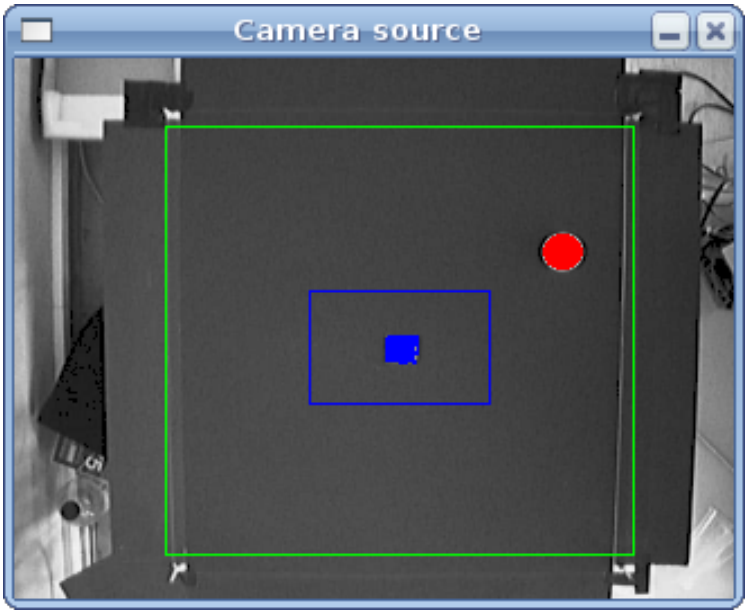
Departamento de
Universidade de Aveiro Electrónica, Telecomunicações e Informática
2008

Nuno Miguel de Melo Figueiredo

Controlo Distribuído de Plataformas para Experiências de Mecatrónica



Nuno Miguel de Melo Figueiredo

Controlo Distribuído de Plataformas para Experiências de Mecatrónica

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Electrónica e Telecomunicações, realizada sob a orientação científica do Professor Doutor Paulo Pedreiras, Professor Convidado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro, co-orientação científica do Professor Doutor Ernesto Martins, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro e colaboração do Professor Doutor Luís Almeida, Professor Auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

aos meus pais
aos meus irmãos
à minha família
aos meus amigos

O Júri

Presidente

Doutor Valeri Skliarov
Professor Catedrático da Universidade de Aveiro

Vogais

Doutor Luis Miguel Moreira Lino Ferreira
Equiparado a Professor Adjunto do Departamento de Engenharia Informática do Instituto Superior de Engenharia do Porto

Doutor Paulo Bacelar Reis Pedreiras (Orientador)
Professor Auxiliar Convidado da Universidade de Aveiro

Doutor Ernesto Fernando Ventura Martins (Co-orientador)
Professor Auxiliar da Universidade de Aveiro

Agradecimentos

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

a Paulo Bacelar Reis Pedreiras pela dedicação e empenho que sempre demonstrou ao longo de todo o trabalho desenvolvido. Pela exemplar orientação científica e técnica que demonstrou no decorrer do tempo. Não posso também deixar de referir o carácter humano que proporcionou um excelente ambiente de trabalho.

a Ernesto Fernando Ventura Martins também pelo empenho e dedicação que pôs neste projecto. A exemplar co-orientação científica e técnica foi uma constante no decorrer da preparação desta dissertação. Gostaria também de salientar o seu carácter humano que mais uma vez proporcionou um excelente ambiente de trabalho.

a Luís Miguel Pinho de Almeida pela sua colaboração neste projecto sempre com empenho e dedicação. De salientar também o seu carácter humano, sempre alegre e descontraído.

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

a Ricardo Marau pela dedicação e empenho que demonstrou no projecto. Ao Ricardo gostaria de agradecer em especial a sempre prestável ajuda que nunca negou, a disponibilidade incondicional e a excelente qualidade técnica e científica que sempre demonstrou. De referir também a sua excelente qualidade como ser humano o que ajudou também a que este trabalho decorresse sempre com grande alegria e descontracção.

a Petia Georgieva Georgieva pela disponibilidade que sempre demonstrou e pela excelente ajuda técnica e científica que prestou no decorrer desta dissertação.

ao Laboratório de Sistemas Electrónicos da Universidade de Aveiro por me ter acolhido e integrado num grupo de trabalho sempre prestável e de excelente qualidade.

a todos os que estiveram envolvidos nesta dissertação

Obrigado

Palavras Chave

Controlo, Distribuído, Mecatrónica, Redes Dedicadas, Ethernet, Sistemas, Bola na Calha, Bola no Plano, Tempo-Real, FTT Ethernet

Resumo

Os Sistemas Distribuídos encontram-se em variadas aplicações do dia-a-dia abrangendo áreas como a aviónica, robótica, automação industrial e automotiva. Um Sistema de Controlo Distribuído implica que vários componentes do sistema troquem informação entre si com o propósito de atingir um objectivo comum. Essa informação é, geralmente, trocada recorrendo a uma rede de comunicação que interliga todos os componentes intervenientes no sistema de controlo. A introdução desta rede no sistema levanta problemas do ponto de vista do comportamento temporal e funcional do sistema pois introduz latência e jitter adicionais na malha de controlo. Muitas aplicações são sensíveis à latência e ao jitter pelo que estes aspectos poderão conduzir à degradação do desempenho do controlo. Esta observação levou ao desenvolvimento de protocolos específicos com propriedades de tempo-real.

Esta dissertação tem como objectivo avaliar experimentalmente o desempenho do protocolo de tempo-real FTT-SE desenvolvido no Laboratório de Sistemas Electrónicos da Universidade de Aveiro. Aspectos como o Isolamento Temporal, Garantias de Reserva de Recursos, Escalonamento das Mensagens e Gestão da Carga na Rede serão tidos em especial consideração. O desempenho do protocolo FTT-SE vai também ser comparado com o desempenho do protocolo Ethernet em situações de utilização similares, destacando os problemas do uso de um protocolo sem propriedades de tempo-real.

A avaliação experimental do protocolo FTT-SE é baseada nas plataformas de mecatrónica "Bola no Plano" e "Bola na Calha" anteriormente desenvolvidas no Laboratório de Sistemas Electrónicos da Universidade de Aveiro. Foram realizadas duas implementações, uma baseada em Ethernet e outra em FTT-SE. Cada implementação foi submetida a diferentes condições de carga e o desempenho foi avaliado tendo em conta diferentes critérios. Os testes realizados permitem observar as interferências que a rede induz nos sistemas físicos bem como avaliar a eficácia dos mecanismos de tempo-real do protocolo FTT-SE.

Key Words

Control, Distributed, Mechatronics, FieldBuses, Ethernet, Systems, Ball on Beam, Ball on Plate, Real-Time, FTT Ethernet

Abstract

Distributed systems are becoming a commonplace, being found in many daily life application areas such as avionics, robotics, industrial automation and automotive. A distributed control system implies that several components of the system have to exchange information among themselves to be able to achieve a common goal. This information is usually exchanged through a communication network joining together those components. The use of the network in itself can pose problems in terms of time and functional behavior of the systems, since it may introduce additional latency and jitter in control loop. Many applications are latency and jitter sensitive and thus the introduction of the network may result in noticeable degradation of performance of control. This observation led to the development of specific protocols with real-time properties.

The aim of this dissertation is to assess experimentally the performance of the FTT-SE real-time protocol, developed in the Electronic Systems Laboratory of the University of Aveiro. Particular attention is devoted to aspects such as temporal isolation, resource reservation guarantees, message scheduling and overload handling. The performance of FTT-SE will also be compared against the use of plain Ethernet in similar situations, highlighting the problems of using non real-time protocols.

The experimental assessment of the FTT-SE protocol is based on the mechatronic platforms " Ball on Plate" and "Ball on Beam" previously developed at the Electronic Systems Laboratory of the University of Aveiro. Two implementations, one based in plain Ethernet and another in FTT-SE, have been realized. Each one is submitted to different load conditions and the control performance assessed according with diverse criteria. These experiments permit observing the impact of network induced perturbations in real-world systems as well as to assess the effectiveness of real-time mechanisms provided by the FTT-SE protocol.

Conteúdo

Lista de Figuras

Lista de Tabelas

Parte I

Introdução

Capítulo 1

Enquadramento e Motivação

Neste capítulo é feita uma introdução ao objecto da dissertação focando o trabalho já realizado em anos anteriores, os objectivos da dissertação e as contribuições que esta trouxe para a área de trabalho em que foi desenvolvida, ou seja, a aplicação de protocolos de comunicação a Sistemas de Controlo Distribuído em Tempo-Real.

No final deste capítulo será descrita detalhadamente a estrutura deste documento.

1.1 Descrição da Dissertação

Sistemas de mecatrónica, sistemas mecânicos e electrónicos, encontram-se em diversificadas aplicações do dia-a-dia em áreas como a robótica, automação, aviação, indústria automóvel, entre outras.

Com o aumento da produção deste tipo de sistemas tem-se verificado que, cada vez mais, se opta por soluções distribuídas, ou seja, compostas por diversos componentes dotados de processameto local que comunicam entre si utilizando uma rede de comunicação. Este tipo de solução apresenta vantagens tais como a redução e simplificação da cablagem, redução da complexidade do sistema, diagnóstico e manutenção simplificada. No entanto a utilização da rede de comunicação agrava os problemas de atrasos e *jitter* quer de amostragem quer de actuação.

Para resolver os problemas criados com a introdução da rede no sistema surgiram os protocolos de comunicação em Tempo-Real. Um exemplo deste tipo de protocolo é o FTT-SE[Ped] desenvolvido no Laboratório de Sistemas Electrónicos da Universidade de Aveiro.

Para estudar o impacto que a rede tem sobre o desempenho do sistema foram elaboradas as plataformas mecatrónicas "Bola na Calha" e "Bola no Plano". O trabalho realizado nesta dissertação visa implementar o controlo distribuído usando para o efeito a rede Ethernet com e sem protocolo FTT-SE. Pretende-se avaliar que impacto tem o uso de protocolos de Tempo-Real no controlo das plataformas citadas.

1.1.1 Trabalho Anterior

As plataformas utilizadas nesta dissertação foram desenvolvidas no âmbito de dois projectos de final de curso. O primeiro[NL04], nos anos 2003/2004, visou essencialmente a construção das plataformas e o controlo destas recorrendo aos microcontroladores Microchip©

18F258. Neste trabalho foi também implementada uma monitorização das plataformas a partir de uma aplicação no computador usando para o efeito uma interface via CAN.

No segundo trabalho [Roq07], realizado nos anos 2006/2007, todas as plataformas foram revistas sendo melhorado o hardware e o controlo. À semelhança do trabalho anterior a este, o controlo baseou-se no processamento local do microcontrolador já referido. Na plataforma Bola no Plano foi alterado o sensor para a câmara Philips PCVC740K, anteriormente era usada uma câmara CMU.

1.1.2 Objectivos

Os objectivos propostos para esta dissertação foram os seguintes:

- Familiarização com sistemas mecatrónicos e com arquitecturas de controlo distribuídas;
- Modificação da interface electrónica das plataformas para permitir o controlo baseado em PC;
- Controlo local dos *set-ups* baseado em PC;
- Estudo da rede Ethernet;
- Distribuição da malha de controlo usando a rede Ethernet;
- Estudo do protocolo FTT-SE;
- Distribuição da malha de controlo usando o protocolo FTT-SE
- Avaliação do desempenho do controlo em função da carga na rede;
- Avaliação das funcionalidades de gestão de QoS do protocolo FTT-SE;
- Implementação e avaliação de diferentes estratégias de controlo.

1.1.3 Contribuição

Esta dissertação apresenta um estudo à viabilidade da aplicação da rede Ethernet no controlo de sistemas, rede não adequada a este tipo de aplicações. Algum trabalho tem sido feito neste sentido [DHL$^+$05][MAP06], esta dissertação apresenta um conjunto de ferramentas que permite aos futuros projectistas optar pela melhor forma de distribuir um sistema bem como mostrar uma aplicação concreta e funcional onde se possam basear para desenvolver este tipo de aplicações.

O trabalho realizado nesta dissertação vai possibilitar o teste do protocolo FTT-SE numa situação concreta de controlo e verificar as propriedades e garantias que este protocolo fornece à rede de comunicação Ethernet.

1.2 Estrutura da Dissertação

Esta dissertação apresenta-se dividida em quatro partes fundamentais:

- *Introdução;*
- *Implementação;*
- *Resultados;*
- *Conclusão.*

A parte da Introdução contém os primeiros três capítulos apresentando os conceitos teóricos que servem se suporte ao trabalho realizado nesta dissertação. Em concreto nos três primeiros capítulos estão focados os seguintes assuntos:

- *Capítulo 1 - Enquadramento e Motivação*

  Neste capítulo é feita uma introdução à dissertação fazendo uma breve análise ao problema e à motivação que levou à elaboração desta dissertação. São focados os trabalhos já realizados e descritas quais as contribuições que esta dissertação trouxe à área onde foi desenvolvida.

- *Capítulo 2 - Conceitos Fundamentais*

  Este capítulo encontra-se dividido em 3 secções principais. Numa secção inicial é feita uma breve introdução aos sistemas físicos passando depois para a descrição das diferentes formas e métodos existentes de controlo. Na segunda secção são introduzidos os sistemas de Tempo-Real tendo em conta as características próprias deste tipo de sistemas computacionais. Na terceira e última secção são introduzidos os sistemas de controlo distribuído tendo em conta as vantagens e desvantagens deste tipo de solução, passando pelos problemas que levanta a distribuição do sistema bem como o impacto que uma rede de comunicação tem no desempenho do controlo do sistema.

- *Capítulo 3 - Redes de Comunicação em Sistemas Distribuídos*

  Este capítulo começa por analisar as diferentes redes usadas para distribuir o controlo nos sistemas. Após esta análise inicial é apresentada a rede Ethernet descrevendo as suas principais características e propriedades. No seguimento desta descrição são apresentados alguns protocolos existentes que permitem fornecer à referida rede propriedades de Tempo-Real. Seguidamenta é apresentado o paradigma FTT e para concluir o capítulo é analisado o protocolo FTT-SE.

A parte dedicada à Implementação contém os 3 capítulos seguintes onde é explicada, de forma detalhada, a metodologia usada para concretizar esta dissertação. Esta parte encontra-se dividida nos seguintes capítulos:

- *Capítulo 4 - Análise dos Sistemas Físicos Usados*

  Neste capítulo é feita uma análise das plataformas usadas tendo em contas as suas principais características e propriedades. Seguidamente é feito um estudo de modelação matemática do sistema físico e projectado um controlador para esse modelo. No final deste capítulo é detalhada a forma como o controlo foi distribuído nas plataformas usadas nesta dissertação.

- *Capítulo 5 - Plataforma Bola na Calha*

  Este capítulo detalha os pormenores de implementação desta plataforma, apresenta a arquitectura geral do sistema e a arquitectura de hardware particularizando as alterações que foram implementadas. Por último descreve toda a estrutura de software implementado.

- *Capítulo 6 - Plataforma Bola no Plano*

  Este capítulo detalha os pormenores de implementação desta plataforma. Primeiramente apresenta a arquitectura do sistema passando de seguida a apresentar os módulos comuns a todo o projecto desta plataforma, detalhando também a forma como a imagem da câmara foi processada. Seguidamente apresenta a implementação em "Raw Ethernet" focando a estrutura do software e explicando detalhadamente a função de cada módulo de rede e a forma como este foi implementado. Por último apresenta a implementação do protocolo FTT-SE começando por apresentar a nova arquitectura do sistema e detalhando a forma como todos os componentes foram implementados.

O *Capítulo 7*, dedicado aos resultados, começa por descrever os testes realizados às plataformas com os protcolos Raw-Ethernet e FTT-SE. Seguidamente são descritos as características dos componentes que interviram nos testes e as medidas que foram realizadas. Após esta descrição inicial são apresentados e analisados os resultados Raw-Ethernet e FTT-SE. No final do capítulo é feito uma análise comparativa do desempenho dos dois protocolos.

Por último, o *Capítulo 8*, faz a conclusão desta dissertação fazendo uma análise global aos resultados obtidos e aos objectivos do trabalho. No final do capítulo são apresentados alguns tópicos para possível trabalho futuro e feita uma apreciação global do trabalho desenvolvido nesta dissertação.

Capítulo 2

Conceitos Fundamentais

Neste capítulo são introduzidos os conceitos que sustentaram teoricamente o trabalho desenvolvido nesta dissertação.

São introduzidos os conceitos de Sistema Físico e as diferentes formas e métodos de os controlar. O conceito de Sistema de Tempo-Real introduzindo a noção de Tarefa focando as suas características, propriedades, requisitos e restrições. Por último será introduzido o conceito de Controlo Distribuído definindo este tipo de controlo de sistemas, distinguindo controlo cantralizado de controlo distribuído e apresentando modelos de comunicação para e tipo de sistemas.

2.1 Sistemas de Controlo

2.1.1 Definição

Na actividade quotidiana deparamo-nos frequentemente com sistemas de controlo, muitas vezes nós próprios desempenhamos acções que envolvem operações de controlo.

No simples exemplo de beber um copo de água facilmente se constata que este envolve o movimento, de um modo coordenado, de várias partes do nosso corpo, como por exemplo, o braço e a mão. Este movimento comandado pelo cérebro é função de informações enviadas pelos diferentes sentidos que possuímos, entre eles, o tacto e a visão.

A definição de sistema é por isso [DHS03]

- *A combinação de componentes que, em conjunto, actuam para desempenhar uma determinada função que cada componente separadamente não a conseguiria realizar.*

De um modo geral, um sistema, é algo a que se pode associar os conceitos de *causa* e *efeito*.

Existem inúmeros tipos de sistemas [Lei04, HP05], entre eles, mecânicos, hidráulicos, electrónicos, informáticos, térmicos, químicos, económicos e sociais.

Deste ponto de vista, o modelo matemático de um sistema, pretende descrever qual o efeito consequente de uma determinada causa. Alguns critérios, tal como, a estabilidade do sistema, o tempo de estabelecimento, o tempo de subida, entre outros, bem como métodos de modelação, como as transformadas "S", ou transformadas "Z", permitem identificar e classificar os sistemas e descrever o seu comportamento.

A teoria de sistemas de controlo procura modelar matematicamente a interacção *causa-efeito* dos fenómenos físicos, tentando encontrar mecanismos de controlo adequados a cada sistema em particular.

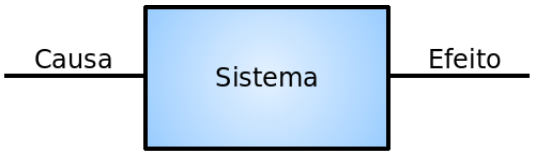


Figura 2.1: Modelo de um Sistema

Um sistema de controlo é por isso

- *Um sistema em que se manipula o elemento causa tendo em vista atingir um efeito desejado.*

Com o aumento da complexidade dos sistemas físicos, bem como a necessidade de lidar com sistemas cada vez mais instáveis, levou à evolução na forma de controlar os sistemas. Este controlo é feito introduzindo mais elementos no sistemas conseguindo assim alterar as suas características tentando alcançar o pretendido comportamento desejado.

2.1.2 Elementos de um Sistema de Controlo

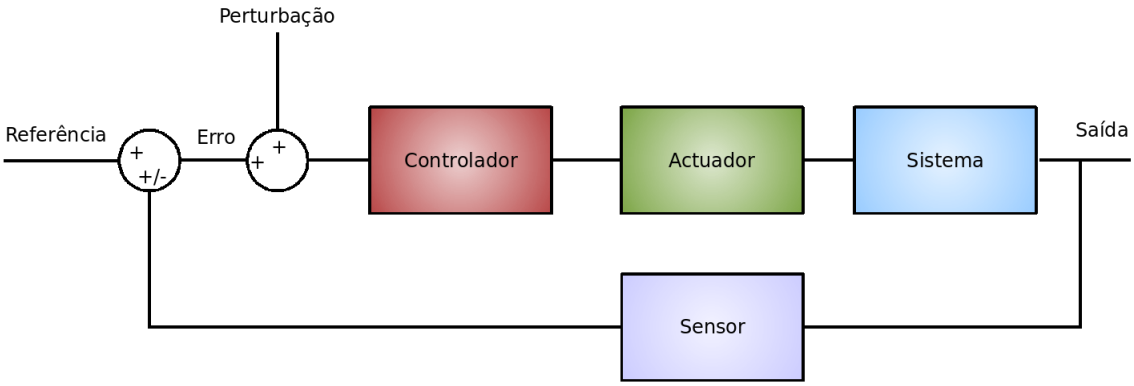


Figura 2.2: Modelo de um Sistema de Controlo

Um sistema de controlo, geralmente, possui os seguintes elementos:

- *Referência* - Sinal de referência do sistema;
- *Perturbação* - Sinal de interferência do sistema, geralmente associado a ruído externo com propriedades aleatórias, que provoca um comportamento não desejado;
- *Erro* - Sinal resultante da comparação entre a Referência e o sinal de 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;

Num sistema de controlo é indispensável que se definam quais as quantidades que se vão controlar e as que se vão observar. Para isso definem-se dois tipos de variáveis:

- *Variáveis de Entrada* - Quantidades que vão ser manipuladas por forma a que o sistema tenha o comportamento desejado;
- *Variáveis de Saída* - Quantidades que deverão ser mantidas nos valores desejados.

2.1.3 Formas de Controlo

Um sistema pode ser controlado das seguintes formas:

- *Controlo em Malha Aberta* [DHS03]

  Sistema de controlo cuja variável de saída não influencia a variável de entrada.

  Esta forma de controlo requer que se conheça muito bem o sistema que se quer controlar bem como os efeitos que o controlador tem sobre o sistema. Por não observar a saída do sistema, este tipo de controlo não tem a capacidade de corrigir possíveis erros de funcionamento, o que faz dele um método inflexível. A grande vantagem deste tipo de controlo é o seu baixo custo e a facilidade de implementação, visto ser apenas necessário um componente adicional, o controlador.

  Um exemplo de aplicação deste tipo de controlo é o controlo da velocidade angular de um motor através da alteração da tensão de alimentação por meio de um potenciómetro. Neste caso cabe ao operador da máquina ajustar o potenciómetro até o motor atingir a rotação desejada.

  A figura 2.3 representa um diagrama de um sistema de controlo em malha aberta.

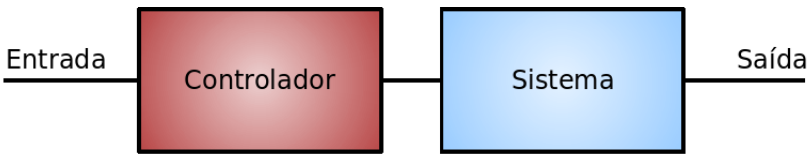


Figura 2.3: Sistema de controlo em malha aberta

- *Controlo em Malha Fechada* [DHS03]

  Sistema de controlo cuja variável de saída influencia directamente a variável de entrada. A este processo dá-se o nome de "Realimentação".

  Existem duas formas de realimentação, a realimentação positiva, em que o valor da saída é somado ao valor de referência, o que faz com que o sistema sature com uma maior

velocidade. Um exemplo do uso deste tipo de realimentação pode ser encontrado nos amplificadores "Schmitt trigger", que são muito usados, entre outras aplicações, para sincronização de relógios. O outro tipo de realimentação é a negativa, em que o valor de saída é subtraído ao valor de referência, dando ao controlador um valor da "distância" a que o sistema se encontra do valor de referência.

A grande desvantagem deste tipo de controlo é a sua complexidade de implementação, geralmente exige mais componentes no sistema, transdutores, processadores, entre outros, o que leva a um muito maior custo de implementação.

Os exemplos de aplicação deste tipo de realimentação são as plataformas usadas neste trabalho, todas elas foram controladas usando sistemas de controlo em malha fechada com realimentação negativa.

A figura 2.4 representa um diagrama de um sistema de controlo em malha fechada.

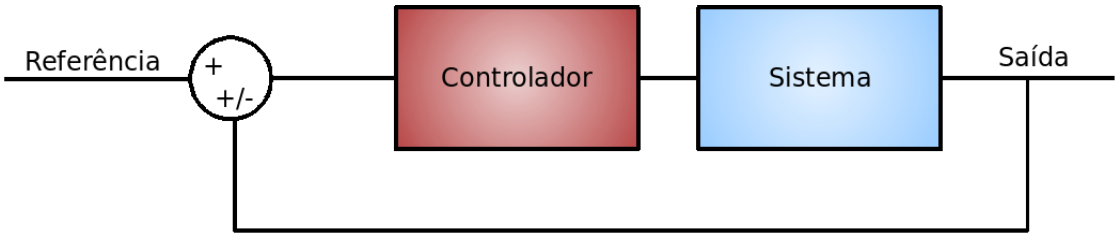


Figura 2.4: Sistema de controlo em malha fechada

Cada uma das formas de controlo de um sistema tem as suas características próprias provocando comportamentos distintos que devem ser ponderados na altura de escolha da forma de controlo. O projectista do sistema deve ter em atenção as especificações de comportamento que deseja que o sistema tenha e escolher a melhor forma de o controlar.

2.1.4 Métodos de Controlo de Sistemas em Malha Fechada

Neste trabalho, como já foi referido, a forma de controlo usada foi o controlo em malha fechada com realimentação negativa. Nesta secção analisam-se os métodos mais comuns de controlo de sistemas na forma referida.

- *Implantação de Pólos*[DHS03]

  Em sistemas em que a resposta é muito sensível a variações no sinal de entrada este método de controlo tenta eliminar a influência que as raízes do numerador, os "zeros" da equação característica, têm no sistema. Normalmente tenta-se eliminar estas componentes pois representam a característica derivativa do sistema, o que na prática se traduz por uma tendência para amplificar ruído tornando o sistema instável. Noutros casos apenas se pretende que o sistema tenha uma resposta mais rápida. Nestes casos tenta-se atenuar a influência que as raízes do denominador, os "pólos", da equação característica, têm no sistema. Geralmente os pólos estão associados a um comportamento integrador o que na prática se traduz por uma resposta lenta do sistema.

Este tipo de análise da equação característica do sistema leva à construção de um controlador com determinados "pólos" e "zeros" que levam o sistema a comportar-se da maneira desejada, tendo em conta as especificações de resposta exigidas.

Este método tem como principal desvantagem a complexidade de implementação, visto ser necessário a construção de um compensador que obedeça exactamente à equação característica desejada, o que por vezes é difícil de realizar. Estes factores levam a que o custo de desenvolvimento seja elevado.

Um diagrama de blocos deste método de controlo de sistemas é mostrado na figura 2.5.

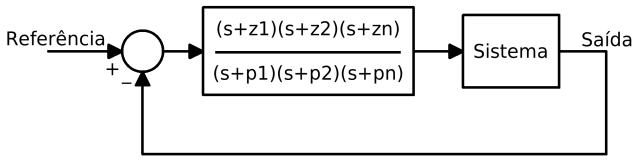


Figura 2.5: Sistema de controlo com implantação de pólos

- *Espaço de Estados* [Bro91]

  Este método exige que o sistema seja analisado de uma forma diferente dos métodos que eram usados na teoria clássica. Nesta nova estratégia são atribuídas ao sistema variáveis de estado, que contêm num determinado instante a informação do estado do sistema. O total das variáveis corresponde à informação total do estado do sistema. Estas variáveis podem ser alteradas consoante a forma que se pretende ver ou actuar no sistema. O método usado para o controlo dos sistemas é a simples realimentação destas variáveis (multiplicadas por um ganho). Esta simples multiplicação permite que haja uma alteração das suas características, ou seja, deslocação dos pólos, de forma a obter o comportamento desejado.

  Do ponto de vista de implementação do controlador, este método é bastante simples, visto serem apenas precisos tantos amplificadores quanto o número de variáveis de estado do sistema. No entanto a teoria que serve de suporte a este método é bastante complexa. Para sistemas mais complexos, com elevado número de variáveis, as matrizes resultantes necessitam de um processamento elaborado para a determinação dessas constantes. Outro factor que leva à não utilização deste método de controlo é que só determinados sistemas, os que possibilitam a determinação dos valores próprios das matrizes "A" e "B", é que são passíveis de serem controlados a partir deste método.

  Um diagrama de blocos deste método de controlo de sistemas é mostrado na figura 2.6

- *Compensador Proporcional, Integral e Derivativo* [DHS03]

  Neste método de controlo a saída resulta da soma de três componentes; a proporcional, que como o nome indica, é um valor proporcional ao erro do sistema nesse instante, a componente integrativa que é um valor proporcional ao integral dos erros durante todo o tempo de funcionamento do sistema até ao instante actual, e uma componente derivativa que quantifica a variação do sistema no instante actual.

  Este compensador pode ser descrito pela seguinte equação no domínio do tempo

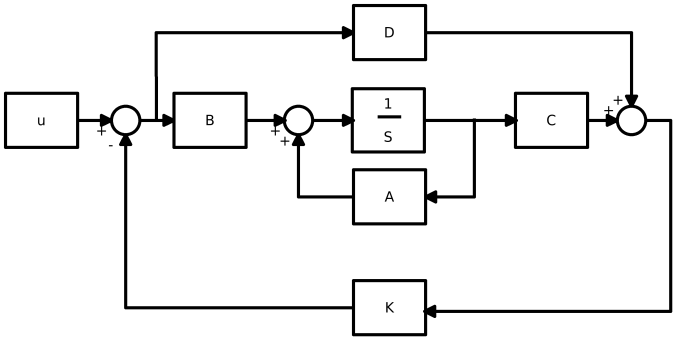


Figura 2.6: Sistema de controlo em Espaço de Estados

$$u(t) = K_p e(t) + K_i \int_0^t e(t) + K_d \frac{de}{dt} \tag{2.1}$$

Sendo u o valor de actuação no sistema, K_p, K_i e K_d as constantes proporcional, integrativa e derivativa respectivamente, e e o valor do erro do sistema em função do tempo.

Este método de controlo é o mais utilizado pois permite controlar grande parte dos sistemas actuais, ajustando apenas as três constantes referidas. Existem à venda muitas implementações deste tipo de compensador sendo relativamente barata a sua aquisição. A passagem deste algoritmo para o domínio de tempo discreto é bastante trivial sendo por isso muito simples de implementar em microcontroladores integrando-o assim em algoritmos de controlo mais complexos.

Um diagrama de blocos deste método de controlo é apresentado na figura 2.7

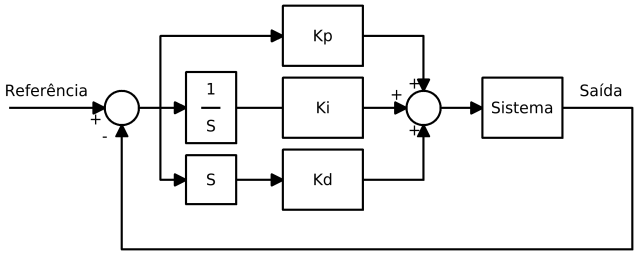


Figura 2.7: Sistema de controlo PID

Ao contrário da escolha da forma de controlo, todos os métodos, com as suas devidas restrições, permitem que o sistema atinja o comportamento desejado. A escolha do método de controlo prende-se com a experiência do projectista e com o domínio das ferramentas teóricas que servem de suporte a cada método.

2.2 Sistemas de Controlo de Tempo-Real

2.2.1 Definição

As evoluções que têm ocorrido na área da eléctronica digital possibilitaram a utilização com maior frequência de controladores digitais, ou processadores, no controlo de sistemas físicos. Com a utilização deste tipo de controlo em sistemas cada vez mais complexos que requerem um nível de resposta crítica, a problemática da segurança dos sistemas e da gestão do processador levou ao aparecimento dos sistemas de tempo real.

Para efeitos de análise, classificam-se os sistemas da seguinte forma [But05, Kop, SR90, OA04]:

- *Sistemas de monitorização;*
- *Sistemas de controlo em malha aberta;*
- *Sistemas de controlo em malha fechada.*

Os sistemas de monitorização são sistemas que não interferem no processo físico, tal como o nome indica são sistemas que apenas se destinam à monitorização de outros sistemas.

Os sistemas de controlo em malha aberta são sistemas que apenas actuam no processo físico. Este tipo de sistemas é sempre dependente de um factor externo que lhe forneça o *setpoint* de funcionamento.

Os sistemas de controlo em malha fechada são sistemas que para além de monitorizar, actuam autonomamente no processo. Para isso ser possível é necessário implementar um controlador. Neste tipo de sistemas a estreita ligação existente entre o meio físico e o controlador, leva a que a resposta deste, para além de ter que ser correcta, tenha que ser produzida num espaço de tempo limitado. *Uma resposta do sistema fora do tempo pode ser inútil ou mesmo perigosa* [LM04].

Um Sistema de Tempo-Real é um sistema reactivo, ou seja, um sistema que recebe constantemente informação do ambiente e actua sobre este a uma cadência adequada à obtenção de um comportamento desejado.

Geralmente as activações das tarefas de controlo podem ser resultado de acontecimentos assíncronos, por exemplo o carregar de um botão, ou de acontecimentos síncronos, por exemplo a amostragem do processo físico com um período constante. Se os componentes do sistema estiverem ligados a uma rede de comunicação, pode associar-se a cada um destes eventos o envio, ou recepção, de uma mensagem. Podendo estas, tal como os eventos, ser assíncronas ou síncronas. A rede de comunicação de suporte é de crucial importância para este tipo de *Sistemas Distribuídos de Tempo Real*.

Estes sistemas podem ser vulgarmente encontrados em objectos tão banais como máquinas de lavar, torradeiras, portas automáticas, bem como em sistemas aviónicos, sistemas de controlo de centrais nucleares ou sistemas de segurança activa nos automóveis, sistemas estes cuja falha poderá representar a perda de vidas humanas e elevados danos materiais.

Importa referir que sistemas de Tempo-Real são *sistemas computacionais*, e as restrições que se impõem são apenas a nível do processamento das tarefas associadas a cada evento.

Do ponto de vista de controlo o que se faz geralmente é encontrar modelos matemáticos digitais que executem a mesma função dos controladores analógicos, por exemplo, a passagem do filtro PID para a equação na forma polinomial

$$u(n) = K_p e(n) + K_i \sum_{i=0}^{n} e(i) + K_d \left(e(n) - e(n\text{-}1)\right) \tag{2.2}$$

Um sistema de Tempo-Real é então [But05][Kop]:

- *Um sistema computacional capaz de responder a eventos dentro de restrições temporais precisas;*
- *Um sistema cujo funcionamento correcto não depende apenas do valor das saídas mas também do instante em que são produzidas;*
- *Um sistema cuja evolução temporal deve estar sincronizada com a do ambiente em que opera.*

2.2.2 Classificação quanto à criticalidade

Nos sistemas actuais é fácil de compreender que coexistam simultaneamente um número elevado de tarefas, e que nem todas tenham a mesma importância para o desempenho requerido do sistema. Por exemplo num automóvel, não pode ser admissível que um evento gerado pelo sistema ABS (Anti Blocking System) tenha a mesma importância que um evento gerado pelo rádio, ou seja, quando existe um evento gerado pelo ABS, este tem forçosamente que adquirir uma importância, no sistema de controlo, mais elevada que o evento do rádio. Esse evento tem que ser atendido imediatamente a fim de não colocar em perigo a vida dos ocupantes. A falha no som do rádio não tem a mesma consequência que a falha no sistema de travagem.

Esta simples constatação dos requisitos do sistema levou à necessidade de classificar as tarefas quanto à sua criticalidade. Esta classificação é atribuída com base nas consequências da falha no atendimento do respectivo evento.

Assim sendo as tarefas podem ser[dO07]:

- *Ordinárias (Non Real-Time)* - Não possuem quaisquer restrições temporais;
- *Tempo-Real Não Críticas (Soft Real-Time)* - se o não cumprimento das restrições temporais causar apenas uma degradação do desempenho do sistema;
- *Tempo-Real Críticas (Hard Real-Time)* - se o não cumprimento das restrições temporais causar efeitos catastróficos no sistema controlado ou no ambiente.

Esta classificação preliminar ajuda na atribuição das características que cada tarefa terá. Por exemplo, no caso do automóvel, a tarefa associada ao ABS iria ter uma prioridade de execução maior que a do rádio. Também neste caso podemos classificar a tarefa de atendimento ao ABS como "Tempo-Real Crítica" e a tarefa associada ao rádio como "Tempo-Real Não Crítica".

Um sistema computacional capaz de gerir tarefas de Tempo-real designa-se "Sistema Operativo de Tempo Real". Se as suas funcionalidades possibilitarem também o gestão de tarefas de "Tempo-Real Críticas" designa-se por "Sistema Operativo Hard Real-Time".

2.2.3 Características

Com o crescimento das capacidades computacionais dos processadores começou a ser possível a um controlador estar encarregue de diversos sistemas físicos. Este aumento do número de processos concorrentes levou a que, na concepção do controlador, fossem usados métodos que permitam separar claramente quais as acções que têm que ser tomadas perante um determinado evento. Ligando cada evento a um determinado conjunto de acções torna-se mais simples a concepção e manutenção deste tipo de sistemas.

As linguagens de programação mais comuns permitem este tipo de separação possibilitando ao projectista atribuir um determinado evento a uma ou mais tarefas.

Uma tarefa é por isso [dO07]:

- *Uma sequência de instruções que, na ausência de outras actividades, é executada ininterruptamente pelo processador até ser completada.*

O que distingue uma tarefa de uma simples função de programação são as propriedades temporais que se lhe atribuem.

Para que o sistema tenha um determinado desempenho é necessário que as tarefas cumpram determinados requisitos. Pode-se compreender uma tarefa como sendo uma função à qual se impõem restrições à sua execução mediante um critério temporal definido.

As tarefas podem ser:

- *Periódicas* - Se atribuídas a eventos periódicos;

  $$\textit{Instância } n \textit{ activada em } a(n) = n.T + \phi$$

  Onde n corresponde à activação, T ao período das activações e ϕ ao instante da primeira activação

- *Esporádicas* - Se atribuídas a eventos para os quais se consegue determinar o tempo mínimo em que estes podem ocorrer;

  $$\textit{mit - tempo mínimo entre activações}$$

- *Aperiódicas* - Se atribuídas a eventos passíveis apenas de caracterizar de uma forma probabilística.

Geralmente as tarefas Periódicas estão associados a processos determinísticos que requerem uma grande coerência temporal. Processos como amostragem de sensores, geração de PWM (Pulse Width Modulation), entre outros.

As tarefas Esporádicas são geralmente atribuídas a eventos passíveis de sofrerem *jitter* de activação, ou seja, o instante em que o evento é despoletado depende de factores que podem atrasar essa activação, ou a tarefas que não são de todo periódicas. Nesse caso é calculado o tempo mínimo entre activações e assume-se, para efeitos de cálculo, que estas tarefas são periódicas com período igual ao tempo mínimo entre activações, considerando-se assim a situação de pior caso. Um exemplo deste tipo de tarefas é o contar de peças numa linha de montagem.

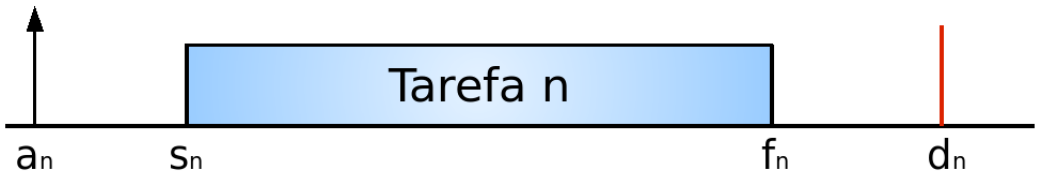


Figura 2.8: Esquema simplificado de uma tarefa

As tarefas Aperiódicas são tarefas em que não é de todo possível calcular um tempo mínimo durante o qual se garanta que o evento não irá ocorrer. O exemplo clássico deste tipo de eventos são os associados a botões de accionamento. Geralmente estes botões só irão ser accionados em casos excepcionais cuja ocorrência não é passível de cálculo determinístico.

Consoante o modo de activação das tarefas pode-se determinar as diferentes propriedades que lhes são inerentes.

Uma tarefa τ_i tem as seguintes propriedades

- C - Tempo máximo de execução da tarefa (WCET - Worst Case Execution Time);
- T - Período da activação (Aplica-se em tarefas periódicas);
- ϕ - Fase relativa, instante da primeira activação;
- *mit (minimum interarrival time)* - Tempo mínimo entre activações, (Aplica-se em tarefas esporádicas);
- a_n - Instante da activação da n^a instância;
- s_n - Instante do início de execução da n^a instância;
- f_n - Instante do fim de execução da n^a instância;
- $c_n(t)$ - Tempo máximo de execução residual da n^a instância no instante t;
- D - Deadline relativa;
- d_n - Instante de terminação da deadline da n^a instância.

2.2.4 Restrições e Requisitos

As restrições impostas às tarefas, estão relacionadas com diversos aspectos sempre dependentes do desempenho desejado do sistema. Estas restrições podem ser do tipo

- *Temporais* - estão relacionadas com os constrangimentos temporais das tarefas, por exemplo, o período com que uma mensagem é enviada, o seu instante inicial de activação, a sua deadline, entre outros;
- *Precedência* - num sistema deste tipo é normal que as tarefas executem segundo uma ordem previamente estabelecida. Por exemplo, não é recomendável que a tarefa de actuação seja executada antes da tarefa de leitura dos sensores. Também não é aconselhável actuar num sistema com informação demasiado antiga pois isso introduziria um atraso que poderia colocar em causa a estabilidade do sistema;

- *Recursos* - O acesso a recursos terá que ser feito com garantias de exclusão mutua.

Tal como é necessário impor restrições às tarefas, é também necessário garantir que estas cumpram determinados requisitos de modo a que, no conjunto, o sistema atinja o comportamento desejado. Neste campo não é só o desempenho do mesmo que tem que ser assegurado; a sua tolerância a falhas é também de suma importância.

Os requisitos podem ser dos seguintes tipos

- *Funcionais* - Normalmente advêm da dinâmica do processo físico que se pretende controlar. Impõem ao sistema as restrições acima referidas que têm que ser cumpridas em todas as activações e não apenas em termos médios;

- *Temporais* - Inerentes ao modelo do controlador do sistema. Estes têm que garantir o mínimo de variações na activação das tarefas. Parâmetros como o *jitter* de terminação, relacionado com a diferença de tempos de execução das tarefas, atrasos de amostragem e actuação, entre outros, permitem identificar quais as restrições temporais das tarefas;

- *Confiabilidade* - Inerentes ao modo como o sistema lida com falhas. Um sistema computacional possui "Confiabilidade" se possuir mecanismos que possibilitem depender a realização de um determinado serviço ou função do mesmo sistema computacional. Aspectos como a prontidão, continuidade, inocuidade e confidencialidade ajudam a caracterizar o grau de "Confiabilidade" que um determinado sistema computacional possui.

Uma análise detalhada dos requisitos e das restrições a que as tarefas estão sujeitas num determinado sistema é de suma importância aquando do seu projecto. O resultado desta análise vai ditar as características que o sistema computacional irá necessitar. Uma análise errada poderá levar à falha do sistema ou ao sobredimensionamento dos seus componentes.

2.2.5 Escalonamento

Da divisão do controlador em tarefas, surge a necessidade de determinar, em cada instante, que tarefa irá ser executada.

Ao procedimento de escolher qual das tarefas, ou mensagens, deverá ser executada num determinado instante dá-se o nome de *Escalonamento*. Ao conjunto de regras que determinam, em qualquer instante, a ordem pela qual as tarefas, ou mensagens, são despachadas dá-se o nome de "Algoritmo de Escalonamento" [Ped].

2.2.5.1 Taxonomia

O Algoritmo de Escalonamento pode ter as seguintes características:

- *Preemptivo vs Não Preemptivo*

  Se for possível, num determinado instante, ser retirada uma tarefa ao processador sendo este atribuído a outra tarefa que tenha, nesse mesmo instante, maior prioridade, então diz-se que o algoritmo de escalonamento é *preemptivo*. Caso contrário, ou seja, se não for possível retirar o processador a uma determinada tarefa sendo esta executada até ao fim, então diz-se que o algoritmo de escalonamento é *não preemptivo*.

- *Estático vs Dinâmico*

  Se as decisões de escalonamento forem tomadas com base em parametros fixos, atribuídos estaticamente às tarefas antes da sua activação, diz-se que o algoritmo de escalonamento é *estático*. Se em vez disso for baseado em parâmetros cujo valor pode variar com o tempo, o algoritmo de escalonamento diz-se *dinâmico*.

- *Off-Line vs On-Line*

  Se todas as decisões forem tomadas antes da activação do sistema, isto é, se a ordem pela qual as tarefas forem executadas estiver armazenada numa tabela, o algoritmo de escalonamento diz-se *off-line*. Se o algoritmo de escalonamento for executado durante a operação do sistema computacional diz-se *on-line*.

- *Óptimo vs Sub-óptimo*

  O algoritmo de escalonamento diz-se *óptimo* se no caso de existir um escalonamento possível, ele for capaz de o determinar, caso contrário designa-se por *sub-óptimo*.

Na figura 2.9 é mostrado um resumo dos termos usados e a forma como estes se interligam.

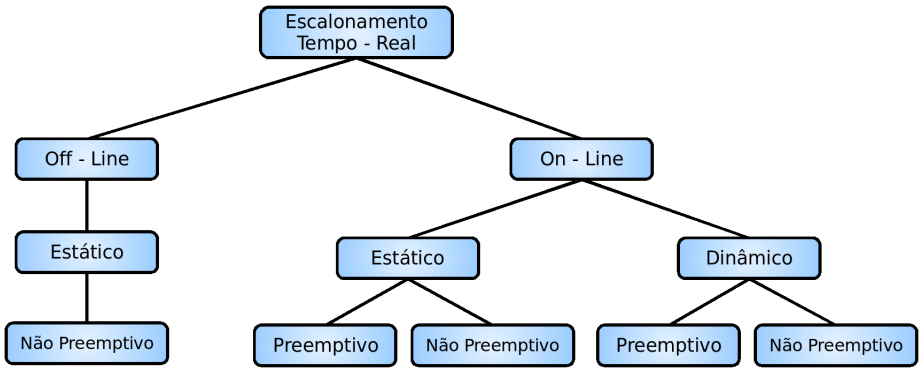


Figura 2.9: Esquema Resumo da Taxonomia

2.2.5.2 Políticas de Escalonamento

Existem diversas Políticas de Escalonamento, entre elas destacam-se:

- *Estático Cíclico;*
- *Rate Monotónico;*
- *Deadline Monotónico;*
- *Earliest Deadline First;*
- *Least Slack First;*
- *First Come First Served.*

Cada Política de Escalonamento tem as suas vantagens e inconvenientes face às outras. Está fora do âmbito desta dissertação explanar com mais detalhe os aspectos inerentes a cada uma. Deste modo analisam-se apenas o *Rate Monotónico* e o *Earliest Deadline First* pois são estes os mais eficientes e vulgarmente utilizados.

- *Rate Monotónico*

Este algoritmo de escalonamento faz a atribuição das prioridades às tarefas de forma inversamente proporcional ao período das mesmas. Ou seja, a tarefa com maior prioridade será a tarefa que tiver o menor período.

Este algoritmo tem as seguintes propriedades[dO07]:

- *Período igual à Deadline, $T_i = D_i$;*

- *Minimiza o atraso máximo da tarefa com T_i menor.*

O algoritmo de escalonamento Rate Monotónico é óptimo dentro dos algoritmos de prioridades fixas, ou seja, se um determinado conjunto de tarefas Γ não for escalonável para este algoritmo também o não vai ser para nenhum outro algoritmo de prioridades fixas.

- *Earliest Deadline First*

Neste algoritmo uma tarefa, adquire maior prioridade do que outra, num determinado instante, se a sua deadline absoluta for a mais próxima. Ou seja, em cada instante é determinado quanto tempo as tarefas têm para serem executadas até atingirem a sua deadline, a tarefa com um tempo mais curto adquire, nesse instante, a maior prioridade do conjunto.

As propriedades deste algoritmo são[dO07]:

- *Prioridade dinâmica, ou seja, depende do instante de activação;*

- *Minimiza o atraso máximo de todas as tarefas, ou mensagens;*

- *Minimiza o número de mudanças de contexto relativamente às políticas RM e DM.*

Este algoritmo é considerado óptimo face a todos os outros algoritmos de escalonamento visto permitir que o processador atinja cem por cento de utilização.

2.2.5.3 Análise de Escalonabilidade

A análise preliminar que deverá ser feita consiste em verificar se o conjunto das tarefas não apresenta uma taxa de utilização do processador superior a cem por cento. Neste caso qualquer escalonamento não será capaz de garantir o cumprimento das restrições impostas.

Esta análise é feita da seguinte forma.

Dado um conjunto de tarefas $\Gamma = \{\tau_i\,(C_i, \phi_i, T_i, D_i, i = 1..n)\}$

A taxa de utilização será

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i} \tag{2.3}$$

Esta medida serve apenas para determinar qual a utilização que o processador está submetido com o conjunto dado. No caso de Políticas de Escalonamento com prioridades dinâmicas este critério define se o conjunto é escalonável, noutros casos apenas este critério não é indicador da escalonabilidade do conjunto de tarefas.

Para Políticas de Escalonamento com prioridades fixas existem alguns métodos que permitem determinar se um determinado conjunto de tarefas é escalonável. Estes métodos podem-se basear no seguinte:

- *Taxa de Utilização do Processador*

  Os seguintes métodos aplicam-se quando o escalonamento permite a preempção das tarefas e quando a deadline é igual ao período, $D_i = T_i$;

  Menor Majorante de Liu&Layland [LL73]

$$\sum_{i=1}^{n} \left(\frac{C_i}{T_i} \right) \leq n \left(2^{\frac{1}{n}} - 1 \right) \tag{2.4}$$

  Se esta condição for verdadeira consegue-se garantir o escalonamento das tarefas

  Majorante Hiperbólico de Bini&Buttazzo&Buttazzo [BBB03]

$$\Pi_{i=1}^{n} \left(\frac{C_i}{T_i} + 1 \right) \leq 2 \tag{2.5}$$

  Se esta condição for verdadeira consegue-se garantir o escalonamento das tarefas

- *Máximo Tempo de Resposta* [ABR$^+$93]

  De acordo com este método o maior tempo de resposta de uma tarefa, ou mensagem, periódica (R_i) é dado pela soma do seu tempo de computação(C_i) com o tempo de interferência que esta pode sofrer por mensagens de mais alta prioridade(I_i), calculado no tempo de activação de pior caso, geralmente considerado como sendo uma activação simultânea de todo o conjunto em estudo.

$$R_i = C_i + I_i \tag{2.6}$$

  A interferência sofrida por tarefas de maior prioridade é dado por

$$I_i = \sum_{\forall j \in \subset hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{2.7}$$

  Onde $hp(i)$ é o conjunto de tarefas com maior prioridade.

  Combinando as duas equações anteriores obtém-se

$$R_i = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{2.8}$$

Como o factor R_i aparece em ambos os membros da equação esta só se consegue resolver de forma recursiva. Seja r_i^n a n^a aproximação do valor real r_i. As sucessivas aproximações são

$$r_i^{n+1} = C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{r_i^n}{T_j} \right\rceil C_j \tag{2.9}$$

A iteração começa com $r_i^0 = 0^+$ e acaba quanto $r_i^{n-1} = r_i^n$, ou seja, termina quando a solução converge para um valor constante. Se durante o processo iterativo $r_i^n > D_i$ então essa tarefa não cumpre os requisitos temporais impostos tornando o conjunto não escalonável.

2.2.6 Sistemas de Tempo-Real em Redes de Comunicação

Analisando o problema da gestão das tarefas num processador e comparando-o com o problema da gestão de mensagens num barramento conclui-se que são, em grande parte, muito semelhantes. Todas as propriedades, requisitos e restrições que se impõem às tarefas têm correspondência directa nas mensagens. Por exemplo uma mensagem poderá ser despoletada por eventos periódicos (o envio de informação de um sensor para um controlador), esporádicos (mensagens de *keep-alive*) ou aperiódicos (mensagem despoletada pelo accionamento de um botão), tal como uma tarefa. Na generalidade dos casos uma mensagem é enviada como resultado final de uma tarefa, e a recepção de uma mensagem despoleta também a execução de uma tarefa.

Esta correspondência entre tarefas e mensagens faz com que grande parte da teoria aplicada às tarefas seja também aplicada a mensagens. Assim, tudo o que já foi referido sobre escalonamento e análise de escalonabilidade, é inteiramente aplicável a mensagens.

Apenas há a referir que a preempção no caso das mensagens é mais difícil de executar do que nas tarefas. Uma solução adoptada por alguns protocolos para minimizar este problema foi estabelecer um tamanho máximo para as mensagens minimizando assim o impacto que a falta de preempção tem sobre a eficiência do escalonador.

O protocolo FTT[Ped] é um exemplo da aplicação da teoria de sistemas de Tempo-Real a barramentos de comunicação.

2.3 Sistemas de Controlo Distribuído

2.3.1 Definição

Um sistema de controlo distribuído pode ser definido, do ponto de vista computacional, da seguinte forma [BPG05]:

- *Um sistema em que vários processadores e/ou bases de dados interagem cooperativamente para atingir um objectivo comum. Os processos coordenam as suas actividades e trocam informações entre si através de dados enviados por uma rede de comunicação.*

Isto é, um sistema de controlo distribuído compreende um conjunto de unidades de processamento, que conjuntamente cooperam para atingirem um determinado objectivo. Para atingir esse objectivo as várias unidades necessitam de trocar informação entre si. Essa informação é trocada através de uma rede de comunicação.

Um sistema distribuído de tempo-real é um sistema distribuído que desempenha tarefas de tempo real.

Algumas tarefas, nas unidades de processamento, poderão necessitar de informação presente noutras unidades. Para que sejam atingidos os requisitos de tempo-real a troca de informação e a execução de tarefas têm que ser realizadas dentro de limites temporais restritos. Para que todo o sistema tenha propriedades de tempo-real todos as unidades de processamento bem como a rede de comunicação têm que possuir essas propriedades.

Alguns exemplos deste tipo de sistemas são aplicações nas áreas da robótica, aplicações multimédia, automação industrial e sistemas de controlo adaptativos.

2.3.2 Propriedades

Um sistema de controlo distribuído pode ser avaliado tendo em conta as propriedades que possui. Essas propriedades podem ser:

- *Agregabilidade (Composability)*

  Um sistema distribuído possui agregabilidade, relativamente a uma propriedade, se a integração de um subsistema não invalidar essa propriedade.

- *Resposta Temporal (Timeliness)*

  Um sistema distribuído possui propriedades de resposta temporal se alguma das suas funções tiver restrições temporais. Geralmente associa-se esta propriedade a sistemas distribuídos de tempo-real.

- *Capacidade de Teste (Testability)*

  Um sistema distribuído possui capacidade de teste de uma funcionalidade se este dispuser de mecanismos que verifiquem o correcto funcionamento do mesmo sem ser necessária a sua integração com os restantes elementos do sistema.

- *Escalabilidade (Scalability)*

  Um sistema distribuído possui escalabilidade se se conseguir fazer aumentar o seu tamanho.

- *Dependabilidade (Dependability)*

  Um sistema distribuído possui a propriedade de dependabilidade quando consegue manter-se em funcionamento com as características tempo-real adequadas na presença de falhas ou erros.

2.3.3 Controlo Centralizado vs Controlo Distribuído

Pode-se optar pelas seguintes topologias de controlo:

- *Controlo centralizado*

  Todas as tarefas de controlo estão concentradas num único componente, o controlador.

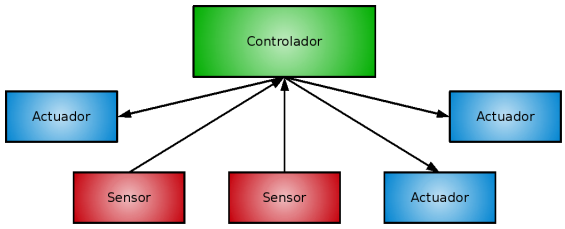


Figura 2.10: Controlo Centralizado

- *Controlo Centralizado em Rede*

  As tarefas continuam a estar centralizadas num único componente, o controlador, mas os componentes comunicam entre si através de uma rede dedicada.

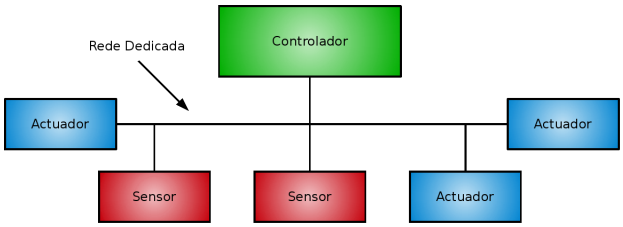


Figura 2.11: Controlo Centralizado em Rede

- *Controlo Distribuído*

  Em que diversos controladores cooperam e trocam informações entre si. As tarefas de controlo estão divididas pelos diversos nós de controlo na rede.

As grandes desvantagens dos sistemas de controlo centralizado são o crescimento dos recursos necessários em função do crescimento da complexidade dos sistemas, a dificuldade de manutenção dos equipamentos e a grande quantidade de ligações necessárias. Em industrias exigentes do ponto de vista de limitação de recursos, como é o caso das industrias automóvel e aviónica, estes problemas adquirem elevada importância tendo em conta que a complexidade dos sistemas é muito elevada e com tendência para aumentar. Estima-se que o comprimento

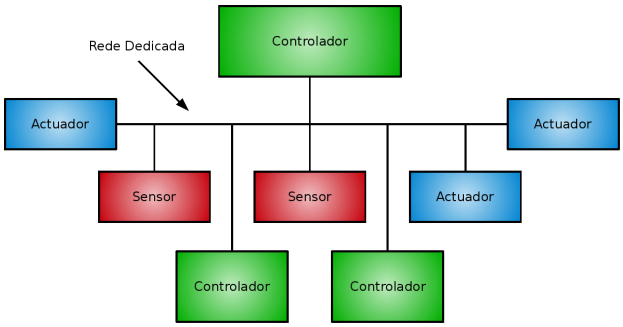


Figura 2.12: Controlo Distribuído

total da cablagem presente num automóvel ronde os dois quilómetros, tendo sido feito um grande esforço por parte deste tipo de industrias para reduzir o peso que os sistemas de controlo têm na estrutura. A detecção de falhas neste tipo de sistemas torna-se mais complicada à medida que o sistema cresce.

Nestes casos opta-se por uma solução distribuída, em que um ou mais componentes de controlo são responsáveis por diversos sistemas simultâneamente cooperando entre si. Esta solução leva a uma redução drástica na quantidade de ligações. Como geralmente é usado uma rede estandardizada, fica facilitada a substituição de componentes, bem como é mais fácil encontrar no mercado sensores adaptados à rede utilizada, produzidos até por diferentes fabricantes. Fica também facilitada a detecção de erros de funcionamento. Com a integração das redes nestes sistemas basta num ponto da rede, onde se tem acesso a todos os componentes de controlo, requerer informação de falhas do sistema e, com base nesta informação, agir em conformidade. Com este tipo de solução consegue-se fazer um diagnóstico a todos os sistemas num tempo muito reduzido. A facilidade de realizar a redundância num sistema distribuído (basta adicionar um novo nó à rede) faz com que esta solução possibilite uma maior tolerância a falhas.

2.3.4 Comunicação Event-Trigger Vs Time-Trigger

Com a aplicação de redes de comunicação a sistemas com restrições temporais surgiu o problema de como seria feito o acesso ao meio de comunicação visto este ser partilhado por todas as estações na rede.

Neste campo surgiram duas abordagens que divergem desde logo na forma [Kop93], a Time-Trigger[TTP99] e a Event-Trigger.

A abordagem Time-Trigger defende que todo o nó produtor de informação o deve fazer num determinado tempo previamente estipulado, sendo esse tempo extremamente inflexível. A consequência disso é uma maior eficiência na utilização do meio visto que o caso de utilização máxima (pior caso) poder ser calculado e todo o sistema ser configurado do modo a evitar falhas. A sua principal desvantagem é o facto de todos os nós terem que estar sincronizados. É portanto um tipo de protocolo inflexível e determinístico.

A Event-Trigger não impõe qualquer tipo de restrição temporal ao lançamento de mensagens para o meio sendo a resolução de colisões da responsabilidade do protocolo de transmissão. A consequência disso é uma maior simplicidade de implementação. Devido à sua imprevisibilidade não se garante a maior rentabilidade do meio. É por isso um tipo de proto-

colo flexível.

Na prática verifica-se que em muitos casos a melhor solução é uma misturas dos dois [Kop93], num sistema é comum coexistirem os dois tipos de tráfegos, se por um lado existe tráfego que necessita de um rigoroso controlo de tempo (tipicamente tráfego de controlo, *streams* de áudio ou vídeo), por outro existe tráfego que necessita do barramento esporadicamente (por exemplo botões de accionamento ou transferências de ficheiros).

Um exemplo de protocolo que implementa este tipo de solução é o protocolo FTT[Ped].

2.3.5 Modelos de Cooperação

A troca de informação numa rede de controlo distribuído não se limita apenas à forma como se trocam fisicamente as mensagens. A forma como as mensagens são distribuídas pelos diferentes nós na rede é um factor relevante a ter em conta. Dependendo da aplicação, um nó pode precisar de informação que resida apenas num dos restantes nós ou em vários nós da rede. A comunicação numa rede pode ser feita das seguintes formas:

- *ponto - a - ponto;*
- *ponto - a - multiponto;*
- *multiponto - a - ponto;*
- *multiponto - a - multiponto.*

Para resolver alguns dos problemas inerentes aos diferentes tipos de comunicação propuseram-se alguns modelos. De seguida explicam-se os modelos mais relevantes[VR01].

- *Produtor - Consumidor*

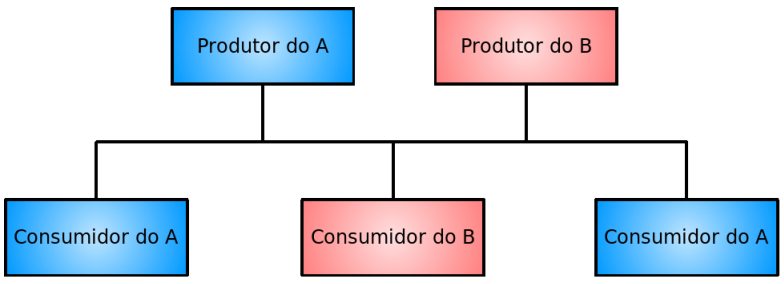


Figura 2.13: Modelo Produtor - Consumidor

Este modelo atribui a cada tipo de mensagem um identificador. A produção de mensagens e a recepção destas é orientada aos identificadores, ou seja, um determinado nó vai ser produtor de um, ou mais, tipos de mensagens, tal como um nó consumidor recebe apenas as mensagens das quais é consumidor.

Este tipo de comunicação não faz qualquer referência ao endereço de destino ou de fonte. Um produtor não necessita de saber quais os consumidores da mensagem que produz, nem

os consumidores necessitam saber quais os produtores que produzem determinado tipo de mensagem.

Este modelo de cooperação suporta os tipos de comunicação ponto - a - ponto e ponto - a - multiponto.

O tipo de comunicação ponto - a - multiponto poderá ter problemas de inconsistência espacial se a rede não possuir um mecanismo de *broadcast* atómico. Uma variável pode ser produzida enquanto outra ainda não foi entregue a todos os nós, neste caso, haveria na rede nós com imagens diferentes do mesmo tipo de informação.

Este modelo não consegue resolver o problema da inconsistência temporal visto poder haver atrasos no processo de produção fazendo com que informação seja entregue fora do tempo estipulado.

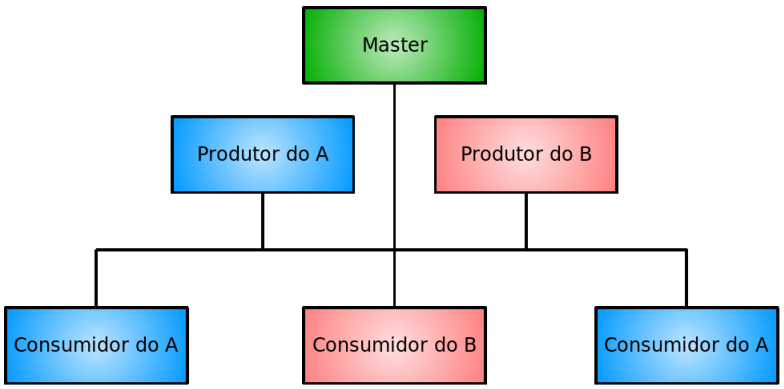


Figura 2.14: Modelo Produtor - Distribuídor - Consumidor

O problema da inconsistência temporal foi resolvido fazendo uma alteração ao modelo produtor - consumidor, criando assim o novo modelo **Produtor - Distribuídor - Consumidor**.

Neste modelo, os produtores actuam como *Slaves* que são geridos por um nó principal chamado de *Master*. Os produtores, ou *Slaves*, só produzem informação quando são autorizados pelo *Master*.

O *Master* possui a informação completa das mensagens que irão ser enviadas e as restrições temporais a que estas estão sujeitas conseguindo assim escalonar as referidas mensagens por forma a respeitarem essas restrições.

O protocolo FTT-SE implementa este modelo de comunicação.

- *Cliente - Servidor*

Neste modelo os nós que queiram consumir (Clientes) têm que efectuar um pedido aos nós produtores (Servidores). Estes respondem a esse pedido com a informação actualizada.

Este modelo suporta comunicação ponto - a - ponto. Para comunicações do tipo ponto - a - multiponto e multiponto - a - ponto surgem os problemas de inconsistência espacial e temporal. Por exemplo, se num determinado momento existir, num servidor, um conjunto de pedidos da mesma informação que estão a ser processados e essa mesma informação vir alterado o seu valor a meio desse processamento, vai-se suceder a inconsistência espacial da

informação nos diferentes nós, pois uns vão ser actualizados com o valor antigo e os outros com o valor mais recente. Por outro lado se um nó necessita fazer um pedido a vários servidores em simultâneo terá que o fazer sequencialmente o que leva à inconsistência temporal da informação.

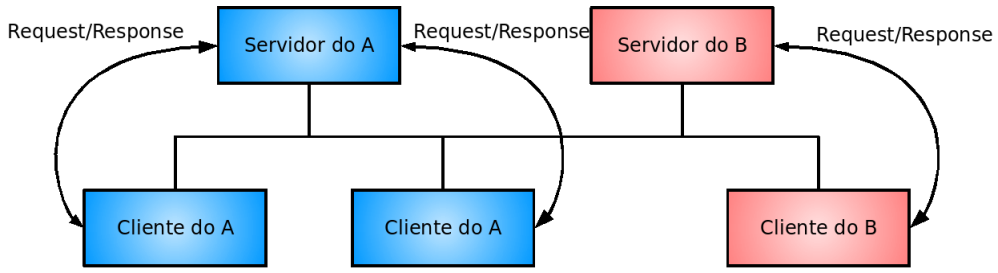


Figura 2.15: Modelo Cliente - Servidor

O facto dos pedidos serem feitos de forma assíncrona, bem como os diferentes tempos de processamento nos servidores faz com o seu comportamento seja caracterizado de forma probabilística fazendo com que este modelo não tenha aplicações em sistemas com restrições temporais.

Capítulo 3

Redes de Comunicação em Sistemas Distribuídos

Neste capítulo é feita uma breve análise às redes de campo mais conhecidas, existentes no mercado, descrevendo as suas principais características. Seguidamente é feita uma descrição do protocolo Ethernet e dos protocolos existentes que conferem propriedades de Tempo-Real à rede Ethernet. Por último é apresentado o paradigma FTT concretizado no protocolo FTT-SE.

3.1 Redes Dedicadas / FieldBuses, breve análise

Existem actualmente um grande número de protocolos com propriedades de tempo-real para aplicações em sistemas distribuídos. Estes protocolos conhecidos como "Redes de Campo", ou "FieldBuses", têm como principal objectivo ligar os vários componentes de um sistema de controlo distribuído, tal como PLC's, actuadores e sensores. Este tipo de protocolos têm como principais requisitos[Pim90, Dec01, Ped]:

- *Lidar com mensagens curtas de modo eficiente;*
- *Suportar tráfego periódico, com diferentes períodos, e tráfego aperiódico;*
- *Resposta dentro de limites temporais definidos;*
- *Tolerantes a falhas;*
- *De baixo custo, tanto no equipamento como na infraestrutura e manutenção.*

Nas secções que se seguem é feita uma pequena descrição das características das mais importantes redes de campo.

3.1.1 CAN - Controller Area Network

Este protocolo[Gmb91] foi criado por Robert Bosch GmbH em meados dos anos 80 para ser utilizado na industria automóvel. O objectivo da criação deste protocolo foi desenvolver um mecanismo eficiente de interligar os diferentes componentes de um automóvel reduzindo a quantidade de cabos necessária.

O protocolo CAN permite transmissões de 1Mbps [ISO93] e de 125Kbps[ISO94]. O meio de transmissão é geralmente um duplo cabo cruzado e o seu comprimento depende da velocidade de transmissão.

É um protocolo com múltiplos *Masters* usando o CSMA/NBA (*Carrier Sense Multiple Access whith Non-destrutive Bitwise Arbitration*) para resolução de colisões, ou seja, usa um mecanismo de prioridades estáticas para determinar, em caso de colisão, qual a estação que transmite, sem destruir a informação que está no barramento. Num determinado instante um nó ganha acesso ao barramento se estiver a transmitir a mensagem com um valor identificador menor, o que corresponde a possuir uma prioridade maior.

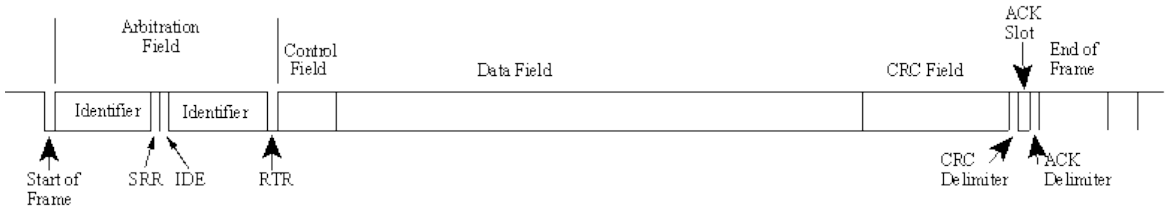


Figura 3.1: Mensagem CAN 2.0A

A figura 3.1, retirada de [KVA], mostra a composição de uma mensagem CAN 2.0A cujos campos possuem as seguintes funções:

- *Identifier* - contém 11 ou 29 bits (CAN 2.0A / CAN 2.0B). Este campo identifica o tipo de mensagem, é utilizado na resolução de colisões determinando qual das estações ganha acesso ao barramento;
- *Data Field* - Campo de dados, possibilita a transmissão de informação desde 0 a 8 bytes;
- *RTR (Remote Transmit Request)* - Este bit é usado por um nó para pedir que outro produza informação; Quando este pedido é feito, o nó requerente envia uma mensagem com o campo de dados vazio e com o identificador da mensagem que quer que seja produzida. O nó produtor responde a essa mensagem com o mesmo identificador e com os dados que foram requeridos;
- *CRC* - Bits de detecção e correcção de erros de transmissão;
- *ACK* - Bits de *Acknowledge*.

A rede dedicada CAN foi largamente utilizada em aplicações não só na industria automóvel como também, mais recentemente, em aplicações de domótica, aviónica, entre outras. No Laboratório de Sistemas Electrónicos da Universidade de Aveiro foi desenvolvido um protocolo[APF02] com base no protocolo CAN que lhe confere propriedades de tempo-real. Este protocolo foi aplicado com sucesso em diversos sistemas de controlo distribuído de entre as quais se destaca a integração no sistema de controlo da equipa de futebol robótico da Universidade de Aveiro CAMBADA[1].

[1] "CAMBADA" é um acrónimo de "Cooperative Autonomous Mobile roBots with Advanced Distributed Architecture".

3.1.2 WorldFIP

O protocolo WorldFIP[CEN96, Com00] é baseado no modelo produtor-distribuidor-consumidor, ou seja, é orientado à variável. Num dado momento é determinada que variável vai ser produzida e o nó responsável pela produção dessa variável terá que a produzir.

Na rede existe um nó cuja única funcionalidade é determinar que variável é produzida no próximo ciclo, a esse nó dá-se o nome de "Bus Arbitror" (BA).

O processo de produção de uma variável obedece aos seguintes passos:

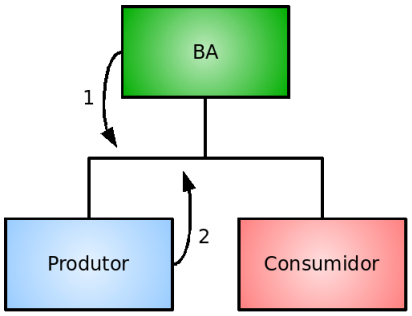


Figura 3.2: WoldFip - Produção de uma Variável

1. O BA envia uma mensagem (ID_DAT) com a identificação da variável que irá ser produzida;

2. O produtor dessa variável responde com uma mensagem (RP_DAT) com o identificador da variável e o valor actualizado da mesma.

Ao receber a mensagem, o consumidor armazena o seu valor nos *buffers* da camada de ligação de dados (DLL). A camada de aplicação (AL) disponibiliza um conjunto de funcionalidades que possibilitam a interacção com os *buffers* da camada DLL.

O escalonamento presente no BA é feito *off-line* com base numa tabela interna estática (BAT) programada antes do arranque do sistema.

O tempo é separado em Ciclos Elementares (EC), durante os quais irão ser produzidas determinadas variáveis. O Ciclo Principal (MC) determina o tempo ao fim do qual um EC se vai repetir. Dito de outro modo, o MC representa todas as entradas da BAT, e o período deste representa o tempo que a tabela demora a ser executada até voltar a ser repetida.

As mensagens aperiódicas serão enviadas no tempo que sobra do EC e seguem os seguintes passos:

1. Enquanto está a ser transmitida uma mensagem periódica, um nó que tenha mensagens aperiódicas para enviar, assinala esse facto activando o bit (RP_DAT) na mensagem;

2. O BA guarda os pedidos e no final da janela síncrona pede aos nós a lista das variáveis aperiódicas;

3. Finalmente, o BA processa a lista das variáveis aperiódicas e determina quais as mensagens que irão ser produzidas.

3.1.3 Profibus

O protocolo Profibus[CEN96, Com00] foi desenvolvido com o intuito de fornecer um modo de comunicação determinístico entre os diversos componentes de uma rede de controlo distribuído, por exemplo, PLC's, sensores e actuadores.

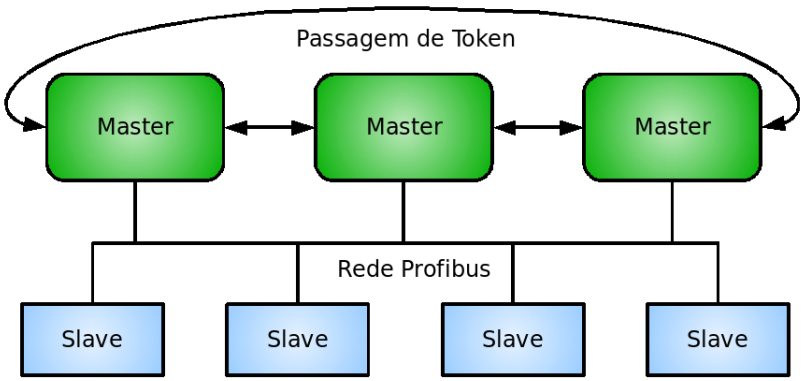


Figura 3.3: Profibus - Modo de Comunicação

O Profibus implementa um modo de comunicação base em passagem do "*Token*". Os *Masters* passam entre si o *Token* determinando assim, em cada instante, qual tem o acesso ao barramento. Quando um *Master* possui o *Token* este estabelece comunicação com os *Slaves*.

O protocolo implementa a segunda camada do modelo OSI, a camada MAC, que neste caso se chama "Field Data Link" (FDL). Esta camada possibilita o suporte de vários serviços, como a comunicação em *broadcast* e a comunicação *unicast*.

O *Master* ao fazer uma acção de Envio, Pedido ou Envio/Pedido (Send, Request, Send/Request) o *Slave* correspondente dispõe de um tempo limitado para responder. Se esse tempo for ultrapassado, novo pedido será feito. Ao fim de um determinado número de pedidos sem resposta será referenciado um erro de comunicação.

Os *Masters* implementam um mecanismo que evita a retenção por tempo ilimitado do *Token*. Esse mecanismo é baseado em *timers* e em mensagens com diferentes prioridades.

3.1.4 DeviceNet

O protocolo DeviceNet[Ass97] foi desenvolvido pela Rockwell Automation como uma rede de campo de livre utilização, baseada em CAN, para aplicações de automação.

Juntamente com a ControlNet e a EtherNet/IP implementa uma camada de aplicação (camada ISO 7) designada "Control and Information Protocol" (CIP). A parte de controlo desta camada lida com os dados de tempo-real, enquanto a parte de informação lida com os dados de configuração, diagnóstico e gestão.

Neste protocolo estão definidos dois tipos de mensagens:

- *Mensagens I/O* - Mensagens para tráfego de tempo-real, geralmente com prioridade alta;
- *Mensagens Explícitas* - Mensagens para tráfego de configuração, geralmente com prioridade baixa e usa exclusivamente comunicação ponto-a-ponto;

A comunicação baseia-se em conexões que são estabelecidas após o arranque do sistema num processo de reserva de recursos para cada mensagem. Por exemplo, no caso do CAN será a reserva de um identificador para uma determinada mensagem.

O protocolo suporta os dois tipos de tráfego, periódico e aperiódico.

- O *Tráfego Periódico* (Cyclic Option) é utilizado tipicamente na malha de controlo. A cada mensagem é associado um período sendo posteriormente enviada segundo esse período;
- O *Tráfego Aperiódico* só é produzido quando há uma variação no valor da variável acima de um limite definido.

Para detectar falhas nos produtores de mensagens aperiódicas o protocolo implementa um mecanismo de "Heartbeat Rate", que não é mais do que um *timer* com um valor predefinido. Se um determinado nó não receber informação de uma determinada variável dentro desse tempo assinala a falha do produtor.

3.1.5 TT-CAN

TT-CAN[fS00] como o nome indica é outro protocolo baseado em CAN. Este protocolo surge da necessidade de resolver alguns problemas que o protocolo CAN tem. O protocolo CAN, como anteriormente foi referido, determina o acesso ao barramento com base no identificador atribuído às mensagens. O acesso ao barramento poderá ser adiado pelos seguintes factores:

- *Quando outra mensagem já se encontra a ser transmitida no barramento;*
- *Quando outra mensagem de mais alta prioridade concorre pelo acesso ao barramento.*

No limite uma mensagem pode ver o seu acesso ao barramento impedido por tempo ilimitado, a este fenómeno dá-se o nome de "Starvation". Na prática o que sucede é um potencial aumento do *jitter* de acesso ao barramento com a diminuição da prioridade da mensagem.

Tendo em conta as limitações do protocolo original, este novo protocolo foi criado com os seguintes objectivos:

- *Reduzir o jitter de acesso ao barramento;*
- *Conferir uma característica determinista ao barramento;*
- *Usar a largura de banda com maior eficiência.*

Para cumprir estes objectivos, o protocolo TT-CAN, determina o envio periódico de uma mensagem principal (Reference Message). Essa mensagem é enviada por um nó especial na rede, o "Time Master". Esta mensagem permite dar uma referência temporal ao sistema, possibilitando a transferência de mensagens durante um tempo definido (Time-Slot). Deste modo, o *jitter* de acesso ao barramento é drasticamente reduzido pois evita o acesso concorrencial ao barramento pelas várias mensagens.

Às janelas temporais onde são transmitidas mensagens síncronas dá-se o nome de "Exclusive Windows". O protocolo permite também reservar janelas temporais para mensagens

assíncronas, a essas janelas dá-se o nome de "Arbitration Windows". Neste espaço de tempo as mensagens acedem ao barramento sendo conferido o acesso à que tiver maior prioridade. A principal diferença do acesso do protocolo CAN é que neste caso as mensagens apenas podem tentar o acesso ao barramento uma única vez, caso não consigam o acesso não voltam a tentar a retransmissão, a este processo dá-se o nome de "Transmissão em *Single-Shot*". Este mecanismo garante que não é excedido o tempo da janela assíncrona.

O tempo entre duas Reference Messages é chamado de "Basic Cycle" (BC). Este ciclo contém várias janelas temporais que possuem diferentes tamanhos e tipos. Ao conjunto completo de BC's dá-se o nome de "System Matrix". Esta matriz é configurada antes do arranque do sistema.

3.1.6 FlexRay

A principal característica deste protocolo[Con01] é a eficiente combinação dos tráfegos síncronos e assíncronos.

Foi desenvolvido para aplicações automóveis sendo utilizado pela BMW, GM, Bosch, Motorola e Philips.

As características deste protocolo são:

- *Elevada taxa de transmissão de dados;*
- *Transmissão com característica determinística;*
- *Tolerância a falhas;*
- *Flexibilidade de forma a suportar os diferentes tipos de componentes de um veículo.*

Este protocolo implementa uma pilha protocolar de quatro níveis:

- *Application Layer;*
- *Presentation Layer;*
- *Transfer Layer;*
- *Physical Layer.*

A comunicação é feita com base em períodos de tempo fixos, designados de "Communication Cycles" (CC) que contêm uma parte dinâmica e uma parte estática. O tráfego síncrono é enviado na parte estática, na parte dinâmica é enviado o tráfego assíncrono.

No começo de cada CC é enviado um símbolo especial, o "Special Control Symbol" (SoC) e a cada janela é atribuído um identificador.

A parte estática é enviada em janelas de duração fixa definida antes do arranque do sistema. As mensagens seguem uma política de escalonamento do tipo TDMA.

A parte Dinâmica é baseada em tempos de espera, usando uma estratégia CSMA/CA. A cada identificador é associado uma prioridade e as mensagens são enviadas com base nessas prioridades. Este tipo de mensagens só é enviada quando há uma solicitação da camada de aplicação.

3.2 Rede Ethernet

3.2.1 Breve descrição do protocolo

O protocolo Ethernet foi criado por Bob Metcalfe, há cerca de 30 anos atrás, no centro de pesquisa da Xerox de Palo ALto. Foi inicialmente criado para ligar um computador a uma impressora. Desde essa altura até aos nossos dias o protocolo foi sofrendo evoluções e tornou-se no standard IEEE 802.3.

Neste momento estão estandardizados três velocidades de transmissão

- *10 Mbits/s* [eth82, IEEa, IEEb, IEEd];
- *100 Mbits/s* [IEEc];
- *1 Gbits/s* [IEEe].

O meio de transmissão usado é um par cruzado e o seu comprimento máximo varia consoante a velocidade de transmissão. Estes comprimentos estão definidos nos standards.

As principais características deste protocolo são:

- *Um único domínio de colisão, ou seja, as mensagens em broadcast são recebidos por todos os nós (NIC - Network Interface Cards) na rede;*
- *O mecanismo de arbitragem, o Carrier Sense Multiple Access With Colision Detection (CSMA/CD).*

O mecanismo de arbitragem actua da seguinte forma:

1. *Quando um NIC tem uma mensagem para ser transmitida espera até o barramento ficar disponível;*
2. *Quando o barramento fica disponível o NIC começa a transmitir;*
3. *Se outros NIC's começarem a transmitir ao mesmo tempo dá-se uma colisão;*
4. *No caso de ocorrer uma colisão todas as estações param a transmissão da respectiva mensagem e passam a transmitir uma sequência especial, a "Jam Sequence", sequência essa que assegura que todas as estações na rede detectam que houve uma colisão;*
5. *De seguida as estações esperam um tempo pseudo-aleatório até voltarem a retransmitir as suas mensagens.*

Este tempo de espera é seleccionado, de forma aleatória, entre um conjunto discreto de valores. O limite superior deste conjunto é duplicado por cada colisão consecutiva, a este mecanismo dá-se o nome de (*Exponencial Back-off*). Após 10 colisões esse valor não cresce mais (*Truncated Exponencial Back-off*). O número máximo de tentativas é de 16, depois desta é assinalado um erro de transmissão.

No caso de sobrecarga da rede este mecanismo diminui a carga útil na rede. Para resolver este problema foi proposta a substituição dos *Hubs* por *Switches*.

Os *Switches* têm as seguintes vantagens:

- Criam um domínio de colisões único para cada nó ligado a ele. Na prática se a cada porta do *switch* estiver ligado apenas um terminal as colisões nunca ocorrem;
- Contém o mapa de correspondência entre as estações e as portas onde estão ligadas. Este mecanismo permite que cada estação apenas receba o tráfego que lhe está endereçado;
- Permite várias transmissões simultâneas entre nós diferentes na rede.

O uso de *switches* permite a partilha de largura de banda e evita as colisões. Estas propriedades permitem aumentar a carga útil no barramento.

A imagem 3.4, retirada de [bip], mostra o formato simplificado de uma trama Ethernet.

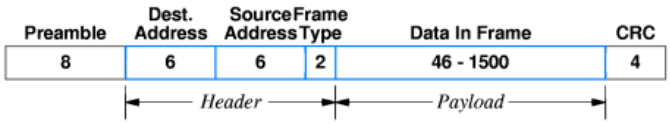


Figura 3.4: Trama Ethernet

A trama Ethernet contém os seguintes campos:

- *Preamble Field (7 Bytes)* - Usado para fazer a sincronização das estações;
- *Start of Frame (1 Byte)* - Indica o início da informação útil;
- *Destiny (6 Bytes)* - Indica o endereço físico da estação de destino;
- *Source (6 Bytes)* - Indica o endereço físico da estação de envio;
- *Data (0 a 1500 Bytes)* - Informação a ser enviada;
- *PAD (0 a 46 Bytes)* - Usado no caso do campo de dados não ter o tamanho mínimo de 46 Bytes;
- *CRC (4 Bytes)*- Detecção e correcção de erros;
- *Interframe Gap (12 Bytes)*- Tempo entre frames consecutivas.

Cada estação é obrigada a possuir um endereço único. O IEEE resolveu o problema atribuindo a cada fabricante um identificador de 3 Bytes, o "Organizational Unique Identifier" (OUI).

A comunicação na rede Ethernet pode ser feita de duas formas:

- *Unicast*

  A estação compara o campo "Destiny" da mensagem que recebeu com o seu endereço físico, se esse endereço corresponder o campo de dados é passado para as camadas superiores da pilha protocolar, caso o endereço de destino não corresponda ao endereço físico da estação a mensagem é descartada;

- *Multicast*

  O protocolo permite definir um conjunto de estações que responderão a um endereço específico de destino, neste caso todas as estações do grupo irão receber este tipo de mensagens. A comunicação em *broadcast* é também possível bastando preencher o endereço de destino com 1's.

A rede Ethernet é vulgarmente usada para realizar redes domésticas e industriais de comunicações, disponibilizando um método simples de interligar sistemas informáticos em rede. A Ethernet possibilita também uma fácil integração com a rede mundial de comunicações, a "Internet". A facilidade de implementação e manutenção e o baixo custo dos componentes fazem com que esta rede seja uma das redes mais utilizadas para elaborar redes de comunicação.

3.2.2 Aplicação a Sistemas Distribuídos

O protocolo Ethernet é um protocolo direccionado para a transferência de ficheiros sem necessidade, por isso, de garantias de tempo-real.

A principal limitação deste protocolo, para aplicações de tempo-real é o seu mecanismo de arbitragem que pelo facto de não ser determinista não permite dar garantias de tempo-real ao sistema.

No entanto o facto de se ter generalizado o uso deste protocolo fez com que o custo de aquisição e manutenção do equipamento seja baixo, comparativamente às outras redes existentes, tornando-a na rede mais fácil de realizar e manter. A sua fácil integração com a rede de Internet mundial e a redes internas de industrias são outros grandes factores que levam a esta rede ser preferida em detrimento de outras.

Para aplicações de tempo-real, como já anteriormente foi referido, esta não é a rede mais indicada, no entanto, é possível fazer algumas adaptações que possibilitam dar garantias temporais às mensagens. Algumas dessas soluções são[Ped]:

- *Master / Slave ;*
- *Token - Passing;*
- *Timed - Token;*
- *TDMA;*
- *Virtual Time Protocol;*
- *Windows Protocol;*
- *Traffic Shaping;*

- *Mini-Sloting;*
- *Switched - Ethernet.*

De todas as soluções a mais eficaz é a utilização de *switches* na rede, esta solução permite resolver o problema das colisões fornecendo um mecanismo baseado em filas de espera.

No entanto o simples uso de um *switch* não é suficiente para conferir propriedades de tempo-real ao sistema. Um protocolo deverá ser adicionado de modo a conferir esse tipo de propriedades.

Existem já implementados alguns desses protocolos, os mais relevantes serão descritos na próxima secção.

O protocolo FTT-SE foi desenvolvido com o objectivo de conferir à rede Ethernet as características de tempo-real necessárias às aplicações de sistemas de controlo distribuído, este protocolo será analisado mais à frente neste capítulo na secção 3.3.2.

3.2.3 Protocolos existentes para Sistemas Distribuídos

Nesta secção é feita uma breve análise às redes com características de tempo-real mais usadas em sistemas distribuídos.

3.2.3.1 ETHERNET Powerlink

ETHERNET Powerlink[PWL] é um protocolo que confere propriedades de tempo-real à rede Ethernet operando sobre Fast-Ethernet.

Este protocolo suporta tanto tráfego Time-trigger (Isócrono) como Event-Trigger (Assíncrono), usando um modelo de comunicação *Master (Powerlink Manager) - Slave (Powerlink Controller)* resolvendo assim o problemas das colisões no meio.

A imagem 3.5, retirada de [PA], mostra o formato do ciclo ETHERNET Powerlink.

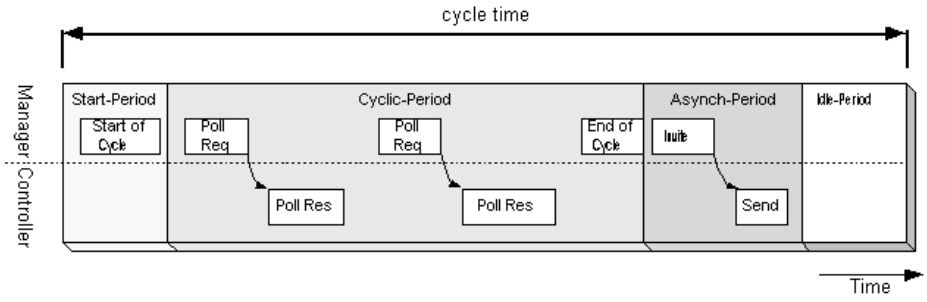


Figura 3.5: Ciclo Elementar do protocolo Powerlink

A troca de informação ocorre em janelas temporais de duração fixa, o "Powerlink Cycle". Este por sua vez divide-se em quatro fases distintas:

- *Start;*

- *Cyclic;*
- *Asynchronous;*
- *Idle Period.*

O *Powerlink Cycle* começa com o envio de uma mensagem do *Manager*, a "Start of Cycle Message", o envio desta mensagem é feito na primeira das fases anteriormente referidas. Esta mensagem é enviada em *broadcast* e a sua função é notificar os *Controller's* de que um novo ciclo vai começar.

Segue-se a fase "Cyclic" onde é enviada o tráfego Isócrono. O *Manager* envia uma mensagem (*PollRequest*) ao *Controller* que vai produzir informação, o *Controller* responde ao pedido com outra mensagem (*PollResponse*), em *broadcast*, que contém a informação desejada. Este mecanismo facilita a distribuição de informação pelos vários nós da rede. No final da fase "Cyclic" o *Manager* envia a mensagem "End of Cycle".

Após a recepção desta mensagem, e se ainda tiver tempo suficiente, dá-se início à fase *Asynchronous* que ocupará o restante tempo até ao ciclo seguinte. Estas mensagens podem ser mensagens Assíncronas de Dados (*Invite/Send*) ou mensagens de Gestão(*Ident/AsyncSend*) usadas pelo *Manager* para detectar estações inactivas. Este tipo de tráfego é também gerido pelo *Manager*. Um *Controller* que tenha mensagens assíncronas para enviar deverá notificar o *Manager* através de um campo específico na mensagem "PoolResponse". No *Manager* estão implementadas filas de espera de mensagens assíncronas.

No final da fase *Asynchronous* é inserido uma janela temporal de tamanho variável chamada de "Idle-Time". A sua função é manter o sincronismo dos ciclos.

O tráfego Ethernet vulgar é também integrado no protocolo sendo enviado na fase *Asynchronous*.

3.2.3.2 EtheReal

O protocolo EtheReal[VC98] é um protocolo que confere à rede Ethernet propriedades de tempo-real.

Este protocolo é implementado em *Switches* sendo estes os responsáveis pela gestão do tráfego na rede conferindo-lhe as propriedades requeridas.

O protocolo suporta os dois tipos de tráfego, tráfego *Time-Trigger* e *Event-Trigger* através de duas classes distintas:

- *Real Time Variable Bit Rate Service Class*

  Dedicado a tráfego de tempo-real. Reserva largura de banda e tenta minimizar o Jitter de transferência das mensagens.

- *Best-Effort Service Class*

  Dedicado a tráfego assíncrono.

A imagem 3.6, retirada de [PA], mostra a arquitectura do protocolo EtheReal.

Os serviços de tempo-real são orientados à conexão, isto é, as aplicações passam por um processo de configuração de ligação antes de poderem transmitir tráfego com estas propriedades. Este processo de configuração é feito da seguinte forma:

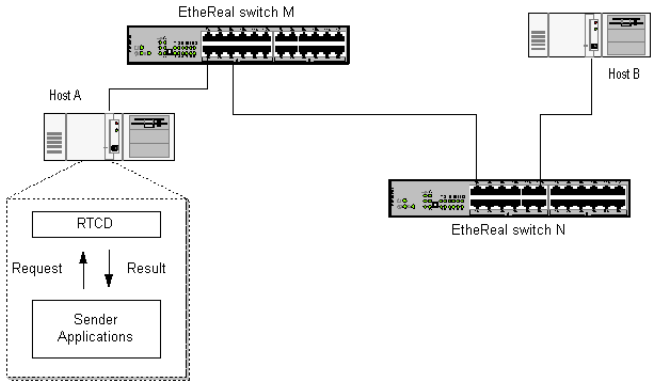


Figura 3.6: Arquitectura EtheReal

1. O processo começa com o envio de um pedido para a camada *Real-Time Comunications Deamon* (RTCD). O pedido de reserva contém os requisitos de QoS da mensagem como o período e o tempo de Burst Máximo, por exemplo.

2. Depois de receber o pedido o RTCD contacta o *switch* EtheReal ao qual está ligado. Este avalia o pedido e determina se tem recursos suficientes para o satisfazer. No caso do destino não estar ligado a este *switch*, este encaminha o pedido para o próximo *switch* e assim sucessivamente até atingir o *switch* ao qual a estação de destino está ligada.

3. No final é enviada uma mensagem de resposta que vai ser propagada pelos *switches* até à estação que fez o pedido. Uma conexão só é estabelecida se todos os *switches*, desde a estação requerente até à estação de destino, tiverem os recursos suficientes para satisfazer os requisitos de QoS solicitados.

A arquitectura EtheReal utiliza "Traffic Shaping" e "Traffic Policing" tanto nas estações como nos *switches*. O primeiro tem como objectivo garantir uma largura de banda constante para o tráfego de tempo-real. O segundo garante que os requisitos de tempo-real das mensagens são cumpridos.

3.2.3.3 Rether

O protocolo Rether[VC94] foi proposto por Venkatramani e Chiueh. Este protocolo opera em modo de Ethernet normal até à chegada de um pedido de tempo-real. Quando esse pedido acontece o modo de funcionamento altera-se para comunicação por passagem de *Token*.

Neste modo o tráfego de tempo-real é considerado periódico e o tempo dividido em ciclos de duração fixa. O acesso ao barramento, tanto pelo tráfego periódico como pelo tráfego aperiódico, é regulado pelo mecanismo de passagem de *Token*.

Primeiramente o *Token* visita todos os nós com mensagens de tempo-real. Depois, se ainda sobrar tempo nesse ciclo, passa pelos nós com tráfego assíncrono.

3.3 FTT - Flexible Time Trigger

3.3.1 Paradigma FTT

O paradigma Flexible Time-Trigger (FTT)[Ped] teve a sua origem no protocolo FTT-CAN[AFF98] desenvolvido nos laboratórios de Sistemas Electrónicos da Universidade de Aveiro.

O protocolo FTT-CAN foi inicialmente desenvolvido com o principal objectivo de conferir determinismo à comunicação sobre o barramento CAN. O indeterminismo inerente ao mecanismo de acesso ao meio presente em CAN (CSMA/CA), é evitado através da presença de um nó principal, *Master*, que gere o acesso ao barramento.

O paradigma FTT é no entanto expansível a outros meios de comunicação, como sejam o *Shared Ethernet* onde o indeterminismo reside no mecanismo de acesso ao meio CSMA/CD, ou o *Switched Ethernet* onde o indeterminismo é implícito nas filas de espera do *switch*.

Este paradigma protocolar suporta ambos os modelos de comunicação, síncrono e assíncrono, definindo a arquitectura do sistema e a interface de software com a camada de aplicação.

Este conjunto de características definidas no paradigma, permitem a sua implementação em diversos meios de comunicação independentemente das restrições e especificidades que cada um possui.

3.3.1.1 Arquitectura do sistema

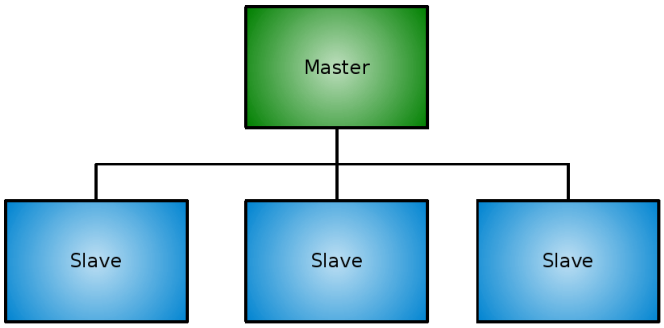


Figura 3.7: Rede com *Master* e *Slaves*

O paradigma FTT implementa o modelo de comunicação "produtor-distribuídor-consumidor" possuindo um nó principal, *Master*, que gere a comunicação entre os nós, *Slaves*, produtores e consumidores. Essa gestão é imposta sobre uma resolução temporal definida por ciclos de duração fixa, EC - Elementary Cycle.

A figura 3.8, retirada de [Ped], mostra a estrutura de um EC que é composto pelos seguintes componentes

- *Trigger Message (TM)*

  A Trigger Message fornece uma referência temporal à comunicação no barramento e indica que mensagens serão enviadas nesse EC.

- *Janela Síncrona*

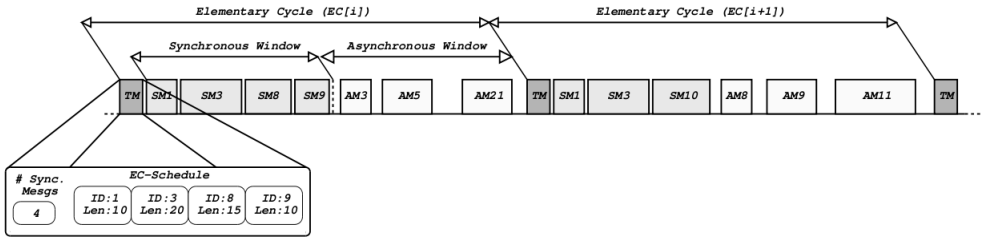


Figura 3.8: FTT-SE Elementary Cycle

Na janela síncrona é enviado o tráfego síncrono de acordo com a informação presente na TM. O tamanho desta janela pode variar de EC para EC de acordo com a quantidade e tamanho das mensagens enviadas.

- *Janela Assíncrona*

  Esta janela terá a duração restante do EC não utilizada pela janela síncrona. Nesta janela é enviado o tráfego Event-Trigger.

- *Idle-Time*

  De modo a manter a sincronização da rede é necessário introduzir um tempo de guarda entre a janela assíncrona e a TM do próximo EC.

3.3.1.2 Modelo de Interface de Software

A interacção com o paradigma FTT é feita tendo por base dois modelos de comunicação, *Event-Trigger* e *Time-Trigger*, que se reflectem nos dois tipos de tráfego, síncrono e assíncrono. No tráfego síncrono existe uma noção de estado, ou seja, existe uma variável no sistema que vai ser actualizada em períodos de tempo bem definidos, a informação presente nessa variável só é válida nesse intervalo de tempo. A noção de evento é inerente ao tráfego assíncrono onde cada mensagem produzida é tratada como única no sistema. A principal diferença no tratamento das mensagens é que no primeiro caso, tráfego síncrono, existe espaço alocado do tamanho da variável produzida sendo a informação desta alterada com a recepção de mensagens. No tráfego assíncrono existe a noção de filas de espera, ou seja, por cada variável recebida é alocada memória e colocada numa fila de espera própria.

A interacção entre os *Slaves* e o *Master* é feita com base na reserva de recursos através de um processo de negociação. Os recursos alocados poderão ser renegociados dinamicamente durante o funcionamento do sistema sem que para isso seja necessário parar a comunicação em nenhum nó da rede.

3.3.1.3 Tolerância a Falhas

No protocolo FTT-CAN foram implementados os seguintes mecanismos de tolerância a falhas

- *Replicação do Master*

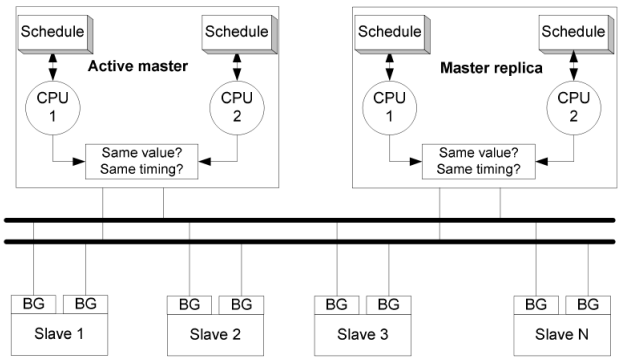


Figura 3.9: Mecanismos de Tolerância a Falhas implementados no Protocolo FTT-CAN

A função de um segundo *Master* é monitorizar o funcionamento do *Master* principal e substituí-lo, em tempo útil, caso ocorra uma falha;

- *Replicação do Barramento*

  Todas as estações possuem um módulo cuja função é detectar e corrigir a perda de ligação num dos barramentos;

- *BusGuardian*

  Dispositivo implementado nos *slaves* que detecta se as mensagens são produzidas no tempo correcto, ou seja, no respectivo EC. Caso ocorra um erro, este dispositivo impede o acesso ao barramento pelo nó onde está ligado. Implementa uma política de "Fail-Silent";

- *Replicação de Processamento*

  Replicação do processamento do algoritmo de escalonamento no *Master*. Um mecanismo auxiliar compara os resultados dos processamentos e caso não sejam coerentes impede o envio da TM. Implementa uma política de "Fail-Silent".

Este tipo de mecanismos deverão ser utilizados por forma a garantir que o sistema, face a uma situação de falha, se consiga adaptar e continuar a fornecer as garantias de tempo-real requeridas. Geralmente estes mecanismos são utilizado em sistemas distribuídos de Hard-Real-Time, ou seja, em que a falha dos mesmos poderia significar a perda de vidas humanas.

A aplicação destes mecanismos noutros meios de comunicação dependerá das características específicas de cada meio em particular.

Estes mecanismos ainda só foram implementados no protocolo FTT-CAN estando prevista a inclusão no paradigma FTT em futuras implementações noutros meios de comunicação.

3.3.2 FTT - SE

3.3.2.1 Funcionalidades do Protocolo

Como já foi referido anteriormente neste capítulo o mecanismo de arbitragem do protocolo Ethernet é o principal responsável pela não adequação a tráfego de tempo-real.

O protocolo FTT-SE propõe-se satisfazer os requisitos [Ped] que são abaixo descritos:

- *Comunicação time-trigger com flexibilidade operacional;*
- *Suporte para alterações dinâmicas tanto no conjunto de mensagens como na política do escalonador;*
- *Controlo de admissão dinâmico a fim de garantir precisão temporal ao tráfego de Tempo-Real;*
- *Indicação do cumprimento temporal das mensagens;*
- *Suporte para os diferentes tipos de tráfego: event-trigger, time-trigger, hard real-time, soft-real-time e non-real-time;*
- *Isolamento Temporal: os diferentes tipos de tráfego não deverão provocar interferências entre si;*
- *Uso eficiente da largura de banda;*
- *Suporte eficiente para mensagens multicast.*

3.3.2.2 Trama FTT-SE

A trama FTT-SE contém os campos mostrados na figura 3.10, retirada de [Ped].

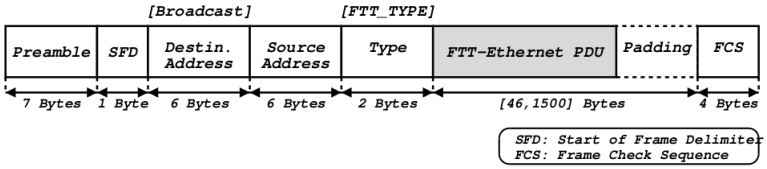


Figura 3.10: Trama FTT-SE

- *Preamble Field (7 Bytes)* - Usado para fazer a sincronização das estações;
- *Start of Frame (1 Byte)* - Indica o início da informação útil;
- *Destiny (6 Bytes)* - Indica o endereço físico da estação de destino;
- *Source (6 Bytes)* - Indica o endereço físico da estação de envio;
- *Type (2 Bytes)* - Indica o tipo de mensagem FTT;
- *FTT-SE PDU (4 a 1500 Bytes)* - Campo de configuração e de dados do protocolo FTT;

- *Padding (0 a 42 Bytes)* - Usado no caso do campo de dados não ter o tamanho mínimo de uma mensagem, 64 Bytes;
- *FCS (4 Bytes)* - Detecção e correcção de erros.

3.3.2.3 Tipos de Mensagens

O protocolo define os seguintes tipos de mensagens:

- *EC Trigger Message [TM_MESG_ID];*
- *Mensagens síncronas de dados [SM_DATA_MESG_ID];*
- *Mensagens assíncronas de dados [AM_DATA_MESG_ID];*
- *Mensagens de controlo [CONTROL_MESG_ID];*
- *Mensagens Ethernet Originais.*

3.3.2.4 Ciclo Elementar

A estrutura do ciclo elementar segue a mesma estrutura apresentada no paradigma FTT.

O EC começa com a TM, que neste caso contém a quantidade, identificação e tamanho das mensagens síncronas que deverão ser produzidas nesse EC. Seguidamente são enviadas as mensagem síncronas, na janela síncrona, e finalmente as mensagens assíncronas na janela com o mesmo nome.

3.3.2.4.1 Trigger Message

O formato da informação presente na trigger message é mostrado na figura 3.11, retirada de [Ped].

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 3.11: FTT-SE Trigger Message

- *Type*
  - TM_TYPE - deverá ter o identificador da Trigger Message [TM_MESG_ID];
  - MASTER_ID - Contém o identificador do *Master* da rede.
- *TM_FLAGS*

  - Reserved;
  - Sequence Number - Incrementado pelo *Master* em cada EC. Este campo serve para a detecção da perda de TM's.

- *Number of Sincronous Messages* - Contém o número de mensagens síncronas que irão ser enviadas nesse EC.

- *ID + Tx_Time* - Nestes campos são enviados os identificadores das mensagens síncronas que serão produzidas nesse EC bem como o tempo que estas dispõem de acesso ao barramento em μs.

3.3.2.4.2 Mensagens Síncronas de Dados

O formato da informação presente na mensagem síncrona é mostrado na figura 3.12, retirada de [Ped].

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 3.12: Formato das Mensagens Síncronas de Dados

- *Type*
  - SDM Type - deverá ter o identificador de Mensagem Síncrona [SM_DATA_MESG_ID];
  - SMD_ID - Contém o identificador da estação que envia esta mensagem.

- *SDM Flags*
  - Reserved;
  - Sequence Number - Incrementado pela estação em cada mensagem que envia. Este campo serve para a detecção da perda de mensagens.

- *Time to Deadline* - Este campo é usado para fornecer informação de "idade" aos dados enviados;

- *Data* - Os dados propriamente ditos.

3.3.2.4.3 Mensagens Assíncronas de Dados

O formato da informação presente na mensagem assíncrona de dados é mostrado na figura 3.13, retirada de [Ped].

- *Type*

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 3.13: Formato das Mensagens Assíncronas de Dados

  - ADM Type - deverá ter o identificador de Mensagem Assíncrona de Dados [AM_DATA_MESG_ID];
  - ADM_ID - Contém o identificador da estação que envia esta mensagem.

- *ADM Flags*
  - Reserved;
  - Sequence Number - Incrementado pela estação em cada cada mensagem que envia. Este campo serve para a detecção da perda de mensagens.

- *Time to Deadline* - Este campo é usado para fornecer informação de "idade" aos dados enviados;

- *Data* - Os dados propriamente ditos.

3.3.2.4.4 Mensagens Assíncronas de Controlo

O formato da informação presente na mensagem assíncrona de controlo é mostrado na figura 3.14, retirada de [Ped].

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 3.14: Formato das Mensagens Assíncronas de Controlo

- *Type*
  - CM Type - deverá ter o identificador de Mensagem Assíncrona de Controlo [CONTROL_MESG_ID];
  - CM_ID - Contém o identificador da estação que envia esta mensagem.

- *ADM Flags*

  - Reserved;
  - Sequence Number - Incrementado pela estação em cada mensagem que envia. Este campo serve para a detecção da perda de mensagens.

- *Time to Deadline* - Este campo é usado para fornecer informação de "idade" aos dados enviados;
- *Data* - Os dados propriamente ditos.

Parte II

Implementação

Capítulo 4

Análise dos Sistemas Físicos Usados

Neste capítulo é feita uma introdução às plataformas didácticas utilizadas nesta dissertação descrevendo as suas principais características. Seguidamente é apresentado um modelo matemático dos Sistemas Físicos usados. Por último é explicado o procedimento usado para distribuir o controlo nas plataformas.

4.1 Propósito das Plataformas Didácticas

As plataformas didácticas são geralmente usadas para estudar, simular e implementar algoritmos de controlo de sistemas físicos. Como foi referido na secção 2.1.1 existem diversos tipos de sistemas podendo, por exemplo, um sistema hidráulico ser transformado num sistema eléctrico com as mesmas características e comportamento. Estas plataformas fornecem um meio prático de estudo destes sistemas. Algumas dessas plataformas são:

- *Bola na Calha* - Sistema cujo objectivo é manter uma bola no centro de uma calha, podendo apenas actuar sobre o ângulo desta. Este sistema é instável em malha aberta, ou seja, basta uma pequena perturbação no ângulo da calha para que a bola deixe a sua posição de equilíbrio divergindo para um dos lados.

  A figura 4.1, retirada de [Lim],mostra uma implementação deste sistema.

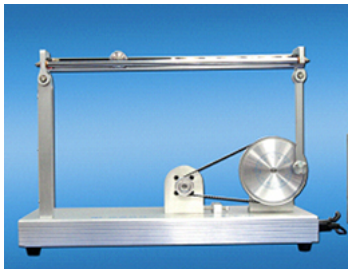


Figura 4.1: Plataforma Bola na Calha, exemplo de aplicação

- *Bola no Plano* - Este sistema é uma generalização do sistema "Bola na Calha" para duas dimensões. Neste caso a bola possui dois graus de liberdade de movimento.

As figuras 4.2 e 4.3, retiradas de [oPape] e [oPle] respectivamente ,mostram implementações deste sistema.

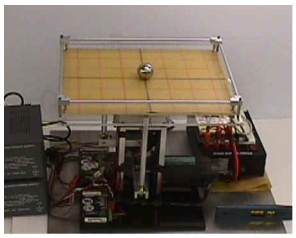


Figura 4.2: Plataforma Bola no Plano, exemplo 1 de aplicação

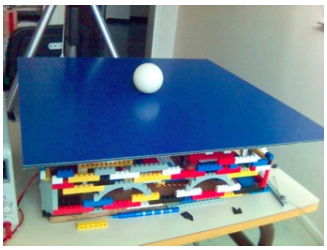


Figura 4.3: Plataforma Bola no Plano, exemplo 2 de aplicação

- *Pêndulo Invertido* - Sistema cujo objectivo é equilibrar uma barra na posição vertical podendo apenas actuar sobre a posição horizontal do mecanismo de suporte da referida barra. Este sistema representa um caso cujo projecto de controlo tem uma complexidade considerável , isto porque a modelação do sistema não é trivial. Este sistema, à semelhança dos anteriores, é instável em malha aberta, ou seja, se a barra for deixada na posição de equilíbrio basta um pequeno movimento para esta divergir para outra posição não desejada.

  Existem algumas variações a esta plataforma, são o caso do "Duplo Pêndulo Invertido" e do "Triplo Pêndulo Invertido" que introduzem o controlo de sistemas caóticos.

  A figura 4.4, retirada de [Lim],mostra uma implementação deste sistema.

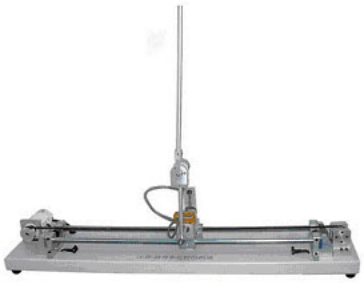


Figura 4.4: Plataforma Pêndulo Invertido, exemplo de aplicação

- *Elevador* - Problema clássico de controlo da posição de um motor. Neste caso o objectivo é parar o ascensor numa determinada posição tendo em conta as especificações pretendidas. Este sistema é estável em malha aberta, ou seja, se o elevador for abandonado a si próprio este não diverge apenas não poderá parar na posição desejada. Neste caso o controlo prende-se apenas com a garantia que o sistema converge para o valor desejado.

  A figura 4.5, retirada de [Kit],mostra uma implementação deste sistema.

Figura 4.5: Plataforma Elevador, exemplo de aplicação

- *Torradeira* - Sistema cujo o objectivo é controlar a temperatura de um processo. Este sistema é em tudo semelhante ao anterior, ou seja, neste caso é também um sistema estável em malha aberta, o que se quer garantir é que a temperatura se mantenha num determinado nível pretendido.

  A figura 4.6, retirada de [Exa],mostra uma implementação deste sistema.

Figura 4.6: Plataforma Torradeira, exemplo de aplicação

- *Fonte de Alimentação* - Tal como nos dois sistemas anteriores neste caso quer-se garantir que a tensão, ou corrente, se mantém num valor pretendido.

  A figura 4.7, mostra uma implementação deste sistema.

Figura 4.7: Fonte de Alimentação, exemplo de aplicação

No caso concreto desta dissertação as plataformas são utilizadas para medir a eficiência da aplicação de um protocolo de comunicação na rede de controlo.

4.2 Plataformas Usadas - Breve Descrição

Esta dissertação teve como base o trabalho realizado anteriormente [Roq07] pelo que os pormenores relativos à construção e montagem do Hardware estão descritos no documento citado com maior pormenor do que serão descritas neste documento.

Nesta dissertação foram usadas duas plataformas descritas na secção anterior, essas plataformas são:

- *Bola na Calha;*
- *Bola no Plano.*

As plataformas usadas possuem as seguintes características:

- *Bola na Calha* - Sistema composto por uma barra de metal possuindo nas pontas sensores ópticos para a medição da posição da bola. A actuação do sistema é feita através de um servomecanismo controlado por um microcontrolador.

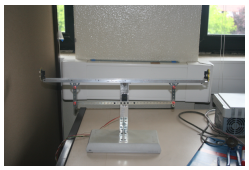


Figura 4.8: Plataforma Bola na Calha

- *Bola no Plano* - Sistema composto por uma superfície plástica revestida com cartolina de cor preta, possuindo no centro uma marca indicando a posição desejada da bola. A posição da bola é determinada com base na imagem de uma webcam situada por cima da referida plataforma. A actuação no sistema é assegurada por dois servomecanismos controlados por um microcontrolador.

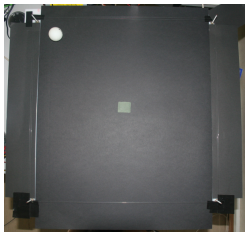


Figura 4.9: Plataforma Bola no Plano

4.3 Modelação das Plataformas

Nesta secção é feita uma análise matemática aos sistemas usados. Numa primeira parte apresenta-se o modelo matemático do sistema convertido para transformadas Z. A opção por apresentar o sistema desta forma reside no facto de poder encontrar uma equação algébrica que pudesse ser aplicada directamente ao algoritmo do controlador usado.

O principal objectivo desta secção é mostrar que com o modelo matemático discreto do controlador PID usado é possível controlar o sistema físico apresentado. Toda a simulação é realizada tendo por base os modelos polinomiais do sistema e do controlador.

4.3.1 Modelação Matemática

Em trabalhos anteriores[Roq07] foi realizada uma modelação do sistema em transformada S. O resultado dessa modelação foi o seguinte

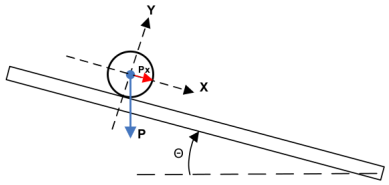


Figura 4.10: Modelo das forças que actuam na bola

$$X(S) = \frac{g}{S^2} \times \Theta(S) \tag{4.1}$$

Onde g representa a aceleração gravítica, θ o ângulo do suporte e X a posição da bola.

Para converter a equação do sistema 4.1 para transformada Z aplicou-se a seguinte conversão

$$S = \frac{2}{T} \times \frac{Z-1}{Z+1} \tag{4.2}$$

Onde T representa a frequência de amostragem do sistema.

Aplicando 4.2 à equação 4.1 obtém-se

$$X(Z) = \frac{g}{(\frac{2}{T} \times \frac{Z-1}{Z+1})^2} \times \Theta(Z) \tag{4.3}$$

Depois de algumas simplificações chega-se à equação

$$(4.Z^2 - 8.Z + 4).X(Z) = (g.T^2.Z^2 + 2.g.T^2.Z + g.T^2).\Theta(Z) \tag{4.4}$$

Dividindo ambos os membros da equação 4.4 por Z^{-2} obtém-se

$$(4 - 8.Z^{-1} + 4.Z^{-2}).X(Z) = (g.T^2 + 2.g.T^2.Z^{-1} + g.T^2.Z^{-2}).\Theta(Z) \tag{4.5}$$

Aplicando a transformada inversa obtém-se a seguinte equação

$$4.x(n) - 8.x(n-1) + 4.x(n-2) = g.T^2.\theta(n) + 2.g.T^2.\theta(n-1) + g.T^2.\theta(n-2) \quad (4.6)$$

Por fim chega-se à equação que descreve o sistema físico

$$x(n) = \frac{1}{4}.g.T^2.\theta(n) + \frac{1}{2}.g.T^2.\theta(n-1) + \frac{1}{4}.g.T^2.\theta(n-2) + 2.x(n-1) - x(n-2) \quad (4.7)$$

Foi realizado um teste no Matlab atribuindo o valor 0.033 ao T e como valores iniciais a posição $0m$ da bola e o ângulo de 1^o do suporte. O teste foi realizado comparando o modelo matemático em transformadas S e em transformadas Z. O código usado para gerar estes gráficos poderá ser consultado nos anexos, secção A.1 deste documento.

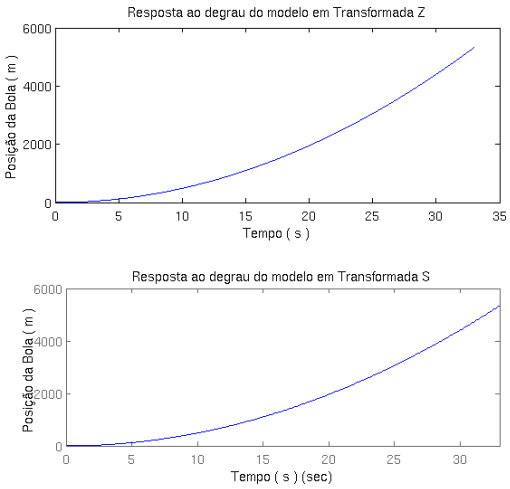


Figura 4.11: Comparação do desempenho dos modelos submetidos ao degrau unitário

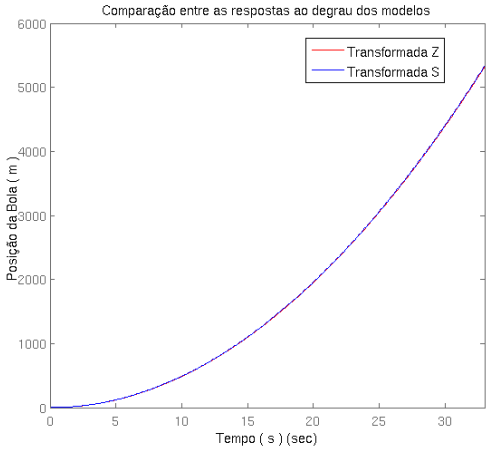


Figura 4.12: Comparação do desempenho dos modelos submetidos ao degrau unitário, gráfico sobreposto

Esta técnica permitiu encontrar uma aproximação muito aceitável do sistema no domínio digital o que possibilita testar algoritmos de controlo sem ser necessário convertê-los para o domínio S. Na secção seguinte é mostrado como foi aplicado o algoritmo de controlo e os resultados que se obtiveram.

4.3.2 Projecto do Controlador

Como já anteriormente foi referido usou-se para controlar as plataformas compensadores PID. Estes compensadores podem ser descritos pela seguinte equação

$$out(n) = Kp.e(n) + Ki.\sum_{i=0}^{n} e(i) + Kd.(e(n) - e(n-1)) \quad (4.8)$$

Onde Kp, Ki e Kd são respectivamente as constantes proporcional, integral e derivativa e e o valor de erro do sistema.

Ao modelo do sistema foi implementado um compensador deste tipo, sendo os seus parâmetros ajustados para conseguir a resposta mostrada no gráfico da figura 4.13. Nesta simulação entrou-se com os efeitos de arredondamento dos valores, comuns na aquisição e armazenamento de valores nos sistemas computacionais e partiu-se de valores iniciais da posição da bola ($-0.2m$) e do ângulo do suporte (0.1^o). Os valores usados nesta simulação foram os valores retirados da plataforma "Bola no Plano". O código que produziu estes resultados pode ser consultado nos anexos deste documento, secção A.2.

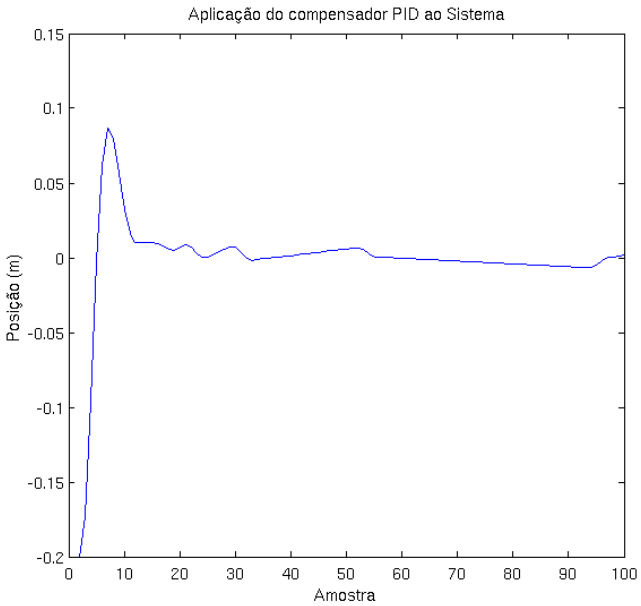


Figura 4.13: Aplicação do compensador PID ao modelo do Sistema

4.4 Plataformas como Sistemas Distribuídos

Como foi referido na secção 2.3.1 um sistema distribuído supõem que, numa rede, um conjunto de processadores troquem informação com o fim de obter o comportamento desejado do sistema. No caso concreto das plataformas usadas o controlo foi distribuído pelos diferentes componentes que compõem um sistema, ou seja:

- *Sensor* - componente responsável pela obtenção e tratamento da informação do "mundo". Este componente adquire informação, trata-a e envia-a para a rede de comunicação.
- *Controlador* - componente responsável pela execução do algoritmo de controlo. Neste componente estão implementados os métodos de controlo.
- *Actuador* - componente responsável pelo controlo dos motores da plataforma. Este componente é o responsável por converter a informação da rede em informação para os *drivers* dos respectivos motores.

Aplicando este método de distribuição às plataformas tem-se:

- *Bola na Calha*
  - Sensor - Computador ligado a um microcontrolador que faz a leitura dos sensores ópticos. O computador comunica com o microcontrolador via protocolo RS232;
  - Controlador - Computador que executa o algoritmo de controlo com base na informação proveniente do sensor;
  - Actuador - Computador ligado a um microcontrolador, via protocolo RS232, que faz a conversão entre a mensagem recebida do controlador e a mensagem a enviar via protocolo RS232.

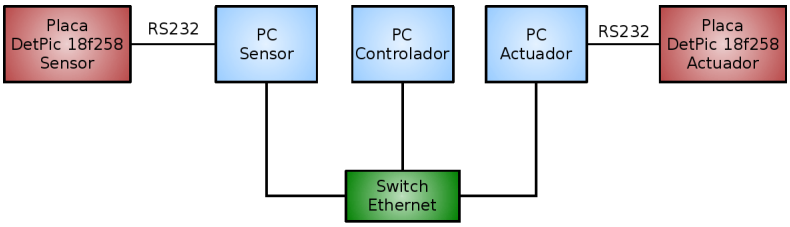


Figura 4.14: Sistema Bola na Calha

- *Bola no Plano*
  - Sensor - Computador ligado a uma webcam. Este componente poderá fazer o processamento da imagem localmente ou enviar a imagem, via rede Ethernet, ao controlador para que este a faça;
  - Controlador - Computador que corre o processamento de imagem caso este não seja feito no sensor, e executa o algoritmo de controlo;
  - Actuador - Computador ligado a um microcontrolador, via protocolo RS232, que faz a conversão entre a mensagem recebida do controlador e a mensagem a enviar via protocolo RS232.

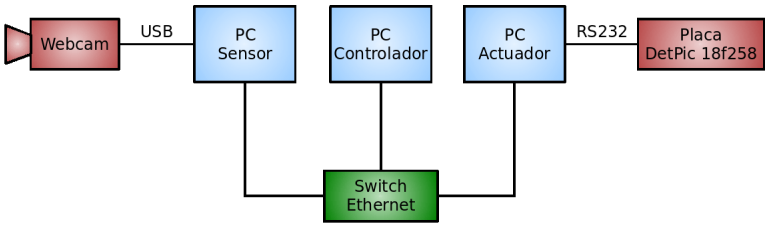


Figura 4.15: Sistema Bola no Plano

Nas plataformas desta dissertação todos os nós cooperam trocando mensagens entre si. Não foi implementado nenhum mecanismo de tolerância a falhas, no entanto é fácil verificar que o nó controlador é redundante, caso fosse detectada uma falha no envio de uma mensagem poderia ser facilmente substituído por um dos restantes nós, por exemplo o actuador, passando este a executar o algoritmo de controlo.

Capítulo 5

Plataforma Bola na Calha

Neste capítulo são detalhados os procedimentos inerentes à implementação do Sistema de Controlo da plataforma "Bola na Calha". Primeiramente será feita uma descrição do Hardware do sistema finalizando com o método de montagem da plataforma. Por último é detalhado o Software implementado explicando as diferentes opções que se tomaram ao longo do decorrer da realização desta dissertação.

5.1 Arquitectura do Sistema

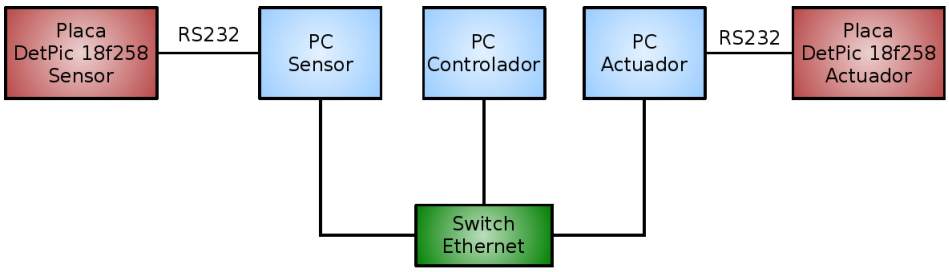


Figura 5.1: Arquitectura do Sistema Bola na Calha

O objectivo deste sistema é manter a bola no centro da calha podendo apenas actuar no ângulo desta. A informação da posição da bola é adquirida através de dois sensores ópticos colocados nas pontas da calha. A actuação é feita através da alteração do ângulo do servomecanismo que está ligado à calha. O valor dos sensores é lido, depois de passar pela placa de filtragem analógica do sinal, através das ADC's do microcontrolador da placa "DetPic 18F258 Sensor". O sinal que é aplicado no servomecanismo é gerado na placa "DetPic 18F258 Actuador", esse sinal depois de gerado é amplificado no módulo de potência e depois aplicado no servomecanismo.

Da distribuição do sistema e da divisão das tarefas de controlo nos computadores "Sensor", "Controlador" e "Actuador" passou a haver um fluxo de informação entre estes componentes. O computador "Sensor" tem como função receber a informação proveniente da placa "DetPic 18F258 Sensor" através do protocolo RS232 e encaminhar essa informação, via Ethernet, para o computador "Controlador". Neste computador essa informação é filtrada, é executado o

algoritmo de controlo e é enviado o resultado desse algoritmo para o computador "Actuador". Este último converte a informação recebida, via Ethernet, para o protocolo RS232 que comunica com a placa "DetPic 18F258 Actuador".

Na figura 5.1 é mostrado o diagrama de blocos do sistema contendo os seus diferentes componentes e o modo como estes se interligam.

Para interligar todos este componentes foi necessário criar protocolos de comunicação de modo a que fosse possível distinguir os diferentes tipos de informação trocada.

Neste sistema existem dois tipos de comunicação, a comunicação via RS232 e via Ethernet. As mensagens enviadas têm o seguinte formato:

- *RS232*

  - *Mensagens do tipo "MS"*

    Mensagens trocadas entre a placa "DetPic 18F258 Sensor" e o computador "Sensor". Nestas mensagens vai a informação dos dois sensores ópticos. Cada valor dos sensores ocupa 2 Bytes pelo que são necessárias 2 mensagens para cada sensor óptico para se conseguir transferir a informação para o computador. A essa informação foi adicionado um identificador que permite fazer a sincronização da informação no receptor.

    Na figura 5.2 é mostrado o formato destas mensagens.

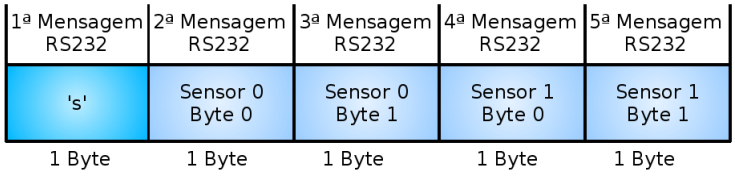


Figura 5.2: Mensagens do tipo "MS"

  - *Mensagens do tipo "MA"*

    Mensagens trocadas entre o computador "Actuador" e a placa "DetPic 18F258 Actuador". A informação presente nestas mensagens é essencialmente o valor de actuação do servomecanismo. Este valor ocupa apenas 1 *Byte* pelo que são necessárias apenas 2 mensagens para enviar esta informação, uma para o identificador e outra para o valor.

    Na figura 5.3 é mostrado o formato destas mensagens.

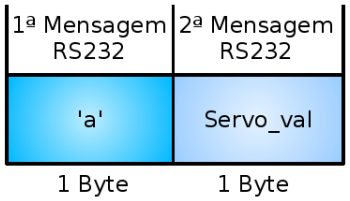


Figura 5.3: Mensagens do tipo "MA"

- *Ethernet*

  – *Mensagens do tipo "S"*

  Mensagens trocadas entre o computador "Sensor" e o computador "Controlador". Estas mensagens possuem a informação dos sensores ópticos. A única alteração efectuada aos valores é a sua conversão para inteiros passando a ocupar 4 *Bytes*. Para enviar esta mensagem são necessários 9 *Bytes*, 2 valores de 4 *Bytes* cada e 1 *Byte* para o identificador.

  Na figura 5.4 é mostrado o formato destas mensagens.

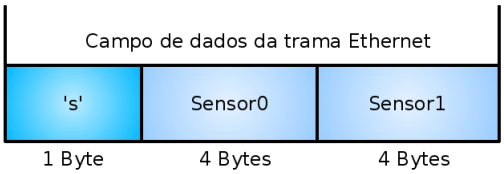


Figura 5.4: Mensagens do tipo "S"

  – *Mensagens do tipo "A"*

  Mensagens trocadas entre o computador "Controlador" e o computador "Actuador". Estas mensagens possuem o resultado do algoritmo de controlo, o valor de actuação do servo.

  Na figura 5.5 é mostrado o formato destas mensagens.

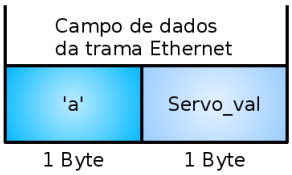


Figura 5.5: Mensagens do tipo "A"

Neste sistema foi aplicado um Filtro de Medianas para filtrar a informação dos sensores. Esse filtro foi implementado no computador "Controlador" e processa um conjunto de 11 amostras.

O filtro de medianas é um filtro de fácil implementação que no caso de o número de amostras ser ímpar o valor de saída é um valor lido e não um valor calculado como por exemplo no filtro de médias.

O princípio de funcionamento do filto de medianas[pri] determina que a probabilidade de, num conjunto de valores, encontrar um superior ao valor de saída é igual à probabilidade de encontrar um valor inferiror.

Este princípio traduz-se num algoritmo de fácil implementação visto apenas ser necessário ordenar os valores das amostras e retirar, no caso do número de amostras ser ímpar, o valor central do conjunto.

Na imagem 5.6 é mostrado um exemplo da aplicação deste filtro a um conjunto de valores.

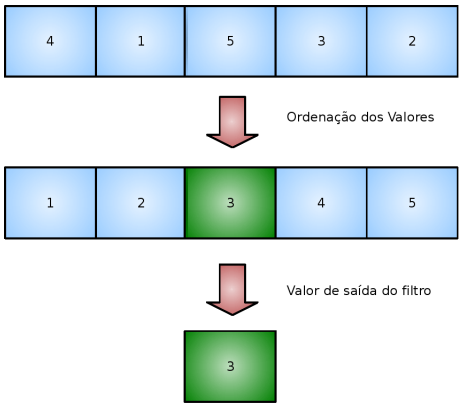


Figura 5.6: Exemplo de aplicação de um filtro de medianas

5.2 Breve Descrição do Hardware

5.2.1 Descrição dos Componentes

A plataforma "Bola na Calha" é constituída pelos seguintes componentes:

- *Barra metálica* - Suporte da bola;
- *2 Sensores Ópticos* - possibilitam a determinação da posição da bola na calha;
- *1 Servomecanismo* - permite alterar o ângulo da calha;
- *Módulo dos Sensores* - placa de filtragem dos sensores;
- *Módulo de Potência* - placa com o driver do servomecanismo;
- *2 Placas DetPic 18F258* - placas com os microcontroladores 18F258 da Microchip©;
- *3 Computadores* - Sensor, Controlador e Actuador respectivamente.

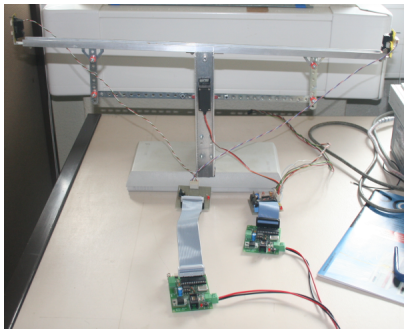


Figura 5.7: Plataforma Bola na Calha Montada

5.2.1.1 Sensores Ópticos

Os sensores ópticos usados nesta plataforma, os Sharp GP2D12 2X, permitem determinar a posição da bola na calha. Estes sensores convertem a distância da bola numa tensão variável sendo esta lida, através da ADC, pelo microcontrolador.

A principal alteração ao trabalho anterior foi realizada nestes sensores. Para possibilitar a distribuição do sistema foi realizada uma placa suplementar composta essencialmente por dois filtros passa-baixo com frequência de corte de 338Hz. Esta placa era depois ligada à placa "DetPic 18F258 Sensor" posteriormente ligada ao computador "Sensor". O esquema desta placa poderá ser consultado no anexo C deste documento.

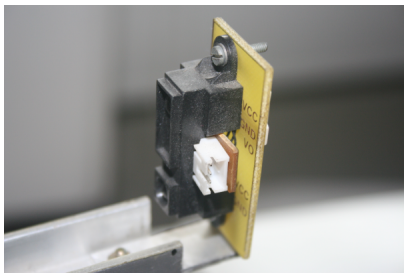


Figura 5.8: Sensores Ópticos Usados

Apenas há a referir que o módulo de potência é o mesmo dos trabalhos anteriores pelo que é sempre possível voltar a uma arquitectura centralizada bastando para isso usar apenas o módulo de potência do mesmo modo que foi utilizado nos trabalhos anteriores [Roq07].

5.2.1.2 Servomecanismo

Nesta plataforma é usado o servomecanismo, Dymond 4000 Servo, para actuar no ângulo da plataforma. Esse ângulo é controlado através de um sinal modulado em largura (PWM) com uma frequência de 50Hz gerado pelo microcontrolador e amplificado no módulo de potência.

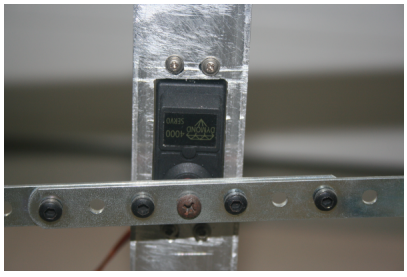


Figura 5.9: Servomecanismo Usado

O servomecanismo não foi ligado directamente à calha, este foi ligado por meio de uns veios metálicos que modificam o ponto de apoio reduzindo assim a carga que a calha exerce no servomecanismo.

5.2.1.3 Redes de Comunicação

Existem duas redes que possibilitam a interligação dos diversos componentes do sistema:

- *RS232* - Ligação entre as placas DetPic 18F258 e os computadores;
- *Ethernet* - Ligação entre os computadores.

Na ligação via RS232 é usado um Baudrate de 115200. Na rede Ethernet é usado uma ligação de 100Mbps usando um *switch* para interligar as estações.

5.2.2 Montagem da Plataforma

Para efectuar a correcta montagem dos diversos componentes da plataforma deve-se seguir os seguintes passos:

1. Ligar os sensores ao Módulo dos Sensores.

   Sensor 1

   - Vcc - Vermelho
   - GND - Branco
   - Vo - Azul

   Sensor 2

   - Vcc - Vermelho
   - GND - Branco
   - Vo - Verde

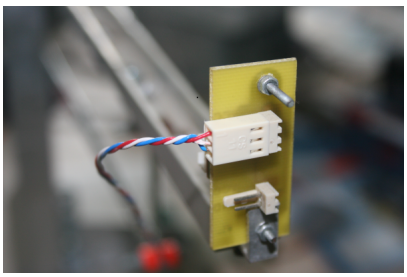


Figura 5.10: Ligação do Sensor 1

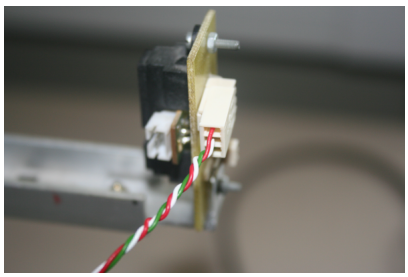


Figura 5.11: Ligação do Sensor 2

2. Ligar o servomecanismo ao módulo de potência.
3. Ligar os Módulo de Potência e dos Sensores às respectivas placas DetPic 18F258 através dos cabos próprios para o efeito.
4. Ligar cada uma das placas DetPic 18F258 ao computador respectivo através de ligações RS232. A placa DetPic 18F258 ligada ao Módulo dos Sensores deverá ser ligada ao computador "Sensor", a placa ligada ao Módulo de Potência terá que ser ligada ao computador "Actuador".

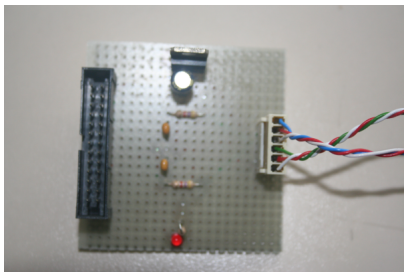


Figura 5.12: Ligação dos Sensores ao Módulo dos Sensores

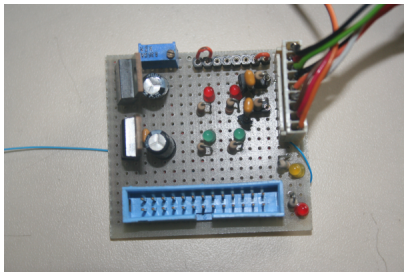


Figura 5.13: Ligação do Servomecanismo ao Módulo de Potência

5. Ligar os computadores em rede através do *switch*.

A alimentação das placas DetPic 18F258 é feita utilizando fontes de alimentação independentes. No entanto há que ter em atenção que as massas destas fontes devem estar ligadas.

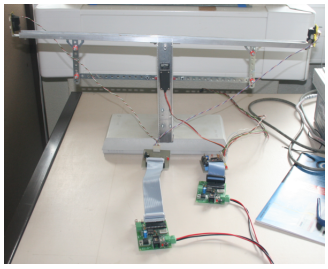


Figura 5.14: Ligação dos Módulos às Placas DetPic18F258

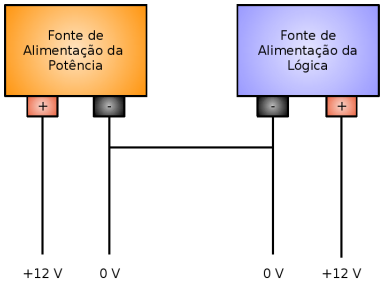


Figura 5.15: Ligação das Massas das Fontes de Alimentação

5.3 Descrição Geral do Software dos Microcontroladores

5.3.1 Estrutura do Software

Todo o software produzido obedeceu a uma arquitectura modular, sendo constituído por módulos estanques que possibilitam um desenvolvimento faseado do projecto bem como um método eficaz de detecção de problemas pela implementação de programas de teste de cada módulo.

O código necessário aos microcontroladores encontra-se dividido nos seguinte módulos

- *serial* - módulo que possibilita a transferência de informação entre os microcontroladores e os computadores;
- *adc* - módulo que possui as funções de configuração e operação das ADC's necessárias à leitura dos valores dos sensores;
- *motor* - módulo que possui as funções de configuração e operação do servomecanismo usado na plataforma;
- *sensor* - programa principal que corre no microcontrolador responsável pela aquisição da informação dos sensores ópticos;
- *actuador* - programa principal que corre no microcontrolador responsável pela geração do sinal de controlo do servomecanismo.

Os módulos *serial*, *adc* e *motor* contêm, cada um deles, um programa de teste que possibilita a detecção de falhas nos respectivos módulos. Os programas de teste poderão ser compilados através das *Makefiles* de cada módulo.

Todos os módulos possuem uma *Makefile* que permite a geração dos executáveis. Para compilar o projecto basta, na pasta principal, executar o seguinte comando

1. make clean
2. make all
3. make install

Nas secções seguintes será feita uma descrição mais detalhada dos códigos do sensor e do actuador implementados.

5.3.1.1 Sensor

A função deste microcontrolador é essencialmente ler os valores dos sensores através das ADC's e enviá-los via porta série. Como os valores retornados pelas ADC's ocupam 2 Bytes, estes valores têm que ser convertidos para um array de Bytes para serem enviados.

Na figura 5.16 é mostrado o diagrama de blocos do código implementado no microcontrolador "Sensor"

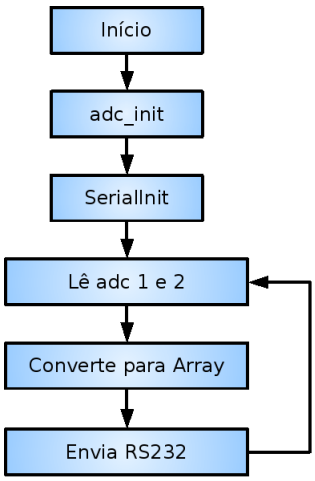


Figura 5.16: Diagrama de Blocos do Microcontrolador Sensor

No ciclo de leitura são realizadas leituras seguidas e é enviada a informação via RS232 não sendo introduzido qualquer tempo de espera o que se traduz na prática por uma frequência de amostragem de 17 KHz. Esta frequência é depois reduzida no computador "Controlador".

5.3.1.2 Actuador

A função do microcontrolador "Actuador" é essencialmente receber a informação proveniente da porta série e gerar o sinal PWM para o servomecanismo.

A forma como o sinal foi gerado é semelhante ao que foi implementado nos trabalhos anteriores [Roq07].

Para que seja possível a sincronização da informação nas mensagens foi implementada uma máquina de estados no atendimento da interrupção proveniente da porta série. Esta máquina de estados permite validar a recepção das mensagens de tipo "MA".

Na figura 5.17 é mostrado o diagrama de blocos do código implementado no microcontrolador "Actuador".

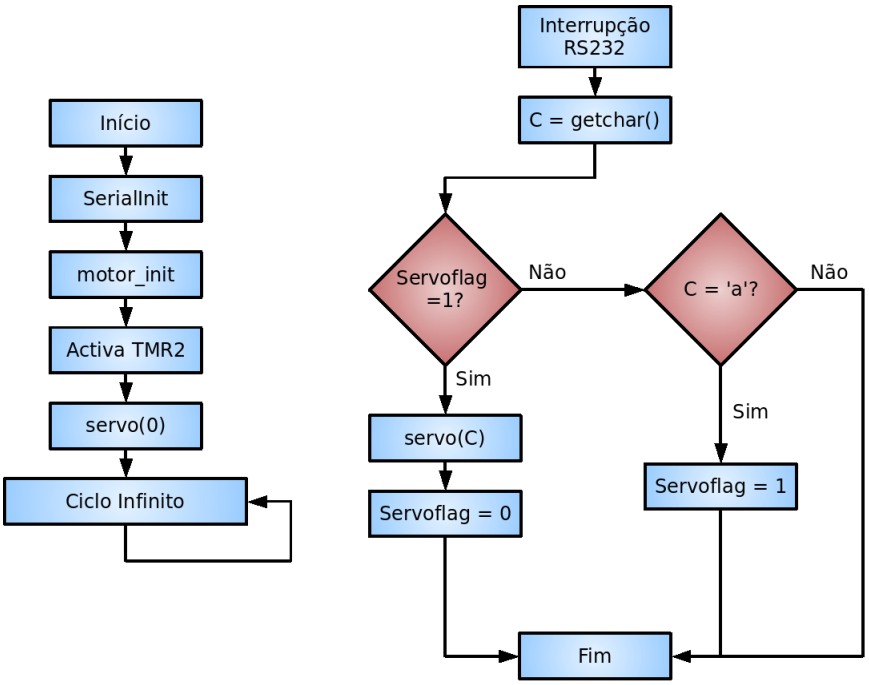


Figura 5.17: Diagrama de Blocos do Microcontrolador Actuador

5.4 Implementação em Raw Ethernet

5.4.1 Estrutura do Software

Esta parte de software corresponde à implementada nos computadores ligados pela rede Ethernet.

Tal como para o código dos microcontroladores o código gerado para estes componentes obedeceu a uma arquitectura modular.

O software realizado para estes componentes encontra-se dividido nos seguintes módulos:

- *serie* - contém as funções de configuração e operação da porta série em linux possibilitando a comunicação com os microcontroladores;
- *eth* - contém as funções de configuração e operação da rede possibilitando a troca de informação entre computadores via ethernet;
- *sensor* - possui o código principal que é executado no computador "sensor";
- *controlador* - possui o código principal que é executado no computador "controlador";
- *actuador* - possui o código principal que é executado no computador "actuador".

O módulo serie possui o programa de teste que possibilita a detecção de problemas de comunicação entre o computador e o microcontrolador.

Cada módulo possui uma *Makefile* que permite gerar automaticamente os executáveis. Para isso basta, na pasta do projecto, executar os seguintes comandos:

1. make clean
2. make all
3. make install

Nas secções seguintes será feita uma análise mais pormenorizada do código implementado em cada um dos módulos fundamentais da rede, Sensor, Controlador e Actuador. Os módulos "serie" e "eth" serão detalhados com mais pormenor no capítulo "Bola no Plano".

5.4.1.1 Sensor

A função principal do sensor é converter a mensagem recebida via porta série para a mensagem a enviar via Ethernet, funciona essencialmente como uma *gateway*.

Devido ao facto de se ter usado um protocolo para receber as mensagens via porta série foi necessário implementar neste nó uma máquina de estados para a recepção destas mensagens.

No diagrama de blocos da figura 5.18 está representado o programa principal que este nó executa.

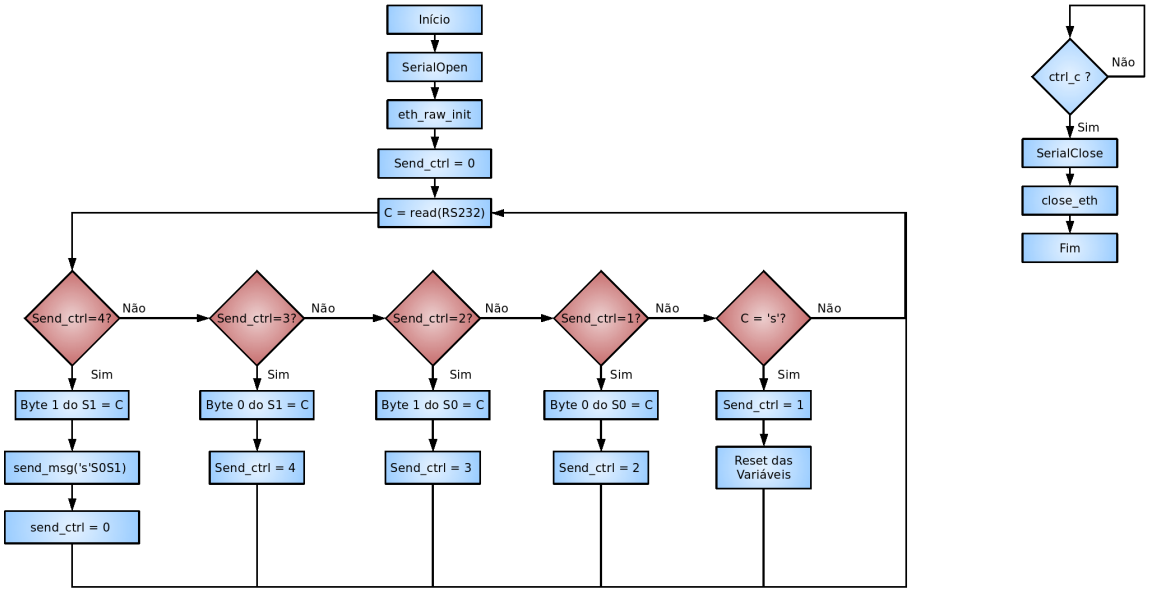


Figura 5.18: Diagrama de Blocos do Sensor

5.4.1.2 Controlador

O controlador tem a função de receber a informação enviada pelo sensor, filtrá-la, aplicar o algoritmo de controlo e enviar essa informação para o nó actuador.

Para reduzir a frequência de actuação foi implementado um mecanismo que aguarda 50 amostras antes de executar o controlador PID.

No diagrama de blocos da figura 5.19 está representado o programa principal que este nó executa.

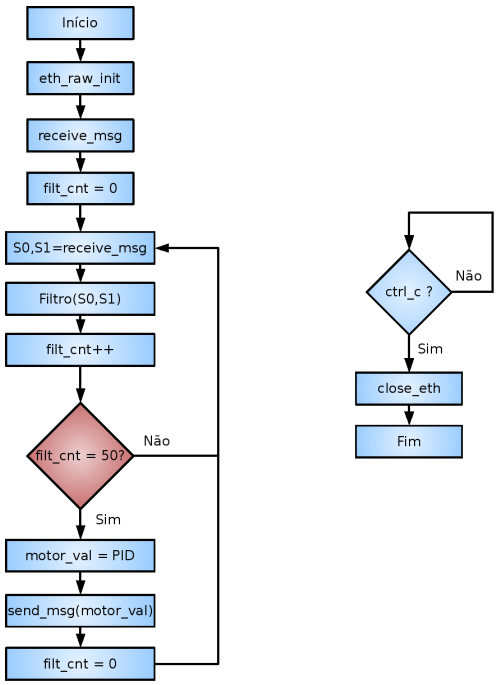


Figura 5.19: Diagrama de Blocos do Controlador

5.4.1.3 Actuador

A função deste componente é apenas converter a mensagem recebida do "controlador" via Ethernet nas mensagens a enviar via porta-série.

Na figura 5.20 é mostrado o diagrama de blocos do código implementado no "Actuador".

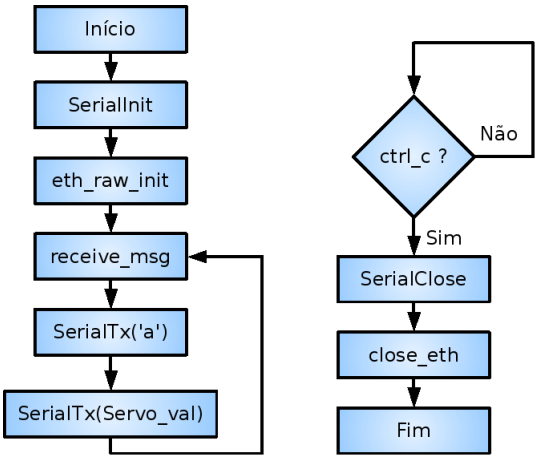


Figura 5.20: Diagrama de Blocos do Actuador

Capítulo 6

Plataforma Bola no Plano

Neste capítulo são detalhados os procedimentos inerentes à implementação do Sistema de Controlo da plataforma "Bola no Plano". Nas secções iniciais será descrito o Hardware que a plataforma possui bem como o método de montagem desta. De seguida detalham-se os diferentes módulos de Software. Estes módulos são apresentados começando por detalhar os módulos inerentes ao sistema de controlo da plataforma, detalhando depois os pormenores de implementação do controlo em Raw-Ethernet e em FTT-SE.

6.1 Arquitectura do Sistema

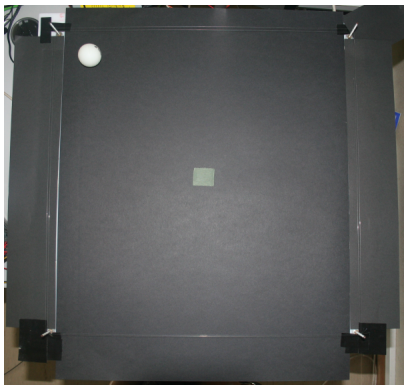


Figura 6.1: Plataforma Bola no Plano

O objectivo principal desta plataforma é fazer com que a bola que se encontra em cima do suporte se mantenha num ponto pré-determinado ao qual se dá o nome de *setpoint*. O único controlo possível é a alteração dos ângulos deste suporte. Essa alteração é feita através da actuação nos servomecanismos com base na informação adquirida pela *webcam*.

Com a distribuição do sistema ficou dedicado a cada um dos elementos, o sensor, o controlador e o actuador, um computador. A comunicação entre estes componentes do sistema é garantida através de uma rede de campo, que neste caso é a rede Ethernet com e sem protocolo FTT-SE.

Na figura 6.2 é mostrado um esquema global do sistema com todos os elementos que o compõem bem como a forma como estes comunicam entre si.

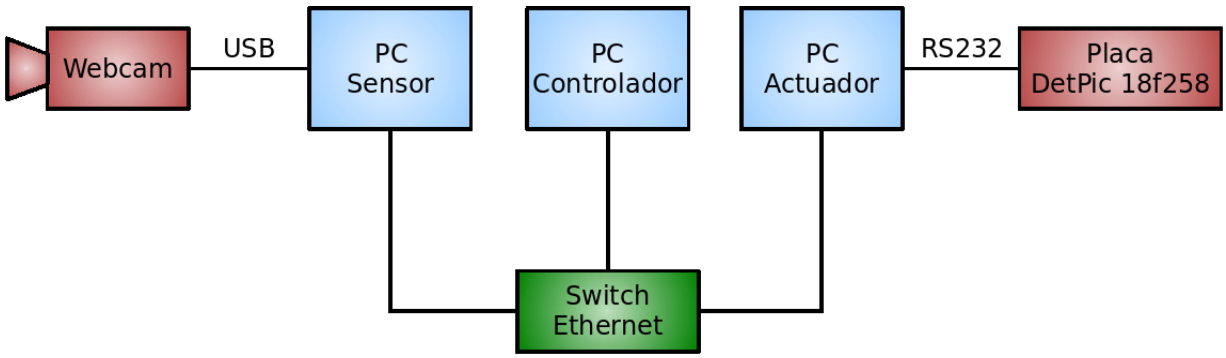


Figura 6.2: Arquitectura do Sistema Bola no Plano

A imagem é adquirida primeiramente pela Webcam e enviada, por USB, para o computador "Sensor". Neste a imagem é convertida para RGB24 e posteriormente para GrayScale. Depois de obtida a imagem esta é enviada, via Ethernet, para o computador "Controlador". Neste computador é usada a informação da posição da bola para calcular os valores a aplicar nos servomecanismos. Esse cálculo é assegurado pelos dois compensadores PID, um para cada dimensão (X e Y). Depois de calculados os valores a aplicar nos servomecanismos estes são enviados para o computador Actuador. No Actuador a informação é encaminhada, via RS232, para a placa DetPic 18F258 onde são gerados os sinais de PWM que irão ser aplicados nos servomecanismos.

Nas próximas secções será feita uma análise da implementação de cada um dos componentes bem como descrito o formato das mensagens que são trocadas.

6.2 Descrição do Hardware

6.2.1 Descrição dos Componentes

A plataforma "Bola no Plano" é constituída pelos seguintes componentes:

- *Plano da bola* - suporte da bola;
- *Webcam* - possibilita a determinação da posição da bola no plano;
- *2 Servomecanismos* - permitem alterar os ângulos do plano;
- *Módulo de Potência* - placa com o *driver* dos servomecanismos;
- *Placa DetPic 18F258* - placa com o microcontrolador 18F258 da Microchip©;
- *3 Computadores* - Sensor, Controlador e Actuador respectivamente.

6.2.2 Montagem da Plataforma

Para efectuar a correcta montagem dos diversos componentes da plataforma devem-se seguir os seguintes passos:

1. *Ligar a Webcam ao computador "Sensor" através do cabo USB;*
2. *Ligar o Módulo de Potência à placa DetPic 18F258 através do cabo próprio para o efeito;*

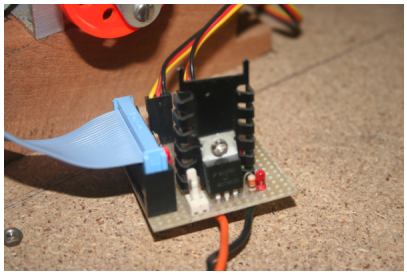


Figura 6.3: Ligação do Módulo de Potência à Placa DetPic18F258

3. *Ligar, através do cabo série, a placa DetPic 18F258 ao computador "Actuador";*
4. *Ligar os computadores em rede através do switch.*

A placa DetPic 18F258 é alimentada a partir do módulo de potência. Este módulo possui duas ligações para as respectivas fontes de alimentação. Deve-se ter em atenção que as fontes de alimentação deverão ter as massas ligadas entre si.

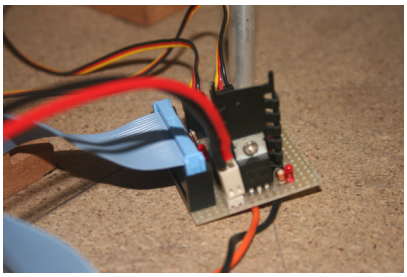


Figura 6.4: Ligação das Alimentações da Plataforma Bola no Plano

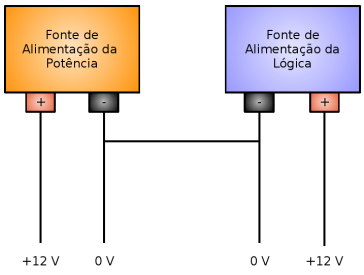


Figura 6.5: Ligação das Massas das Fontes de Alimentação

6.3 Descrição Geral do Software

Tal como na plataforma "Bola na Calha" o software foi implementado de forma modular. Nesta plataforma existem dois módulos essenciais, o Módulo Cam e o Serie, que permitem lidar com a *Webcam* e com a comunicação RS232. Estes módulos serão detalhados nas duas secções seguintes.

Por fim será descrito a forma como a imagem foi processada. O processamento da imagem sofreu bastantes alterações no decorrer desta dissertação pelo que será dado uma maior atenção à forma como a imagem foi tratada.

6.3.1 Módulo Cam

Este módulo é responsável pela obtenção da imagem proveniente da *Webcam*. Contém as funções de configuração e operação da câmara.

Ao longo do trabalho este módulo foi sofrendo alterações. Primeiramente começou por se usar a biblioteca "V4L2", biblioteca que cria uma abstracção dos drivers da câmara fornecendo um conjunto de funções de configuração e operação desta. A grande vantagem do uso desta biblioteca era a possibilidade de, com o mesmo código, poder suportar outras *webcams*, ou seja, era possível substituir a *webcam* por outro modelo sem que fosse necessário alterar o código, desde que a *webcam* utilizada fosse suportada pela biblioteca.

No entanto quando se implementou o protocolo FTT-SE nos diferentes computadores disponíveis, verificou-se que a versão do *Kernel* usada, a 2.6.9-RT, não trazia suporte para esta biblioteca. Optou-se então a por abandonar esta biblioteca e trabalhar directamente sobre o *driver* da *webcam*, o PWC.

Esta conversão foi feita de modo a que se mantivessem os nomes das funções de forma a que não fosse preciso alterar o código do "Sensor", o facto de usar módulos na implementação agilizou este processo de alteração mostrando desta forma a grande vantagem do uso deste tipo de solução.

Neste módulo está também embebida a biblioteca SDL que permite a visualização da imagem no ecrã. Mais uma vez quando se passou para a implementação do FTT-SE não foi possível conciliar esta biblioteca com a versão do *Kernel* e do compilador, em grande parte pelo facto de apenas um dos computadores usados possuir ambiente gráfico e haver a necessidade de visualizar a imagem não só no "Sensor" como também no "Controlador" visto ter passado este a ser o responsável pelo processamento da mesma. Para resolver este problema foi criada uma *flag* de compilação que permite compilar o módulo com, ou sem, esta biblioteca.

Este módulo contém as seguintes funções de operação e configuração da câmara:

- *open_device* - configura e estabelece comunicação com o device da câmara;
- *init_device* - configura os parâmetros da câmara como por exemplo o período de amostragem;
- *read_frame* - lê uma frame, a leitura é bloqueante;
- *yuv420_to_rgb24* - transforma a imagem de yuv para rgb;
- *close_device* - liberta o device da câmara.

Possui também as seguintes funções de configuração e visualização da imagem usando a biblioteca SDL:

- *init_sdl* - inicia a biblioteca SDL;
- *display_image* - actualiza a imagem no ecrã.

Na figura 6.6 mostra-se a forma como estas funções devem ser chamadas pelo programa principal incluindo a visualização da imagem.

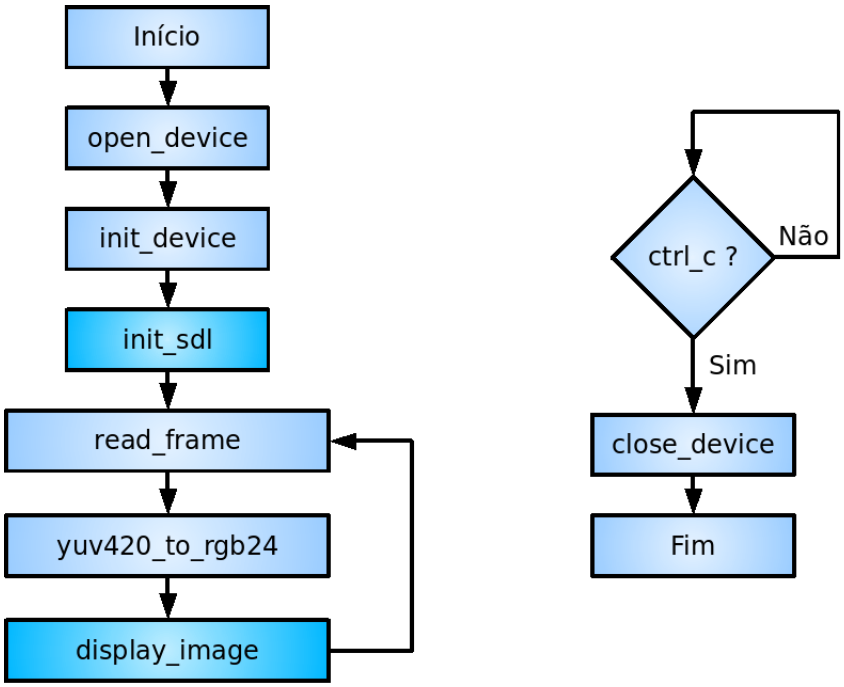


Figura 6.6: Módulo CAM - Exemplo de Utilização

No caso de se pretender processar a imagem, dever-se-á criar uma função local ao programa e deverá ser introduzida entre a função "yuv420_to_rgb24" e a função "display_image".

6.3.2 Módulo Serie

Este módulo possui as funções de configuração e operação do protocolo de comunicação RS232 para o sistema operativo *Linux*.

Este módulo possui as seguintes funções:

- *SerialOpen* - estabelece a comunicação com respectivo *device* RS232;
- *SerialClose* - liberta o *device*;
- *SerialTx* - envia informação;
- *SerialRx* - recebe informação de forma bloqueante.

Na figura 6.7 mostra-se a forma como estas funções devem ser chamadas no programa principal.

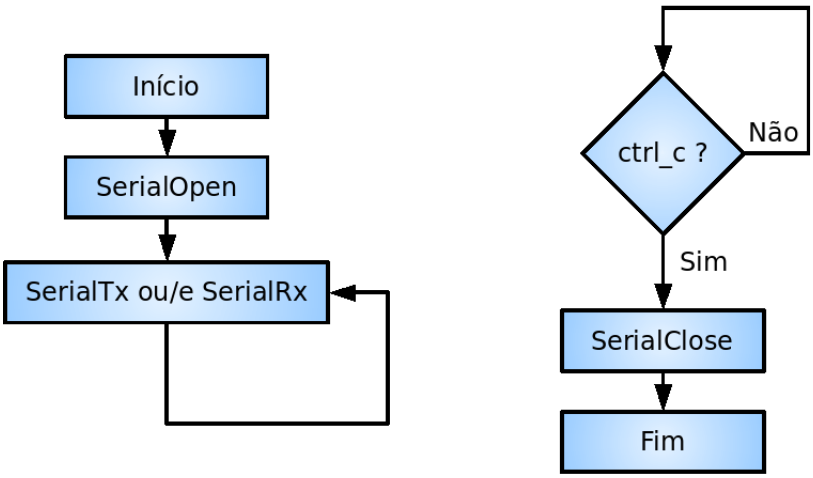


Figura 6.7: Módulo Serie - Exemplo de Utilização

6.3.3 Processamento da Imagem

A imagem recebida da câmara, depois de convertida para RGB obedece ao formato mostrado na figura 6.8

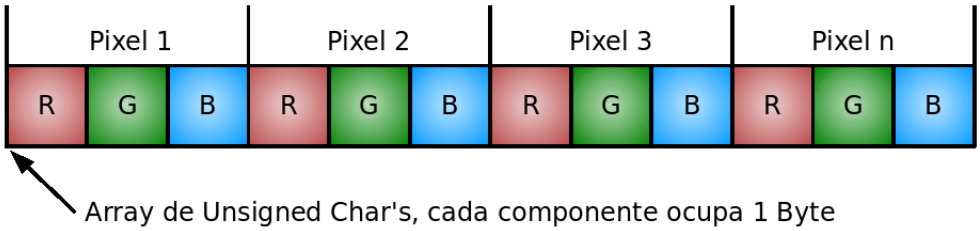


Figura 6.8: Formato da Imagem RGB

Quando se começou a realizar testes de carga da rede Ethernet verificou-se que a transferência de imagem pela rede ocupava demasiada largura de banda pelo que se teve que reduzir o tamanho da imagem. A solução encontrada foi alterar o formato da imagem para GrayScale. Optou-se por alterar o menos possível o que já tinha sido feito pelo que a obtenção da imagem continuou a ser feita a cores sendo apenas criado um *array* novo que contém a imagem a cores. Para evitar alterar o módulo "Cam" optou-se por manter o tamanho deste novo *array* igual aos *arrays* para imagens a cores sendo que cada *pixel* continuaria a ocupar três *Bytes* de informação só que neste caso esses três *Bytes* contêm o mesmo valor. Esta solução permite também usar a função "display_image" já com a imagem em GrayScale. A fórmula usada para passar a imagem em RGB para GreyScale [tG] foi a seguinte:

$$GS = R \times 0.3 + G \times 0.59 + B \times 0.11 \tag{6.1}$$

Na figura 6.9 é ilustrada a conversão utilizada e o formato com que a imagem fica no final dessa conversão

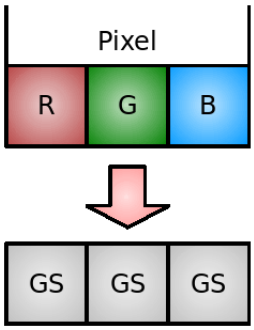


Figura 6.9: Conversão de RGB para GrayScale

Com esta alteração consegue-se reduzir o tamanho da imagem para um terço de uma imagem a cores.

Para enviar a imagem para o controlador apenas é seleccionado o primeiro *Byte* de cada *pixel* sendo depois reconstruída a imagem. Com este mecanismo conseguiu-se reduzir a ocupação do barramento de cerca de 60% para cerca de 20%.

De uma forma geral o algoritmo de processamento executa as seguintes tarefas

- Executa uma pesquisa completa da imagem processando pixel a pixel;
- Pesquisa o centro, dentro de um rectângulo predefinido marcando esse rectângulo a azul. Este mecanismo foi implemetado para evitar falsas validações do centro;
- Troca a cor dos pontos que identifica como centro para Azul e conta esses pontos;
- Pesquisa a bola dentro de um rectângulo com dimensões também predefinidas marcando esse rectângulo a verde, evitando assim detecções de falsos pontos da bola;
- Troca a cor dos pontos que identifica como bola para vermelho e conta esses pontos;
- Calcula a posição do centro do Plano;
- Calcula a posição do centro de massa da Bola.

O processamento realizado à imagem em GrayScale trouxe maiores dificuldades na identificação dos objectos. Foi necessário implementar alguns mecanismos adicionais para evitar falsas detecções. Os mecanismos mais importantes que foram implementados são listados de seguida:

- *Criação de zonas de pesquisa específicas*

  A detecção da bola foi restrita a um rectângulo marcado a verde na figura, a detecção dos pontos da bola só é feita dentro deste rectângulo. Para a detecção do centro do plano foi adoptado uma solução semelhante, os pontos do centro só são pesquisados dentro de um rectângulo que é marcado com cor azul. O centro destes dois rectângulos é a posição do centro do plano sendo esta actualizada em cada iteração.

- *Não pesquisa do centro quando a bola se encontra dentro do rectângulo central*

  Este mecanismo foi implementado para corrigir um problema que surgiu da passagem da imagem para GrayScale. Frequentemente era detectado o centro na sombra da bola pelo que a posição de equilíbrio não poderia ser atingida visto variar com a posição da bola. A solução encontrada foi não actualizar a posição do centro do plano se a bola se encontrasse dentro do rectângulo do centro. Este mecanismo permitiu manter uma posição fixa do ponto central quando a bola se aproxima do ponto central.

As figuras 6.10 e 6.11 mostram uma imagem sem processamento e outra com o resultado do processamento feito à imagem.

A figura 6.12 mostra o diagrama de blocos simplificado do processamento de imagem.

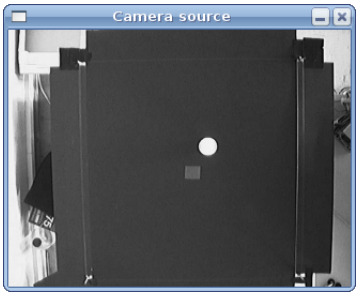


Figura 6.10: Imagem sem processamento

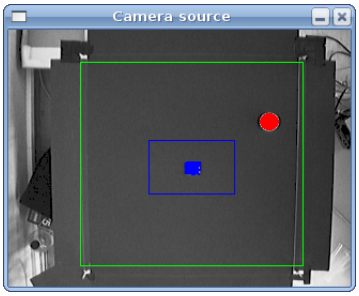


Figura 6.11: Imagem com processamento

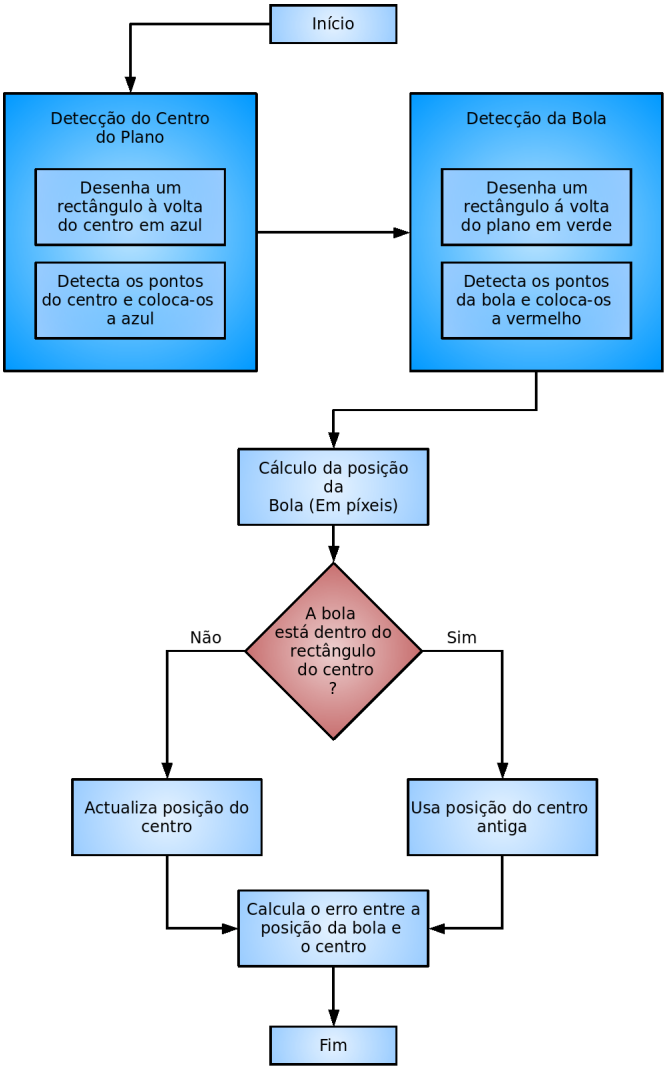


Figura 6.12: Diagrama de Blocos Simplificado do Processamento de Imagem

6.4 Implementação em Raw Ethernet

6.4.1 Formato das Mensagens

Para que os vários componentes dos sistema comuniquem eficientemente foi necessário estabelecer um formato para as mensagens que são trocadas.

Na rede Ethernet existem 3 tipos de mensagens essenciais:

- *Mensagem Tipo "s"* - Mensagem trocada entre o computador "Sensor" e o computador "Controlador"

  Esta mensagem é gerada quando o processamento de imagem é feito no "Sensor". Optou-se por manter esta possibilidade apenas para garantir uma compatibilidade com trabalho já feito pois para a realização dos testes é enviada a imagem por completo para o "Controlador" onde é processada. A informação que esta mensagem transporta é essencialmente a informação da posição da bola em relação ao centro do plano.

  A figura 6.13 mostra o formato da mensagem bem como o tamanho que esta ocupa. Estes campos são enviados na secção de dados da trama Ethernet.

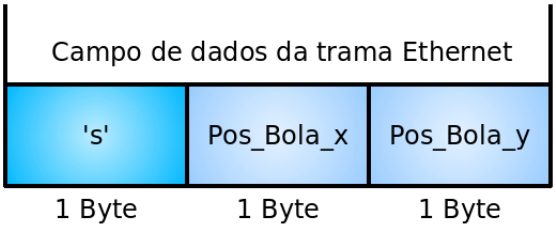


Figura 6.13: Mensagem do tipo "s"

  Os campos da mensagem "s" são :

  - Identificador - neste caso irá sempre com o caracter "s";
  - Posição x da bola;
  - Posição y da bola.

- *Mensagem Tipo "b"* - Mensagem trocada entre o computador "Sensor" e o computador "Controlador"

  Esta mensagem é gerada quando a imagem é enviada do computador "Sensor" para o computador "Controlador". Como a imagem necessita de ser fragmentada é adicionado um campo que permite identificar o número do fragmento. Esse número permite, no destino, colocar o fragmento no sítio correcto. Os restantes 1498 *Bytes* do campo de dados da trama Ethernet vão preenchidos com o respectivo segmento da imagem.

  A figura 6.14 mostra o formato da mensagem bem como o tamanho que esta ocupa. Estes campos são enviados na secção de dados da trama Ethernet.

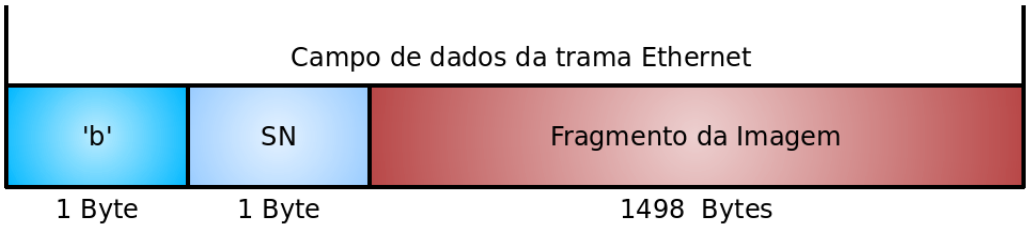


Figura 6.14: Mensagem do tipo “b”

Os campos da mensagem “b” são:

- Identificador - neste caso irá sempre com o caracter “b”;
- SN - Identifica o fragmento da imagem;
- Fragmento da Imagem - porção de imagem correspondente ao fragmento identificado no campo “SN”.

- *Mensagem Tipo “a”* - Mensagem trocada entre o computador “Controlador” e o computador “Actuador”

  Esta mensagem é enviada com o resultado da execução dos controladores PID. A informação enviada por esta mensagem é essencialmente os *setpoints* dos servomecanismos que irão actuar na plataforma.

  A figura 6.15 mostra o formato da mensagem bem como o tamanho que esta ocupa. Este campo refere-se ao conteúdo que é enviado na secção de dados da trama Ethernet.

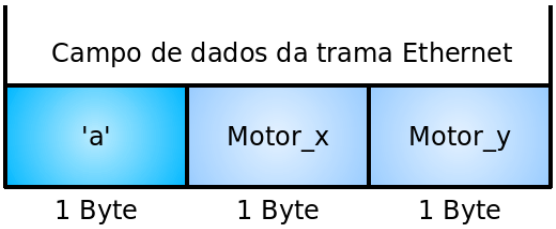


Figura 6.15: Mensagem do tipo “a”

Os campos da mensagem “a” são:

- Identificador - neste caso irá sempre com o caracter “a”;
- Valor do motor x;
- Valor do motor y.

Além das mensagem trocadas na rede Ethernet foi também preciso definir o formato das mensagens trocadas pelo protocolo RS232:

- *Mensagem do tipo “j”* - Mensagens trocadas entre o computador “Actuador” e a placa DetPic 18F258

Estas mensagens definem um protocolo de comunicação entre o computador "Actuador" e a placa DetPic 18F258. Primeiramente é enviado um identificador que permite identificar que a informação que se segue é relativa às posições dos motores. Foi necessário implementar este protocolo pois há necessidade de sincronizar a informação nos dois nós, caso apenas se enviassem os valores nunca se conseguiria identificar a que motor se referia a informação. No microcontrolador foi implementada uma máquina de estados que permite sincronizar a informação nos dois nós.

A figura 6.16 mostra o protocolo de comunicação entre os componentes.

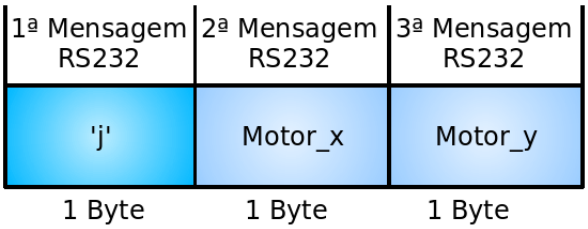


Figura 6.16: Mensagem do tipo "j"

A figura 6.17 mostra o diagrama de blocos da máquina de estados implementada no microcontrolador para a recepção deste tipo de mensagens.

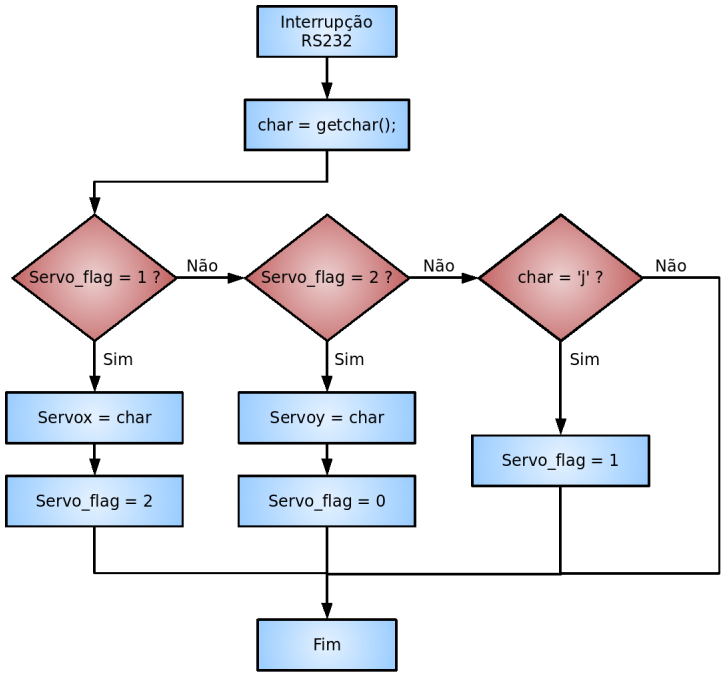


Figura 6.17: Diagrama de blocos do protocolo implementado no microcontrolador

6.4.2 Estrutura do Software

6.4.2.1 Módulo Eth

À semelhança dos módulos "Cam" e "Serie" foi desenvolvido um módulo para lidar com a transferência de informação via Ethernet no *Linux*. Este módulo implementa funções de configuração e operação da camada de rede do sistema operativo criando uma abstracção que permite simplicar o código do programa principal e fornecer uma forma eficaz de detectar e corrigir problemas.

Este módulo define também uma estrutura de dados que permite unificar o modo como se interage com as funções. Esta estrutura tem a seguinte forma

- *src_mac* - Endereço MAC da estação de envio;
- *dst_mac* - Endereço MAC da estação de destino;
- *data[1500]* - Campo de dados;
- *data_size* - tamanho, em *bytes*, da informação que é enviada no campo de dados.

Neste módulo estão também implementadas as seguintes funções:

- *eth_raw_init;*
- *send_msg;*
- *receive_msg;*
- *close_eth.*

A figura 6.18 mostra a forma como se deve utilizar as funções deste módulo.

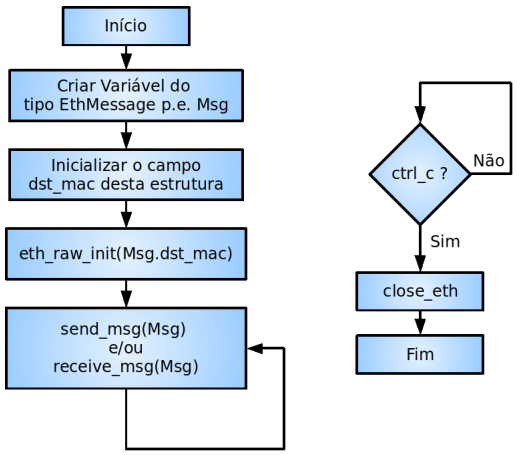


Figura 6.18: Módulo Eth - Exemplo de Utilização

6.4.2.2 Sensor

Tendo em conta tudo o que já foi referido sobre o papel do computador "Sensor" vai-se nesta secção analisar em pormenor a implementação deste componente. Em suma as tarefas realizadas são:

- *Ler a informação proveniente da câmara;*
- *Caso esteja definido, processar a imagem;*
- *Enviar a informação da Bola através de mensagens do tipo "s" ou enviar a imagem através de mensagens do tipo "b".*

No caso deste nó realizar o processamento da imagem o código segue o diagrama de fluxo representado na figura 6.19.

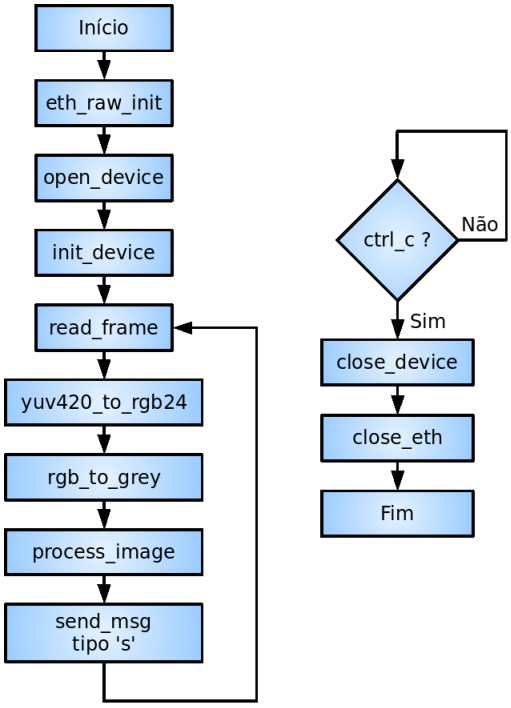


Figura 6.19: Diagrama de Blocos do Sensor com processamento de imagem

Quando este nó não executa o processamento da imagem esta é enviada por inteiro para o computador "Controlador". A imagem, depois de passada para GrayScale, ocupa 52 mensagens Ethernet, visto

- A imagem ocupa 240 píxeis de altura e 320 de largura o que perfaz um total de

$$240 \times 320 = 76800 \text{ píxeis} \tag{6.2}$$

- Depois da passagem para GrayScale cada pixel ocupa 1 *Byte* logo perfaz um total de 76800 *Bytes* por imagem.

- Numa imagem Ethernet consegue-se enviar 1498 *Bytes* de informação, logo para enviar a imagem completa precisa-se de

$$\frac{76800}{1498} \approx 51,258 \text{ mensagens} \tag{6.3}$$

  Ou seja 51 mensagens com 1498 *Bytes* e 1 com 402 *Bytes*.

A figura 6.20 mostra o diagrama de blocos do código do "Sensor" quando este envia a imagem completa para o "Controlador".

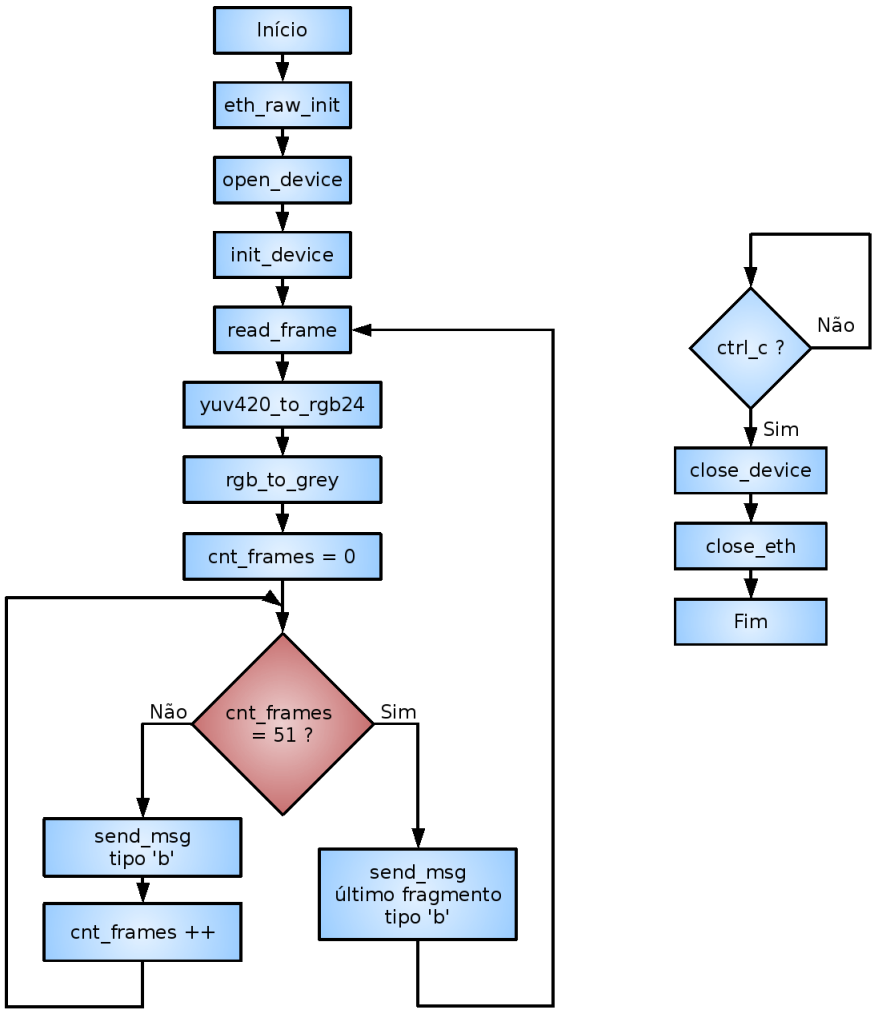


Figura 6.20: Diagrama de Blocos do Sensor sem processamento de imagem

6.4.2.3 Controlador

O computador "Controlador" é responsável pela recepção da informação proveniente do "Sensor", pela execução dos compensadores PID e pelo envio da mensagem para o computador "Actuador". No caso da imagem ser enviada pela rede este nó terá também que executar o processamento da mesma. Este nó recebe os tipos de mensagens "s" e "b" pelo que tem que adoptar diferentes estratégias consoante o tipo de mensagem que recebe. No caso de receber a mensagem de tipo "b" terá que, adicionalmente, reconstruir a imagem.

Na figura 6.21 é mostrado o diagrama de blocos do código implementado no "Controlador".

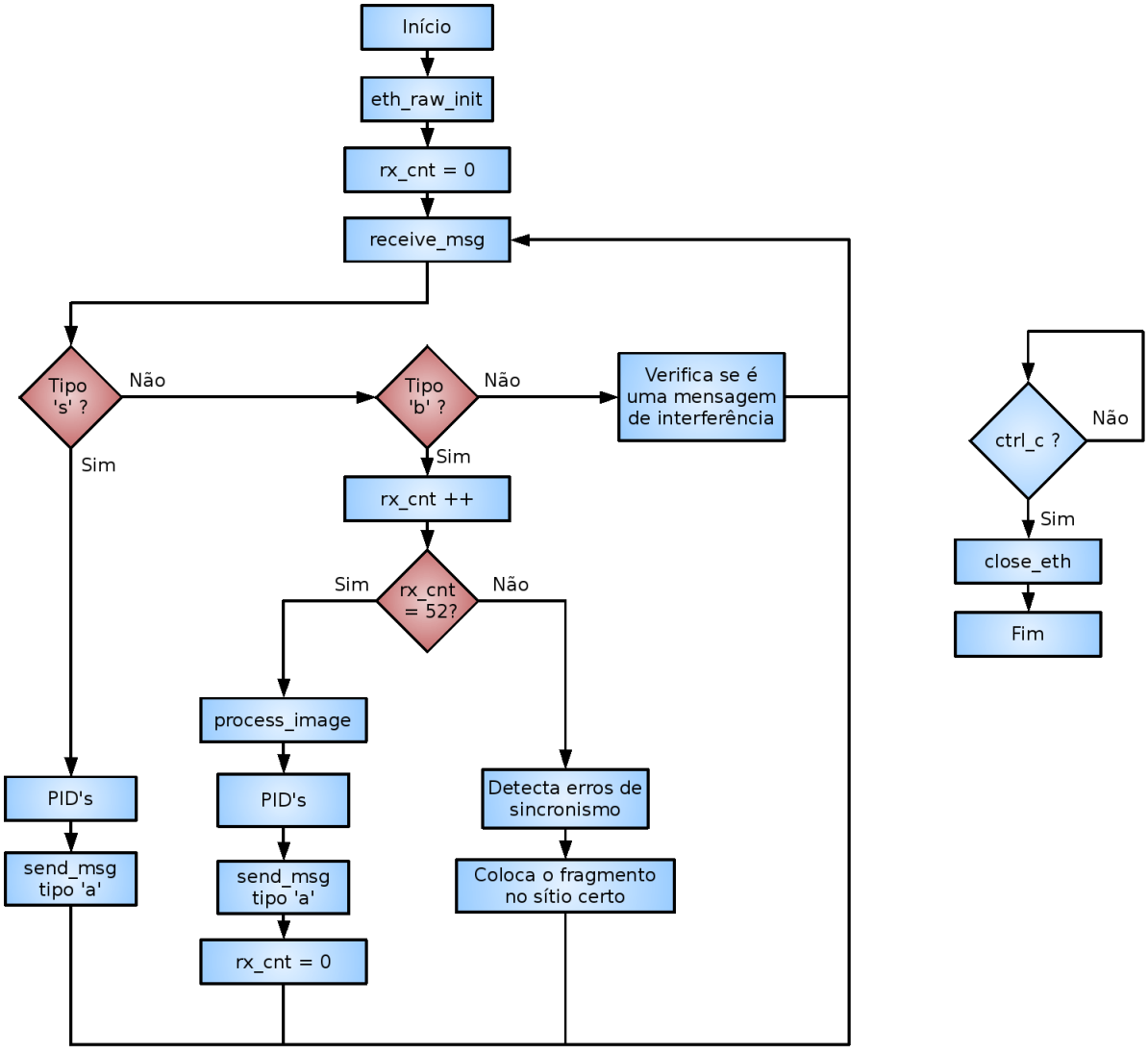


Figura 6.21: Diagrama de Blocos do Controlador

6.4.2.4 Actuador

A função principal deste componente é traduzir as mensagens provenientes do controlador, via Ethernet, e encaminhar a informação para a porta série.

Na figura 6.22 é mostrado o diagrama de blocos do código implementado no "Actuador".

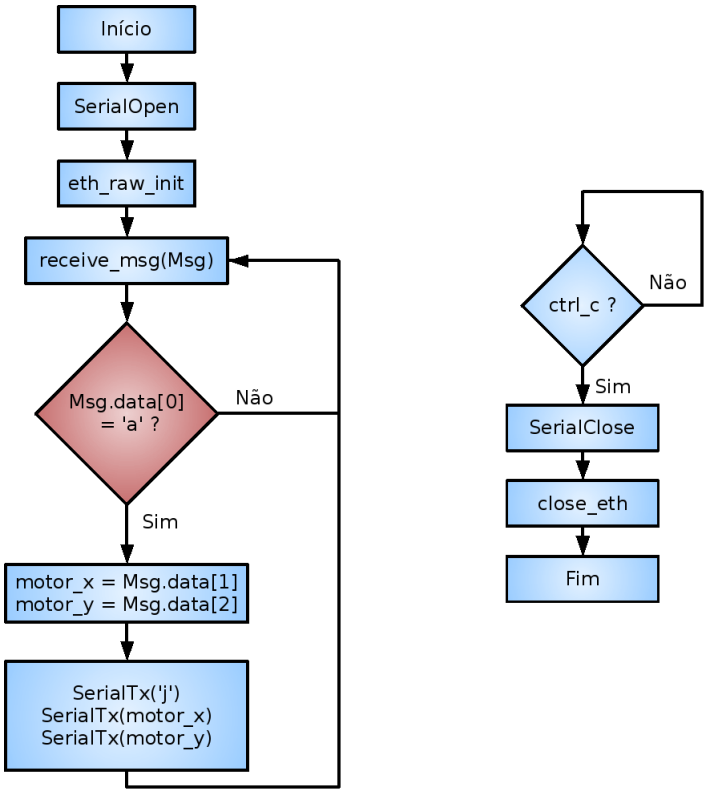


Figura 6.22: Diagrama de Blocos do Actuador

6.4.2.5 Cam-Int

Este programa foi criado para gerar tráfego de interferência. Permite gerar tráfego com características iguais ao do "Sensor", transferência de imagens, permitindo simular até 10 câmaras com frequência até 100 fps. Este programa permite também gerar tráfego de tamanho mais reduzido simulando tráfego de configuração. Este último tráfego caracteriza-se por realizar apenas a transferência de uma mensagem com 1000 *Bytes* de informação com frequência até 100 mensagens por segundo.

Por exemplo, para gerar tráfego de uma câmara idêntica à do sensor com uma frequência de 50 fps basta executar o seguinte comando:

- *cam_int -c 3 -f 50*

A opção "-c 3" determina que este nó vai gerar tráfego idêntico à câmara do sensor com a frequência de "-f 50" 50 fps.

Para gerar tráfego de controlo com 60 mensagens por segundo executa-se o seguinte comando

- *cam_int -s 1 -f 60*

Cada tipo de mensagem é identificado com um identificador diferente permitindo assim que o controlador distinga os diferentes tráfegos na rede.

6.5 Implementação em FTT-SE

Da implementação em FTT-SE surge logo uma alteração à arquitectura do sistema que é a introdução de um nó suplementar na rede, o Master. Outra alteração importante foi a necessidade de analisar, em termos do comportamento temporal, os tipos de tráfego transaccionado entre os diferentes nós. Numa primeira análise verificou-se que todo o tráfego transaccionado possuía um período bem definido tendo por isso propriedades de tráfego síncrono. Numa primeira implementação do protocolo à plataforma usou-se esta abordagem o que mais tarde veio a trazer problemas devido a não se ter conseguido sincronizar a imagem com o protocolo, ou seja, no máximo o atraso seria de 66 ms, duas vezes o período da mensagem com a agravante deste atraso ser variável o que para o algoritmo de controlo é difícil de corrigir. Com esta primeira opção não se conseguiu controlar a plataforma levando a que se optasse por outra solução.

Para solucionar este problema surgiram duas opções, uma delas era implementar um mecanismo de sincronismo que permitisse sincronizar a rede pela câmara, a outra solução era colocar o tráfego da câmara na janela assíncrona. Optou-se pela segunda opção por ser mais simples de implementar e por conseguir resolver o problema de sincronismo da rede. Optou-se também por manter o tráfego do actuador na janela síncrona.

As propriedades do tráfego transaccionado são as seguintes:

- *Tráfego da câmara* - tráfego assíncrono;
- *Tráfego do actuador* - tráfego síncrono com período de 1 ms.

A nova arquitectura do sistema é mostrada na figura 6.23.

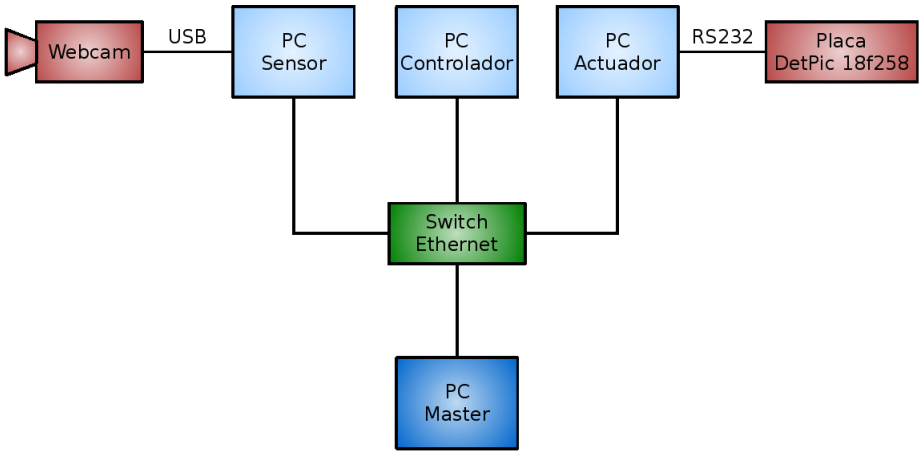


Figura 6.23: Arquitectura do Sistema com o Protocolo FTT-SE

Na implementação em FTT-SE teve que se ter em atenção que a interface com o protocolo foi toda criada em *Kernel-Space* pelo que teve de se adaptar toda a estrutura do código já implementado para esta nova realidade.

O protocolo FTT-SE está implementado em *RT-Linux*, uma versão do *Kernel* que confere ao Sistema Operativo *Linux* propriedades de Tempo - Real.

Nas secções seguintes é explicada a forma como se adaptou o código e como este está estruturado.

6.5.1 Arquitectura do Software

Para conseguir fazer o interface com o código já implementado foi necessário idealizar uma arquitectura que permitisse fazer a comunicação entre *User-Space* e *Kernel-Space*, isto porque era de todo impossível transferir todo o código produzido para *Kernel-Space* visto, por exemplo, as funções de interface com a câmara serem em *User-Space* e não se coneseguir, em tempo útil, implementar *drivers* para a câmara em *Kernel-Space*, o mesmo sucede com os *drivers* da porta-série. O facto de se usar *RT-Linux* dificultou também a tarefa de encontrar *drivers* para estes componentes.

Tendo em conta todas as limitações optou-se por implementar um mecanismo de comunicação baseado em FIFO's e Memória Partilhada. Estes mecanismos permitem a transferência, de um modo eficiente, de informação entre o *User-Space* e o *Kernel-Space*. No caso do "Sensor" foi usado Memória Partilhada visto haver a necessidade de haver transferências de uma grande quantidade de informação. No "Actuador" foram usados apenas FIFO's para fazer a transferência de informação visto trata-se de pequenas quantidades de informação.

Para facilitar a implementação do código foram criadas bibliotecas de interface que permitem criar uma abstracção dos pormenores de comunicação entre *User-Space* e *Kernel-Space*.

Na figura 6.24 é mostrada a arquitectura do software implementado.

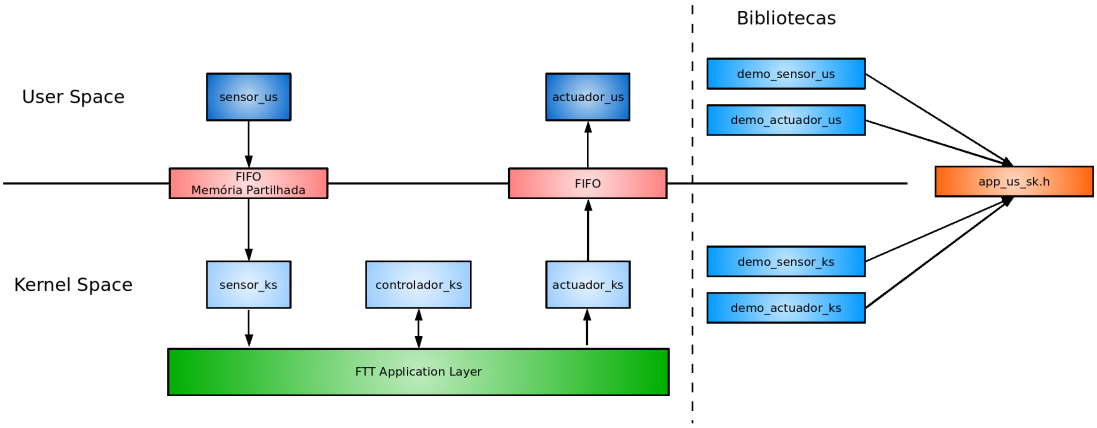


Figura 6.24: Arquitectura do Software implementado em FTT-SE

O único nó que não foi necessário implementar este tipo de mecanismos foi o controlador, isto deve-se ao facto de este não lidar com nenhum dispositivo externo.

6.5.2 Estrutura do Software

Tal como já foi referido na secção anterior, foram criadas bibliotecas que permitem criar uma abstracção da comunicação entre *Kernel-Space* e *User-Space*.

Do ponto de vista do *User-Space* todas as acções de interface com a rede Ethernet passaram a ser feitas com os FIFO's, ou Memória Partilhada, em vez do envio da informação directamente para a rede. Esta foi a principal alteração à estrutura do código já realizado.

Para resolver o problema do programa em *Kernel-Space* ficar bloqueado no acesso às FIFO's a implementação foi feita com recurso a *Threads*.

Nas próximas secções será explicado a forma como foram implementadas as bibliotecas

bem como é feito o interface entre as aplicações e os FIFO's / Memória Partilhada. Será também explicada a forma como as *Threads* foram usadas na implementação do sistema.

6.5.2.1 Bibliotecas

Foi necessário criar 4 bibliotecas de interface. As bibliotecas foram criadas com o intuito de fornecer funções que permitam aceder aos FIFO's e à Memória Partilhada nos dois espaços. Assim as aplicações que operem em *Kernel-Space* terão bibliotecas compiladas em *Kernel-Space* que fazem as inicializações necessárias à correcta operação dos FIFO's. Já as aplicações que operam em *User-Space* utilizam as funções que fazem a comunicação com os FIFO's e à Memória Partilhada a bibliotecas que estão compiladas em *User-Space*. Estas bibliotecas são:

- *demo_sensor_us* - Contém as funções que permitem inicializar e operar a informação dos FIFO's e da Memória Partilhada;
- *demo_sensor_ks* - Contém as funções de criação, configuração e operação dos FIFO's e da Memória Partilhada;
- *demo_actuador_us* - Contém as funções que permitem inicializar e operar a informação dos FIFO's;
- *demo_actuador_ks* - Contém as funções de criação, configuração e operação dos FIFO's de comunicação.

Para entender as funções presentes nestas bibliotecas é necessário fazer uma análise detalhada do problema. No caso do "Sensor" o fluxo de informação é proveniente do *User-Space* assim, além das funções de inicialização e fecho da comunicação com os FIFO's e a Memória Partilhada, é também fornecida uma função "write_fifo" que permite transferir para o *Kernel-Space* a imagem recebida da câmara. Já no *Kernel-Space* a biblioteca implementa uma função que vai buscar a informação da câmara, essa função é a "read_sensor".

Se uma análise semelhante for feita para o caso do "Actuador" verifica-se que o fluxo de informação faz-se do *Kernel-Space* para o *User-Space* pelo que a biblioteca de *Kernel-Space* possui uma função de escrita no FIFO, a "write_actuador", e a biblioteca de *User-Space* possui uma função de leitura do FIFO, a "read_fifo".

Na figura 6.25 é mostrada a forma como as bibliotecas do sensor interagem com as aplicações "sensor_ks" e "sensor_us" e como se processa o fluxo de informação.

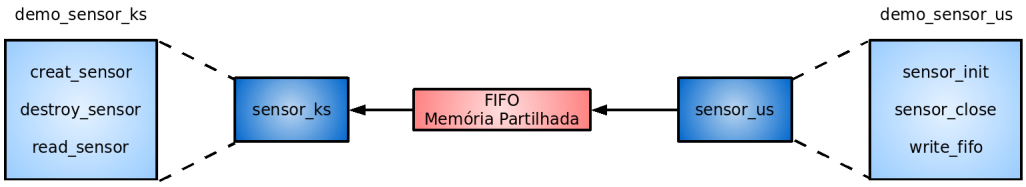


Figura 6.25: Bibliotecas demo_sensor_ks e demo_sensor_us

Na figura 6.26 é mostrada a forma como as bibliotecas do actuador interagem com as aplicações "actuador_ks" e "actuador_us" e como se processa o fluxo de informação.

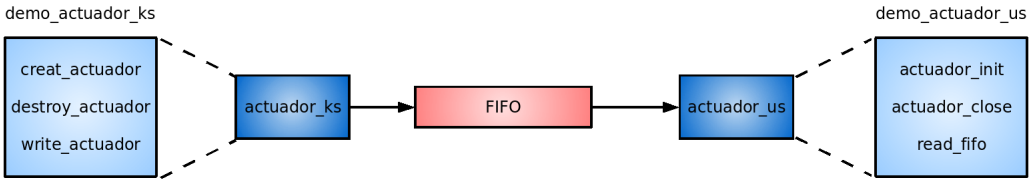


Figura 6.26: Bibliotecas demo_actuador_ks e demo_actuador_us

6.5.2.2 Sensor

Analisando o papel do nó "Sensor" no sistema verifica-se que a principal função deste é adquirir a imagem da câmara e enviá-la, depois de convertida em GrayScale, para o nó "Controlador".

A separação entre *Kernel-Space* e *User-Space* introduziu a comunicação baseada em Memória Partilhada e em FIFO's, neste caso o FIFO serve apenas para indicar ao *Kernel-Space* que se encontra uma imagem pronta para ser enviada.

O uso da *Thread* no *Kernel-Space* serve para que a aplicação não fique bloqueada no acesso ao FIFO. Adicionalmente acrescentou-se uma pausa no processamento de 100 ns.

A versão do RT-Linux usado implementa um mecanismo de inserção de módulos no *Kernel* automatizado. É gerado um ficheiro com extensão ".rtl" que carrega automaticamente os módulos necessários no *Kernel*. Este programa contém duas funções essenciais, a "my_app_init" e a "my_app_close", que são chamadas, respectivamente, no momento de inserção e remoção do módulo.

Nas figuras 6.27 e 6.28 é mostrado o diagrama de blocos do código implementado no *Kernel-Space* (sensor_ks) e *User-Space* (sensor_us) do nó "Sensor".

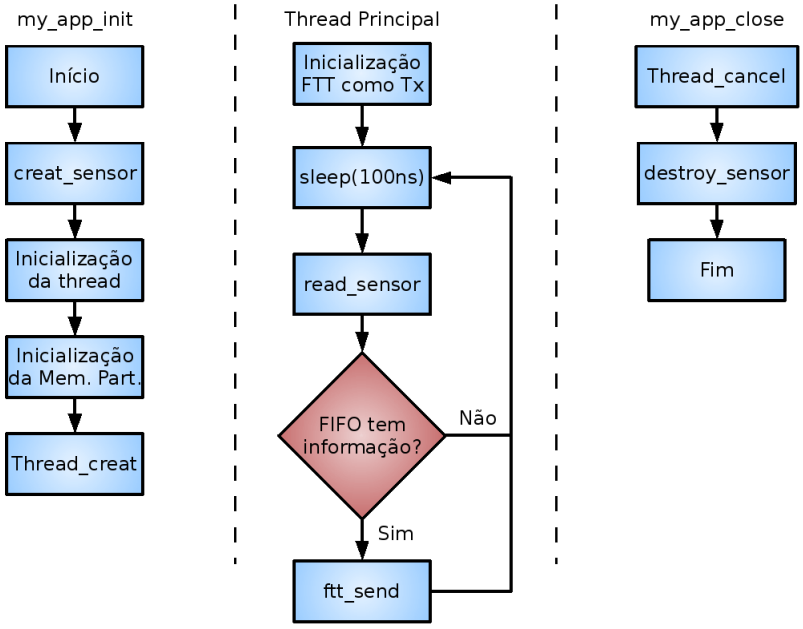


Figura 6.27: Diagrama de Blocos do Sensor em *Kernel-Space*

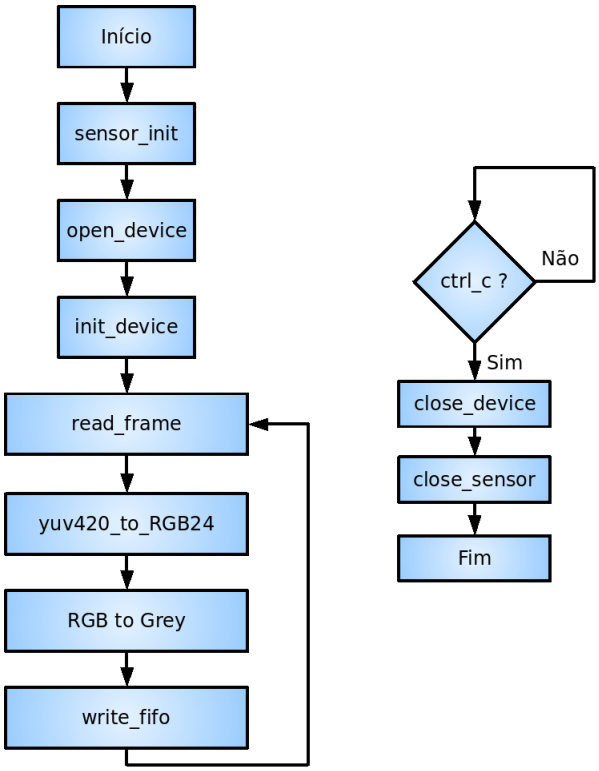


Figura 6.28: Diagrama de Blocos do Sensor em *User-Space*

6.5.2.3 Controlador

O código do nó "Controlador" foi totalmente desenvolvido para correr unicamente em *Kernel-Space*, isto porque é o único nó que não necessita aceder a nenhum dispositivo externo nem necessita de bibliotecas externas. A função deste nó é receber a informação proveniente do Sensor e dos nós de interferência, executar o algoritmo de processamento de imagem e de controlo e enviar os resultados para o nó "Actuador".

Neste nó foi necessário adoptar uma estratégia diferente, comparativamente aos restantes nós, no que diz respeito à quantidade de *Threads* lançadas. Para perceber o porquê de haver mais *Threads* lançadas é necessário analisar o problema com detalhe. Nos nós "Sensor" e "Actuador" é apenas lançada uma *Thread* principal de execução porque apenas é gerada um tipo de mensagens nesses nós, no nó "Sensor" é gerada uma mensagem e no nó "Actuador" há apenas a recepção de uma mensagem. No nó "Controlador" existem quatro tipos de mensagens que são transaccionadas, as mensagens recebidas do "Sensor", dois tipos de mensagens recebidas das câmaras de interferência e as mensagens enviadas para o "Actuador". No código implementado foram usados apenas 3 *Threads* isto porque corresponde à quantidade de mensagens de tipos diferentes que se recebe, e para a recepção das mensagens é necessário que o código bloqueie à espera dessa mensagem o que já não acontece com a transmissão, daí o facto de não haver necessidade de ter uma *Thread* para a mensagem do "Actuador". O que é importante reter é que a aplicação em *Kernel-Space* não pode bloquear e onde bloqueia é na recepção de mensagens, neste nó como há a recepção de 3 tipos de mensagens é necessário

existir 3 *Threads* de leitura. Foi necessário, no entanto, criar uma *Thread* adicional, esta *Thread* tem como função lançar as restantes. Esta nova *Thread* foi necessária devido ao facto de quando o módulo é inserido apenas poder ficar bloqueado uma única vez, caso este facto não se verificasse esta última *Thread* não seria necessária.

Na figura 6.29 é mostrado o diagrama de blocos do código implementado no *Kernel-Space* (controlador_ks) do nó "Controlador".

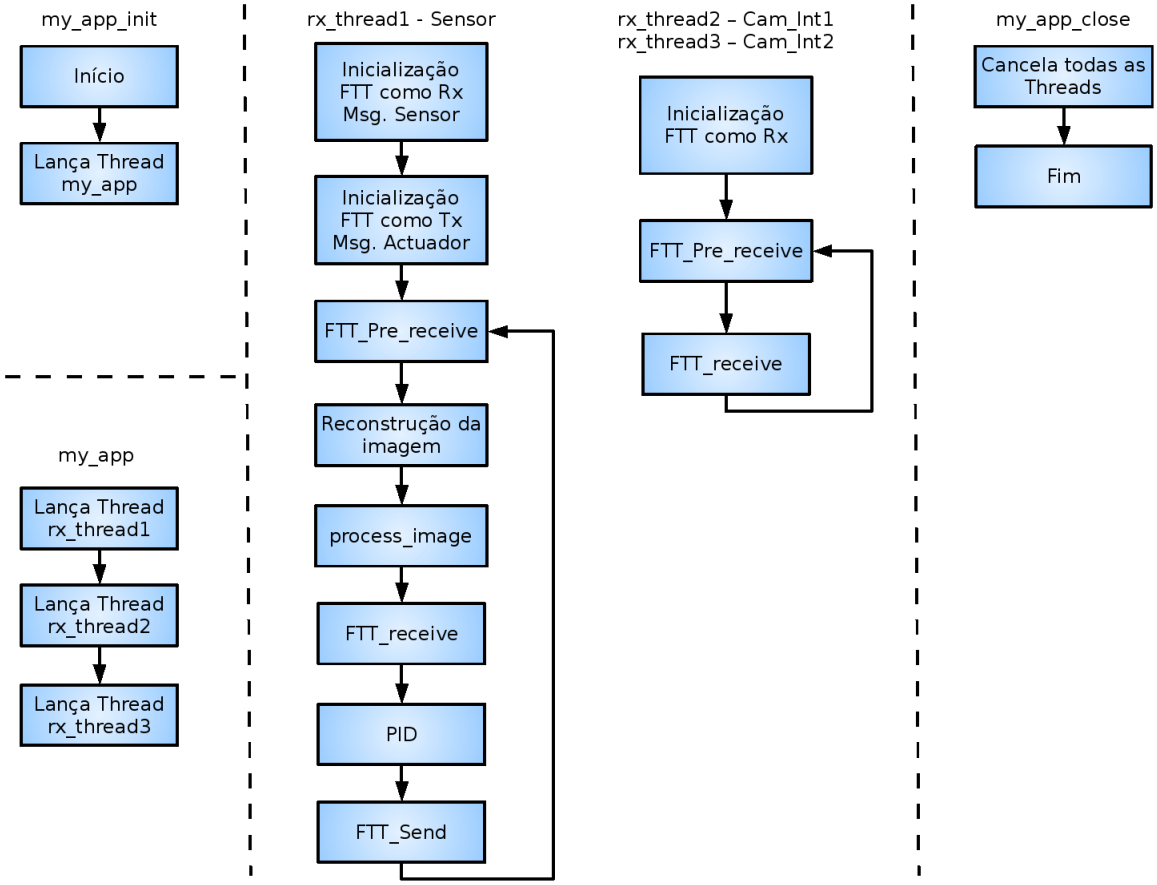


Figura 6.29: Diagrama de Blocos do Controlador em *Kernel-Space*

6.5.2.4 Actuador

O nó "Actuador" tem como principal função transferir para o microcontrolador a informação recebida do "Controlador".

O uso da *Thread* no *Kernel-Space* serve para que a aplicação não fique bloqueada no acesso ao FIFO.

Nas figuras 6.30 e 6.31 é mostrado o diagrama de blocos do código implementado no *Kernel-Space* (actuador_ks) e *User-Space* (actuador_us) do nó "Actuador".

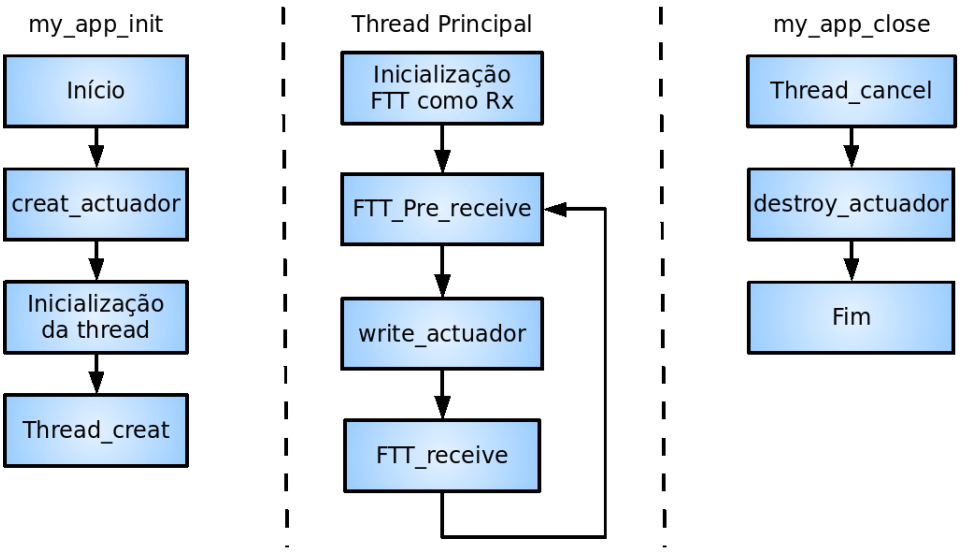


Figura 6.30: Diagrama de Blocos do Actuador em *Kernel-Space*

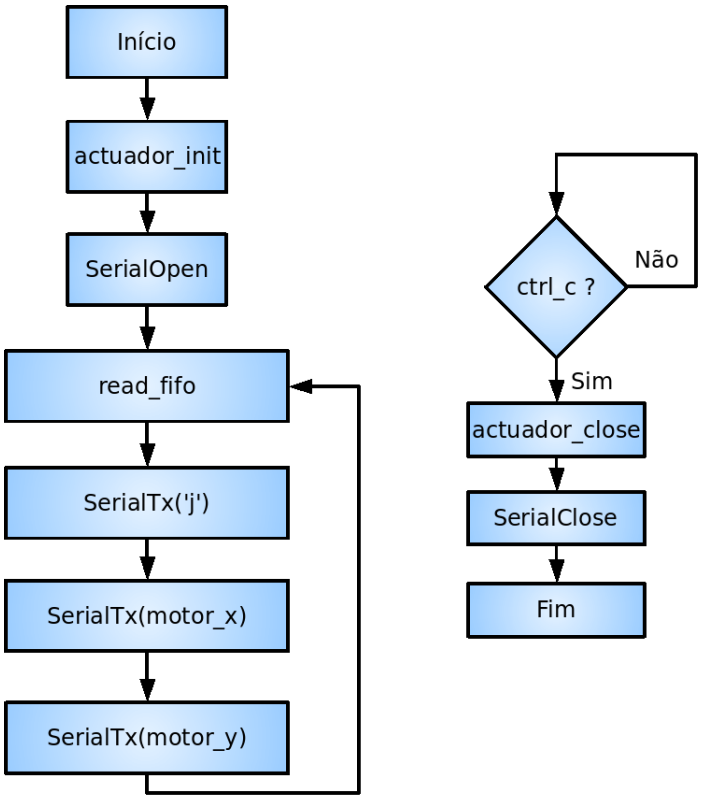


Figura 6.31: Diagrama de Blocos do Actuador em *User-Space*

6.5.2.5 Cam-Int

Este nó serviu apenas para, na fase de testes, simular a transferência de tráfego semelhante ao do nó "Sensor". Usou-se, para o *Kernel-Space*, a aplicação "sensor_ks" tendo apenas que fazer uma pequena alteração ao programa já elaborado para comunicar com FIFO e Memória Partilhada em User-Space.

Parte III

Resultados

Capítulo 7

Resultados

Neste capítulo são apresentados os resultados dos testes realizados à plataforma "Bola no Plano". Nas secções iniciais do capítulo são descritos os aspectos relacionados com os testes tendo em conta as especificidades do protocolo Raw-Ethernet e do protocolo FTT-SE.

Seguidamente são apresentados e analisados os resultados dos testes em Raw-Ethernet e em FTT-SE.

No final do capítulo é feito um estudo comparativo dos resultados obtidos nas experiências realizadas.

7.1 Descrição dos testes

Os teste realizados com a plataforma "Bola no Plano" tiveram como principal objectivo comparar o desempenho do sistema usando o protocolo FTT-SE com o desempenho usando Raw-Ethernet na presença de interferência na rede.

Foi uma preocupação no planeamento destes testes que estes fossem inspirados em situações reais. Na prática isto traduziu-se na escolha, para os elementos de interferência, de nós com tráfego idêntico ao nó do sensor. Os testes partiram do seguinte problema:

- *Numa determinada indústria tem-se um sistema de controlo distribuído baseado em visão, composto por 3 nós: um sensor que essencialmente é uma câmara com um tamanho de imagem de 320x240 em formato GrayScale (8Gpp) e uma frequência de 30 frames por segundo; um controlador que processa a informação proveniente do sensor executando de seguida o algoritmo de controlo e enviando o resultado para um terceiro nó, o actuador, que se encarrega de actuar no sistema físico.*

- *Juntamente com este processo essa indústria possui mais duas câmaras de vigilância com características de imagem idênticas às do sensor, ou seja, imagem com tamanho 320x240 em formato GrayScale. As imagens destas cãmaras são enviadas para o computador "Controlador".*

Todos estes componentes encontram-se ligados a uma rede Ethernet e pretende-se determinar que impacto estas câmaras têm no controlo do sistema. Para isso fez-se variar a taxa de utilização da rede desde o mínimo, ou seja, apenas os componentes de controlo em funcionamento (Sensor, Controlador e Actuador) até taxas para as quais o controlo não era possível.

A variação da taxa de utilização da rede foi conseguida fazendo variar a frequência de *frames* das câmaras de vigilância.

Na figura 7.1 é mostrado o esquema de teste do sistema implementado.

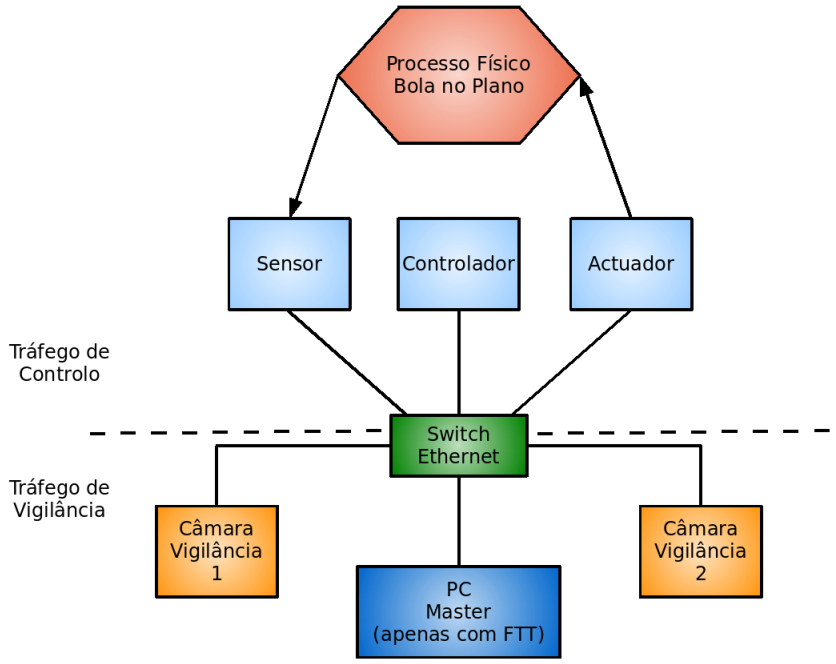


Figura 7.1: Esquema de Teste do Sistema

Na prática, foi necessário simular as câmaras de vigilância. Para isso foi usado a aplicação "Cam_Int" descrita com pormenor na parte de Implementação deste documento.

Dada a elevada quantidade de resultados produzidos, foram desenvolvidos *scripts* no Matlab© que se encontram no capítulo B dos Anexos.

7.1.1 Testes em Raw Ethernet

O objectivo destes testes foi encontrar o ponto a partir do qual se começa a verificar uma degradação significativa do desempenho do sistema.

De uma forma geral pretendeu-se verificar as seguintes propriedades:

- *Falta de isolamento temporal das mensagens, ou seja, a concorrência pelo acesso ao barramento leva a um atraso variável no envio da informação;*
- *Perda de frames no switch avaliando o impacto que este fenómeno tem no controlo.*

7.1.2 Testes com o Protocolo FTT-SE

No protocolo FTT-SE foram realizados testes para demonstrar algumas das suas propriedades.

Foi atribuído uma prioridade superior às mensagens do sensor deixando as mensagens de interferência com prioridade menor. Estas mensagens são enviadas na janela assíncrona sendo a mensagem do actuador enviada na janela síncrona com um período de 1ms.

Atribuiu-se uma percentagem de 85% da largura de banda às mensagens assíncronas (sensor e câmaras de vigilância) ficando o restante destinado às mensagens síncronas (mensagem para o actuador).

De uma forma geral pretendeu-se verficar as seguintes propriedades:

- *O isolamento temporal das mensagens do sensor, ou seja, pretendeu-se verificar que as mensagens do sensor não sofrem Jitter*;
- *A não degradação do desempenho do controlo, ou seja, a gestão da rede feita pelo Master impede que as mensagens do sensor sejam perdidas degradando o tráfego de interferência e não o de controlo.*

7.1.3 Realização Prática

Com a finalidade de comparar o desempenho do sistema nas diferentes condições dos testes foram calculados os seguintes parâmetros:

- *Erro Quadrático Médio da posição da bola segundo os eixos X e Y;*
- *Valor Médio da posição da bola segundo os eixos X e Y;*
- *Desvio Padrão da posição da bola segundo os eixos X e Y;*
- *Tempo Médio entre Frames recebidas;*
- *Máximo Tempo entre Frames recebidas;*
- *Mínimo Tempo entre Frames recebidas;*
- *Desvio Padrão do Tempo entre Frames;*
- *Jitter Absoluto do Tempo entre Frames;*
- *Percentagem de Frames Perdidas.*

Para calcular o Erro Quadrático Médio, Valor Médio e Desvio Padrão da posição da Bola foi determinado, para todas as experiências, um intervalo de 1000 valores centrado na amostra 1000, calculando posteriormente os parâmetros a partir desse intervalo.

Para efeitos de análise do sistema, o nó controlador guardou as sucessivas posições da bola ao longo das experiências, bem como os instantes de recepção das *frames* e o número de cada *frame*. No final de cada experiência esses valores foram recolhidos e processados com a finalidade de calcular os parâmetros anteriormente descritos.

Para a realização destes testes foram usados os seguintes componentes:

- *PC Sensor* - Pentium 4 a 2.8 GHz, 450 MB de Memória RAM e Sistema Operativo Linux com kernel 2.6.9-RT;
- *PC Controlador* - Pentium III a 450 MHz, 190 MB de Memória RAM e Sistema Operativo Linux com kernel 2.6.9-RT;
- *PC Actuador* - Pentium III a 550 MHz , 256 MB de Memória RAM e Sistema Operativo Linux com kernel 2.6.9-RT;

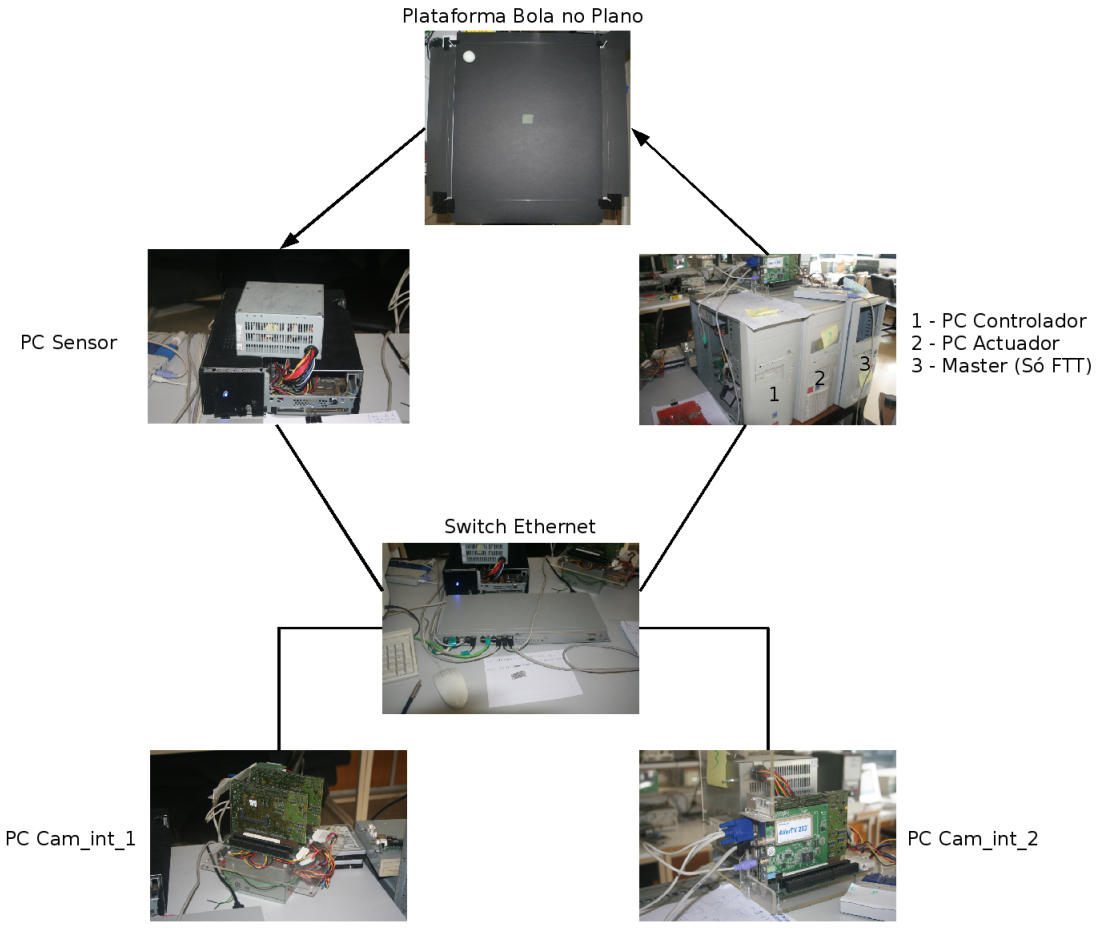


Figura 7.2: Montagem Prática dos Componentes

- *PC Master (apenas nas experiências FTT-SE)* - Pentium III a 350 MHz , 256 MB de Memória RAM e Sistema Operativo Linux com kernel 2.6.9-RT;

- *PC Cam_Int_1* - Mobile Pentium MMX a 267 MHz, 128 MB de Memória RAM e Sistema Operativo Linux com kernel 2.6.9-RT;

- *PC Cam_Int_2* - Mobile Pentium MMX a 267 MHz, 128 MB de Memória RAM e Sistema Operativo Linux com kernel 2.6.9-RT;

- *Switch* - Allied Telesyn AT-8024 Fast-Ethernet com todos os serviços desactivados (Spanning-Tree, Priority Queues Management, Weighted Fair Queues)

Nos teste realizados a rede foi submetida a diferentes taxas de utilização. A variação da taxa de utilização foi conseguida fazendo variar o período entre *Frames* das Câmaras de Interferência. Nas secções 7.2.1 e 7.3.1 é explicado como estas percentagens foram calculadas. Nas sucessivas experiências foi-se alterando a taxa de utilização da rede até ser visível uma degradação considerável no desempenho do controlo.

7.2 Resultados Raw Ethernet

7.2.1 Cálculo da Taxa de Utilização da Rede

Para o cálculo da taxa de utilização da rede é necessário analisar o tráfego presente na ligação do Controlador, visto ser neste que as medidas foram retiradas. Este nó recebe informação proveniente do Sensor e das Câmaras de Interferência e envia informação para o Actuador. A ligação, por ser em *Full-Duplex*, permite que os canais de recepção e envio de informação estejam separados, logo, a ligação crítica nesta situação é a ligação de recepção com capacidade para 100 Mbits/s. Os cálculos que a seguir se apresentam referem-se apenas ao tráfego presente na ligação de recepção do Controlador.

Para o cálculo das taxas de utilização do barramento é necessário determinar que tamanho ocupam os dados que são enviados pela rede. Estes dados são analisados de seguida.

- *Imagem (Mensagens de tipo "b")*

  Para o cálculo da quantidade de *Bytes* transmitidos por cada imagem tem-se em conta o tamanho dos dados da imagem bem como os campos introduzidos pelo protocolo Ethernet (Preamble (8 *Bytes*), Interframing Gap (12 *Bytes*), Header e CRC (18 *Bytes*)). A última mensagem de imagem contém, na zona de dados, os 402 *Bytes* do último fragmento, 2 *Bytes* para os cabaçalhos (ID e SN) e 4 *Bytes* adicionais para o *Frame Number*, o que perfaz o total de 408 *Bytes*,

$$51 \times (1500 + 18 + 12 + 8) + 1 \times (408 + 18 + 12 + 8) = 78884 \ Bytes \tag{7.1}$$

  Este valor, convertido para bits, corresponde à quantidade de dados que são transaccionados por cada imagem enviada.

$$IL_R = 631072 \ bits/imagem \tag{7.2}$$

Na tabela 7.1 estão as respectivas taxas de utilização dos testes realizados. Estas taxas foram calculadas usando a seguintes fórmula:

$$\frac{(SensFps + CI1Fps + CI2Fps) \times IL_R}{100 \times 10^6} \times 100 \tag{7.3}$$

Onde *SensFps* representa a frequência de aquisição de *frames* da câmara do Sensor em *frames/segundo* (fps), *CI1Fps* e *CI2Fps* a frequência de aquisição de *frames* das câmara de interferência 1 e 2 respectivamente em fps e IL_R representa o tamanho de cada imagem em bits.

Sensor	Cam. Int. 1	Cam. Int. 2	Taxa de Utilização
30 fps	0 fps	0 fps	18.932%
30 fps	15 fps	15 fps	37.864%
30 fps	29 fps	24 fps	51.379%
30 fps	38 fps	28 fps	60.583%
30 fps	37 fps	38 fps	65.263%
30 fps	42 fps	42 fps	71.942%
14 fps	49 fps	53 fps	72.204%
13 fps	57 fps	57 fps	78.884%

Tabela 7.1: Tabela das Taxas de Utilização dos Testes em Raw-Ethernet

As frequências de aquisição de *frames* usadas para determinar a taxa de utilização da rede foram medidas para cada uma das experiências com base nos dados obtidos. Os cálculos usados para determinar estas frequências podem ser encontrados no anexo B.1 deste documento.

Os resultados obtidos são apresentados nas secções seguintes.

7.2.2 Apresentação dos Resultados

Os resultados dos testes realizados em Raw-Ethernet são mostrados nas tabelas 7.2 e 7.3.

T. Util. (%)	eqm_x (cm^2)	eqm_y (cm^2)	$\bar{x}$ (cm)	$\bar{y}$ (cm)	σx (cm)	σy (cm)
18.932	8.304	12.123	0.044	0.364	2.882	3.465
37.864	12.465	6.007	0.245	0.407	3.524	2.418
52.379	5.048	7.677	0.000	1.224	2.247	2.487
60.583	15.660	14.911	0.026	0.330	3.959	3.849
66.263	10.223	13.215	0.168	0.087	3.195	3.636
71.942	17.136	12.742	0.008	0.100	4.142	3.570
73.204	267.877	372.182	2.596	1.681	16.172	19.233
78.884	393.301	362.695	-0.029	-0.098	19.845	19.057

Tabela 7.2: Tabela dos resultados dos testes em Raw-Ethernet, análise da posição da Bola

T. Util. (%)	T (ms)	Máx. T (ms)	Mín. T (ms)	σT (ms)	Jit. Abs. (ms)	L. Fr. (%)
18.932	33.332	41.0	29.0	4.707	12.0	0.0
37.864	33.328	44.0	26.0	5.266	18.0	0.0
52.379	33.332	44.0	27.0	5.012	17.0	0.0
60.583	33.336	43.0	23.0	6.269	19.0	0.0
66.263	33.331	46.0	25.0	5.467	21.0	0.0
71.942	33.333	45.0	27.0	4.861	17.0	0.0
73.204	74.042	239.0	29.0	74.042	209.0	54.984
78.884	125.734	300.0	30.0	77.722	259.0	73.490

Tabela 7.3: Tabela dos resultados dos testes em Raw-Ethernet, análise temporal das *Frames*

As figuras 7.3, 7.4, 7.5, 7.6, mostram os gráficos da posição da bola em função do tempo durante a experiência sem interferência (19% de taxa de utilização da rede), e com o máximo de interferência (79%), respectivamente.

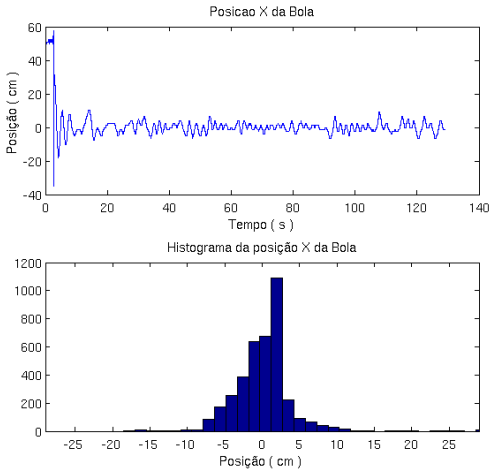


Figura 7.3: Posição X da Bola, Raw-Ethernet com Taxa de Utilização de 19%

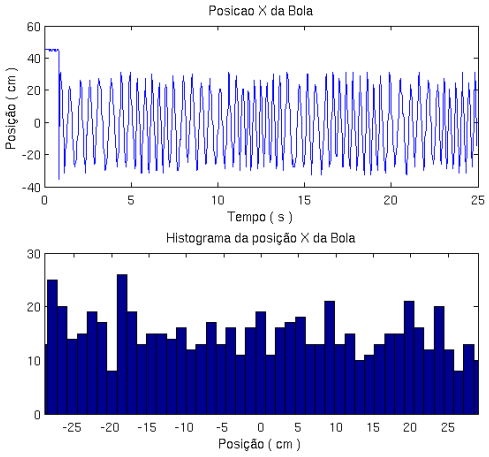


Figura 7.4: Posição X da Bola, Raw-Ethernet com Taxa de Utilização de 79%

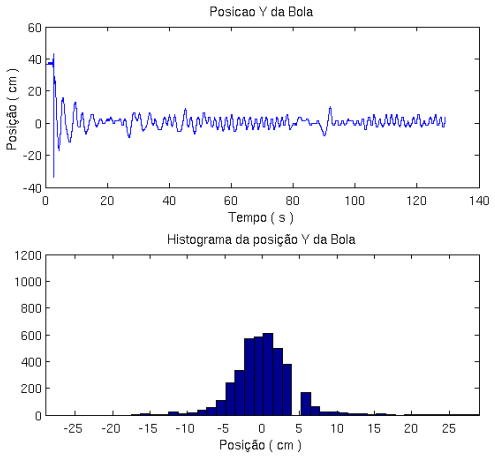


Figura 7.5: Posição Y da Bola, Raw-Ethernet com Taxa de Utilização de 19%

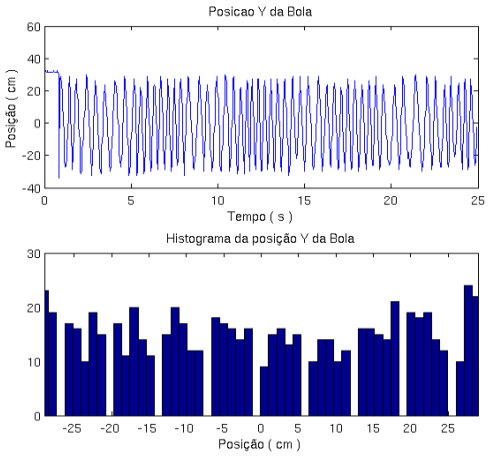


Figura 7.6: Posição Y da Bola, Raw-Ethernet com Taxa de Utilização de 79%

O segundo conjunto de Gráficos refere-se ao *Jitter Relativo* da recepção de *Frames* nas primeira, segunda, penúltima e na última experiências. A inclusão destes gráficos serve para mostrar o espalhamento do histograma do *Jitter Relativo* logo que as câmaras de interferência começam a gerar tráfego.

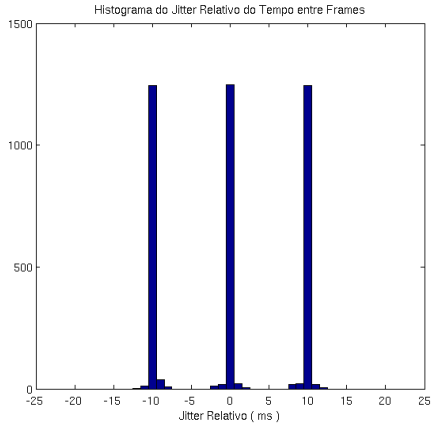


Figura 7.7: *Jitter Relativo* do Tempo entre *Frames*, Raw-Ethernet com Taxa de Utilização de 19%

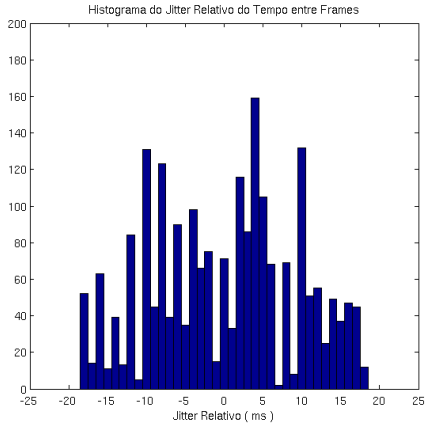


Figura 7.8: *Jitter Relativo* do Tempo entre *Frames*, Raw-Ethernet com Taxa de Utilização de 38%

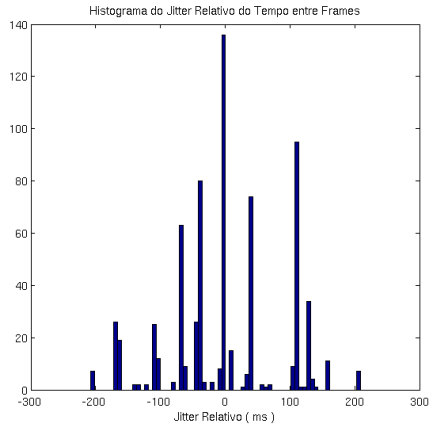


Figura 7.9: *Jitter Relativo* do Tempo entre *Frames*, Raw-Ethernet com Taxa de Utilização de 73%

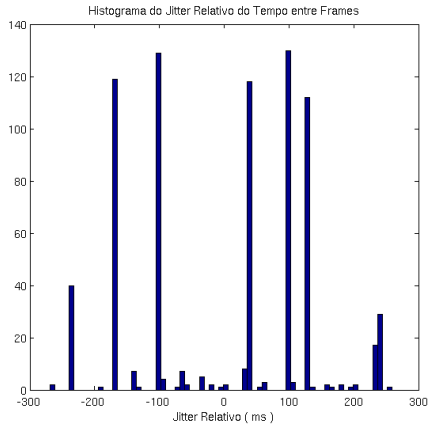


Figura 7.10: *Jitter Relativo* do Tempo entre *Frames*, Raw-Ethernet com Taxa de Utilização de 79%

7.2.3 Avaliação do Desempenho

As figuras 7.3, 7.4, 7.5, 7.6, permitem comparar a situação inicial (sem tráfego de interferência) com a situação final (com o máximo de interferência). A figuras 7.11 e 7.12 mostram os gráficos da evolução do desempenho do sistema com as diferentes taxas de utilização da rede.

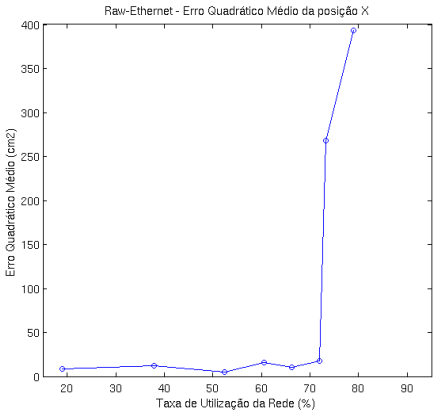


Figura 7.11: Evolução do Erro Quadrático Médio da Posição X da Bola em Função da Taxa de Utilização da Rede

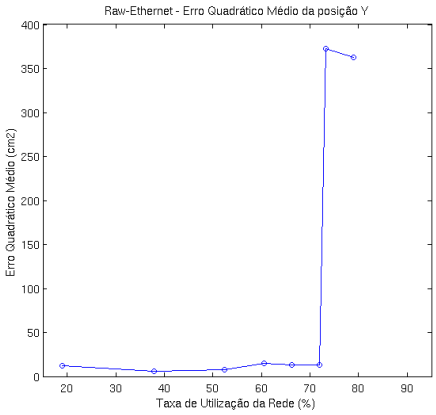


Figura 7.12: Evolução do Erro Quadrático Médio da Posição Y da Bola em Função da Taxa de Utilização da Rede

Pela análise da tabela 7.2 verifica-se que os valores do erro quadrático médio se mantêm estáveis para taxas de utilização menores a 72%. Era espectável que se notasse uma degradação progressiva do comportamento[PLA03], no entanto tal não se verificou. Uma explicação para o sucedido é o facto do *Jitter Absoluto* máximo ocorrido neste conjunto de experiências, 21 ms, ser inferior ao período de amostragem, 33 ms, conjugado com o facto de a câmara possuir uma latência elevada, na ordem dos 60 ms, fez com que o *Jitter* não afectasse significativamente o controlo. Para confirmar esta hipótese teriam que ser repetidas mais experiências para corrigir algumas variações aos valores apresentados e feita uma análise mais pormenorizada ao impacto do atraso da transmissão no comportamento do sistema.

Analisando apenas os gráficos 7.3, 7.4, 7.5 e 7.6, demonstrativos das experiências, é perceptível uma situação em que claramente o sistema é controlado e outra em que esse controlo não foi eficaz. Ao longo das experiências o valor do erro quadrático médio, enquanto o valor da taxa de utilização foi inferior a 72%, manteve-se dentro de valores que permitiram atingir, em todos eles, a convergência do sistema. A explicação para tal não ter acontecido a partir destas taxas de utilização deduz-se dos gráficos 7.13 e 7.14 que mostram a evolução do *Jitter Absoluto* e da *Percentagem de Frames Perdidas* em função da taxa de utilização da rede.

Na análise do comportamento temporal das *Frames* há a destacar o gráfico 7.7, gráfico correspondente à experiência sem tráfego de interferência. Neste gráfico destacam-se três valores, -10, 0 e 10. Estes valores devem-se ao facto da aplicação que efectua a leitura do *Device Driver* da câmara apenas ser activada em instantes de tempo múltiplos de 10 ms. O comportamento visível no gráfico reflecte o funcionamento normal da câmara.

Numa primeira análise aos gráficos presentes nas figuras 7.7, 7.8, 7.9 e 7.10 salienta-se, logo a partir do momento em que é introduzido tráfego de interferência, uma alteração significativa

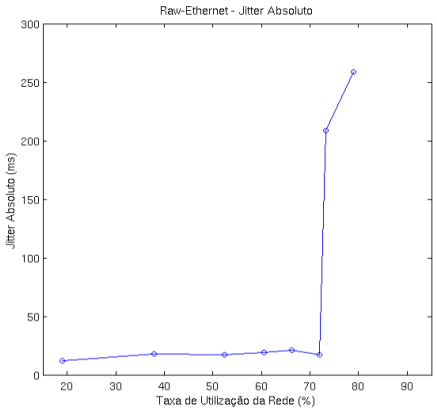


Figura 7.13: Evolução do *Jitter Absoluto* em Função da Taxa de Utilização da Rede

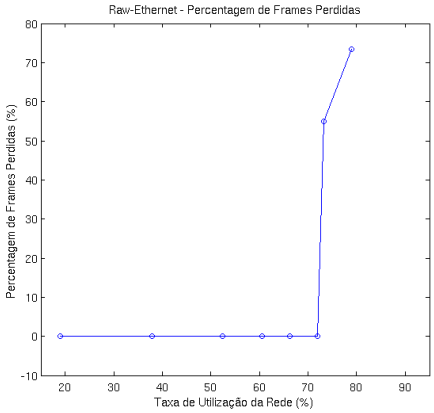


Figura 7.14: Evolução da *Percentagem de Frames Perdidas* em Função da Taxa de Utilização da Rede

do *Jitter Relativo*. É visível um espalhamento em torno dos valores centrais espectáveis.

Analisando os gráfico do *Jitter Relativo*, em certos casos nota-se uma diminuição do valor central "0", este facto não foi constante ao longo das experiências, umas experiências apresentavam um valor elevado outras apresentavam-no baixo. Esta variação poderá dever-se ao facto dos nós que geram o tráfego, Sensor e Câmaras de Interferência, não estarem sincronizados e o tempo de cada experiência (cerca de 1 minuto) não ter sido suficiente para permitir todas as combinações de relógios possíveis. A averiguação deste fenómeno poderá ser o objecto de um trabalho futuro.

Os gráficos 7.13 e 7.14 mostram a evolução do *Jitter Absoluto* e da *Percentagem de Frames Perdidas* com as diferentes taxas de utilização da rede. Mais uma vez nota-se a degradação do comportamento temporal das *Frames* quando a taxa de utilização ultrapassa os 72%. Até este valor o comportamento temporal, *Jitter Absoluto*, manteve-se em valores aproximadamente constantes, no entanto, quando é atingido a taxa de utilização de 72% o valor do *Jitter Absoluto* atinge valores elevados motivados pela *Percentagem de Frames Perdidas* elevada.

Analisando os resultados na sua globalidade é visível que a degradação do desempenho do sistema é motivado pela perda de *frames* provenientes do sensor. Como o algoritmo de controlo só é executado aquando da chegada de uma *frame*, o facto destas se perderem faz com que haja uma degradação significativa no desempenho do sistema devido ao atraso variável que é introduzido. Nas situações limite este tempo entre frames chegou, no máximo, a valores de cerca de 200 ms (valor do *Jitter Absoluto*). Para o correcto funcionamento do sistema este valor deveria rondar os 33ms (Período de Amostragem) nestes casos foram atingidos valores uma ordem de grandeza superior que produz um impacto significativo no controlo.

7.3 Resultados FTT-SE

7.3.1 Cálculo da Taxa de Utilização da Rede

No caso do protocolo FTT-SE além das imagens, na ligação de recepção do Controlador, é também recebida a *Trigger Message*.

- *Imagem*

  Neste caso foi usado um MTU de 1400 *Bytes*, ou seja, no máximo em cada mensagem irão apenas 1400 *Bytes* da imagem. Refazendo os cálculos relativos à quantidade de mensagens enviadas chega-se ao valor de 55 mensagens, 54 com 1400 *Bytes* e 1 com 1200 (mais 4 *Bytes* do *frame number*). Aos dados juntam-se os restantes *Bytes* próprios da rede Ethernet já referenciados e ainda 8 *Bytes* suplementares usados pelo FTT-SE na transmissão.

$$54 \times (1400 + 18 + 8 + 12 + 8) + 1 \times (1204 + 18 + 8 + 12 + 8) = 79334 \ Bytes \tag{7.4}$$

  Este valor convertido para bits corresponde à quantidade de dados que é transaccionada por cada imagem enviada.

$$IL_F = 634672 \ bits/imagem \tag{7.5}$$

- *Trigger Message*

  Esta mensagem para ser enviada precisa apenas de

$$46 + 18 + 12 + 8 + 8 = 92 \ Bytes \tag{7.6}$$

  Este valor convertido para *bits* corresponde à quantidade de dados que é transaccionada por cada mensagem enviada.

$$TML_F = 736 \ bits/TM \tag{7.7}$$

Na tabela 7.4 estão as respectivas taxas de utilização dos testes realizados. Estas taxas foram calculadas usando a seguintes fórmula:

$$\frac{(SensFps + CI1Fps + CI2Fps) \times IL_F + 1000 \times TML_F}{100 \times 10^6} \times 100 \tag{7.8}$$

Onde *SensFps* representa a frequência de aquisição de *frames* da câmara do Sensor em *frames/segundo* (fps), *CI1Fps* e *CI2Fps* a frequência de aquisição de *frames* das câmara de interferência 1 e 2 respectivamente em fps e TML_F a quantidade de dados enviados numa Trigger Message em *bits*.

Sensor	Cam. Int. 1	Cam. Int. 2	Taxa de Utilização
30 fps	0 fps	0 fps	19.776%
30 fps	15 fps	15 fps	38.816%
30 fps	29 fps	24 fps	53.414%
30 fps	38 fps	28 fps	61.665%
30 fps	37 fps	38 fps	67.377%
30 fps	42 fps	42 fps	73.089%
30 fps	49 fps	53 fps	84.513%
30 fps	57 fps	57 fps	90.859%

Tabela 7.4: Tabela das Taxas de Utilização dos Testes com FTT-SE

Para a determinação das taxas de utilização foram usadas as frequências medidas para rede Ethernet e os testes repetidos nas mesmas condições. Apenas há a referir que nas duas últimas experiências, ao contrário da rede Ethernet, conseguiu-se obter uma frequência do Sensor de 30 fps pelo que este facto foi tido em conta no cálculo das taxas de utilização.

7.3.2 Apresentação dos Resultados

Os resultados dos testes realizados ao sistema com o protocolo FTT-SE são mostrados nas tabelas 7.5 e 7.6

T. Util. (%)	eqm_x (cm^2)	eqm_y (cm^2)	$\bar{x}$ (cm)	$\bar{y}$ (cm)	σx (cm)	σy (cm)
19.776	11.294	12.642	0.171	0.138	3.358	3.555
38.816	5.065	17.467	0.177	-0.004	2.245	4.181
53.414	4.094	7.018	-0.074	0.059	2.023	2.650
61.665	7.453	8.996	-0.082	0.257	2.730	2.990
67.377	3.935	6.877	-0.064	0.709	1.984	2.526
73.089	6.510	8.044	-0.065	1.656	2.552	2.304
84.513	9.450	8.493	0.185	0.153	3.070	2.912
90.859	5.169	22.695	0.116	0.283	2.272	4.758

Tabela 7.5: Tabela dos resultados dos testes com FTT-SE, análise da posição da Bola

T. Util. (%)	T (ms)	Máx. T (ms)	Mín. T (ms)	σT (ms)	Jit. Abs. (ms)	L. Fr. (%)
19.776	33.314	41.994	27.993	4.717	12.010	0.0
38.816	33.333	41.198	28.797	4.710	12.009	0.0
53.414	33.288	41.200	28.787	4.709	12.411	0.0
61.665	33.325	41.994	27.995	4.711	12.016	0.0
67.377	33.299	41.003	28.976	4.707	12.016	0.0
73.089	33.302	41.001	28.885	4.709	12.014	0.0
84.513	33.298	42.004	27.992	4.710	12.024	0.0
90.859	33.356	41.003	28.980	4.731	12.019	0.0

Tabela 7.6: Tabela dos resultados dos testes com FTT-SE, análise temporal das *Frames*

Os gráficos que se mostram de seguida referem-se à posição da bola da primeira experiência (sem tráfego de interferência) e à posição da bola da última experiência (com a interferência máxima na rede).

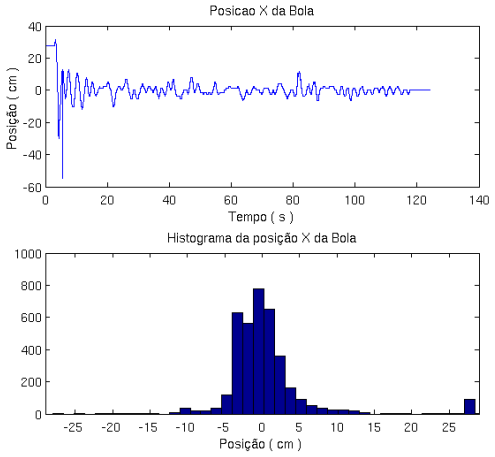


Figura 7.15: Posição X da Bola, FTT-SE com Taxa de Utilização de 20%

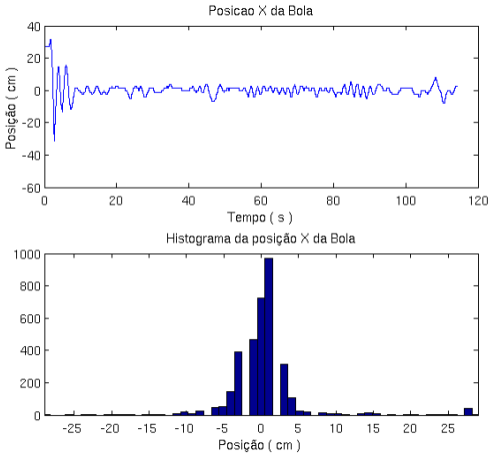


Figura 7.16: Posição X da Bola, Raw-Ethernet com Taxa de Utilização de 91%

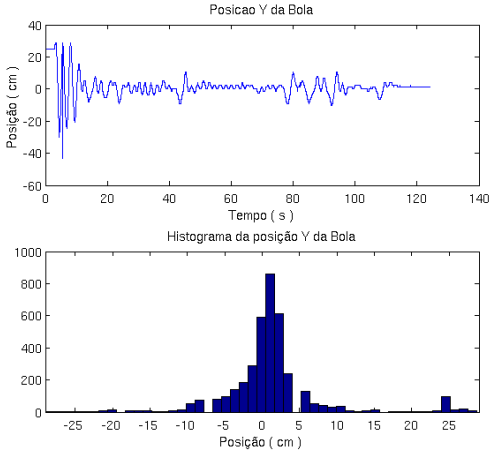


Figura 7.17: Posição Y da Bola, FTT-SE com Taxa de Utilização de 20%

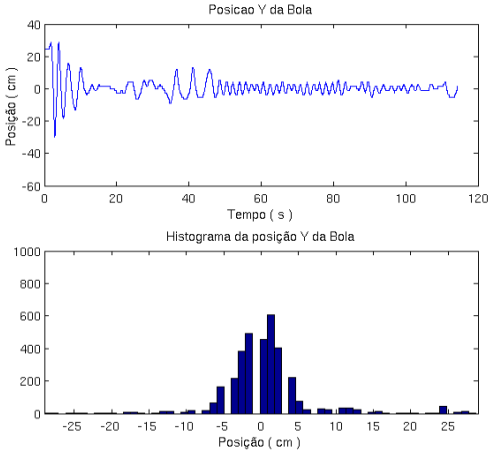


Figura 7.18: Posição Y da Bola, Raw-Ethernet com Taxa de Utilização de 91%

O segundo conjunto de gráficos referem-se ao *Jitter Relativo* da recepção de *Frames* nas primeira, segunda, penúltima e na última experiências.

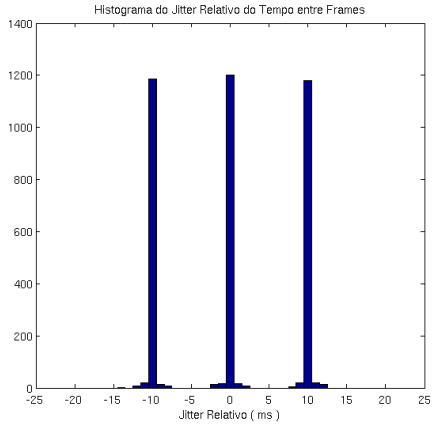


Figura 7.19: *Jitter Relativo* do Tempo entre *Frames*, FTT-SE com Taxa de Utilização de 20%

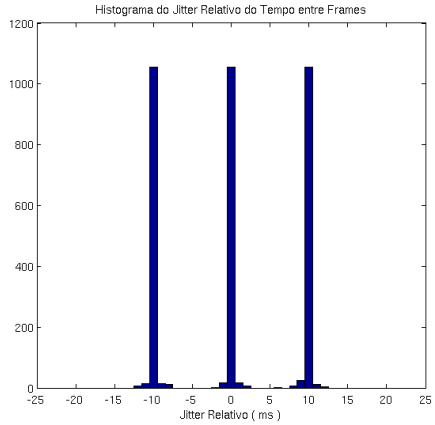


Figura 7.20: *Jitter Relativo* do Tempo entre *Frames*, FTT-SE com Taxa de Utilização de 39%

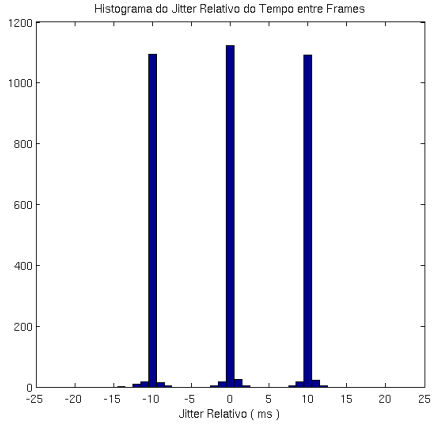


Figura 7.21: *Jitter Relativo* do Tempo entre *Frames*, FTT-SE com Taxa de Utilização de 85%

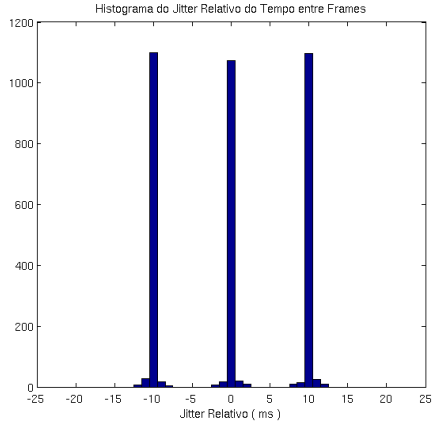


Figura 7.22: *Jitter Relativo* do Tempo entre *Frames*, FTT-SE com Taxa de Utilização de 91%

7.3.3 Avaliação do Desempenho

As figuras 7.15, 7.16, 7.17 e 7.18 mostram os gráficos do desempenho do sistema nas situações em que não há qualquer interferência, e quando a interferência é máxima. Os gráficos evidenciam um desempenho semelhante nas duas situações.

Para analisar com maior detalhe a evolução do desempenho do sistema com a variação da taxa de utilização da rede, são apresentados as figuras 7.23 e 7.24 apresentando os gráficos escalas semelhantes aos gráficos do Raw-Ethernet.

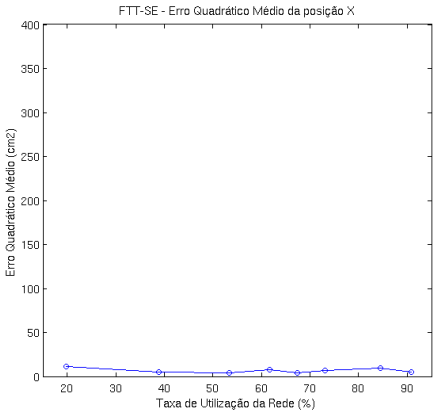


Figura 7.23: Evolução do Erro Quadrático Médio da Posição X da Bola em Função da Taxa de Utilização da Rede

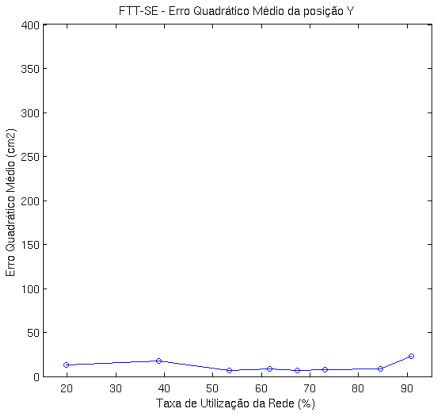


Figura 7.24: Evolução do Erro Quadrático Médio da Posição Y da Bola em Função da Taxa de Utilização da Rede

No cálculo das taxas de utilização chegou-se a valores superiores a 85% para o tráfego das câmaras, tráfego assíncrono. Estes valores nunca poderiam ser atingidos devido ao facto de se ter limitado o tamanho da janela assíncrona a 85% da largura de banda. Não foi possível, no decorrer das experiências, recolher informação temporal da recepção das *frames* de interferência, pelo que não foi possível determinar a real taxa de utilização para este conjunto de experiências. Perante esta impossibilidade as experiências e os cálculos foram feitos com base na informação retirada na rede Ethernet a fim de manter a coerência das experiências e estabelcer um método de fácil comparação entre os dois protocolos. Como o tráfego assíncrono não poderia exceder os 85%, a gestão realizada pelo *Master*, limitou este tráfego fazendo com que o tráfego das Câmaras de Interferência perdesse qualidade de serviço, tal como era pretendido.

Da análise da evolução do desempenho do sistema com a variação da taxa de utilização da rede verifica-se que o desempenho se mantém substancialmente constante ao longo das experiências. Este facto pode ser explicado pelo comportamento temporal das *Frames* mostrado nos gráficos das figuras 7.25 e 7.26.

A análise dos gráficos das figuras 7.19, 7.20, 7.21 e 7.22 mostra que nas experiências exemplo o tráfego de controlo não sofreu *Jitter* revelando o isolamento temporal que o protocolo FTT-SE conferiu ao tráfego do Sensor.

Pela análise dos gráficos das figuras 7.25 e 7.26 verifica-se que o comportamento temporal foi constante ao longo das experiências o que levou a que em nenhuma delas fossem perdidas *frames*. O isolamento temporal é também visível na regularidade do *Jitter Absoluto* da

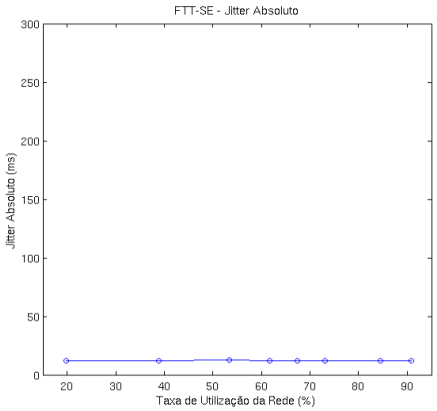


Figura 7.25: Evolução do *Jitter Absoluto* em Função da Taxa de Utilização da Rede

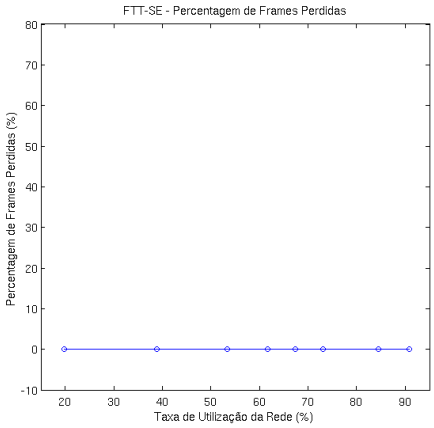


Figura 7.26: Evolução da *Percentagem de Frames Perdidas* em Função da Taxa de Utilização da Rede

recepção de *frames*.

Analisando os resultados na sua globalidade verifica-se que o isolamento temporal conferido ao tráfego do sensor levou a que não fossem perdidas *frames* e que o *Jitter* fosse constante, o que levou a que o algoritmo de controlo fosse executado sem sofrer qualquer tipo de interferência. Este factores fizeram com que em todas as experiências fosse atingido um comportamento estável do sistema.

7.4 Análise Comparativa

Os gráficos das figuras 7.27 e 7.28 mostram a evolução do desempenho do sistema, em condições semelhantes, ou seja, a mesma frequência de geração de frames de interferência, nas experiências com Raw-Ethernet e com o protocolo FTT-SE.

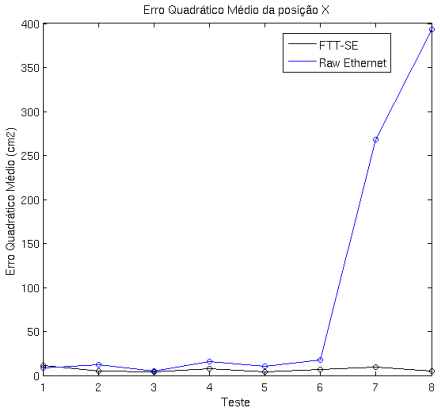


Figura 7.27: Erro Quadrático Médio da Posição X da bola

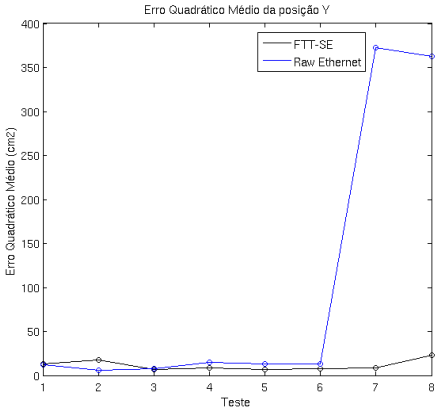


Figura 7.28: Erro Quadrático Médio da Posição Y da bola

Os gráficos mostram um comportamento semelhante da rede Raw-Ethernet e da rede com o protocolo FTT-SE para taxas de utilização baixas. Quando é atingido o ponto de ruptura da rede Raw-Ethernet o controlo deixa de ser possível nesta rede. No entanto o desempenho do sistema com o protocolo FTT-SE não sofre alteração significativa.

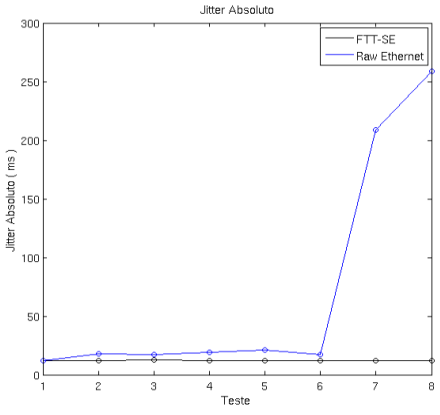


Figura 7.29: *Jitter Absoluto*

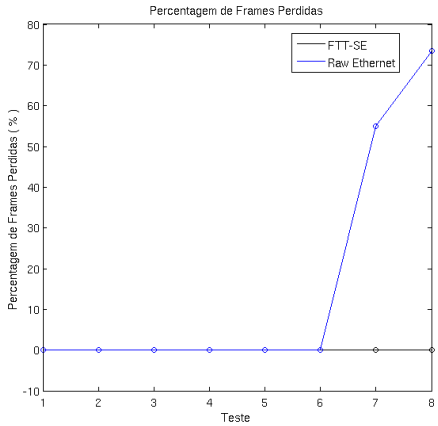


Figura 7.30: *Percentagem de Frames Perdidas*

Em todos os aspectos alvo deste estudo o protocolo FTT-SE mostrou um melhor desempenho do ponto de vista da diferenciação de tráfego que a rede Raw-Ethernet. Ficou demonstrado que para este tipo de aplicações o protocolo confere à rede garantias de um

correcto funcionamento, independentemente da carga a que a rede está sujeita. De notar também que nas experiências feitas com o protocolo FTT-SE as mensagens do sensor, por terem a maior prioridade, não sofrem Jitter considerável o que mais uma vez demonstra a eficiência do uso do protocolo FTT-SE.

Parte IV

Conclusão

Capítulo 8

Conclusão

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.

8.1 Análise Global dos Resultados

Os resultados do trabalho desenvolvido permitiram confirmar a grande vantagem da utilização de um protocolo de Tempo-Real em aplicações de controlo de sistemas. Esta dissertação permitiu montar uma plataforma de testes que permitiu não só controlar os sistemas físicos como sobmete-los a testes de desempenho.

Os resultados foram satisfatórios conseguindo-se verificar experimentalmente a aplicabilidade do protocolo FTT-SE em sistemas de Controlo Distribuídos em Tempo-Real, em particular as capacidades de diferenciação de tráfego e isolamento temporal.

8.2 Análise dos Objectivos

Nas plataformas que foram alvo de estudo desta dissertação os principais objectivos foram alcançados. As plataformas foram convertidas para controlo distribuído sendo mantida a possibilidade de voltar ao controlo centralizado mantendo assim a compatibilidade com o trabalho passado e possibilitando, no futuro, voltar a este tipo de controlo. O controlo das plataformas foi alvo de significativas melhorias, no caso particular da plataforma "Bola no Plano" o processamento de imagem foi alvo de grandes alterações conseguindo um desempenho global do sistema satisfatório.

A implementação do protocolo FTT-SE na plataforma "Bola no Plano" foi feita com sucesso conseguindo alcançar um desempenho do sistema melhor que a utilização apenas da rede Ethernet.

A alteração dinâmica de QoS não foi alvo do estudo nesta dissertação.

A aplicação de outros métodos de controlo não foi também estudado pelo facto de se ter conseguido resultados satisfatórios com o método utilizado.

8.3 Trabalho Futuro

Em trabalho futuro poderá ser avaliado o interesse de aplicar o protocolo FTT-SE às restantes plataformas implementando um sistema que contemple todas as plataformas a funcionar simultaneamente.

Ficaram por fazer alguns testes que poderiam ter interesse. Existem alguns serviços nos *switches* que podem ser utilizados para reservar largura de banda para um determinado porto. Seria interessante comparar os resultados do FTT-SE com a rede Ethernet com esses serviços activos, por exemplo a atribuição de prioridades diferentes aos portos, ou outros serviços contemplados na norma IEEE 802.1D-2004.

O estudo da alteração dinâmica de QoS poderá também ser alvo de estudo, testando assim outra característica do protocolo FTT-SE.

O estudo de outras técnicas de controlo, como controlo em Espaço de Estados, controlo *Fuzzy*, controlo adaptativo, poderá ser feito a fim de melhorar o controlo das plataformas. Neste campo um trabalho sério de modelação da plataforma traria mais valias ao trabalho.

Ainda como protosta de trabalho futuro a modificação de algumas características físicas das plataforma, com a finalidade de melhorar a resposta do sistema, poderia ser ponderado a fim de melhorar a qualidade do controlo e dos testes realizados.

8.4 Considerações Finais

A dissertação "Controlo Distribuído de Plataformas para Experiências de Mecatrónica" apresenta um carácter multidisciplinar correndo transversalmente as áreas ligadas à electrónica e mecânica. É um trabalho em constante progresso não ficando por isso encerrado aqui. As plataformas pelo seu carácter didáctico, serão alvo de novas implementações a fim de testar outros protocolos e controladores à medida que estes forem evoluindo.

Este documento tem o intuito de ser um comprovativo do trabalho realizado durante o tempo de preparação desta dissertação fornecendo uma introdução aos conceitos fundamentais que suportaram a parte de implementação, parte essa também aqui alvo de uma atenção especial fornecendo assim documentação para a continuação futura do trabalho. Este documento pretende servir de testemunho dos resultados obtidos durante a realização desta dissertação.

A dissertação que aqui finda, pelos resultados que atingiu e por todo o trabalho que desenvolveu, foi concluída com sucesso.

Parte V

Anexos

Apêndice A

Código da Implementação

A.1 Código da Modelação do Sistema

```
 clear all; clc;

x = [0 0];
o = ones(1,1000);

T = 0.033;
g = 9.81;

%Transformada Z
for n = 3:1000
  x(n) = 0.25*g*T^2*o(n) + 0.5*g*T^2*o(n-1) + 0.25*g*T^2*o(n-2) + 2*x(n-1) - x(n-2);
end

t=(1:1000)*T; % Passagem de amostras para segundos


%Transformada S
s=tf('s');
gs=g/(s*s);



%Gráficos
figure
subplot(2,1,1)
plot(t,x)
title('Resposta ao degrau do modelo em Transformada Z')
xlabel('Tempo ( s )')
ylabel('Posição da Bola ( m )')

subplot(2,1,2)
```

```
step(gs, 1000*T)
title('Resposta ao degrau do modelo em Transformada S')
xlabel('Tempo ( s )')
ylabel('Posição da Bola ( m )')



figure
plot(t,x, 'r')
hold on
step(gs, 1000*T)
hold off
title('Comparação entre as respostas ao degrau dos modelos')
xlabel('Tempo ( s )')
ylabel('Posição da Bola ( m )')
legend('Transformada Z', 'Transformada S')
```

A.2 Código da Modelação do Compensador PID

```
close all; clear all; clc;

s_fact = 23/0.30;
a_fact = 45/23;

x = [-0.2, -0.2];
o = [0.1 0.1];

T = 0.033;
g = 9.81;



P = 0.03;
I = 0;
D = 0.3;

integral=0;
lasterro=0;

for n = 3:100

  %Sensor
  sensor = round(x(n-1)*s_fact);
  erro = -sensor;
```

```
  %controlador
  o(n) = (P*erro+D*(erro - lasterro) + I*integral);
  lasterro=erro;

  %actuador
  o(n)=o(n)*a_fact;

  %sistema físico
  x(n)=0.25*g*T^2*o(n) + 0.5*g*T^2*o(n-1) + 0.25*g*T^2*o(n-2) + 2*x(n-1) - x(n-2);
end

%Gráfico
figure(1);
plot(x)
title('Aplicação do compensador PID ao Sistema')
xlabel('Amostra')
ylabel('Posição (m)')
```

Apêndice B

Código dos Resultados

B.1 Código usado para gerar os Resultados Raw-Ethernet

```
close all; clear all; clc;

valid_int = 1000;
s_fact = 30 / 23 ;
sensorinfo=load('sensor.log');

%Análise da Posição
pos_x = sensorinfo(:,3)*s_fact;
pos_y = sensorinfo(:,4)*s_fact;
t = (1:length(pos_x))*0.033;


figure(1)
subplot(2,1,1)
plot(t,pos_x)
title('Posicao X da Bola')
xlabel('Tempo ( s )')
ylabel('Posição ( cm )')
subplot(2,1,2)
hist(pos_x, 61)
title('Histograma da posição X da Bola')
xlabel('Posição ( cm )')
axis([-29 29 0 500])



figure(2)
subplot(2,1,1)
plot(t, pos_y)
title('Posicao Y da Bola')
xlabel('Tempo ( s )')
```

```
ylabel('Posição ( cm )')
subplot(2,1,2)
hist(pos_y, 61)
title('Histograma da posição Y da Bola')
xlabel('Posição ( cm )')
axis([-29 29 0 500])


  % erro quadrático médio calculado apenas no intervalo de 1001 valores
  em_x = mean(pos_x(valid_int-500:valid_int+500).^2)
  em_y = mean(pos_y(valid_int-500:valid_int+500).^2)

  med_x = mean(pos_x(valid_int-500:valid_int+500))
  med_y = mean(pos_y(valid_int-500:valid_int+500))

  std_x = std(pos_x(valid_int-500:valid_int+500))
  std_y = std(pos_y(valid_int-500:valid_int+500))


% Análise temporal

t_sensor = sensorinfo(:,1);

  % sensor Timestamp
  for i=2:length(t_sensor)
  tstamp_sensor(i-1) = t_sensor(i) - t_sensor(i-1);
  end

  sensor_fps = mean(tstamp_sensor)
  sensor_maxfps = max(tstamp_sensor)
  sensor_minfps = min(tstamp_sensor)
  sensor_fpsstd = std(tstamp_sensor)

  for i=2:length(tstamp_sensor)
  jitter_rel(i-1) = tstamp_sensor(i) - tstamp_sensor(i-1);
  end

  jitter_abs = max(jitter_rel)

  figure(3)
  hist(jitter_rel, -70:70)
  title('Histograma do Jitter Relativo do Tempo entre Frames')
  xlabel('Jitter Relativo ( ms )')
  axis([-25 25 0 140])

% Perda de frames
```

```
frames_ns = sensorinfo(:,2);
fr_total = frames_ns(end)-frames_ns(1);

  for i=2:length(frames_ns)
  delta_frames(i-1) = frames_ns(i) - frames_ns(i-1);
  end

  lost_frames_tot = delta_frames(find(delta_frames > 1)) -1;

  lost_frames = (sum(lost_frames_tot) / fr_total) * 100


%frequencia da camara de interferencia 1
cam1=load('CAM1.log');

% cam1 Timestamp
for i=2:length(cam1)
tstamp_cam1(i-1) = cam1(i) - cam1(i-1);
end

cam1_fps = mean(tstamp_cam1)
cam1_maxfps = max(tstamp_cam1)
cam1_minfps = min(tstamp_cam1)

%frequencia da camara de interferencia 2
cam2=load('CAM2.log');

% cam2 Timestamp
for i=2:length(cam2)
tstamp_cam2(i-1) = cam2(i) - cam2(i-1);
end

cam2_fps = mean(tstamp_cam2)
cam2_maxfps = max(tstamp_cam2)
cam2_minfps = min(tstamp_cam2)
```

B.2 Código usado para gerar os Resultados FTT-SE

```
close all; clear all; clc;

sensorinfo=load('log.txt');

valid_int = 1000;
s_fact = 30 / 23 ;
```

```
%Análise da Posição
pos_x = sensorinfo(:,3)*s_fact;
pos_y = sensorinfo(:,4)*s_fact;
t = (1:length(pos_x))*0.033;


figure(1)
subplot(2,1,1)
plot(t,pos_x)
title('Posicao X da Bola')
xlabel('Tempo ( s )')
ylabel('Posição ( cm )')
subplot(2,1,2)
hist(pos_x, 61)
title('Histograma da posição X da Bola')
xlabel('Posição ( cm )')
axis([-29 29 0 2000])



figure(2)
subplot(2,1,1)
plot(t, pos_y)
title('Posicao Y da Bola')
xlabel('Tempo ( s )')
ylabel('Posição ( cm )')
subplot(2,1,2)
hist(pos_y, 61)
title('Histograma da posição Y da Bola')
xlabel('Posição ( cm )')
axis([-29 29 0 2000])


  % erro quadrático médio calculado apenas no intervalo de 1001 valores
  em_x = mean(pos_x(valid_int-500:valid_int+500).^2)
  em_y = mean(pos_y(valid_int-500:valid_int+500).^2)

  med_x = mean(pos_x(valid_int-500:valid_int+500))
  med_y = mean(pos_y(valid_int-500:valid_int+500))

  std_x = std(pos_x(valid_int-500:valid_int+500))
  std_y = std(pos_y(valid_int-500:valid_int+500))


% Análise temporal
```

```
t_sensor = sensorinfo(:,1);

  % sensor Timestamp
  for i=2:length(t_sensor)
  tstamp_sensor(i-1) = t_sensor(i) - t_sensor(i-1);
  end

  s_periodo = mean(tstamp_sensor(find(tstamp_sensor > 0))) / 1000
  s_tmax = max(tstamp_sensor(find(tstamp_sensor > 0))) / 1000
  s_tmin = min(tstamp_sensor(find(tstamp_sensor > 0))) / 1000
  s_std = std(tstamp_sensor(find(tstamp_sensor > 0))) /1000

  tstamp_sensor=tstamp_sensor(find(tstamp_sensor > 0)) / 1000;
  for i=2:length(tstamp_sensor)
  jitter_rel(i-1) = tstamp_sensor(i) - tstamp_sensor(i-1);
  end

  jitter_abs = max(jitter_rel)

  figure(3)
  hist(jitter_rel, -70:70)
  title('Histograma do Jitter Relativo do Tempo entre Frames')
  xlabel('Jitter Relativo ( ms )')
  axis([-25 25 0 1200])

% Perda de frames

frames_ns = sensorinfo(:,2);
fr_total = frames_ns(end)-frames_ns(1);

  for i=2:length(frames_ns)
  delta_frames(i-1) = frames_ns(i) - frames_ns(i-1);
  end

  lost_frames_tot = delta_frames(find(delta_frames > 1)) -1;

  lost_frames = (sum(lost_frames_tot) / fr_total) * 100
```

Apêndice C

Circuíto do Módulo Sensor da Plataforma Bola na Calha

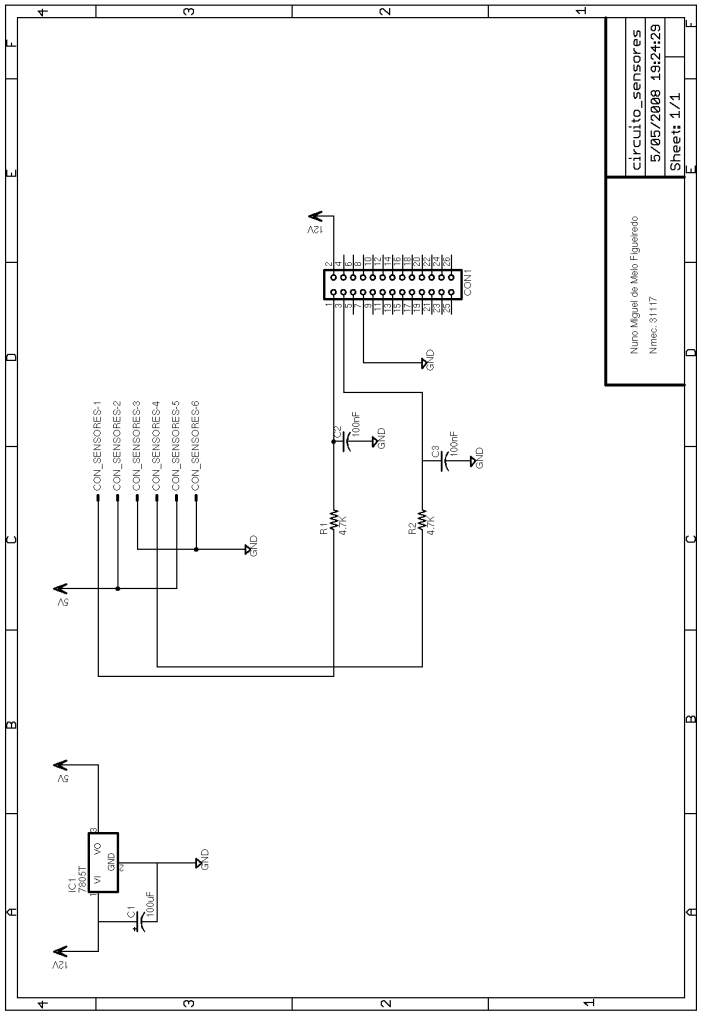


Figura C.1: Circuito do Módulo Sensor da Plataforma Bola na Calha

Parte VI

Bibliografia

Bibliografia

[ABR+93] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. Wellings. Applying new sheduling theory to static priority pre-emptive scheduling. *Software Engeneering Journal, 8(5):284-293*, September 1993.

[AFF98] Luis Almeida, Jose Fonseca, and Pedro Fonseca. Flexible time-triggered communication on a controller area network. December 98.

[APF02] Luís ALmeida, Paulo Pedreiras, and José Alberto G. Fonseca. The ftt-can protocol: Why and how. IEEE Transactions on Industrial Electronics, Vol. 49, No. 6, 2002.

[Ass97] Inc ODVA Open DeviceNet Vendor Association. *DeviceNet Specification - release 2.0, Vol. I and II.* 1997.

[BBB03] Enrico Bini, Giorgio C. Buttazzo, and Giuseppe M. Buttazzo. Rate monotonic analysis: The hyperbolic bound. *IEEE Trans. Comput.*, 52(7):933–942, 2003.

[bip] Doug Comer's book internet page. Hardware addressing on http://penguin.dcs.bbk.ac.uk/academic/networks/data-link-layer/hardware-addressing/index.php.

[BPG05] Hector Benitez-Perez and D. Fabian Garcia. *Reconfigurable Distributed Control.* Springer, 2005.

[Bro91] William L. Brogan. *Modern Control Theory.* 1991.

[But05] Giorgio C. Buttazo. *Hard Real-Time Computing Systems: Predictable Algogorithms and Applications.* Springer, 2005.

[CEN96] CENELEC. *European Committe for Electrotechnical Standardisation. European Standard EN 50170: Fieldbus: Vol. 1: P-Net; Vol. 2: PROFIBUS; Vol. 3: WorldFIP.* 1996.

[Com00] IEC International Electrotechnical Comitee. *IEC International Standard 61158: Fieldbus standard for use in industrial control systems - Type 1: Existing IEC TS61158 parts 3-6 (Foundation Fieldbus H1); Type 2: ControlNet, Type 3: PROFIBUS; Type 4:P-Net; Type 5: Fieldbus Foundation HSE; Type 6: SwiftNet; Type 7: WorldFip; Type 8: Interbus-S.* 2000.

[Con01] FlexRay Consortium. *FlexRay Requirements Specification Version 1.9.7.* September 2001.

[Dec01] J-D. Decotignie. A perspective on ethernet as a fieldbus. Proceedings of the 4th FeT'2001 - International Conference on Fieldbus Systems and their Applications, November 2001.

[DHL+05] W. DeVan, S. Hicks, G. Lawson, W. Wagner, D. Wantland, and E. Williams. Using a control system ethernet network as a field bus. 2005.

[DHS03] John Joachim D'Azzo, Constantine H. Houpins, and Stuart N. Sheldon. *Linear Control System Analysis and Design with Matlab*. CRC Press, 2003.

[dO07] Arnaldo Silva Rodrigues de Oliveira. *Especialização e Síntese de Processadores para Aplicação em Sistemas de Tempo-real*. PhD thesis, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, 2007.

[eth82] *DIX Ethernet V2.0 specification*, 1982.

[Exa] Toaster Control Example. http://www.embeddedarm.com/software/arm-netbsd-toaster.php.

[fS00] International Organization for Standardization. *ISO/WD111898-4. Road Vehicles - Control Area Network (CAN) - Part 4: Time Triggered Communication*. 2000.

[Gmb91] Robert Bosch GmbH. *CAN Specification version 2.0*. 1991.

[HP05] Diederich Hinrichsen and A. J. Pritchard. *Mathematical Systems Theory*. Springer, 2005.

[IEEa] IEEE. *IEEE 802.3 10BASE3 standard*.

[IEEb] IEEE. *IEEE 802.3 10BASE5 standard*.

[IEEc] IEEE. *IEEE 802.3c 100BASE-T standard*.

[IEEd] IEEE. *IEEE 802.3i 10BASE-T standard*.

[IEEe] IEEE. *IEEE 802.3z 1000BASE-T standard*.

[ISO93] ISO-11898. *Road Vehicles - Interhange of digital information - Controller Area Network (CAN) for high-speed communications*. 1993.

[ISO94] ISO-11519-2. *Road Vehicles - Low-speed serial data communication - Part 2: Low-speed controller area network (CAN)*. 1994.

[Kit] Elevator Control System Trainig Kits. http://statefirmcorp.com/elevator.html.

[Kop] Herman Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1 edition.

[Kop93] H. Kopetz. Should responsive systems be event-triggered or time-triggered? IEICE Trans. Inform. Syst., vol. E76-D, pp. 1325–1332, 1993.

[KVA] Advanced CAN solutions KVASER. The can protocol tour on http://www.kvaser.com/can/protocol.

[Lei04] James R. Leigh. *Control Theory.* IET, 2004.

[Lim] Googol Technology Limited. http://www.googoltech.com/web/eng/main.jsp.

[LL73] C. L. Liu and J.W. Lyland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the Association for Computing Machinery, 20(1)*, 1973.

[LM04] Pedro Leite and Ricardo Marau. Rtkpic18 – real time kernel pic18fxx8. *Revista do DETUA Vol. 4, N. 2, DETIUA*, 2004.

[MAP06] Ricardo Marau, Luís Almeida, and Paulo Pedreiras. Enhancing real-time communication over cots ethernet switches. 2006.

[NL04] Márcio Neves and Marco Leonor. Construção de plataformas para experiências de mecatrónica. Technical report, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, 2004.

[OA04] Arnaldo Oliveira and Luís Almeida. Ensaio sobre os sistemas de tempo real. *Revista do DETUA Vol. 4, N. 2, DETIUA*, 2004.

[oPape] Ball on Plate a practical example. http://web.mit.edu/ shorya/www/ms/bop.htm.

[oPle] Ball on Plate lego example. http://blog.makezine.com/archive/lego/5.html.

[PA] P. Pedreiras and L. Almeida. Approaches to enforce real-time behavior in ethernet.

[Ped] Paulo Bacelar Reis Pedreiras. *Suporte de Comunicações Tempo - Real Flexíveis em Sistemas Distribuídos.* PhD thesis, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro.

[Pim90] J. Pimentel. *Communication Networks for Manufacturing.* Prentice-Hall International Editions, 1990.

[PLA03] P. Pedreiras, R. Leite, and L. Almeida. Characterizing the real-time behavior of prioritized switched ethernet. Proceedings of 2 International Workshop on Real-Time LANs in the Internet Age (RTLIA'03), July, 1 2003.

[pri] Median principle. http://en.wikipedia.org/wiki/median.

[PWL] *ETHERNET Powerlink protocol, available at www.ethernet-powerlink.org.*

[Roq07] Tiago Roque. Construção de plataformas para experiências de mecatrónica. Technical report, Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, 2007.

[SR90] John A. Stankovic and Krithi Ramamritham. *What is Predictability for Real-Time Systems?* 1990.

[tG] Convert RGB Image to Grayscale. http://www.dfanning.com/ip_tips/color2gray.html.

[TTP99] Ttp/c protocol version 0.5. TTTech Computertechnik GmbH, 1999.

[VC98] S. Varadarajan and T. Chiueh. Ethereal: A host-transparent real-time fast ethernet switch. October 1998.

[VC94] C. Venkatramani and T. Chiueh. Supporting real-time traffic on ethernet. December 94.

[VR01] Paulo Veríssimo and Luís Rodrigues. *Distributed Systems for System Architects*. Kluwer Academic Publishers, 2001.