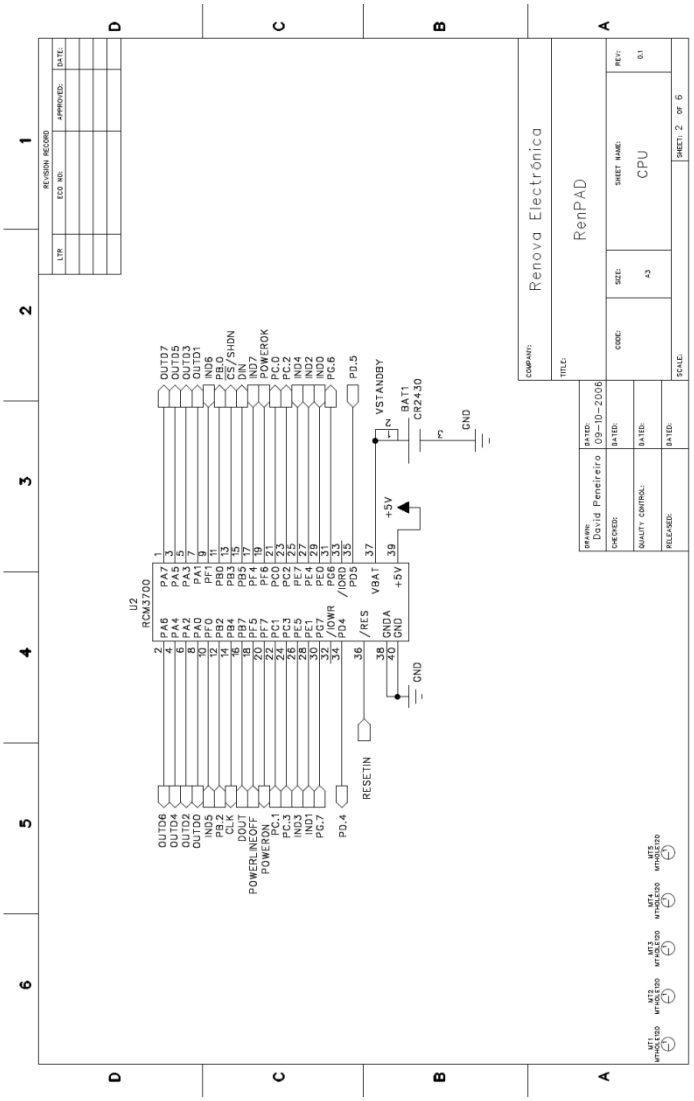


Figura 78: Esquema geral das ligações ao módulo microprocessador RCM3700

3.5.3 – Entradas Digitais

Na figura seguinte é apresentada a secção de esquema designada por Entradas_Digitais. Estes circuitos foram idealizados para proteção dos portos do microcontrolador criando uma barreira física de isolamento entre o pino da ficha externa e o porto de ligação correspondente, por intermédio do transístor BC546.

Em virtude destes circuitos de proteção, os sinais presentes nos portos do microcontrolador têm valor lógico inverso (sinal negado) ao sinal real presente nos respetivos pinos da ficha externa.

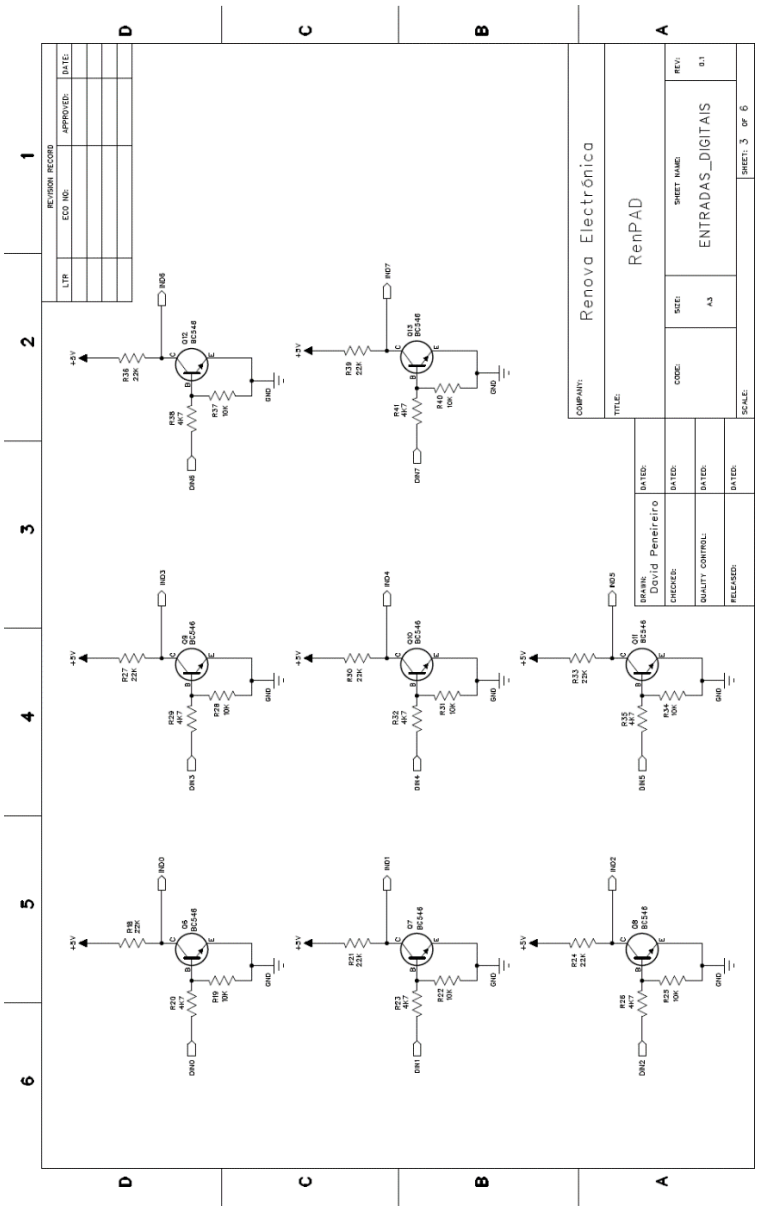


Figura 79: Esquema geral do sistema de isolamento das entradas digitais entre a ficha de ligações e o módulo processador RCM3700 para proteção do mesmo.

3.5.4 – Entradas Analógicas

Na figura seguinte é apresentada a secção de esquema designada por Entradas_Analógicas. Neste circuito podemos observar o esquema de ligações do ADC MCP3208 com a respetiva referência de tensão de precisão MCP1525, um sensor de temperatura LM60 ligado diretamente na placa, e a ligação das massas analógica e digital por meio do *shunt* X5.

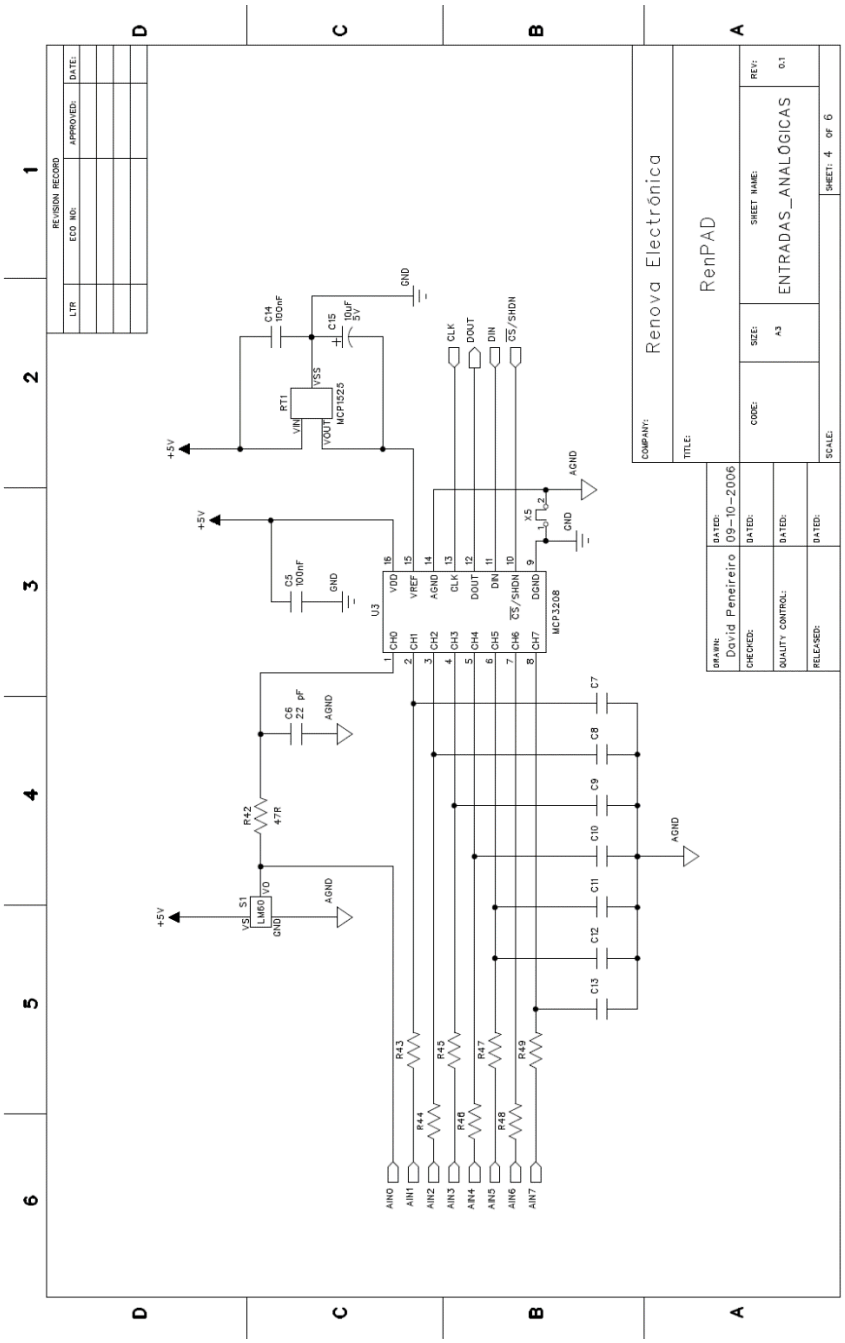


Figura 80: Esquema geral de ligação do ADC MCP3208

3.5.5 – Saídas Digitais

Na figura seguinte é apresentada a secção de esquema designada por Saidas_Digitais. Nestes circuitos, que funcionam de modo similar e com intuito idêntico aos circuitos apresentados na secção 3.5.3 designada por Entradas Analógicas, podemos observar que qualquer saída pode fornecer 5V ou 12V, selecionáveis por meio de um *jumper* de configuração, e que todas as saídas digitais estão dotadas de um díodo de roda livre de proteção contra correntes reversas, que possam ser impostas pelo circuito externo (tais como as correntes geradas pela bobine de um relé no momento do seu desatraque).

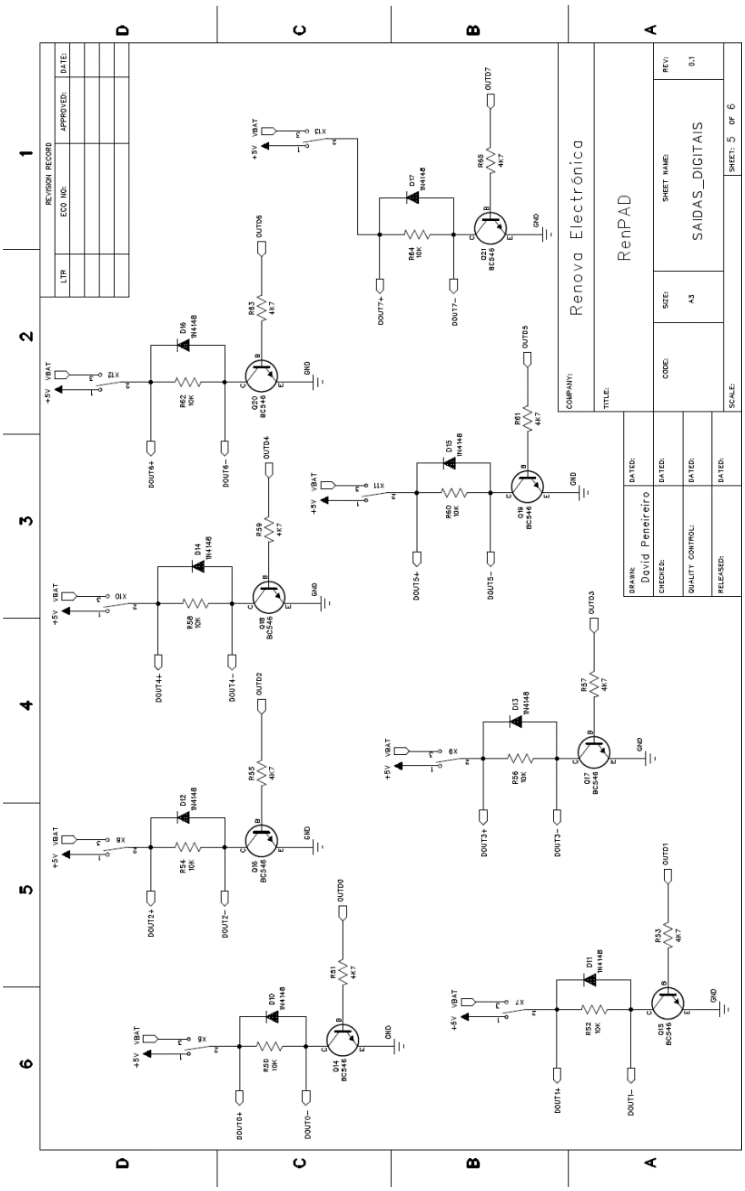


Figura 81: Esquema geral do sistema de isolamento das saídas digitais entre a ficha de ligações e o módulo processador RCM3700 para proteção do mesmo.

3.5.6 – Fichas

Na figura seguinte é apresentada a secção de esquema designada por Fichas que apresenta a configuração de todas as fichas externas, e sinais aplicados a cada pino de cada ficha.

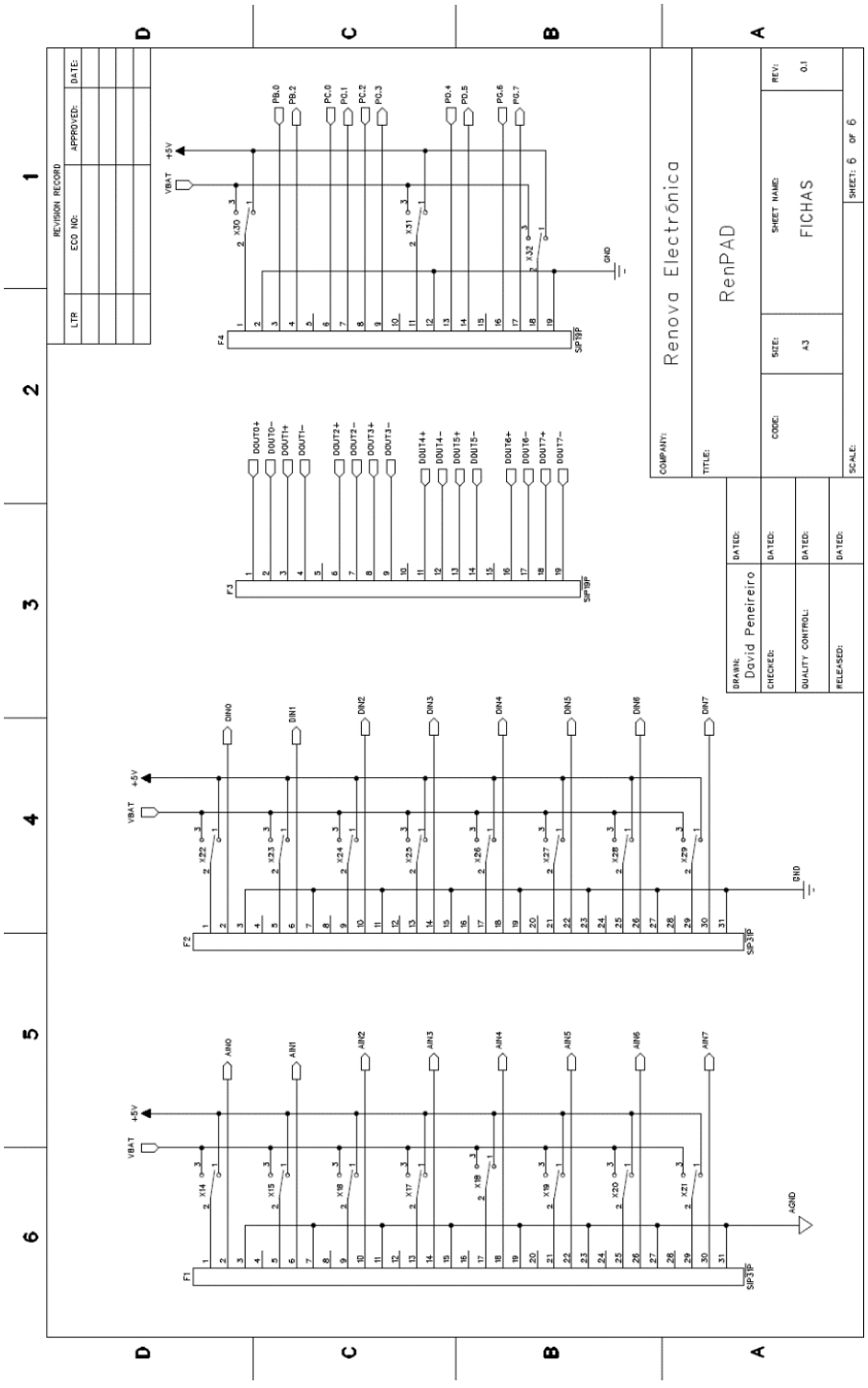


Figura 82: Esquema geral das fichas de ligações externas.

3.6 – Programação do microprocessador Rabbit 3000

Como já mencionado na secção 3.1.1 O ambiente de desenvolvimento integrado (IDE) utilizado para programação do microcontrolador instalado no módulo RCM3700 é proprietário, e é disponibilizado pelo fabricante do módulo com a designação de DynamicC.

Este IDE tem muitas semelhanças com qualquer ambiente de desenvolvimento integrado de *software*, estando no entanto vocacionado para os módulos *Rabbit* (*Digi* como são agora denominados após a aquisição da *Z-World* pela *Digi International® Inc.*) e implementa todas as funções relacionadas com a programação, *debug*, configuração, *etc*, desses módulos sem necessidade de recurso a outras ferramentas.

A linguagem de programação utilizada partilha tanto a sintaxe como a maioria das funções implementadas em C facilitando bastante a sua aprendizagem, tendo no entanto livrarias próprias dada a especificidade do produto. Pode o programador, caso pretenda, utilizar também a linguagem de mais baixo nível (*assembly*) em qualquer excerto do programa de forma fácil e integrada na solução global.

Com a integração do *Kernel* MicroC/OS-II no IDE, é possível implementar tarefas de tempo real, procedimentos multi-tarefa, semáforos, e todo um conjunto de soluções para processamento paralelo, sem grande esforço e com relativa facilidade de aprendizagem.

Por forma a facilitar a programação e atualização do *software* embebido nos terminais RenPAD, utilizamos também uma ferramenta de terceiros (*SHDesigns*), modificada para a realidade das aplicações Renova, e que permite a programação dos terminais via OTA (*Over the Air*) através do protocolo de comunicações *UDP*. Com esta ferramenta podemos reprogramar/atualizar qualquer placa de aquisição de dados RenPad à distância, através da rede *Ethernet*, sem necessidade de ir ao local conectar o cabo de programação poupando tempo, trabalho e a deslocação de um técnico ao local físico de instalação do dispositivo.

3.6.1 – Livrarias criadas

Por forma a tornar o código mais acessível, de mais fácil interpretação e mais organizado para consulta futura, eventuais alterações e aproveitamento para outras

aplicações, foi criada uma estrutura de programa subdividida nas diversas livrarias descritas a seguir:

- "Hardware.lib" – Livraria que contém todas as funções básicas inerentes ao funcionamento do *hardware* tais como:
  - "boardInit" – inicialização da *board*, sentidos dos portos, funções especiais dos portos, modo de funcionamento das portas série, entre outros.
  - "linhaKo" – verifica o estado da linha de alimentação.
  - "pilhasOk" – Verifica o estado das pilhas.
  - "powerOffTerminal" – Desliga o dispositivo.
  - …
- "Defines.lib" – livraria onde estão configuradas todas as definições do sistema tais como: "OS_TICKS_PER_SEC", "UDP_SOCKETS", "MAX_UDP_SOCKET_BUFFERS", "N_TASKS", "TASK_STK_SIZE", "OS_SEM_EN", entre outras.
- "MemGest.lib" – Livraria de gestão de memória, onde estão definidas algumas funções relacionadas com o teste e integridade da memória (por exemplo: "memcheck", "setchkbit", "InitLastError", crc32…), inicialização e mapeamento dos diversos blocos de memória (por exemplo: "heap_init", "heap_format", "heap_alloc", "heap_free"…), entre outras.
- "TCPIP_Config.lib" - Livraria que implementa funções de configuração da ligação de *TCP-IP* tais como: "myifconfig", "my_tick", "rnv_setip", entre outras.
- "UDPDOWNL.lib" – Livraria comercializada pela *SHDesigns* e alterada pela Renova, que providencia à aplicação a capacidade de programação remota dos terminais via *Ethernet*. Esta livraria faz parte de *software* desenvolvido e comercializado por terceiros, e como tal é confidencial não sendo disponibilizada como parte integrante deste projeto.
- "Tab_Var.lib" – Livraria onde estão definidas todas as variáveis, tabelas e estruturas de dados, criadas para o projeto.

- "Comandos.lib" – Livraria que caracteriza o protocolo de comunicações com a placa de aquisição de dados RenPAD, implementando toda a descodificação, verificação, validação e resposta aos comandos enviados via *TCP-IP*.
  O protocolo de comunicações e lista de comandos estão detalhados no Anexo 6.
- "Auxiliar.lib" – Livraria que implementa funções auxiliares ao processo tais como:
  - "tmp75In" – Leitura dos valores de temperatura presentes em todos os sensores TMP75. Esta função encontra-se detalhada no Anexo 2.
  - "sht75In" – Leitura dos valores de temperatura presentes em todos os sensores SHT75. Esta função encontra-se detalhada no Anexo 3.
  - "analogIn" - Leitura dos valores de temperatura presentes em todos os sensores analógicos. Esta função encontra-se detalhada no Anexo 1.
  - "inicializaVariaveis" – Inicialização de todas as variáveis com os valores por defeito aquando da formatação do sistema;
  - "printfXY" – Impressão de texto no monitor do computador em modo de debug
  - …
- "custom_config.lib" – Apesar desta livraria ser de sistema, é necessário alterá-la definindo o modo de funcionamento e as configurações de rede pretendidas.
- "Tarefas.lib" – Livraria que implementa as diversas tarefas a serem chamadas de forma concorrencial. Existem neste momento apenas três tarefas concorrentes:
  - "tcp_ip" – Aguarda por conexões *TCP-IP* e *UDP*. Quando recebe algo por *TCP-IP* valida se é um comando e em caso afirmativo, através da chamada das funções presentes em "Comando.lib" trata de obter a informação pretendida e dá resposta. Caso seja um comando *UDP*, valida se é um pedido de atualização de *software* e providencia a operação por meio das funções presentes em "UDPDOWNL.lib".

    Esta livraria implementa um semáforo que impede determinadas operações, tais como o desligar do dispositivo se a alimentação não estiver disponível e o processador for avisado de bateria fraca (neste caso o dispositivo apenas desligará após envio correto da resposta ao

comando recebido), ou escrita de valores em memória que possam estar relacionados com o comando em causa, entre outros.

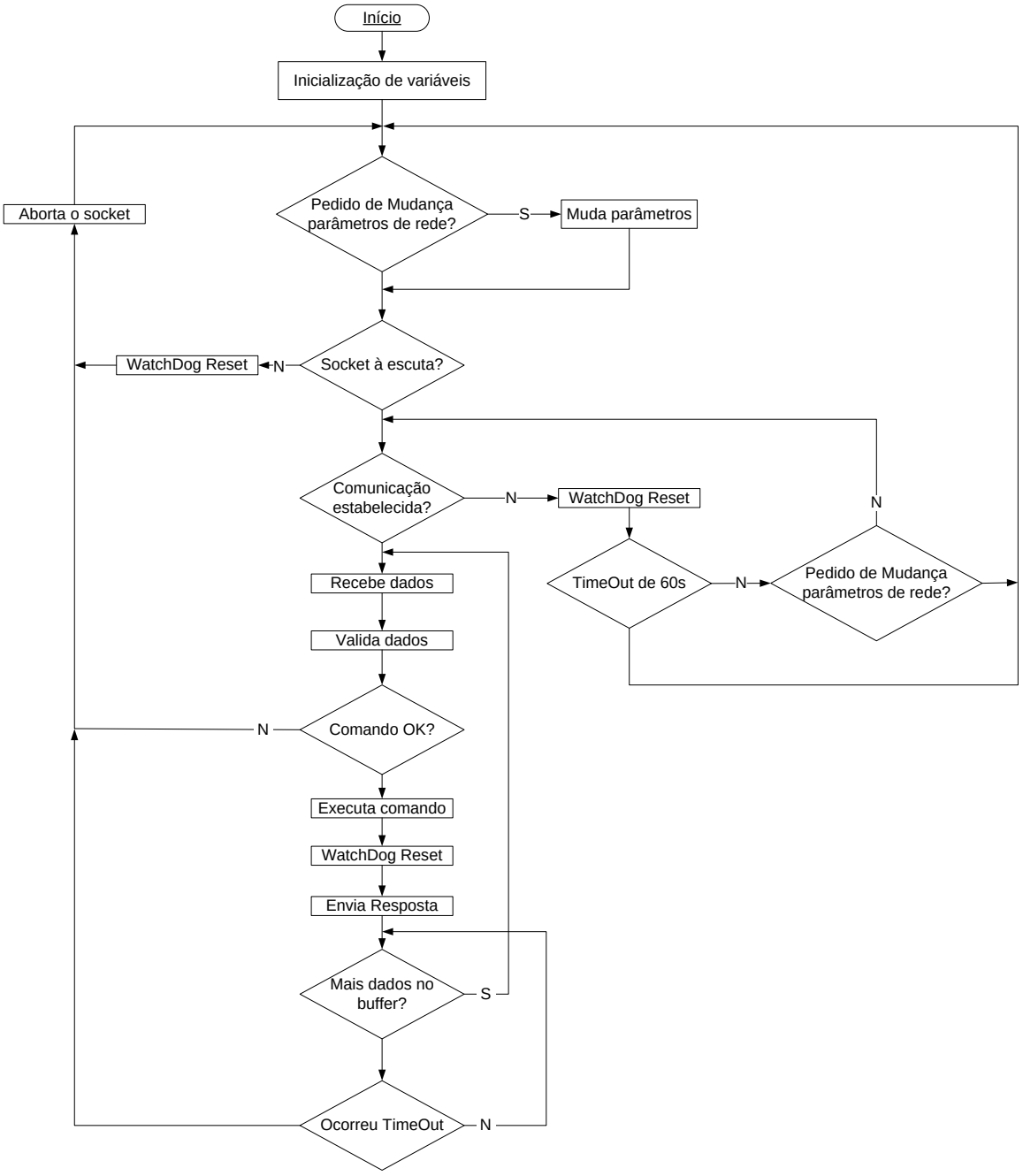


Figura 83: Fluxograma exemplificativo do funcionamento da tarefa “tcp_ip”

- “trata_saidas” – Valida se existe alguma ocorrência que force o estado de uma saída. Esta ocorrência poderá ocorrer por efeito de um comando de ativação manual da saída ou por ativação de qualquer tipo de alarme. Foi criada uma tarefa isolada para tratar deste tipo de ocorrências para

que o despoletar destes eventos seja realizado de forma assíncrona e independente do resto do programa. Esta tarefa corre de 500 em 500ms (excetuando-se o tempo de processamento) garantindo uma resposta quase imediata à ocorrência de determinado evento.

O modo de funcionamento dos alarmes e possíveis configurações encontram-se descritos no Anexo 7.

- "trataEntradas" – Esta tarefa corre de cinco em cinco segundos (excetuando-se o tempo de processamento) e providencia não somente a leitura dos valores de todos os sensores para armazenar em memória, mas também a mudança de horário de verão/inverno na data configurada e o *shutdown* do dispositivo no caso deste se encontrar em funcionamento por meio exclusivo da bateria, de existir sinal de bateria fraca e existir indicação por parte do *software* de que nenhuma operação crítica está a decorrer.

  Pela descrição anterior depreende-se que a aquisição dos valores presentes em cada sensor pelo programa de monitorização descrito no capítulo 4, não é obtido aquando da realização do pedido mas sim e indiferenciado, com um lapso temporal máximo de 5 segundos. Este fato não descaracteriza os requisitos do sistema, uma vez que este hiato máximo de tempo entre amostras não é significativo para todos os tipos de sensores que testamos.

  Aliás, existe mesmo uma frequência máxima de amostragem recomendada para alguns sensores (como o SHT75 por exemplo) a ter em conta, para que o seu correto funcionamento não seja afetado por um aquecimento excessivo derivado da operação de obtenção de valores e sua conversão de analógico para digital. Depois de analisados todos estes factos, decidimos optar pela amostragem de 5 em 5ms pois é um valor seguro em termos de aquecimento de operação, mantendo-se o sistema a funcionar em "tempo real", existindo também um menor atraso na mudança de hora de verão/inverno.

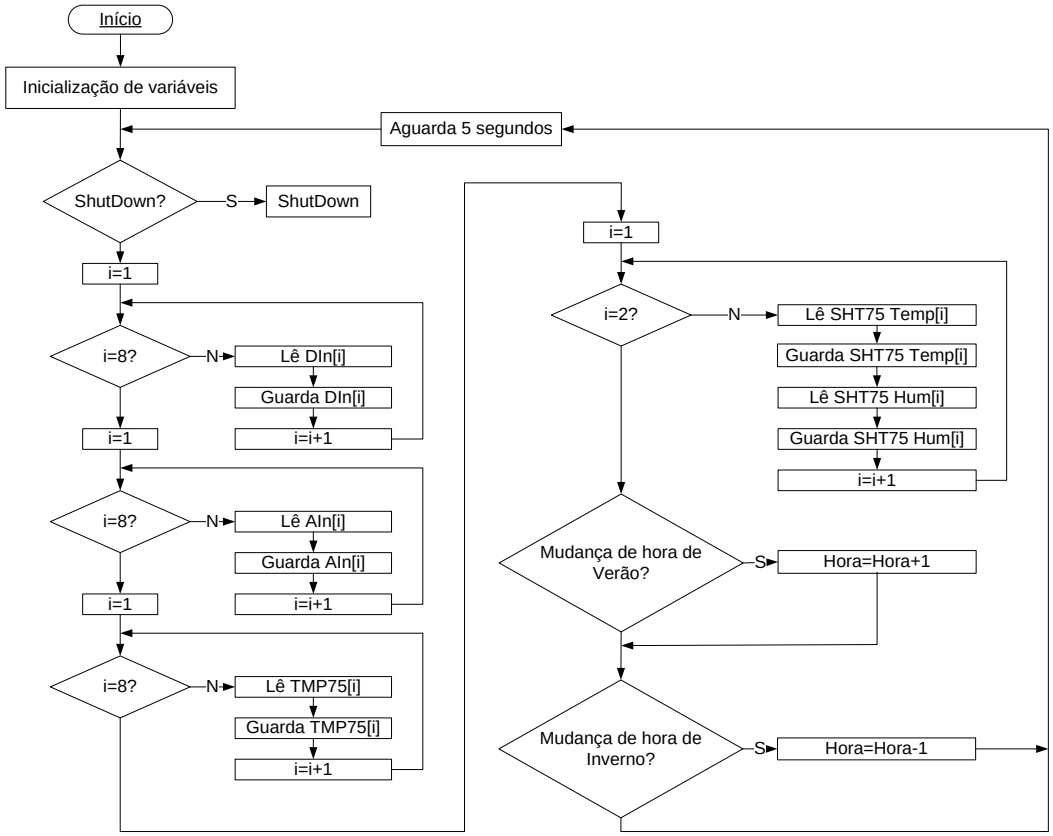


Figura 84: Fluxograma exemplificativo do funcionamento da tarefa “trataEntradas”.

O código desenvolvido e implementado nas livrarias atrás mencionadas pode ser consultado no Anexo 5.

3.6.2 – Criação das tarefas de funcionamento

Todo o programa funciona suportado pelas tarefas definidas em “tarefas.lib”, existe apenas uma função inicial chamada no arranque do sistema, designada por “main” presente no ficheiro “RenPad.c”, encarregue de configurar e lançar as tarefas sempre que o dispositivo inicia. Além disso a função “main” também é responsável por averiguar se a inicialização foi despoletada por um evento geral ou por um evento de atualização do *software* e nesse caso, providencia a entrada do dispositivo em modo de programação. É também responsável pela inicialização das variáveis iniciais e pelo teste de memória para verificação de necessidade da sua formatação devido a erros.

De modo a elucidar o anteriormente descrito, apresenta-se de seguida o programa principal que inclui as livrarias criadas, implementa a função “main”, *etc*:

```
/*********************************************************************************
       Programa principal, inclui as livrarias, cria os semáforos e as tarefas
              e faz arrancar todo o sistema de tempo real.
*********************************************************************************/
#class auto   // Change default storage class for local variables: on the stack

#memmap xmem

#use "Defines.lib"
#use "ucos2.lib"
#use "dcrtcp.lib"
#use "udpdownl.lib"
#use "Tab_Var.lib"
#use "TCPIP_Config.lib"
#use "Hardware.lib"
#use "Tarefas.lib"
#use "Auxiliar.lib"
#use "Comandos.lib"
#use "MemGest.lib"
/*
*********************************************************************************
*                               VARIÁVEIS
*********************************************************************************
*/
char      TaskData[N_TASKS];
OS_EVENT       *HeapSem;       // semáforo da heap
OS_EVENT       *IntComandoSem; // semáforo para as funções de interpretaComando();
OS_EVENT       *TimeSem;       // sem. para as funções de atualização da data/hora

OS_EVENT       *EntradasAnalogicasSem;// sem. para as funções de entradas analógicas
OS_EVENT       *EntradasDigitaisSem;  // sem. para as funções de entradas digitais
OS_EVENT       *SaidasDigitaisSem;    // sem. para as funções de saídas digitais
OS_EVENT       *SHT75Sem; // sem. para as funções relacionadas com os sensores SHT75
OS_EVENT       *TMP75Sem; // sem. para as funções relacionadas com os sensores
TMP175

void main (void)
{
       // Placa de aquisição de dados RenPAD

    // Inicialização do uC/OS-II
    OSInit();
    //Heap.crc=0;

    // Criação dos semáforos
    HeapSem = OSSemCreate(1);      // Cria o semáforo de acesso à Heap.
    IntComandoSem = OSSemCreate(1);// Cria o semáforo de acesso à rotina
                                   // interpretaComandos();
    TimeSem = OSSemCreate(1);

    EntradasAnalogicasSem = OSSemCreate(1);
    EntradasDigitaisSem = OSSemCreate(1);
```

```
    SaidasDigitaisSem = OSSemCreate(1);
    SHT75Sem = OSSemCreate(1);
    TMP75Sem = OSSemCreate(1);

    // Inicialização da board
    boardInit();

    // Valida se a reinicialização foi devida a um pedido de atualização
    // de software. Se sim prepara o dispositivo para a atualização.
    if(mytcpconfig.p.upgrade &&
      (mytcpconfig.crc==(crc32((char*)&mytcpconfig.p,sizeof(mytcpconfig.p)))))
   {
        myifconfig(1,"Preparado para download");
        Heap.crc=0;
        while(1) my_tick(NULL);
    }

    // Inicializa a tabela de erros.
    InitLastError();
    //Inicializa o WatchDog
    Dick=VdGetFreeWd(255);

    // Criação da tarefa de TCP-IP - Esta tarefa é criada aqui para que se
    // possa comunicar com o dispositivo mesmo que a inicialização de memória
    // falhe. Isto é útil para encontrar o dispositivo na rede ou para enviar
    // um pedido de inicialização ao dispositivo.
    OSTaskCreate(tcp_ip, (void *)0, 1024, 50);
    //OSTaskCreate(Porta4, (void *)0, 1024, 45);
    // Inicialização da memória: alocação e organização dos blocos da heap.
    if(memoryInit())
    {
      if(chkWDTO())
      {
          // Se a reinicialização do dispositivo foi provocada por um WatchDog
          // reset, o contador desses eventos é incrementado de uma unidade.
          WDRST++;
      }
      // Incremento da variável que contém o contador de reset's.
      AllRST++;

      // Criação das restantes tarefas. Estas tarefas apenas são criadas caso
      // a inicialização da memória corra bem pois se a memória estiver
      // fisicamente danificada, o dispositivo não funcionará corretamente.
/*********************************************************************************
 SYNTAX:char OSTaskCreate(void (*task)(),void *pdata,INT16U stk_size,char priority);
DESCRIPTION:  This function is used to have uC/OS-II manage the execution
                    of a task. Tasks can either be created prior to the start of
                    multitasking or by a running task.  A task cannot be created by
                    an ISR.
PARAMETER1: pointer to task function
PARAMETER2: pointer to an optional data area which can be used to pass parameters to
            the task when the task first executes.  Where the task is concerned it
```

```
            thinks it was invoked and passed the argument 'pdata' as follows:
              void Task (void *pdata){for (;;) { Task code ;}  }
PARAMETER3: size of task's stack in bytes.
PARAMETER4: the task's priority.  A unique priority ( 0-62 ) MUST be assigned to
each task and the lower the number, the higher the priority. (63 is for idle task)
RETURN VALUE: OS_NO_ERR     - if the function was successful.
              OS_PRIO_EXIT - if the task priority already exist
                             (each task MUST have a unique priority).
*********************************************************************************/
      OSTaskCreate(trataEntradas, (void *)0, TASK_STK_SIZE,10);
      OSTaskCreate(trataSaidas, (void *)0, 1024,15);
    }

    // Arranque do sistema de tempo real e inicialização do multitasking.
    OSStart();
}
```

3.7 – Testes e otimização dos dispositivos RenPAD

Ao longo do desenrolar das atividades inerentes á evolução deste produto, e de cada vez que algum novo desenvolvimento assinalável surgia, ia sendo exaustivamente testado para poder garantir o máximo de fiabilidade possível.

Este processo permitiu que, cerca de nove meses passados, se tenha obtido um sistema fiável completo e capaz, com muito poucas alterações a realizar numa revisão futura.

Devo deixar presente que, após desenhadas as placas de circuito impresso, apenas foram fabricados seis protótipos da revisão 0.1. Até à data, todas as alterações identificadas, são alterações do foro técnico que em nada prejudicam o bom funcionamento dos dispositivos.

Neste momento, e para a revisão 0.2 as alterações já implementadas são: furações dos componentes Z1 e REG1 que devem ser alargadas para uma mais fácil dessoldagem em caso de avaria, e alinhamento do *shunt* X32 com X31 e X30, que foram detetados aquando da montagem dos protótipos.

4 – Desenvolvimento do *Software* – Monitorizador RenPAD

No contexto deste projeto, entende-se por *software* o programa de computador denominado Monitorizador RenPAD, concebido para realizar recolhas dos dados obtidos através da placa de aquisição de dados RenPAD e monitorizar a sua evolução longo do tempo.

O programa foi concebido na linguagem de programação C# (*C Sharp*) que faz parte do conjunto de ferramentas oferecidas pela *Microsoft* como parte da plataforma .NET (*dot Net*) e surge como uma linguagem simples, robusta, orientada a objetos, fortemente tipada e altamente escalável, a fim de permitir que uma mesma aplicação possa ser executada em diversos dispositivos de *hardware*, independentemente de estes serem *PC's*, *handhelds* ou qualquer outro dispositivo móvel. A sua sintaxe orientada a objetos foi baseada no C++ mas inclui muitas influências de outras linguagens de programação, como *Object Pascal* e *Java*. [27]

O IDE (*Integrated Development Environment*) utilizado no desenvolvimento foi o disponibilizado pela *Microsoft*, o *Visual Studio* 2013 *pro*.

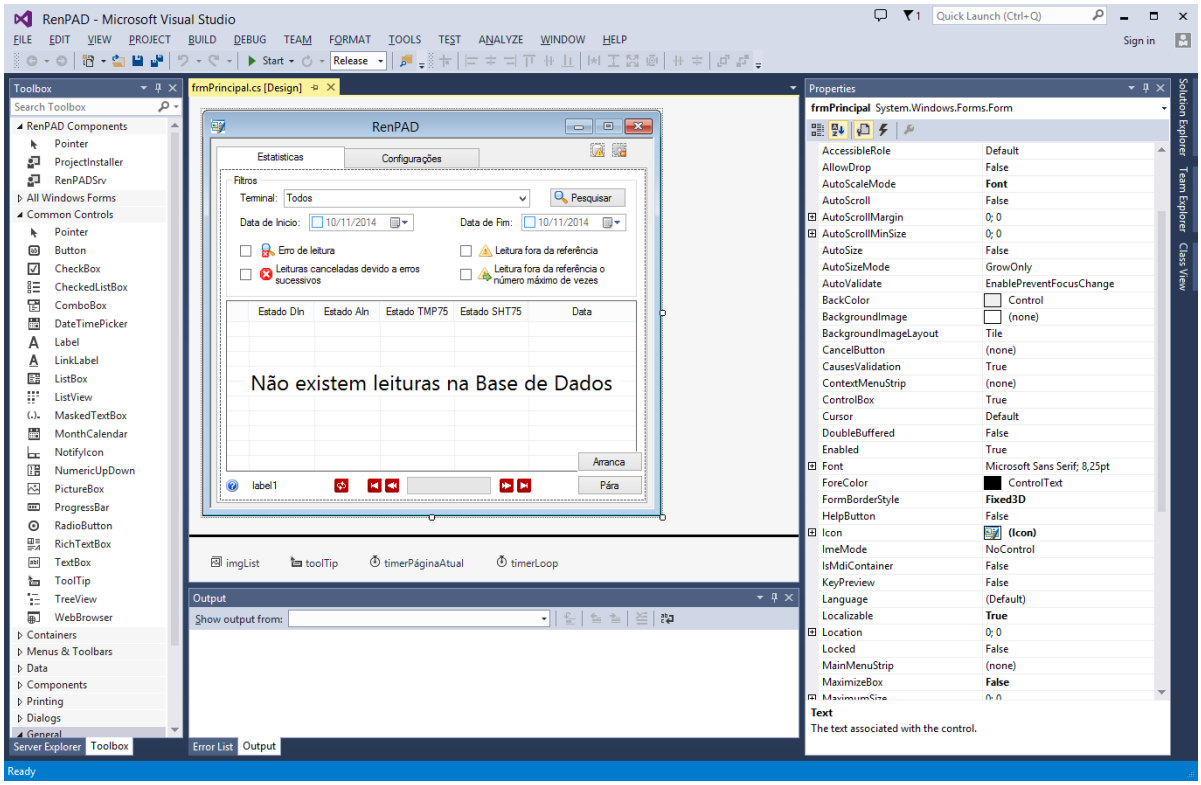


Figura 85*: Microsoft Visual Studio* 2013

O Monitorizador RenPAD pode dividir-se em três áreas distintas:

- Base de Dados – Estrutura em *Microsoft SQL Server*, onde são armazenados todos os dados recolhidos e configurações da aplicação.
- Serviço de recolha de dados, RenPADSrv, responsável pelo lançamento e gestão de todas as tarefas associadas à recolha e armazenamento dos dados provenientes das placas RenPAD configuradas na base de dados e presentes na rede *Ethernet* e da geração da sinalização dos alarmes correspondentes.
- Programa de configurações e análise de dados, responsável por obter da base de dados os dados recolhidos das placas RenPAD em monitorização, pela disponibilização desses dados para análise, e por todos os processos de instalação, configuração e manutenção tanto da base de dados, como do serviço associado.

4.1 – Base de Dados RenPAD

Como dito atrás, a base de dados está implementada em *SQL* e armazena não só os dados recolhidos das placas RenPAD, mas também todas as configurações de funcionamento do programa.

Por opção foram criadas apenas duas tabelas, uma de configurações onde são armazenadas todas as configurações relativas ao funcionamento de cada placa RenPAD instalada (tabela "Dispositivos"), e outra com os dados recolhidos de todos os dispositivos (tabela "Leituras").

DPENEIREIRODESK\S...AD - dbo.Leituras

Column Name	Data Type	Allow Nulls
ID	int	☐
[ID Dispositivo]	int	☐
Data	datetime	☐
DIn1	bit	☑
DIn2	bit	☑
DIn3	bit	☑
DIn4	bit	☑
DIn5	bit	☑
DIn6	bit	☑
DIn7	bit	☑
DIn8	bit	☑
[Status DIn]	varchar(8)	☐
AIn1	smallint	☐
AIn2	smallint	☐
AIn3	smallint	☐
AIn4	smallint	☐
AIn5	smallint	☐
AIn6	smallint	☐

Column Name	Data Type	Allow Nulls
AIn7	smallint	☐
AIn8	smallint	☐
[Status AIn]	varchar(8)	☐
TMP75_1	smallint	☐
TMP75_2	smallint	☐
TMP75_3	smallint	☐
TMP75_4	smallint	☐
TMP75_5	smallint	☐
TMP75_6	smallint	☐
TMP75_7	smallint	☐
TMP75_8	smallint	☐
[Status TMP75]	varchar(8)	☐
SHT75Tmp_1	smallint	☐
SHT75Tmp_2	smallint	☐
SHT75Hum_1	smallint	☐
SHT75Hum_2	smallint	☐
[Status SHT75]	varchar(4)	☐

Figura 86:Tabela "Leituras" de armazenamento dos dados recolhidos das placas RenPAD.

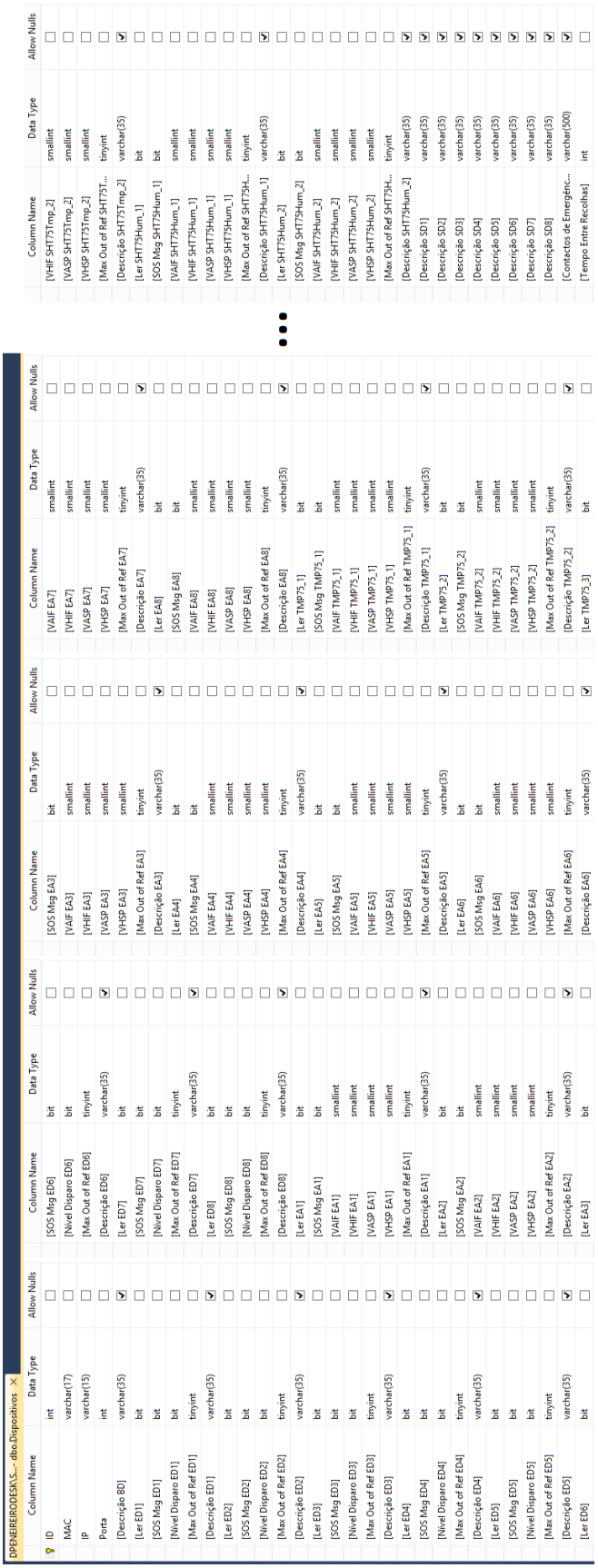


Figura 87: Tabela “Dispositivos” de configurações desejadas para cada placa RenPAD.

Na tabela "Dispositivos" estão representados os seguintes campos:

- "ID" – Identificador único, auto incrementável (incrementado automaticamente sem influencia do utilizador) que identifica univocamente cada dispositivo configurado nesta base de dados. Podemos ter dispositivos "iguais", com o mesmo endereço *TCP-IP*, *MAC*, *etc.* configurados na base de dados sem que isso prejudique o funcionamento. Na verdade o dispositivo será efetivamente sempre o mesmo, mas com configurações diferentes para podermos alternar entre modos de funcionamento dispares sem perca de tempo.
- "MAC" – Endereço *MAC* do dispositivo.
- "IP" – Endereço *IP* do dispositivo.
- "Porta" – Porta de comunicação *TCP-IP* e *UDP*.
- "Descrição BD" – Descrição que designa o dispositivo na base de dados para uma mais fácil identificação, por exemplo: "Sala de Servidores 1".
- "Ler EDX" – Indica se a entrada EDX deste dispositivo está a ser monitorizada.
- "SOS Msg EDX" – Indica se, em caso de ocorrência de alarme um número definido de vezes, o dispositivo envia uma mensagem para um endereço de correio eletrónico predefinido.
- "Nível Disparo EDX" – Nível lógico de funcionamento considerado anormal, que ativa um sinal de alarme (ver Anexo 7).
- "Max Out of Ref EDX" – Número máximo de vezes que um alarme terá que ocorrer para que seja enviada uma mensagem de correio (SOS Msg)
- "Descrição EDX" – Descrição que designa a entrada digital na base de dados para uma mais fácil identificação, por exemplo: "Porta Traseira".
- … (As definições são semelhantes e com os mesmos significados para os outros sensores)
- "VAIF EAX" – Valor de alarme inferior da entrada analógica X. Limite inferior que quando atingido ou ultrapassado, despoleta um sinal de alarme (ver Anexo 7).

- "VHIF EAX" – Valor de histerese inferior na entrada analógica X. Limite a partir do qual é considerado o rearme do alarme (ver Anexo 7).
- "VASP EAX" – Valor de alarme superior da entrada analógica X. Limite superior que quando atingido ou ultrapassado, despoleta um sinal de alarme (ver Anexo 7).
- "VHSP EAX" – Valor de histerese superior na entrada analógica X. Limite abaixo do qual é considerado o rearme do alarme (ver Anexo 7).
- … (As definições são semelhantes e com os mesmos significados para os outros sensores)
- "Contactos de Emergência" – Endereços de correio eletrónico para onde devem ser enviadas as mensagens de emergência (SOS Msg).
- "Tempo Entre Recolhas" – Temporização entre recolhas de dados da placa RenPAD.

Na tabela "Leituras" estão representados os seguintes campos:

- "ID" – Identificador único, auto incrementável (incrementado automaticamente sem influencia do utilizador) que identifica univocamente cada leitura configurado nesta base de dados.
- "ID Dispositivo" – O mesmo que "ID" da tabela atrás discriminada.
- "Data" – Data e hora da recolha dos dados. Não podem existir duas leituras ao mesmo dispositivo no mesmo segundo.
- "DInX" – Estado da entrada digital X.
- "Status DIn" – Informação do que ocorreu durante a obtenção do valor das entradas digitais. Cada *bit* deste campo representa uma entrada é preenchido com uma das seguintes descrições:
    - 0 - Entrada não monitorizada.
    - 1 - Leitura efetuada com sucesso e dentro dos parâmetros.
    - 2 - Erro ao ler a entrada.
    - 3 - Enviada mensagem de emergência indicando que houve erros sucessivos na leitura desta entrada e que por esse motivo a monitorização vai ser desligada.

- 4 - Valor da entrada analógica abaixo referência inferior (<=VAIF) ou valor da entrada digital fora da referência.
- 5 - Valor da entrada analógica abaixo referência inferior (<=VAIF) ou valor da entrada digital fora da referência o número máximo de vezes (enviada mensagem de emergência se configurado).
- 6 - Valor da entrada analógica acima referência superior (>=VASP).
- 7 - Valor da entrada analógica acima referência superior (>=VASP) o número máximo de vezes (enviada mensagem de emergência se configurado).

- … (As definições são semelhantes e com os mesmos significados para os outros sensores)

Os valores presentes nas tabelas são os valores hexadecimais obtidos da placa RenPAD e convertidos em decimal. Apenas os valores referentes aos sensores TMP75 que não sendo lineares, conforme apresentado na secção 3.1.4, são linearizados no ato da conversão para melhor sua melhor compreensão e para que mais facilmente os possamos validar. Linearizam-se também os dados inseridos correspondentes a VAIF, VHIF, VHSP e VASP pelo mesmo motivo.

Assim, os valores linearizados vão de 0 a 2640 (0x000 .. 0xA50) e a conversão faz-se da seguinte forma:

- Se o valor guardado na base de dados for inferior a 640 o valor real obtém-se somando a esse valor 3456 (0xD80).

  - Se $V_{bd} < 640 \Rightarrow V_{real} = V_{bd} + 3456$

- Caso contrário, se o valor guardado na base de dados for igual ou superior a 640 o valor real obtém-se subtraindo 640 ao valor guardado na base de dados.

  - Se $V_{bd} \geq 640 \Rightarrow V_{real} = V_{bd} - 640$

  Temos como exemplos:

$V_{bd} = 0 \Rightarrow V_{real} = V_{bd} + 3456 \Rightarrow V_{real} = 0 + 3456 = 3456 = 0xD80 = -40^{o}C$

$V_{bd} = 639 \Rightarrow V_{real} = V_{bd} + 3456 \Rightarrow V_{real} = 639 + 3456 = 4095 = 0xFFF = -0,06^{o}C$

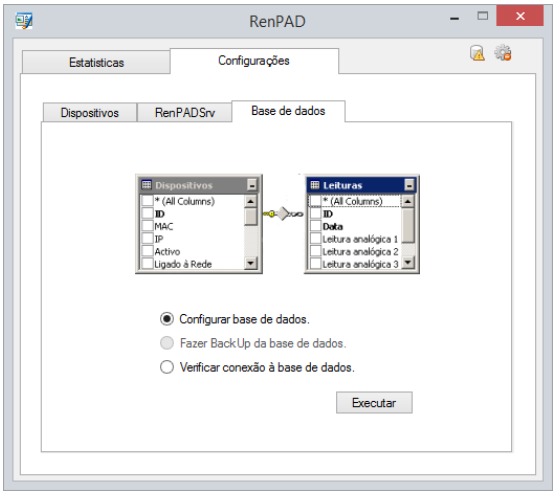
$V_{bd} = 640 \Rightarrow V_{real} = V_{bd} - 640 \Rightarrow V_{real} = 640 - 640 = 0 = 0x000 = 0^{o}C$

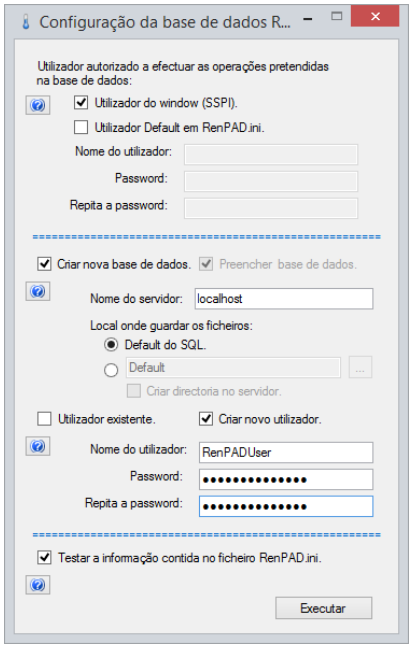
$V_{bd} = 641 \Rightarrow V_{real} = V_{bd} - 640 \Rightarrow V_{real} = 641 - 640 = 1 = 0x001 = 0{,}06^{o}C$

$V_{bd} = 2640 \Rightarrow V_{real} = V_{bd} - 640 \Rightarrow V_{real} = 2640 - 640 = 2000 = 0x7D0 = 125^{o}C$

Para acomodar a base de dados criada, podemos utilizar o *Microsoft SQL Server express edition* facultado gratuitamente pela *Microsoft* e cujas limitações em nada afetam o seu funcionamento nesta aplicação.

O programa Monitorizador RenPAD criado permite a instalação automática da base de dados de forma fácil e intuitiva com diversos menus de ajuda, bastando inserir o caminho da instância, e os dados de autenticação do *SQL Server*, o nome que queremos dar à base de dados e um utilizador de serviço a criar para *interface* entre o programa e o *SQL Server*.





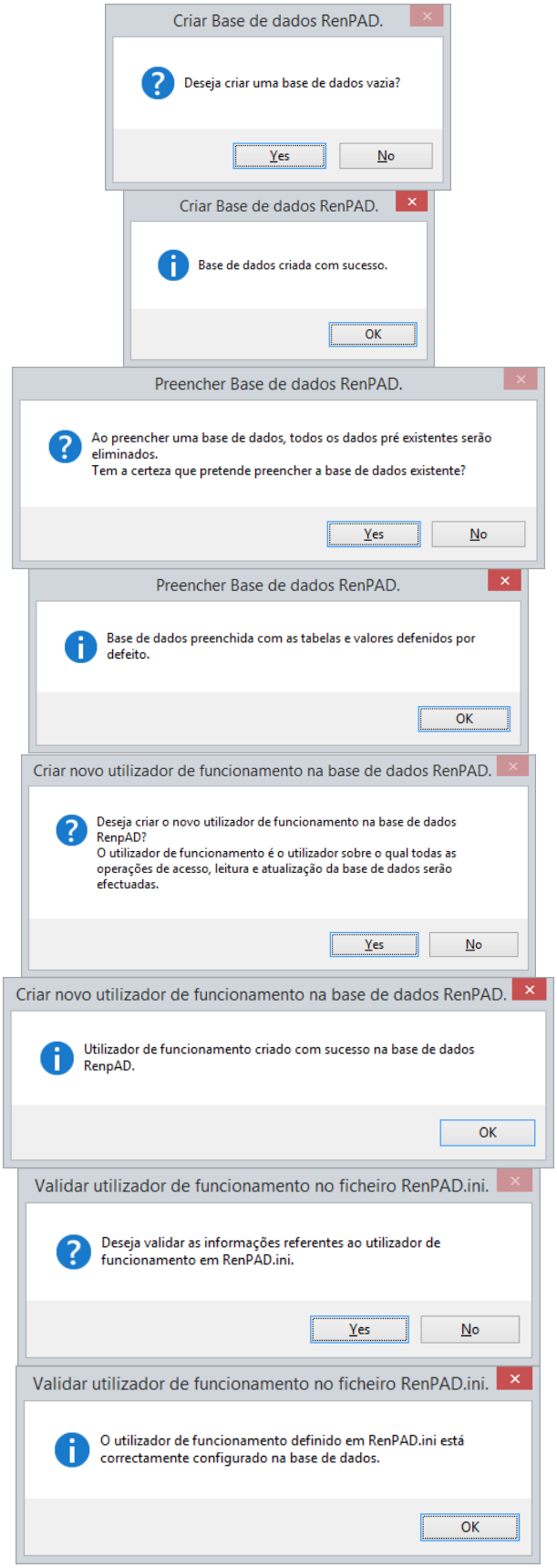


Figura 88:Os vários passos de criação da base de dados pelo programa Monitorizador RenPAD.

Caso ocorra algum erro durante o processo, a mensagem é sinalizada mas o processo continua podendo o utilizador parar ou não a instalação.

Figura 89: Exemplo de mensagem de erro ocorrido durante o processo de instalação da base de dados.

Para a implementação deste processo de instalação da base de dados, bem como toda a comunicação com a mesma, desde a inserção e atualização de dados, à sua obtenção e eliminação, foram utilizados comandos *SQL* executados por meio de métodos "base" implementados na livraria "BDUtils.cs" que pode ser consultada na íntegra no anexo 8, e que garantem o correto envio e receção de dados.

4.2 – Serviço de recolha de dados RenPADSrv.

Para a recolha de dados foi criado um serviço do *Microsoft Windows* (*Windows Service*) que, uma vez instalado, inicia a sua execução de forma automática durante a fase de arranque do computador, e permanece em execução em segundo plano enquanto o *Windows* estiver em execução, sem necessidade de *interface* com o utilizador.

Um serviço *Windows* é geralmente uma aplicação isolada, isto é: quando um projeto necessita de um serviço que corra automaticamente e de um programa de *interface* com o utilizador, são normalmente criadas duas aplicações distintas. Para este projeto, no entanto, optei por utilizar a mesma aplicação para as duas finalidades, ficando o serviço integrado na aplicação global. Desta forma, não só não necessitamos de um ficheiro executável extra, como a gerência do serviço pode ser toda feita na própria aplicação de *interface* com o utilizador (mais "*user friendly*") e esta, sabendo em que estado se encontra o serviço pode tomar medidas de proteção da integridade de dados. Por exemplo, não poderemos alterar as configurações de monitorização de um dado dispositivo na rede sem parar o serviço de recolha de dados, garantindo assim que as alterações tomam

efetivamente efeito na próxima consulta depois do serviço arrancar novamente. Esta solução também é mais elegante caso o produto entre na gama de soluções comercializadas pela Renova Eletrónica.

Para implementar este tipo de integração é necessário uma filosofia de desenvolvimento um pouco diferente que a da abordagem normal. Quando queremos criar um simples serviço, podemos escolher logo de raiz esse tipo particular de aplicação "serviço" durante a fase de criação do projeto. O IDE trata de incluir de forma automática as livrarias necessárias, os métodos associado, instalador, entre outros e por defeito, o único método de *interface* que o serviço tem com o utilizador é por intermédio de linha de comandos. Neste cenário, o executável criado implementa apenas as ações a executar durante as fases de operação do serviço (execução, instalação, desinstalação, arranque, paragem e reinicialização) e a sua gestão é feita via linha de comandos segundo as seguintes sintaxes:

- Instalar um *Windows service*:
  - **SC CREATE "**Nome do *Windows Service*" **binpath=** "C:\caminhoDoWindowsService\WindowsService.exe"
- Desinstalar um *Windows service*:
  - **SC DELETE** "Nome do *Windows Service*"
- Arrancar um *Windows service*:
  - **SC START** "Nome do *Windows Service*"
- Parar um *Windows service*:
  - **SC STOP** "Nome do *Windows Service*"
- Reiniciar um *Windows service*:
  - **SC CONTINUE** "Nome do *Windows Service*"

Já com a abordagem seguida, e uma vez que integrámos o serviço na aplicação, é necessário não apenas criar manualmente todo o serviço (instalador, métodos de instalação, arranque, paragem *etc*.) como diferenciar se a chamada da aplicação geral é de execução do serviço ou de arranque da aplicação de *interface* com o utilizador "Monitorizador RenPAD".

Essa diferenciação foi conseguida pesquisando na rotina inicial de arranque da aplicação ("main") o nome do processo que deu origem à sua chamada, denominado de "Processo Pai". Caso o "Processo Pai" se chame "services.exe" a aplicação foi iniciada

como um serviço e arranca nessa condição. Caso contrário a aplicação foi chamada por outro processo qualquer (geralmente o "explorer.exe") e arranca como aplicação de *interface* com o utilizador em modo "*Windows Form*". Como proteção, foi também inserido nesta rotina um método de validação, que analisa se a aplicação está já a correr em modo "*Windows Form*" para evitar que ocorram duas aplicações em execução em simultâneo evitando assim incongruências por engano do utilizador.

Podemos analisar em seguida, a título de exemplo, a rotina "main" com estas particularidades implementadas:

```
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
     InfoProcesso.pai = InfoProcesso.NomeProcessoPai();
     if (InfoProcesso.pai != "services.exe")
     {
         Process instancia=InfoProcesso.ExisteInstancia();
         frmPrincipal formPrincipal;
         if (instancia != null)
         {   // Não conta com a instancia criada pelo serviço.
             MessageBox.Show("Já existe uma instância do programa em" +
                 "curso.","RenPAD",MessageBoxButtons.OK,MessageBoxIcon.Information);
             InfoProcesso.MostraInstancia(instancia);
             //InfoProcesso.MostraInstancia();
             return;
         }
         Application.EnableVisualStyles();
         Application.SetCompatibleTextRenderingDefault(false);
         //Application.Run(new frmPrincipal(InfoProcesso.pai));
         formPrincipal = new frmPrincipal();
         Application.Run(formPrincipal);
     }
     else
     {
        ServiceBase[] ServicesToRun;
        ServicesToRun = new ServiceBase[] { new RenPADSrv() };
        ServiceBase.Run(ServicesToRun);
    }
}
```

A implementação dos métodos utilizados na rotina "main" pode ser consultada no Anexo 8 na livraria "InfoProcesso.cs".

Com a solução implementada podemos gerir o serviço não apenas via linha de comandos cujas sintaxes atrás se descreveram, mas também por meio de uma *interface* gráfica que implementa todas as opções possíveis de forma facilitada, intuitiva e integrada na solução global:

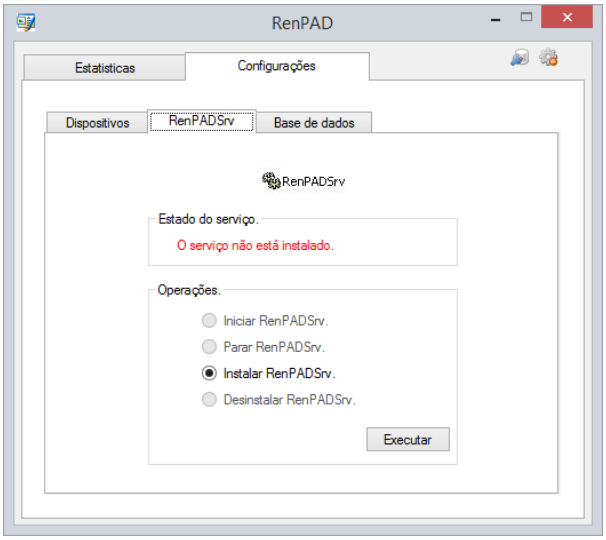


Figura 90: *Interface* de gestão do serviço RenPADSrv.

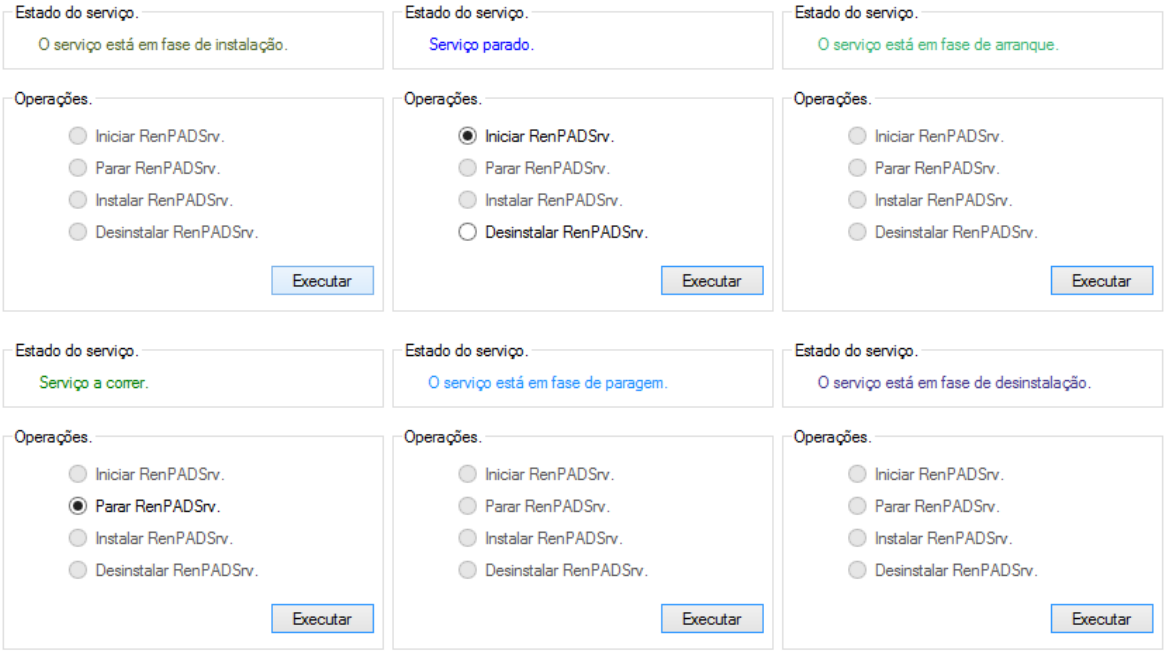


Figura 91: Outros estados de operação do serviço RenPAD e opções de mudança de estado disponíveis para cada cenário.

Após a instalação do serviço, este estará visível tanto no *Task Manager* como na consola de gestão de serviços do *Windows*:

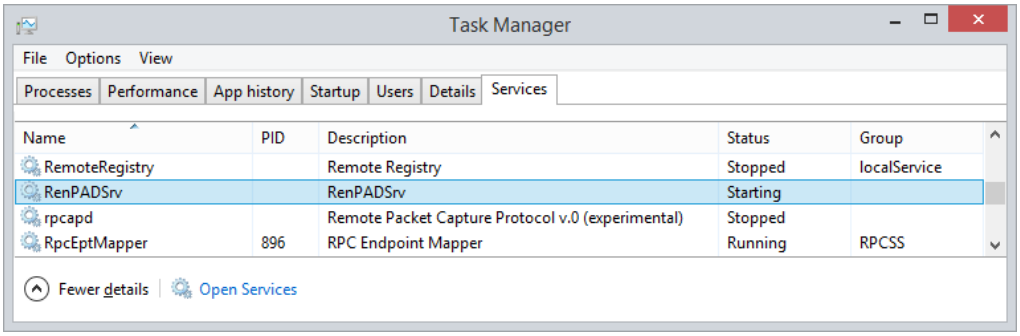


Figura 92: Serviço RenPADSrv apresentado no *Task Manager*.

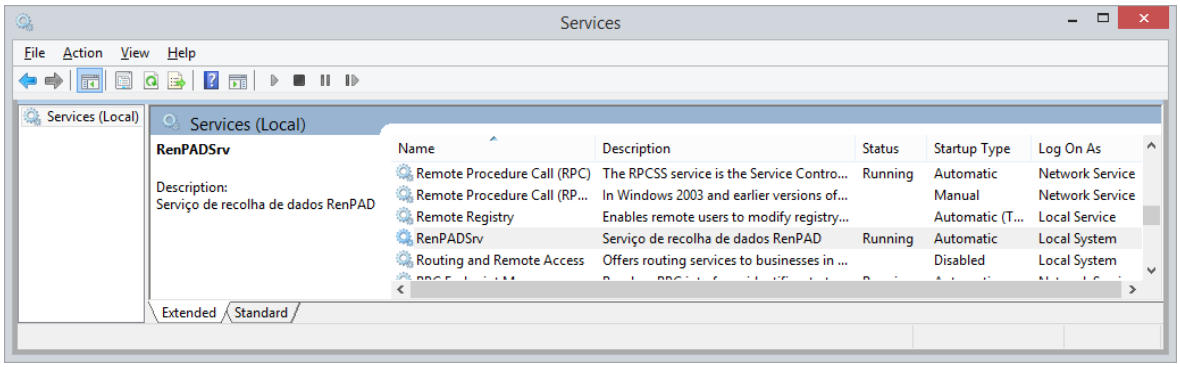


Figura 93: Serviço RenPADSrv apresentado na consola de gestão de serviços.

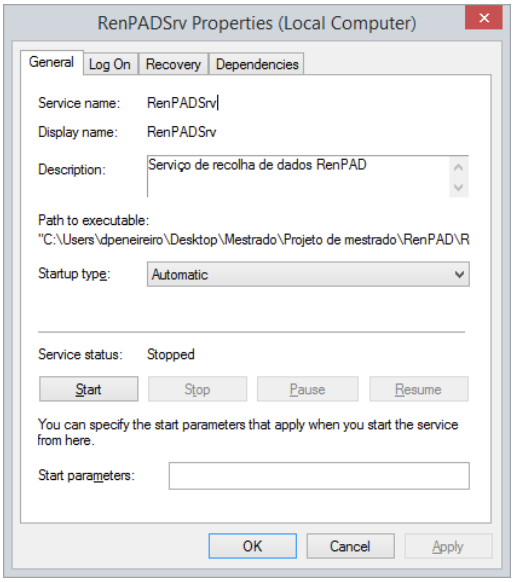


Figura 94: Propriedades do serviço RenPADSrv instalado.

Conforme indicado anteriormente, não é possível aceder a qualquer separador de configurações enquanto o serviço RenPAD estiver em modo de execução, para evitar

incongruências entre os modos de funcionamento pretendido e efetivo. Por esse motivo sempre que se tenta aceder a qualquer aba de parametrização e configuração, seja da base de dados ou de dispositivos, e o serviço esteja em execução, é apresentada a seguinte mensagem de erro, e o respetivo separador selecionado é apresentado com as opções todas desabilitadas (tornadas inábeis, não selecionáveis):

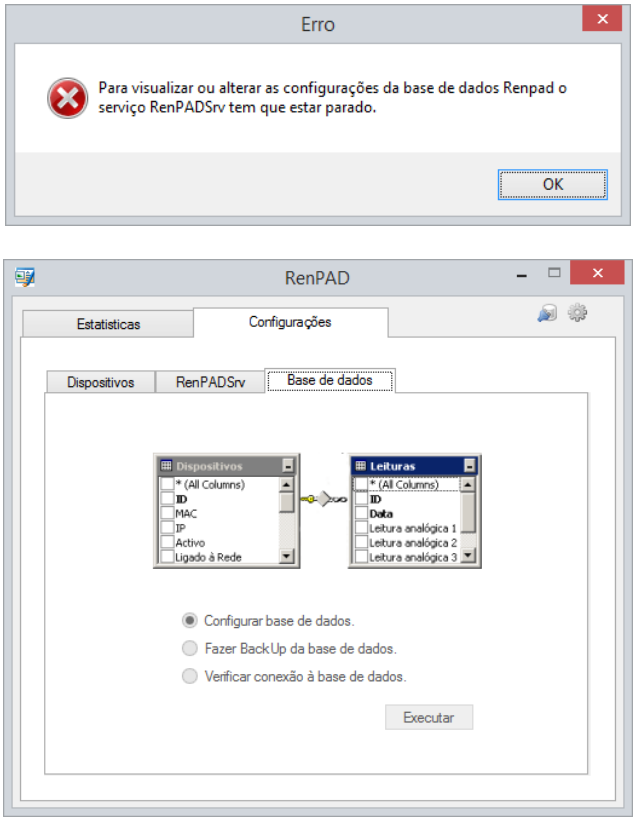


Figura 95: Mensagem de erro apresentada quando se acede a qualquer separador de configuração e o serviço se encontra em modo de execução e respetivo separador com todas as opções desabilitadas.

De forma a garantir que esta proteção é efetiva em todos os cenários, existe uma tarefa, lançada durante o arranque da aplicação, responsável por monitorizar o estado do serviço "em tempo real" e que capta qualquer alteração ao seu modo de operação, quer esta seja requisitada dentro da própria aplicação, via linha de comandos ou pela consola de gestão de serviços do Windows. A implementação desta tarefa pode ser visualizada no Anexo 8 na livraria "ServicoUtils.cs", nomeadamente as classes "RenPADSrvVerificaEstadoEventArgs" que deriva de "System.EventArgs" e "RenPADSrvVerificaEstadoEvent", o *delegate* "RenPADSrvVerificaEstadoEventHandler" e o método "EstadoDoServiço" da classe "ControladorDoServico".

Como nos temos vindo a aperceber ao longo desta secção, um serviço passa por vários estados internos durante o seu tempo de vida:

- Primeiramente, o serviço é instalado no sistema em que será executado. Para isso são executados os instaladores do serviço e este é carregado no Gestor de Serviços do Windows.
- Após o serviço estar carregado, deve ser iniciado. Iniciar o serviço, ou como se diz mais vulgarmente "arrancar com o serviço", permite que este entre em execução. O serviço pode ser iniciado a partir da consola do gestor de serviços do *Windows*, via linha de comandos, ou a partir da aba de configuração do serviço do programa "Monitorizador RenPAD". O evento de iniciar o serviço chama o método "Start*"* que passa o controlo do processamento para o método "OnStart" e executa qualquer código que tenha sido especificado neste método.
- Um serviço em execução pode permanecer nesse estado indefinidamente até que termine por si mesmo no final de executada a sua tarefa, seja interrompido por algum erro que ocorra durante a sua execução, seja manualmente parado, ou até que o computador seja desligado.
- Depois de instalado, um serviço pode estar num de três estados básicos: em execução (*Running*), em pausa (*Paused*) ou parado (*Stopped*). O serviço também pode informar o estado de um comando pendente: *ContinuePending*, *PausePending*, *StartPending*, ou *StopPending*. Estes estados indicam que um comando foi enviado, por exemplo, como um comando enviado para parar o serviço em execução, mas que ainda não foi executado.
- Um serviço em execução pode ser colocado em pausa, pode ser parado ou, no caso de estar em execução mas estar em pausa, pode ser recolocado em execução. Cada uma destas ações pode chamar um método associado: "OnStop", "OnPause" ou "OnContinue", onde é possível definir algum processamento adicional a ser executado quando o serviço altera de estado. [28]

Note-se que, em virtude dos sistemas de segurança do *Windows*, as operações descritas nos pontos anteriores têm que ser executadas em modo de administrador.

A forma de implementação dos métodos descritos nos pontos anteriores podem ser consultados no Anexo 8, na livraria ServicoUtils.cs nomeadamente a classe

ControladorDoServiço que implementa os métodos de instalação (“InstalaServiço”), desinstalação (“DesistalaServiço”), paragem (“ParaServiço”), arranque (“ArrancaServiço”) e obtenção do estado do serviço (“EstadoDoServiço”) entre outras, e cuja implementação de alguns destes métodos é feita de forma assíncrona por meio do lançamento de tarefas que correm paralelamente ao processamento principal do programa e na livraria RenPADSrv.cs onde são implementados os métodos “OnStart” e “OnStop”, que são os únicos comandos aceites por este serviço e que, a titulo ilustrativo e dada a sua importância para a compreensão do que se vai descrever em seguida, são agora transcritos e descritos:

```
/// <summary>
/// Método chamado na fase de arranque do serviço.
/// </summary>
protected override void OnStart(string[] args)
{
    LogoUtils logoMsg = new LogoUtils("RenPADSrv a arrancar.");
    LogoUtils.EscreveMensagem(logoMsg);
    try
    {
        System.Threading.Thread.Sleep(2000);
        ServicoUtils.stopService = false;
        ServicoUtils.RunMainThread();
    }
    catch /*(Exception exp)*/
    {
        // Escreve no logo que correu mal.
        //throw exp;
    }
}

/// <summary>
/// Método chamado na fase de paragem do serviço.
/// </summary>
protected override void OnStop()
{
    LogoUtils logoMsg = new LogoUtils("RenPADSrv a parar.");
    LogoUtils.EscreveMensagem(logoMsg);
    ServicoUtils.stopService = true;
    System.Threading.Thread.Sleep(2000);
}
```

Analisando o método “OnStop”, podemos facilmente verificar que este método nada mais faz do que colocar a variável “stopService” como *true* e escreve no ficheiro de registos (denominado por “Logo”) a ordem de paragem do serviço.

Já o método “OnStart” assinala no ficheiro de registos o evento de arranque do serviço, coloca a variável “stopService” no estado *false* e chama o método

"RunMainThread" que é efetivamente responsável pela inicialização e lançamento da tarefa "ServiceMainThread" que controla a inicialização e lançamento de forma concorrencial de todas as tarefas inerentes ao funcionamento do serviço.

Uma vez que o serviço é constituído por diversas tarefas concorrentes, é utilizada a variável "stopService" como mecanismo de sincronização, indicando a todo o processo que ocorreu uma ordem de paragem e que todas as tarefas devem terminar corretamente o seu processamento e dar por finda a sua execução.

```
/// <summary>
/// Método que arranca com a thread main do serviço
/// </summary>
public static void RunMainThread()
{
    try
    {
        if (BDUtils.ExisteBD() == false)
        {
            throw new Exception("Não foi encontrada a base de dados RenPAD.");
        }
        serviceMainThread = new Thread(new ThreadStart(ServiceMainThread));
        serviceMainThread.IsBackground = true;
        serviceMainThread.Start();
        LogoUtils logoMsg = new LogoUtils("O serviço arrancou correctamente.");
        LogoUtils.EscreveMensagem(logoMsg);
    }
    catch(Exception exp)
    {
        //ControladorDoServico.ParaServico();
        LogoUtils logoMsg = new LogoUtils("O serviço RenPADSrv não pode ser
                    iniciado." + Environment.NewLine + "Erro: " + exp.Message);
        LogoUtils.EscreveMensagem(logoMsg);
        serviceStopperThread = new Thread(new ThreadStart(ServiceStopperThread));
        serviceStopperThread.IsBackground = true;
        serviceStopperThread.Start();
    }
}
```

No código anterior verificamos que a tarefa "ServiceMainThread" é lançada num processo à parte. A opção pela inclusão desta tarefa intermédia anteriormente transcrita, prendeu-se pelo fato de simplificar ao máximo o código do método "OnStart" para uma leitura e compreensão facilitadas do programa global.

Seguidamente é apresentada a "ServiceMainThread". Está incluída nesta secção e não em anexo dada a sua importância, e a sua leitura atenta auxiliará a compreensão do funcionamento global do serviço RenPADSrv.

```
/// <summary>
///
/// Thread main do serviço. Esta thread faz arrancar todas as threads
/// de monitorização, uma independente por dispositivo a monitorizar.
/// e controla o seu devido encerramento.
///
/// </summary>
public static void ServiceMainThread()
{
    LogoUtils logoMsg=new LogoUtils("Sem mensagem.");
    int tempo=1000;
    int i = 0;
    ArrayList listaDispositivosBD = new ArrayList();
    ArrayList listaThreadsRecolhaDados = new ArrayList();

    serviceEnviaMensagemCorreioThread = new Thread(new
                            ThreadStart(ServiceEnviaMensagemCorreioThread));
    serviceEnviaMensagemCorreioThread.IsBackground = true;
    serviceEnviaMensagemCorreioThread.Start();

    //AddMensagemCorreioEnviar("ola\nOla\n1234", "david.peneireiro@renova.pt");

    listaDispositivosBD = DispositivoBD.PesquisaBD("WHERE " +
            "[Ler ED1]='True' OR [Ler ED2]='True' OR [Ler ED3]='True'  OR [Ler
ED4]='True' OR [Ler ED5]='True' OR [Ler ED6]='True' OR [Ler ED7]='True' OR [Ler
ED8]='True' OR " +
            "[Ler EA1]='True' OR [Ler EA2]='True' OR [Ler EA3]='True'  OR [Ler
EA4]='True' OR [Ler EA5]='True' OR [Ler EA6]='True' OR [Ler EA7]='True' OR [Ler
EA8]='True' OR " +
            "[Ler TMP75_1]='True' OR [Ler TMP75_2]='True' OR [Ler TMP75_3]='True'
OR [Ler TMP75_4]='True' OR [Ler TMP75_5]='True' OR [Ler TMP75_6]='True' OR [Ler
TMP75_7]='True' OR [Ler TMP75_8]='True' OR " +
            "[Ler SHT75Tmp_1]='True' OR [Ler SHT75Tmp_2]='True' OR [Ler
SHT75Hum_1]='True'  OR [Ler SHT75Hum_2]='True'");

  ////////////////////////////////////////////////////////////////////////////
  // Se houverem dispositivos configurados com pelo menos uma entrada a monitorizar,
  // vai buscar as informações e arranca as threads.
  ////////////////////////////////////////////////////////////////////////////
    if (listaDispositivosBD.Count > 0)
```

```
  {
////////////////////////////////////////////////////////////////////////////////
// Criar lista de threads a arrancar. Uma por cada serviço
////////////////////////////////////////////////////////////////////////////////
      foreach (DispositivoBD dispositivo in listaDispositivosBD)
      {
          DispositivoBD dispositivoMonitorizar;

          try
          {
              dispositivoMonitorizar = DispositivoBD.
                        ObterConfiguraçõesDispositivoBD(dispositivo.presenteNaBD);

              AddDispositivoListaMonitorizar(dispositivoMonitorizar);

              listaThreadsRecolhaDados.Add(new Thread(new
                           ThreadStart(ServiceMonitorizaçãoDispositivosThread)));
          }
          catch (Exception)
          {
             logoMsg = new LogoUtils("Ocorreu um erro ao carregar da BD os dados"+
                   " referentes ao dispositivo " + dispositivo.endereçoIP + " (" +
                   dispositivo.endereçoMAC + ") " + " e por esse motivo não vai" +
                   " ser monitorizado.");
              LogoUtils.EscreveMensagem(logoMsg);
          }
      }
////////////////////////////////////////////////////////////////////////////////
// ArrancarThreads
////////////////////////////////////////////////////////////////////////////////
      foreach (Thread threadRecolhaDados in listaThreadsRecolhaDados)
      {
          threadRecolhaDados.Start();
      }

      do
      {
          //mensagemEnviar = i.ToString();
          //i++;
          System.Threading.Thread.Sleep(tempo);
```

```
        } while (!stopService);

        // Aguarda 500ms para que a tarefa serviceEnviaMensagemCorreioThread acabe.
        i = 0;
        do
        {
            System.Threading.Thread.Sleep(100);
            i++;
        } while (serviceEnviaMensagemCorreioThread.ThreadState !=
                                                    ThreadState.Stopped && i < 5);

        if (serviceEnviaMensagemCorreioThread.ThreadState == ThreadState.Stopped)
        {
            //MessageBox.Show("A thread ServiceEnviaMensagemCorreioThread finalizou
            //correctamente");
            logoMsg = new LogoUtils("A thread serviceEnviaMensagemCorreioThread" +
                                                      "finalizou corretamente.");
            LogoUtils.EscreveMensagem(logoMsg);
        }
        else
        {
            //MessageBox.Show("A thread ServiceEnviaMensagemCorreioThread NÃO
            //finalizou correctamente");
          logoMsg = new LogoUtils("A thread serviceEnviaMensagemCorreioThread NÃO" +
                                           " finalizou corretamente.");
            LogoUtils.EscreveMensagem(logoMsg);
        }
        logoMsg = new LogoUtils("O serviço parou corretamente.");
        LogoUtils.EscreveMensagem(logoMsg);
        //MessageBox.Show("A sair da thread main");
    }
    else
    {
        logoMsg = new LogoUtils("O serviço não detetou nenhum dispositivo" +
                              " configurado para recolha de dados na base de dados" +
                              " e por esse motivo vai parar.");
        LogoUtils.EscreveMensagem(logoMsg);
        ControladorDoServico.ParaServico();
    }
}
```

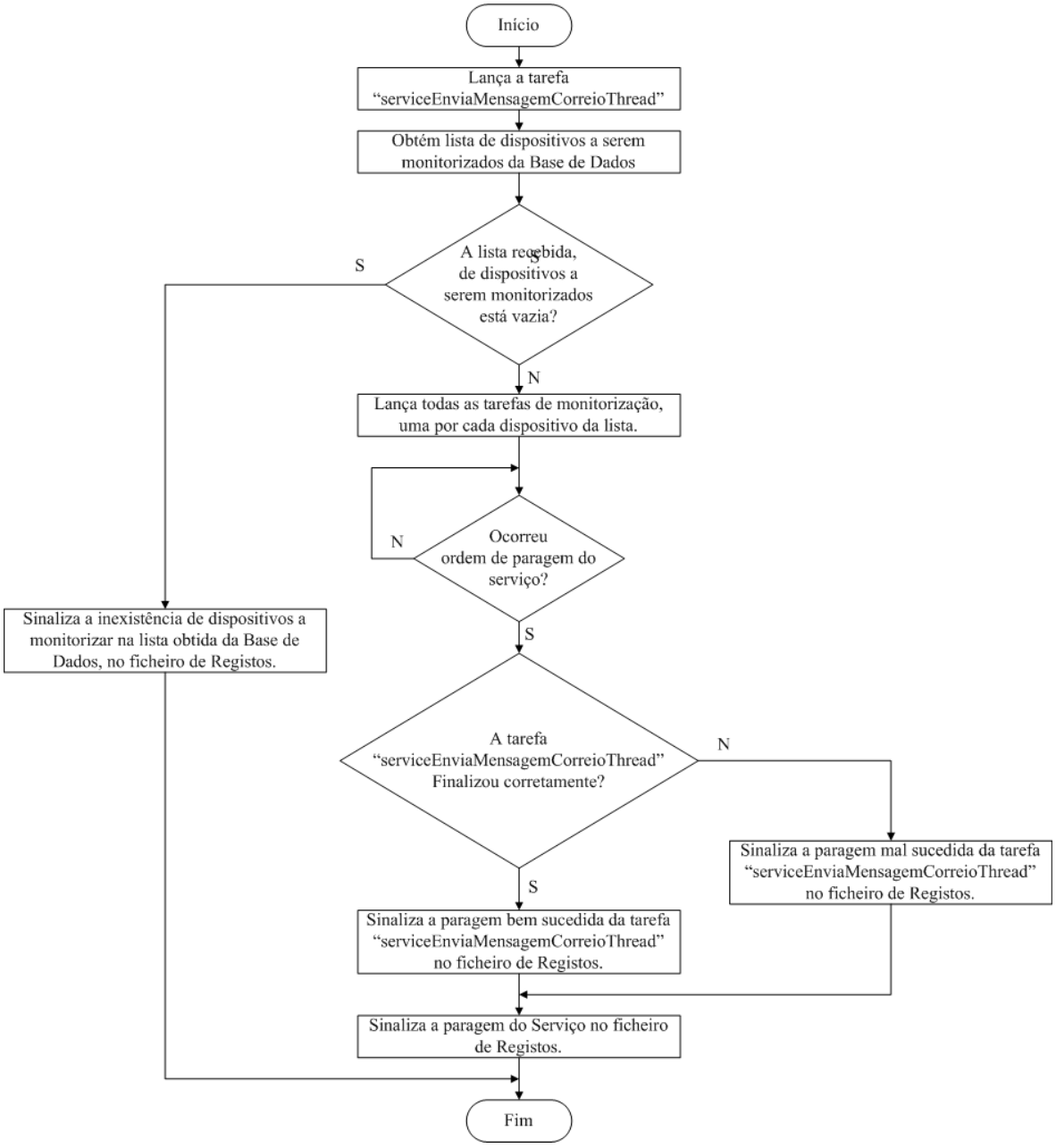


Figura 96: Fluxograma de processamento do método"ServiceMainThread".

Da análise do código de implementação do método "ServiceMainThread" e do seu fluxograma representativo apresentado na figura anterior, podemos analisar as seguintes etapas:

1. É inicializada e lançada em modo concorrente a tarefa que gere o envio das mensagens de correio eletrónico ("serviceEnviaMensagemCorreioThread") em caso de alarme (SOSMsg). Optou-se por uma tarefa específica para controlar o envio das mensagens de correio para que essa tarefa ocorra de forma assíncrona,

não bloqueando as tarefas de monitorização com o processo de envio destas mensagens (que está dependente do controlo de programas terceiros). Esta tarefa é alimentada por uma lista de mensagens (“listaMensagensEnviar”) onde vão sendo inseridas todas as mensagens a enviar pelo sistema. Uma vez que esta lista é acedida por diversas tarefas em simultâneo, o seu acesso é controlado por um semáforo.

2. É obtida da base de dados uma lista (“listaDispositivosBD”) contendo a informação de todos os dispositivos RenPAD configurados com pelo menos uma entrada a monitorizar.

3. Se a lista de dispositivo obtida anteriormente não contiver nenhum elemento, o serviço é terminado.
4. Caso a lista de dispositivos (“listaDispositivosBD”) obtida anteriormente tenha pelo menos um elemento, é criada e iniciada uma tarefa de recolha e monitorização de dados (“threadRecolhaDados”) independente para cada elemento dessa lista.

5. Neste ponto, a tarefa “ServiceMainThread” fica parada indefinidamente enquanto não ocorra um evento de paragem que coloque a variável “stopService” a *true*.

6. Quando a variável “stopService” ganha o valor *true*, todas as tarefas recebem o sinal de paragem. Uma vez que a tarefa que gere o envio das mensagens de correio eletrónico (“serviceEnviaMensagemCorreioThread”) é a mais sensível a um caso de paragem mal controlada, com risco de se perderem avisos de ocorrência de alarmes importantes ao gestor do sistema, o seu correto encerramento é monitorizado pela tarefa “ServiceMainThread” e o resultado dessa monitorização é inscrito no ficheiro de registos.

Depois de lançada a tarefa “ServiceMainThread”, todo o funcionamento do serviço fica dependente da tarefa que gere o envio das mensagens de correio eletrónico (“serviceEnviaMensagemCorreioThread”) e de todas as tarefas de recolha e monitorização.

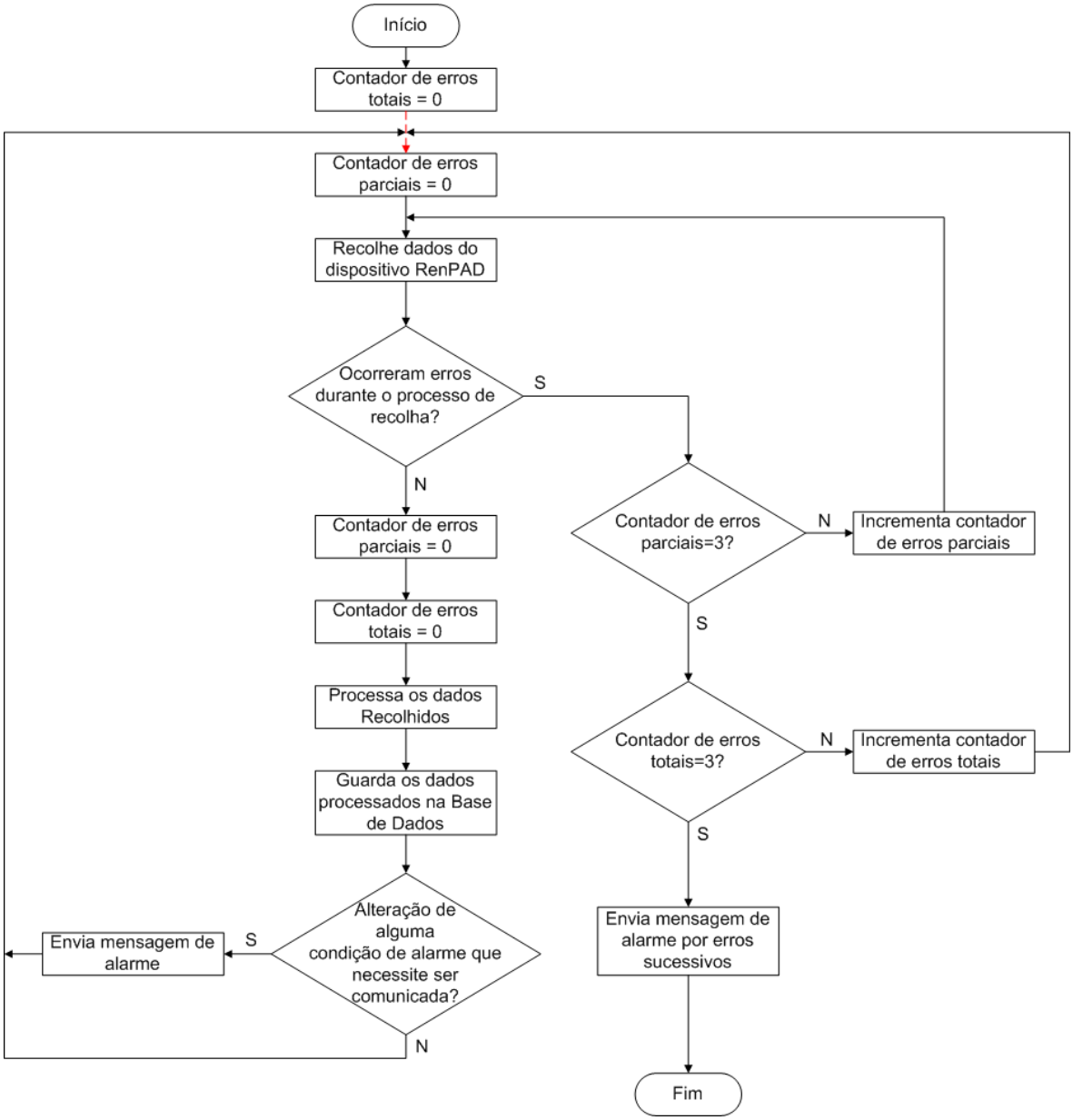


Figura 97: Fluxograma básico de funcionamento de uma tarefa de recolha e monitorização.

Como se pode analisar no fluxograma na figura anterior, cada tarefa de monitorização de dados implementa todas as funções relativas a:

- Recolha de todos os dados referentes às entradas e sensores da respetiva placa de aquisição de dados, configurados na base de dados para serem monitorizados.
- Verificação e validação dos dados recolhidos: se estão dentro dos parâmetros predefinidos, se fazem sentido ou se representam um erro de leitura *etc*.

- Inserção desses dados na base de dados. Os dados são preenchidos na tabela Leituras, conforme descrito na secção anterior.
- Inserção na lista "listaMensagensEnviar" de eventuais mensagens de erro (SOSMsg) que sejam passíveis de geração desse tipo de mensagens.

Como em todos os sistemas, também este não é imune a avarias. Para tentar despistar anomalias de funcionamento foi também implementado o método que podemos analisar no fluxograma da figura anterior e que é seguidamente pormenorizado, que embora básico, poderá fornecer informações úteis em algumas situações:

- Perante a ocorrência de um erro durante a obtenção de uma resposta do dispositivo RenPAD, a tarefa de recolha de dados volta a tentar obter uma nova resposta válida três vezes consecutivas intervaladas de um segundo entre cada vez (erro parcial = 3 ciclos).
- Caso o erro persista durante essas três tentativas, aborta a operação e volta a tentar novamente no período programado para a recolha seguinte, durante mais três ciclos da mesma forma que é descrita no ponto anterior (erro total = 3 x erro parcial = 3 x 3 x ciclos de 1 segundo).
- Caso ao fim desses três ciclos do erro total a resposta dê sempre erro, a tarefa é terminada. É também colocada na lista de mensagens de correio a enviar uma mensagem, referindo que a monitorização deste dispositivo foi suspensa porque este deixou de responder por avaria, ou foi desligado, ou qualquer outro problema.
- O mesmo sistema é utilizado para os erros que ocorrem em cada entrada. Se uma entrada tiver cinco erros consecutivos em cinco leituras diferentes é enviada uma mensagem de correio e é abortada a monitorização dessa entrada.
- Para garantir que, no caso de não existirem endereços de correio eletrónico configurados para envio das mensagens estas sejam entregues a alguém e não se percam, as mensagens deste tipo são enviadas, não só para os responsáveis do sistema que tenham o endereço de correio registado no sistema, mas também para os supervisores cujos endereços estão configurados na secção "appSettings" do ficheiro de configurações "RenPad.exe.config". Por exemplo:

```
<add key="mailTo" value="david.peneireiro@renova.pt"/>
```

Também com o intuito de deteção de avarias, de mal funcionamentos, de análise de funcionamento da aplicação *etc*., existe uma forma de *tracking* de ocorrências. Como podemos verificar, ao longo de toda a aplicação existem duas instruções que se repetem:

```
logoMsg = new LogoUtils("--- Mensagem ---");
LogoUtils.EscreveMensagem(logoMsg);
```

Esta é a forma como podemos inserir mensagens no ficheiro de registos (“RenPAD.log”) da aplicação. Todas as entradas neste ficheiro são datadas e separadas como mostra a imagem seguinte:

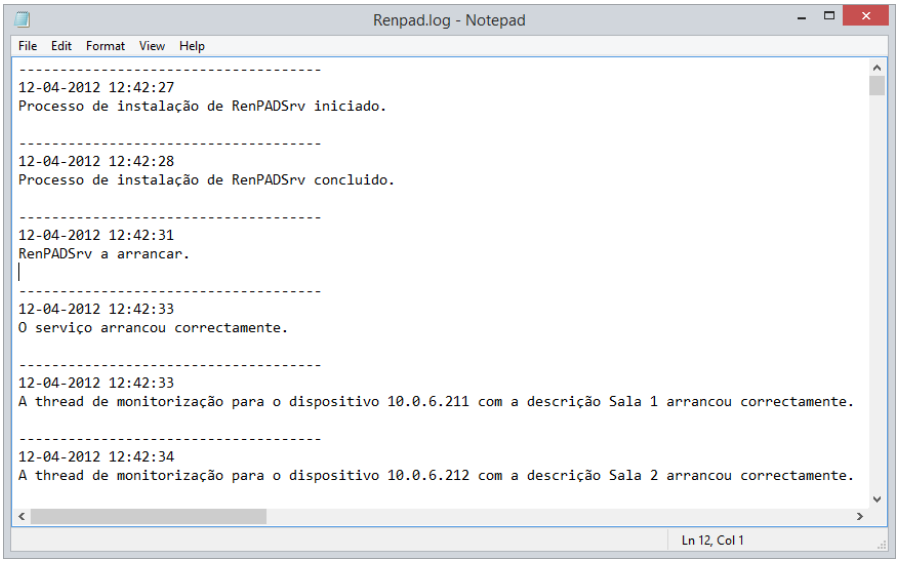


Figura 98: Exemplo de informação contida num ficheiro “RenPAD.log”.

A forma de implementação deste mecanismo baseia-se na criação de uma tarefa por mensagem a inserir no ficheiro, responsável pela correta inserção dessa mensagem de forma concorrente sem prejudicar o normal funcionamento do restante programa. Uma vez que o ficheiro apenas pode ser acedido por uma tarefa de cada vez, foi utilizado um *Mutex* (uma variante de semáforo) que restringe esse acesso. Todo o código referente ao processo de inserção de mensagens no ficheiro de registos pode ser consultado no Anexo 8, na livraria “LogoUtils.cs”.

4.3 – Programa de configuração e análise de dados

Como referido anteriormente, este é o programa de configurações gerais e análise de dados.

Implementa uma *interface* gráfica, de utilização fácil e intuitiva por meio de separadores (abas), que permite tanto a disponibilização dos dados armazenados na base de dados para análise, como ainda todas as ferramentas necessárias aos processos de instalação, configuração e manutenção dessa mesma base de dados e do serviço de recolha de dados. Permite ainda a configuração do modo de funcionamento e suas variáveis nas placas de aquisição de dados RenPad presentes na rede informática tais como: configuração de alarmes, configuração do endereço de rede, acerto horário, obtenção dos valores presentes em cada sensor, entre outros.

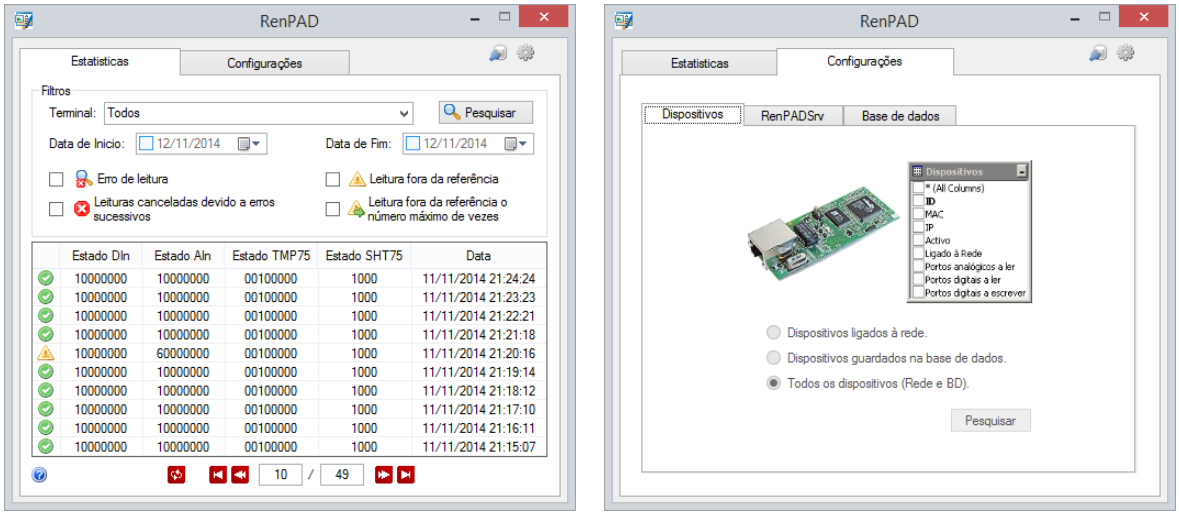


Figura 99: *Interface* gráfica do programa Monitorizador RenPAD: Separador de visualização de dados e separador de configurações diversas.

Na figura anterior podemos observar os dois separadores principais do programa:

- Separador de visualização de dados armazenada na base de dados, denominada de "Estatísticas". Este separador será descrito com maior rigor na secção 4.3.2.
- Separador de configurações diversas, denominado de "Configurações".

4.3.1 – Separador "Configurações" – subseparador "Dispositivos"

Dado que nas duas secções anteriores já foram abordados indiretamente os subseparadores "RenPADSrv" e "Base de Dados" deste separador de configurações, nomeadamente na descrição dos processos referentes às operações possíveis de realizar

para o funcionamento da base de dados e do serviço RenPADSrv, nesta secção apenas se irá descrever com maior pormenor o subseparador Dispositivos.

Como facilmente se depreende no decorrer desta tese, o projeto define dois paradigmas de deteção e sinalização de anomalias distintos e não exclusivos, isto é: a placa de aquisição de dados RenPAD pode ser instalada e configurada num modo "*stand alone*", em que fica responsável única pela deteção e sinalização de eventuais sinais de alarme que ocorram durante o seu funcionamento, conforme descrito na secção 3.6.1 no ponto "tarefas.lib" (tarefa "trata_saidas") e no Anexo 7. Neste modo de funcionamento não há necessidade de recolher os valores presentes nas suas entradas para a base de dados para futura análise, e por conseguinte o programa Monitorizador RenPAD é utilizado apenas para realizar as respetivas configurações da placa. É no entanto possível em qualquer altura alterar esse paradigma de funcionamento, passando a armazenar na base de dados os dados recolhidos para futura consulta e análise, de forma fácil e configurável independentemente, passando o serviço associado ao programa Monitorizador RenPAD a detetar e assinalar os eventos de erro configurados na mesma base de dados. É ainda possível configurar a solução para que uns eventos de alarme sejam sinalizados apenas pela placa RenPAD, outros apenas pelo Monitorizador RenPAD e outros ainda de ambas as formas.

Para responder a esta liberdade de escolha de metodologias de funcionamento, foi idealizado e implementado uma *interface* simples, que permite realizar as operações de configuração desejadas para cada dispositivo em todas as vertentes possíveis:

Figura 100: Opções disponíveis no subseparador de configuração de dispositivos.

4.3.1.1 – Opção: “Dispositivos ligados à rede”.

Esta opção permite configurar todas as variáveis de uma placa de aquisição de dados RenPAD.

Quando se escolhe esta opção é apresentado um quadro que permite configurar o tipo de pesquisa por dispositivos na rede que pretendemos:

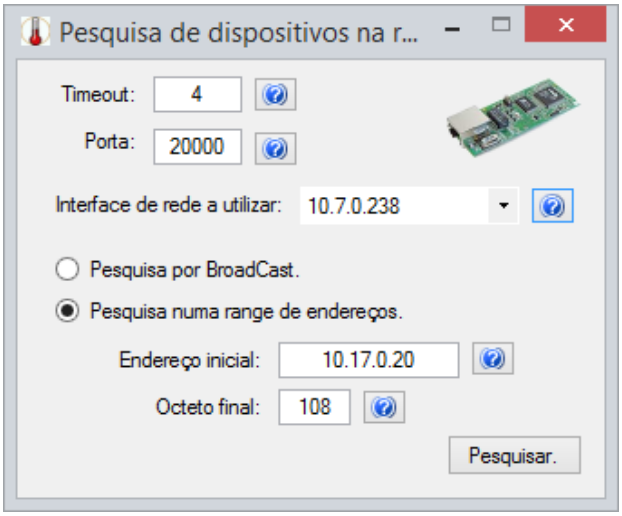


Figura 101: Propriedades de pesquisa de dispositivos na rede *Ethernet*.

- “TimeOut” – representa o tempo máximo que pretendemos aguardar pela resposta de um dispositivo.

  Dependendo do condicionamento do tráfego da rede informática, condições de funcionamento da mesma ou outros fatores, a resposta de um dispositivo pode demorar mais ou menos tempo a chegar ao servidor onde está instalado o programa Monitorizador RenPAD e de onde partiu a inquirição por dispositivos na rede. Com este parâmetro podemos ajustar o tempo máximo que o programa espera por uma resposta.

- “Porta” – Define a porta de comunicações sobre a qual os dispositivos estão à escuta.
- “Interface de rede a utilizar” – Define o *interface* de rede físico da máquina onde está instalado o Monitorizador RenPAD por onde sairá a pergunta de pesquisa de dispositivos na rede.

Uma vez que uma máquina pode ter mais que um *interface* de rede configurado, devido ao fato de poder ter duas placas de rede instaladas, ou por ter algum tipo de *software* que crie *interfaces* desse tipo por exemplo (p.e. *VMWare*), é necessário definir qual o *interface* a utilizar pelo Monitorizador RenPAD para que a pergunta de pesquisa saia pelo *interface* que garanta que o segmento de rede *Ethernet* onde estão efetivamente ligadas as placas RenPAD seja atingível.

- "Pesquisa por BroadCast" – Indica que a pergunta por dispositivos na rede é difundida por *BroadCast* através de um comando que tem por base o protocolo *UDP*. Esta é a forma mais rápida de se obterem todos os dispositivos na rede, mas tem a particularidade de se poderem perder algumas respostas de dispositivos que demorem mais tempo que o definido em "Timeout" a chegar, e não existe nenhum controlo sobre isso, a não ser enviar a pergunta diversas vezes e esperar que numa delas o dispositivo pretendido apareça. Além disso o segmento de rede entre o servidor e os dispositivos deve permitir a difusão de comandos por *UDP*. Existem dispositivos de rede (*hub's*, *switches etc*.) que bloqueiam esse tipo de pacotes de dados fazendo com que a pergunta nunca chegue ao dispositivo pretendido. A título de exemplo, na Renova todos os dispositivos RenPAD estão configurados num segmento de rede próprio com uma gama de *ip's* distintos e não são visualizáveis através deste tipo de comandos por qualquer máquina que se encontre fora dessa gama de *ip's*, fazendo com que quando se corre este comando a resposta recebida seja:

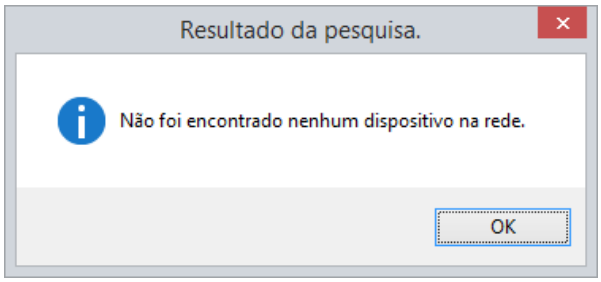


Figura 102: Resposta recebida a um comando de pesquisa de dispositivos por comando *UDP* na rede da Renova, por uma máquina configurada numa gama de *ip's* diferente da dos dispositivos.

- "Pesquisa numa range de endereços" – Para contornar as limitações da opção anterior e uma vez que os dispositivos podem estar inatingíveis a comandos enviados por protocolo *UDP*, mas serem atingíveis por comando enviado por protocolo *TCP-IP*, construiu-se este mecanismo, que quando selecionado executa

uma rotina que pesquisa por todos os dispositivos com endereços definidos entre os campos "endereço (*ip*) inicial" e "octeto final". Como exemplo, assumindo o caso da Figura 101 em que o "endereço inicial" está configurado como 10.17.0.20 e o "octeto final" como 108, a tarefa de pesquisa realiza uma pergunta individual por *TCP-IP* através do "comando 01" descrito no Anexo 6, enviada pela Porta configurada, para cada um dos endereços situados na gama entre 10.17.0.20 e 10.17.0.108 aguardando por uma resposta durante o tempo definido em "Timeout". Caso se obtenha resposta afirmativa, este endereço pertence a um dispositivo RenPAD, caso não exista resposta nenhuma, este endereço não está associado a uma placa RenPAD. Como podemos deduzir, estre processo é muito moroso, e demora mais consoante o número de endereços envolvidos na pesquisa.

- Depois de obtidos os endereços associados a dispositivos RenPAD presentes na rede, é realizada um conjunto de questões ao dispositivo e o resultado é apresentado numa lista de resultados como a da figura seguinte:

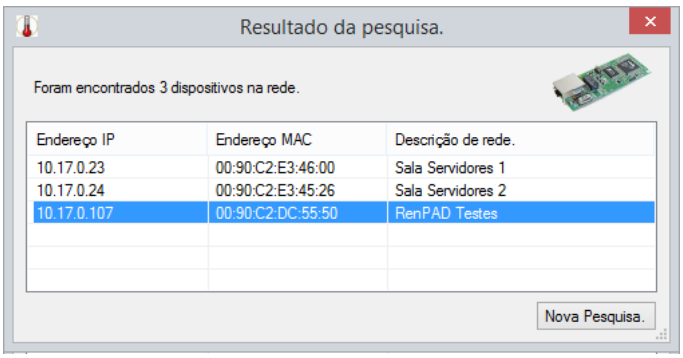


Figura 103: Resultado da pesquisa por dispositivos na rede.

Para esta lista de resultados (e todas as outras que irei referir posteriormente) foram criados métodos de ordenação pelas categorias consideradas mais importantes (neste caso podemos ordenar a lista por Endereço *IP* ou por Endereço *MAC*), para se conseguir pesquisar mais facilmente pelo dispositivo que se deseja numa lista que pode conter diversos dispositivos.

Depois de obtidos os dispositivos presentes na rede, basta escolher aquele que se deseja configurar. Esse dispositivo é novamente inquirido, são obtidos todos os parâmetros nele configurados e os valores atuais de todas as entradas e sensores. Estes dados são apresentados numa nova janela em formato de separadores:

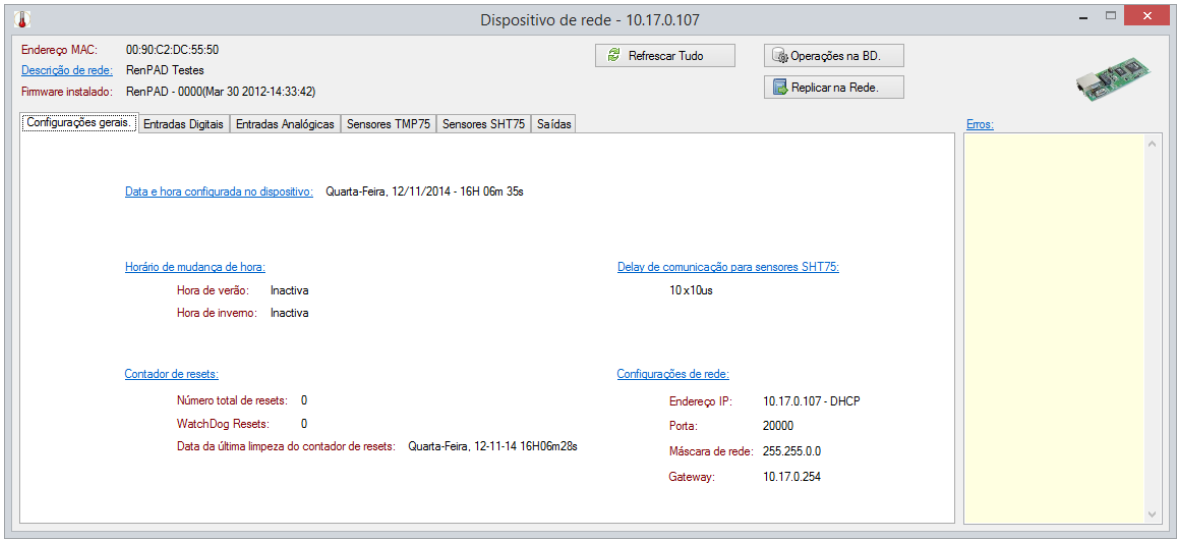


Figura 104: Dispositivo de rede: configurações gerais.

A figura anterior mostra a janela carregada inicialmente, cujo separador ativo é por defeito o de "Configurações gerais" e que apresenta as variáveis de funcionamento definidos.

Apenas as variáveis cuja *label* se apresenta em formato de *hiperlink* (a azul sublinhado) são alteráveis, e podem ser alteradas clicando na respetiva *label*, processo este que despoleta o aparecimento de uma janela de configuração da variável a alterar.

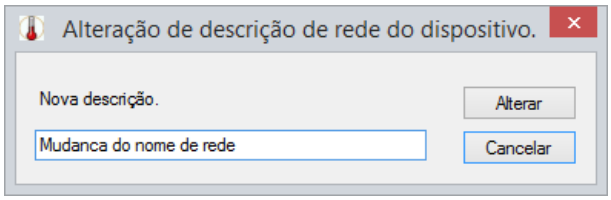


Figura 105: Exemplo de janela de configuração de variável a alterar – "Descrição de rede do dispositivo".

A alteração de algumas destas variáveis implica o intercâmbio de informação entre janelas (por exemplo a alteração da descrição de rede de um dispositivo altera a descrição da rede tanto nesta janela, como na anterior que apresenta o resultado de pesquisa de dispositivos na rede) enquanto outras implicam o encerramento de todas as janelas abertas até à janela inicial, obrigando a nova pesquisa de dispositivos na rede e novo carregamento de dados (como é o caso de uma reinicialização de um dispositivo, ou uma alteração do endereço de rede).

O separador seguinte mostra a secção referente às entradas digitais: seus estados aquando da recolha de informação e suas configurações:

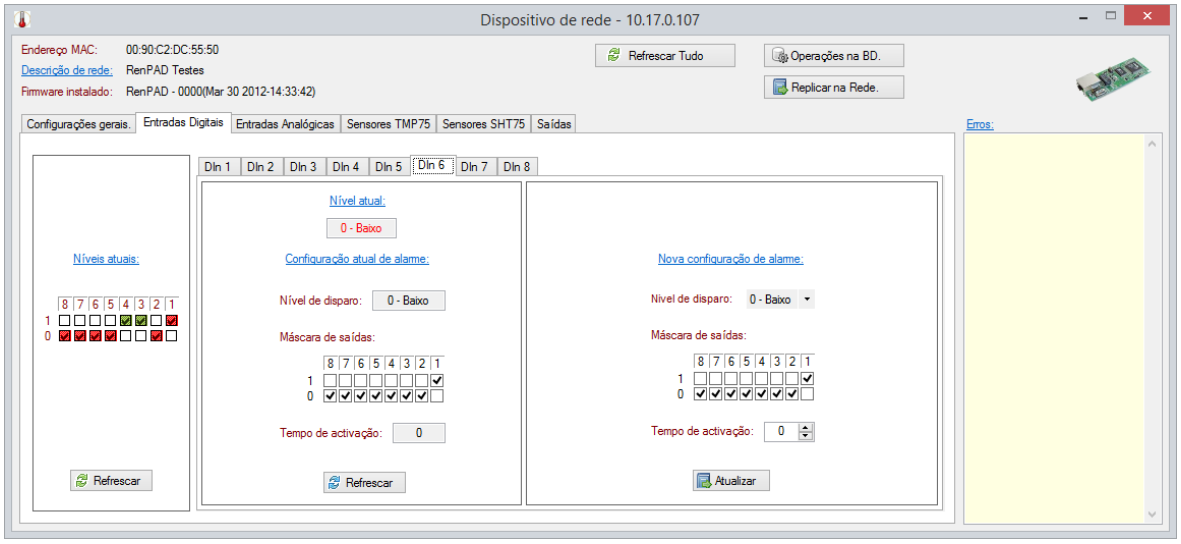


Figura 106: Dispositivo de rede: "Entradas digitais".

Neste separador referente às entradas digitais e suas configurações podemos diferenciar as seguintes áreas:

- No quadro mais à direita, com a *label* "Níveis atuais" podemos visualizar a que nível se encontra cada uma das entradas digitais, e se está no nível predefinido (a verde) ou não (a vermelho). Neste quadro, tanto o botão "Refrescar" como o *hiperlink* "Níveis atuais" recarregam do dispositivo, os dados referentes a este quadro.
- No quadro da esquerda são apresentados oito separadores intitulados por "DIn X" referentes às configurações de cada uma das entradas digitais.
- Dentro de cada separador "DIn X" existem dois quadros:
    - Um à esquerda que contém todos os valores definidos nas configurações do dispositivo RenPAD. Neste quadro, tanto o botão "Refrescar" como o *hiperlink* "Níveis atuais" recarregam do dispositivo, os dados referentes a este quadro.
    - Um à direita onde poderão ser alterados essas configurações. Para proceder a essa alteração, definimos as novas configurações pretendidas e passamo-las para o dispositivo acionando o botão "Atualizar" ou o *hiperlink* "Novas configurações de alarme". (ver a descrição de configurações de alarmes no Anexo 7).

Uma vez que todos os outros separadores têm funcionamento idêntico a este, apenas irei apresentar as imagens a eles referentes descrevendo apenas um ou outro pormenor mais importante.

O separador seguinte mostra a secção referente às entradas digitais: seus valores aquando da recolha de informação e suas configurações:

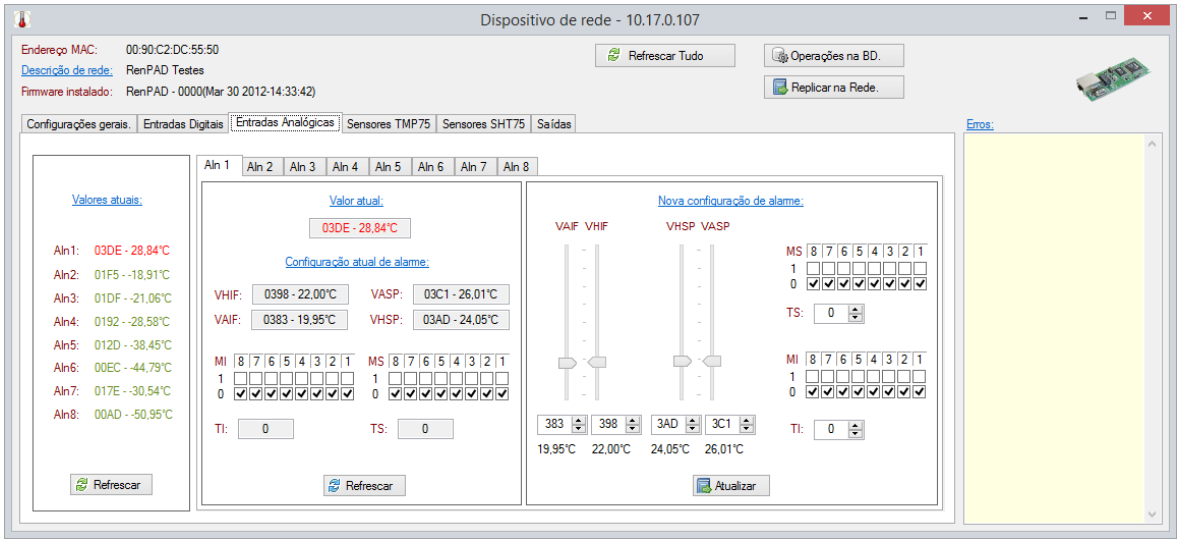


Figura 107: Dispositivo de rede: “Entradas Analógicas”.

Note-se que no quadro “Valores atuais” apenas AIn 1 está conectada realmente a um sensor LM60 que apresenta 28.84ºC de temperatura. Todas as outras entradas apresentam valores “disparatados”. Este fenómeno deve-se à necessidade de conectar as estradas destes sensores à massa quando estas não estão em utilização, processo esse que não foi realizado na altura desta leitura para poder ser aqui apresentado.

Na janela de “Novas configurações de alarme” podem-se destacar ainda duas particularidades:

- Os valores de VAIF, VHIF, VHSP e VASP não dão temperaturas lineares mas sim em “saltos” (por exemplo, o valor hexadecimal 383 corresponde a uma temperatura de 19.95ºC enquanto que o valor seguinte 384 corresponde a uma temperatura de 20,05ºC). Este fato prende-se com a implementação da equação de conversão que pode ser analisada na secção 3.1.3.
- Os valores de VAIF, VHIF, VHSP e VASP referentes a estas entradas analógicas são os únicos definidos em hexadecimal, e não é graus centígrados como no caso dos sensores TMP75 e SHT75. Este fato prende-se com uma possível futura

integração com outros tipos de sensores analógicos. Caso a seleção destes valores fosse realizada em graus centígrados, quando se pretendesse configurar um sensor de caudal, por exemplo, esse processo de configuração não faria muito sentido e confundiria o utilizador. Não obstante a conversão é apresentada numa *label* própria para uma configuração mais simplificada mas que não interfere no processo.

O separador seguinte mostra a secção referente aos oito sensores TMP75: seus valores aquando da recolha de informação e suas configurações:

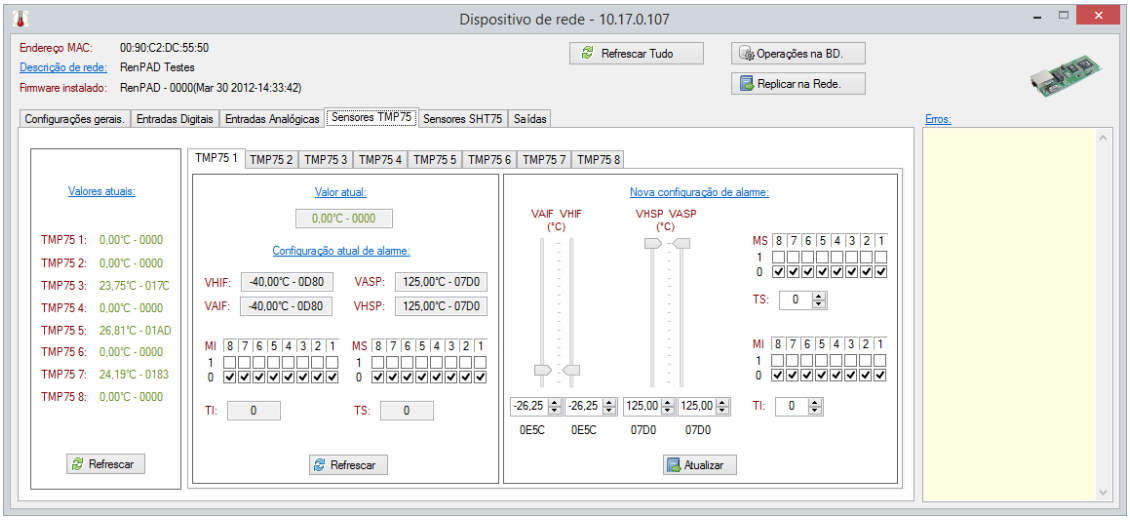


Figura 108: Dispositivo de rede: “Sensores TMP75”.

O separador seguinte mostra a secção referente aos dois sensores SHT75: seus valores aquando da recolha de informação e suas configurações:

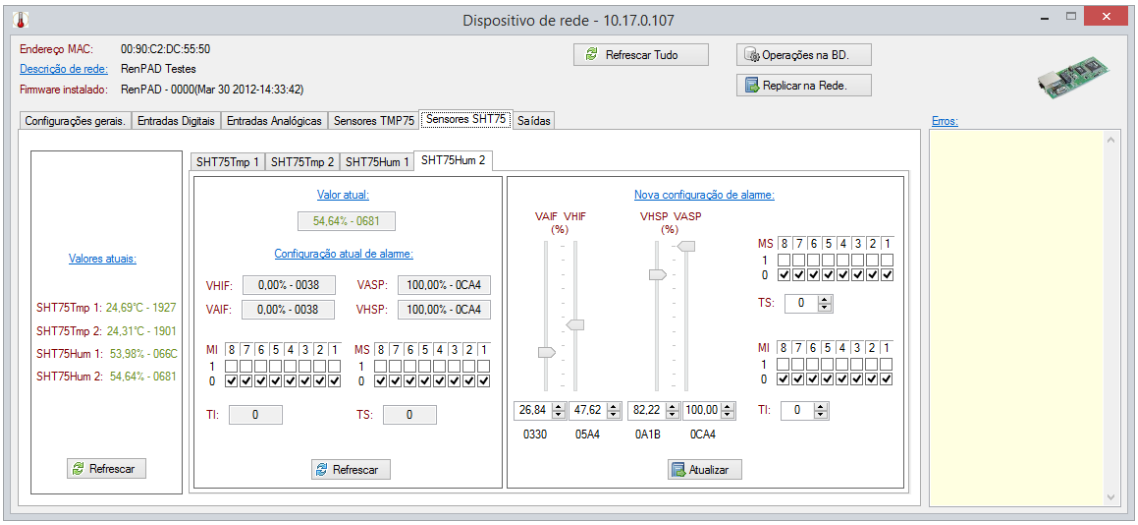


Figura 109: Dispositivo de rede: “Sensores SHT75”.

O separador seguinte mostra a secção referente ao estado das saídas digitais e respetivos testes que se podem realizar:

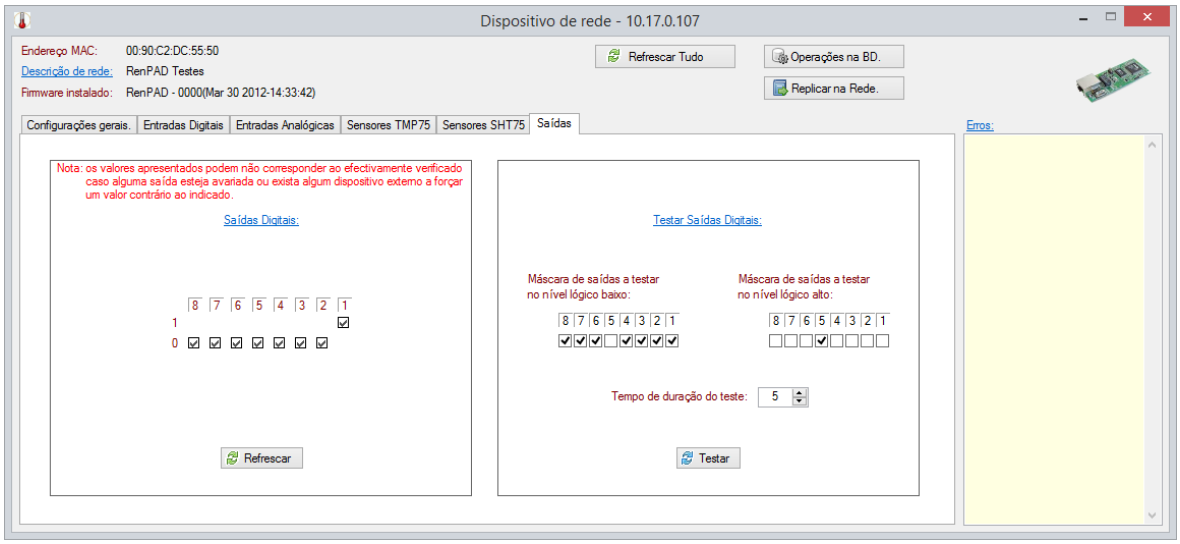


Figura 110: Dispositivo de rede – "Saídas".

Neste separador podemos visualizar à esquerda o quadro correspondente ao estado de cada saída digital. No caso apresentado podemos verificar que a saída 1 está no nível lógico alto, assinalando o alarme referente à entrada digital DIn 6 (ver Figura 106).

No quadro da direita podemos configurar as saídas a testar, em que nível as queremos testar e o tempo de duração do teste.

Já no canto superior direito desta janela de "Dispositivo de rede" existem 3 botões de comando:

- "Refrescar tudo" – providência o refrescamento de todos os dados presente na aplicação com os dados efetivos do dispositivo.
- "Replicar na rede" – permite replicar todos os parâmetros selecionados para outro dispositivo. Isto possibilita, de forma rápida configurar um dispositivo recém-introduzido na rede com os dados de um outro dispositivo previamente configurado poupando tempo e trabalho, desde que as configurações dos dados a sincronizar sejam iguais.

  Ao selecionar este botão é apresentada a janela de campos a replicar, onde são selecionados todos os campos comuns aos dois dispositivos e que se desejam replicar:

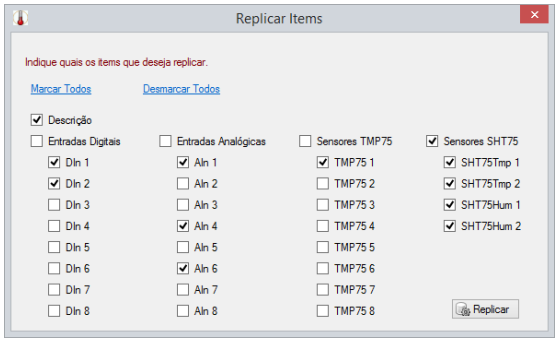


Figura 111: Campos a replicar entre dispositivos de rede.

Depois de escolhidos os campos a replicar é apresentada uma janela com todos os dispositivos presentes na rede e passiveis de receber a replicação:

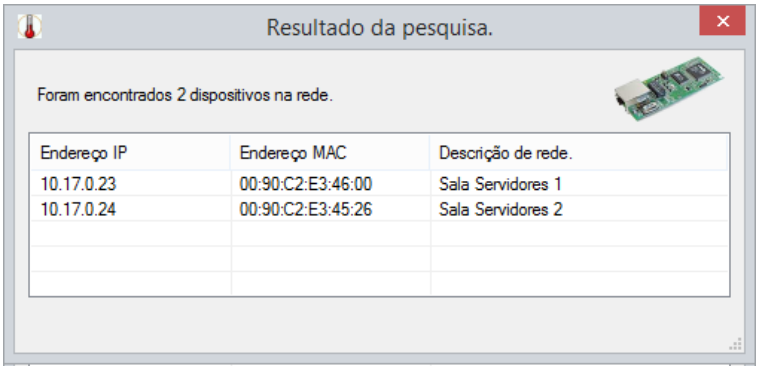


Figura 112: Dispositivos na rede passíveis de receber a replicação de dados.

Depois de escolhido o dispositivo pretendido os dados são replicados e é aberta uma nova janela de "Dispositivo de rede" referente ao dispositivo que recebeu a replicação. Na figura seguinte é apresentada uma sobreposição de duas janelas de "Dispositivos de rede" representando a replicação do campo "Descrição de rede" do dispositivo 10.17.0.107 para o dispositivo 10.17.0.23.

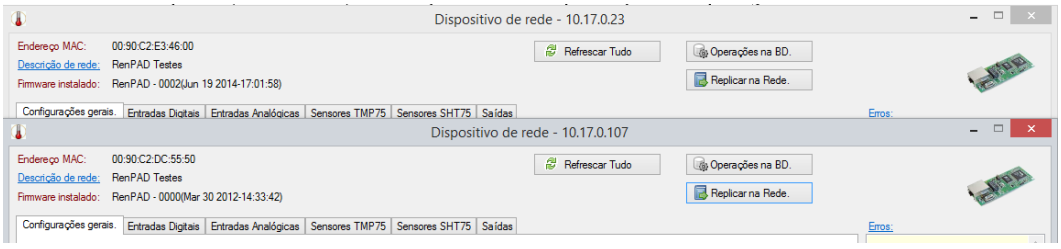


Figura 113: Janelas sobrepostas representando a replicação de dados.

- "Replicar na BD" – permite, de forma rápida replicar para a base de dados o modo de funcionamento de um dispositivo previamente configurado na rede, desde que a configuração dos dados a sincronizar seja a mesma.

O procedimento de replicação de dados para a base de dados é semelhante ao processo apresentado no ponto anterior: é lançada a mesma janela de dados a replicar, igual à da Figura 111, e quando se seleciona o botão "Replicar", é realizada uma pesquisa por dispositivos existentes na base de dados com o mesmo endereço *IP* ou o mesmo endereço *MAC* para a seleção de replicação. Nesta fase podem ocorrer 2 cenários:

- Não existe na base de dados nenhum dispositivo configurado com o mesmo endereço *IP* nem com o mesmo endereço *MAC* do dispositivo de rede selecionado – Neste caso é apresentada uma janela com a possibilidade de criação deste dispositivo na base de dados:

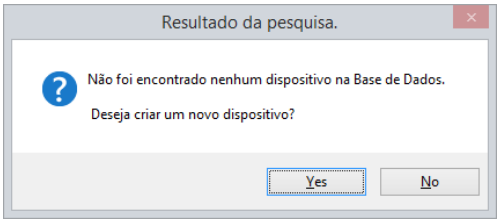


Figura 114: Janela de criação de novo dispositivo na base de dados com as configurações de replicação selecionadas.

Caso se opte pela criação de um novo dispositivo, este é inserido na base de dados com as configurações selecionadas e é aberta uma janela de "Dispositivo na base de dados" (ver secção seguinte).

- Existe na base de dados pelo menos um dispositivo com o mesmo endereço *IP* ou com o mesmo endereço *MAC* do dispositivo de rede selecionado – Neste caso é apresentada uma janela de "Resultados de pesquisa" contendo todos os dispositivos passiveis de serem objeto de replicação de dados:

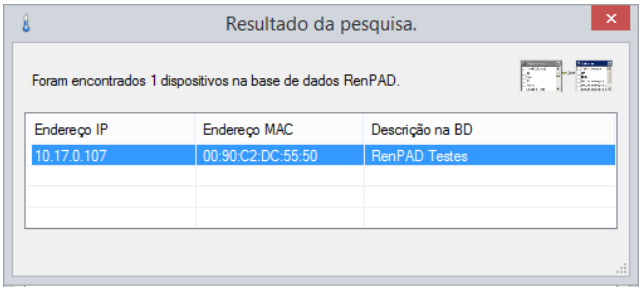


Figura 115: Dispositivos na base de dados passíveis de receber a replicação de dados.

Depois de selecionado o dispositivo na base de dados que se pretende atualizar com os dados a replicar, os campos compatíveis com a replicação

são efetivamente atualizados e no final é aberta uma janela de “Dispositivo na BD” do dispositivo atualizado, onde são apresentados os novos dados configurados (ver secção seguinte).

Na figura seguinte é apresentada uma sobreposição de duas janelas representando a replicação do campo “Descrição de rede” do dispositivo 10.17.0.107 presente na rede para o dispositivo 10.17.0.107 presente na base de dados:

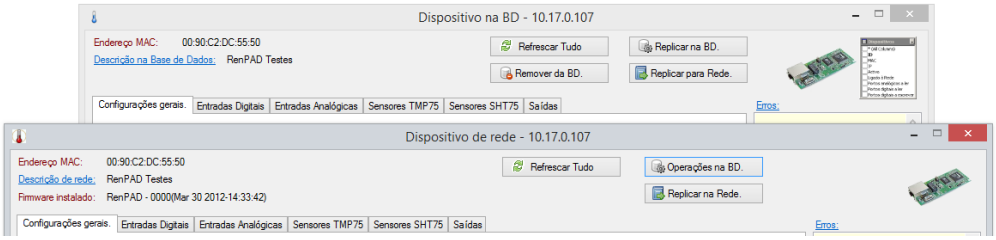


Figura 116: Janelas sobrepostas representando a replicação de dados.

Todos os processos de atualização e recolha e replicação de dados estão sujeitos à eventual ocorrência de erros ou anomalias. Esses eventos são sinalizados e descritos ao pormenor no quadro “Erros”:

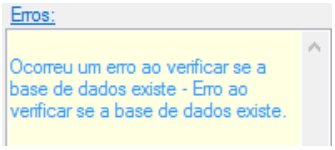


Figura 117: Quadro "Erros" onde são descritos todos os erros e anomalias ocorridos.

4.3.1.2 – Opção: “Dispositivos guardados na base de dados”.

Esta opção permite configurar todas as variáveis de um dispositivo guardado na base de dados.

Um dispositivo guardado na base de dados é um dispositivo reconhecido pela tarefa de recolha de dados, associada ao serviço RenPadSrv descrito na secção 4.2.. Qualquer dispositivo pode estar ligado na rede em funcionamento no modo “*stand alone*”, mas para poder ser monitorizado e os valores presentes às suas diversas entradas serem recolhidos e armazenados na base de dados, este terá que existir e estar devidamente configurado na mesma base de dados.

Assinale-se que o procedimento de criação de um dispositivo na base de dados parte sempre da replicação das configurações de um dispositivo já configurado na rede, conforme descrito na secção anterior.

Uma vez que o método de atualização dos dados, os menus apresentados e as opções disponíveis para as operações abordadas nesta secção são iguais ou muito semelhantes aos abordados na secção anterior, irei apenas descrever mais pormenorizadamente algum aspeto que considero mais importante, apresentando o restante apenas de modo global.

Quando esta opção é selecionada são pesquisados todos os dispositivos configurados na base de dados e o resultado apresentado numa tabela como a da figura seguinte:

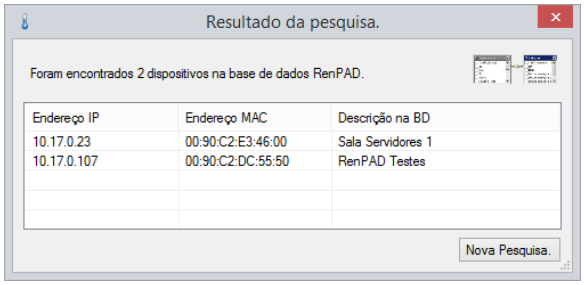


Figura 118: Resultado da pesquisa por dispositivos na base de dados.

E selecionando um dispositivo obtém-se a janela "dispositivo na BD" onde é possível configurar todas as suas opções de funcionamento.

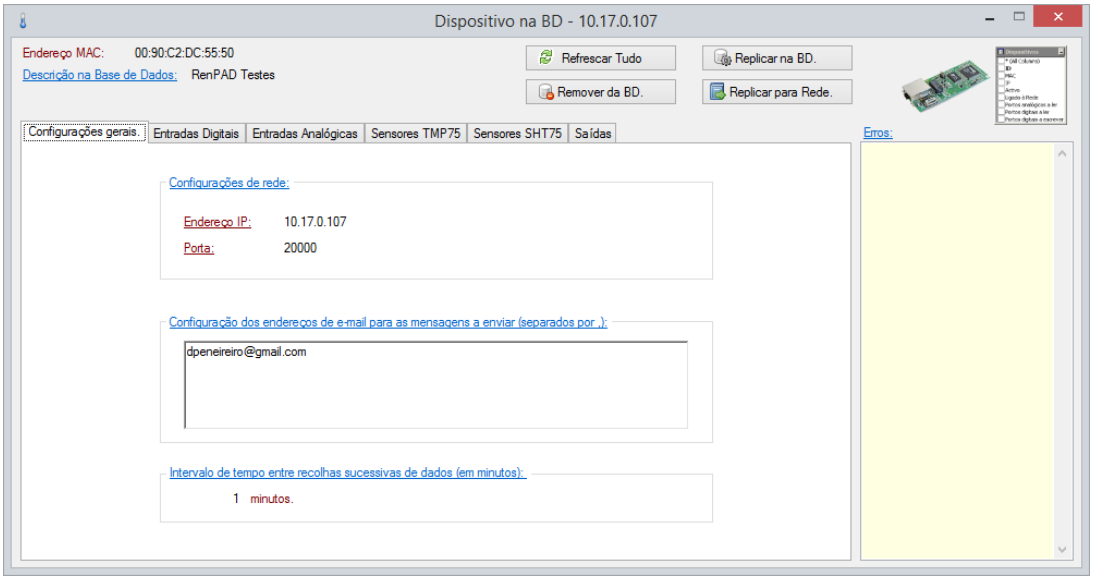


Figura 119: Dispositivo na BD – "Configurações gerais".

É no separador selecionado por defeito no lançamento desta janela denominado por "Configurações gerais" e apresentado na figura anterior, que se pode configurar os endereços de correio eletrónico dos responsáveis por este sistema e para onde serão enviadas todas as mensagens de correio eletrónico. É também neste separador que podemos configurar o intervalo em minutos, associado à recolha sucessiva de dados deste dispositivo. Note-se que a aquisição de dados pode ser executada em intervalos diferentes de dispositivo para dispositivo.

Nas figuras seguintes são apresentados todos os separadores referentes à configuração de todas as entradas e sensores:

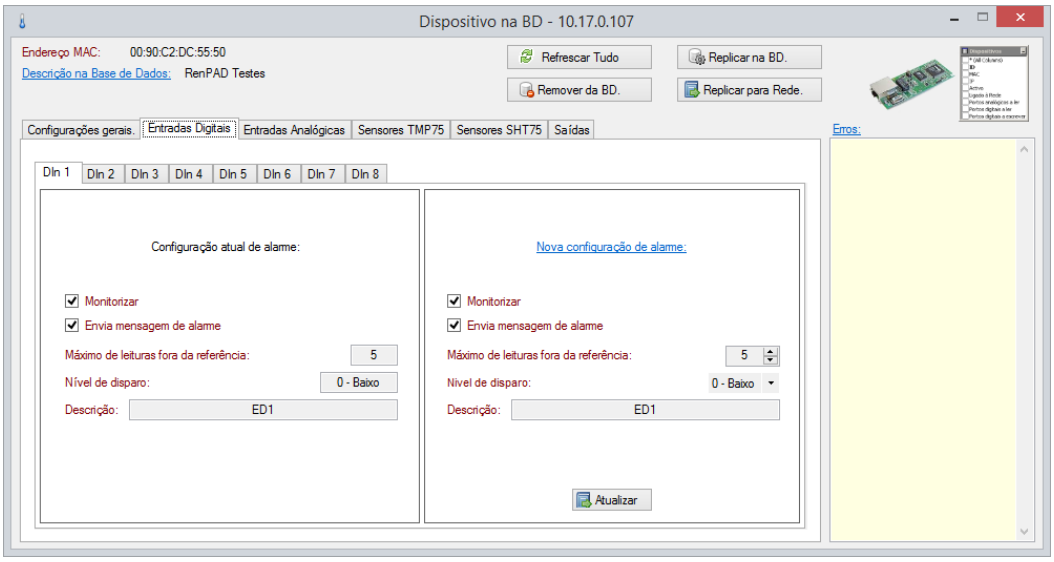


Figura 120: Dispositivo na BD – "Entradas Digitais".

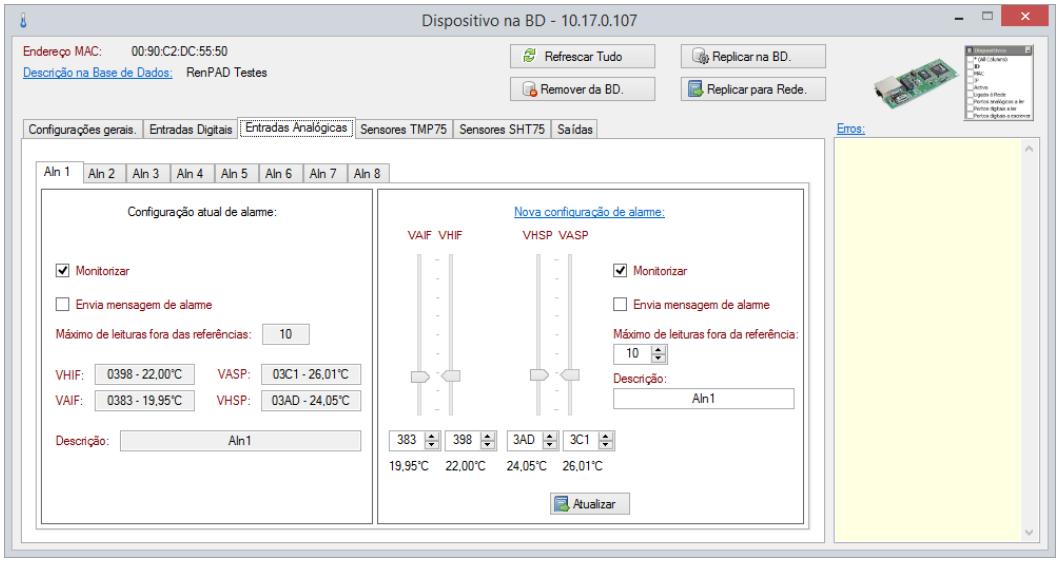


Figura 121: Dispositivo na BD – "Entradas Analógicas".

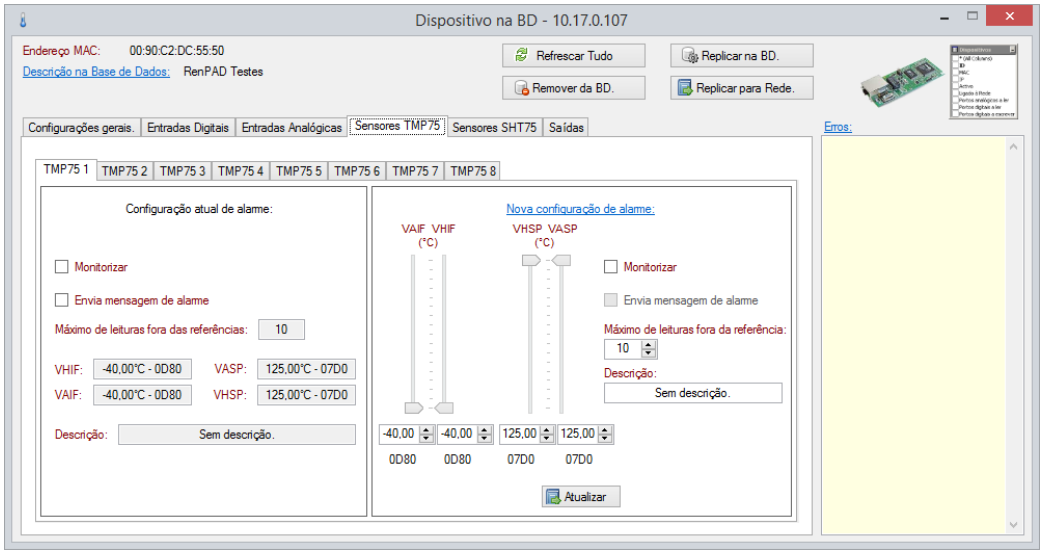


Figura 122: Dispositivo na BD – "Sensores TMP75".

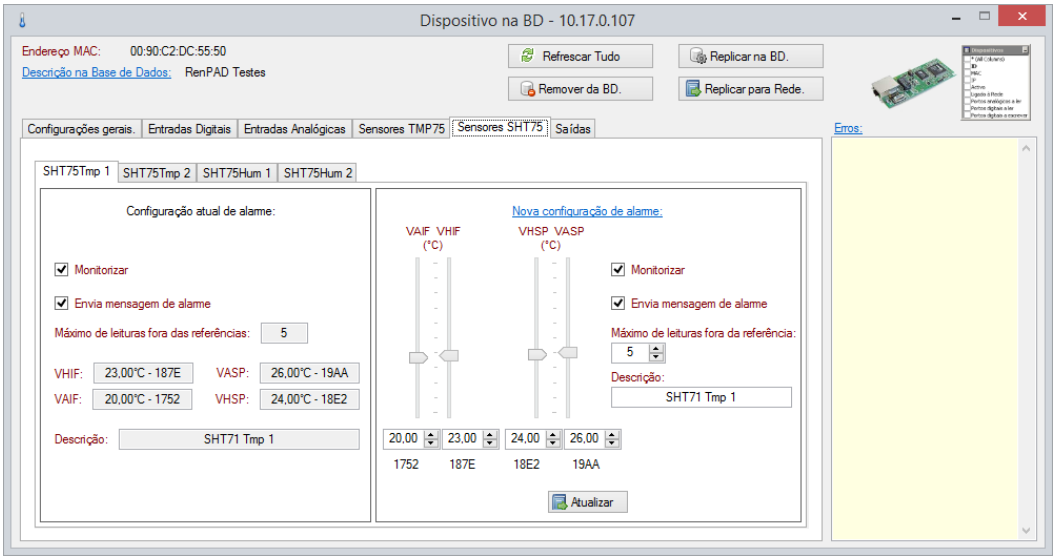


Figura 123: Dispositivo na BD – "Sensores SHT75".

Nas imagens anteriores podemos inferir que cada entrada ou sensor pode ser configurada para que os seus valores sejam monitorizados (recolhidos para a base de dados em intervalos regulares), podem-se configurar os valores limites de funcionamento, o número máximo de vezes consecutivas que um valor recolhido pode permanecer fora do limite estabelecido antes que seja enviada uma mensagem de alarme para os endereços de correio configurados, e a sua descrição para uma mais fácil interpretação dos resultados obtidos. Para mais informações sobre os modos de funcionamento dos alarmes deve consultar-se o Anexo 7, e das restantes configurações a secção 4.1.

O separador "Saídas" apresenta apenas a descrição de cada saída para facilitar a sua identificação:

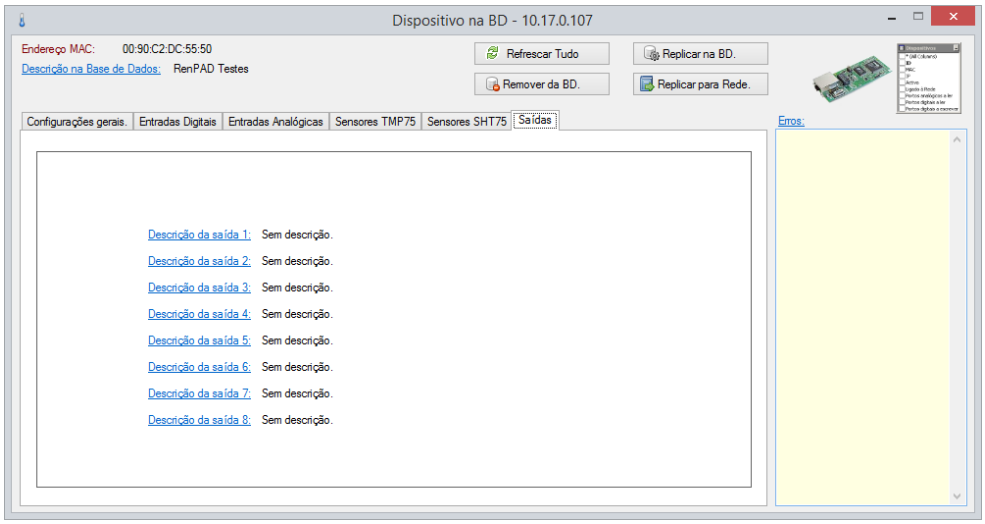


Figura 124: Dispositivo na BD – "Saídas".

No canto superior direito desta janela são apresentados quatro botões com as seguintes funções:

- "Refrescar Tudo" – Recarrega todos os valores efetivamente configurados na base de dados para os respetivos campos apresentados na janela "Dispositivo na BD".
- "Remover da BD" – Permite remover este dispositivo da base de dados eliminando-o completamente mantendo-se no entanto as suas leituras associadas.
- "Replicar para Rede" – permite iniciar o processo de replicação de dados para um dispositivo na rede.

  O procedimento de replicação de dados de um dispositivo configurado na base de dados para um dispositivo presente na rede é semelhante ao processo inverso apresentado na secção anterior: é lançada a mesma janela de dados a replicar, igual à da Figura 111, e quando se seleciona o botão "Replicar", é realizada uma pesquisa por dispositivos existentes na rede através do mesmo método descritivo da Figura 101, para a seleção do dispositivo que irá receber a replicação. Nesta fase podem ocorrer 2 cenários:

  - Não é encontrado na rede nenhum dispositivo e é apresentada a respetiva mensagem.
  - É encontrado pelo menos um dispositivo na rede e o dispositivo selecionado recebe a atualização dos campos enviados para

replicação. Neste processo não existem validações nenhumas podendo replicar-se inclusivamente dados para dispositivos com endereços diferentes configurados na base de dados e na rede

- "Replicar na BD" – Faz exatamente o mesmo que o ponto anterior, mas desta vez replica os dados para um dispositivo já existente na base de dados. Esta opção encontra-se inativa caso apenas exista um dispositivo criado na base de dados.

4.3.1.3 – Opção: "Todos os dispositivos (Rede e BD)".

Esta é a opção onde se juntam as duas opções descritas nas secções anteriores, e se trabalha o dispositivo como um todo, existente na rede e base de dados.

Numa situação regular de configuração de um dispositivo, todo o processo deve ser gerido nesta opção, sendo as outras duas utilizadas apenas em caso de recurso ou para configuração de um dispositivo "*stand alone*".

Em seguida, ir-se-á desenvolver esta secção tendo em consideração os seguintes pressupostos:

- Existe instalada uma placa de aquisição de dados RenPAD em que apenas se configuraram os seus parâmetros de rede (*IP*, máscara, *gateway*) e a descrição de rede. Este dispositivo terá o endereço *IP* 10.17.0.23 e a descrição de rede "Sala de servidores 1" e não está configurado na base de dados, tendo sido instalado para substituir o dispositivo com o endereço *IP* 10.17.0.130 que entretanto avariou.
- Existe já configurado na base de dados um dispositivo com o endereço *IP* 10.17.0.107 com a descrição "RenPAD Teste", que está também presente na rede mas existem diferenças entre as configurações na rede e na base de dados.
- Existe um dispositivo configurado apenas na base de dados, com o endereço *IP* 10.17.0.130 e descrição "Inexistente" e que não existe na rede (avariou e foi substituído pelo dispositivo com endereço *IP* 10.17.0.23).
- Existe ainda um dispositivo, configurado com o endereço *IP* 10.17.0.24 e a descrição de rede "Sala de servidores 2" que está a funcionar no modo "*stand alone*" mas que irá passar a ser monitorizado, sendo para isso criado um novo dispositivo na base de dados com as mesmas configurações deste.

Desta forma conseguem-se descrever todos os processos de configuração possíveis no sistema.

Uma vez que o método de atualização dos dados, os menus apresentados e as opções disponíveis para as operações abordadas nesta secção são iguais ou muito semelhantes aos abordados nas secções anterior, irei apenas descrever mais pormenorizadamente algum aspeto que considero mais importante, apresentando o restante apenas de modo genérico.

Ao executar esta opção aparece uma imagem que adverte para o caso de algumas operações poderem ser bastante demoradas. Efetivamente, operações como a pesquisa de dispositivos na rede por *range* de *IP's* pode demorar alguns minutos.

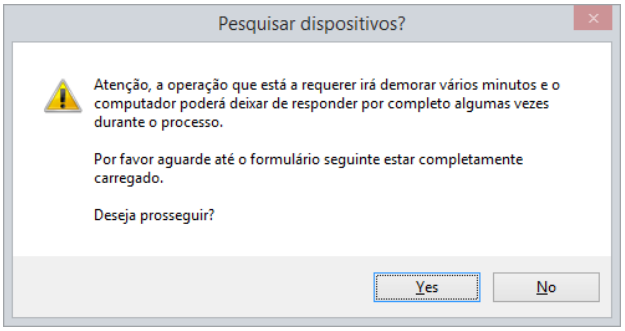


Figura 125: Mensagem de aviso de demora durante a pesquisa de dispositivos.

Seguidamente é apresentada a janela de "Pesquisa de dispositivos na rede" apresentada na Figura 101 e após confirmação de "Pesquisar" é lançada uma tabela "Resultado da pesquisa" contendo todos os dispositivos encontrados tanto na rede *Ethernet* como na base de dados:

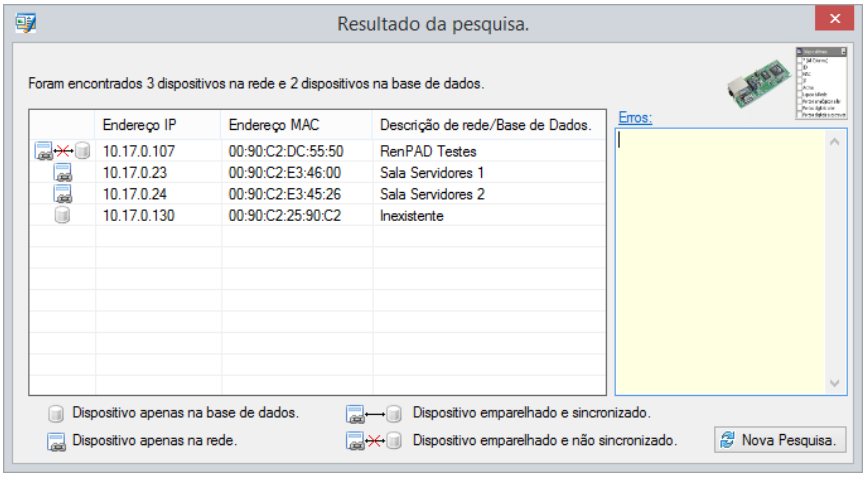


Figura 126: Resultado da pesquisa global de dispositivos (Rede e BD).

Esta tabela é diferente das apresentadas anteriormente e no modo de seleção de cada dispositivo mostra as opções de operação que este pode assumir e que se descrevem pormenorizadamente nos pontos seguintes:

1. Opções disponíveis para o dispositivo com endereço *IP* 10.17.0.130 e descrição Inexistente, presente apenas na base de dados, e cuja placa RenPAD a que estava associado (com o mesmo endereço *IP*) avariou, tendo sido entretanto substituída pela placa RenPAD com o endereço *IP* 10.17.0.24, que é mostrado na figura seguinte como estando presente apenas da rede *Ethernet*:

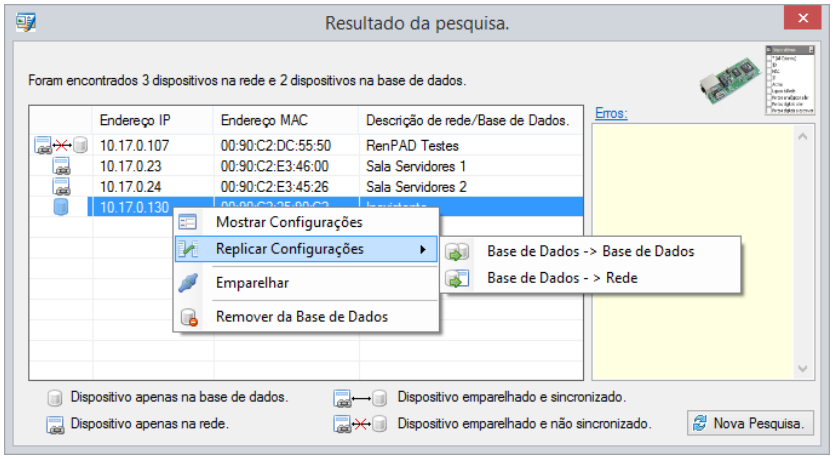


Figura 127: Opções de configuração disponíveis para um dispositivo encontrado unicamente na base de dados.

- "Mostrar configurações" – Mostra as configurações deste dispositivo numa janela "Dispositivo na BD" como descrito na secção anterior, com a única diferença de que os botões de replicação não estão visíveis.
- "Replicar Configurações / Base de Dados->Base de Dados" - Executa o mesmo procedimento que o botão "Replicar na BD" da janela "Dispositivo na BD" descrito na secção anterior.
- "Replicar Configurações / Base de Dados->Rede" - Executa o mesmo procedimento que o botão "Replicar para Rede" da janela "Dispositivo na BD" descrito na secção anterior.
- "Emparelhar" – permite "emparelhar" este dispositivo configurado apenas na base de dados com qualquer um outro presente apenas na rede. O emparelhamento serve para realizar a troca de dispositivos que se avariaram ou foram substituídos, como é o caso do dispositivo com endereço *IP*

10.17.0.130 e o dispositivo com endereço *IP* 10.17.0.23. Com este emparelhamento, o dispositivo na rede seria configurado com o endereço *IP* e a porta de comunicações presente na base de dados (neste caso *IP* = 10.17.0.130 e porta=20000) e seria atualizado o campo do endereço *MAC* do dispositivo na base de dados com o endereço *MAC* real do dispositivo na rede. Desta forma, as leituras associadas ao dispositivo inicial que foi substituído, são automaticamente associadas ao novo dispositivo sem existir perca de indexação. Não foi no entanto este o modo de emparelhamento selecionado para esta demonstração, tendo-se optado antes pelo modo de emparelhamento descrito no ponto seguinte designado de ponto 2.

Este processo de "emparelhamento" lança uma janela de "Resultado da pesquisa" contendo os dispositivos presentes na rede passíveis serem emparelhados com este (sem estarem já emparelhados com outros dispositivos),

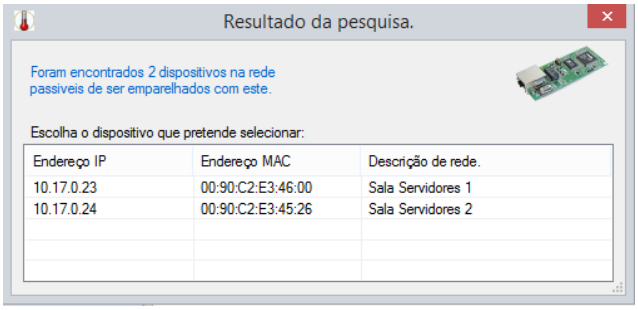


Figura 128: Janela "Resultado da pesquisa" para emparelhamento com dispositivo presente na rede.

e depois de selecionado o dispositivo pretendido para o emparelhamento é mostrada a mensagem de aviso a seguir apresentada e que depois de confirmado o pedido realiza as operações de emparelhamento.

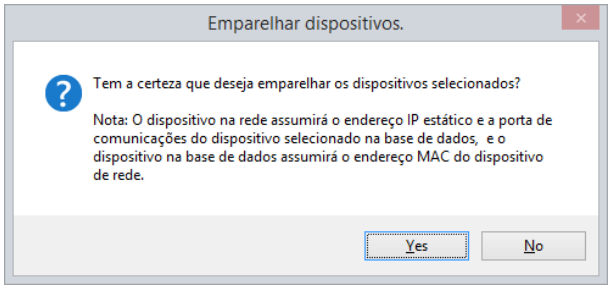


Figura 129: Mensagem de aviso durante o emparelhamento de um dispositivo presente na base de dados com um dispositivo presente na rede.

- "Remover da base de dados" – Executa o mesmo procedimento que o botão "Remover da BD" da janela "Dispositivo na BD" descrito na secção anterior.

2. Opções disponíveis para o dispositivo com endereço *IP* 10.17.0.23 e descrição "Sala Servidores 1", presente apenas na rede *Ethernet* e que foi instalado para substituir a placa RenPAD com endereço *IP* 10.17.0.130 que estava associada ao dispositivo na base de dados com o mesmo endereço *IP*, e que é mostrado na figura seguinte como estando presente apenas na base de dados. Estas opções são iguais às do dispositivo com o endereço *IP* 10.17.0.24 que está em funcionamento em modo "*stand alone*" mas que vai ser colocado em monitorização, sendo para isso criado na base de dados. Como os processos referentes aos dois dispositivos são complementares, vão ser abordados em conjunto e as opções que são válidas para um também serão válidas para outro:

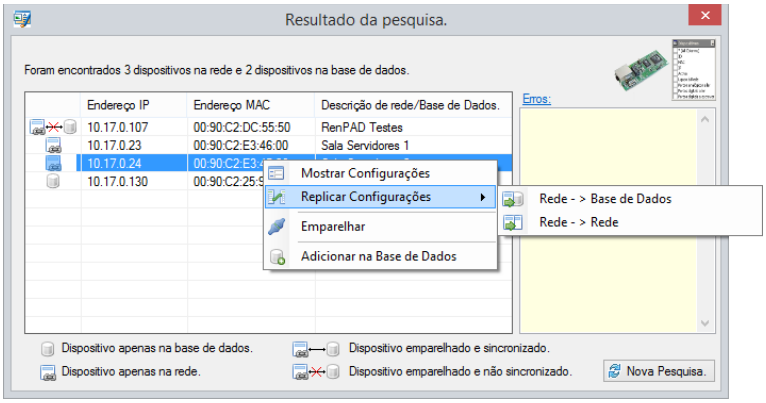


Figura 130: Opções de configuração disponíveis para um dispositivo encontrado unicamente na rede *Ethernet*.

- "Mostrar configurações" – Mostra as configurações deste dispositivo numa janela "Dispositivo de Rede" como descrito na secção 4.3.1.1, com a única diferença de que os botões de replicação não estão visíveis.
- "Replicar Configurações / Rede->Base de Dados" - Executa o mesmo procedimento que o botão "Operações na BD" da janela "Dispositivo de Rede" descrito na secção anterior, com a diferença que não existe qualquer controlo ou validação de endereço *IP* dos dispositivos para onde os parâmetros irão ser replicados, podendo os parâmetros serem replicados

para qualquer dispositivo configurado na base de dados, e não é dada a opção de criar novo dispositivo na base de dados uma vez que esta opção se encontra disponível diretamente.

- "Replicar Configurações / Rede->Rede" - Executa o mesmo procedimento que o botão "Replicar na Rede" da janela "Dispositivo de Rede" descrito na secção 4.3.1.1.
- "Emparelhar" – Faz o mesmo que a opção "Emparelhar" descrita no ponto 1, mas neste "emparelhamento" apenas os campos da base de dados são afetados, não existindo alteração de qualquer parâmetro do dispositivo presente na rede. São atualizados unicamente o endereço *IP*, a porta de comunicações (neste caso *IP* = 10.17.0.23 e porta=20000) e o campo do endereço *MAC* na base de dados, com os valores configurados no dispositivo de rede, todos os outros campos de configurações ficam no seu estado original não ocorrendo a sua sincronização. Este foi o método de emparelhamento selecionado para esta demonstração e podemos observar na Figura 133 que os dados do dispositivo com endereço *IP* 10.17.0.23 presente apenas na rede foram replicados para o dispositivo anteriormente configurado com o endereço *IP* 10.17.0.130. O emparelhamento afeta apenas os campos indicados no parágrafo anterior, e caso se pretenda, após a realização do emparelhamento podem-se replicar todos os campos de configurações utilizando qualquer uma das opções a isso referentes (sincronização, replicação de dados…).

  O processo de "emparelhamento" lança a seguinte janela de "Resultado da pesquisa", contendo os dispositivos presentes na base de dados passíveis de serem emparelhados com este (sem estarem já emparelhados com outros):

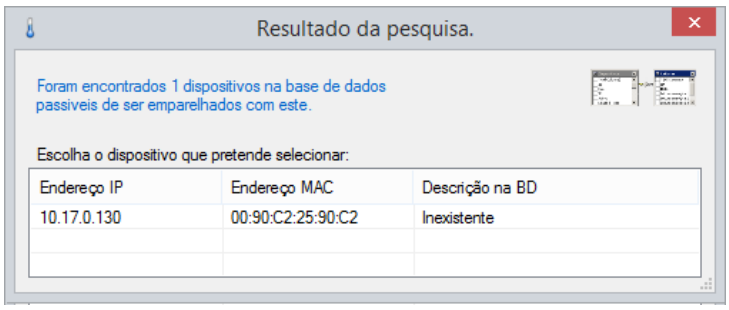


Figura 131: Janela "Resultado da pesquisa" para emparelhamento com dispositivo presente na base de dados.

Depois de selecionado o dispositivo pretendido para emparelhar é mostrada a mensagem de aviso, e após a sua confirmação realiza as operações de emparelhamento.

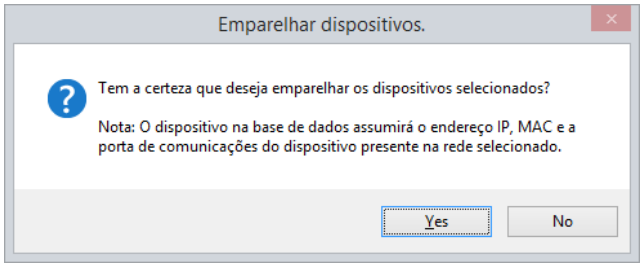


Figura 132: Mensagem de aviso durante o emparelhamento de um dispositivo presente na rede com um dispositivo presente na base de dados.

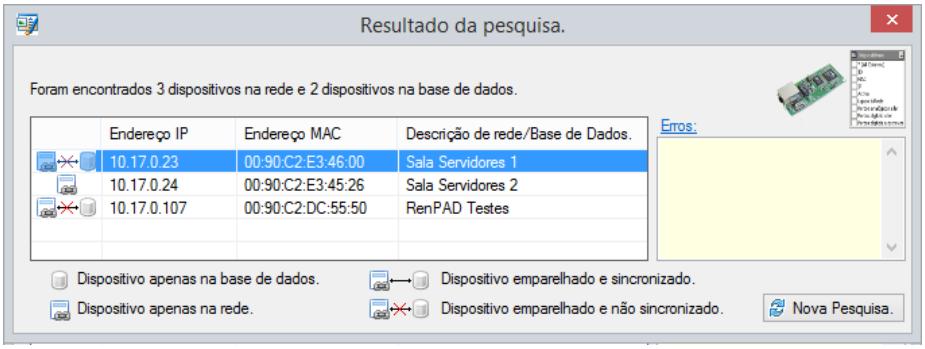


Figura 133: Janela "Resultado da pesquisa" de todos os dispositivos encontrados numa pesquisa global, após emparelhamento de um dispositivo de rede com um dispositivo de base de dados.

- "Adicionar na Base de Dados" – Adiciona este dispositivo de rede na Base de dados. Todos os campos de configurações do dispositivo criados na base de dados assumem os dados configurados na placa RenPAD ficando os dispositivos sincronizados, como podemos observar na Figura 135.

  No despoletar deste processo é lançada a seguinte janela de aviso:

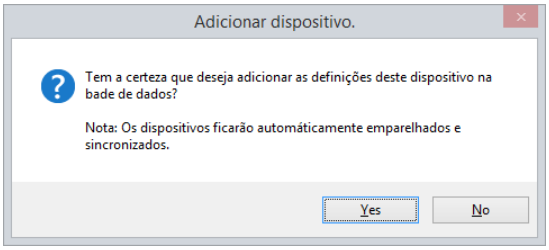


Figura 134:Mensagem de aviso durante a adição de um dispositivo presente na rede, na base de dados.

Depois de adicionado na base de dados, o dispositivo aparece da seguinte forma na tabela "Resultado da pesquisa":

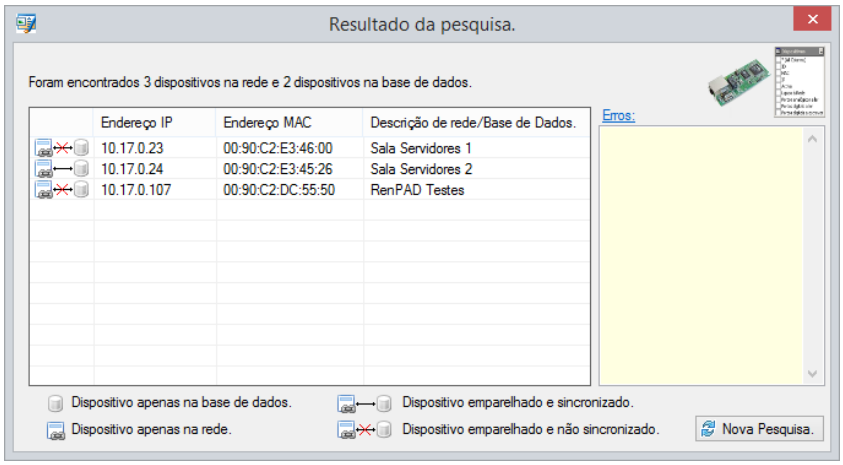
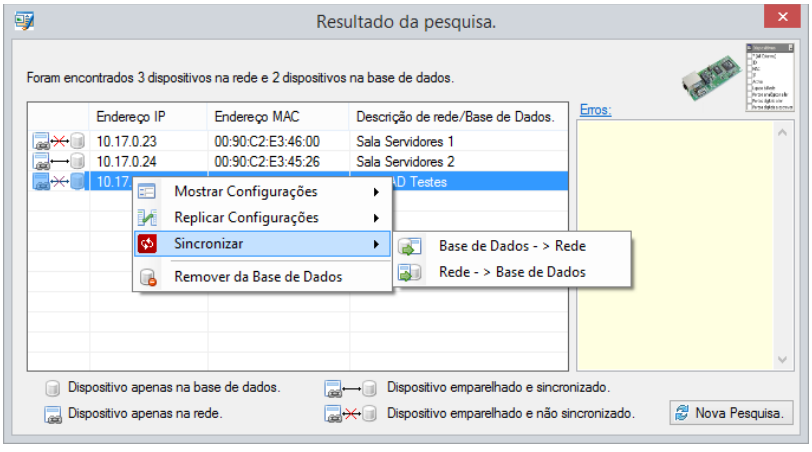


Figura 135: Resultado da pesquisa global de dispositivos (Rede e BD) após a adição de um dispositivo presente na rede, na base de dados.

3. Opções disponíveis para o dispositivo com endereço *IP* 10.17.0.107 e descrição "RenPad Testes", presente na rede *Ethernet* e na base de dados mas cujas configurações estão dessincronizadas ou seja, existem diferentes configurações para campos idênticos nos dois dispositivos.



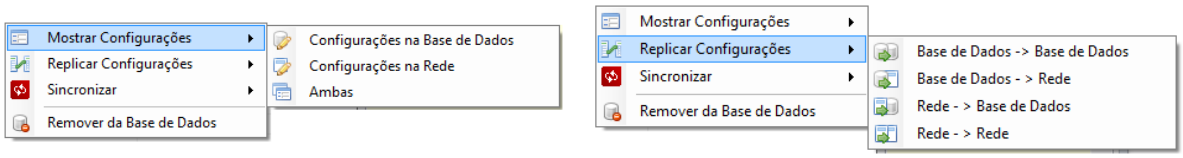


Figura 136: Opções de configuração disponíveis para um dispositivo encontrado na rede *Ethernet* e na base de dados mas cujas configurações não estão sincronizadas.

- Neste caso, todas as operações apresentadas, à exceção da opção "sincronizar", já foram apresentadas nos pontos 1 e 2 e os processos de configuração são os mesmos.
- "Sincronizar" – faz o mesmo que as operações de replicação de dados com a diferença de não ser necessário escolher o dispositivo para onde se querem replicar os dados nem quais os dados a replicar. Os dados de configurações são todos sincronizados do dispositivo de rede para o dispositivo presente na base de dados ou *vice-versa* conforme a opção selecionada.

  No despoletar deste processo é lançada a seguinte janela de aviso:

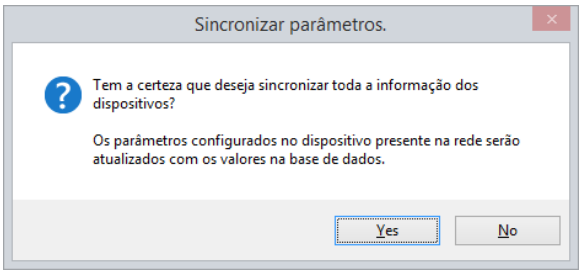


Figura 137: Mensagem de aviso durante a sincronização de dois dispositivos emparelhados.

e após a confirmação, os dispositivos são sincronizados conforme a opção selecionada.

4. Opções disponíveis para o dispositivo com endereço *IP* 10.17.0.107 e descrição "RenPad Testes", referido no ponto anterior, que passou por um processo de sincronização encontrando-se neste momento no estado "Sincronizado", em que todas as configurações do dispositivo, criado na base de dados e existente na rede *Ethernet*, estão alinhadas.

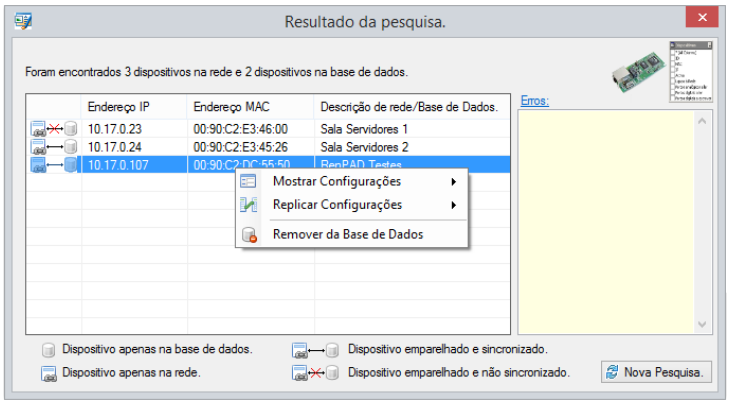


Figura 138: Opções de configuração disponíveis para um dispositivo encontrado na rede *Ethernet* e na base de dados e cujas configurações estão sincronizadas.

Como podemos depreender dos pontos anteriores, e como assinalado ao longo desta secção, este é o modo de configurações mais importante deste separador de configurações. Por este motivo foram apresentadas todas as imagens no corpo do texto e não num anexo, por forma a dar-lhes a devida importância.

Chamo também a atenção que em algumas situações particulares, algumas destas opções estão indisponíveis. Isto acontece quando o contexto da opção não faz sentido em determinada etapa do processo: por exemplo, a opção de "Emparelhar" desaparece (fica invisível) quando não existem na lista apresentada na janela "Resultado da pesquisa", dispositivos que satisfaçam as condições de emparelhamento.

4.3.2 – Separador "Estatísticas"

Este é o separador selecionado por defeito logo que se executa o programa.

Neste separador são apresentadas de dez em dez linhas, todos os estados de cada leitura, com identificação visual das ocorrências anormais, filtradas por dispositivo ("Terminal"), data, tipo de ocorrência entre outros.

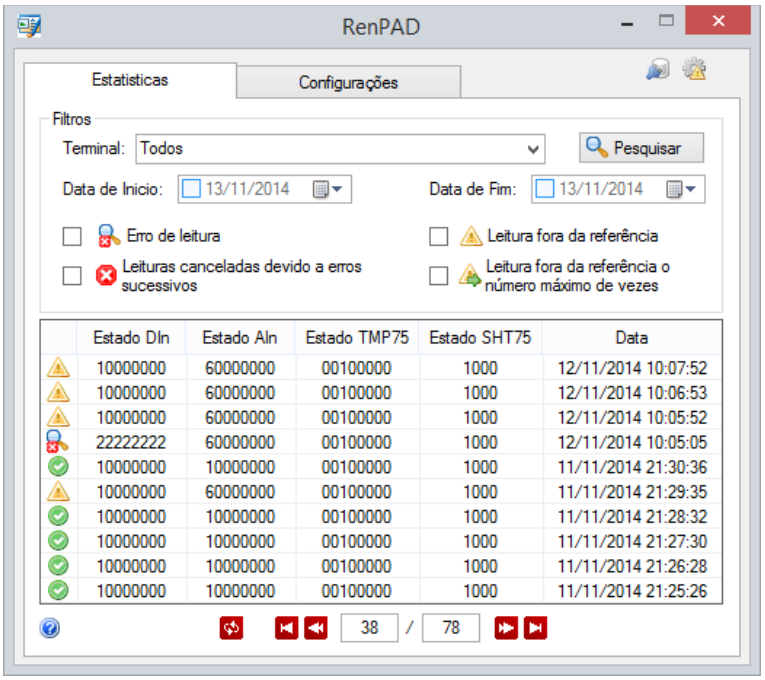


Figura 139: Separador Estatísticas.

No menu de ajuda (ícone com ponto de interrogação no canto inferior esquerdo) podem consultar-se os vários estados possíveis que cada entrada pode assumir:

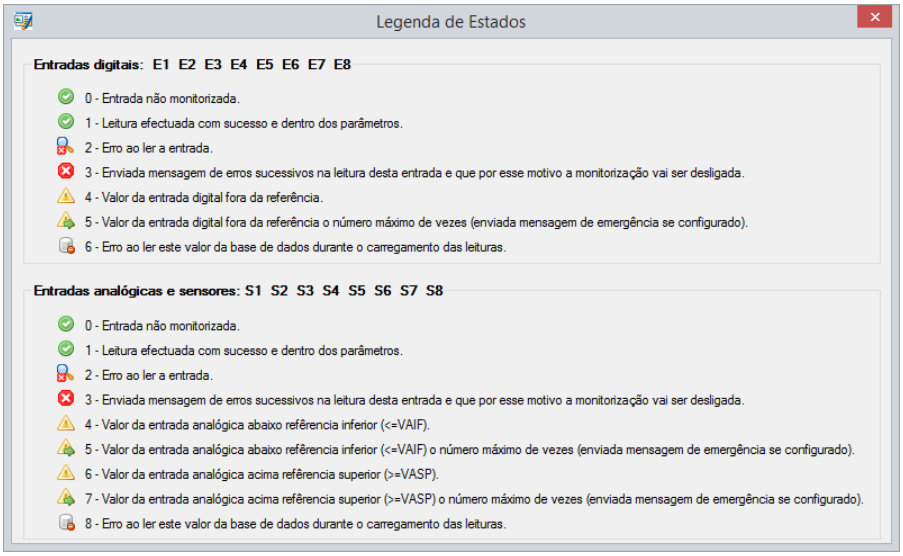


Figura 140: Estados que cada entrada pode assumir.

Na secção de filtros existe uma gama de filtragem que permite pesquisar por eventos de forma simples e rápida. Estes filtros são cumulativos podendo ser escolhidos diversos filtros em simultâneo por forma a refinar a pesquisa o mais possível:

- "Terminal" – permite que possamos obter leituras apenas de um determinado dispositivo sobre o qual existam leituras na base de dados. O nome "Terminal" deriva de internamente no departamento, as placas de aquisição de dados RenPAD serem usualmente denominadas de terminais RenPAD, dispositivos RenPAD, equipamentos RenPAD ou placas RenPAD. É também por este motivo que no decorrer desta tese aparecem estas referências distintas para esta mesma denominação, sem que tenha feito um esforço de uniformização nesta nomenclatura em particular para evidenciar este fato.

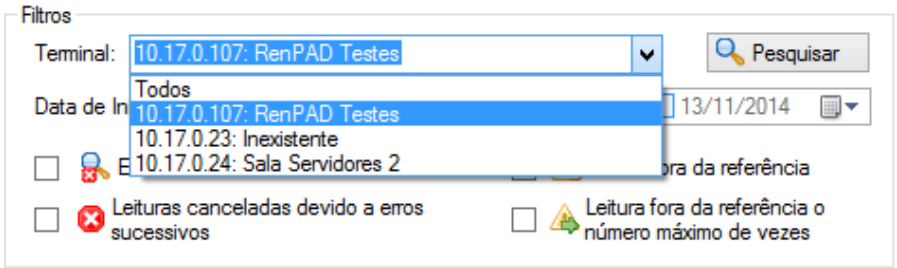


Figura 141: Filtragem de leituras por terminal.

- "Data de Inicio" e "Data de Fim" – permite filtrar a pesquisa à base de dados por leituras recolhidas numa dada gama de datas configuradas:

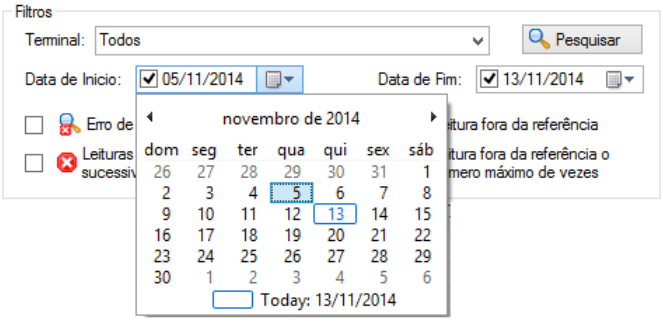


Figura 142: Filtragem de leituras por datas.

- Filtragem por tipo de erro – permite filtrar o resultado selecionando apenas leituras em que pelo menos uma das entradas apresente os erros selecionados e que são descritos no menu de ajuda apresentado na Figura 140:

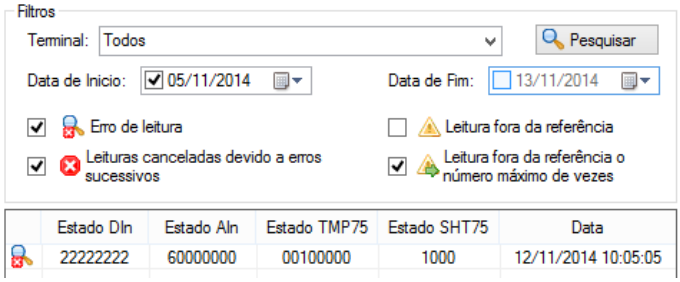


Figura 143: Filtragem de leituras por tipo de erro.

- "Pesquisar" – geralmente o ato de alterar o valor de qualquer filtro é suficiente para despoletar o evento de recarregamento de dados da base de dados. No entanto, poderemos desejar despoletar uma nova pesquisa para obtenção de eventuais leituras mais recentes que tenham entretanto sido inseridas na base de dados pelo serviço de recolha de dados sem alterar as definições de filtragem. Esse pode ser despoletado executando este botão "Pesquisar".

Este separador apresenta ainda uma barra de navegação que permite mover a seleção de leituras obtidas da base de dados, em grupos de dez leituras por página. Esta barra permite ainda executar o recarregamento de dados de forma automática de dez em dez segundos. Esta opção despoleta o mesmo evento que o botão "Pesquisar" mas de forma automática a cada dez segundos, e é útil para monitorizar "em tempo real" o estado das entradas selecionadas para visualização através dos filtros escolhidos.

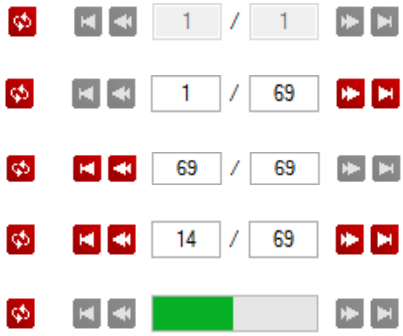


Figura 144: Os vários estados possíveis da barra de navegação.

Após preenchidos todos os filtros pretendidos, e de termos encontrados as leituras que necessitamos analisar, basta selecionar a leitura pretendida, e aparece a janela com todos os seus dados:

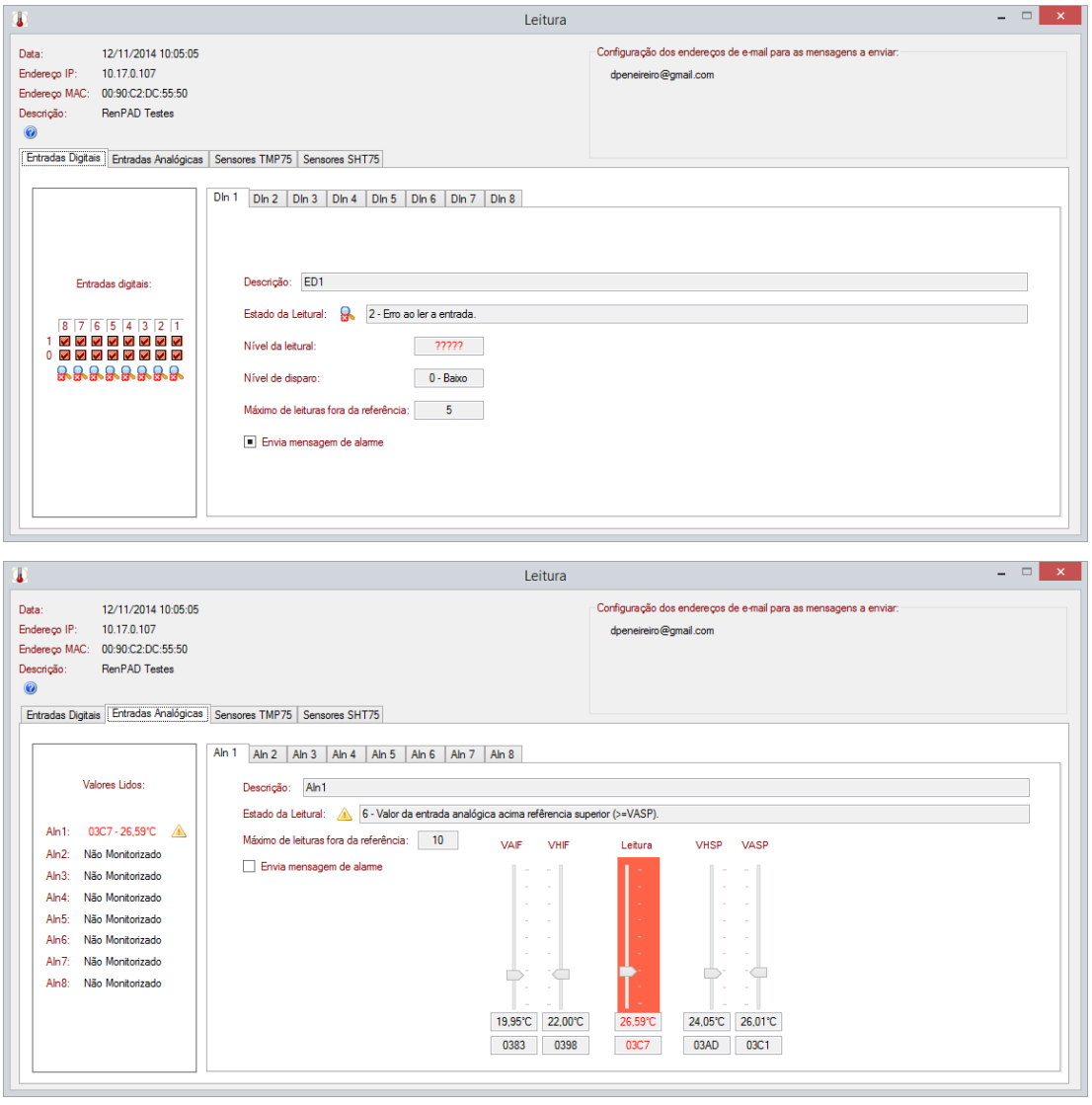


Figura 145: Vários painéis demonstrativos dos dados obtidos para cada leitura.

Conforme se pode observar na sequência de painéis representados na imagem anterior, a análise dos dados presentes em cada entrada numa dada leitura e o estudo de eventos anómalos, é facilitado através do uso de informações gráficas e descrições pormenorizadas, facilitando a sua compreensão.

Com o processo implementado, teremos apenas que ter em atenção o seguinte:

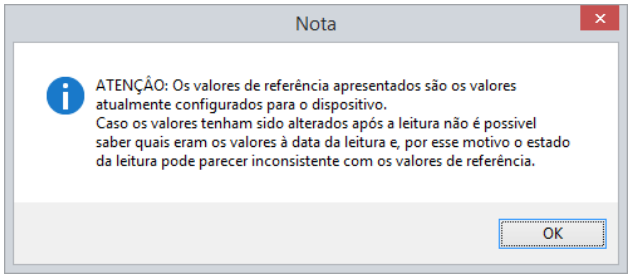


Figura 146: Mensagem de aviso sobre alteração dos parâmetros de alarme.

Irei ilustrar com um exemplo para ser mais fácil a perceção deste fenómeno: imagine-se que numa dada altura foi recolhida uma leitura onde a entrada analógica AIn1 tinha o valor de 30ºC e que o seu limite superior de alarme VASP estava configurado em 28ºC. Durante este processo de recolha, o estado desta entrada foi registado na respetiva *flag* de *status* com o valor 6, indicativo de valor de entrada analógica acima do valor de referência VASP. Entretanto, uns dias depois, o responsável pelo sistema, depois de estudar melhor o ambiente em que este se encontra, decide alterar as especificações e determinar que afinal o valor de VASP deverá ser, não de 28ºC mas sim de 32ºC. Após a alteração do valor de referência, a leitura continua a apresentar na *flag* de *satus* referente à entrada AIn 1 o valor 6. Nesta situação, quando se abre de novo as propriedades da leitura, irá ocorrer uma incongruência no separador referente a AIn 1, onde será indicado que ocorreu uma sinalização porque o valor lido é superior ao valor máximo VASP, mas depois nos gráficos vai aparecer um valor lido de 30ºC inferior ao VASP configurado de 32ºC.

Contornar esta *feature* não seria difícil, bastando guardar todas as configurações definidas (VASP, VHSP, …) juntamente com os dados da leitura, ou então fazer um histórico das alterações a estas configurações de maneira a deixar "um rasto" para se poderem fazer as devidas correspondências e obter sempre os valores consistentes. Optou-se no entanto por não implementar nenhuma destas soluções porque por norma, ocorrências deste tipo serão pontuais e o ganho que estas alterações trariam não compensam o gasto em complexidade de processo, o tamanho ocupado na base de dados, *etc*.

No entanto, e caso se pretenda mesmo que estas alterações sejam sinalizadas, poderemos sempre criar um dispositivo novo na base de dados com as novas definições e a monitorização passa a ser feita sobre esse novo dispositivo. Assim, as leituras antigas ficam associadas a um dispositivo que já não existe fisicamente na rede, mas podem ser consultadas pois, o dispositivo ainda está configurado na base de dados, e as leituras após a alteração passam a ser registadas sob um novo dispositivo não interferindo nas antigas. Este método, apesar de funcional, não é elegante e nunca foi implementado.

4.4 – Testes e otimização.

O Monitorizador RenPAD foi sendo testado paralelamente ao seu próprio crescimento. À medida que algum novo desenvolvimento o permitiu, entrou de imediato em testes exaustivos. Todas as possibilidades de configuração de alarmes foram sendo experimentadas à medida que eram concebidas, em testes rigorosos e atentos por forma a conseguir um desempenho fiável em condições de testes reais.

O Monitorizador RenPAD sofreu também muitas evoluções no seu código base ao longo do tempo.

Fruto de uma evolução constante, muitos dos métodos originais foram sendo melhorados, sempre que se assegurou que as alterações propostas seriam benéficas e iriam trazer mais-valias ao processo.

O exemplo mais evidente de uma destas melhorias trata-se da opção de configuração de dispositivos “Todos os dispositivos (Rede e BD)”, descrita na secção 4.3.1.3 que, não sendo parte integrante do conceito original, pela sua conceção veio facilitar grandemente a forma de configurar o sistema, a sua aprendizagem e compreensão, interligando numa única opção as duas já existentes.

5 – Conclusões e possibilidade de desenvolvimentos futuros

5.1 - Conclusões

Durante toda a conceção do projeto foram sendo atualizados os requisitos e especificações, mediante a experiência adquirida e a reanálise das suas potencialidades globais. De um dispositivo que inicialmente se pretendia que "lesse" apenas uns valores de temperatura, passou-se para uma solução completa, altamente evoluída e facilmente escalável, que além da premissa inicial contempla uma maior e mais considerável panóplia de características, que podem ainda crescer conforme as solicitações que nesse sentido ocorram.

A solução está já implementada e em funcionamento há vários meses nas duas salas de servidores, mesmo antes da solução global estar completa, mais propriamente desde que o serviço de recolha de dados passou sem falhas nos testes laboratoriais.

Em termos operacionais, esta solução mostrou as mais-valias que lhe estão inerentes aquando da instalação de um novo sistema de ar condicionado num dos *OpenSpace* da Renova, em Agosto deste ano. Inicialmente, e por defeito de conceção, o sistema montado não trabalhava corretamente fazendo circular ar frio enquanto as suas tubagens internas não aquecessem, o que normalmente demorava muito tempo, arrefecendo ainda mais o ambiente que se pretendia aquecer. Além disso, devido ao fato dos sensores de temperatura estarem montados no interior dos difusores, que geralmente estão a uma temperatura superior à do ambiente que pretendem aquecer, estes forneciam uma temperatura da sala "viciada" ao sistema central, que desligava a geração de calor com base nessa leitura. Desta forma, a temperatura da sala nunca atingia a temperatura de conforto predefinida, e para que isso acontecesse teria que se programar a temperatura desejada para um valor muito superior ao real. Estes fenómenos podem parecer "contornáveis", mas para um sistema desta envergadura e com os custos inerentes, são inconcebíveis.

O "clima" deste *OpenSpace* foi monitorizado por meio de dois sistemas RenPAD e a análise dos dados, permitiu que se compreendessem e provassem perante o instalador os fenómenos associados a este funcionamento deficiente, tendo dado origem a alterações tanto de programação do modo de funcionamento do climatizador, como à instalação de

mais uma quantidade de sensores de temperatura em vários pontos distintos, no *OpenSpace*, que permitem ao sistema obter as temperaturas reais das várias secções.

Entretanto está também projetada a instalação de vários sistemas RenPAD para monitorizar o rigoroso processo de recolha de ar para arrefecimento de algumas máquinas que interferem nos diversos processos de fabrico. O ar é recolhido do exterior e depois de passar por filtros de limpeza é seco num secador, tendo que chegar aos sistemas a refrigerar com as características de climatização certas. Nesta configuração os dispositivos RenPAD irão controlar a entrada de ar, mediante a aquisição de temperatura e humidade geral do sistema em vários pontos, através da configuração dos seus alarmes cujas saídas darão ordens, tanto ao sistema de extração e ventilação, como ao sistema de secagem. Caso este projeto avance, os dispositivos RenPAD irão controlar sistemas críticos do processo de fabrico, demonstrando a confiança neles depositada.

Este é um produto com o selo Renova Eletrónica.

Figura 147: Placa de aquisição de dados RenPAD e sensor SHT75 instalados na sala de servidores 1.

5.2 – Possibilidade de desenvolvimentos futuros

Em termos de otimização, existem certamente muitos aspetos que podem ser melhorados, desde a elaboração de um novo esquemático e alterações ao programa de monitorização que contemplem as alterações propostas nas secções 5.2.1 e 5.2.2, até às infindáveis otimizações de *software* que podem ser realizadas no desenvolvimento de qualquer programa e que nunca se esgotam.

Considero no entanto, e pela experiência que tenho, que para uma primeira versão, este projeto se apresenta já muito próximo da versão final, podendo inclusivamente ser comercializado "*as is*", sem temer eventuais reclamações por parte dos clientes compradores.

5.2.1 – *Hardware* – Dispositivo RenPAD

Este projeto pode, em termos de *hardware*, ser melhorado futuramente em duas vertentes:

- Normalização do *hardware* para certificação CE
- Extensão de funcionalidades para guardar os valores presentes em cada sensor, lidos ao longo do tempo (*datalogger*)

Para o primeiro caso, e dada a experiência adquirida no processo de normalização de outros produtos Renova existem algumas alterações de raiz a ter em conta, tais como:

- Inserção de um filtro de entrada que atenue as emissões eletromagnéticas para a rede (propagação eletromagnética) como por exemplo o *SHAFFNER* FN406-3.

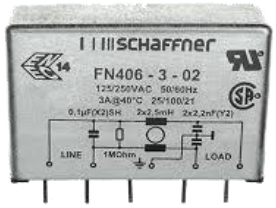


Figura 148: SHAFFNER FN406-3

- Criação de dois *layer's* internos na placa de circuito impresso ocupando toda a área possível (exceto as furações necessárias), um acoplado à massa e outro a 5V, criando assim um condensador com a função de filtrar ao máximo a radiação emitida pela placa.
- Inserção de filtros RC em todas as saídas passíveis de serem conectadas a cabos relativamente extensos, que funcionarão como antenas emissoras de radiação eletromagnética associada aos *clock's* internos do processador, relógio de tempo real *etc*.

No segundo caso basta criar uma lista de valores associados a uma nova estrutura de dados, que acomode as leituras de todos os sensores, a data de aquisição dessa leitura, apontador para a próxima leitura e eventualmente apontador para a leitura anterior, e passar a armazenar os dados recolhidos nessa estrutura de dados de forma sequencial. As livrarias "mem_gest.lib" e "Tab_Var.lib" já contêm no seu corpo algumas funcionalidades orientadas para este desenvolvimento.

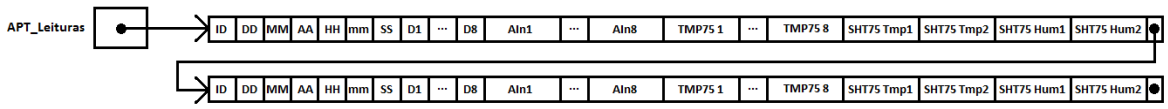
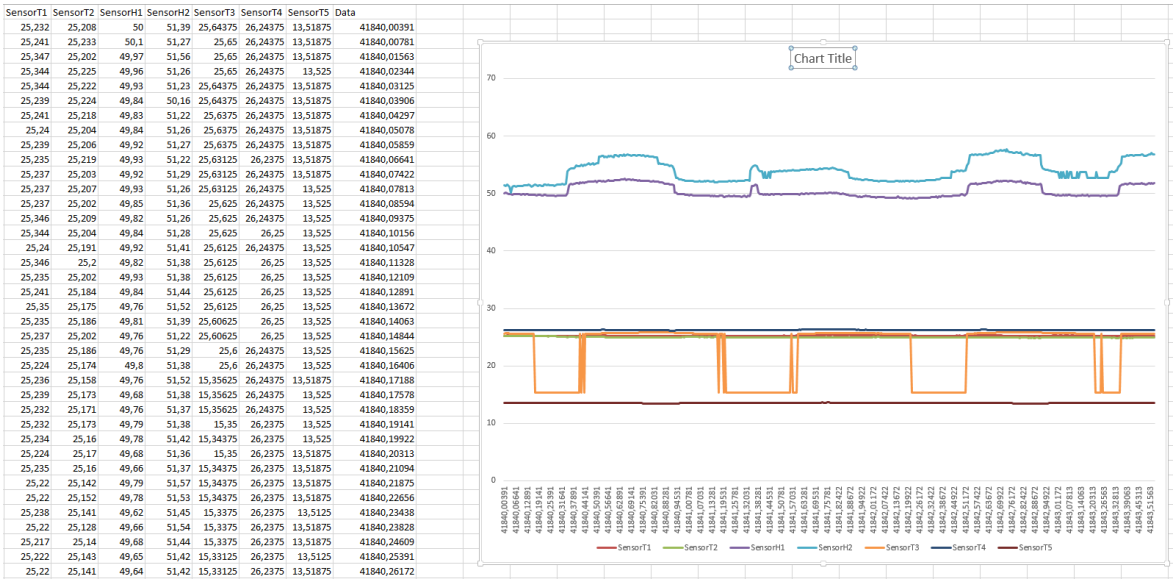


Figura 149: Exemplo de lista de armazenamento dos dados a monitorizar para implementação de um *datalogger*.

5.2.2 – *Software* – Monitorizador RenPAD

Não estão para já previstos nenhuns desenvolvimentos que possam ocorrer num futuro próximo. Contudo este não é um produto fechado e portanto, em virtude de alguma exigência ou necessidade urgente, poderá em qualquer altura sofrer algum tipo de reajuste às suas especificações atuais.

Subsiste, no entanto, a intenção de implementar um módulo gráfico complementar à solução original. Durante o desenrolar deste projeto foram realizados alguns testes pelo meu orientador de projeto na Renova, onde se abordou um exemplo de solução gráfica que poderá servir de base para a implementação deste futuro módulo:



SensorT1	SensorT2	SensorH1	SensorH2	SensorT3	SensorT4	SensorT5	Data
25,232	25,208	50	51,39	25,64375	26,24375	13,51875	41840,00391
25,241	25,233	50,1	51,27	25,65	26,24375	13,51875	41840,00781
25,347	25,202	49,97	51,56	25,65	26,24375	13,51875	41840,01563
25,344	25,225	49,96	51,26	25,65	26,24375	13,525	41840,02344
25,344	25,222	49,93	51,23	25,64375	26,24375	13,51875	41840,03125
25,239	25,224	49,84	50,16	25,64375	26,24375	13,51875	41840,03906
25,241	25,218	49,83	51,22	25,6375	26,24375	13,51875	41840,04297
25,24	25,204	49,84	51,26	25,6375	26,24375	13,51875	41840,05078
25,239	25,206	49,92	51,27	25,6375	26,24375	13,51875	41840,05859
25,235	25,219	49,93	51,22	25,63125	26,2375	13,51875	41840,06641
25,237	25,203	49,92	51,29	25,63125	26,24375	13,51875	41840,07422
25,237	25,207	49,93	51,26	25,63125	26,24375	13,525	41840,07813
25,237	25,202	49,85	51,36	25,625	26,24375	13,525	41840,08594
25,346	25,209	49,82	51,26	25,625	26,24375	13,525	41840,09375
25,344	25,204	49,84	51,28	25,625	26,25	13,525	41840,10156
25,24	25,191	49,92	51,41	25,6125	26,24375	13,525	41840,10547
25,346	25,2	49,82	51,38	25,6125	26,25	13,525	41840,11328
25,235	25,202	49,93	51,38	25,6125	26,25	13,525	41840,12109
25,241	25,184	49,84	51,44	25,6125	26,25	13,525	41840,12891
25,35	25,175	49,76	51,52	25,6125	26,25	13,525	41840,13672
25,235	25,186	49,81	51,39	25,60625	26,25	13,525	41840,14063
25,237	25,202	49,76	51,22	25,60625	26,25	13,525	41840,14844
25,235	25,186	49,76	51,29	25,6	26,24375	13,525	41840,15625
25,224	25,174	49,8	51,38	25,6	26,24375	13,525	41840,16406
25,236	25,158	49,76	51,52	15,35625	26,24375	13,51875	41840,17188
25,239	25,173	49,68	51,38	15,35625	26,24375	13,525	41840,17578
25,232	25,171	49,76	51,37	15,35625	26,24375	13,525	41840,18359
25,232	25,173	49,79	51,38	15,35	26,2375	13,525	41840,19141
25,234	25,16	49,78	51,42	15,34375	26,2375	13,525	41840,19922
25,224	25,17	49,68	51,36	15,35	26,2375	13,51875	41840,20313
25,235	25,16	49,66	51,37	15,34375	26,2375	13,51875	41840,21094
25,22	25,142	49,79	51,57	15,34375	26,2375	13,51875	41840,21875
25,22	25,152	49,78	51,53	15,34375	26,2375	13,51875	41840,22656
25,238	25,141	49,62	51,45	15,3375	26,2375	13,5125	41840,23438
25,22	25,128	49,66	51,54	15,3375	26,2375	13,51875	41840,23828
25,217	25,14	49,68	51,44	15,3375	26,2375	13,51875	41840,24609
25,222	25,143	49,65	51,42	15,33125	26,2375	13,5125	41840,25391
25,22	25,141	49,64	51,42	15,33125	26,2375	13,51875	41840,26172

Figura 150: Solução de visualização gráfica das variações dos valores presentes em cada entrada ao longo do tempo.

Uma outra vertente de expansão do projeto, que não está para já encerrada, é a possível futura implementação de outros tipos de sensores, alargando a abrangência do produto final. Esta inclusão de novos sensores será realizada quando ocorra essa real necessidade.

Referências Bibliográficas

[1] – Declaração Ambiental 2013 – Renova S.A.

[2] – Wikipédia - http://en.wikipedia.org/wiki/Renova_(brand)

[3] – FaceBook Renova - https://www.facebook.com/RenovaPT

[4] – Análise de funções de segurança num processo industrial e num posto de transformação de energia eléctrica, numa industria papeleira, aplicando a metodologia SFA - Jacinto, Celeste*;* Carracinha, Filipe José Martins - http://hdl.handle.net/10362/2558

[5] – http://renovaquuspolis.blogspot.pt/2008_05_01_archive.html

[6] – http://ginestal8a.blogspot.pt/2008_06_01_archive.html

[7] – http://profdanielamex.blogspot.pt/2010/09/maquina-de-papel_03.html

[8] – http://www.magnificat-azores.com/

[9] – http://watersonglass.blogspot.pt/2010/06/magnificat.html

[10] – http://renova.soup.io/tag/Main?newer=1&since=78142148

[11] – http://www.rtp.pt/acores/index.php?article=18340&visual=3&tm=5&layout=10

[12] – http://www.myrenova.com/p/185/first-silk-sensation-normal-com-abas

[13] – http://www.renovaelectronica.com/

[14] – http://www.apc.com/solutions/display.cfm?id=9B0D4608-3B9B-4C4B-BB198BE92BB28D96&ISOCountryCode=BR

[15] – http://www.digi.com/products/wireless-wired-embedded-solutions/solutions-on-module/rabbitcore/rcm3700#specs

[16] – RabbitCore RCM3700 – C-Programmable Core Module with Ethernet, Serial Flash, and Enhanced Software – User's Manual - Part Number 019-0136_L – ©2003–2010 Digi International Inc. (http://www.acierta-it.com/jas/v3.0/CD%20CON%20TODO/DCRABBIT_9.21/Docs/manuals/RCM3700/rcm3700_.htm)

[17] – https://sites.google.com/site/so20102ita/pesquisa-s-o/microc-os-ii---igor

[18] – Datasheet MCP3208 – http://ww1.microchip.com/downloads/en/DeviceDoc/21298e.pdf

[19] – Datasheet LM60 – http://www.ti.com.cn/cn/lit/ds/symlink/lm60.pdf

[20] – Datasheet TMP75 – http://pdf.datasheetcatalog.com/datasheet2/f/0csfwcfzegxa3qkpeacelapf2h3y.pdf

[21] – datasheet SHT75 – http://www.sensirion.com/fileadmin/user_upload/customers/sensirion/Dokumente/Humidity/Sensirion_Humidity_SHT7x_Datasheet_V5.pdf

[22] – datasheet cicon CFM1003S – http://www.mouser.com/ds/2/75/CFM10_15-10343.pdf

[23] – datasheet SI8050S – http://www.bel.4567.cz/images/stories/virtuemart/product/si-8000s.pdf

[24] – datasheet DS1231 – http://datasheets.maximintegrated.com/en/ds/DS1231.pdf

[25] – datasheet MCP1525 – http://ww1.microchip.com/downloads/en/DeviceDoc/21653C.pdf

[26] – datasheet SI9435BDY – http://www.vishay.com/docs/70126/70126.pdf

[27] – http://pt.wikipedia.org/wiki/C_Sharp

[28] – http://msdn.microsoft.com/pt-BR/library/d56de412(v=vs.110).aspx

Anexo 1 – Função criada para implementar o mecanismo de comunicação para a leitura de uma entrada analógica ligada ao ADC MCP3208

Seguidamente é apresentada a função criada para implementar o mecanismo de comunicação para a leitura de uma entrada analógica ao conversor analógico digital MAP3208. Todas as etapas do processo estão identificadas pelos comentários (texto de cor verde).

```
/* START FUNCTION DESCRIPTION ********************************************
analogIn                    <Auxiliar.LIB>

Sintaxe:           int analogIn(myBYTE entrada);

Descrição:         Lê o valor de uma entrada analógica.

Parâmetro 1:       Entrada a ler - 0..7.

Valor de retorno:  Valor lido.

Ver também:

END DESCRIPTION **************************************************************/
int analogIn(myBYTE entrada)
{
  int leitura, leitura_aux;

  leitura=0;
  //Din no valor alto
  BitWrPortI(PBDR,&PBDRShadow,1,5);
  //cs no valor baixo
  BitWrPortI(PBDR,&PBDRShadow,0,3);

  //clk start bit com din=1
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  delay10uS(1);  //10us em cima
  //clk sgl/dif bit com din=1-single
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  delay10uS(1);  //10us em cima

  //clk d2 bit
  if(entrada==0||entrada==1||entrada==2||entrada==3)
  {
```

```
    BitWrPortI(PBDR,&PBDRShadow,0,5); //clk d2 bit com din=0
  }
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  delay10uS(1);  //10us em cima

  //clk d1 bit
  if(entrada==0||entrada==1||entrada==4||entrada==5)
  {
     BitWrPortI(PBDR,&PBDRShadow,0,5); //clk d1 bit com din=0
  }
  else
  {
     BitWrPortI(PBDR,&PBDRShadow,1,5); //clk d1 bit com din=1
  }

  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  delay10uS(1);  //10us em cima

  //clk d0 bit
  if(entrada==0||entrada==2||entrada==4||entrada==6)
  {
        BitWrPortI(PBDR,&PBDRShadow,0,5); //clk d0 bit com din=0
  }
  else
  {
    BitWrPortI(PBDR,&PBDRShadow,1,5); //clk d0 bit com din=1
  }
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  delay10uS(1);  //10us em cima
  //clk dont care bit com din=x
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  delay10uS(1);  //10us em cima

  //Troca para leitura
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  delay10uS(1);  //10us em cima
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  //vai acima
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  //leitura do b11
  leitura_aux=BitRdPortI(PBDR,7);
```

```
//printf("%d",leitura_aux);
leitura|=leitura_aux;
leitura<<=1; // shift left de uma posição
delay10uS(1);  //10us em cima
//leitura do b10
BitWrPortI(PBDR,&PBDRShadow,0,4);
delay10uS(1); //10us em baixo
//vai acima
BitWrPortI(PBDR,&PBDRShadow,1,4);
leitura_aux=BitRdPortI(PBDR,7);
//printf("%d",leitura_aux);
leitura|=leitura_aux;
leitura<<=1; // shift left de uma posição
delay10uS(1);  //10us em cima

//leitura do b9
BitWrPortI(PBDR,&PBDRShadow,0,4);
delay10uS(1); //10us em baixo
//vai acima
BitWrPortI(PBDR,&PBDRShadow,1,4);
leitura_aux=BitRdPortI(PBDR,7);
//printf("%d",leitura_aux);
leitura|=leitura_aux;
leitura<<=1; // shift left de uma posição
delay10uS(1);  //10us em cima

//leitura do b8
BitWrPortI(PBDR,&PBDRShadow,0,4);
delay10uS(6); //10us em baixo
//vai acima
BitWrPortI(PBDR,&PBDRShadow,1,4);
leitura_aux=BitRdPortI(PBDR,7);
//printf("%d",leitura_aux);
leitura|=leitura_aux;
leitura<<=1; // shift left de uma posição
delay10uS(1);  //10us em cima

//leitura do b7
BitWrPortI(PBDR,&PBDRShadow,0,4);
delay10uS(1); //10us em baixo
//vai acima
BitWrPortI(PBDR,&PBDRShadow,1,4);
leitura_aux=BitRdPortI(PBDR,7);
//printf("%d",leitura_aux);
leitura|=leitura_aux;
leitura<<=1; // shift left de uma posição
delay10uS(1);  //10us em cima

//leitura do b6
BitWrPortI(PBDR,&PBDRShadow,0,4);
delay10uS(1); //10us em baixo
//vai acima
```

```
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  leitura_aux=BitRdPortI(PBDR,7);
  //printf("%d",leitura_aux);
  leitura|=leitura_aux;
  leitura<<=1; // shift left de uma posição
  delay10uS(1);  //10us em cima

  //leitura do b5
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  //vai acima
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  leitura_aux=BitRdPortI(PBDR,7);
  //printf("%d",leitura_aux);
  leitura|=leitura_aux;
  leitura<<=1; // shift left de uma posição
  delay10uS(1);  //10us em cima

  //leitura do b4
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  //vai acima
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  leitura_aux=BitRdPortI(PBDR,7);
  //printf("%d",leitura_aux);
  leitura|=leitura_aux;
  leitura<<=1; // shift left de uma posição
  delay10uS(1);  //10us em cima

  //leitura do b3
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  //vai acima
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  leitura_aux=BitRdPortI(PBDR,7);
  //printf("%d",leitura_aux);
  leitura|=leitura_aux;
  leitura<<=1; // shift left de uma posição
  delay10uS(1);  //10us em cima

  //leitura do b2
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  //vai acima
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  leitura_aux=BitRdPortI(PBDR,7);
  //printf("%d",leitura_aux);
  leitura|=leitura_aux;
  leitura<<=1; // shift left de uma posição
  delay10uS(1);  //10us em cima

  //leitura do b1
```

```
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  //vai acima
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  leitura_aux=BitRdPortI(PBDR,7);
  //printf("%d",leitura_aux);
  leitura|=leitura_aux;
  leitura<<=1; // shift left de uma posição
  delay10uS(1);  //10us em cima

  //leitura do b0
  BitWrPortI(PBDR,&PBDRShadow,0,4);
  delay10uS(1); //10us em baixo
  //vai acima
  BitWrPortI(PBDR,&PBDRShadow,1,4);
  leitura_aux=BitRdPortI(PBDR,7);
  //printf("%d",leitura_aux);
  leitura|=leitura_aux;
  delay10uS(1);  //10us em cima

  //colocar cs a valor alto.
  BitWrPortI(PBDR,&PBDRShadow,1,3);
  //colocar clk a valor alto.
  BitWrPortI(PBDR,&PBDRShadow,1,4);

  return leitura;
}
```

Anexo 2 – Função criada para implementar o mecanismo de comunicação para a leitura da temperatura obtida por um sensor TMP75

Seguidamente é apresentada a função criada para implementar o mecanismo de comunicação para a leitura da temperatura obtida por um sensor TMP75. Todas as etapas do processo estão identificadas pelos comentários (texto de cor verde).

```
/*** Beginheader tmp75In*/
int tmp75In(myBYTE enderecoSensor,myBYTE *erro);
/*** endheader */

/* START FUNCTION DESCRIPTION ****************************************
tmp75In                    <Auxiliar.LIB>

Sintaxe:      int tmp75In(myBYTE enderecoSensor,myBYTE *erro);

Descrição:    Lê o valor de um sensor TMP75 ligado ao bus formado por
              PG.6 (SCL) e PG.7 (SDA).

Parâmetro 1:  Endereço a ler: 1..8 - Endereço do sensor no barramento.

Parâmetro 3:  Erro a enviar:
                    0 - Não ocorreu erro nenhum na fase de configuração.
                    1 - Não recebeu ack ao endereço enviado na fase de
configuração.
      2 - Não recebeu ack ao Pointer Register enviado na fase de configuração.
      3 - Não recebeu um ack ao valor a aplicar ao registo de configurações
          (R0 e R1 a 1 para resolução de 12 bit's) na fase de configuração.
      4 - Não recebeu ack ao endereço enviado na fase de reconfiguração do
          pointer register para temperature register.
      5 - Não recebeu ack ao Pointer Register enviado na fase de reconfiguração
          do pointer register para temperature register.
      6 - Não recebeu ack ao endereço enviado na fase de leitura da temperatura.
      7 - Número (endereço do sensor) inválido.

Valor de retorno:  Valor lido.

END DESCRIPTION ***************************************************/
int tmp75In(myBYTE enderecoSensor,myBYTE *erro)
{
   myBYTE endereçoConvertido;
   myBYTE auxiliar;
   int PXDDR;
   char PXDDRShadow;
   int PXDR;
   char PXDRShadow;
```

```
myBYTE SDA;
myBYTE SCL;
int leitura,i;

  PXDDR=PGDDR;
  PXDDRShadow=PGDDRShadow;
  PXDR=PGDR;
  PXDRShadow=PGDRShadow;
  SDA=7;
  SCL=6;

  i=0;
  leitura=0;

  switch (enderecoSensor)
  {
      case 0:
               endereçoConvertido = 72; // 1001000
                           break;
      case 1:
               endereçoConvertido = 73; // 1001001
                   break;
      case 2:
               endereçoConvertido = 74; // 1001010
                   break;
      case 3:
               endereçoConvertido = 75; // 1001011
                   break;
      case 4:
               endereçoConvertido = 76; // 1001100
                   break;
      case 5:
               endereçoConvertido = 77; // 1001101
                   break;
      case 6:
               endereçoConvertido = 78; // 1001110
                   break;
      case 7:
               endereçoConvertido = 79; // 1001111
                   break;
          default:
                   *erro=7;
                   return 0;
   }

   *erro=0;

   // SDA configurada como saida
   BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
   // SCL configurado como saida
   BitWrPortI(PXDDR,&PXDDRShadow,1,SCL);
```

```
// SDA Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SDA);
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);  // SDA 10us em cimacom SCL em cima

// Enviar um START,
// para isso SDA terá que vir a 0 enquanto SCL está a 1

// SDA Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
delay10uS(1); // SDA 10us em baixo com SCL em cima
// SCK Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SCL);
delay10uS(1);

// Enviar o endereço de 7 bit's do bit mais
// significativo para o menos significarivo

for(i=6;i>=0;i--)
{
   BitWrPortI(PXDR,&PXDRShadow,((endereçoConvertido>>i)&1),SDA);
   //delay10uS(1);
   // SCK Colocada com o Valor 1
   BitWrPortI(PXDR,&PXDRShadow,1,SCL);
   delay10uS(1);
   // SCK Colocada com o Valor 0
   BitWrPortI(PXDR,&PXDRShadow,0,SCL);
   delay10uS(1);
}

// Agora envia o comando de escrita (0)
// para escrever as configurações
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
//delay10uS(1);
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);
// SCK Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SCL);

// Apanha o ack
// SDA configurada como entrada
BitWrPortI(PXDDR,&PXDDRShadow,0,SDA);

// Último clk (9) - CLK do ack
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);

if(!BitRdPortI(PXDR,SDA))
{
```

```
//////////////////////////////////////////////////////////////////////
// ack bem recebido
// Enviar o Pointer Register das configurações
//////////////////////////////////////////////////////////////////////

  // SCK Colocada com o Valor 0
 BitWrPortI(PXDR,&PXDRShadow,0,SCL);
 delay10uS(1);

 // SDA configurada como Saida
 BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
 // SDA Colocada com o Valor 1
 //BitWrPortI(PXDR,&PXDRShadow,1,SDA);

 auxiliar=1; //00000001 - Configuration Register.

 for(i=7;i>=0;i--)
 {
     BitWrPortI(PXDR,&PXDRShadow,((auxiliar>>i)&1),SDA);
     //delay10uS(1);
     // SCK Colocada com o Valor 1
     BitWrPortI(PXDR,&PXDRShadow,1,SCL);
     delay10uS(1);
     // SCK Colocada com o Valor 0
     BitWrPortI(PXDR,&PXDRShadow,0,SCL);
     delay10uS(1);
 }

 // Apanha o ack
 // SDA configurada como entrada
 BitWrPortI(PXDDR,&PXDDRShadow,0,SDA);

 // Último clk (9) - CLK do ack
 // SCK Colocada com o Valor 1
 BitWrPortI(PXDR,&PXDRShadow,1,SCL);
 delay10uS(1);


 if(!BitRdPortI(PXDR,SDA))
 {   ////////////////////////////////////////////////////////////
     // ack bem recebido
     // Enviar o valor de configuração com R0 e R1 a 1 - 01100000
     // 12 bit's de resolução
     ////////////////////////////////////////////////////////////

     // SCK Colocada com o Valor 0
     BitWrPortI(PXDR,&PXDRShadow,0,SCL);
     delay10uS(1);

     // SDA configurada como Saida
     BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
     // SDA Colocada com o Valor 1
```

```
//BitWrPortI(PXDR,&PXDRShadow,1,SDA);
//delay10uS(1);

auxiliar=96; // 01100000 - Configuration Register.

for(i=7;i>=0;i--)
{
    BitWrPortI(PXDR,&PXDRShadow,((auxiliar>>i)&1),SDA);
    //delay10uS(1);
    // SCK Colocada com o Valor 1
    BitWrPortI(PXDR,&PXDRShadow,1,SCL);
    delay10uS(1);
    // SCK Colocada com o Valor 0
    BitWrPortI(PXDR,&PXDRShadow,0,SCL);
    delay10uS(1);
}

// Apanha o ack
// SDA configurada como entrada
BitWrPortI(PXDDR,&PXDDRShadow,0,SDA);

// Último clk (9) - CLK do ack
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);

if(!BitRdPortI(PXDR,SDA))
{
  ////////////////////////////////////////////////////////////////
  // ack bem recebido
  // Enviar o STOP Data Transfer
  ////////////////////////////////////////////////////////////////

  // SCK Colocada com o Valor 0
  BitWrPortI(PXDR,&PXDRShadow,0,SCL);
  delay10uS(1);

  // NÂO ESQUECER QUE QUANDO O 9º CLK (CLK DO ACK) DESCE,
  // O SENSOR COLOCA (FORÇA) SDA A 1

  // SDA configurada como Saida
  BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
  // SDA Colocada com o Valor 0
  BitWrPortI(PXDR,&PXDRShadow,0,SDA);
  delay10uS(1);

  // SCK Colocada com o Valor 1
  BitWrPortI(PXDR,&PXDRShadow,1,SCL);
  delay10uS(1);

  // SDA Colocada com o Valor 1
  BitWrPortI(PXDR,&PXDRShadow,1,SDA);
```

```
delay10uS(1);

////////////////////////////////////////////////////////////////
// RECONFIGURAR AGORA O POINTER REGISTER PARA TEMPERATURE REGISTER
// 1º configurar o pointer register
// Enviar um START,
// para isso SDA terá que vir a 0 enquanto SCL está a 1
////////////////////////////////////////////////////////////////
// SDA Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
delay10uS(1); // SDA 10us em baixo com SCL em cima
// SCK Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SCL);
delay10uS(1);

// Enviar o endereço de 7 bit's do bit mais
// significativo para o menos significarivo

for(i=6;i>=0;i--)
{
    BitWrPortI(PXDR,&PXDRShadow,((endereçoConvertido>>i)&1),SDA);
    //delay10uS(1);
    // SCK Colocada com o Valor 1
    BitWrPortI(PXDR,&PXDRShadow,1,SCL);
    delay10uS(1);
    // SCK Colocada com o Valor 0
    BitWrPortI(PXDR,&PXDRShadow,0,SCL);
    delay10uS(1);
}

// Agora envia o comando de escrita (0)
// para configurar o pointer Register
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
//delay10uS(1);
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);
// SCK Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SCL);

// Apanha o ack
// SDA configurada como entrada
BitWrPortI(PXDDR,&PXDDRShadow,0,SDA);

// Último clk (9º) - CLK do ack
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);

if(!BitRdPortI(PXDR,SDA))
{
    ////////////////////////////////////////////////////////////
```

```
// ack bem recebido
// Enviar o Pointer Register da temperatura
////////////////////////////////////////////////////////////

// SCK Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SCL);
delay10uS(1);

// SDA configurada como Saída
BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
// SDA Colocada com o Valor 1
//BitWrPortI(PXDR,&PXDRShadow,1,SDA);

auxiliar=0; //00000000 - Temperature Register.

for(i=7;i>=0;i--)
{
   BitWrPortI(PXDR,&PXDRShadow,((auxiliar>>i)&1),SDA);
   //delay10uS(1);
   // SCK Colocada com o Valor 1
   BitWrPortI(PXDR,&PXDRShadow,1,SCL);
   delay10uS(1);
   // SCK Colocada com o Valor 0
   BitWrPortI(PXDR,&PXDRShadow,0,SCL);
   delay10uS(1);
}

// Apanha o ack
// SDA configurada como entrada
BitWrPortI(PXDDR,&PXDDRShadow,0,SDA);

// Último clk (9º) - CLK do ack
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);


if(!BitRdPortI(PXDR,SDA))
{
   ////////////////////////////////////////////////////////////
   // ack bem recebido
   // Enviar o STOP Data Transfer
   ////////////////////////////////////////////////////////////

   // SCK Colocada com o Valor 0
   BitWrPortI(PXDR,&PXDRShadow,0,SCL);
   delay10uS(1);

   // NÂO ESQUECER QUE QUANDO O 9 CLK (CLK DO ACK) DESCE, O
   // SENSOR COLOCA (FORÇA) SDA A 1

   // SDA configurada como Saida
```

```
BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
// SDA Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
delay10uS(1);

// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);

// SDA Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SDA);
delay10uS(1);

////////////////////////////////////////////////////////
// LER AGORA A TEMPERATURA
// Enviar um START,
// para isso SDA terá que vir a 0 enquanto SCL está a 1
// Enviar um START,
// para isso SDA terá que vir a 0 enquanto SCL está a 1
////////////////////////////////////////////////////////

// SDA Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
delay10uS(1); // SDA 10us em baixo com SCL em cima
// SCK Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SCL);
delay10uS(1);

// Enviar o endereço de 7 bit's do bit mais
// significativo para o menos significativo

for(i=6;i>=0;i--)
{
   BitWrPortI(PXDR,&PXDRShadow,(
                              (endereçoConvertido>>i)&1),SDA);
   //delay10uS(1);
   // SCK Colocada com o Valor 1
   BitWrPortI(PXDR,&PXDRShadow,1,SCL);
   delay10uS(1);
   // SCK Colocada com o Valor 0
   BitWrPortI(PXDR,&PXDRShadow,0,SCL);
   delay10uS(1);
}

// Agora envia o comando de leitura (1)
// para ler a temperatura
BitWrPortI(PXDR,&PXDRShadow,1,SDA);
//delay10uS(1);
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);
// SCK Colocada com o Valor 0
```

```
BitWrPortI(PXDR,&PXDRShadow,0,SCL);

// Apanha o ack
// SDA configurada como entrada
BitWrPortI(PXDDR,&PXDDRShadow,0,SDA);

// Último clk (9º) - CLK do ack
// SCK Colocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);

if(!BitRdPortI(PXDR,SDA))
{//////////////////////////////////////////
 // ack bem recebido
 // Ler a temperatura
 //////////////////////////////////////////

  // SCK Colocada com o Valor 0
  BitWrPortI(PXDR,&PXDRShadow,0,SCL);
  delay10uS(1);

  ////////////////////////////////////////////////////////
  // vamos agora ler os 8bit's mais significativos.
  ////////////////////////////////////////////////////////
  for(i=0;i<8;i++)
  {
     // SCK Colocada com o Valor 1
     BitWrPortI(PXDR,&PXDRShadow,1,SCL);
     delay10uS(1);
     leitura<<=1;
     leitura += BitRdPortI(PXDR,SDA);
     // SCK Colocada com o Valor 0
     BitWrPortI(PXDR,&PXDRShadow,0,SCL);
     delay10uS(1);
  }

  ////////////////////////////////////////////////////////
  // lido o primeiro byte
  // Fazer ack à leitura do 1º byte
  // Para isso SDA tem que estar a zero quando o 9º clk
  // vai a cima.
  ////////////////////////////////////////////////////////

  // SDA configurada como Saida
  BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
  // SDA Colocada com o Valor 0
  BitWrPortI(PXDR,&PXDRShadow,0,SDA);
  delay10uS(1);
  // SCKColocada com o Valor 1
  BitWrPortI(PXDR,&PXDRShadow,1,SCL);
  delay10uS(1);
  // SCK Colocada com o Valor 0
```

```
BitWrPortI(PXDR,&PXDRShadow,0,SCL);
delay10uS(1);

// SDA configurada como entrada
BitWrPortI(PXDDR,&PXDDRShadow,0,SDA);

//////////////////////////////////////////////////////
// Vamos agora ler os 8bit's menos significativos.
//////////////////////////////////////////////////////
for(i=0;i<8;i++)
{
   // SCK Colocada com o Valor 1
   BitWrPortI(PXDR,&PXDRShadow,1,SCL);
   delay10uS(1);
   leitura<<=1;
   leitura += BitRdPortI(PXDR,SDA);
   // SCK Colocada com o Valor 0
   BitWrPortI(PXDR,&PXDRShadow,0,SCL);
   delay10uS(1);
}
//////////////////////////////////////////////////////
// lido o segundo byte
// Fazer ack à leitura do 2º byte
// Para isso SDA tem que estar a zero quando o 9º clk
// vai a cima.
//////////////////////////////////////////////////////

// SDA configurada como Saida
BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
// SDA Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
delay10uS(1);
// SCKColocada com o Valor 1
BitWrPortI(PXDR,&PXDRShadow,1,SCL);
delay10uS(1);
// SCK Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SCL);
delay10uS(1);

//////////////////////////////////////////////////////
// Enviar o STOP Data Transfer
//////////////////////////////////////////////////////

// NÂO ESQUECER QUE QUANDO O 9º CLK (CLK DO ACK) DESCE,
// O SENSOR COLOCA (FORÇA) SDA A 1

// SDA configurada como Saida
BitWrPortI(PXDDR,&PXDDRShadow,1,SDA);
// SDA Colocada com o Valor 0
BitWrPortI(PXDR,&PXDRShadow,0,SDA);
delay10uS(1);
```

```
                    // SCK Colocada com o Valor 1
                    BitWrPortI(PXDR,&PXDRShadow,1,SCL);
                    delay10uS(1);

                    // SDA Colocada com o Valor 1
                    BitWrPortI(PXDR,&PXDRShadow,1,SDA);
                    delay10uS(1);

                    // Como a temperatura é a 12 bit's os últimos
                    // 4 bit's recebidos são lixo
                    return leitura>>4;
                }
                else
                {
                     // não recebeu um ack ao endereço aquando da leitura
                     // da temperatura
                     *erro=6;
                }
             }
             else
             {
                    // não recebeu um ack ao pointer register aquando da
                    // reconfiguração do pointer register para temp. register
                    *erro=5;
               }
            }
            else
            {
               // não recebeu um ack ao endereço aquando da reconfiguração do
               // pointer register para temperature register
               *erro=4;
            }
            ////////////////////////////////////////////////////////////////
        }
        else
        {
            // não recebeu um ack ao valor a aplicar ao registo de configurações
            *erro=3;
        }
    }
    else
    {
        // não recebeu um ack ao pointer register
        *erro=2;
    }
}
else
{
    // não recebeu um ack ao endereço
    *erro=1;
}
return 0;
```

```
//      // Fazer reset às comunicações.
//      // DATA configurada como saida
//      BitWrPortI(PXDDR,&PXDDRShadow,1,DATA);
//      // SCK configurado como saida
//      BitWrPortI(PXDDR,&PXDDRShadow,1,CLK);
//
//      // DATA Colocada com o Valor 1
//      BitWrPortI(PXDR,&PXDRShadow,0,DATA);
//      // CLK Colocada com o Valor 0
//      BitWrPortI(PXDR,&PXDRShadow,0,CLK); // Se ficar neste estado mais de 54ms
faz reset às comunicações
}
```

Anexo 3 – Função criada para implementar o mecanismo de comunicação para a leitura da temperatura ou humidade obtidas por um sensor SHT75

Seguidamente é apresentada a função criada para implementar o mecanismo de comunicação para a leitura da temperatura ou humidade obtida por um sensor SHT75. Todas as etapas do processo estão identificadas pelos comentários (texto de cor verde).

```
/*** Beginheader sht75In*/
int sht75In(myBYTE entrada, myBYTE tipo_leitura,myBYTE *erro);
/*** endheader */

/* START FUNCTION DESCRIPTION ************************************************
sht75In                    <Auxiliar.LIB>

Sintaxe:        int sht75In(myBYTE entrada, myBYTE tipo_leitura,myBYTE *erro);

Descrição:      Lê o valor de um sensor SHT75.

Parâmetro 1:    Entrada a ler: 1 - sensor SHT75 ligado no PB.0(CLK); PB.2(DATA).
                               2 - sensor SHT75 ligado no PD.4(CLK); PD-5(DATA).

Parâmetro 2:    Tipo de leitura: 1 - Temperatura.
                                 2 - Humidade.

Parâmetro 3:    Erro a enviar: 0 - Não ocorreu erro nenhum.
                               1 - Não recebeu ack ao comando enviado.
                               2 - Não ficou com a linha DATA num nível alto
                                   enquanto faz a leitura
                               3 - Não ficou com a linha DATA num nível baixo
                                   no final da leitura
                               4 - Entrada inválida.
                               5 - Tipo de leitura inválido;

Valor de retorno:  Valor lido.

END DESCRIPTION ***********************************************************/
int sht75In(myBYTE entrada, myBYTE tipo_leitura,myBYTE *erro)
{
   int i, ok, leitura;
   int PXDDR;
   char PXDDRShadow;
   int PXDR;
   char PXDRShadow;
   myBYTE DATA;
   myBYTE CLK;
```

```
i=0;
ok=0;
leitura=0;

  switch (entrada)
  {
          case 1:
                  PXDDR=PBDDR;
                  PXDDRShadow=PBDDRShadow;
                  PXDR=PBDR;
                  PXDRShadow=PBDRShadow;
                  DATA=2;
                  CLK=0;
                  break;
          case 2:
                  PXDDR=PDDDR;
                  PXDDRShadow=PDDDRShadow;
                  PXDR=PDDR;
                  PXDRShadow=PDDRShadow;
                  DATA=5;
                  CLK=4;
                  break;
          default:
                  *erro=4;
                  return 0;
  }

  switch (tipo_leitura)
  {
          case 1:
          case 2:
                  break;
          default:
                  *erro=5;
                  return 0;
  }

  *erro=0;

  // DATA configurada como saida
  BitWrPortI(PXDDR,&PXDDRShadow,1,DATA);
  // SCK configurado como saida
  BitWrPortI(PXDDR,&PXDDRShadow,1,CLK);

  // DATA Colocada com o Valor 1
  BitWrPortI(PXDR,&PXDRShadow,1,DATA);

  // 9 SCK para fazer reset às comunicações
  ////////////////////////////////////////////////////////////////////////
  BitWrPortI(PXDR,&PXDRShadow,0,CLK);
  delay10uS(DelaySHT75Comm); //10us em baixo
  BitWrPortI(PXDR,&PXDRShadow,1,CLK);
```

```
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(10); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(10);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

BitWrPortI(PXDR,&PXDRShadow,0,CLK);
delay10uS(DelaySHT75Comm); //10us em baixo
BitWrPortI(PXDR,&PXDRShadow,1,CLK);
delay10uS(DelaySHT75Comm);  //10us em cima

// Neste momento temos DATA= 1 e SCK=1, fazer Transmition Start colocando
// DATA=0, esperar 10us,
// SCK=0, esperar 10us,
// SCK=1 esperar 10us,
// DATA=1, esperar 10us,
```

```
// SAK=0, esperar 30us para restabelecimento das linhas,
BitWrPortI(PXDR,&PXDRShadow,0,DATA); // DATA=0
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,1,DATA); // DATA=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(10);   // tinha 3
////////////////////////////////////////////////////////////////////////
// Fim do reset das comunicações.
// Vamos agora enviar um comando começando pela cadeia de Transmition Start
////////////////////////////////////////////////////////////////////////
// Neste momento temos DATA= 1 e SCK=0, fazer Transmition Start colocando
// SCK=1 esperar 10us,
// DATA=0, esperar 10us,
// SCK=0, esperar 10us,
// SCK=1 esperar 10us,
// DATA=1, esperar 10us,
// SAK=0, esperar 10us,
// DATA=0, esperar 10us
BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,DATA); // DATA=0
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,1,DATA); // DATA=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,DATA); // DATA=0
delay10uS(DelaySHT75Comm);

// Neste momento temos tudo ok, vamos enviar o comando
// O comando para medir a humidade é: 000 00011 (End + Comando)

// Temos então SCK=0 e DATA=0 e vamos enviar o comando:
// SCK=1 esperar 10us,
// SCK=0, esperar 10us, A2=0
// SCK=1 esperar 10us,
// SCK=0, esperar 10us, A1=0
// SCK=1 esperar 10us,
// SCK=0, esperar 10us, A0=0
// SCK=1 esperar 10us,
// SCK=0, esperar 10us, C4=0
// SCK=1 esperar 10us,
// SCK=0, esperar 10us, C3=0
```

```
// SCK=1 esperar 10us,
// DATA=1, esperar 10us, se humidade
// SCK=0, esperar 10us, C2=0 para temperatura | c2=1 para humidade
// DATA=1, esperar 10us,      DATA=0, esperar 10us, se humidade
// SCK=1 esperar 10us,
// SCK=0, esperar 10us, C1=1 para temperatura | c1=0 para humidade
// SCK=1 esperar 10us,
// SCK=0, esperar 10us, C0=1
BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);          // A2=0

BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);          // A1=0

BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);          // A0=0

BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);          // C4=0

BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);          // C3=0

switch (tipo_leitura)
{
        case 1:
                // Temperatura
                BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
                delay10uS(DelaySHT75Comm);
                BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
                delay10uS(DelaySHT75Comm);          // C2=0

                BitWrPortI(PXDR,&PXDRShadow,1,DATA); // DATA=1
                delay10uS(DelaySHT75Comm);

                BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
                delay10uS(DelaySHT75Comm);
                BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
                delay10uS(DelaySHT75Comm);          // C1=1
            break;
        case 2:
            // Humidade
```

```
                    BitWrPortI(PXDR,&PXDRShadow,1,DATA); // DATA=1
                    delay10uS(DelaySHT75Comm);

                    BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
                    delay10uS(DelaySHT75Comm);
                    BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
                    delay10uS(DelaySHT75Comm);          // C2=1

                    BitWrPortI(PXDR,&PXDRShadow,0,DATA); // DATA=0
                    delay10uS(DelaySHT75Comm);

                    BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
                    delay10uS(DelaySHT75Comm);
                    BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
                    delay10uS(DelaySHT75Comm);          // C1=0
                    break;
}

BitWrPortI(PXDR,&PXDRShadow,1,DATA); // DATA=1
delay10uS(DelaySHT75Comm);

BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
delay10uS(DelaySHT75Comm);      // C0=1
////////////////////////////////////////////////////////////////////////
// Fim do envio do comando
// Temos neste momento o comando enviado e vamos receber o ack
////////////////////////////////////////////////////////////////////////
// vamos configurar o DATA para input
// vamos aguardar 10us  para estabilizar o sinal do ack
// SCK=0, esperar 10us
// SCK=1 esperar 10us,
// SCK=0, esperar 10us
// Validar que DATA=0

//  BitWrPortI(PBDDR,&PBDDRShadow,0,2);  // Data configurada como entrada
//  delay10uS(1);

BitWrPortI(PXDDR,&PXDDRShadow,0,DATA);  // Data configurada como entrada
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,0,CLK);     // SCK=0
delay10uS(DelaySHT75Comm);
BitWrPortI(PXDR,&PXDRShadow,1,CLK);     // SCK=1
delay10uS(DelaySHT75Comm);

if(!BitRdPortI(PXDR,DATA))
{
   ////////////////////////////////////////////////////////////////////
  // ack bem recebido
  // Verificar que o sensor voltou a colocar DATA=1 Esta linha irá ficar
  // no valor lógico alto enquanto a leitura não terminar e
  // dura aproximadamente 320ms
  ////////////////////////////////////////////////////////////////////
```

```
// Colocar de novo a linha de clock a 0 para ficar a postos
BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
delay10uS(DelaySHT75Comm);

delay10uS(10); // esperamos 100us para dar tempo a que DATA=1 estabilize

if(BitRdPortI(PXDR,DATA))
{
     // Vamos então esperar 250ms e validemos que a linha foi colocada
     // no nivel baixo, se não tiver sido aguardamos mais 6x50ms
     // se DATA nunca for colocada a 0 abortamos.
     OSTimeDly(OS_TICKS_PER_SEC/4);

     for(i=0;i<6;i++)
     {
        OSTimeDly(OS_TICKS_PER_SEC/20);
        if(!BitRdPortI(PXDR,DATA))
        {
          ok=1;
          break;
        }
     }
     if(ok)
     {
       ////////////////////////////////////////////////////////////////
       // Ok, terminou a leitura,
       //vamos agora ler os 8bit's mais significativos.
       ////////////////////////////////////////////////////////////////
       for(i=0;i<8;i++)
       {
         leitura<<=1;
         leitura += BitRdPortI(PXDR,DATA);
         BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
         delay10uS(DelaySHT75Comm);
         BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
         delay10uS(DelaySHT75Comm);
       }
       ////////////////////////////////////////////////////////////////
       // lido o primeiro byte
       // Enviar o ack
       ////////////////////////////////////////////////////////////////
       // enviar 1 SCK
       //BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
       //delay10uS(1);
       //BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
       //delay10uS(1);

       // DATA configurada como saída
       BitWrPortI(PXDDR,&PXDDRShadow,1,DATA);
       // DATA Colocada com o Valor 0
       BitWrPortI(PXDR,&PXDRShadow,0,DATA);
```

```
                delay10uS(DelaySHT75Comm);

                // enviar 1 SCK
                BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
                delay10uS(DelaySHT75Comm);
                BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
                delay10uS(DelaySHT75Comm);

                BitWrPortI(PXDDR,&PXDDRShadow,0,DATA); // Data conf.  como entrada
                delay10uS(DelaySHT75Comm);
                ////////////////////////////////////////////////////////////////
                // Ack enviado
                // Vamos agora ler os 8bit's menos significativos
                ////////////////////////////////////////////////////////////////
                for(i=0;i<8;i++)
                {
                  leitura<<=1;
                  leitura += BitRdPortI(PXDR,DATA);
                  BitWrPortI(PXDR,&PXDRShadow,1,CLK); // SCK=1
                  delay10uS(DelaySHT75Comm);
                  BitWrPortI(PXDR,&PXDRShadow,0,CLK); // SCK=0
                  delay10uS(DelaySHT75Comm);
                }
                ////////////////////////////////////////////////////////////////
                // Ok, leitura dos 8bit's menos significativos concluída

                return leitura;
                //sprintf(cauda, "%sFER%.2f ", MAC_Addr, 0.01* leitura-39.7);
              }
              else
              {
                 // não ficou com a linha DATA num nível baixo no sinal da leitura
                 *erro=3;
                 //sprintf(cauda, "%sN03", MAC_Addr);
              }
           }
           else
           {
             // não ficou com a linha DATA num nível alto enquanto faz a leitura
             *erro=2;
                  //sprintf(cauda, "%sN02", MAC_Addr);
           }
        }
        else
        {
           // não recebeu um ack
           *erro=1;
           //sprintf(cauda, "%sN01", MAC_Addr);
        }
        // se ocorrer algum erro
        return 0;
}
```

Anexo 4 – Placa de circuito impresso

Neste anexo são apresentadas as três vistas mais importantes dos layout's das placas de circuito impresso:

- Vista *TOP – Layer* de fabricação da face superior da placa de circuito impresso, designada internamente por face dos componentes, onde figuram todas as pistas, *pad's*, plano de massa e de *keepout* desta face.
  A vista *TOP* é apresentada na Figura 151.

- Vista *Buttom - Layer* de fabricação da face inferior da placa de circuito impresso, designada internamente por face das soldaduras, onde figuram todas as pistas, *pad's*, plano de massa e de *keepout* desta face.
  A vista *Buttom* é apresentada na Figura 152.

- Vista *TOP* com *SilkScreen* – vista da face superior da placa de circuito impresso, designada internamente por face dos componentes, com o local de colocação e designação de todos os componentes.
  A vista *TOP* com *SilkScreen* é apresentada na Figura 153.

As placas são fabricadas por uma empresa externa e tem as seguintes especificações:

- *RoHS* Compliant (sem chumbo, cádmio, mercúrio, cromo hexavalente, bifenilos polibromados nem éteres difenil-polibromados)
- Material: Fibra de vidro FR4 1,6mm
- Dupla Face
- Furo Metalizado 35/35
- Estanhagem HAL (*Hot Air (Solder) Leveling* – pistas cobreadas com as ilhas estanhadas)
- Marcação de componentes a branco
- *Solder Resist* (verniz) verde

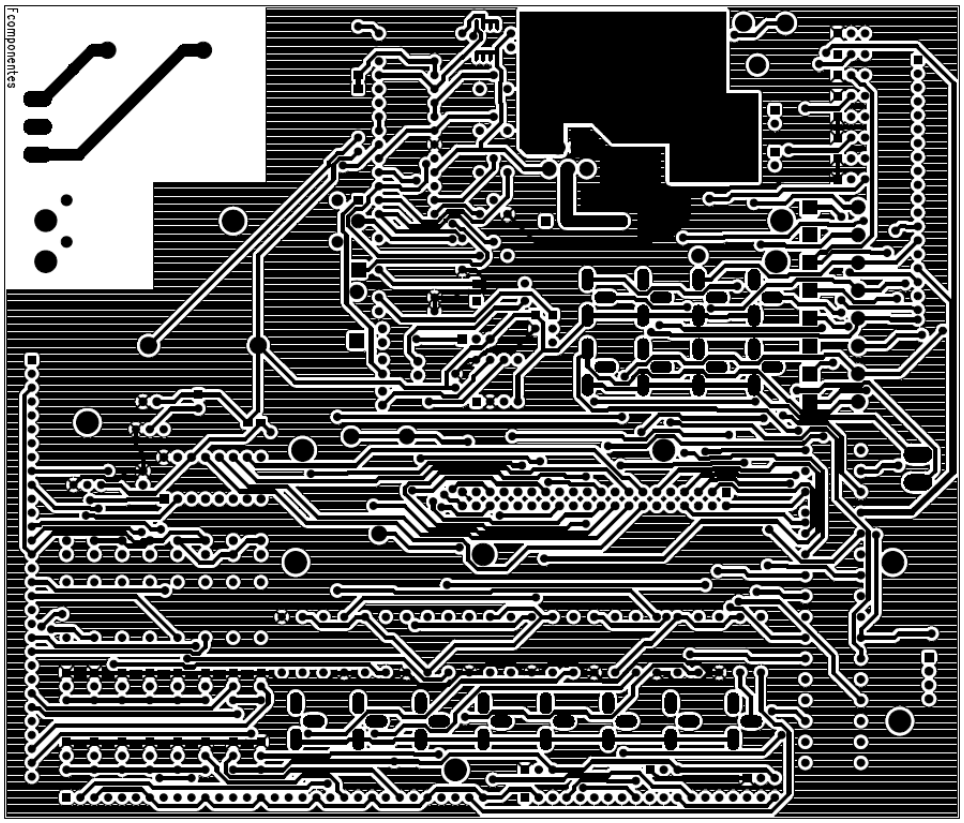


Figura 151: Placa de circuito impresso – *Layer* de fabricação *TOP*.

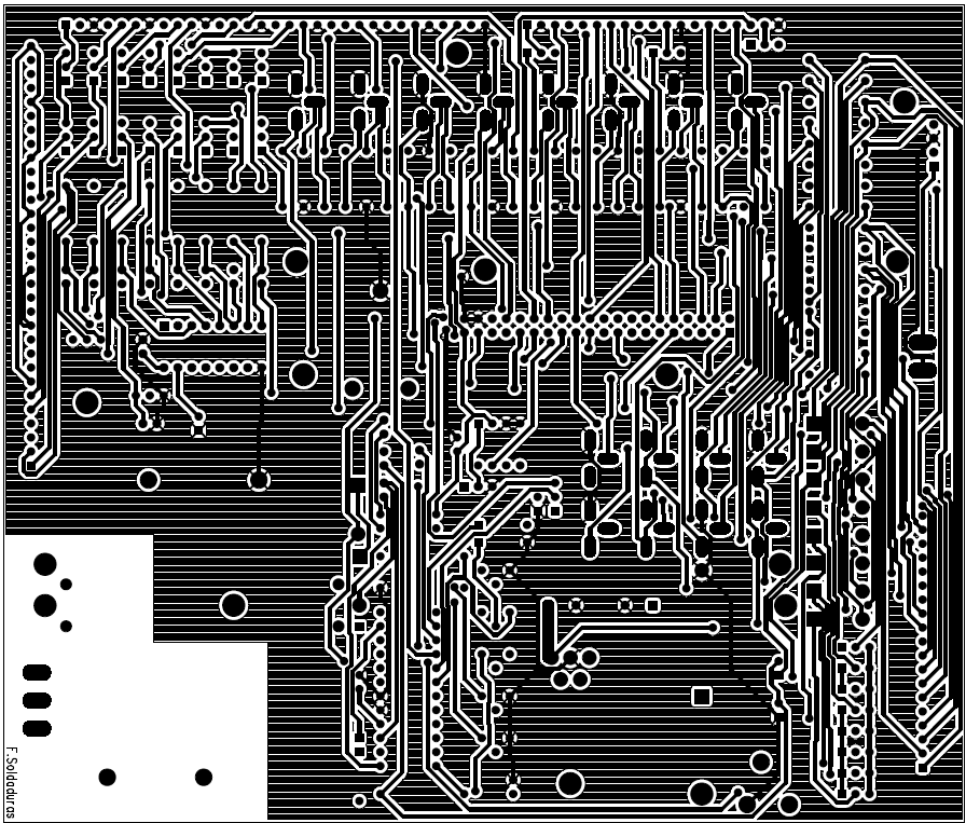


Figura 152: Placa de circuito impresso – *Layer* de fabricação *Buttom*.

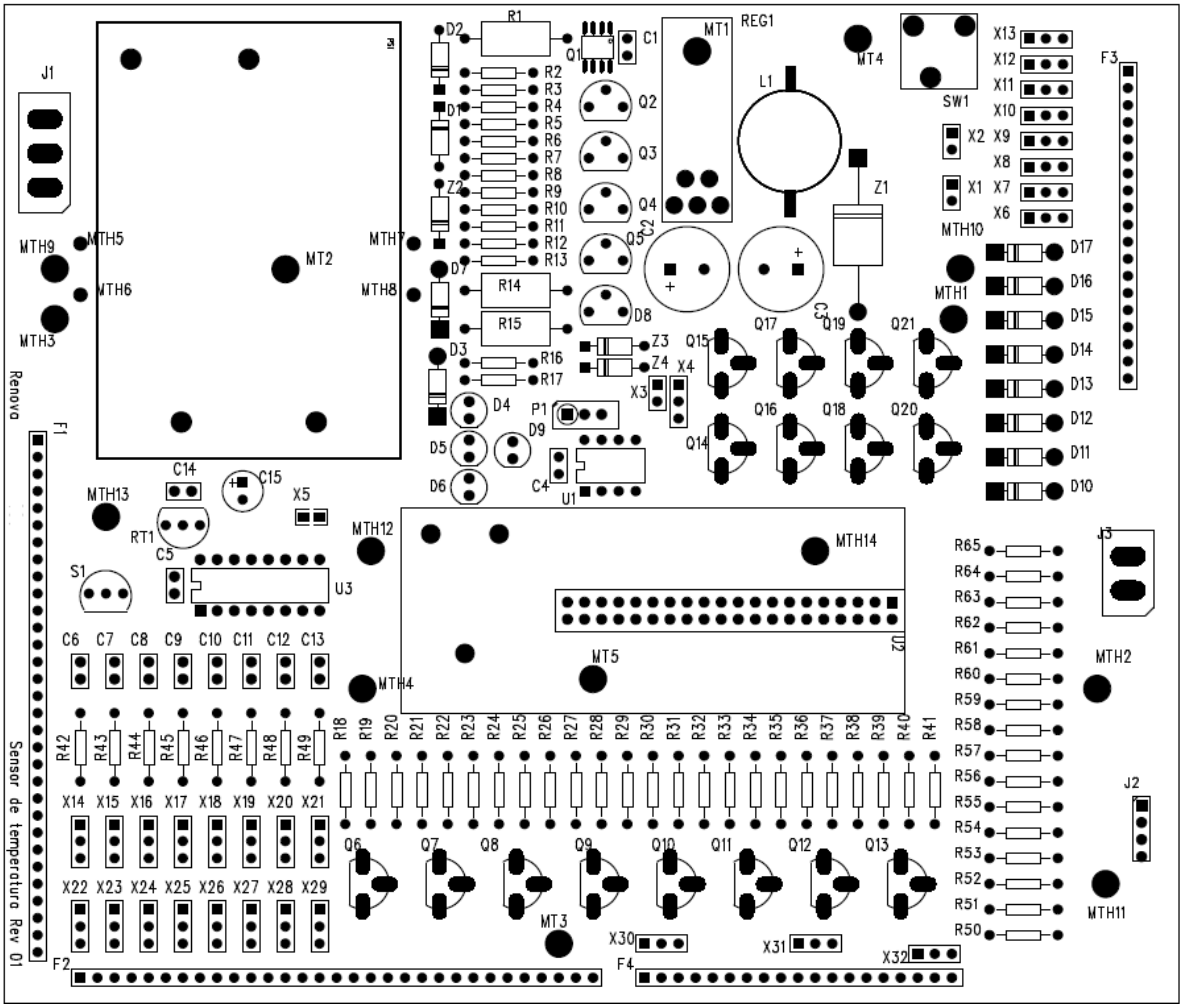


Figura 153: Placa de circuito impresso – Vista *TOP* com *SilkSceen* (face de colocação dos componentes).

As figuras apresentadas não exibem as medidas reais da placa que mede efetivamente 175mm x 149mm.

Anexo 5 – Livrarias de código do programa de controlo do microcontrolador.

(Dada a extensão do texto associado a este anexo, o mesmo apenas se encontra disponível para consulta no cd-rom que acompanha o relatório, na pasta "Anexos/Código Microcontrolador" e é constituído por um ficheiro por livraria com o nome correspondente.)

Anexo 6 – Protocolo de comunicações e lista de comandos de interação com a placa de aquisição de dados RenPAD.

Protocolo de comunicação

O protocolo de comunicação desenvolvido consiste no envio de uma cadeia de caracteres com a seguinte estrutura:

@<Endereço><Mensagem><Cadeia de validação><Terminador>

Podemos então observar que a mensagem é iniciada pelo caracter '**@**' (caracter 64 da tabela ASCII), seguida do **endereço** a 12 bytes do terminal para o qual se destina a mensagem ou, no caso de ser uma mensagem enviada por *broadcast* 12 zeros, da **mensagem** propriamente dita, da **Cadeia de Validação**, constituída por uma cadeia de quatro caracteres que representam a soma hexadecimal de todos os caracteres constituintes da mensagem excepto a '@' e o terminador, e por fim o **identificador de fim de mensagem** que corresponde ao caracter *Line-Feed* (caracter 10 da tabela ASCII).

Nota: Todos os caracteres são enviados no formato ASCII.

Exemplos:

@000000000000402A4

@0090C2C0E10008W05030201200507020306C4

O terminal, depois de recebida uma mensagem, pode assumir cinco tipos distintos de resposta:

1. Se a mensagem for bem recebida e o comando corretamente interpretado e executado, envia um "Ack":

   @<Endereço>A<Validação><LF>

   Exemplo:

   @0090C2C0E100A02C8

   Nota: O "Ack" pode ser precedido por informação relativa ao modo de aceitação do comando como acontece por exemplo no comando 06.

2. Se a mensagem for mal recebida, o comando mal interpretado ou mal executado é enviado um “Nack”:

   @<Endereço>N<Validação><LF>

   Exemplo:

   @0090C2C0E100**N**02D5

   Nota: O “Nack” pode ser precedido por informação relativa ao modo de não-aceitação parcial ou total do comando, como acontece por exemplo no comando 06.

3. Se a mensagem for bem recebida e o comando for de leitura, a resposta é enviada segundo a sintaxe do próprio comando juntamente com o resultado da leitura:

   @<Endereço><Mensagem><Validação><LF>

   Exemplo de resposta ao comando 08R:

   @0090C2C0E100**07R0;20000;10.0.6.205;255.255.0.0;0.0.0.0;**0AA3

4. Se o comando executado for de leitura interativa, isto é, necessitar de confirmação por parte do PC de correta receção para que o terminal envie a informação seguinte, quando o terminal tiver enviado todos os dados, avisa o PC com uma cadeia de caracteres denominada por “EOT” constituída da seguinte forma:

   @<Endereço>F<Validação><LF>

   Exemplo:

   @0090C2C0E100**F**02CD

5. Se a mensagem enviada não necessitar de resposta ou não for identificada como dirigida ao terminal, este não responde.

A figura seguinte mostra um exemplo de pergunta e resposta ao comando 08 de um terminal de recolha de dados TRDN, que utiliza o mesmo protocolo de comunicações, através de programa trdnw32 de linha de comandos.

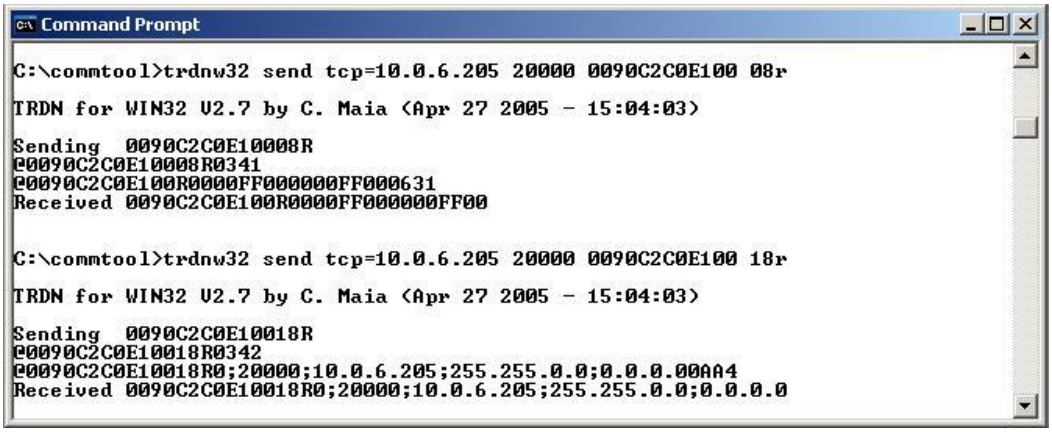


Figura 154: Envio de comandos e receção da resposta para a placa RenPAD por linha de comandos.

Notar que na figura, depois de executado o comando, é descrito o que foi enviado, a cadeia efetivamente enviada com o caracter @ e a validação, a cadeia efetivamente recebida e a cadeia recebida depois de formatada.

O programa trdnw32 não vai ser abordado pois sai do âmbito do projeto.

Lista de comandos implementados

Existem quatro grupos distintos de comandos:

1. Do comando 00 ao comando 0F – Comandos referentes ao funcionamento da aplicação.

   Comando 00 – Procura de dispositivos num dado intervalo por *broadcast*.

   Comando 01 – Procura de dispositivos por endereço.

   Comando 02 – Acerto/Leitura de data/hora por *broadcast*.

   Comando 03 – Acerto do horário de mudança de hora de verão/Inverno.

   Comando 04 – Leitura da informação referente à versão da aplicação, tipo de *hardware*…

   Comando 05 – Comando de controlo da aplicação (leitura/limpeza dos contadores de *resets*, *reset* total com formatação da memória, *reset* parcial).

Comando 06 – Comando de teste dos periféricos (entradas analógicas, entradas digitais, saídas e sensores).

Comando 07 – Programação das configurações de rede pretendidas.

Comando 08 – Leitura ou escrita da descrição do terminal.

Comando 09 – Leitura ou escrita do fator de *delay* de comunicações para os sensores SHT75.

Restantes valores – Futura aplicação.

2. Do comando 10 ao comando 1F – Comandos referentes à leitura de entradas analógicas, digitais e sensores.

   Comando 10 – Leitura de todas as entradas do dispositivo em simultâneo.

   Comando 11 – Leitura de todas as entradas digitais em simultâneo.

   Comando 12 – Leitura de todas as entradas analógicas em simultâneo.

   Comando 13 – Leitura dos valores de temperatura e humidade de todos os sensores SHT75 em simultâneo.

   Comando 14 – Leitura dos valores de temperatura de todos os sensores TMP75 em simultâneo.

   Restantes valores – Futura aplicação.

3. Do comando 20 ao comando 2F – Comandos referentes à configuração de alarmes inerentes ao funcionamento ao funcionamento das entradas analógicas, digitais e sensores.

   Comando 21 – Configuração dos alarmes associados às entradas digitais.

   Comando 22 – Configuração dos alarmes associados às entradas analógicas.

   Comando 23 – Configuração dos alarmes associados aos sensores SHT75.

   Comando 24 – Configuração dos alarmes associados aos sensores TMP75.

   Restantes valores – Futura aplicação.

4. Do comando 30 ao comando 3F – Comandos referentes à leitura do estado das saídas.

   Comando 30 – Leitura do estado de todas as saídas em simultâneo.

   Restantes valores – Futura aplicação.

Podem ainda ser usados para futuras aplicações as designações livres até à FF.

Descrição pormenorizada de cada comando

Comando 00

Como descrito anteriormente o comando 00 procura terminais situados num dado intervalo de endereços por *broadcast* e tem a seguinte configuração:

00(end. baixo)(end. alto)

Sendo que;

- end. baixo – representa a base do intervalo de endereços.
- end. alto – representa o topo do intervalo de endereços.

Dado que todos os terminais situados no intervalo referido vão responder a este comando com um "Ack" a resposta recebida pelo PC será incompreensível exceto se existir apenas um terminal no intervalo fornecido.

Exemplo: Procura de terminais situado no intervalo de endereços 000000000010-0FFFFFFFFFFF.

Figura 155: Exemplo de envio do comando de pesquisa de terminais por *broadcast*.

Comando 01

Este comando faz uma procura de um terminal através do seu endereço *MAC*.

Se o terminal existir na rede envia um "Ack" senão, nada é recebido.

O comando tem a seguinte configuração:

01

Exemplo: Procura o terminal com o endereço 0090C2C2D8D1.

Sending 0090C2C0E10001
@0090C2C2C0E1000102E8
@0090C2C2C0E100A02C8

Received 0090C2C0E100A

Comando 02

Comando de acerto e leitura da data e hora do terminal. O comando de acerto pode ser enviado por *broadcast*.

- O comando tem a seguinte configuração para o acerto da data e hora:

  02W(AA)(MM)(DD)(WW)(HH)(mm)(SS)

  O terminal responde com um "Ack caso seja bem interpretado e executado. Caso contrário responde com um "Nack".

- Para leitura da hora definida no sistema:

  02R

  O terminal responde com uma cadeia de caracteres idêntica à cadeia de escrita:

  02R(AA)(MM)(DD)(WW)(HH)(mm)(SS)

  Com:

  - AA – Ano
  - MM – Mês
  - DD – Dia
  - WW – Dia da semana (0 → Domingo .. 6 → Sábado)
  - HH – Horas
  - mm – Minutos
  - SS – Segundos

  O "Ack" e o "Nack" deste comando assumem a seguinte forma:

  (ADD)X(IG)

  Com:

  - X – A – Aceite sem erros

    N – Negado ou parcialmente aceite.

  - IG – Informação geral.

00 – Operação efectuada sem erros.

01 – Cadeia de caracteres enviada com dimensão inferior ao esperado.

02 – Valor de AA inválido.

03 – Valor de MM inválido.

04 – Valor de DD inválido.

05 – Valor de WW inválido.

06 – Valor de HH inválido.

07 – Valor de mm inválido.

08 – Valor de ss inválido.

Exemplo: Acerto da data e hora para segunda (WW=01), 3 (DD=03) de Janeiro (MM=01) de 2005 (AA=05) 15H35m10s (HH=15; mm=35; SS=10).

Sending 0090C2C0E1000205010301153510

@0090C2C0E1000205010301153510 05A2

@0090C2C0E100A02C8

Received 0090C2C0E100A

Comando 03

Comando para carregar a tabela de data de mudança de horário de Verão/Inverno.

Quando a memória de dados do terminal é formatada, estas datas são apagadas, sendo necessário carregá-las novamente.

Este comando dispõe de três configurações possíveis:

- Para escrita dos dados:

  03W(AP)(MP)(DP)(WP)(HP) (AI)(MI)(DI)(WI)(HI)

  O terminal responde com um "Ack caso seja bem interpretado e executado. Caso contrário responde com um "Nack".

- Para leitura de dados:

  03R

  O terminal responde com uma cadeia de caracteres idêntica à cadeia de escrita:

03R(AP)(MP)(DP)(WP)(HP) (AO)(MO)(DO)(WO)(HO)

Caso alguma data já tenha sido cumprida ou caso a memória seja formatada temos a seguinte cadeia:

Ano – 00
Mês – 00
Dia – 00
Dia da semana – FF
Hora – 00

- Para eliminação dos dados (coloca a cadeia de caracteres vista em cima, nas datas):

03D

O terminal responde com um "Ack" caso seja bem interpretado e executado. Caso contrário responde com um "Nack".

Com:

- W – *Write* (escrita).
- R – *Read* (leitura).
- D – *Delete* (apagar).
- AP – Ano de mudança de hora Inverno/Verão.
- MP – Mês de mudança de hora Inverno/Verão.
- DP – Dia de mudança de hora Inverno/Verão.
- WP – Dia da semana de mudança de hora Inverno/Verão(0→Dom..6→Sáb).
- HP – Hora de mudança do horário de Inverno/Verão.
- AO – Ano de mudança de hora Verão/Inverno.
- MO – Mês de mudança de hora Verão/Inverno.
- DO – Dia de mudança de hora Verão/Inverno.
- WO – Dia da semana de mudança de hora Verão/Inverno.
- HO – Hora de mudança do horário de Verão/Inverno.

O "Ack" e o "Nack" deste comando assumem a seguinte forma:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efectuada sem erros.

  01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida.

  02 – Valor de AP inválido.

  03 – Valor de MP inválido.

  04 – Valor de DP inválido.

  05 – Valor de WP inválido.

  06 – Valor de HP inválido.

  07 – Valor de AO inválido.

  08 – Valor de MO inválido.

  09 – Valor de DO inválido.

  0A – Valor de WO inválido.

  0B – Valor de HO inválido.

Exemplo: Escrita das datas quarta 23-03-2005 às 2h; sexta 21-10-2005 às 3h.

Sending 0090C2C0E10003W05032303020510210503
@0090C2C0E10003W0503230302051021050307 24
@0090C2C0E100A02C8
Received 0090C2C0E100A

Leitura da data.

Sending 0090C2C0E10003R
@0090C2C0E10008R033C
@0090C2C0E10003R05230302102105030 5F2
Received 0090C2C0E10003R0523030210210503

Apagar a data.

Sending 0090C2C0E10003D

@0090C2C0E10003D0333
@0090C2C0E100A02C8
Received 0090C2C0E100A

Nova leitura.

Sending 0090C2C0E10003R
@0090C2C0E10003R0341
@0090C2C0E10003R0000FF000000FF000631
Received 0090C2C0E10003R0000FF000000FF00

Comando 04

Este comando adquire a informação relativa às especificações técnicas do terminal tais como: versão e data da aplicação, versão do *hardware*, tipo de processador.

O comando tem a seguinte configuração:

04

A resposta a este comando é constituída pela seguinte cadeia de caracteres:

04(TTTT)(VVVV)(AA)(MM)(DD)(P)(HH)

Com:

- TTTT – Versão de *software* (diferencia uma placa RenPAD de um TRDN por exemplo).
- VVVV – Versão da aplicação
- AA – Ano da versão.
- MM – Mês da versão.
- DD – Dia da versão.
- HH – Hora da versão.
- mm – Minuto da versão
- SS – Segundo da versão.
- P – Tipo de processador (6-RCM3700).
- HHHH – Versão do *hardware*.

Nota:

TTTT é dividido em dois grupos sendo o primeiro relacionado com o tipo de *software* presente no terminal e o segundo a versão. Por exemplo: 0500 -> 05=RenPAD e 00=v0.

HHHH é dividido em dois grupos sendo o primeiro relacionado com o tipo de *hardware* presente no terminal e o segundo a versão. Por exemplo: 6001 -> 60= RenPAD e 01= *boards* de revisão rev1.

Comando 05

Este é o comando de controlo da aplicação e tem as seguintes configurações:

- Leitura dos contadores de *Resets*:

  05R

  A resposta a este comando consiste na seguinte cadeia de caracteres:

  05R(TR)(WR)(AA)(MM)(DD)(WW)(HH)(MM)(SS)

- Limpeza dos contadores de *Reset*:

  05D

  O terminal responde a este comando com um "Ack" ou com um "Nack".

- Forçar um *reset* total ou parcial:

  05F(T)

  O terminal responde a este comando com um "Ack" ou com um "Nack".

  Com:

  - R – *Read* (leitura).
  - D – *Delete* (apagar).
  - F – Forçar um *reset*.
  - T – Tipo de *reset*: 1 – *reset* total da aplicação com formatação da memória (que implica total perca de dados e colocação do terminal no estado igual à saída da fábrica).

0 – *reset* parcial da aplicação (apenas formata a memória caso os dados estejam inconsistentes).

- TR – Número total de *Resets* (*Hardware*+*Software*+*WatchDog*).
- WR – Número total de *WatchDog resets*.
- AA – Ano da última limpeza dos contadores de *resets*.
- MM – Mês da última limpeza dos contadores de *resets*.
- DD – Dia da última limpeza dos contadores de *resets*.
- WW – Dia da semana da última limpeza dos contadores de *resets* (0→Domingo a 6→Sábado).
- HH – Hora da última limpeza dos contadores de *resets*.
- MM – Minutos da última limpeza dos contadores de *resets*.
- SS – Segundos da última limpeza dos contadores de *resets*.

Quando a aplicação é atualizada, os contadores de *resets* podem adquirir valores aleatórios e por este motivo, é aconselhável limpá-los sempre depois de uma atualização.

O "Ack" e o "Nack" deste comando assumem a seguinte forma:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efetuada sem erros.

  01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida.

  02 – Valor de T inválido.

Comando 06

Este comando permite testar o funcionamento dos periféricos ligados ao terminal para verificação de eventuais avarias.

O teste é realizado procedendo à leitura efetiva dos valores das entradas/sensores a testar ou afetando diretamente as saídas a testar.

O "Ack" e o "Nack" deste comando assumem a seguinte forma:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral (descrita em cada configuração do comando).

O comando tem as seguintes configurações possíveis:

- <u>Teste de saídas digitais:</u>

  06SD(M0)(M1)(TT)

  Com:

  - M0 – Valor hexadecimal correspondente às saídas a colocar no nível lógico 0 – 00 a FF.

    b7 b6 b5 b4 b3 b2 b1 b0
    S8 S7 S6 S5 S4 S3 S2 S1

    Por exemplo, para se colocarem as saídas S8, S6 e S1 a zero este valor deverá ser – A1

  - M1 – Valor hexadecimal correspondente às saídas a colocar no nível lógico 1 – 00 a FF.

    b7 b6 b5 b4 b3 b2 b1 b0
    S8 S7 S6 S5 S4 S3 S2 S1

    Por exemplo, para se colocarem as saídas S7, S4,S3 e S1 a um este valor deverá ser – 4D

  - TT – Valor hexadecimal correspondente ao tempo em segundos, de duração do teste (01 a FF).

Notas:

Os valores não contemplados na reunião de M0 e M1 permanecerão no nível lógico em que se encontram imediatamente antes do teste (o teste não interferirá com o estado destas saídas).

Os valores contemplados na intersecção de M0 e M1 assumirão o nível lógico baixo.

Durante o teste a atualização do estado dos alarmes será suspensa.

Assumindo, por exemplo, que as saídas se encontram no estado 0x1F antes do teste e que enviamos o comando de teste com a configuração 07SD8143 teremos o seguinte resultado final:

	S8	S7	S6	S5	S4	S3	S2	S1
Estado inicial – 0x1F	0	0	0	1	1	1	1	1
Saídas a testar a 0 – 0x85	0	X	X	X	0	X	X	0
Saídas a testar a 1 – 0x43	X	1	X	X	X	X	1	1
Estado de teste de saídas – 0x56	0	1	0	1	0	1	1	0

Tabela 23: Exemplo dos estados das saídas, assumidos durante o envio de um comando 06

A resposta a este comando consiste no envio de um "Ack" ou de um "Nack" com a forma descrita anteriormente com IG definido por:

00 – Operação efetuada sem erros.
01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida.
02 – Valor de M0 inválido – 00 a FF.
03 – Valor de M1 inválido – 00 a FF.
04 – Valor de TT inválido (01 a FF).
05 – Já está a decorrer um teste de saídas.

- <u>Teste de entradas digitais:</u>

06ED(X)

O terminal responde a este comando enviando uma cadeia de caracteres com o seguinte formato:

06ED(X)-(E)

Com:

- X – Entrada digital a ler – 1 a 8.
- E – Valor binário do estado lógico em que se encontra a entrada X.

Caso ocorra algum erro de interpretação ou validação do comando a resposta consiste no envio de um "Nack" com a forma descrita anteriormente com IG definido por:

01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida.

- Teste de entradas analógicas:

06AIn(X)

O terminal responde a este comando enviando uma cadeia de caracteres com o seguinte formato:

06AIn(X)-(EEEE)-(CC.CC)*C

Com:

- X – Entrada analógica a ler – 1 a 8.
- EEEE – Valor hexadecimal presente na entrada X – 0000 a FFFF.
- CCCC – Valor de EEEE convertido em temperatura através da fórmula de conversão do sensor LM60 – Vo=(+6.25mV/ªC x T ªC)+424mV =>

T = (((EEEE*2.5/4096)-0.424)/6.25e-3) ºC

Caso ocorra algum erro de interpretação ou validação do comando a resposta consiste no envio de um "Nack" com a forma descrita anteriormente com IG definido por:

01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida.

- Teste dos sensores SHT75:

  06SHT75(III)(X)

  Com:

  - III – Tipo de leitura:

    Tmp – Leitura de temperatura.
    Hum – Leitura de humidade.

  - X – Sensor a ler – 1ou 2.

  O terminal responde a este comando enviando uma cadeia de caracteres com o seguinte formato:

  Se III=Tmp:

  06SHT75Tmp(X)-(EEEE)-(CC.CC)*C

  Se III=Hum:

  06SHT75Hum(X)-(EEEE)-(CC.CC)%

  Com:

  - EEEE – Valor hexadecimal da leitura – 0000 a FFFF.
  - CCCC – Valor de EEEE convertido em temperatura ou humidade através da fórmula de conversão do sensor SHT75:
    Se III = Tmp:

    $T = 0.01 * EEEE - 39.7\ ^{o}C$

    Se III = Hum:

    $RH = -2.0468 + 0.0367 * EEEE - 1.5955 * 10^{-6} * EEEE^{2}\ \%$

  Caso ocorra algum erro de interpretação ou validação do comando a resposta consiste no envio de um "Nack" com a forma descrita anteriormente com IG definido por:

  01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida ou endereço do sensor inválido.

02 – Não recebeu o "Ack" ao comando interno enviado para o sensor.

03 – Não ficou com a linha DATA num nível lógico alto enquanto faz a aquisição de dados no sensor (leitura do parâmetro indicado).

04 – Não ficou com a linha DATA num nível baixo no final da aquisição de dados no sensor.

- Teste de sensores TMP75:

06TMP75(X)

O terminal responde a este comando enviando uma cadeia de caracteres com o seguinte formato:

06TMP75(X)-(EEEE)-(CC.CC)*C

Com:

- X – Sensor a ler – 1 a 8.
- EEEE – Valor hexadecimal presente na entrada X – 0000 a FFFF.
- CCCC – Valor de EEEE convertido em temperatura através da fórmula de conversão do sensor TMP75

$$T = EEEE/4*0.25\ ^{o}C$$

Caso ocorra algum erro de interpretação ou validação do comando a resposta consiste no envio de um "Nack" com a forma descrita anteriormente com IG definido por:

01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida.

02 – Endereço de sensor inválido – 1 a 8.

03 – Não recebeu ack ao endereço enviado na fase de configuração do sensor.

04 – Não recebeu ack ao *Pointer Register* enviado na fase de configuração do sensor.

05 – Não recebeu um ack ao valor a aplicar ao registo de configurações (R0 e R1 a 1 para resolução de 12 *bit's*) na fase de configuração do sensor.

06 – Não recebeu ack ao endereço enviado na fase de reconfiguração do *pointer register* para *temperature register*.

07 – Não recebeu ack ao *pointer register* enviado na fase de reconfiguração do *pointer register* para *temperature register*.

08 – Não recebeu ack ao endereço enviado na fase de leitura da temperatura.

Comando 07

Este comando permite configurar as definições de rede pretendidas: endereço *IP*, Máscara, Porta de funcionamento, entre outros.

Deve existir cuidado na atribuição da porta a utilizar pois esta define não só a porta de comunicações de *TCP-IP* (P) mas também a porta utilizada para comunicações do tipo mestre-escravo em que a porta a utilizar como conversor *TCP-IP/RS-232* é P+1.

Caso não saibamos o endereço atual do terminal, podemos adquiri-lo utilizando o programa de atualização da aplicação "rabbitp" (apenas o endereço e nunca a porta). Este programa não é apresentado pois sai do âmbito deste projeto

Mesmo utilizando *DHCP*, o endereço e a máscara a utilizar como definições estáticas devem ser enviados no formato correto, Caso contrário o comando não é aceite. Apenas a *gateway* pode ser desativada (enviando apenas 0).

Uma vez que alguns campos deste comando poderão ter dimensão variável, é necessário enviar o delimitador de campo, constituído pelo caracter ';'.

O comando apresenta então as seguintes configurações:

- Escrita do comando:

  07W(D);(P);(I);(M);(G);

  O terminal responde a este comando com um "Ack" ou com um "Nack".

- Leitura dos valores configurados:

07R

O terminal responde a este comando enviando uma cadeia de caracteres com o seguinte formato:

07R(D);(P);(I);(M);(G);

- Leitura dos valores efetivamente ativos:

07C

O terminal responde a este comando enviando uma cadeia de caracteres com o seguinte formato:

07C(D);(P);(I);(M);(G);

Com:

- W – Escrita das configurações pretendidas
- R – Leitura das configurações pretendidas
- C – Leitura das configurações efetivas (Corrente)
- ; − Caracter delimitador de cadeia
- D – 0→Usar a configuração estática definida nos campos seguintes,
  1→Usar DHCP
- P – Porta a utilizar (por defeito é a 20000).
- I – Endereço *IP* no formato III.III.III.III.
- M – Máscara de rede no formato MMM.MMM.MMM.MMM.
- G – *Gateway* no formato GGG.GGG.GGG.GGG.

O "Ack" e o "Nack" deste comando assumem a seguinte estrutura:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efetuada sem erros.

01 – Cadeia de caracteres enviada com dimensão inferior ao esperado ou operação inválida.
02 – Porta a utilizar inválido.
03 – Endereço *IP* inválido.
04 – Mascara de rede inválida.
05 – Endereço *Gateway* inválido → caso seja este o erro apresentado, significa que todas as definições foram aceites e corretamente configuradas exceto o endereço de *Gateway* que é colocado como 0.0.0.0.

O valor corrente pode ser diferente do valor configurado (por exemplo se se estiver a utilizar um endereço *DHCP* diferente do endereço estático configurado).

Comando 08

Este comando permite ler ou configurar a descrição do terminal na rede.

A descrição do terminal na rede é um parâmetro que permite mais facilmente reconhecer um terminal. Esta descrição consiste numa cadeia de 35 caracteres que identificam de forma inequívoca o terminal.

Um exemplo de descrição válida é: "Sala de servidores 1".

Por defeito após uma formatação o valor desta descrição é colocado como "Sem descrição".

O comando tem as seguintes duas configurações possíveis:

- Escrita da descrição:

  08W(TTT)

  O terminal responde a este comando com um "Ack" ou com um "Nack".

- Leitura da descrição:

  08R

  A resposta a este comando consiste na seguinte cadeia de caracteres:

  08R(TTT)

Com:

- W – Escrita das configurações pretendidas
- R – Leitura das configurações pretendidas
- TTT – Texto com 35 caracteres no máximo.

O "Ack" e o "Nack" deste comando assumem a seguinte estrutura:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efetuada sem erros.

  01 – Operação inválida.

Comando 09

Este comando permite ler ou configurar o fator de *delay* de comunicações entre os sensores SHT75 e o dispositivo.

Durante a fase de testes constatou-se que a partir de uma dada distância, a comunicação entre os sensores SHT75 e o dispositivo degradava-se a tal ponto de se tornar impossível a correta leitura dos parâmetros.

Assim, e por forma a eliminar esse efeito, criou-se uma variável que comporta o fator de *delay* pretendido entre transmissão de *bit's* sucessivos.

Esse fator de *delay* é diretamente proporcional à distância e deve ser programado para que o *bit* a ser lido estabilize na linha, antes de ser adquirido pelos intervenientes da comunicação.

Por defeito, após uma formatação, o valor deste fator de *delay* é colocado com o valor 10 que representa um tempo entre *bit's* sucessivos de aproximadamente 10x10µs=100µs.

O comando tem as seguintes duas configurações possíveis:

- Escrita do novo fator de *delay*:

  09W(D)

O terminal responde a este comando com um "Ack" ou com um "Nack".

- Leitura do fator de *delay* configurado:

  09R

A resposta a este comando consiste na seguinte cadeia de caracteres:

08R(D)

Com:

- W – Escrita das configurações pretendidas
- R – Leitura das configurações pretendidas
- D – Fator de *delay* pretendido, que a multiplicar por 10µs dá o valor aproximado do *delay* de transmissão de *bit's* consecutivos. Esta variável pode assumir valores entre 1 e 15 e é enviado em formato hexadecimal (0x1..0xF).

O "Ack" e o "Nack" deste comando assumem a seguinte estrutura:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efetuada sem erros.
  01 – Operação inválida.
  02 – Valor enviado inválido (0x1..0xF).

Comando 10

Este comando permite ler o valor/estado de todas as entradas analógicas, digitais e sensores do terminal de uma só vez.

O comando tem a seguinte configuração:

- Leitura de todos os valores presentes nas entradas:

  10R

A resposta a este comando consiste na seguinte cadeia de caracteres:

10R(ED)(AIn1)(AIn2)(AIn3)(AIn4)(AIn5)(AIn6)(AIn7)(AIn8)(SHT1)(SHT2)(SHH1)(SHH2)(TMP1)(TMP2)(TMP3)(TMP4)(TMP5)(TMP6)(TMP7)(TMP8)

Com:

- ED – Estado de todas as entradas digitais:

  b0 – Entrada digital 1.

  ...

  b7 – Entrada digital 8.

  Este valor é enviado em dois dígitos hexadecimais, por exemplo o valor 0x21 indica que as entradas digitais 1 e 6 estão no valor lógico '1' e as restantes '0'
- AInx – Valor analógico entregue pelo sensor x. Este valor é enviado em hexadecimal a 32 *bit's*, por exemplo 0x03A5.
- SHTx – Valor analógico referente ao valor de temperatura lido pelo sensor SHT75x. Este valor é enviado em hexadecimal a 32 *bit's*
- SHHx – Valor analógico referente ao valor de humidade lido pelo sensor SHT75x. Este valor é enviado em hexadecimal a 32 *bit's*.
- TMPx – Valor analógico referente ao valor de temperatura lido pelo sensor TMP75x. Este valor é enviado em hexadecimal a 32 *bit's*.

Comando 11

Este comando permite ler o estado de todas as entradas digitais do terminal de uma só vez.

O comando tem a seguinte configuração:

- Leitura de todos os valores presentes nas entradas digitais:

  11R

A resposta a este comando consiste na seguinte cadeia de caracteres:

11R(ED)

Com:

- ED – Estado de todas as entradas digitais:

  b0 – Entrada digital 1.

  ...

  b7 – Entrada digital 8.

  Este valor é enviado em dois dígitos hexadecimais, por exemplo o valor 0x21 indica que as entradas digitais 1 e 6 estão no valor lógico '1' e as restantes '0'

Comando 12

Este comando permite ler o valor/estado de todas as entradas analógicas.

O comando tem a seguinte configuração:

- Leitura de todos os valores presentes nas entradas analógicas:

  12R

A resposta a este comando consiste na seguinte cadeia de caracteres:

12R(AIn1)(AIn2)(AIn3)(AIn4)(AIn5)(AIn6)(AIn7)(AIn8)

Com:

- AInx – Valor analógico entregue pelo sensor x. Este valor é enviado em hexadecimal a 32 *bit's*, por exemplo 0x03A5. Se não existir um sensor presente a entrada deve ser ligada ao *GND* para que o valor apresentado seja 0x0000, caso contrário o valor poderá representar o erro inerente ao processo de conversão analógico/digital.

Comando 13

Este comando permite ler os valores de temperatura e humidade medidas pelos dois sensores SHT75 de uma só vez.

O comando tem a seguinte configuração:

- Leitura de todos os valores presentes nas entradas:

  13R

A resposta a este comando consiste na seguinte cadeia de caracteres:

13R(SHT1)(SHT2)(SHH1)(SHH2)

Com:

- SHTx – Valor analógico referente ao valor de temperatura lido pelo sensor SHT75x. Este valor é enviado em hexadecimal a 32 *bit's*
- SHHx – Valor analógico referente ao valor de humidade lido pelo sensor SHT75x. Este valor é enviado em hexadecimal a 32 *bit's*.

Nota: Caso algum dos sensores não esteja conectado, o valor transmitido será de 0x0000 tanto para a humidade como para a temperatura.

Comando 14

Este comando permite ler os valores de temperatura medidos pelos oito sensores TMP75 de uma só vez.

O comando tem a seguinte configuração:

- Leitura de todos os valores presentes nas entradas:

  14R

A resposta a este comando consiste na seguinte cadeia de caracteres:

14R(TMP1)(TMP2)(TMP3)(TMP4)(TMP5)(TMP6)(TMP7)(TMP8)

Com:

- TMPx – Valor analógico referente ao valor de temperatura lido pelo sensor TMP75x. Este valor é enviado em hexadecimal a 32 *bit's*.

Nota: Caso algum dos sensores não esteja conectado, o valor transmitido será de 0x0000.

Comando 21

Este comando permite ler ou escrever as configurações de alarme de cada uma das entradas digitais.

Para uma melhor compreensão do funcionamento deste comando recomenda-se o estudo do anexo 7 – Modo de configuração e funcionamento dos alarmes.

O comando tem a seguinte configuração:

- Escrita das configurações de alarme:

  21W(E)(N)(MM)(TT)

  O terminal responde a este comando com um "Ack" ou com um "Nack".

- Leitura das configurações de alarme:

  21R(E)

  A resposta a este comando consiste na seguinte cadeia de caracteres:

  21R(E)(N)(MM)(TT)

  Com:

  - W – Escrita das configurações pretendidas
  - R – Leitura das configurações pretendidas
  - E – Número da entrada digital a ler ou configurar (1 a 8).
  - N – Nível de disparo do alarme (0 ou 1). Se o nível de disparo for 0 a linha deverá estar normalmente num nível lógico alto e sempre que desce ao nível lógico baixo origina uma ordem de alarme.
  - MM – Máscara de saídas a atuar em caso de alarme em hexadecimal. Por exemplo se este valor for 0x11 as saídas 5 e 1 serão ativadas sempre que ocorrer um alarme e durante o tempo estipulado.
    Para desativar qualquer alarme, este valor deverá ser colocado com o valor 0x00.
  - TT – Tempo de ativação do alarme em segundos (valor hexadecimal). Se este valor for por exemplo 0x0A, o alarme estará ativo durante 10 segundos. Caso este valor seja enviado a zero (0x00) o alarme

estará ativo sempre, e enquanto a entrada N esteja no nível lógico de alarme.

"Ack" e o "Nack" deste comando assumem a seguinte estrutura:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efetuada sem erros.
  01 – Operação inválida.
  02 – Número de entrada digital inválido – 1 a 8.
  03 – Nível de disparo do alarme – 0 ou 1.
  04 – Máscara de alarmes inválida – 0x00 a 0xFF.
  05 – Tempo de alarme inválido – 0x00 a 0xFF.

Comando 22

Este comando permite ler ou escrever as configurações de alarme de cada uma das entradas analógicas.

Para uma melhor compreensão do funcionamento deste comando recomenda-se o estudo do anexo 7 – Modo de configuração e funcionamento dos alarmes.

O comando tem a seguinte configuração:

- Escrita das configurações de alarme:

  22W(E)(VAIF)(VHIF)(MI)(TI) (VASP)(VHSP)(MS)(TS)

  O terminal responde a este comando com um "Ack" ou com um "Nack".

- Leitura das configurações de alarme:

  22R(E)

  A resposta a este comando consiste na seguinte cadeia de caracteres:

22R(E)(VAIF)(VHIF)(MI)(TI) (VASP)(VHSP)(MS)(TS)

Com:

- W – Escrita das configurações pretendidas
- R – Leitura das configurações pretendidas
- E – Número da entrada analógica a ler ou configurar (1 a 8).
- VAIF – Valor hexadecimal representante da referência de nível inferior a que o alarme deve disparar. Por exemplo 0x03A5.
- VHIF – Valor hexadecimal representante da histerese de desativação a que o valor de alarme inferior deve ser desativado. Por exemplo 0x03AF.
- MI – Máscara de saídas a atuar, em valor hexadecimal, em caso de alarme inferior. Por exemplo se este valor for 0x11 as saídas 5 e 1 serão ativadas sempre que ocorrer um alarme e durante o tempo estipulado.

  Para desativar qualquer alarme, este valor deverá ser colocado com o valor 0x00.
- TI – Tempo de ativação do alarme em segundos (valor hexadecimal). Se este valor for por exemplo 0x0A, o alarme estará ativo durante 10 segundos. Caso este valor seja enviado a zero (0x00) o alarme estará ativo sempre que a entrada N passe para baixo do valor de referência, e enquanto esteja abaixo do valor de histerese de alarme.
- VASP – Valor hexadecimal representante da referência de nível superior a que o alarme deve disparar. Por exemplo 0x04A5.
- VHSP – Valor hexadecimal representante da histerese de desativação a que o valor de alarme superior deve ser desativado. Por exemplo 0x0400.
- MS – Máscara de saídas a atuar, em valor hexadecimal, em caso de alarme superior. Por exemplo se este valor for 0x11 as saídas 5 e 1 serão ativadas sempre que ocorrer um alarme e durante o tempo estipulado.

Para desativar qualquer alarme, este valor deverá ser colocado com o valor 0x00.

- TS – Tempo de ativação do alarme em segundos (valor hexadecimal). Se este valor for por exemplo 0x0A, o alarme estará ativo durante 10 segundos. Caso este valor seja enviado a zero (0x00) o alarme estará ativo sempre que a entrada N passe para cima do valor de referência, e enquanto esteja acima do valor de histerese de alarme.

O "Ack" e o "Nack" deste comando assumem a seguinte estrutura:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efetuada sem erros.
  01 – Operação inválida.
  02 – Número de entrada digital inválido – 1 a 8.
  03 – Valor de referência do alarme inferior inválido.
  04 – Valor de desativação (histerese) de alarme inferior inválido.
  05 – Máscara de saídas a ativar em caso de alarme inferior inválida – 0x00 a 0xFF.
  06 – Tempo de ativação de alarme inferior inválido – 0x00 a 0xFF.
  07 – Valor de referência do alarme superior inválido.
  08 – Valor de desativação (histerese) de alarme superior inválido.
  09 – Máscara de saídas a ativar em caso de alarme superior inválida – 0x00 a 0xFF.
  0A – Tempo de ativação de alarme superior inválido – 0x00 a 0xFF.
  0B – O valor de referência da ocorrência de alarme inferior é menor que o valor de desativação do mesmo alarme (histerese).

0C – O valor de referência da ocorrência de alarme superior é maior que o valor de desativação do mesmo alarme (histerese).

0D – O valor de referência da ocorrência de alarme superior é igual ou menor que o valor de referência da ocorrência de alarme inferior.

Comando 23

Este comando permite ler ou escrever as configurações de alarme de cada um dos sensores SHT75.

Para uma melhor compreensão do funcionamento deste comando recomenda-se o estudo do anexo 7 – Modo de configuração e funcionamento dos alarmes.

O comando tem a seguinte configuração:

- Escrita das configurações de alarme:

  23W(E)(T)(VAIF)(VHIF)(MI)(TI) (VASP)(VHSP)(MS)(TS)

  O terminal responde a este comando com um "Ack" ou com um "Nack".

- Leitura das configurações de alarme:

  23R(E)(T)

  A resposta a este comando consiste na seguinte cadeia de caracteres:

  23R(E)(T)(VAIF)(VHIF)(MI)(TI) (VASP)(VHSP)(MS)(TS)

  Com:

  - W – Escrita das configurações pretendidas
  - R – Leitura das configurações pretendidas
  - E – Número da entrada analógica a ler ou configurar (1 ou 2).
  - T – Tipo de leitura: T=temperatura, H=humidade.
  - VAIF – Valor hexadecimal representante da referência de nível inferior a que o alarme deve disparar. Por exemplo 0x03A5.

Nota: Para a temperatura este valor está compreendido entre 0x0000 e 0x3FFF e para a humidade este valor está compreendido entre 0x0038 e 0x0CA4.

- VHIF – Valor hexadecimal representante da histerese de desativação a que o valor de alarme inferior deve ser desativado. Por exemplo 0x03AF.

  Nota: Para a temperatura este valor está compreendido entre 0x0000 e 0x3FFF e para a humidade este valor está compreendido entre 0x0038 e 0x0CA4.

- MI – Máscara de saídas a atuar, em valor hexadecimal, em caso de alarme inferior. Por exemplo se este valor for 0x11 as saídas 5 e 1 serão ativadas sempre que ocorrer um alarme e durante o tempo estipulado.

  Para desativar qualquer alarme, este valor deverá ser colocado com o valor 0x00.

- TI – Tempo de ativação do alarme em segundos (valor hexadecimal). Se este valor for por exemplo 0x0A, o alarme estará ativo durante 10 segundos. Caso este valor seja enviado a zero (0x00) o alarme estará ativo sempre que a entrada N passe para baixo do valor de referência, e enquanto esteja abaixo do valor de histerese de alarme.

- VASP – Valor hexadecimal representante da referência de nível superior a que o alarme deve disparar. Por exemplo 0x04A5.

  Nota: Para a temperatura este valor está compreendido entre 0x0000 e 0x3FFF e para a humidade este valor está compreendido entre 0x0038 e 0x0CA4.

- VHSP – Valor hexadecimal representante da histerese de desativação a que o valor de alarme superior deve ser desativado. Por exemplo 0x0400.

  Nota: Para a temperatura este valor está compreendido entre 0x0000 e 0x3FFF e para a humidade este valor está compreendido entre 0x0038 e 0x0CA4.

- MS – Máscara de saídas a atuar, em valor hexadecimal, em caso de alarme superior. Por exemplo se este valor for 0x11 as saídas 5 e 1 serão ativadas sempre que ocorrer um alarme e durante o tempo estipulado.

  Para desativar qualquer alarme, este valor deverá ser colocado com o valor 0x00.
- TS – Tempo de ativação do alarme em segundos (valor hexadecimal). Se este valor for por exemplo 0x0A, o alarme estará ativo durante 10 segundos. Caso este valor seja enviado a zero (0x00) o alarme estará ativo sempre que a entrada N passe para cima do valor de referência, e enquanto esteja acima do valor de histerese de alarme.

O "Ack" e o "Nack" deste comando assumem a seguinte estrutura:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.
- IG – Informação geral.

  00 – Operação efetuada sem erros.
  01 – Operação inválida.
  02 – Número de entrada digital inválido – 1 ou 2.
  03 – Tipo de leitura errado: H=humidade, T=temperatura.
  04 – Valor de referência do alarme inferior inválido.
  05 – Valor de desativação (histerese) de alarme inferior inválido.
  06 – Máscara de saídas a ativar em caso de alarme inferior inválida – 0x00 a 0xFF.
  07 – Tempo de ativação de alarme inferior inválido – 0x00 a 0xFF.
  08 – Valor de referência do alarme superior inválido.
  09 – Valor de desativação (histerese) de alarme superior inválido.
  0A – Máscara de saídas a ativar em caso de alarme superior inválida – 0x00 a 0xFF.

0B – Tempo de ativação de alarme superior inválido – 0x00 a 0xFF.

0C – O valor de referência da ocorrência de alarme inferior é menor que o valor de desativação do mesmo alarme (histerese).

0D – O valor de referência da ocorrência de alarme superior é maior que o valor de desativação do mesmo alarme (histerese).

0E – O valor de referência da ocorrência de alarme superior é igual ou menor que o valor de referência da ocorrência de alarme inferior.

Comando 24

Este comando permite ler ou escrever as configurações de alarme de cada um dos sensores TMP75.

Para uma melhor compreensão do funcionamento deste comando recomenda-se o estudo do anexo 7 – Modo de configuração e funcionamento dos alarmes.

O comando tem a seguinte configuração:

- Escrita das configurações de alarme:

  24W(E)(VAIF)(VHIF)(MI)(TI) (VASP)(VHSP)(MS)(TS)

  O terminal responde a este comando com um "Ack" ou com um "Nack".

- Leitura das configurações de alarme:

  24R(E)

  A resposta a este comando consiste na seguinte cadeia de caracteres:

  24R(E)(VAIF)(VHIF)(MI)(TI) (VASP)(VHSP)(MS)(TS)

  Com:

  - W – Escrita das configurações pretendidas
  - R – Leitura das configurações pretendidas
  - E – Número da entrada analógica a ler ou configurar (1 a 8).

- VAIF – Valor hexadecimal representante da referência de nível inferior a que o alarme deve disparar. Por exemplo 0x03A5.
- VHIF – Valor hexadecimal representante da histerese de desativação a que o valor de alarme inferior deve ser desativado. Por exemplo 0x03AF.
- MI – Máscara de saídas a atuar, em valor hexadecimal, em caso de alarme inferior. Por exemplo se este valor for 0x11 as saídas 5 e 1 serão ativadas sempre que ocorrer um alarme e durante o tempo estipulado.

  Para desativar qualquer alarme, este valor deverá ser colocado com o valor 0x00.
- TI – Tempo de ativação do alarme em segundos (valor hexadecimal). Se este valor for por exemplo 0x0A, o alarme estará ativo durante 10 segundos. Caso este valor seja enviado a zero (0x00) o alarme estará ativo sempre que a entrada N passe para baixo do valor de referência, e enquanto esteja abaixo do valor de histerese de alarme.
- VASP – Valor hexadecimal representante da referência de nível superior a que o alarme deve disparar. Por exemplo 0x04A5.
- VHSP – Valor hexadecimal representante da histerese de desativação a que o valor de alarme superior deve ser desativado. Por exemplo 0x0400.
- MS – Máscara de saídas a atuar, em valor hexadecimal, em caso de alarme superior. Por exemplo se este valor for 0x11 as saídas 5 e 1 serão ativadas sempre que ocorrer um alarme e durante o tempo estipulado.

  Para desativar qualquer alarme, este valor deverá ser colocado com o valor 0x00.
- TS – Tempo de ativação do alarme em segundos (valor hexadecimal). Se este valor for por exemplo 0x0A, o alarme estará ativo durante 10 segundos. Caso este valor seja enviado a zero (0x00) o alarme

estará ativo sempre que a entrada N passe para cima do valor de referência, e enquanto esteja acima do valor de histerese de alarme.

O "Ack" e o "Nack" deste comando assumem a seguinte estrutura:

(ADD)X(IG)

Com:

- X – A – Aceite sem erros

  N – Negado ou parcialmente aceite.

- IG – Informação geral.

  00 – Operação efetuada sem erros.
  01 – Operação inválida.
  02 – Número de entrada digital inválido – 1 a 8.
  03 – Valor de referência do alarme inferior inválido.
  04 – Valor de desativação (histerese) de alarme inferior inválido.
  05 – Máscara de saídas a ativar em caso de alarme inferior inválida – 0x00 a 0xFF.
  06 – Tempo de ativação de alarme inferior inválido – 0x00 a 0xFF.
  07 – Valor de referência do alarme superior inválido.
  08 – Valor de desativação (histerese) de alarme superior inválido.
  09 – Máscara de saídas a ativar em caso de alarme superior inválida – 0x00 a 0xFF.
  0A – Tempo de ativação de alarme superior inválido – 0x00 a 0xFF.
  0B – O valor de referência da ocorrência de alarme inferior é menor que o valor de desativação do mesmo alarme (histerese).
  0C – O valor de referência da ocorrência de alarme superior é maior que o valor de desativação do mesmo alarme (histerese).

0D – O valor de referência da ocorrência de alarme superior é igual ou menor que o valor de referência da ocorrência de alarme inferior.

Uma vez que este sensor apenas admite temperaturas ente -40℃ a 125℃ os valores de VAIF, VHIF, VASP e VHSP apenas pode tomar valores nos seguintes intervalos:

0xD80 ---------- 0xFFF : (de -40℃ a -0.06℃)
0X000 ---------- 0x7D0 : (de 0℃ a 125℃)

Comando 30

Este comando permite ler o estado atual de todas as saídas de uma só vez.

Este comando lê em que estado as saídas deverão estar, podendo no entanto mostrar valores que não correspondem à verdade caso alguma saída esteja queimada, ou algum dispositivo externo esteja a obrigar a saída a tomar um valor contrário ao indicado pelo processador.

O comando tem a seguinte configuração:

- Leitura de todos os valores presentes nas entradas:

30R

A resposta a este comando consiste na seguinte cadeia de caracteres:

30R(SS)

Com:

- SS – Estado de todas as saídas digitais:

b0 – Saída digital 1.
...
b7 – Saída digital 8.

Este valor é enviado em dois dígitos hexadecimais, por exemplo o valor 0x21 indica que as saídas digitais 1 e 6 estão no valor lógico ‘1’ e as restantes ‘0’

Anexo 7 – Modo de configuração e funcionamento dos alarmes.

Na figura seguinte podemos observar o conceito básico de funcionamento de qualquer sinal de alarme:

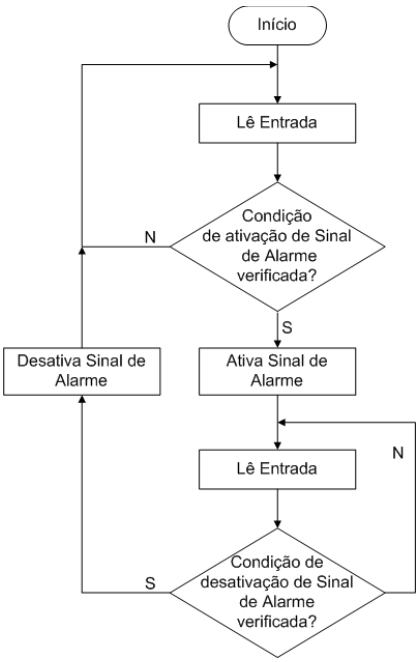


Figura 156: Conceito básico do modo de funcionamento associado aos sinais de alarme.

Como se depreende da figura anterior, a ativação e desativação de qualquer alarme depende da ocorrência um evento externo associado, que pode ser diferente para cada caso.

As secções seguintes descrevem em mais pormenor este conceito de sinalização de eventos ocorridos por meio de sinais de alarme, que eventos podem ser associados à sua ativação e desativação desses sinais, e seus métodos de configuração.

Modo de funcionamento dos alarmes inerentes às Entradas Digitais.

Cada entrada digital pode estar configurada com uma condição de alarme de forma que, sempre que essa condição se verifique, o dispositivo reaja de determinada maneira.

Para tal são guardados no dispositivo alguns parâmetros inerentes ao funcionamento dos alarmes:

N - Valor lógico para disparo do alarme.

MM – Máscara de saídas a ativar sempre que o alarme esteja ativo.

TT – Tempo de ativação do alarme.

Estes parâmetros são definidos de forma independente para cada entrada, de maneira a que cada entrada tenha o seu modo próprio de funcionamento que pode ser diferente de entrada para entrada.

Temos então que:

- Sempre que a máscara de saídas MM esteja definida com um valor superior a 0x00 a entrada está habilitada a gerar alarmes, caso contrário não existe nenhuma condição de alarme associada a essa entrada.
- Um alarme é gerado sempre que o nível lógico da entrada iguale o valor lógico de disparo N.
- Sempre que um alarme é acionado as saídas definidas em MM são ativadas durante o tempo de alarme TT.
- Caso o tempo de alarme TT esteja definido com o valor 0x00 as saídas ativadas por um alarme assumirão o nível lógico alto enquanto a entrada estiver na condição de alarme (valor lógico de alarme N).
- Uma vez que duas condições de alarme definidas por entradas diferentes podem estar associadas á mesma saída, em caso de alarmes simultâneos, apenas a condição de alarme referente à entrada com maior prioridade será atendida (ver secção – prioridades das entradas nos alarmes neste anexo).

Exemplos:

- Parâmetros de alarme para a entrada digital 1 definidos como:

  N = 0;
  MM = 0x00;
  TT = 0x05;

Como MM está definida com o valor 0x00, a entrada não está a ser monitorizada e os alarmes referentes a esta entrada estão inativos.

- Parâmetros de alarme para a entrada digital 2 definidos como:

  N = 0;

  MM = 0x1A;

  TT = 0x05;

A entrada digital 2 deve estar normalmente no nível lógico alto.

Logo que esta entrada assuma o nível lógico baixo, as saídas 2, 4 e 5 (MM=0b00011010) ficam ativas durante 5 segundos (TT=0x05).

- Parâmetros de alarme para a entrada digital 3 definidos como:

  N = 1;

  MM = 0x02;

  TT = 0x00;

A entrada digital 3 deve estar normalmente no nível lógico baixo.

Logo que esta entrada assuma o nível lógico alto, a saída 2 (MM=0b00000010) fica ativa durante 5 segundos (TT=0x05).

Como a saída 2 tinha sido definida como saída de alarme referenciada pela entrada digital 2 no passo anterior, se ocorrer um alarme simultâneo, o alarme que ganha o controlo da saída é o alarme referente à saída 2 porque é mais prioritária. Esse alarme ocorrerá durante 5 segundos (TT da entrada digital 2) e a saída 2 será desativada ao fim desse tempo, para logo de seguida ser novamente ativada para atender ao alarme referente à entrada digital 3, se o valor lógico desta entrada se mantiver ainda no nível de alarme N. Caso o valor lógico da entrada 3 volte ao nível de referência antes de passarem os 5 segundos referentes ao tempo de alarme ativo da entrada 2, o alarme referente à entrada 3 ocorre sem que nunca se dê por isso e não ficará registado em lado algum.

Modo de funcionamento dos alarmes inerentes às Entradas Analógicas e Sensores SHT75 e TMP75.

O funcionamento dos alarmes inerentes às entradas analógicas e sensores SHT75 e TMP75 é muito semelhante ao funcionamento dos alarmes inerentes às entradas digitais

divergindo apenas em alguns pontos, derivado das características dos valores controlados por este tipo de entradas.

Assim, em vez de apenas um alarme, estas entradas podem ter associados dois alarmes diferentes, um de limite superior e outro de limite inferior que podem ser configurados de diversas maneiras por forma a melhor satisfazer as necessidades.

Nos parâmetros configuráveis relativos aos alarmes relacionados com as entradas analógicas ou sensores temos:

NSUP – Nível superior, em hexadecimal, a partir do qual o alarme é acionado.

HSUP – Nível superior, em hexadecimal, de desativação do alarme.

MS – Máscara de saídas a ativar sempre que o alarme superior esteja ativo.

TS – Tempo de ativação do alarme superior.

NINF – Nível inferior, em hexadecimal, a partir do qual o alarme é acionado.

HINF – Nível inferior, em hexadecimal, de desativação do alarme.

MI – Máscara de saídas a ativar sempre que o alarme inferior esteja ativo.

TI – Tempo de ativação do alarme inferior.

Estes parâmetros são definidos de forma independente para cada entrada, de maneira a que cada entrada tenha o seu modo próprio de funcionamento, que pode ser diferente de entrada para entrada.

Temos então que:

- Sempre que a máscara de saídas MS esteja definida com um valor superior a 0x00 a entrada está habilitada a gerar alarmes de nível superior, caso contrário não existe nenhuma condição de alarme de nível superior associada a essa entrada.
- Um alarme superior é gerado sempre que o nível da entrada iguale ou supere o nível de disparo de alarme superior NSUP.
- O alarme só é rearmado quando o nível da entrada atinja o valor de desativação do alarme (HSUP) criando um efeito de histerese. Isto significa

que quando o alarme é disparado só volta a poder existir outro alarme quando a entrada descer do nível HSUP.

- Sempre que um alarme superior é acionado as saídas definidas em MS são ativadas durante o tempo de alarme TT.
- Caso o tempo de alarme TT esteja definido com o valor 0x00 as saídas ativadas por um alarme assumirão o nível lógico alto enquanto a entrada estiver na condição de alarme (o nível HSUP não tiver sido atingido).
- Uma vez que duas condições de alarme definidas por entradas diferentes podem estar associadas à mesma saída, em caso de alarmes simultâneos, apenas a condição de alarme referente à entrada com maior prioridade será atendida (ver secção – prioridades das entradas nos alarmes neste anexo).
- Estas condições também são válidas para o caso dos alarmes inferiores.

Resumindo: sempre que o valor de NSUP seja atingido, e caso a entrada esteja habilitada a monitorizar alarmes de nível superior máximo (MS>0x00), um alarme é gerado e todas as saídas indicadas em MS ficarão ativas no nível lógico alto durante TT segundos. O alarme é assim satisfeito, e enquanto o nível da entrada não voltar abaixo do nível HSUP não serão gerados mais alarmes de nível superior. Caso TT seja 0x00, as saídas definidas em MS ficarão ativas no nível lógico alto durante todo o tempo até que a condição de nível de entrada lido seja inferior ao nível de desativação de alarme superior HSUP.

As configurações relativas ao alarme inferior são igual às referidas para o alarme superior.

Os parâmetros destes alarmes têm que respeitar as seguintes condições para que tudo funcione corretamente:

- NSUP > NINF
- NSUP > HSUP
- NINF < HINF

Exemplos:

- Parâmetros de alarme para a entrada analógica 1 definidos como:

NSUP = 0x03AF

HSUP = 0x03A1

MS = 0x00

TS = 0x05

NINF = 0x02A5

HINF = 0x02E1

MI = 0x01

TI = 0x05

Como MS está definida com o valor 0x00, a entrada não está a ser monitorizada e os alarmes superiores referentes a esta entrada analógica estão inativos.

Como MI está definida com um valor superior a 0x00, a entrada está a ser monitorizada e sempre que o seu valor seja igual ou inferior a 0x02A5, um alarme é disparado e a saída 1 (MI=0x01) ficará ativa durante 5 segundos (TI=0x05). Após os 5 segundos a saída 1 é desativada, e apenas será novamente ativada quando o nível da entrada analógica descer abaixo de 0x02E1 (HINF) e volte novamente a subir acima de 0x02A5 (Caso esta saída esteja associada a outros alarmes poderá ser ativada por eles sem se verificarem estas condições).

- Parâmetros de alarme para a entrada analógica 2 definidos como:

NSUP = 0x03AF

HSUP = 0x03A1

MS = 0x05

TS = 0x00

NINF = 0x02A5

HINF = 0x02E1

MI = 0x01

TI = 0x05

Como MS está definida com o valor superior a 0x00, a entrada está a ser monitorizada e podem ocorrer alarmes superiores referentes a esta entrada analógica. Assim, sempre que o valor na entrada iguale ou suba acima de 0x03AF (NSUP) ocorre um alarme superior e as saídas 1 e 3 (MS=0b00000101) serão ativadas. Como TT=0x00, as

saídas estarão ativas até que o valor à entrada analógica desça abaixo do valor 0x03A1 (HSUP).

No caso do alarme inferior, o funcionamento é igual ao descrito no exemplo anterior.

Prioridade das entradas em caso de alarmes simultâneos para a mesma saída.

Como existe a possibilidade de definir a mesma saída para satisfazer alarmes diferentes, existe a necessidade de criar um mecanismo de prioridades caso esses alarmes ocorram em simultâneo.

Nenhum alarme é perdido devido a este mecanismo, fica é mascarado debaixo de outro mais prioritário e pode nunca ser percetível.

Imagine-se que dois alarmes disparam simultaneamente e que ambos têm associada a mesma saída. Essa saída é ativada pelo alarme mais prioritário e permanecerá ativa enquanto este assim o definir. Logo que o alarme mais prioritário tenha as suas condições satisfeitas liberta a saída e, caso o alarme menos prioritário ainda não tenha satisfeito as suas condições também, volta a ativar. Este processo não é percetível e por isso há a possibilidade de algumas condições de alarme não serem ocultadas por este processo. Se existir alguma condição de alarme extremamente importante, deverá ser alocada uma saída apenas para esse alarme por forma a certificarmo-nos que este alarme é percetível.

Um alarme apenas é perdido caso ocorra numa fração de tempo muito curto. Uma vez que a leitura do estado das entradas é refrescada a intervalos de 5 segundos, se alguma entrada assumir um nível acima do nível de alarme entre 2 leituras consecutivas, esta condição de alarme será ignorada.

Prioridades dos alarmes por ordem decrescente:

Entrada digital 1 – Mais prioritária

Entrada digital 2

Entrada digital 3

Entrada digital 4

Entrada digital 5

Entrada digital 6

Entrada digital 7

Entrada digital 8

Entrada analógica 1

Entrada analógica 2

Entrada analógica 3

Entrada analógica 4

Entrada analógica 5

Entrada analógica 6

Entrada analógica 7

Entrada analógica 8

Sensor TMP75 1

Sensor TMP75 2

Sensor TMP75 3

Sensor TMP75 4

Sensor TMP75 5

Sensor TMP75 6

Sensor TMP75 7

Sensor TMP75 8

Sensor SHT75 Temperatura 1

Sensor SHT75 Temperatura 2

Sensor SHT75 Humidade 1

Sensor SHT75 Humidade 2 – Menos prioritária

Anexo 8 – Livrarias de código do programa Monitorizador RenPAD.

(Dada a extensão do texto associado a este anexo, o mesmo apenas se encontra disponível para consulta no cd-rom que acompanha o relatório, na pasta "Anexos/Código RenPAD" e é constituído por um ficheiro por livraria com o nome correspondente.)

Anexo 9 – Datasheets e manuais de utilizador diversos.

(Dada a quantidade de documentos associados a este anexo, os mesmos apenas se encontram disponíveis para consulta no cd-rom que acompanha o relatório, na pasta "Anexos/Datasheets" e é constituído por um ficheiro por *datasheet* ou manual com o nome correspondente.)