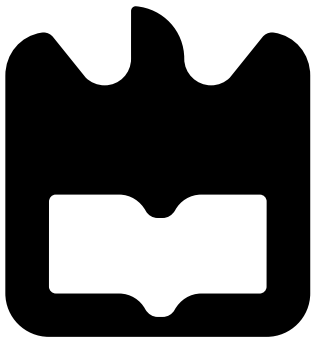
Universidade de Aveiro
Departamento de Electrónica, Telecomunicações e Informática,
2013

Ivo Alexandre Costa Alves Angélico

Demonstrador para *Switch Ethernet* Tempo-Real



Ivo Alexandre Costa Alves Angélico

Demonstrador para *Switch Ethernet* Tempo-Real

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessàrios à obtenção do grau de Mestre em Engenharia Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Paulo Pedreiras do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e do Professor Doutor Joaquim Ferreira da Escola Superior de Tecnologia e Gestão de Águeda.

Trabalho financiado por Fundos Nacionais através da "FCT - Fundação para a Ciência e a Tecnologia" no âmbito do projecto "Serv-CPS" PTDC/EAA-AUT/122362/2010

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA EDUCAÇÃO E CIÊNCIA

o júri / the jury

Presidente / President	**Professor Doutor Alexandre Manuel Moutela Nunes da Mota** Professor Associado da Universidade de Aveiro
Arguente Principal / Main Examiner	**Professor Doutor Luis Miguel Pinho de Almeida** Professor Associado da Faculdade de Engenharia da Universidade do Porto
Orientador / Advisor	**Professor Doutor Paulo Bacelar Reis Pedreiras** Professor Auxiliar da Universidade de Aveiro
Co-Orientador / Co-Advisor	**Professor Doutor Joaquim José de Castro Ferreira** Professor Adjunto da Escola Superior de Tecnologia e Gestão de Àgueda

Agradecimentos

Durante a preparação desta dissertação estiveram envolvidas directa e indirectamente diversas pessoas. A todas elas expresso a minha gratidão e agradecimento. No entanto, por esse envolvimento ter sido maior, agradeço em particular:

a Paulo Pedreiras, não só pelo carácter humano que proporcionou um excelente ambiente de trabalho, como também pela exemplar orientação científica e técnica que demonstrou no decorrer do tempo.

a Joaquim Ferreira pela sua orientação científica e técnica sempre com empenho e dedicação.

aos meus orientadores gostaria de agradecer também a oportunidade que me deram de realizar esta dissertação.

a Luís Silva pela dedicação e empenho que demonstrou no projecto. Ao Luís gostaria de agradecer em especial a sempre prestável ajuda que nunca negou e a disponibilidade incondicional. Agradeço o seu contributo e ajuda na preparação do sistema de comunicação e o facto de ter estado presente na batalha para resolver os muitos desafios que foram aparecendo. De referir também a sua excelente qualidade como ser humano o que ajudou também a que este trabalho decorresse sempre com grande alegria e descontração.

a Milton Cunguara pelas suas observações sempre pertinentes relativamente ao sistema de controlo.

aos meus pais e restante família todo o apoio e motivação que me deram durante o meu percurso académico. Obrigado por estarem sempre disponíveis quando precisava e nunca faltarem com nada.

à minha namorada pelo carinho e amizade, pela força e compreensão.

a todos aqueles com quem me cruzei ao longo deste percurso e que de uma forma ou outra me tocaram. João Miranda, César Gomes, Paulo Simões, Francisco Martins, André Barata, Fábio Rico, entre outros, tornaram o meu percurso académico mais harmonioso e enriquecedor. Tenho sorte em poder hoje contar com estas pessoas como amigos.

a todos os que estiveram envolvidos nesta dissertação

Obrigado

Resumo

O uso de sistemas embutidos distribuídos há muito que está generalizado, podendo os mesmos ser encontrados em àreas como a aviónica, robótica e automação industrial. Um sistema embutido distribuído implica que vários nós do sistema troquem informação entre si de forma a atingir um objectivo comum. Para efectuar a troca de informação entre os nós é normalmente usada uma rede de comunicação.

A introdução desta rede nos sistemas embutidos distribuídos afecta o seu comportamento, pois, estes sistemas são normalmente sensíveis à latência e ao *jitter*. De forma a resolver estes problemas foram criados protocolos específicos com propriedades de tempo-real.

No seio do projecto de investigação Serv-CPS está a ser desenvolvida uma arquitectura de rede baseada na tecnologia *Ethernet* comutada, com suporte explícito e eficiente a metodologias de gestão de tráfego orientadas a componentes.

O trabalho a desenvolver nesta dissertação prevê, para além da identificação de possíveis sistemas a adoptar que possibilitem a demonstração do correcto funcionamento dos sistemas de comunicação, a implementação de um desses sistemas de forma a poder avaliar o desempenho da arquitectura de rede desenvolvida.

Abstract

The use of distributed embedded systems have long been widespread, and they may be found in areas such as avionics, robotics and industrial automation. A distributed embedded system implies that multiple nodes of the system exchange information with each other in order to achieve a common goal. To make the exchange of information between nodes it is usually used a communications network.

The use of this network in distributed embedded systems affects their behavior, because these systems are usually sensitive to latency and jitter. In order to solve these problems there were created specific protocols with real-time properties.

Within the research project Serv-CPS, it is currently being developed a network architecture based on switched Ethernet technology, with explicit support and efficient traffic management methodologies oriented to components.

The work in this dissertation provides, in addition to identifying possible systems to adopt that enable the demonstration of the correct behaviour of the communication systems, the implementation of such a system in order to be able to evaluate the performance of the network architecture developed.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Acrónimos

AVB Audio Video Bridging.

CBA Component based automation.

CIP Common Industrial Protocol.

COTS Commercial Off-The-Shelf.

EC Elementary Cycle.

EtherNet/IP Ethernet Industrial Protocol.

FTT Flexible Time Triggered.

FTT-SE Flexible Time Triggered comunication over Switched Ethernet.

HaRTES Hard Real Time Ethernet Switching.

IRT tempo real isócrono.

LAN redes locais.

M2M machine-to-machine.

OSI Open Systems Interconnection.

PROFINET Process Field Network.

QoS Qualidade de Serviço.

RT tempo real.

SW switched Ethernet.

TM Trigger Message.

TTEthernet Time Triggered Ethernet.

Capítulo 1

Introdução

Esta dissertação foi desenvolvida no âmbito do Projecto Serv-CPS: Arquitectura de comunicação *Ethernet* baseada em servidores para sistemas ciber-físicos. Para esse projecto está a ser desenvolvida uma arquitectura de rede baseada na tecnologia *Ethernet* comutada, para sistemas ciber-físicos, com suporte explícito e eficiente a metodologias de gestão de tráfego orientadas a componentes. Como forma de confirmar os resultados do projecto e de disseminar os mesmos serão desenvolvidos demonstradores que consistem num conjunto de sistemas de controlo distribuído que utilizarão o sistema de comunicação *Ethernet* desenvolvido.

1.1 Motivação

O uso de sistemas embutidos nos diversos tipos de aplicações e equipamentos do quotidiano, há muito que está generalizado como o prova o artigo de 1999 do *The New York Times*, *Honey, I Programmed the Blanket* [3]. Ao contrário dos sistemas de uso geral, os sistemas embutidos são sistemas especializados que realizam uma tarefa sendo eficientes em termos de tamanho e consumo de potência, apresentando um baixo custo de desenvolvimento e produção. Avanços na electrónica e a queda dos custos das tecnologias a si associadas, tem permitido aumentar a capacidade de computação e permitiu a integração de comunicação.

A evolução dos sistemas embutidos, em conjunto com as vantagens que advêm da modularidade (facilidade de instalação e manutenção, baixo custo de instalação, maior tolerância a falhas) levaram ao desenvolvimento de aplicações embutidas distribuídas. Os sistemas embutidos distribuídos integram um conjunto de nós que cooperam para um objectivo comum, sendo utilizados em várias indústrias, desde a automóvel à aeroespacial, passando pelo controlo de processo e indústrias de manufatura. Os sistemas embutidos distribuídos apresentam restrições temporais rigorosas. A resposta temporal de um sistema distribuído completo depende de vários elementos, incluindo a capacidade do sistema de comunicação.

O protocolo de comunicação *Ethernet* tem levantado um crescente interesse para ser utilizado como interligação entre sistemas embutidos pois, para além de ter uma boa largura de banda e baixo custo de instalação e manutenção, pode ser facilmente integrado com a Internet. Desta forma é possível ligar os escritórios de empresas às plantas de produção onde se encontram os sistemas embutidos[4, 5].

Dado que as interacções entre os nós do sistemas embutidos distribuídos são constantes. a comunicação entre elas deve ser eficiente, segura e garantida. Os sistemas *Ethernet* tradicionais não conseguem responder adequadamente dado o indeterminismo temporal que o seu

funcionamento apresenta. Para além de restrições temporais, estes sistemas incluem diversos tipos de tráfego (periódico, esporádio ou aperiódico) que não são explicitamente suportados pela *Ethernet standard*. Têm sido desenvolvidas várias soluções de *Ethernet* de tempo real mas estas ainda falham em conseguir associar eficientemente aos requisitos de tempo real, os requisitos de Qualidade de Serviço (QoS) dos diferentes tipos de tráfego.

No âmbito do projecto Serv-CPS está a ser desenvolvido um sistema de comunicação *Ethernet* de tempo real baseado no *switch* desenvolvido no projecto Hard Real Time Ethernet Switching (HaRTES) que gere eficientemente a largura de banda mantendo as garantias de cumprimento do tempo real e permitindo escalonamento de tráfego *online*, isto é, executando o algoritmo de escalonamento durante a operação do sistema computacional.

Para testar as capacidades do sistema de comunicação do projecto serão identificados possíveis demonstradores e posteriormente implementado um deles que consistirá num um sistema distribuído utilizando 3 eixos. Com este módulo será gerado tráfego periódico, esporádico e aperiódico, com requisitos de tempo real, sensível ao *jitter* e que necessita de uma baixa latência pois será usado para fechar a malha de controlo. Tendo sido o módulo devidamente implementado e controlado, será possível realizar testes que comprovem o devido funcionamento do sistema de comunicação. O demonstrador será um instrumento importante para teste e validação dos resultados do projecto pois o facto de mostrar de uma forma física, visual e quantificável, o correcto funcionamento do sistema de comunicação tem um impacto maior do que meras simulações de tráfego de dados.

1.2 Objectivos

O trabalho a desenvolver nesta dissertação prevê a identificação de possíveis sistemas a adoptar que possibilitem a demonstração do correcto funcionamento dos sistemas de comunicação e a implementação de um desses sistemas. Este sistema deve ser altamente modular de forma a permitir uma fácil preparação e adaptação aos futuros testes dos sistemas de comunicação. Para atingir este objectivo ter-se-á que percorrer os seguintes passos:

- familiarização com os conceitos de sistema de controlo, sistema de tempo real e de sistema de comunicação *Ethernet* de tempo real;
- identificar possíveis sistemas demonstradores;
- projectar e desenvolver a interface electrónica do demonstrador selecionado;
- desenvolver *device drivers*;
- desenvolver software de controlo;
- desenvolver aplicações de demonstração;
- testes e validações.

1.3 Estrutura do documento

Esta dissertação está dividida em 5 capítulos:

- introdução: é apresentada a motivação para esta dissertação;

- conceitos fundamentais: são explicados os conceitos fundamentais, a terminologia utilizada e são analisados os protocolos de comunicação e o *switch* HaRTES;
- equipamentos para o demonstrador: especificação das características necessárias ao demonstrador e são analisadas as possíveis implementações para o demonstrador; é introduzido o módulo mecânico, é apresentado o sistema electrónico e de controlo desenvolvido;
- testes e resultados: é testado e validado o funcionamento do sistema implementado sendo apresentados os resultados;
- conclusões: é avaliado o trabalho desenvolvido e são feitas considerações no que diz respeito a trabalho futuro.

Capítulo 2

Conceitos fundamentais

Este capítulo introduz os conceitos fundamentais relativos aos sitemas de controlo, aos sistemas ciber-físicos e sistemas de comunicação *Ethernet* de tempo real.

2.1 Sistemas de Controlo

Na actividade quotidiana deparamo-nos frequentemente com sistemas de controlo. No simples exemplo de andar de bicicleta facilmente se constata que este envolve o movimento, de um modo coordenado, de várias partes do nosso corpo, como por exemplo, os braços e as pernas. Este movimento é comandado pelo cérebro em função de informações enviadas pelos diferentes sentidos que possuímos.

Um sistema de controlo pode definir-se então como um sistema em que se manipula um elemento tendo em vista atingir, se possível, um efeito desejado [6].

O controlo dos sistemas é feito introduzindo elementos no sistemas que vão permitir a alteração das características de forma a alcançar o comportamento desejado.

2.1.1 Elementos de um Sistema de Controlo

Um sistema de controlo possui geralmente os seguintes elementos (figura 2.1) [7]:

- Referência: sinal de referência do sistema;
- Peturbação: sinal de interferência do sistema que provoca um comportamento indesejado, tipicamente devido a ruído ou interferência;
- Erro: sinal resultante da comparação da referência com a realimentação;
- Controlador: componente responsável pelo controlo do sistema;
- Actuador: componente responsável pela transformação do sinal recebido do controlador num sinal de entrada do sistema físico;
- Sensor: componente responsável pela transformação do sinal recebido da saída do sistema físico num sinal de entrada do controlador.

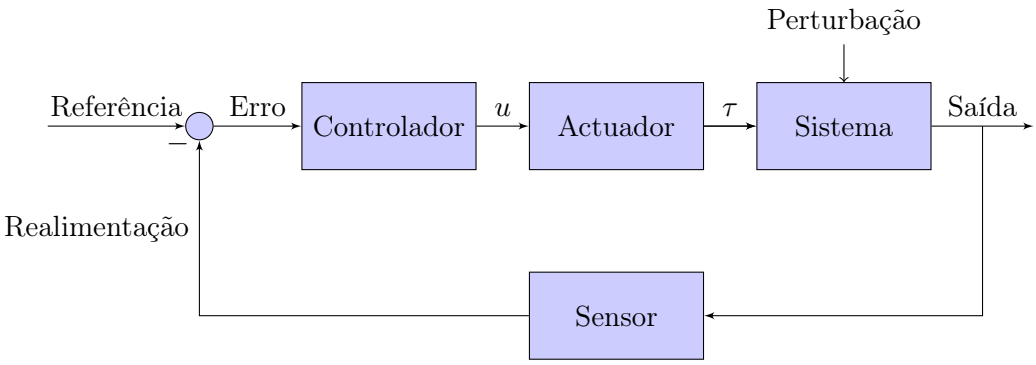


Figura 2.1: Modelo de um sistema de controlo

2.1.2 Formas de Controlo

Um sistema de controlo pode ser controlado em malha aberta ou em malha fechada.

Controlo em malha aberta

Designa-se por sistema de controlo em malha aberta (figura 2.2) ou não realimentado um sistema cuja entrada (referência) não é afectada pela saída [6]. Por não se observar a saída do sistema, neste tipo de controlo não se tem a capacidade de corrigir possíveis erros de funcionamento, pelo que é necessário que se conheça muito bem o sistema que se quer controlar bem como os efeitos que o controlador tem sobre o sistema.

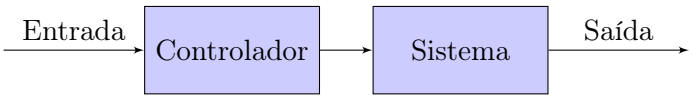


Figura 2.2: Sistema de controlo em malha aberta

Controlo em malha fechada

Designa-se por sistema de controlo em malha fechada (figura 2.3) o sistema em que uma função do sinal de saída, ou este, é adicionada algebricamente ao sinal de referência [6]. Normalmente o sinal de realimentação é subtraído ao sinal de referência originando o sinal de erro (realimentação negativa), podendo no entanto ser somado (realimentação positiva). Dada a operação anterior, qualquer alteração do sinal existente ao longo da cadeia constituída pelo controlador e sistema a controlar, resultante ou da alteração dos parâmetros do sistema ou de perturbações, será introduzida, com sinal contrário (se realimentação negativa), na entrada do controlador originando a sua atenuação na saída.

2.1.3 Compensador Proporcional, Integral e Derivativo

O sistema a implementar no âmbito desta dissertação será controlado em malha fechada. Os controladores a utilizar serão controladores proporcionais, integrais e derivativos (figura

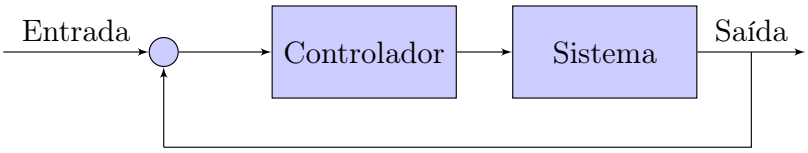


Figura 2.3: Sistema de controlo em malha fechada

2.4). Nestes controladores a saída resulta da soma de três componentes: a proporcional, que é proporcional ao erro; a integrativa, que é proporcional ao integral dos erros durante o tempo de funcionamento do sistema; e uma componente derivativa proporcional à variação do erro.

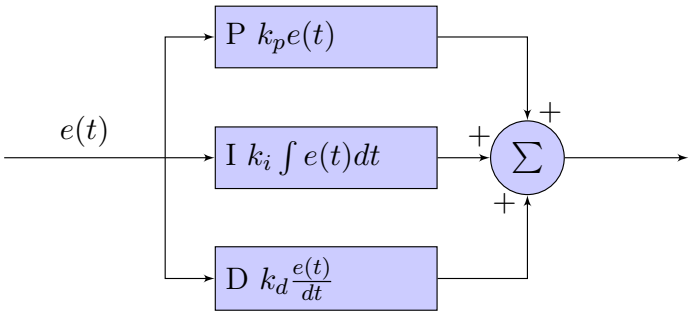


Figura 2.4: Controlador PID

Dado que o sistema implementado é discreto no tempo, a equação 2.1 representa o PID.

$$u_n = k_p * e(n) + k_i * \sum_{k=0}^{n} e(k) + k_d * (e(n) - (e(n-1)) \tag{2.1}$$

2.2 Sistemas ciber-físicos

Os sistemas ciber-físicos são sistemas onde a teoria do controlo, comunicações e de computação de tempo real são combinadas em aplicações embutidas que interagem com o mundo físico (figura 2.5) [8]. Desta forma ir-se-á apresentar de seguida os conceitos fundamentais relativamente aos sistemas de tempo real e aos sistemas embutidos. Dado que os módulos do demonstrador são distribuídos serão também apresentados os conceitos relativos aos sistemas distribuídos.

2.2.1 Sistemas de tempo real

Um sistema de tempo real descreve um processo computacional que tem que responder a eventos internos e externos num determinado período de tempo [9]. Desta forma os resultados das computações deverão estar logicamente correctos e ser produzidos a tempo. O instante final em que um resultado deve estar pronto é chamado *deadline*. Os sistemas de tempo real podem ser classificado em relação ao *deadline* da seguinte forma:

- *Soft*: Se o resultado da computação continua a ser útil mesmo ultrapassado o *deadline* (embora haja deterioração da qualidade de serviço);

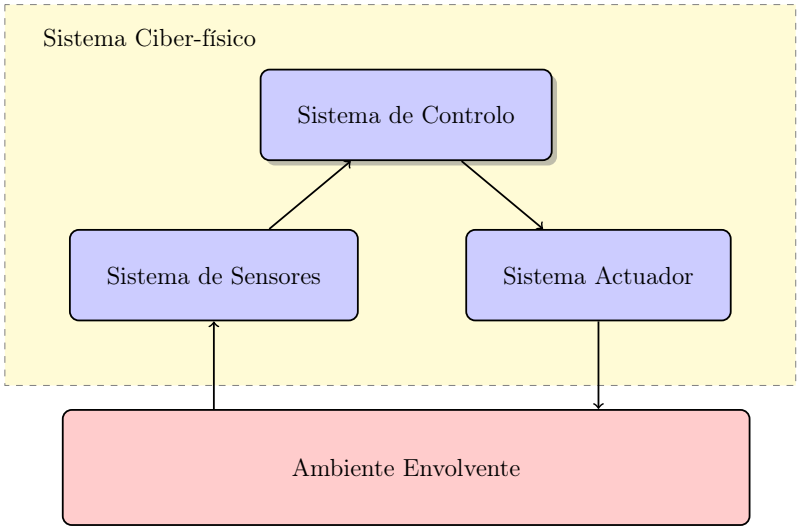


Figura 2.5: Sistema Ciber-físico

- *Firm*: Se o resultado da computação for inútil após ultrapassado o *deadline*;
- *Hard*: Se o não cumprimento do *deadline* puder levar a resultados catastróficos.

Os requisitos impostos aos sistemas de tempo real são:

- funcionais;
- temporais;
- de dependibilidade.

Os requisitos funcionais dos sistemas de tempo real relacionam-se com as funções que o sistema deve realizar, podendo agrupar-se em recolha de dados (amostragem das variáveis do sistema), controlo digital directo (acesso do controlador aos sensores e atuadores) e interacção com o operador (informações do estado do sistema e suporte à operação do sistema). Os requisitos temporais são resultado do sistema de tempo real ao qual é necessário responder. Estes requisitos impõem intervalos máximos ao tempo morto entre o instante em que um acontecimento ocorre e o instante em que o mesmo é observado pelo sistema e ao tempo de processamento entre o instante em que o acontecimento é observado e o instante em que uma resposta é executada. Os requisitos temporais impõe também restrições à variação dos tempos anteriores (*jitter*). O *jitter* é especialmente indesejado em sistemas *time-triggered* pois aumentam significativamente os tempos de resposta total do sistema de tempo real [10]. Os sistemas de tempo real são *time-triggered* se a execução das suas actividades for disparada por um sinal de controlo baseado na progressão de tempo e são *event-triggered* se a execução das actividades for disparada por um evento significativo sentido pelo sistema. Os requisitos de dependibilidade implicam que o sistema de tempo real se mantenha em funcionamento correcto mesmo que ocorram erros ou falhas. Os sistemas de tempo real com esta caracteristíca são então adequados a aplicações críticas.

Os sistemas de tempo real podem também ser caracterizados através da regularidade de eventos:

- periódicos: eventos ocorrem regularmente, a intervalos de tempo específicos;
- esporádicos: eventos não são periódicos, mas há um tempo mínimo entre eventos consecutivos;
- aperiódicos: não é possível prever o instante em que os eventos vão ocorrer, nem existe qualquer restrição à ocorrencia dos mesmos.

2.2.2 Sistemas Embutidos e Distribuídos

Um sistema embutido é um sistema baseado em microprocessador projectado para a execução de uma determinada função ou funções, ou seja tem um uso específico não sendo de uso geral [11]. Tipicamente faz parte de um sistema ou produto maior. Sendo sistemas tipicamente especializados tipicamente são projectados para serem eficientes em termos de tamanho e consumo de energia, apresentando um baixo custo de desenvolvimento e produção.

Muitas aplicações embutidas críticas são implementadas usando uma arquitectura distribuída, ou seja, a funcionalidade do sistema é distribuída por vários nós que estão ligados por um sistema de comunicação.

Um sistema distribuído é constituído por um conjunto de unidades de processamento independentes que cooperam na realização de uma dada tarefa, sendo vistos pelo utilizador como sendo um sistema coerente e único [12]. Os sistemas distribuídos, se devidamente projectados, apresentam as seguintes características vantajosas:

- dependabilidade, isto é, contenção de erros nos nós;
- composabilidade, ou seja, o sistema é composto integrando subsistemas;
- escabilidade, dada a facilidade de adição de novos nós.

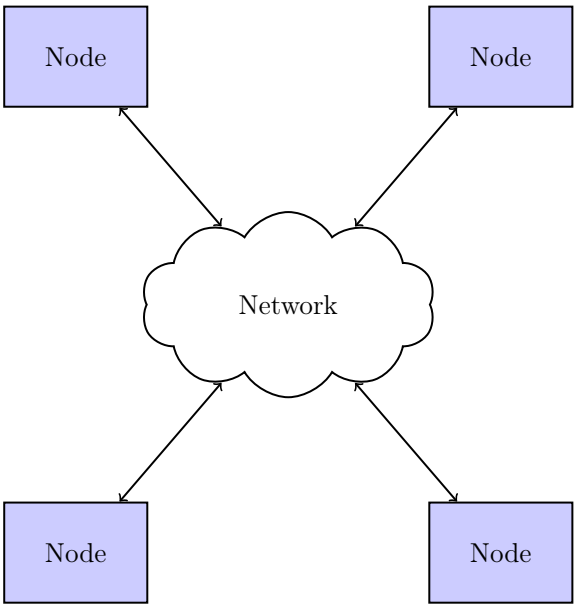


Figura 2.6: Sistema distribuído

Os sistemas embutidos distribuídos apresentam a vantagem de estarem mais próximos da fonte de informação o que permite a simplificação dos cabos. Apresentam também facilidades de manutenção pois é fácil substituir os nós.

2.3 Ethernet de tempo real

A *Ethernet* é a tecnologia de comunicação mais utilizada para redes de telecomunicações, sendo que mais de 95 por cento das redes locais (LAN) são baseadas na *Ethernet* [13].

A tecnologia *Ethernet* foi criada nos anos 70 como uma forma de transferir informação entre um computador e impressoras [14]. Ao longo do tempo houveram evoluções e modificações de forma a aumentar a largura de banda, robustez e fiabilidade. As primeiras topologias de rede usadas eram em anel ou em barramento. Estas topologias apresentavam a grande desvantagem de, caso houvesse uma avaria na cablagem ou fosse necessário adicionar ou remover um nó, toda a rede poder ser afectada, ou ficar parte dela desconectada. Começou então a ser utilizada a topologia de estrela. Nesta topologia os nós vão-se ligar a um elemento central, ficando simplificada a adição/remoção de nós e evitando que avarias num cabo afectem vários nós. Inicialmente este elemento central era um repetidor (*hub*) que enviava a informação que recebesse numa porta para todas as outras. A utilização deste elemento tem a desvantagem de manter um único domínio de colisão. Apesar de existirem mecanismos de resolução de colisões, estes não são determinísticos e portanto não é possível a determinação de limites absolutos para os tempos de transmissão de mensagens. Passou então a utilizar-se um elementro central que trabalha no segundo nível do modelo de referência Open Systems Interconnection (OSI) e que permitia criar múltiplos domínios de colisão, chamado *switch* (comutador).

Os sistemas embutidos distribuídos podem gerar vários tipos de tráfego: periódico, aperiódico ou esporádico. Para além disso, o tipo de tráfego gerado por estes sistemas não é estático, podendo variar ao longo do tempo. O sistema de comunicação utilizado para ligar os sistemas embutidos distribuídos deve então ser capaz de efectuar *online* a mudança dos requisitos de comunicação sem colocar em risco as exigências temporais do sistema. A combinação desta flexibilidade com garantias temporais requer a existência de um controlo de admissão adequado, inexistente nos *switches* Commercial Off-The-Shelf (COTS) [15].

Ir-se-á de seguida descrever de forma breve alguns dos protocolos *Ethernet* tempo real que tentam ultrapassar os problemas anteriores. Uma descrição mais aprofundada poderá ser encontrada no anexo A. O protocolo Flexible Time Triggered comunication over Switched Ethernet (FTT-SE) será analisado com mais detalhe pois foi utilizado no projecto HaRTES.

2.3.1 EtherNet/IP

Ethernet Industrial Protocol (EtherNet/IP) é um protocolo de comunicação para utilização em aplicações de automação industrial que consigam suportar alguma quantidade de não determinismo [16]. É membro da família de redes que implementa o Common Industrial Protocol (CIP) nas suas camadas superiores. Na fase de instalação da rede, cada nó *Ethernet* é definido como sendo de um tipo específico previamente definido. O EtherNet/IP usa as mesmas tecnologias que a *Ethernet* tradicional sendo portanto compatível com os equipamentos *standard*.

2.3.2 AVB

As redes Audio Video Bridging (AVB) consistem num conjunto de protocolos desenvolvidos pelo grupo de trabalho de *Bridging Audio/Video* IEEE 802.1, com o objectivo de providenciar as especificações que vão permitir fluxos de audio e vídeo com baixa latência sincronizados no tempo em redes 802 [17]. Para realizar a sincronização, dispositivos AVB trocam mensagens com informação de tempo, o que permite aos nós sincronizar de forma precisa o relógio de referência. Isto permite a sincronização de múltiplos fluxos e ainda a existência de tempos comuns de amostragem/recepção de informação. A transmissão de tramas é controlada por um sistema modulador baseado em créditos, o que vai limitar a largura de banda e evitar rajadas de informação que poderiam levar à perda de informação. É realizado controlo de admissão, sendo que um novo fluxo só é aceite na rede depois de serem reservados os recursos necessários ao mesmo.

2.3.3 PROFINET

Process Field Network (PROFINET) é um protocolo de *Ethernet* desenvolvido para automação industrial. Este protocolo usa *switched ethernet* com as vantagens apresentadas anteriormente, permitindo transmissão de informação determinística e isocronamente [18]. O PROFINET apresenta uma estrutura modular composta por duas perspectivas chamadas PROFINET Component based automation (CBA) e PROFINET IO. O PROFINET CBA é utilizado para comunicações machine-to-machine (M2M) baseada em componentes utilizando TCP/IP. O PROFINET IO é utilizado para troca de informação entre arquitecturas distribuídas estando dividido em dois modos: comunicações de tempo real (RT) e de tempo real isócrono (IRT), sendo que o modo RT é utilizado por comunicação de tempo real com requisitos *soft*, e o modo IRT por comunicações de tempo real com requisitos *hard*. Cada ciclo de comunicação engloba um período de comunicação isócrono seguido por um período em que o canal fica aberto para o restante tráfego.

2.3.4 TTEthernet

Time Triggered Ethernet (TTEthernet) é um protocolo switched Ethernet (SW) de tempo real usado em aplicações de segurança em transportes ou em automação industrial, integrável com os equipamentos de *Ethernet* tradicionais [19]. Este protocolo foi desenvolvido de forma a permitir a existência de comunicações *time-triggered* em *ethernet*. Estas comunicações coexistem com o restante tráfego. Isto é feito com recurso a um controlador das comunicações *time-triggered* que é capaz de sincronizar o tráfego com os restantes controladores e *switches*. Este controlador envia mensagens em instantes determinados através da sincronização com o restante tráfego. Neste protocolo o tráfego *time-triggered* é tratado como tendo requisitos de tempo real *hard* tendo precedência sobre todos os outros tipos de mensagens. Tráfego *event-triggered* é tratado como não crítico.

2.3.5 FTT-SE

O protocolo FTT-SE foi projectado para suportar aplicações de tempo real adaptáveis e reconfiguráveis. Seguindo o paradigma Flexible Time Triggered (FTT), tem as principais vantagens do mesmo: coordenação do tráfego global na mesma linha temporal, a possibilidade de actualizar rápida e atomicamente as características dos fluxos, a capacidade de suportar

um largo conjunto de diferentes períodos dos fluxos e a capacidade de gerir tráfego periódico, aperiódico e esporádico com isolamento temporal [20].

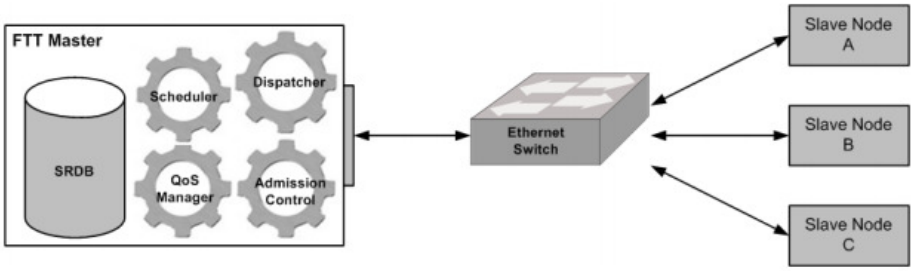


Figura 2.7: Arquitectura do sistema

O protocolo FTT-SE utiliza controlo de admissão *master/multislave*, utilizando uma arquitectura centralizada de agendamento, o que permite o controlo global do tráfego na rede evitando que os *buffers* de entrada dos *switches* fiquem sobrecarregados mantendo assim a previsibilidade do sistema. O *master* é o responsável pelo agendamento do tráfego dos *slaves*. São aproveitadas as características das redes comutadas para adaptar os requisitos de comunicação e o agendamento do tráfego individualmente a cada nó, permitindo desta formas adaptações *online* da largura de banda atribuída.

O protocolo funciona em ciclos (Elementary Cycle (EC)) que são iniciados pelo envio de uma mensagem de início por parte do *master*(Trigger Message (TM)). Cada EC é dividido em duas partes, uma para a troca de mensagens síncronas e outra para troca de mensagens assíncronas.

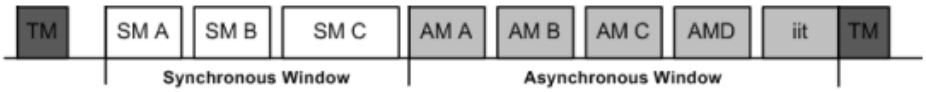


Figura 2.8: Ciclo Elementar

O agendamento do tráfego sincrono é feito pelo *master* e disseminado para os nós *slave* através da TM. Os nós *slave* recebem a TM, analisam-na e emitem imediatamente a seguir as mensagens agendadas. O agendamento feito pelo *master* tenta aproveitar ao máximo o paralelismo de caminhos disjuntos de informação (figura 2.9).

Para a transmissão do tráfego assíncrono, existe nos nós *slaves* filas onde este tráfego é armazenado. No final de cada EC, os nós devem transmitir ao *master* o estado destas filas, indo o *master* proceder ao agendamento deste tráfego. Este procedimento impede que tráfego assíncrono vá interferir com tráfego síncrono ou com as TM.

Mais informações relativamente a este protocolo poderão ser encontradas no Anexo A.

2.4 HaRTES

HaRTES é uma arquitectura num *switch* adaptado para comunicações de tempo real. Este *switch* resolve algumas das limitações do protocolo FTT-SE. O protocolo necessita que

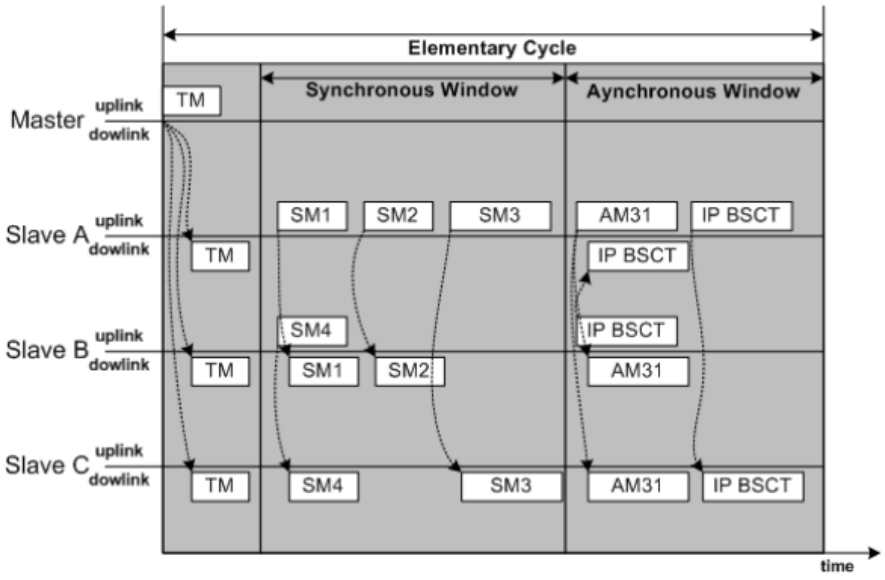


Figura 2.9: Tráfego de mensagens

todos os nós respeitem os EC e as suas restrições temporais. O *switch* HaRTES aumenta a integridade do sistema de comunicação ao garantir que comunicações com comportamentos anómalos são bloqueadas nas portas do *switch*. Para além disso, o nó *master* da rede está inserido neste *switch*, não tendo as mensagens deste nó que passar pelas camadas do protocolo FTT-SE permitindo uma maior precisão na transmissão das mensagens pois as mesmas são transmitidas directamente do *switch* para a rede. Esta melhoria no *jitter* e na latência é importante em aplicações que necessitem de uma elevada sincronização da rede. O *switch* HaRTES evita ainda que o tráfego das comunicações assíncronas tenha que ser chamado *a posteriori*, guardando-o na memória e transmitindo-o quando apropriado.

Juntando este *switch* com o protocolo FTT-SE analisado anteriomente, consegue-se [21]:

- introduzir capacidades de controlo de transmissão nos *switches ethernet* de forma a permitir a sincronização de fluxos e tansmissão com baixo *jitter*.
- introduzir escalonamento flexível e serviços de gestão da QoS para permitir mudanças *online* mantendo os requisitos temporais;
- implementar gestão de tráfego para separar diferentes classes de tráfego e geri-los com isolamento, permitindo desta forma nós com *ethernet* tradicional.

Para atingir as melhorias referidas, é utilizada no *switch* uma arquitectura baseada em servidores que vai lidar com a janela assíncrona de tráfego. Esta arquitectura forma uma hierarquia que pode ser representada como uma àrvore invertida de servidores e fluxos de dados, que permitem a divisão e subdivisão da largura de banda das comunicações assíncronas de cada porto. Desta forma são criados diferentes canais virtuais que separam o tráfego garantindo os requisitos de tempo real e tendo isolamento temporal.

Na figura 2.10 é apresentada a arquitectura interna do *switch* HaRTES.

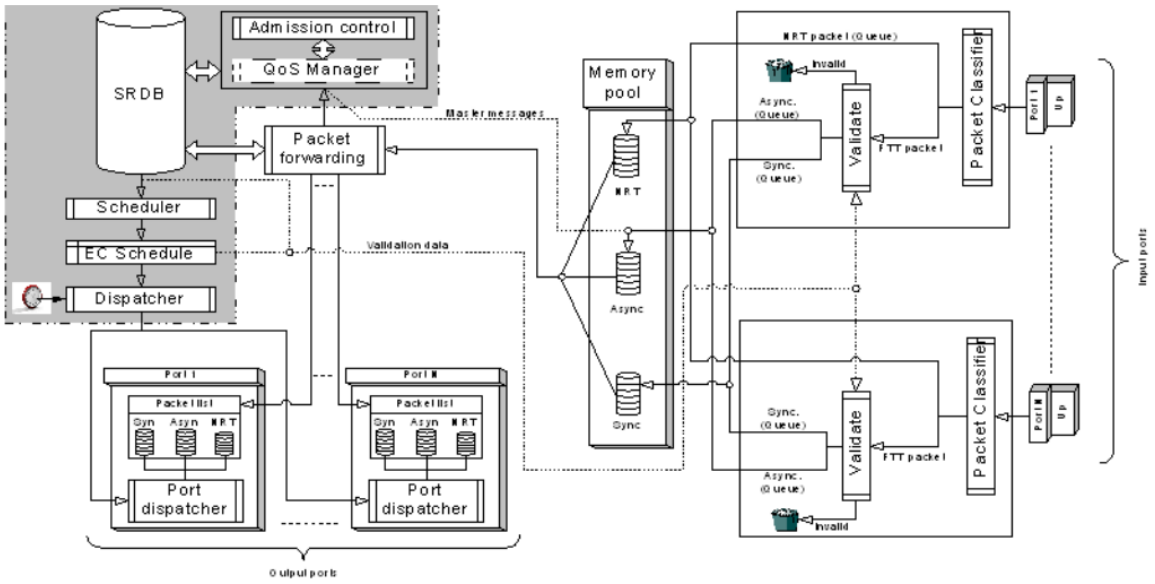


Figura 2.10: Arquitectura do *switch* HaRTES

O módulo *master* corresponde ao *master* FTT que é reponsável pelo controlo de admissão e agendamento do tráfego. O módulo *Switching and Control Logic* gere a recepção e envio de pacotes de dados. O módulo *MAC interface* é responsável pela recepção dos dados e seu armazenamento no módulo *Memory Pool*. Permite também controlar o *MAC IP Core*. O módulo *Memory Pool* serve para armazenar dados. Os módulos *Rx Multiplexing Unit* e *Tx Multiplexing Unit* permitem a leitura ou escrita na memória. A unidade *MAC IP Core* disponibiliza os relógios com a frequência necessária para a recepção e transmissão de dados através do PHY Ethernet.

Capítulo 3

Equipamentos para o demonstrador

Os demonstradores são importantes como forma de mostrar que a investigação realizada vem responder a necessidades reais da indústria, para além de confirmarem os resultados do projecto e permitirem a disseminação dos mesmos. Uma componente importante destes demonstradores é permitirem verificar de forma física, visual e quantificável o correcto funcionamento dos sistemas de comunicação do projecto.

Dada a especificidade do projecto, optou-se por procurar estruturas mecânicas podendo a electrónica e software necessários ser desenvolvidos pelos elementos do projecto.

3.1 Critérios de escolha

Para a escolha dos equipamentos a usar como demonstradores tiveram-se em conta os seguintes critérios:

- **Critérios obrigatórios para todos os equipamentos:**
  - *Sistema Ciber-Físico* - Este trabalho visa demonstrar uma arquitectura de comunicação para este tipo de sistemas, pelo que os equipamentos devem ser sistemas ciber-físicos;
  - *Sistema de Tempo Real* - Os equipamentos devem ter requisítos temporais rigorosos e exigentes de forma a conseguir testar a capacidade do sistema de comunicação;
  - *Arquitectura Aberta* - Os equipamentos devem ter uma arquitectura aberta de forma a poderem ser facilmente adaptados;
- **Critérios externos que condicionam os equipamentos:**
  - *Preço* - O projecto está condicionado a um financiamento que é limitado;
  - *Tamanho* - O demonstrador terá que ser transportado pelo que deve ser compacto;
  - *Simplicidade* - Os equipamentos devem ser suficientemente simples para permitir a finalização do sistema final num período de tempo que é limitado.
- **Critérios opcionais:**
  - *Tráfego de vídeo* - Desta forma o próprio demonstrador geraria tráfego que necessita de elevada largura de banda, ajudando a colocar o *switch* a testar sob *stress*;

– *Integração com Rede Ethernet Tradicional* - Demonstrando desta forma uma das mais valias deste projecto.

3.2 Soluções

Foi realizada uma pesquisa exaustiva das soluções existentes no mercado, tanto soluções industriais como soluções para educação. Foi feita também uma pesquisa das soluções existentes na universidade que possam ser adaptadas. Dessa pesquisa foram selecionados um conjunto de equipamentos que eram mais adequados. De seguida é feita uma pequena introdução a cada um deles.

Braço Robótico

Está disponível para uso no projecto um braço robótico com 5 graus de liberdade. Esta é uma solução de robótica bastante robusta podendo ter várias aplicações. Os actuadores desta solução são motores DC que têm integrados encoders. Dado que o braço robótico já se encontra disponível, esta solução não implicaria custos significativos. No entanto, esta solução implicaria o desenvolvimento, entre outras coisas, de algoritmos de controlo e de planeamento de trajectória, tarefas que demorariam mais tempo que o disponível para a realização desta dissertação.

Sistema de controlo de fluxo e de nível

O Sistema de controlo de fluxo e de nível da *Feedback Instruments* (figura 3.1) é um sistema que combina o controlo do nível e fluxo hidráulico com controlo de temperatura [22]. Apesar de interagir intimamente com o meio ambiente, os requisítos temporais deste módulo são demasiado lentos para o pretendido.

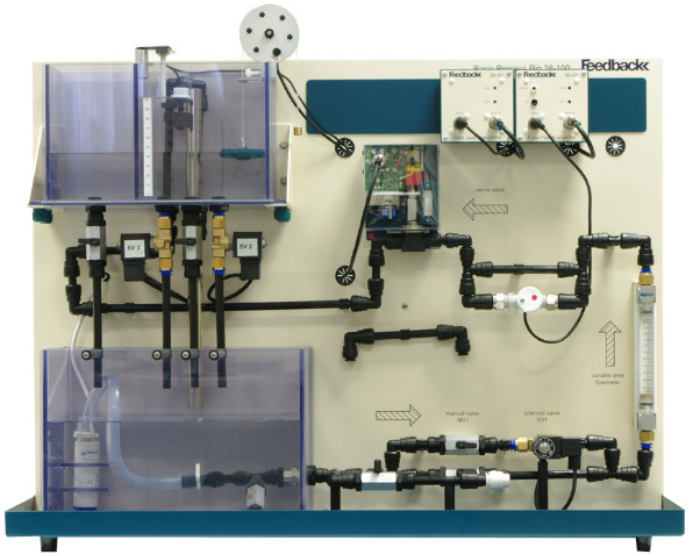


Figura 3.1: Sistema de controlo de fluxo e de nível

Sistema de controlo de plano e bola

O Sistema de controlo de plano e bola da *Feedback Instruments* (figura 3.2) permite o controlo da posição de uma bola num plano [23]. A malha de controlo é fechada através de um sinal de vídeo. O controlo do plano é realizado através de indução magnética. Esta solução tem uma arquitectura fechada que impede fácil adaptação, não tendo a flexibilidade necessária para a sua adaptação.

Figura 3.2: Sistema de controlo de plano e bola

Bola no plano

Sistema de controlo que permite o controlo de uma bola num plano. A malha de controlo é fechada através de uma superfície táctil. O controlo do plano é realizado através de dois servos. A arquitectura aberta deste módulo permite a sua fácil adaptação.

Pêndulo invertido auto erigido

O Pêndulo invertido auto erigido da *Quanser* (figura 3.3)consiste num motor DC que se desloca ao longo de um carril [24]. Este é um sistema modular o que permite facilmente ser adaptado. O preço é no entanto demasiado elevado para o orçamento disponível.

Impressora

Este é um sistema de dois eixos, semelhante a um sistema de *pick and place* construído pela *FESTO*. Este sistema possui um marcador funcionando como uma impressora sendo possível ver eventuais quebras dos requisitos de tempo real nos desenhos.

Demonstrador de sistemas industriais distribuídos

No projecto DESIRE foi desenvolvido pela Universidade de Aveiro e pelo Instituto Politécnico de Castelo Branco um demonstrador para sistemas industriais distribuídos controlados

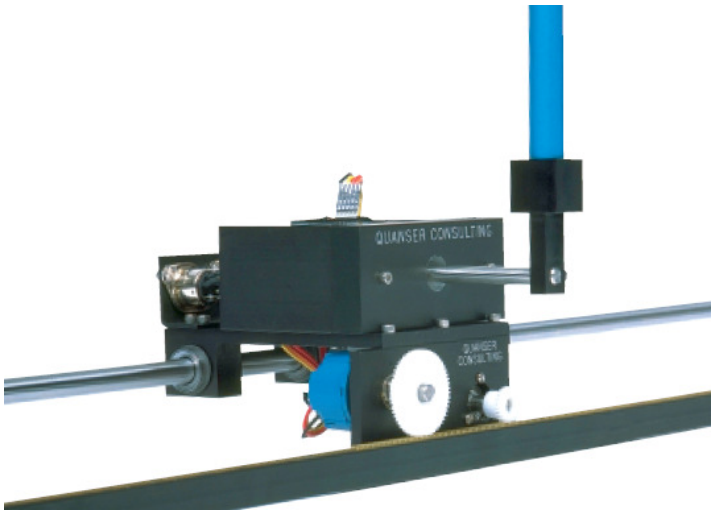


Figura 3.3: Pêndulo invertido auto erigido

remotamente desenvolvido para testar o desempenho de protocolos de acesso ao meio. Este módulo consiste numa estrutura portátil com 3 eixos com motores e encoders: um eixo horizontal rotativo de 40 cm que está perpendicularmente cruzado com dois outros eixos.

Pêndulo invertido controlado através de rede

Este pêndulo invertido da *Googol Technology* (figura 3.4) difere da solução anterior por permitir que o controlo seja efectuado por rede, em concreto através de *Ethernet* [25]. Para além disso o motor deste pêndulo é AC, enquanto o da solução previamente apresentada era DC. O preço do módulo consumiria uma parte significativa do orçamento disponível.

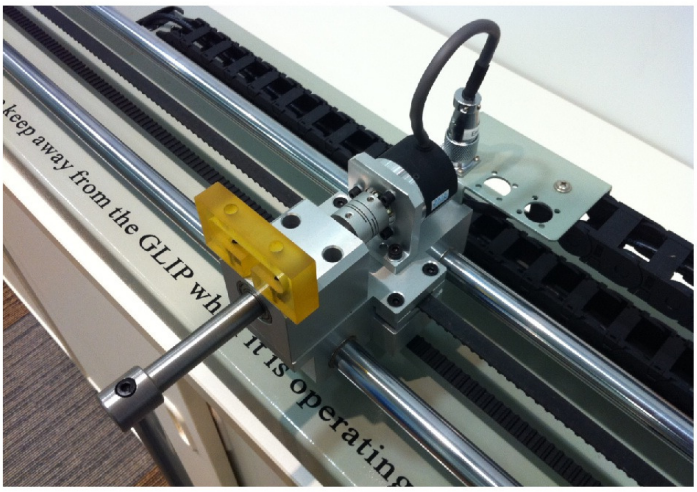


Figura 3.4: Pêndulo invertido controlado através de rede

3.3 Análise comparativa

Na tabela 3.1 encontra-se sumarizada uma comparação entre os equipamentos e os critérios de selecção dos mesmos para o demonstrador.

Da análise da tabela é possível verificar que os módulos *Braço Robótico*,*Sistema de controlo de fluxo e de nível* e *Sistema de controlo de plano e bola* não cumprem critérios obrigatórios, pelo que se excluem. A solução *Pêndulo invertido auto erigido* ultrapassa o financiamento existente, pelo que foi excluída. O controlo do módulo *Pêndulo invertido controlado através de rede* foi considerado complexo pelo que foi excluído.

De entre conjunto dos restantes módulos que cumpriam os critérios definidos, optou-se por adquirir a *Bola no plano* e o *Impressora*. Dado já estar disponível, optou-se por adaptar no imediato o *Demonstrador de sistemas industriais distribuídos*, sendo sobre este que recairá a restante dissertação. De seguida será apresentado com mais detalhe o *Demonstrador de sistemas industriais distribuídos* e a solução a ser implementada com essa estrutura.

Módulo \Critério	Sistema Ciber-Físico	Sistema de Tempo Real	Arquitectura Aberta	Tráfego de Dados de Tempo Real	Tráfego de vídeo	Integração com Rede Ethernet Tradicional	Preço	Tamanho	Simplicidade
Braço Robótico	✓	✓	X	✓	X	X	✓	✓	X
Sistema de controlo de fluxo e de nível	✓	X	✓	✓	X	X	✓	X	✓
Sistema de controlo de plano e bola	✓	✓	X	✓	✓	X	✓	✓	✓
Bola no Plano	✓	✓	✓	✓	✓	X	✓	✓	✓
Pêndulo invertido auto erigido	✓	✓	✓	✓	X	X	X	✓	✓
Impressora	✓	✓	✓	✓	X	X	✓	✓	✓
DESIRE	✓	✓	✓	✓	X	✓	✓	✓	✓
Pêndulo invertido controlado através de rede	✓	✓	✓	✓	X	X	✓	✓	X

Tabela 3.1: Tabela sumária dos equipamentos e critérios de selecção

3.4 Demonstrador de sistemas industriais distribuídos

Este módulo servia inicialmente para demonstrar um sistema industrial distribuído controlado remotamente. Para além da estrutura mecânica, o módulo continha um barramento CAN, sendo controlado através de uma aplicação no computador. Este computador tinha um servidor que permitia o controlo remoto do módulo. Entre o que foi referido apenas será aproveitado para o novo demonstrador a componente mecânica.

O módulo a ser desenvolvido pretende servir de demonstrador para um sistema de controlo distribuído ligado através de *Ethernet* e consiste numa estrutura portátil com 3 eixos com motores e encoders.

3.4.1 Módulo Físico

Tal como é possível observar na figura 3.5, a componente física do módulo é composta por um eixo horizontal rotativo de 40 cm que está perpendicularmente cruzado com dois outros eixos. Este eixo tem duas palhetas que formam entre si um ângulo de 188º, e os outros dois eixos têm duas ranhuras. O objectivo final do trabalho será conseguir sincronizar os eixos de tal forma que as palhetas passem através das ranhuras. Se esta sincronização falhar as palhetas baterão nos discos sem causar danos pois são flexíveis. Para além dos discos já referidos, o módulo possui mais 3 que têm uma ranhura que é detectada através de um emissor/receptor de infravermelhos (daqui em diante designado detector de passagem por 0) e que servirão para fazer uma calibração inicial da posição do sistema. O eixo 2 tem um ângulo de 92º entre as duas ranhuras e o eixo 3 tem um ângulo de 277º.

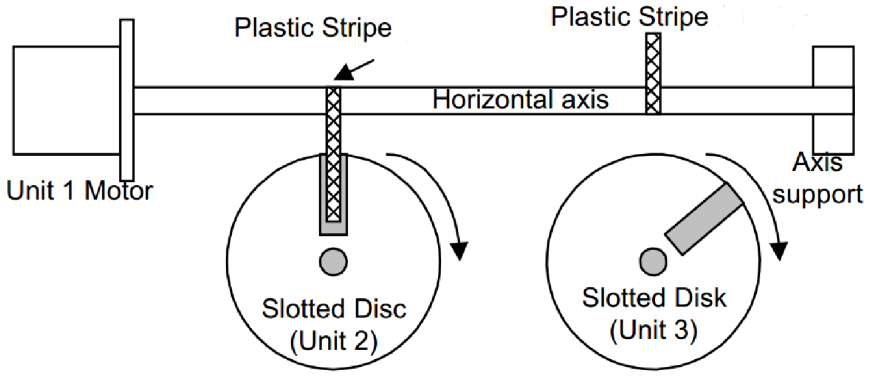


Figura 3.5: Esquema da componente física

3.4.2 Sistema a implementar

Para controlar este módulo foi implementado o sistema representado pelo diagrama de blocos da figura 3.6. A componente responsável pelo controlo (daqui em diante designado de *Process Controller*) será um programa a correr num computador, que também será usado como interface para os utilizadores. A informação necessária ao controlo, chegará ao *Process Controller* através de *Ethernet* sendo enviada através do *switch* a ser testado. A este *switch* estão ligados 3 nós (um para cada eixo) que irão gerar o tráfego de tempo real do sistema.

Estes nós consistem em *gumstix* que é um sistema embutido que servirá de interface com a rede de comunicação Ethernet, para além de receberem os sinais do sistema físico (através do sensor de Posição) e gerarem os sinais de controlo dos eixos (sinal de PWM). O sensor de posição é composto pelo *encoder* e pelo detector de passagem por 0.

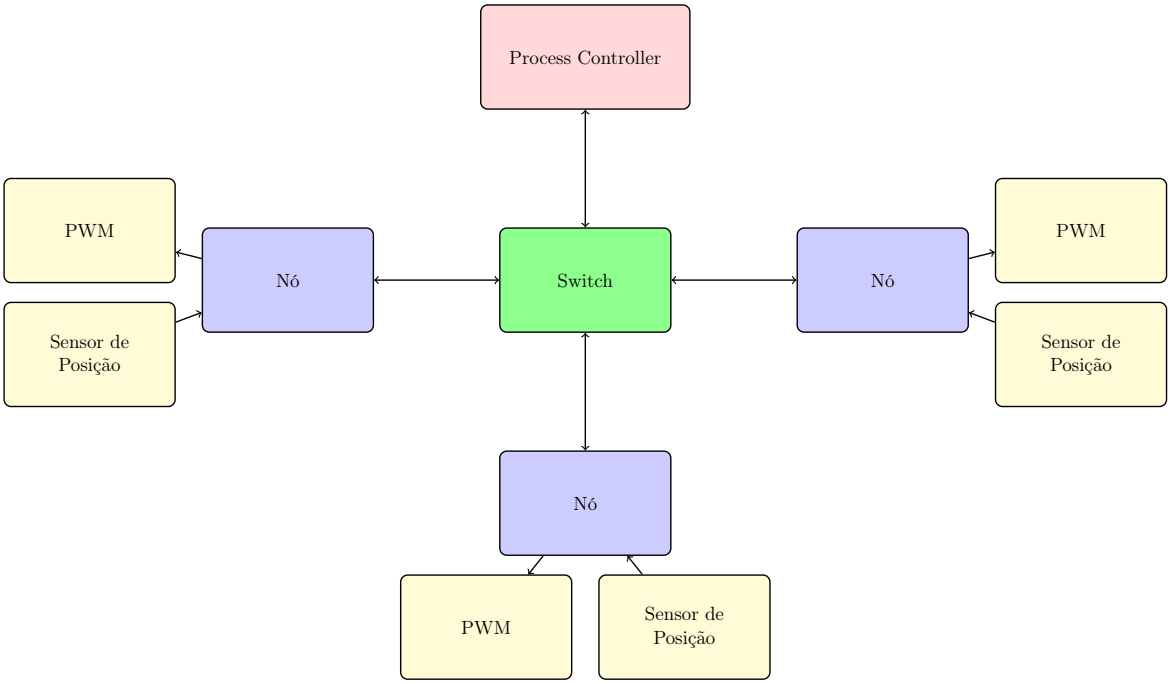


Figura 3.6: Diagrama de blocos do sistema

Para além dos nós apresentados na figura, será acrescentado um outro que consiste num computador que irá gerar tráfego de interferência. Este nó será ligado ao *switch*. As mensagens serão direccionadas ao *Process Controller* que as rejeitará.

3.4.3 Gumstix

Nesta dissertação, entende-se por *gumstix* o conjunto funcional das placas *Verdex-pro*, *Robostix* e *Netpro* (figura 3.10).

A placa *Verdex-pro* que corre o sistema operativo Linux 2.6 é baseada num processador Marvell PXA270 de 400MHz e é a responsável pelo envio, recepção e respectivo processamento das mensagens do módulo. As mensagens são enviadas através de *sockets*.

O envio e recepção das mensagens *Ethernet* é feito através da placa *Netpro*. Esta placa contém um interface para *Ethernet*.

A placa *Robostix* que contém um Atmega168 é a responsável pela leitura do HCTL através de GPIO (actuando no HCTL também através de pinos de GPIO), da leitura do detector de passagem por 0 através de um pino configurado por interrupção e de gerar o sinal PWM de controlo do motor.

A *Verdex-pro* irá a cada $10ms$ enviar uma mensagem à *Robostix* com um pedido de informação da posição do eixo(tal como representado pela seta 1 da figura 3.10), ficando depois num estado ocupado à espera da resposta. A *Robostix* recebe este pedido, faz a leitura do HCTL e envia uma resposta para a *Verdex-pro* (seta 2 da figura 3.10). Por fim, a *Verdex-pro*

constrói uma trama Ethernet que passa à *Netpro* que por sua vez a envia através da rede de comunicação.

3.4.4 Placa de interface

Na figura 3.7 é possível observar o diagrama de blocos para um dos eixos de uma placa que foi desenvolvida para servir de interface entre a electrónica ligada ao sistema físico e a electrónica do sistema de controlo e comunicação.

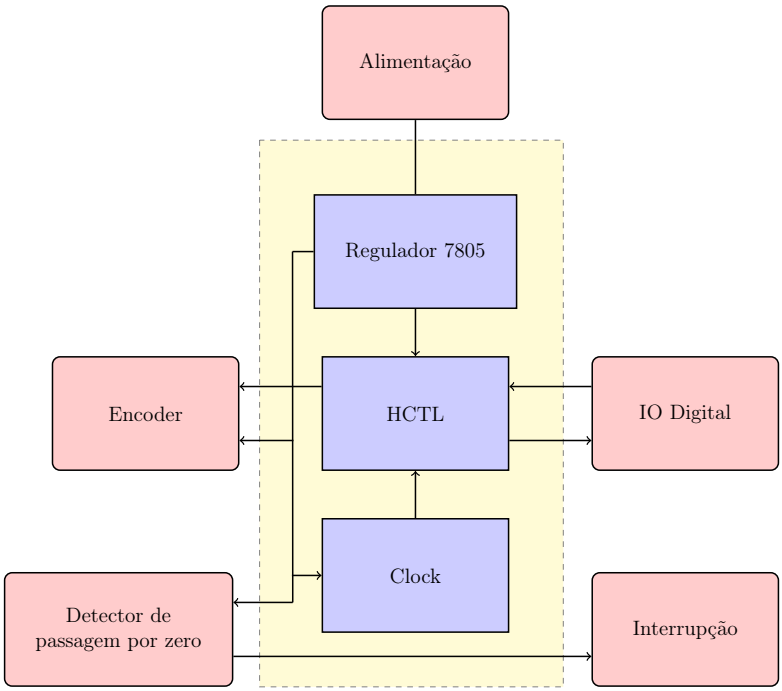


Figura 3.7: Diagrama de blocos para um dos eixos

De seguida é apresentada com mais detalhe a electrónica do módulo.

3.4.5 Módulo de Electrónica

Detector de passagem por 0

A detecção da passagem por 0, isto é pela posição de referência inicial, será feita com um HOA6980 da Honeywell. O HOA6980 consiste num díodo emissor de infravermelho virado para um detector de infravermelho com um *Schmitt Trigger*.

3.4.6 Encoders

Os encoders utilizados são os RI32-AR1000 da Hengstler que têm como saída 1000 pulsos por rotação por canal. Dado que cada encoder possuí dois canais em quadratura ter-se-á então uma resolução de 4000 passos por rotação (figura 3.8).

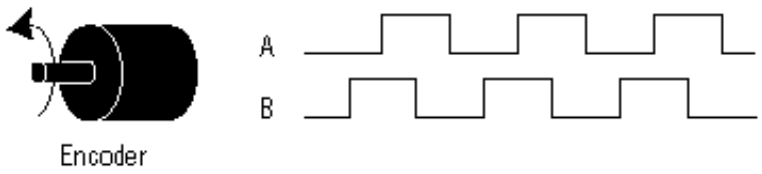


Figura 3.8: Sinais do encoder

HCTL

De forma a evitar desperdício de processamento da *gumstix*, foi utilizada um HCTL-2016 que irá fazer a interface entre a *gumstix* e o encoder fazendo a contagem dos sinais em quadratura.

Este dispositivo será operado no modo de 16 *bits* sendo primeiro lido o *byte* mais significativo e depois o menos significativo. Para um correcto funcionamento do HCTL, o sinal de *Clock* aplicado, deverá ter pelo menos 3 vezes a frequência máxima dos sinais do *encoder*. Como referido, cada canal do encoder gera 1000 pulsos por rotação. A velocidade máxima de rotação do módulo é inferior a 4 voltas por segundo, logo ter-se-á no pior caso 4000 pulsos por canal, por segundo. A frequência mínima do *Clock* teria então que ser 12KHz. A frequência máxima que o HCTL suporta é aproximadamente 14MHz.

3.4.7 Clock

Para gerar o sinal de *clock* do HCTL recorreu-se a um circuito com um 74LS04. Este é um integrado com 6 portas NOT. A frequência escolhida foi de aproximadamente 6MHz, o que se encontra dentro da gama admissível para esta situação.

3.4.8 Alimentação

A alimentação será efectuada por um transformador de $220V$ AC para $12V$ DC, seguido por um regulador 7805 que converterá os $12V$ em $5V$.

3.4.9 Driver do Motor

Os módulos MR001 foram usadas como *drivers* do motor. Estes módulos são baseados no integrado L298 que é uma ponte H, podendo controlar dois motores e fornecendo até 2A para cada. Os módulos foram adaptados de forma a controlar-se um único motor fornecendo 4A. Na figura 3.9 é possível observar um destes módulos e a função das respectivas entradas. A placa de alimentação anteriormente referida está directamente ligada ao módulo. Dado que os motores só rodarão num sentido ir-se-á colocar um sinal estático nos *inputs*, neste caso de '1' no *input A* e de '0' no *input B*. No *enable* será colocado o sinal de PWM.

3.5 Módulo de Software

Tendo selecionado o *hardware* a utilizar, e desenvolvida a placa de interface da electrónica, apresenta-se de seguida o módulo de *software* implementado.

Dado o extenso trabalho de *debugging* que foi necessário realizar de forma a colocar o sistema a funcionar, ir-se-á apresentar de forma detalhada o *software* implementado. De

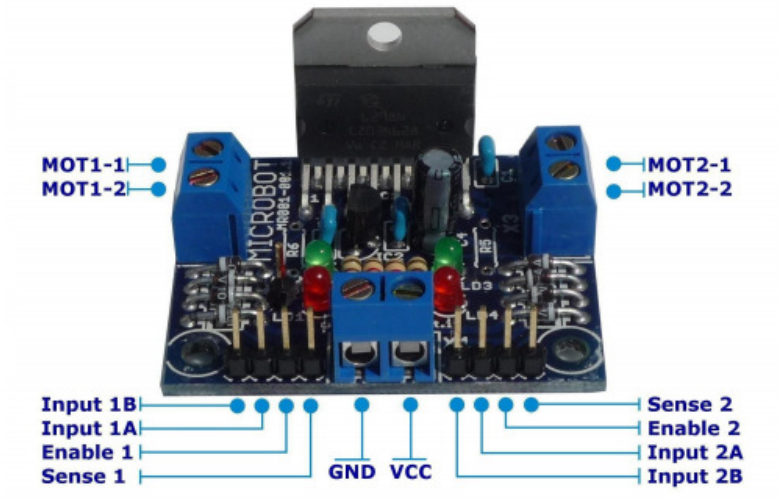


Figura 3.9: Driver do motor

seguida apresentar-se-ão os principais problemas que se foram encontrando e que influenciaram a solução final.

Na figura 3.10, está representado um diagrama de blocos que representa o percurso da informação ao longo do sistema. Ir-se-á de seguida analisar o fluxo da informação de baixo para cima.

3.5.1 Robostix

A *Robostix* é responsável por gerar o PWM que será aplicado aos motores, por ler o valor do HCTL e detectar através de interrupção a passagem por 0.

Vai receber três tipos de mensagens:

- SET_PWM, que vai actualizar o valor do PWM;
- $HCTL\_RESET$, que vai colocar o contador do HCTL a 0;
- $HCTL\_GET\_READING$, que vai ler o valor actual do HCTL e retorna-lo através de uma mensagem $HCTL\_READING$ pela USART.

Estas mensagens são recebidas através da USART e tratadas num ciclo infinito na *Robostix*. A *Robostix*, por sua vez, vai gerar mensagens do tipo:

- $HCTL\_READING$, quando receber uma mensagem do tipo $HCTL\_GET\_READING$;

A quando da passagem por zero, será gerada uma interrupção, a qual fará o contador do HCTL ficar a 0.

3.5.2 Verdex-Pro

Tal como referido anteriormente, esta placa corre o sistema operativo Linux 2.6. Esta placa corre duas *threads*. Uma que irá processar as mensagens *Ethernet* recebidas, fazendo

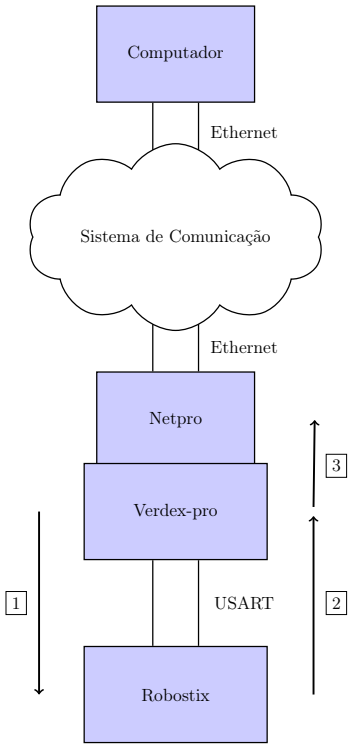


Figura 3.10: Percurso da informação ao longo do sistema

os pedidos respectivos à *Robostix* e que para além disso irá de 10 em $10ms$ fazer um pedido de leitura do HCTL à *Robostix*, sendo este o período de amostragem do sistema. Uma outra responsável por receber as mensagens da USART, trata-las e construir a trama *Ethernet* que será enviada.

Ticks

Os sistema operativos precisam de ter uma forma de agendar trabalho de forma precisa. Para tal, alguns sistemas operativos possuem uma interrupção que, de forma periódica, interrompe o processador de forma a correr o trabalho agendado. Esta interrupção é conhecida como *tick* do relógio do sistema, ou, abreviadamente, por *tick*. A cada *tick*, o processador:

- executa as actividades agendadas;
- verifica se algum dos *timers* expirou e executa as funções que lhe estão associadas;
- agenda actividades para serem executadas no *tick* seguinte.

O sistema operativo presente nesta placa trabalha desta forma, possuindo, de origem, *ticks* de $10ms$. O *kernel* Linux foi reconfigurado de tal forma, que passasse a ter os *ticks* a cada $1ms$.

3.5.3 Netpro

Irá receber e enviar as mensagens *Ethernet*.

3.5.4 Process Controller

Este é o elemento responsável pelo controlo do sistema e consiste num programa a correr num computador portátil. Este programa recebe as mensagens vindas da Ethernet com a leitura do HCTL, respondendo com um sinal de controlo adequado a essa leitura.

Controlo

Controlo do eixo 1

O controlo da posição deste eixo é feito face a uma velocidade angular pretendida, tal como apresentado na equação 3.1.

$$\frac{d\omega_{pretendido}}{dt} - \frac{d\omega_1}{dt} = 0 \tag{3.1}$$

O controlador implementado é um PID.

Controlo dos eixos 2 e 3

O controlo da posição destes eixos é feito face à posição actual do eixo 1. Isto é, os eixos 2 e 3 tentam ter uma variação de fase nula face ao eixo 1, tal como apresentados nas equações 3.2, 3.4.

$$\frac{d\theta_2}{dt} - \frac{d\theta_1}{dt} + \theta_{ranhura_2} = 0 \tag{3.2}$$

$$\frac{d\theta_3}{dt} - \frac{d\theta_1}{dt} + \theta_{ranhura_3} = 0 \tag{3.3}$$

Dado que os relógios de funcionamento das gumstix não estão sincronizados, os instantes em que chegarão as leituras do HCTL não é conhecido à priori. De forma a compensar a diferença de tempo entre os instantes de leitura dos HCTL, foram implementados estimadores. Estes estimadores calculam a variação de fase do eixo 1 no intervalo de tempo entre as leituras, considerando que o eixo 1 manteve a velocidade. A estas fases há ainda que acrescentar as diferenças de fase entre a ranhura onde passa a palheta e o detector de passagem por 0, que são adicionadas ao erro. Desta forma as equações de controlo passam a ser:

$$\frac{d\theta_2}{dt} - (\frac{d\theta_1}{dt} + (tempo_2 - tempo_1) * \omega_1) + \theta_{ranhura_2} = 0 \tag{3.4}$$

$$\frac{d\theta_3}{dt} - (\frac{d\theta_1}{dt} + (tempo_3 - tempo_1) * \omega_1) + \theta_{ranhura_3} = 0 \tag{3.5}$$

O facto de a posição inicial ser desconhecida não é um problema dado que o contador do HCTL é colocado a zero aquando da passagem por zero, sendo a fase nesse ponto conhecida.

O controlador implementado é um PID.

Linearização

O motor é um elemento altamente não linear, tal como se pode verificar pela figura 3.11. Como tal, foi necessário realizar uma linearização do seu comportamento para se poder aplicar controlo linear neste sistema.

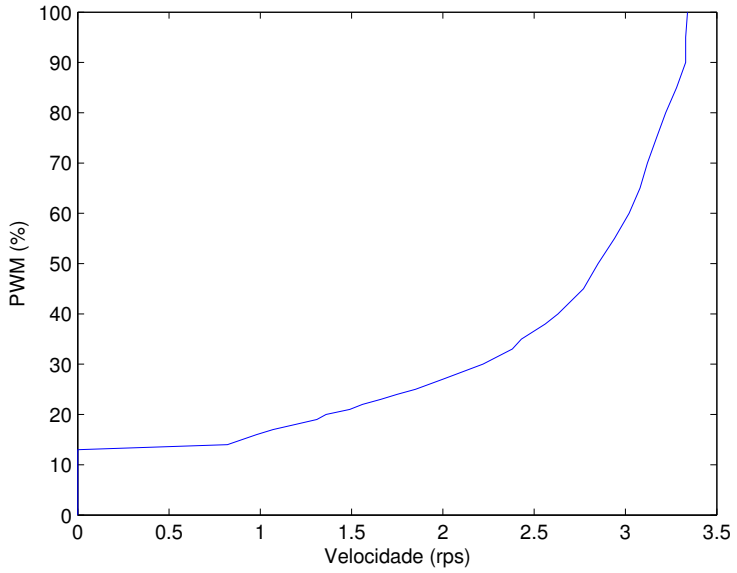


Figura 3.11: Velocidade (rps) em função do duty cycle do PWM

Observando a figura 3.11 pode verificar-se que o PWM tem uma forma do género

$$PWM(v) = e^{(a0+a1*v+a2*v^2)} \tag{3.6}$$

Tendo uma função da velocidade em relação ao pwm, obtida através de medição, procedeu-se aos seguintes cálculos matemáticos e correndo o código indicado, obtiveram-se os valores das constantes $a0$, $a1$ e $a2$.

$$ln(PWM(velocidade)) = a0 + a1 * velocidade + a2 * velocidade^2 \tag{3.7}$$

** Retiraram-se os valores abaixo do pwm igual 12, cuja velocidade era 0, de forma a poder-se ter uma função injectiva. Colocou-se a equação anterior numa forma matricial para poder ser utilizado no Matlab*

```
pwm_alterado=log(pwm(13:end));
velocidade_alterada = [velocidade(13:end).^2;
velocidade(13:end);
ones(size(velocidade(13:end)))];
```

** Verificou-se que velocidade_ alterada era injectiva de forma a poder ser invertida. Neste caso, e como a função não é quadrada, tem que se usar a pseudo-inversa de Moore-Penrose.*

```
A = pwm_alterado * pinv(velocidade_alterada)
```

Para se verificar que os valores anteriores anteriores estavam correctos representou-se a linearização efectuada e comparou-se com os valores reais antes medidos aplicando-lhes a linearização (figura 3.12).

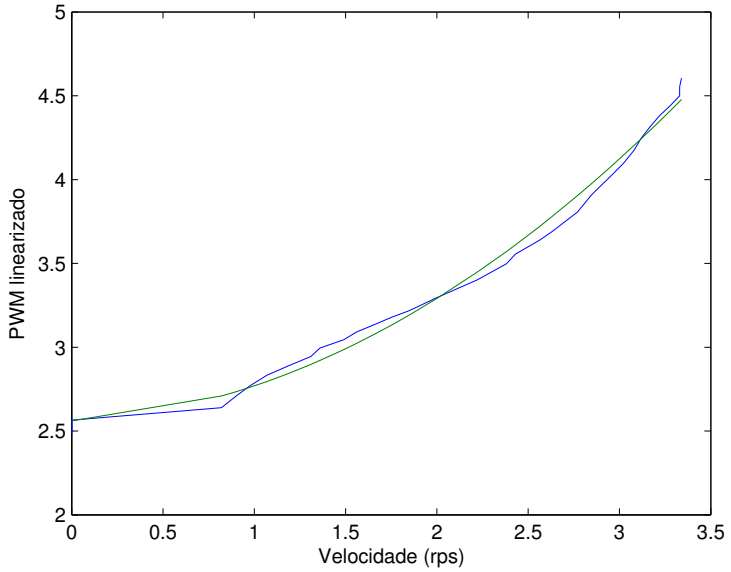


Figura 3.12: Linearização

O procedimento anterior foi repetido para os restantes motores.

Debugging

Como referido anteriormente, foi necessário executar um extenso trabalho de *debugging* que moldou significativamente a estrutura da solução implementada. Nesta sub-secção serão apresentados os elementos mais relevantes desse trabalho e o seu impacto na estrutura final.

Ticks Dada a elevada capacidade de processamento das gumstix e o facto de só se estar a correr duas *threads*, foi considerado inicialmente que as latências introduzidas por ela seriam desprezáveis. No entanto, após efectuar medidas verificou-se que as mesmas ultrapassavam os $20ms$. Isto devía-se ao facto de os *ticks* do sistema serem de $10ms$ e serem gastos dois *ticks*. Por esse motivo, se reconfigurou o *Kernel* do sistema para suportar *ticks* de $1ms$.

Dessincronização do envio de mensagens entre Robostix e Verdex-Pro Inicialmente, o envio das mensagens de leitura do HCTL, eram despoletadas por uma interrupção regular na Robostix. Isto tinha como desvantagem o facto de as mensagens recebidas poderem não serem imediatamente tratadas pela Verdex-Pro caso esta se encontrasse em *idle* mas somente no *tick* seguinte (figura 3.13). De forma a sincronizar as mensagens, as leituras do HCTL passaram a ser despoletadas pela Verdex-Pro que depois executará um ciclo *for* de forma a não passar a *idle* até a mensagem da Robostix chegar. Desta forma, a latência passou

a ser constante pois deixou de haver um desfasamento, que era diferente de cada vez que se ligava o sistema, entre o instante em que a mensagem era enviada pela Robostix e o instante em que era efectivamente tratada pela Verdex-Pro.

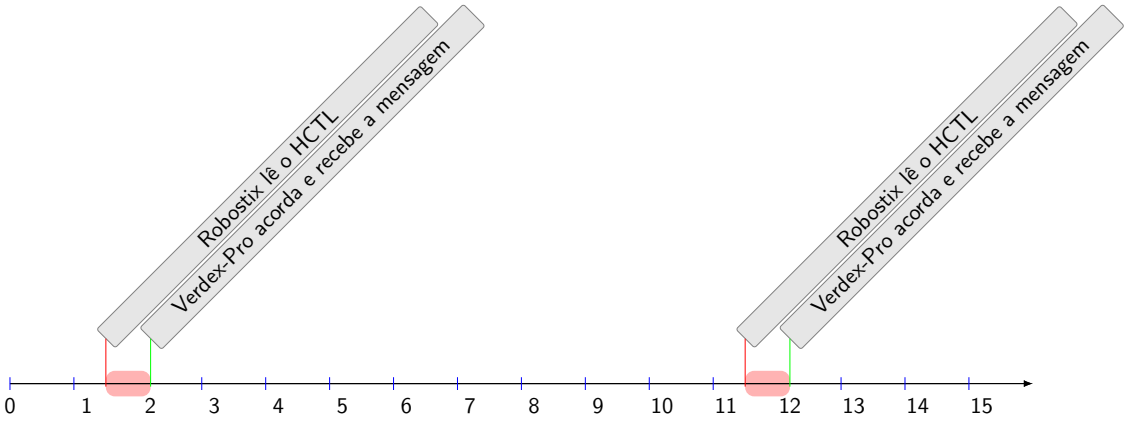


Figura 3.13: Diagrama temporal (ms)

Latências variáveis devido ao uso do mesmo meio de comunicação por várias mensagens Inicialmente, existia uma mensagem que era gerada aquando da passagem por zero. Esta mensagem, era enviada através da mesma USART por onde era enviada a mensagem de leitura do HCTL, entrando em conflito com a mesma pelo meio, e provocando latências variáveis. Esta mensagem foi eliminada, passando o contador do HCTL a 0 de cada vez que há uma passagem por zero.

Dessincronização entre os relógios das Gumstix As Gumstix não estão sincronizadas pelo que as mensagens de leitura do HCTL de cada nó serão enviadas em momentos distintos. De forma a minimizar o erro causado por essas diferenças de tempo, foram implementados no controlo estimadores.

Mensagem enviada fora de tempo Foi detectado um erro que acontece de forma muito esporádica, no qual uma mensagem é enviada fora daquele que devia ser o seu momento de envio. Não foi possível confirmar a origem deste problema. A solução encontrada foi detectar e rejeitar esta mensagem no Process Controller. Desta forma o impacto desta mensagem anómala no sistema foi praticamente nulo.

3.5.5 Contributos

Dada a sua experiência anterior com sistemas Linux e implementação de sistemas de comunicação Ethernet, contou-se com o contributo de Luís Silva, bolseiro de investigação do projecto SERV-CPS, para a implementação do sistema de comunicação, tendo sido de sua exclusiva autoria a reconfiguração do kernel Linux. Contou-se também com o contributo de Milton Cunguara, aluno de Doutoramento da Universidade de Aveiro a realizar tese sobre o controlo de sistemas com perdas, que sugeriu possíveis soluções para o controlo.

Capítulo 4

Testes e Resultados

Neste capítulo serão apresentados os testes efectuados com o objectivo de comparar o desempenho do sistema de comunicação baseado no *switch* HaRTES com um sistema de comunicação *Ethernet* tradicional, bem como as calibrações feitas ao sistema e os resultados dos testes.

4.1 Sistemas de testes

Existiram dois sistemas de testes (figura 4.1). Um sistema composto pelo demonstrador anterior, um computador pessoal onde correrá o *Process Controller*, um *switch* COTS e um outro computador que irá gerar tráfego de interferência. No segundo sistema o *switch* COTS foi substituído pelo *switch* HaRTES.

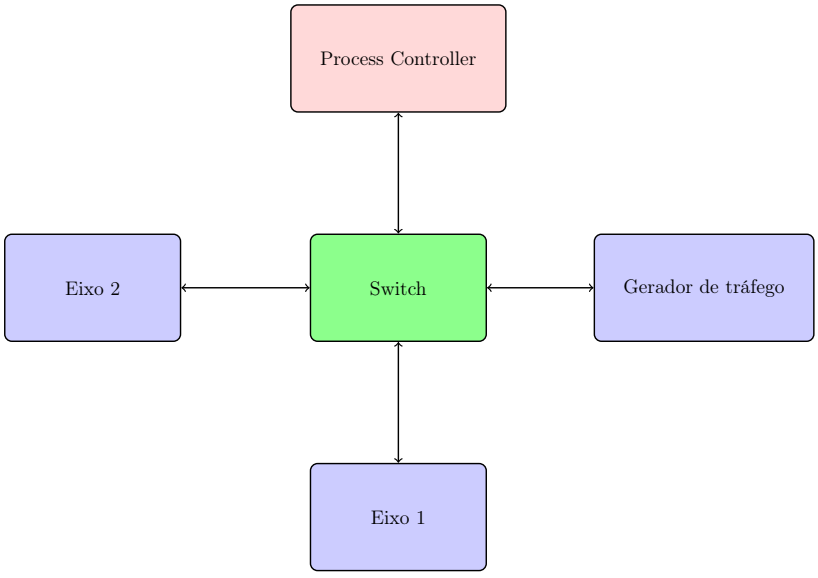


Figura 4.1: Diagrama de blocos do sistema de testes

Os testes tiveram a duração de 180 segundos, com os eixos em posições iniciais aleatórias tendo como velocidade angular pretendida para o eixo 1 $10\frac{rad}{s}$ ou seja aproximadamente 1.6 rotações por segundo. A esta velocidade ainda é possível aos observadores verificarem

visualmente quando a palheta passa na ranhura, e já é uma velocidade suficientemente alta para que eventuais falhas do sistema se traduzirem em impactos regulares da palheta no eixo.

Os resultados foram obtidos gravando num ficheiro os valores de interesse e representando-os de seguida com recurso ao MATLAB.

Os programas da Gumstix foram colocados em execução através de acesso remoto usando SSH. Durante o período do teste este protocolo estará activo introduzindo tráfego na rede de comunicação. No entanto, as únicas mensagens a serem trocadas pelo sistema durante este período serão mensagens Keep Alive, que são curtas e esporádicas, sendo o seu efeito no sistema desprezável.

4.2 Calibrações

O perímetro dos eixos é de aproximadamente $36.7cm$, e a largura da ranhura é de $1cm$. Desta forma, e considerando que a palheta deverá passar no centro da ranhura, dever-se-á ter um erro inferior na fase dos eixos 2 e 3 de 1.36%.

Foram realizados testes que permitiram efectuar a calibração dos parâmetros dos controladores PID de tal forma que os erros fossem inferiores aos que se tinham definido anteriormente como erros máximos.

Nas secções seguintes são apresentados os resultados dos testes.

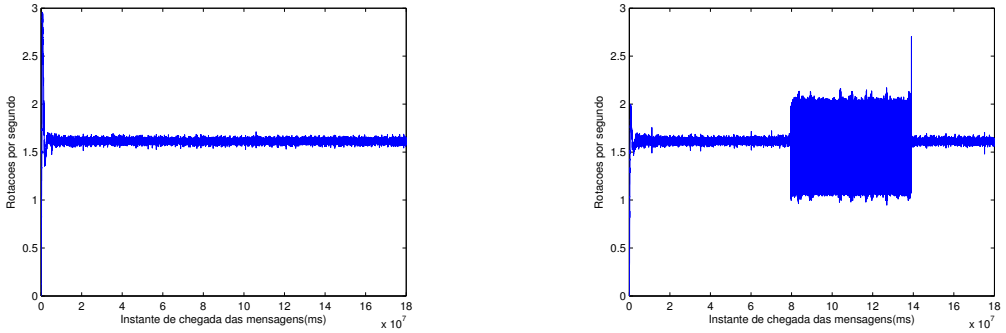
4.3 Resultados do teste com switch tradicional

Neste teste foi utilizado um *switch* COTS, tendo o teste sido executado duas vezes, uma vez sem qualquer tráfego de interferência e uma segunda em que foi introduzido no sistema tráfego de interferência entre os 80 e os 140 segundos. O tráfego de interferência consiste em mensagens de 1400 *bytes* enviadas a cada $1ms$, tendo como destino o computador pessoal onde corre o *Process Controller* que as irá rejeitar. Este tráfego de interferência equivale a uma carga de 11,2% do sistema de comunicação dado que o mesmo trabalha a $100Mbits/s$.

Eixo 1

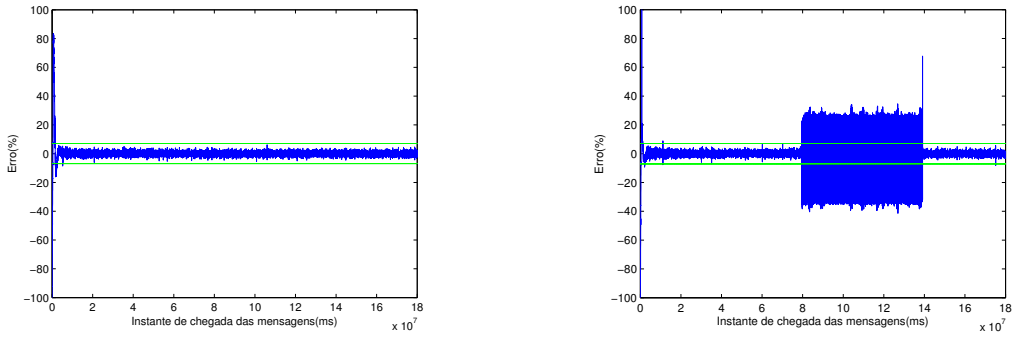
Na figura 4.2 está representada a velocidade em rotações por segundo do eixo 1. No teste sem interferência (figura 4.2a), o sistema apresenta uma percentagem de *overshoot* na ordem dos 87% e um tempo de estabelecimento de 31 segundos, enquanto que no teste com interferência (figura 4.2b) a percentagem de *overshoot* é menor, sendo da ordem dos 33%, e o tempo de estabelecimento é maior, sendo da ordem dos 37 segundos. Esta diferença deve-se ao facto de as posições iniciais do sistema serem diferentes.

Com a interferência, a velocidade passou a oscilar aproximadamente entre uma rotação por segundo e duas rotações por segundo. O motivo que leva à deterioração do comportamento do sistema será apresentado posteriormente. Após o fim do envio do tráfego de interferência o sistema recupera, voltando a estar dentro dos valores pretendidos de erro 38 segundos após o fim do envio do tráfego de interferência, tal como é possível verificar pela figura 4.3. Na figura 4.3 é possível visualizar a percentagem do erro da velocidade angular (linha azul), e quando a mesma é inferior a 7% (linhas verdes). Quando não é transmitido tráfego de interferência, o erro percentual médio é de aproximadamente 2.3% em ambos os casos, passando para 17% quando é transmitido esse tráfego.



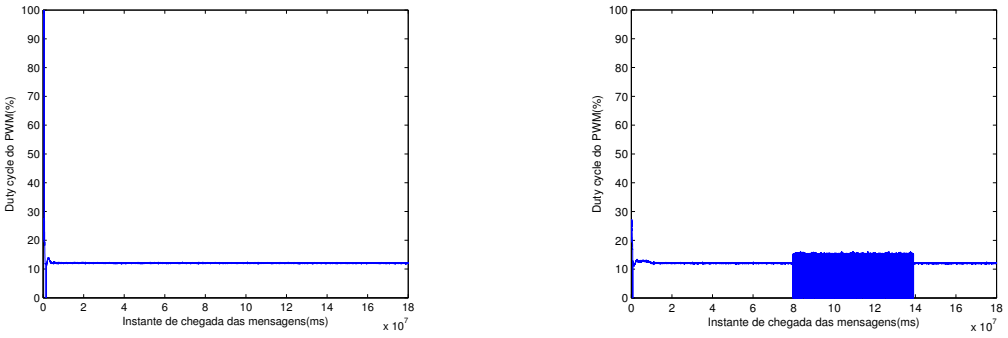
(a) sem interferência (b) com interferência

Figura 4.2: Velocidade do eixo 1 em rotações por segundo



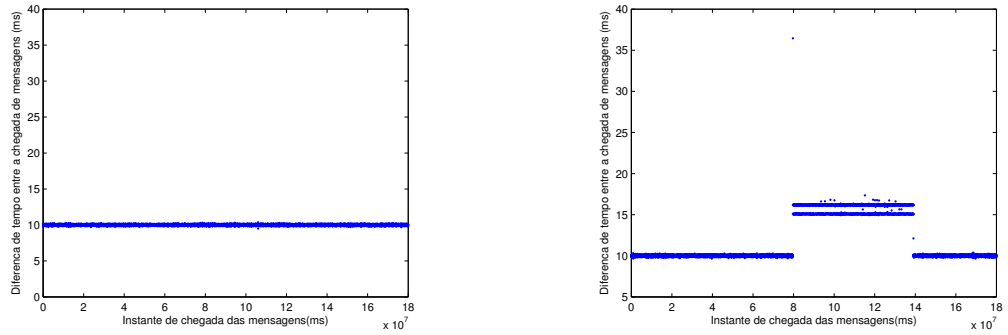
(a) sem interferência (b) com interferência

Figura 4.3: Erro da velocidade angular do eixo 1 em percentagem



(a) sem interferência (b) com interferência

Figura 4.4: Sinal de controlo (percentagem de PWM) do eixo 1



(a) sem interferência (b) com interferência

Figura 4.5: Diferença de tempo entre a chegada de mensagens do eixo 1

Na figura 4.4 está representado o sinal de controlo (percentagem de PWM), que estabiliza mostrando que o sistema está perfeitamente estável quando não existe interferência e que fica instável com a mesma.

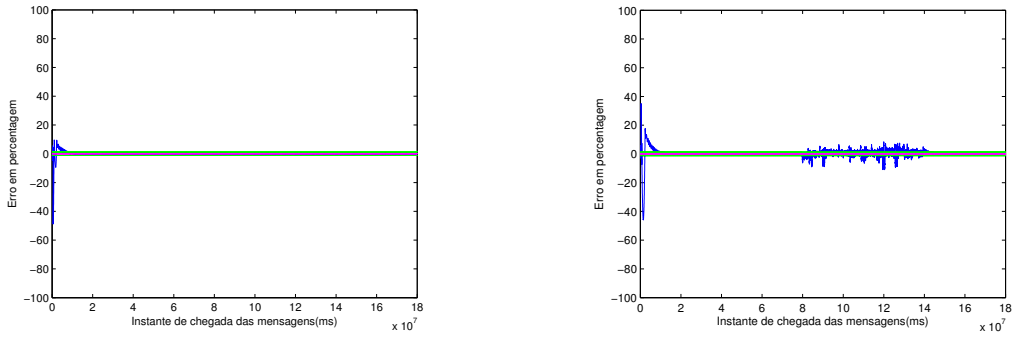
Na figura 4.5 está representado a diferença de tempo entre a chegada das mensagens com a leitura da posição do eixo.

Na situação em que não há interferência (figura 4.5a), o *jitter* médio é de 54,6 microsegundos com um *jitter* máximo de 498 microsegundos. Na situação em que há interferência é perdida uma mensagem a cada três mensagens enviadas. Para além disso, uma das mensagens recebidas em cada conjunto de duas mensagens recebidas, tem um atraso médio de 5 milisegundos. É esta perda de informação e atraso de outras que faz com que o sistema destabilize com a existência de mais tráfego.

Eixo 2 e 3

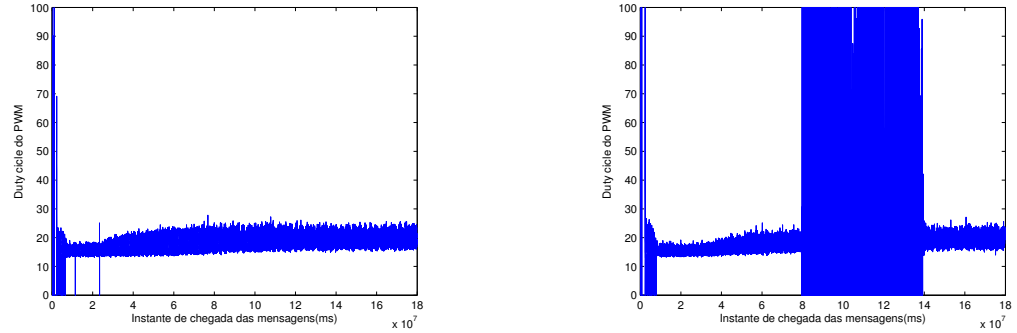
Dado que existiu uma avaria no eixo 3 e de forma a comparar os resultados deste teste com os do *switch* HaRTES que só tem disponíveis 4 portas, o que obrigará a que um dos eixos não seja ligado, este teste foi efectuado só com o eixo 2.

Na figura 4.6 está representada a percentagem do erro da fase do eixo 2 face ao eixo 1 e quando a mesma é inferior a 1.3% (linhas verdes). No caso em que não há interferência, o



(a) sem interferência (b) com interferência

Figura 4.6: Percentagem de erro da fase do eixo 2 face ao eixo 1



(a) sem interferência (b) com interferência

Figura 4.7: Sinal de controlo (percentagem de PWM) do eixo 2 em percentagem

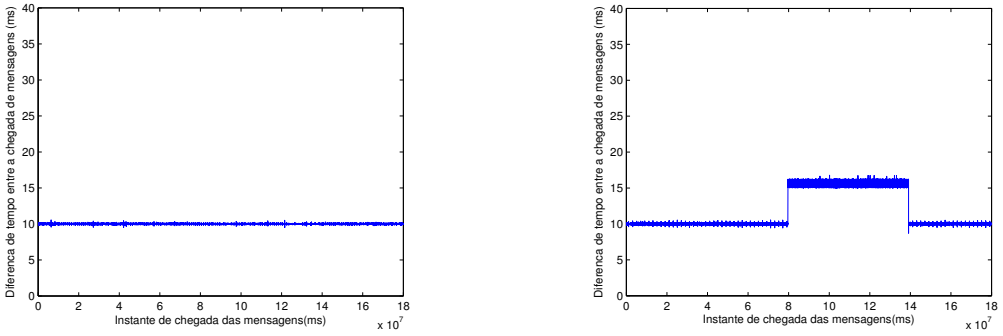
---

sistema tem um tempo de estabelecimento de 16 segundos e no caso em que há interferência existe um tempo de estabelecimento de 18 segundos. Na primeira situação, o erro percentual médio é de 0.3%, sendo semelhante na segunda situação, fora do período de interferẽncia do tráfego. No período de interferência o erro percentual médio é de 1.6%.

Através da figura 4.7b verifica-se que o sistema está perfeitamente estável até à introdução do tráfego de interferência, instabilizando devido ao mesmo e voltanto a estabilizar quando esse tráfego deixa de ser introduzido.

Na figura 4.8 está representado a diferença de tempo entre a chegada das mensagens.

Na situação em que não há interferência (figura 4.8a), o *jitter* médio é de 41,6 microsegundos com um *jitter* máximo de 635 microsegundos. Na situação em que há interferência (4.8b), e tal como acontecia para o eixo 1, é perdida uma mensagem a cada três mensagens enviadas. Para além disso, uma das mensagens recebidas em cada conjunto de duas mensagens recebidas, tem um atraso médio de 5.75 milisegundos. Mais uma vez, é esta perda de informação e atraso de outras que faz com que o sistema destabilize com a existência de mais tráfego.



(a) sem interferência (b) com interferência

Figura 4.8: Diferença de tempo entre a chegada de mensagens do eixo 2

4.4 Resultados do teste com switch HARTES

Neste teste foi utilizado o *switch* HaRTES, tendo o teste sido executado duas vezes, uma vez sem qualquer tráfego de interferência e uma segunda em que foi introduzido no sistema tráfego de interferência entre os 80 e os 140 segundos. Tal, como na situação anterior, o tráfego de interferência consiste em mensagens de 1400 *bytes* enviadas a cada $1ms$, tendo como destino o computador pessoal onde corre o *Process Controller* que as irá rejeitar.

Para separar o tráfego de tempo real do tráfego de interferência, foram configurados no *switch* HaRTES servidores com capacidade para 1600 *bytes*, com um EC de 5 milisegundos. Optou-se por este tempo de forma a evitar que um eventual envio de uma mensagem fora de tempo possa afectar o normal envio das restantes mensagens de tempo real. O tráfego de tempo real não deverá ser afectado pelo tráfego de interferência pois este encontra-se confinado ao seu servidor.

Eixo 1

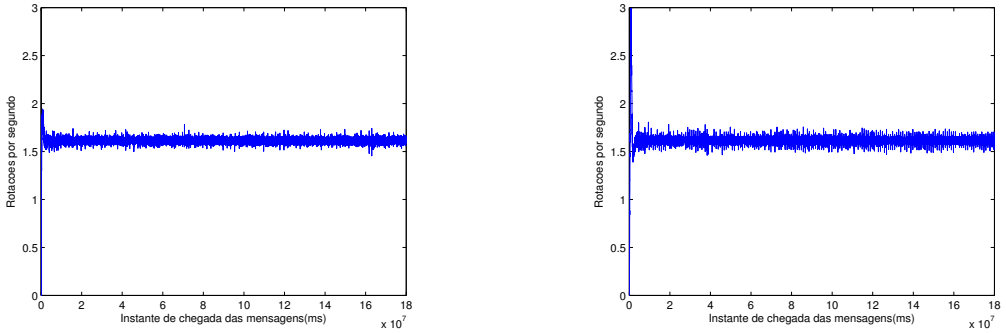
Na figura 4.9 está representada a velocidade em rotações por segundo do eixo 1. No teste sem tráfego de interferência (figura 4.9a), o sistema apresenta uma percentagem de *overshoot* na ordem dos 25% e um tempo de estabelecimento de 33 segundos, enquanto que no teste com tráfego de interferência (figura 4.9b) a percentagem de *overshoot* é maior, sendo da ordem dos 97%, e o tempo de estabelecimento é de 47 segundos.

Na figura 4.10 é possível visualizar a percentagem do erro da velocidade angular (linha azul), e quando a mesma é inferior a 5% (linhas verdes). O tráfego tem um erro médio de 0.3% com e sem interferência. Ao contrário do que aconteceu com os testes com o *switch* tradicional, o facto de existir tráfego não tempo real a ser enviado, não afecta o tráfego de tempo real.

Na figura 4.11 está representado o sinal de controlo (percentagem de PWM) que estabiliza mostrando que o sistema está perfeitamente estável.

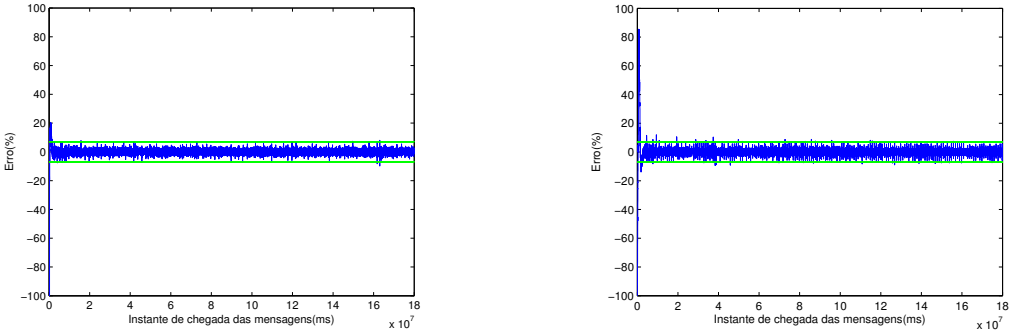
Na figura 4.12 está representado a diferença de tempo entre a chegada das mensagens.

Na situação em que não há tráfego de interferência (figura 4.5a), o *jitter* médio é de 40.2 microsegundos com um *jitter* máximo de 682 microsegundos. Na situação em que há tráfego de interferência (figura 4.5a), o *jitter* médio é de 48.7 microsegundos com um *jitter* máximo de 462 microsegundos.



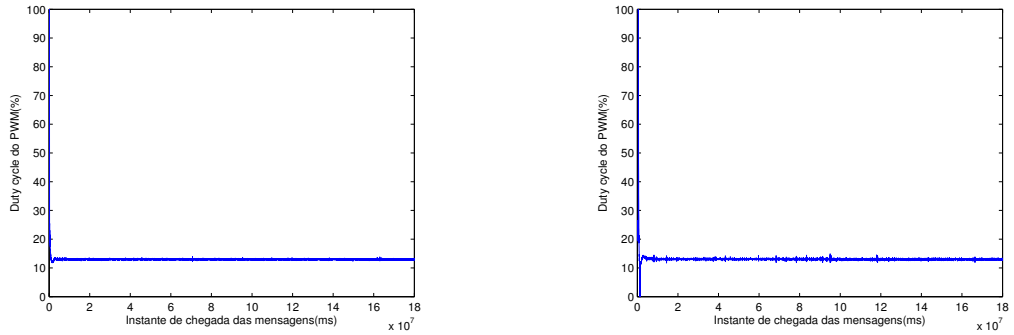
(a) sem interferência (b) com interferência

Figura 4.9: Velocidade do eixo 1 em rotações por segundo



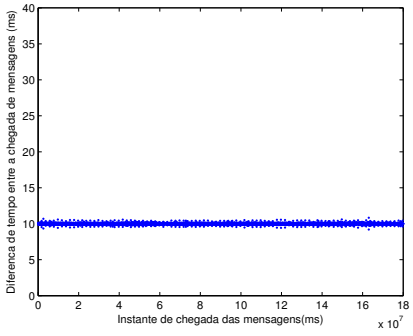
(a) sem interferência (b) com interferência

Figura 4.10: Erro da velocidade angular em percentagem

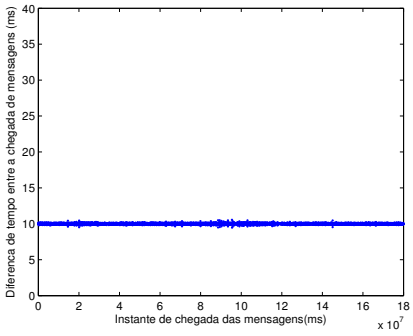


(a) sem interferência (b) com interferência

Figura 4.11: Sinal de controlo (percentagem de PWM) do eixo 1 em percentagem

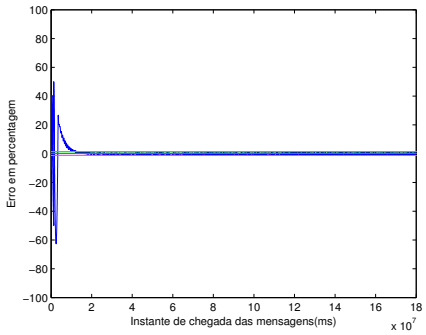


(a) sem interferência

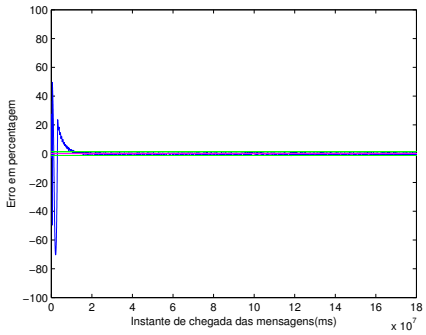


(b) com interferência

Figura 4.12: Diferença de tempo entre a chegada de mensagens do eixo 1



(a) sem interferência



(b) com interferência

Figura 4.13: Percentagem de erro da fase do eixo 2 face ao eixo 1

Eixo 2 e 3

Tal como referido anteriormente, este teste foi efectuado só com o eixo 2.

Na figura 4.13 está representada a percentagem do erro da fase do eixo 2 face ao eixo 1 e quando a mesma é inferior a 1.3% (linhas verdes). No caso em que não há interferência, o sistema tem um tempo de estabelecimento de 22 segundos e no caso em que há interferência existe um tempo de estabelecimento de 15 segundos. Em ambas as situações, o erro percentual médio é de 0.3%.

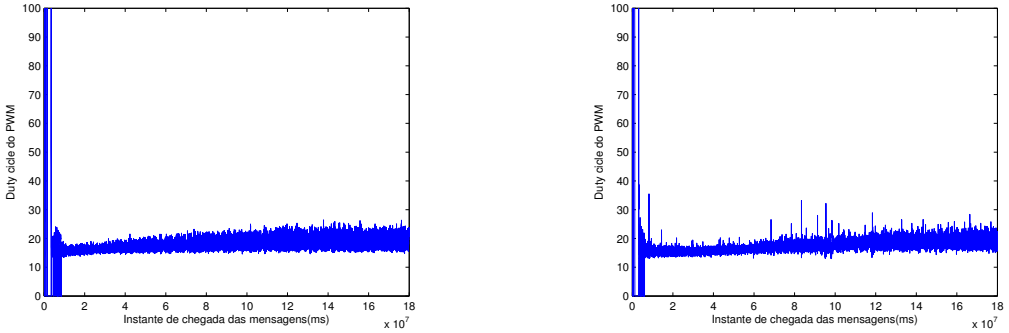
Através da figura 4.14b verifica-se que o sistema está perfeitamente estável.

Na figura 4.15 está representado a diferença de tempo entre a chegada das mensagens.

Na situação em que não há tráfego de interferência (figura 4.15a), o *jitter* médio é de 48.7 microsegundos com um *jitter* máximo de 462 microsegundos. Na situação em que há tráfego de interferência (figura 4.15b), o *jitter* médio é de 39.8 microsegundos com um *jitter* máximo de 648 microsegundos.

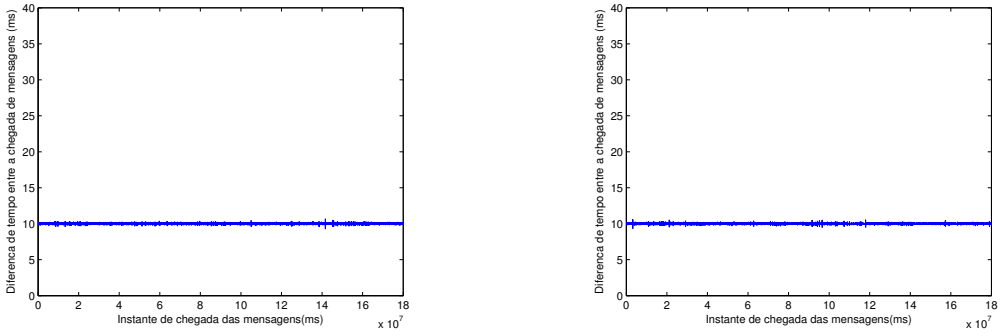
4.5 Comparação dos resultados dos switches

Comparando os resultados do sistema com os dois *switches* é possível verificar que, sem o tráfego de interferência, o *switch* COTS tem um *overshoot* de 87%, um tempo de estabe-



(a) sem interferência (b) com interferência

Figura 4.14: Sinal de controlo (percentagem de PWM) do eixo 2 em percentagem



(a) sem interferência (b) com interferência

Figura 4.15: Diferença de tempo entre a chegada de mensagens do eixo 2

lecimento de 38 segundos e um *jitter* médio de 54.6 microsegundos (eixo 1) e de 41.6 microsegundos (eixo 2), enquanto que o *switch* HaRTES tem um *overshoot* de 25%, um tempo de estabelecimento de 33 segundos e um *jitter* médio de 40.2 microsegundos (eixo 1) e de 48.7 microsegundos (eixo 2). No teste em que há interferência, o *switch* COTS vai perder mensagens enquanto que o enquanto que o *switch* HaRTES tem um *overshoot* de 97%, um tempo de estabelecimento de 47 segundos e um *jitter* médio de 48.7 microsegundos (eixo 1) e de 39.8 microsegundos (eixo 2). Desta forma, à parte do *overshoot*, o *switch* HaRTES apresenta um comportamento semelhante com e sem tráfego de interferência. Isto deve-se ao facto de este *switch* efectuar uma segregação do tráfego de controlo do tráfego de interferência, sendo cada um confinado a um servidor dentro do *switch*. O *switch* COTS por sua vez, não apresenta garantias de qualidade de serviço, tendo, com a presença do tráfego de interferência, perdido mensagens de tráfego de controlo. A diferença dos valores de *overshoot* deve-se ao facto do ponto de início de funcionamento em cada um dos testes ter sido diferente. Num dos testes, a diferença de fases inicial foi relativamente pequena e o sistema acaba por não ter um *overshoot* elevado e no outro a diferença de fases inicial é maior e por isso o sistema tem um *overshoot* maior.

Capítulo 5

Conclusões e Trabalho Futuro

Neste capítulo é feita uma apreciação global do trabalho realizado tendo em conta os resultados obtidos e a realização dos objectivos propostos. No final são indicadas propostas para trabalho futuro.

Foram identificados e adquiridos módulos que são adequados para servir como demonstradores para o *switch* Ethernet de Tempo Real.

O demonstrador de sistemas industriais distribuídos foi devidamente adaptado, tendo sido criados os *device drivers*, *hardware* de interface e *software* de controlo necessários ao trabalho do mesmo.

Foi criada uma aplicação de demonstração que mostra o correcto funcionamento do demonstrador e as potencialidades de uso do mesmo como demonstrador para o *switch* Ethernet de Tempo Real.

A estrutura criada é bastante modular permitindo a fácil adaptação para utilização com diversos protocolos de redes de comunicação a testar ou para introdução de novas fontes de tráfego.

Foi testado o sistema de comunicação baseado no *switch* HaRTES e comprovadas as suas mais valias no que diz respeito à segregação de tráfego.

Como trabalho futuro, poderão ser introduzidas melhorias a este sistema, por exemplo, fazendo com que as Gumstix passem a ter um relógio comum, deixando assim de ser necessário o estimador no controlo. Será ainda necessário adaptar as restantes plataformas e criar aplicações de teste.

Apêndice A

Ethernet de tempo real

Este anexo contém a descrição de alguns dos protocolos *Ethernet* tempo real.

A.1 EtherNet/IP

EtherNet/IP é um protocolo de comunicação desenvolvido pela *Rockwell Automation* e mantido pela *Open Device Vendors Association* (ODVA) para utilização em aplicações de automação industrial que consigam suportar alguma quantidade de não determinismo [16].

O protocolo EtherNet/IP é membro da família de redes que implementa o CIP nas suas camadas superiores. O CIP engloba uma vasta gama de serviços e mensagens para um grupo de aplicações de automação – controlo, segurança, sincronização, movimento, configuração e informação. Os pacotes de informação CIP são encapsulados em telegramas TCP ou UDP sendo depois enviados através de *Ethernet*. Durante a fase de instalação da rede, cada nó *Ethernet* é definido como sendo de um tipo específico previamente definido e que tem um comportamento específico através de ficheiros EDS (*electronic device datasheets*) [26].

O EtherNet/IP usa as mesmas tecnologias que a *Ethernet* tradicional sendo portanto compatível com os equipamentos *standard*.

A.2 AVB

As redes AVB consistem num conjunto de protocolos desenvolvidos pelo grupo de trabalho de *Bridging Audio/Video* IEEE 802.1. O objectivo deste grupo é providenciar as especificações que vão permitir fluxos de audio e vídeo com baixa latência sincronizados no tempo em redes 802 [17]. Para permitir fluxos de baixa latência sincronizados no tempo, o protocolo implementa:

- sincronização precisa;
- modelamento de tráfego para aplicações media;
- controlo de admissão;
- identificação de aparelho que participam rede AVB.

Para realizar a sincronização, dispositivos AVB trocam mensagens com informação de tempo, o que permite aos nós sincronizar de forma precisa o relógio de referência. Isto permite a sincronização de múltiplos fluxos e ainda a existência de tempos comuns de amostragem/recepção de informação. É usado o relógio de um nó como sendo *master*, servindo como referência para todos os restantes.

As redes AVB possuem duas classes dependendo dos requisitos de latência (figura A.1):

- **Classe A** - máximo de $2ms$ de latência;
- **Classe B** - máximo de $50ms$ de latência ao longo de 7 saltos.

Para além destas classes, existe ainda um mecanismo *best effort* para as tramas provenientes de redes não AVB.

A transmissão de tramas é controlada por um sistema modulador baseado em créditos. Uma trama é enviada se houver créditos disponíveis, sendo os créditos atribuídos por critérios temporais. O modulador limita o número de créditos disponíveis de tal forma que a largura de banda seja limitada e sejam evitadas rajadas de informação que poderiam levar à perca de informação no caso de os *buffers* dos dispositivos ficarem completamente preenchidos. No caso de nós não-AVB transmitirem mensagens, a prioridade das mesmas é alterada de forma a cumprir o critério *best effort*. Desta forma os tráfego AVB tem sempre a prioridade mais alta.

É realizado controlo de admissão, sendo que um novo fluxo só é aceite na rede depois de serem reservados os recursos necessários ao mesmo. Para realizar estas reservas, são trocadas mensagens entre os nós emissor, receptor e ainda os nós que ficam no caminho do tráfego.

Uma das vantagens desta rede é utilizar *bridges* apenas ligeiramente alteradas, de forma a que aparelhos alterados possam comunicar com aparelhos não alterados. É implementado um sistema de identificação dos dispositivos participantes na rede AVB.

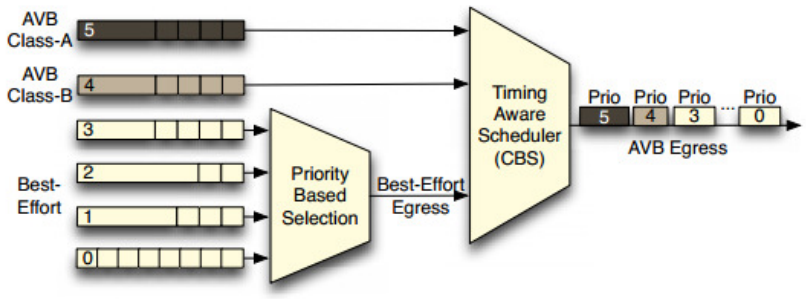


Figura A.1: Classes de tráfego [1]

A.3 PROFINET

PROFINET é um protocolo de *Ethernet* desenvolvido pela PROFIBUS & PROFINET *International* para automação industrial. Este protocolo usa *switched ethernet* com as vantagens apresentadas anteriormente, permitindo transmissão de informação determinística e isocronamente[18].

O PROFINET apresenta uma estrutura modular composta por duas perspectivas chamadas PROFINET CBA e PROFINET IO (figura A.2). O PROFINET CBA (*Component*

based automation) é utilizado para comunicações M2M baseada em componentes utilizando TCP/IP. O PROFINET IO é utilizado para troca de informação entre arquitecturas distribuídas estando dividido em dois modos: comunicações de RT e de IRT, sendo que o modo RT é utilizado por comunicação de tempo realcom requisitos *soft*, e o modo IRT por comunicações de tempo real com requisitos *hard*.

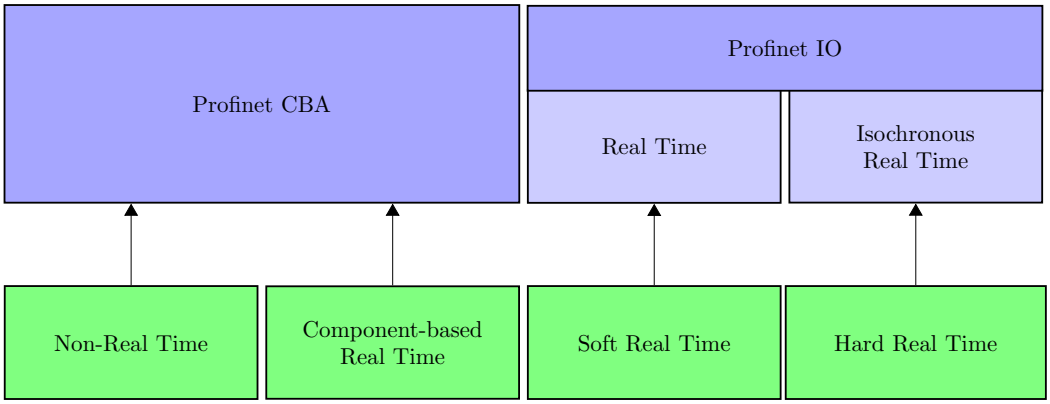


Figura A.2: Estrutura modular PROFINET

Com o PROFINET IO, a informação é sempre transmitida em RT sendo este modo a base para troca de informação. Neste modo, a duração dos ciclos de barramento está na ordem das centenas de milisegundos. No caso do modo de IRT, a informação é trocada em ciclos de barramento cujo início é muito preciso. A duração dos ciclos de barramento pode chegar até um milisegundo tendo o início do ciclo de barramento um *jitter* máximo de um microsegundo. Este modo é útil para sistemas que necessitam de sincronização submilissegundo como é o caso dos sistemas de controlo de movimento. Para conseguir os tempos de precisão e *jitter* necessários, foram desenvolvidas soluções de hardware com nós Ethernet sincronizados.

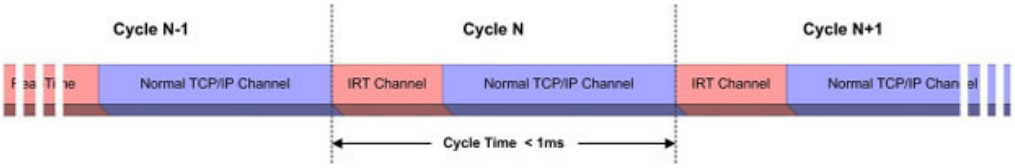


Figura A.3: Ciclos de comunicação PROFINET [2]

Cada ciclo de comunicação engloba um período de comunicação isócrono seguido por um período em que o canal fica aberto para o restante tráfego sendo usado através de TCP/IP (figura A.3).

A.4 TTEthernet

TTEthernet é um protocolo de comunicação desenvolvido pela Universidade de Tecnologia de Viena e pela *TTTech Computertechnick* sendo compatível com os equipamentos de *Ethernet*

tradicionais. Este protocolo aberto de SW de tempo real é usado em aplicações de segurança em transportes ou em automação industrial.

Esta rede possui as seguintes características [19]:

- compatibilidade, como já tinha sido referido;
- escalibilidade: existem várias possibilidades de escolha de topologias e velocidades de trabalho, sendo ainda possível adoptar redundância de ligações;
- sincronização e determinismo: este protocolo implemente um sistema de relógio *multi-master* distribuído e tolerante a falhas, com uma sincronização superior a um microsegundo o que garante qualidade de serviço e permite determinismo.
- protecção e disponibilidade: a redundância de ligações ao nível da camada de rede permite que a comunicação seja replicada garantindo a transmissão das mensagens. Isto é feito sem adicionar atrasos com comutações o que aumenta a disponibilidade;
- segurança: dado as características anteriores realizadas em tempo real, não terem intervenção do utilizador, são seguras.

Este protocolo foi desenvolvido de forma a permitir a existência de comunicações *time-triggered* em *ethernet*. Estas comunicações co-existem com o restante tráfego. Isto é feito com recurso a um controlador das comunicações *time-triggered* que é capaz de sincronizar o tráfego com os restantes controladores e *switches*. Este controlador envia mensagens em instantes determinados através da sincronização com o restante tráfego. Neste protocolo o tráfego *time-triggered* é tratado como tendo requisitos de tempo real *hard* tendo precedência sobre todos os outros tipos de mensagens. Tráfego *event-triggered* é tratado como não crítico. Existem 3 tipos de comunicação dentro deste protocolo[27]:

- *Time-triggered*: Este tipo de comunicação segue o *standard* SAE 6802. Todos os elementos da rede sabem qual informação que devem transportar e em que instante o devem fazer, o que evita o envio de mensagens de *acknowledge*. Desta forma a largura de banda é usada de forma óptima. Aplicações de tempo real devem utilizar este tipo de comunicação.
- *Rate-constrained*: Este tipo de comunicação segue o *standard* ARINC 664. No caso de num determinado momento do tempo não existirem comunicações *time-triggered* agendadas, podem ser efectuadas comunicações *rate-constrained* com largura de banda predefinida. É reservada largura de banda na rede para este tipo de comunicação. Este tipo de comunicação garante que uma quantidade determinada de informação é transmitida, tornando-se ideal para aplicações com requisitos de tempo menos restritos como é o caso das aplicações audiovisuais.
- *Best-effort*: Utiliza o protocolo *ethernet* tradicional. A informação é enviada nos intervalos entre a transmissão das comunicações *time-triggered* e *rate-constrained*. Não é portanto possível garantir qualquer tipo de qualidade de serviço, devendo ser usada somente por tráfego que não tenha requisitos temporais e cujas falhas não sejam críticas.

A.5 FTT-SE

De seguida é apresentada a implementação do protocolo FTT-SE.

A.5.1 Implementação do protocolo

Trama

A trama FTT-SE tem os seguintes campos(figura A.4) [28]:

- *Preamble*: usado para sincronização;
- *Start of Frame*: indica o início da trama útil;
- *Destination Adress*: indica o endereço do nó destino;
- *Source Adress*: indica o endereço do nó origem;
- *Type*: indica o tipo de mensagem FTT;
- *FTT-Ethernet PDU*: campo de dados;
- *Padding*: bits de enchimento caso a mensagem seja inferior ao mínimo necessário;
- *FCS*: detecção e correcção de erros.

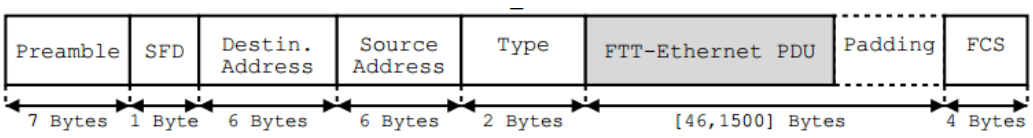


Figura A.4: Formato da Trama FTT-SE

Tipos de mensagens

Existem 5 possíveis tipos de mensagens neste protocolo:

- TM;
- mensagem síncrona de dados;
- mensagem assíncrona de dados;
- mensagem de controlo;
- mensagem *ethernet* tradicional.

Trigger Message Formato da mensagem TM (figura A.5):

- *Type*: campo com o identificador de TM e a identificação do *master*;
- *TM FLAGS*: contém um espaço reservado e o número de sequência;
- *Number of Sincronous Messages*: número de mensagens síncronas nesse *Elementary Cycle*;
- *ID + Tx Time*: identificação das mensagens síncronas e momentos de acesso ao meio.

Type		TM Flags		Num.	ID	Tx	...
TM Type	Master ID	Reserv.	Seq. Num.	Synch. Mesgs		Time	
2 Bytes		2 Bytes		2 Bytes	2 Bytes	1 Byte	...
[b15..b12]	[b11..b0]	Undef.	[b7..b0]	[b15..b0]	[b15..b0]	[b7..b0]	...
TM_MESG_ID	0 to 4096	Undef.	0 to 256	0 to 65535	0 to 65535	0 to 256	...

Figura A.5: Formato da Trama da *Trigger Message*

Mensagem Síncrona Formato da mensagem síncrona (figura A.6):

- *Type*: campo com o identificador da mensagem síncrona e a identificação do nó que envia a mensagem;
- *SDM FLAGS*: contém um espaço reservado e o número de sequência;
- *Time to Deadline*: identificação temporal de duração dos dados enviados;
- *Data*: informação.

Type		SDM Flags		Time to	Message
SDM Type	SDM ID	Reserved	Seq. Num.	Deadline	Data
2 Bytes		2 Bytes		2 Bytes	up to
[b15..b12]	[b11..b0]	Undef.	[b7..b0]	[b15..b0]	1494
DATA_MESG_ID	0 to 4096	Undef.	0 to 256	0 to 65535	Bytes

Figura A.6: Formato da Trama da mensagem síncrona

Mensagem Assíncrona Formato da mensagem assíncrona (figura A.7):

- *Type*: campo com o identificador da mensagem assíncrona e a identificação do nó que envia a mensagem;
- *ADM FLAGS*: contém um espaço reservado e o número de sequência;
- *Time to Deadline*: identificação temporal de duração dos dados enviados;
- *Data*: informação.

Mensagem de Controlo Formato da mensagem de controlo (figura A.8):

- *Type*: campo com o identificador da mensagem de controlo e a identificação do nó que envia a mensagem;

Type		SDM Flags		Time to	Message
ADM Type	ADM ID	Reserv.	Seq. Num.	Deadline	Data
2 Bytes		2 Bytes		2 Bytes	up to
[b15..b12]	[b11..b0]	Undef.	[b7..b0]	[b15..b0]	1494
AM_DATA_MESG_ID	0 to 4096	Undef.	0 to 256	0 to 65535	Bytes

Figura A.7: Formato da Trama da mensagem assíncrona

- *ADM FLAGS*: contém um espaço reservado e o número de sequência;
- *Time to Deadline*: identificação temporal de duração dos dados enviados;
- *Data*: informação.

Type		SDM Flags		Time to	Message
CM Type	CM ID	Reserv.	Seq. Num.	Deadline	Data
2 Bytes		2 Bytes		2 Bytes	up to
[b15..b12]	[b11..b0]	Undef.	[b7..b0]	[b15..b0]	1494
CONTROL_MESG_ID	0 to 4096	Undef.	0 to 256	0 to 65535	Bytes

Figura A.8: Formato da Trama de controlo

Bibliografia

[1] T. Steinbach, F. Korf, T.C. Schmidt, D. Herrscher, and A. Wolisz. A competitive evaluation of ieee 802.1 avb and time-triggered ethernet. In *Vehicular Technology Conference (VTC Fall)*, pages 1 – 5, 2012.

[2] Industrial ethernet university:real time ethernet. http://www.industrialethernetu.com/courses/402_4.htm.

[3] Katie Hafner. Honey, i programmed the blanket. *The New York Times*, 1999.

[4] Jan Axelson. *Embedded Ethernet and Internet Complete*. Lakeview Research, 2003.

[5] J. D. Decotignie. A perspective on ethernet-tcp/ip as a fieldbus. In *IFAC International Conference on Fieldbus Systems and their Applications*.

[6] Antonio Pereira de Melo. *Teoria dos sistemas de controlo lineares*. Universidade de Aveiro, 2011.

[7] John Joachim D'Azzo, Constantine H. Houpins, and Stuart N. Sheldon. *Linear Control System Analysis and Design with Matlab*. CRC Press, 2003.

[8] Adrian Noguero, Isidro Calvo, and Luis Almeida. *A Time-Triggered Middleware Architecture for Ubiquitous Cyber Physical System Applications*. Springer Berlin Heidelberg, 2012.

[9] Hermman Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1997.

[10] José Javier Gutiérrez García and Michael González Harbour. Minimizing the effects of jitter in distributed hard real-time systems. *Journal of Systems Architecture*, 1996.

[11] Michael Barr. Embedded systems glossary, Nov 2007. http://www.barrgroup.com/Embedded-Systems.

[12] Andrew Tanenbaum. *Operating Systems: Design and Implementation*. Prentice-Hall, 1987.

[13] Michael Beck. *Ethernet in the first mile*. McGraw-Hill, 2005.

[14] Charles Spurgeon. *Ethernet: The Definitive Guide*. O'Reilly and Associates, 2000.

[15] Paulo Pedreiras, Paolo Gai, Luis Almeida, and Giorgio Buttazzo. Ftt-ethernet: A flexible real-time communication protocol that supports dynamic qos management on ethernet-based systems. In *IEEE Transactions on Industrial Informatics*, Aug 2005.

[16] ODVA. *EtherNet/IP Quick Start for Vendors Handbook: A Guide for EtherNet/IP Developers*, 2008.

[17] Michael Teener. *No-excuses Audio/Video Networing: the Technology behind AVnu.* AVnu, 2009.

[18] PROFIBUS Nutzerorganisation. *PROFINET: System Description*, 2009.

[19] Ttethernet: Deterministic ethernet network. http://www.tttech.com/ technologies/ttethernet/.

[20] Rui Gabriel Viegas dos Santos. *Enhanced Ethernet Switching Tecnology for Adaptibe Hard Real-Time Applications.* PhD thesis, Universidade de Aveiro, 2010.

[21] Luís Emanuel Moutinho da Silva. Ligação de alto desempenho entre fpgas para switch ethernet ftt. Master's thesis, Universidade de Aveiro, 2010.

[22] Level and flow control system brochure. http://www.feedback-instruments.com/pdf/brochures/38-003_datasheet_temperature+Level_flow_control_04_2013.pdf.

[23] Ball and plate control system brochure. http://www.feedback-http://www.feedback-instruments.com/pdf/brochures/33-052_Ball_Plate_datsheet.pdf.

[24] Self erected inverted pendulum product information sheet. http://www.quanser.com/english/downloads/products/Linear/IP02_PIS_031108.pdf.

[25] Network controled inverted pendulum experimental manual. http://www.googoltech.com/uploads/catalog/569/GLIP

[26] Ethernet/ip. http://www.hms.se/technologies/ethernetip.shtml, 2009.

[27] TTTech Computertechnik AG. *TTEthernet: A Powerful Network Solution for All Purposes*, 2011.

[28] Paulo Bacelar Reis Pedreiras. *Suporte de Comunicações Tempo - Real Flexíveis em Sistemas Distribuídos.* PhD thesis, Universidade de Aveiro, 2003.