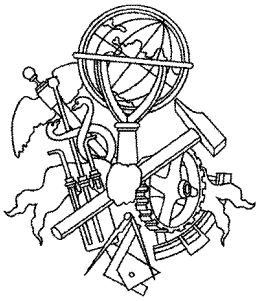
Supervisão, Comando e Controlo para Sistemas Robóticos

NUNO ALEXANDRE NETO DIAS



Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização em Sistemas Autónomos

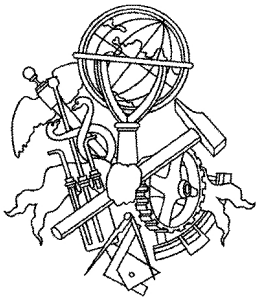
Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

Outubro de 2009

Supervisão, Comando e Controlo para Sistemas Robóticos

NUNO ALEXANDRE NETO DIAS



Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização em Sistemas Autónomos

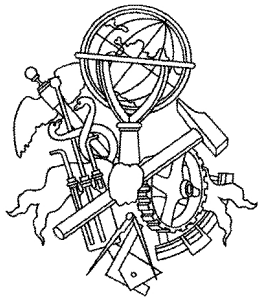
Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

Outubro de 2009

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha de Disciplina de Tese/Dissertação, do 2ºano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Nuno Alexandre Neto Dias, Nº980153, 1980153@isep.ipp.pt
Orientação Científica: Doutor Eduardo Alexandre Pereira da Silva (ISEP).



Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Sistemas Autónomos
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto

Agradecimentos

Queria agradecer em primeiro lugar ao meu orientador Eng. Eduardo Silva pela sua enorme paciência e dedicação que teve na orientação do trabalho aqui apresentado bem como pelas oportunidades e confiança que me deu.

Queria agradecer também as longas e proveitosas discussões com o Eng. Alfredo Martins e Eng. José Miguel Almeida.

A todos os meus colegas do Laboratório de Sistemas Autónomos e em especial ao André Dias, Carlos Almeida, Hugo Ferreira, Hugo Silva e Luís Lima, pela amizade e colaboração.

Queria agradecer também o carinho que recebi por parte de alguns dos meus colegas de trabalho da ESEIG/IPP.

Aos meus pais e irmã pelos sacrifícios e apoio. Apesar de estarem longe estiveram sempre presentes.

À Daniela pela companhia, paciência e incentivo.

Obrigado a todos.

Resumo

Esta dissertação aborda a temática da interacção homem-máquina no controlo, comando e supervisão de veículos autónomos. O tópico de Supervisão tem vindo a ganhar importância com o surgimento de novos e florescentes nichos de mercado para aplicação de sistemas robóticos tais como a monitorização ambiental e a monitorização de infra-estruturas, onde se destaca, a título de exemplo, a robótica submarina que tem sido impulsionada pela indústria extractiva em *off-shore* e pela indústria da construção de infraestruturas costeiras ou em *off-shore*. As funcionalidades apresentadas por estes sistemas ainda dependem fortemente da supervisão do utilizador humano. As características da supervisão, por sua vez, dependem do grau de autonomia com que estes sistemas são capazes de desempenhar as tarefas que lhes são atribuídas. As questões de supervisão já há algum tempo têm despertado em si alguma atenção por parte da comunidade robótica motivo pelo qual se assiste à transferência de projectos do seio da comunidade de "I&D"para a comunidade de desenvolvimento e serviços, onde a problemática da supervisão assume um papel fundamental para o sucesso das aplicações dos sistemas robóticos. O trabalho desenvolvido permitiu a apresentação de uma arquitectura para a supervisão de sistemas autónomos. O contributo do trabalho apresentado nesta dissertação alicerça-se numa abordagem experimental, que através da análise de casos de estudo, ocorre ao longo de duas linhas de trabalho: uma de carácter conceptual dirigida ao projecto e que resulta numa proposta de arquitectura orientada para a supervisão; e outra que resulta em linhas orientadoras das interfaces de comando homem-máquina. A motivação para este trabalho surge como resposta aos desafios colocados pelos projectos que ocorrem no LSA/ISEP/IPP que protagonizam um papel relevante na interacção homem máquina, e que são por exemplo o futebol robótico, a re-

colha de dados oceanográficos (batimetria) executada por um ASV (Veículo Autónomo de Superfície) e a localização de "tags"(dispositivos identificadores) dentro de edifícios fechados usando pontos de acesso de redes sem fio.

Complementarmente à solução conceptual para a supervisão desenvolvemos diferentes e variadas interfaces gráficas como elementos integrantes da interacção homem - sistema autónomo e que resultaram na implementação do sistema de supervisão da equipa de futebol robótico ISePorto e do sistema de batimetria com o veículo ROAZ. Os resultados obtidos, confirmaram que os paradigmas de supervisão tendo em conta a iniciativa mista, promovem soluções mais versáteis e operacionalmente mais eficazes.

Palavras Chave: Iniciativa Mista, Interacção Homem-robô, Sistemas Autónomos, Interfaces gráficas

Abstract

The main focus of this dissertation is the man-machine interaction problem in autonomous vehicles control and supervision. The Supervision topic has been gaining importance with the arise of new and growing market niches for robotic systems applications in environmental and infrastructures monitoring like submarine robotics supported by both off-shore extractive industry and off-shore and coast infrastructure construction industry. The systems functionalities are still strongly dependent on human supervision. The supervision characteristics are, in return, dependent on the autonomy degree of the systems capability of performing the designated tasks. The supervision problematic have gathered the attention of the robotics community for some time, in terms of the project transfer from the R&D community to service and development community, where supervision takes a fundamental role in robotic systems applications success. This dissertation's contribute is based in an experimental approach, by analysis of study cases, along two work lines: one, of conceptual character and project driven, results in a supervision guided architecture proposal; and the other results in man-machine command interfaces guidelines. This works' motivation comes as an answer to the challenges placed by the projects in LSA/ISEP/IPP that take a relevant part in man-machine interaction, like robotics football, ocean data recovery (bathymetry) executed by an ASV (Autonomous Surface Vehicle) and the localization of tags (identifying devices) inside closed buildings using wireless net points.

Complementary to the conceptual solution for supervision were developed several different graphical interfaces as integration elements in a human-autonomous system interaction and that result from ISePorto robotic football team supervision systems and the bathymetry systems with the vehicle ROAZ.

The results obtained have confirmed that the supervision paradigms, considering a mixed initiative, promote more versatile and operationally more effective solutions.

Keywords: Human-robot Interaction, Autonomous Systems, Mixed-Initiative, Graphical Interfaces

Conteúdo

Lista de Figuras

Acrónimos

ACK Acknowledge.

AP Access Point.

API Application Programming Interface.

ASV Autonomous Surface Vehicle.

CAN Controler Area Network.

CTD Conductivity, Temperature and Depth.

GPS Global Positioning System.

GUI Graphical User Interface.

HMI Human-Machine Interaction.

HRI Human-Robot Interaction.

IP Internet Protocol.

IPP Instituto Politécnico do Porto.

ISEP Instituto Superior de Engenharia do Porto.

MSL Middle Size League.

LGPL Lesser General Public License.

LSA Laboratório de Sistemas Autónomos

PDA Personal Digital Assistant.

RTOS Real Time Operating System.

RTPS Real Time Publish Subscribe.

ROV Remote Operated Vehicle.

TCP Transmission Control Protocol.

UDP User Datagram Protocol.

UGV Unmanned Ground Vehicle.

Capítulo 1

Introdução

Conteúdo

1.1 Problema

A utilização de robôs em cenários não estruturados tem vindo a crescer significativamente nos últimos anos [16, 17, 18]. Este crescimento, deve-se não só pelo aumento da perícia, do desempenho e da autonomia dos robôs mas também pela emergência de metodologias e técnicas que permitem a utilização de equipas de robôs em cenários de operação mais complexos. Contudo, grande parte do desenvolvimento de sistemas robóticos com elevado grau de autonomia ainda ocorre (domínio de aplicações limitado) no seio de programas de investigação e desenvolvimento [6, 17, 18, 19, 20] ou no seio da indústria orientada ao espaço e à segurança [16, 21, 22, 23]. Tem-se assistido nos últimos anos ao emergir de novos e florescentes nichos de mercado para a aplicação destes sistemas robóticos como são na monitorização ambiental [24] e a monitorização de infra-estruturas, onde se destaca a título de exemplo, a robótica submarina [25] impulsionada pela indústria extractiva em *off-shore* e pela indústria da

construção de infra-estruturas costeiras ou em *off-shore*.

As funcionalidades apresentadas por estes sistemas, ainda dependem fortemente da supervisão do utilizador humano [16, 17, 18, 22]. As características da supervisão por sua vez dependem do grau de autonomia com que estes sistemas são capazes de desempenhar as tarefas que lhe foram atribuídas. Consequentemente, surge na primeira linha de pensamento a seguinte questão: como deve ser a interacção de um utilizador com um sistema multi-robótico para que o elemento humano o possa supervisionar, controlar e comandar de modo a que as decisões tomadas sejam as mais adequadas? Torna-se então necessário, entre outros, formalizar a especificação das missões, a supervisão de todo o sistema e ainda a forma de intervenção durante a execução de uma missão.

Na Figura 1.1 pode observar-se um diagrama que ilustra as possíveis interacções entre o homem e os agentes[1] e estes entre si, e as possíveis acções de supervisão.

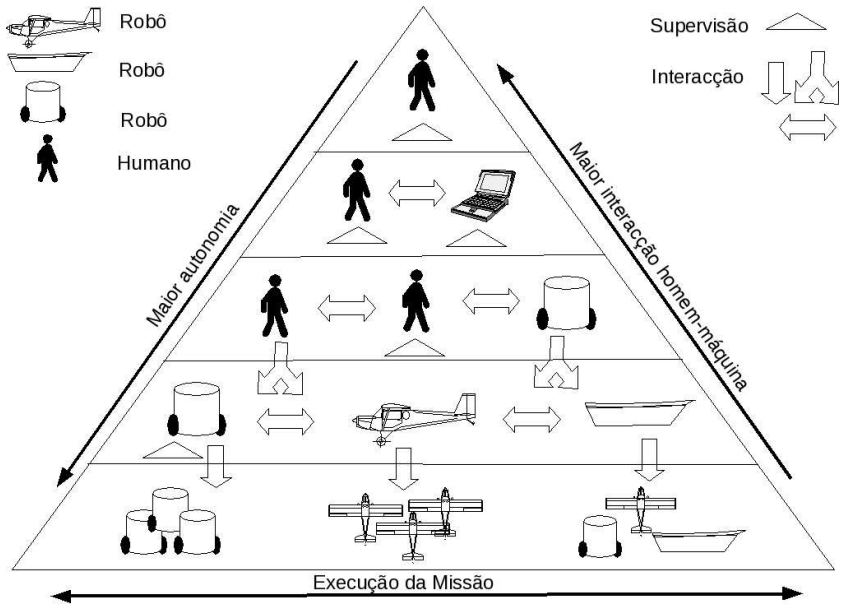


Figura 1.1: Diferentes níveis de interacção entre o homem e as máquinas

A interacção entre o humano e os sistemas robóticos pode ocorrer em diversas

[1] Um agente é um elemento não humano que integra um sistema robótico podendo ser uma peça de software ou mesmo um robô em si e que desempenha autonomamente determinadas tarefas.

fases que aqui sumariamente definimos: no "desenvolvimento do sistema"; no *setup* do sistema; e/ou durante a sua execução.

Na fase de "desenvolvimento do sistema", esta interacção ocorre quase sempre através de especialistas, e geralmente as questões que a comunidade aborda, são ao nível das ferramentas de desenvolvimento, das ferramentas de simulação como auxiliares à implementação, das ferramentas de especificação como auxiliares no *design* e concepção de comportamentos e funcionalidades e mais recentemente, auxiliares de implementação baseado em *case tools*.

Na fase de *setup* geralmente são utilizadas ferramentas de detecção de falhas e diagnóstico [26, 27, 28] e ferramentas de especificação de missões e comportamentos [29, 30, 31, 32] que permitem ao utilizador/operador preparar o sistema para a funcionalidade projectada.

Para a fase em que o sistema está em execução, geralmente são utilizadas ferramentas que permitem vários tipos de interacção que vão desde a supervisão do sistema [33], passando pela teleprogramação [22] até uma interacção mais "directa"como é a teleoperação [21].

O crescente aumento dos domínios de aplicação e o continuado enriquecimento funcional dos sistemas robóticos tem conduzido esta área para um espectro de aplicações em que o utilizador humano ocupa simultaneamente o papel de um agente do sistema e o papel de supervisor, denominando-se este tipo de aplicações de **iniciativa mista** [34, 35, 36, 37, 38, 39, 40].

As questões de supervisão já há algum tempo têm atraído em si alguma atenção por parte da comunidade robótica em função da transferência de projectos do seio da comunidade de "I&D"para a comunidade de desenvolvimento e serviços, onde a problemática da supervisão assume um papel fundamental para o sucesso das aplicações dos sistemas robóticos. O contributo desta dissertação alicerça-se numa abordagem experimental, que através da análise de casos de estudo, ocorre ao longo de duas linhas de trabalho: uma de carácter conceptual dirigida ao projecto e que resulta numa proposta de arquitectura orientada para a supervisão; e outra que resulta em linhas orientadoras das interfaces de comando homem-máquina, conforma se pode observar pela Figura 1.2.

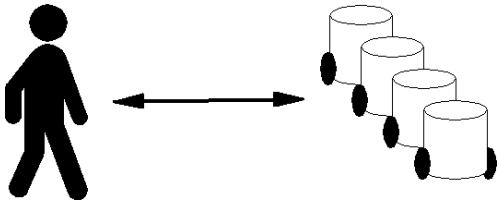


Figura 1.2: A interface homem robô discutida nesta tese

1.2 Motivação e enquadramento

Desde o ano 2000 que o Laboratório de Sistemas Autónomos (LSA) do ISEP/IPP (Instituto Superior de Engenharia do Porto - Instituto Politécnico doPorto) tem vindo a desenvolver vários sistemas robóticos, contribuindo para um conjunto de soluções aplicado a um variado elenco de cenários tais como, por exemplo, o caso do futebol robótico, a monitorização ambiental ou vigilância marítima.

A consolidação da experiência tem ocorrido no desenvolvimento de projectos aplicados aos cenários anteriormente referenciados permitiram identificar um conjunto de problemas que urge resolver no sentido de facilitar e promover o uso de sistemas multi-robóticos. Uma das questões mais importantes surge na chamada interacção homem-máquina. Esta interacção só por si lança desafios ao nível da optimização das respectivas interfaces mas, de uma forma mais vasta, ao nível da arquitectura organizacional e funcional destes sistemas.

A motivação para este trabalho surge como resposta aos desafios colocados pelos projectos que ocorrem no LSA/ISEP/IPP que protagonizam um papel relevante na interacção homem máquina, e que são os seguintes: o futebol robótico, onde uma equipa com vários robôs terrestres de forma coordenada jogam futebol; a recolha de dados oceanográficos (batimetria) executada por um ASV (Veículo autónomo de superfície); a detecção precoce de fogos florestais com recurso a um ou mais UAV; e a localização de "tags"(dispositivos identificadores) dentro de edifícios fechados usando pontos de acesso de redes sem fio.

O futebol robótico da Middle Size League (MSL) promovido pela ROBOCUP [20] é um cenário onde um grupo de robôs joga um jogo de futebol de forma autónoma tendo as regras com as quais se regem sido adaptadas previamente. O ISePorto [41] tem vindo a participar ao longo dos últimos anos no campeonato da MSL. Apesar da equipa ser autónoma durante o jogo, a para-

metrização de alguns elementos, tal como o número de jogadores, as tácticas iniciais e a aplicação das regras de jogo são sempre com recurso a humanos. Do mesmo modo a supervisão dos dados que permitem caracterizar o estado da equipa durante o jogo é também realizado por um operador humano. Há também ainda a referir que durante o desenvolvimento dos algoritmos do jogador é necessária a existência de ferramentas que permitam que o humano aceda à malha de controlo, que efectue manobras usando a teleoperação e visualize algumas variáveis internas da aplicação do robô.

A recolha de dados oceanográficos é realizada pelo ASV ROAZ [42] é feita de forma autónoma onde, através de uma consola de operações, um operador define a missão a ser executada e envia-a para o ASV através do *link* de comunicações existente. No processo de recolha de dados com ASV é necessário começar por definir uma área de trabalho e o tipo de dados que serão objecto de estudo/recolha. Este cenário de operação é definido no início da missão mas pode ser estendido dinamicamente caso haja essa necessidade. Durante uma recolha de dados típica é usada a manobra do tipo 'corta-relva' onde se torna necessário definir as distâncias e o raio de cada curvatura entre cada perfil. O supervisor da aplicação, durante uma missão, tem necessidade de verificar o estado actual da missão, alterá-la em tempo real e verificar se os módulos necessários à conclusão da missão permanecem a funcionar correctamente.

O sistema "Wireless Positioning System"consiste numa plataforma móvel para efectuar mapas de potências de sinais de *access points* WiFi, para posterior utilização na localização e posicionamento de pessoas através de "tags"ou computador de bolso. Na localização de "tags"em edifícios fechados, numa fase inicial, torna-se necessário criar mapas de potências dos *access points* disponíveis. Para tal é fundamental a existência de um mecanismo que permita que um operador faça o registo dessas potências para posterior utilização.

1.3 Objectivos e contribuições

O objectivo deste trabalho é o de fornecer directivas que permitam a construção de equipas de robôs, no seus diversos componentes. Consequentemente, pretende-se contribuir para o desenvolvimento de um modelo de referência que

permita abordar de forma uniformizada e sistemática a interacção homem-sistema autónomo. Desta forma pretende-se que este trabalho dê alguns contributos tais como:

1. Avaliação de diferentes casos de estudo com o objectivo de sistematizar acções de supervisão;
2. Definição de uma arquitectura de interacção homem-robô que abranja diferentes níveis de autonomia;
3. Propor soluções de supervisão de sistemas robóticos;
4. Desenvolver a interacção homem-sistema autónomo.

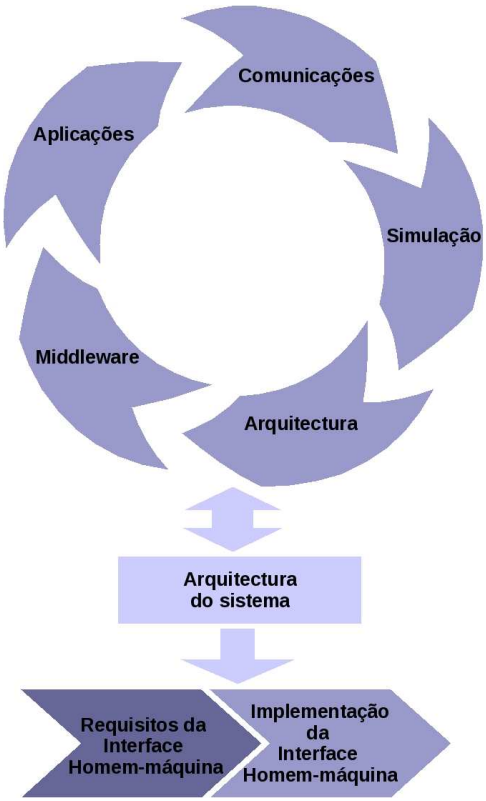


Figura 1.3: Abordagem à interacção homem-sistema autónomo

A metodologia para síntese e definição do modelo referido parte da análise dos diferentes estágios de desenvolvimento, bem como dos elementos estruturantes de um sistema robótico e do qual resultará uma arquitectura para o sistema.

Após esta fase resultaram também orientações para a definição de requisitos e estratégias para o modelo de interface homem-sistema autónomo, tal como se pode observar na Figura 1.3.

1.4 Organização da tese

Esta dissertação está organizada da seguinte forma:

- No capítulo 2 começa-se por descrever as arquitecturas que permitem desenvolver sistemas multi-robóticos que envolvem equipas de robôs homogéneos ou heterogéneos, detalhando ainda os blocos principais de um ambiente deste género. São ainda referidos alguns pontos relativos à iniciativa mista.

- No capítulo 3 são apresentados três casos de estudo diferentes sobre os quais recairá a parte prática desenvolvida.

- No capítulo 4 são discutidos alguns aspectos de implementação de interfaces de supervisão, onde é definida uma arquitectura de implementação, bem como as directivas principais no desenvolvimento de aplicações para Iniciativa Mista.

- No capítulo 5 são apresentados os desenvolvimentos práticos da interface de supervisão bem como muitos detalhes de implementação.

- O capítulo 6 é constituído por alguns comentários, bem como algumas sugestões para melhorias futuras do trabalho aqui apresentado.

(Esta página foi intencionalmente deixada em branco.)

Capítulo 2

Estado da arte

Conteúdo

2.1 Introdução

Ainda está longe o cenário que permita através de uma especificação formal de objectivos e propriedades (físicas, lógicas, operativas e temporais), obter de forma automática o projecto de um sistema autónomo que seja solução de uma determinada especificação. Contudo, enquanto não se estabelecem essas metodologias, existe um grande conjunto de "processos rotineiros"que ajudam na realização de uma forma não automática o projecto de sistemas autónomos. O largo conjunto de tópicos e problemas ainda em aberto (exemplo: síntese de controladores, verificação, navegação, sensores, actuadores, arquitecturas computacionais, RTOS, etc.) associados a este processo construtivo, é ainda um obstáculo para que a vertente "de como fazer"apareça como uma abordagem integrada, surgindo quase sempre limitada ao tópico teórico ou tecnológico objecto de investigação. No entanto, os grupos de investigação têm vindo a focar

a questão do desenvolvimento do projecto quase através da descrição da organização proposta para as relações dos componentes através do que normalmente se identifica como a arquitectura [1, 2, 3, 4, 5, 6].
Já existem esforços para o desenvolvimento de metodologias formais que orientam a possível aplicação da verificação formal e da síntese automática de controladores do comportamento. Contudo, estes esforços na verificação e na síntese automática têm-se limitado a desenvolvimentos em níveis de grande abstracção e geralmente aplicam-se a abordagens conceptuais, não sendo ainda o seu uso generalizado a sistemas de elevada complexidade e heterogeneidade. Com o ritmo crescente de inovação tecnológica tem-se assistido à multiplicação de projectos de desenvolvimento de sistemas autónomos e, consequentemente, à tentativa de automatizar o projecto, ou, numa perspectiva intermédia, definir ferramentas que suportam o projecto e promovam a síntese automática de alguns componentes. Exemplo muito concreto foi o esforço levado a cabo pelo projecto "Smart Vehicle"[18] que teve por objectivo o desenvolvimento de uma metodologia que integra ferramentas que auxiliam o projecto de sistemas ligados à indústria da automação.O "Smart Vehicle"integra tecnologia a desenvolvida no projecto MoBIES [43]. O ambiente que o MoBIES pretende proporcionar é caracterizado por: modelização e simulação de sistemas híbridos, análise e síntese de sistemas de controlo, prototipagem de controladores, geração automática de código, verificação de código via simulação e verificação de componentes electrónicos via *"Hardware-in-the-Loop"*. O grande desafio do projecto "Smart Vehicle"será o de proporcionar às equipas de projecto um conjunto de ferramentas que permita melhorar performances do projecto em parâmetros como a qualidade e o *"time-to-market"*, bem como uma abordagem de "modelo único"nas múltiplas vistas. Este modo, permite aos engenheiros das diferentes disciplinas trabalhar no seu domínio específico e, simultaneamente, manter a consistência e a eficiência de uma abordagem de modelo único. Outro aspecto fundamental deste projecto consiste na utilização de modelos dos componentes no desenvolvimento de modelos dos sistemas, facilitando desta forma a reutilização de código nas diversas fases de projecto.

Como facilmente se constata não é ainda fácil distinguir de forma clara a abrangência de cada um dos tópicos de investigação associados à robótica e

identificar a respectiva contribuição para a supervisão de sistemas autónomos. A nossa abordagem ao estado da arte realizar-se-á essencialmente na análise das arquitecturas e da forma de como estas endereçam a interacção homem-máquina.

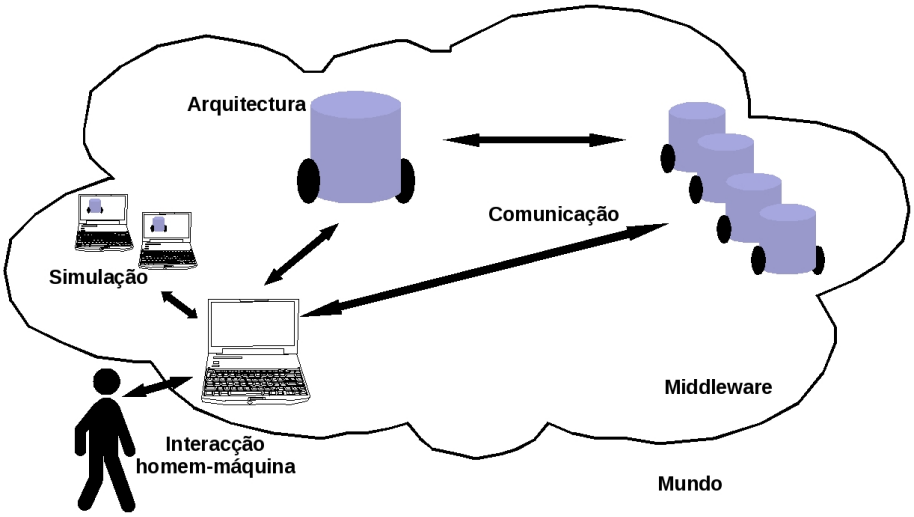


Figura 2.1: Principais tópicos de desenvolvimento em sistemas multi-robóticos

A descrição destas arquitecturas, bem como o trabalho associado, tem sempre em vista todos os elementos que possam contribuir para a definição mais clara e concisa de como poderá ser construída uma interface de supervisão para vários veículos autónomos, cujo objectivo principal seja o desempenho com sucesso de uma determinada missão. A Figura 2.1 apresenta um esboço dos campos que têm sido mais abordados.

2.2 Arquitecturas

A sequência das abordagens aqui descritas visam apenas tipificar os diferentes paradigmas existentes e a forma como estas são utilizadas. Algumas das abordagens à organização de sistemas autónomos parecem possuir características muitos distinta. Contudo, estas aparentes diferenças resultam mais da perspectiva de quem as descreve do que da sua respectiva essência.

A arquitectura subsumption

Subsumption [1] é, no que respeita a arquitecturas de robôs, uma das mais conhecidas e aquela que marcou um ponto de viragem na definição de arquitecturas da robótica actual. Sendo uma arquitectura definida por diferentes níveis é caracterizada por ser reactiva. O seu autor, Rodney Brooks, começa por alterar e decompor de uma forma vertical o modelo até então usado, Figura 2.2.

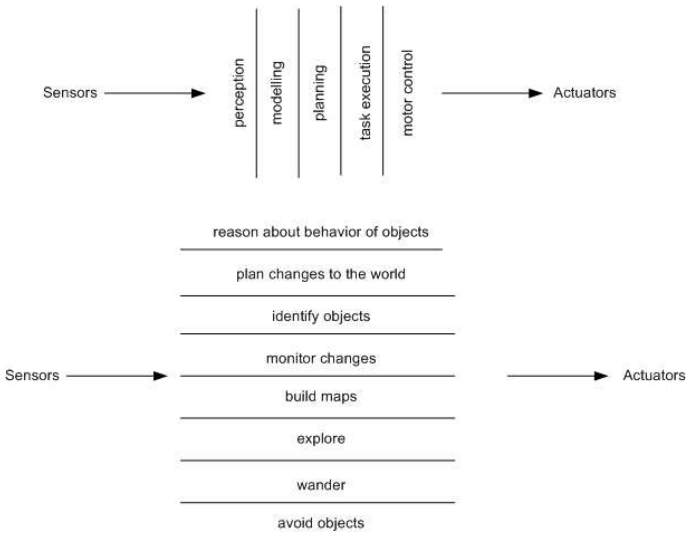


Figura 2.2: Forma vertical aplicada na robótica até então [1].

Ao fazer esta alterações o autor ficou com uma arquitectura com diferentes níveis em que, cada vez que se vai subindo de nível, as funções associadas ao mesmo se tornam mais abstractas, Figura 2.3. A mais valia desta arquitectura é a de poder introduzir outros níveis de competência, em que cada nível de competência funciona independentemente do nível que lhe está acima, mas necessita do nível que está imediatamente abaixo para funcionar correctamente.

Na documentação conhecida desta arquitectura não se conhecem referências ao supervisor/operador provavelmente porque quando surgiu esta arquitectura, em 1986, ainda não se dava muita ênfase ao humano como supervisor do sistema robótico.

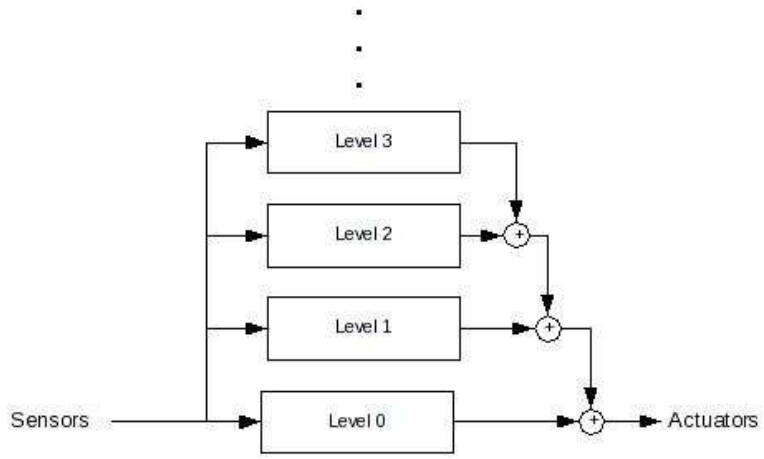


Figura 2.3: Modelo idealizado em subsumption por Brooks [1].

A arquitectura DAMN

A arquitectura DAMN [2] funciona de forma eleitoral, todos os comportamentos distribuídos têm votos, quer contra quer a favor, para que se execute uma determinada acção, fazendo desta arquitectura uma arquitectura não hierárquica. Estes votos são enviados para um árbitro central.

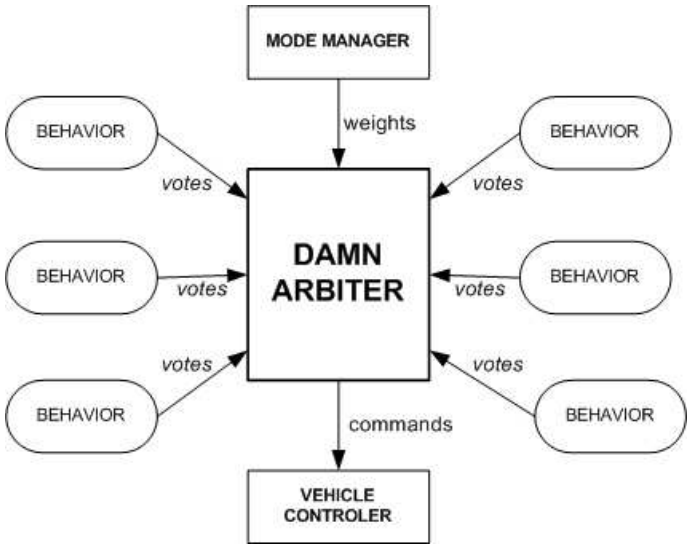


Figura 2.4: Arquitectura DAMN [2].

O DAMN ARBITER, comforme é esquematizado na Figura2.4, é responsável

por combinar todos os votos dos diferentes comportamentos sendo o resultado é enviado ao controlador do veículo. O árbitro consegue determinar as necessidades principais do veículo através dos pesos que são atribuídos aos diferentes comportamentos. Por exemplo, as funções responsáveis pela detecção e desvio de obstáculos deverão ter um peso maior do que qualquer outra tarefa. Os autores desta arquitectura não referem nem relacionam os comportamentos do robô face à interacção homem-máquina não estando a arquitectura pensada para poder ser usada em sistemas robóticos que necessitem de supervisão. Eventualmente só através dos "Mode Manager"é que um supervisor poderia actuar na configuração dos pesos que são atribuídos aos comportamentos.

A arquitectura desenvolvida no LAAS

A arquitectura desenvolvida no LAAS [3] é uma arquitectura hierárquica com três níveis distintos. Temos o nível funcional, o nível de execução e o nível de decisão. Esta arquitectura é provida de capacidades de planeamento e capacidades de reacção. O esquema da arquitectura LAAS tem os seguintes níveis:

- O nível funcional que é o responsável por toda a parte de acção e percepção do robô. Acção, porque é daqui que, por exemplo, partem os comandos para os motores. Percepção porque é a partir daqui que é feito o modelo do mundo através da informação que é fornecida pelos sensores.
- O nível de execução que controla e coordena a execução das funções de acordo com os comandos que recebe sobre as tarefas que são necessárias efectuar. Estas tarefas são geradas pelo nível de decisão.
- O nível de decisão que é responsável pelo planeamento e monitorização das funções que são necessárias executar para o cumprimento da missão. Este nível está sempre atento aos eventos que possam surgir dos outros níveis.

Como se pode observar na Figura 2.5 o operador já é tido em conta no processo de supervisão. Neste caso, o operador é responsável pela definição da missão e pela supervisão dos dados fornecidos pelas peças de software. Esta abordagem ainda não menciona como é que o operador poderia interagir com uma equipa de veículos .

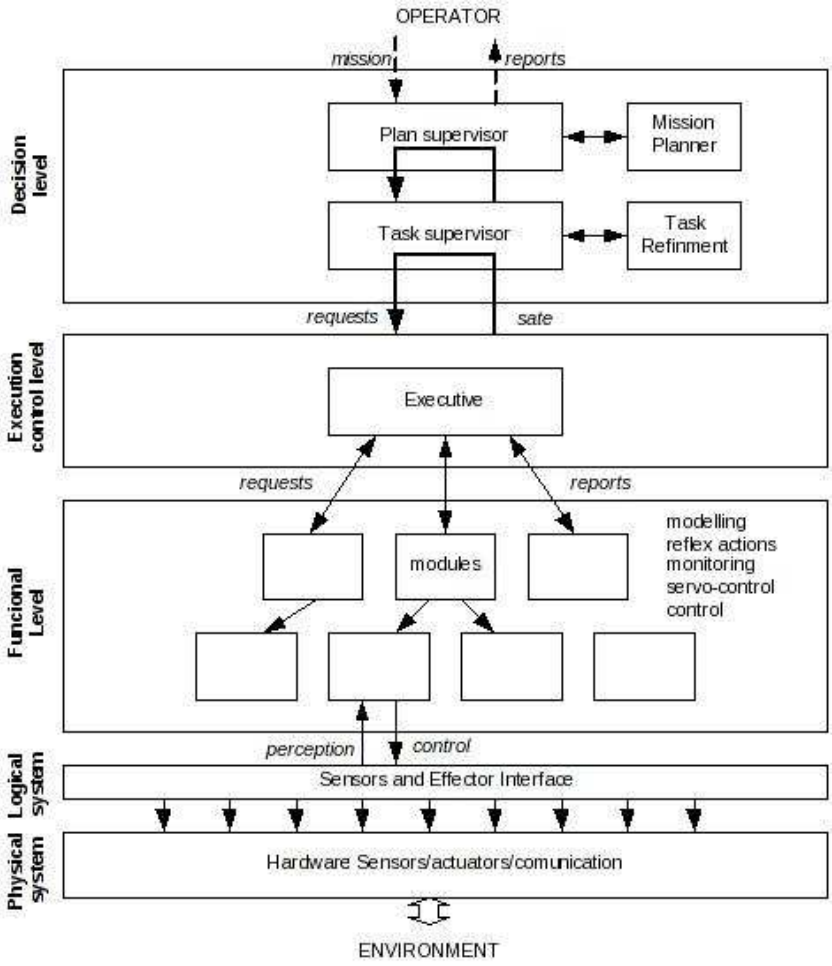


Figura 2.5: Arquitectura desenvolvida no LAAS [3].

A layered arquitecture for coordination of mobile robots (LACMR)

Esta arquitectura [4] divide o controlo dos robôs em três camadas, o planeamento, a execução e os comportamentos. A camada planeamento é responsável pelas decisões de modo a que se cumpra a missão. A camada de execução é responsável pela execução e monitorização das tarefas necessárias para cumprir o que foi decidido pelo planeamento. A camada comportamento, neste caso, funciona como a ligação entre o *software* e o *hardware*.

Esta arquitectura faz referência explícita ao cenário dos múltiplos robôs. Na Figura 2.6 podemos observar que a informação pode ser trocada entre os vários níveis, e entre robôs. A coordenação entre os vários robôs é feita através da co-

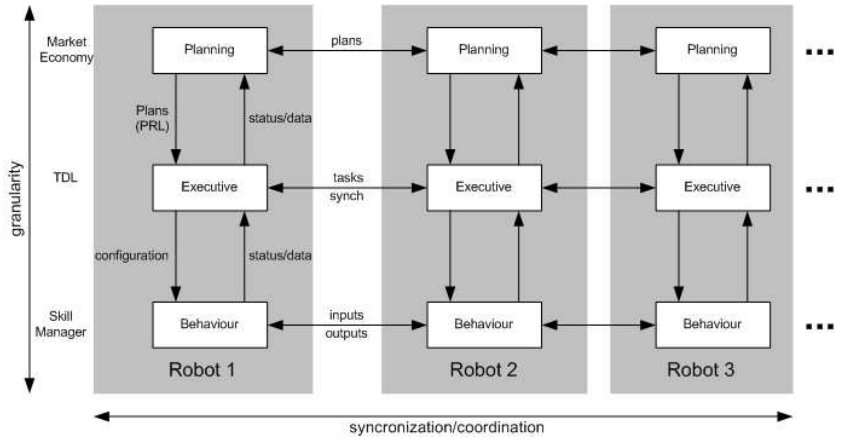


Figura 2.6: *A Layered Arquitecture for Coordination of Mobile Robots* [4].

municação entre os mesmos níveis de diferentes robôs. No caso da comunicação entre os módulos de planeamento dos diferentes robôs temos a coordenação entre toda a missão e aquilo que cada robô tem de executar. A decisão da divisão das tarefas é realizada entre todos os robôs, é aquele que é mais capaz que a executa. A execução necessita da comunicação com os outros robôs por motivos de sincronização. Enquanto que o nível anterior funciona como nível de decisão entre robôs, este funciona como sincronização das tarefas. O comportamento permite a ligação dos sensores de um robô aos actuadores de outro e vice-versa. Esta estratégia pode funcionar como um *master-slave* no caso de robôs homogéneos.

Na documentação não se encontram referências à supervisão, sendo que o operador apenas pode definir a missão a executar pelos múltiplos veículos.

A arquitectura ALLIANCE

A arquitectura ALLIANCE [5] baseia o seu desempenho na atribuição aos agentes que compõem, os atributos impaciência e a aquiescência de modo a que a selecção das acções se torne tolerante a falhas. A impaciência e aquiescência destes agentes aumentam ou passam a nulas consoante as tarefas que existam para serem realizadas. A comunicação é importante mas não é fundamental. A aquiescência pode fazer com que um agente desista de realizar uma determinada tarefa e a aquiescência faz com que um agente acabe a tarefa que não foi concluída pelo outro agente. Os robôs têm motivações, que têm como *input* os dois

estados internos já referidos (aquiescência e impaciência) e mais dois factores externos que são a comunicação e os seus sensores.

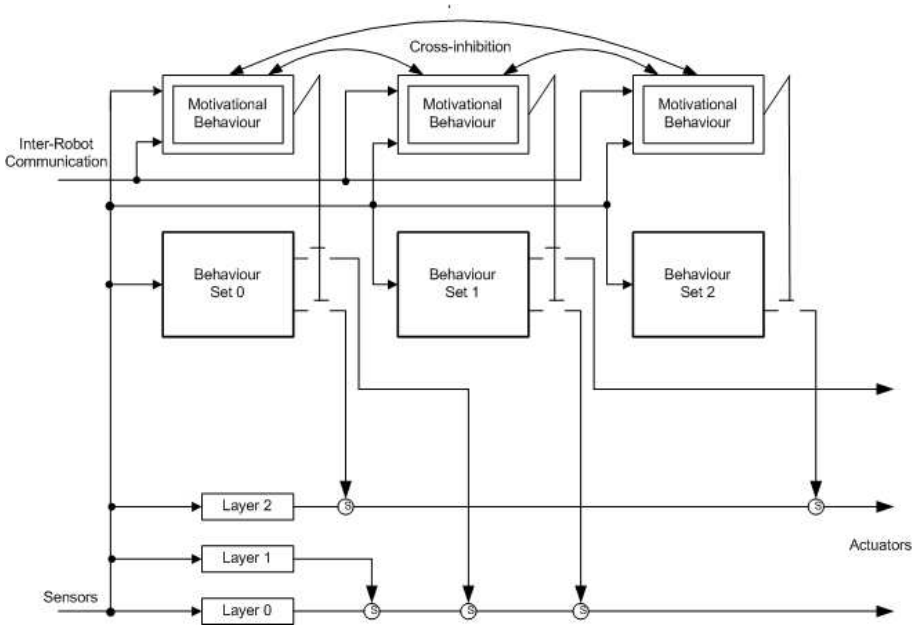


Figura 2.7: Arquitectura Alliance [5].

Esta arquitectura, Figura 2.7, é orientada ao comportamento não tendo portanto em atenção o factor humano. Os sistemas que usam esta arquitectura têm que ter uma autonomia muito elevada. Pode dizer-se que a supervisão já é realizada pelos robôs através da aquiescência e da impaciência.

A arquitectura CAMPOUT

A arquitectura CAMPOUT [6] é baseada na ALLIANCE e define várias camadas para o comportamento dos robôs.

Esta arquitectura, ao dividir os comportamentos em diferentes níveis, faz com que o controlo de alto nível apenas se preocupe em tomar decisões para a finalização da missão e os outros níveis se encarreguem por exemplo da navegação segura ou da comunicação.

O nível de comportamentos primários ou básicos, é onde estão armazenados todos os comportamentos que o robô em causa consegue executar. Os comportamentos compostos, não são mais do que a junção de vários comportamentos

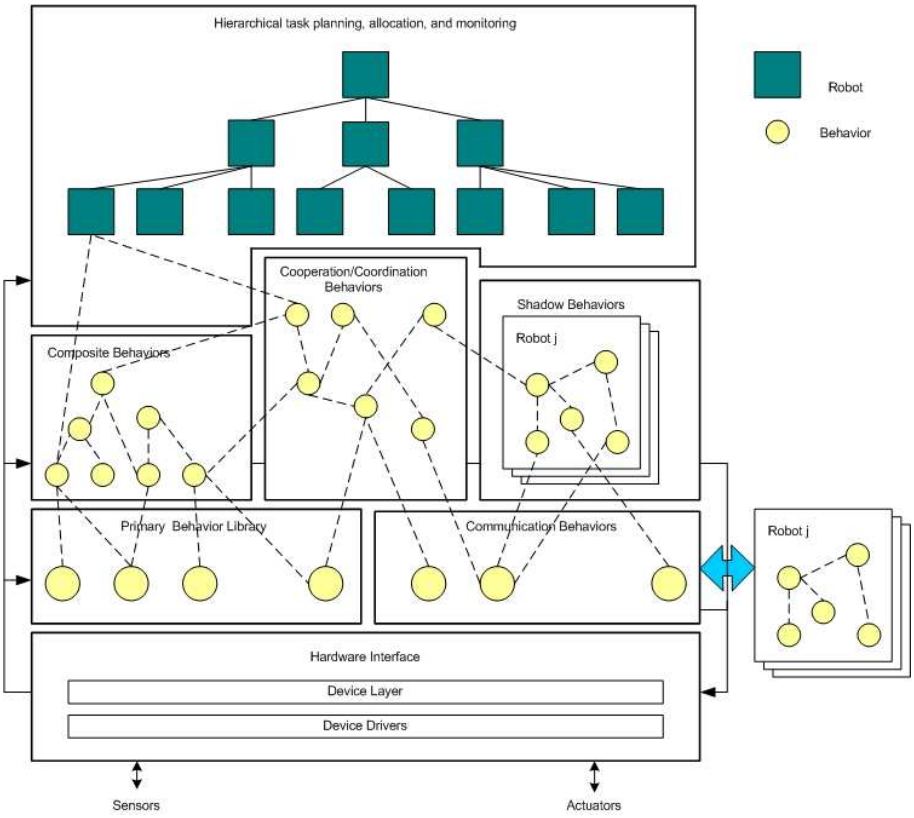


Figura 2.8: Arquitectura CAMPOUT desenvolvida no JPL - Jet Propulsion Laboratory [6] .

básicos para a execução de uma manobra mais complexa. A comunicação serve para que possa existir uma interacção entre os múltiplos agentes. Os comportamentos escondidos são uma espécie de base de dados de informações sobre os comportamentos dos outros robôs, apesar de serem tratados como comportamentos normais. Esta base de dados pode requerer comunicação entre eles.

A cooperação e coordenação são realizadas através dos comportamentos básicos, compostos e os escondidos. Esta arquitectura funciona como se de um robô apenas se tratasse. Os comportamentos de cada robô são considerados por todos os outros como se fossem eles que os estivessem a fazer através do uso dos comportamentos escondidos.

A arquitectura CAMPOUT é orientada à supervisão de múltiplos veículos (Figura 2.8). Esta arquitectura foi desenvolvida com o intuito de ser usada na

exploração espacial, onde um grupo de veículos executam missões com algum grau de autonomia e interacção entre eles, sendo sempre supervisionados por um ou mais operadores a partir de uma estação base em terra. As missões a executar são definidas sempre através de um operador.

2.3 Interacção homem-robô

Ao longo do (pouco) tempo de estudo e desenvolvimento de controlo de múltiplos robôs, cedo se percebeu que uma arquitectura hierárquica podia prover bons resultados. As arquitecturas hierárquicas por norma dividem determinadas partes do controlo em diferentes níveis onde cada nível é responsável por determinadas tarefas habitualmente distintas umas das outras. As arquitecturas não hierárquicas não são tão usadas mas também existem, como o caso da DAMN cuja aplicação prática também surtiu bons resultados.

Outro tópico de discussão é o facto de se descentralizar o controlo dos agentes. A descentralização dá ao sistema uma boa redundância para o caso de falhas, no entanto torna complicado o controlo e coordenação dos agentes, tornando-se assim necessária a comunicação, apesar de existirem arquitecturas, como a ALLIANCE [5], que prevêem soluções para o caso de um controlo descentralizado e falhas nas comunicações. A centralização do controlo pode ser, no entanto, fundamental para que este responda bem. Não é necessário que cada agente seja responsável por decidir as suas acções, podendo este agente responder perante um coordenador superior que, por sua vez pode estar dependente de outro de uma forma hierárquica e assim sucessivamente. Este tipo de arquitectura permite que no topo da hierarquia esteja apenas um agente, embora seja possível ter vários agentes no topo. Este tipo de controlo é muito usado em formação de aviões.

Como esta área de sistemas distribuídos ainda está numa fase embrionária, não se pode ainda afirmar qual das formas é melhor que a outra, já que tudo depende da aplicação em causa. A formação de aviões é um bom exemplo disso já que uma autonomia total de um dos aviões pode levar à catástrofe. No entanto outras aplicações existem em que um agente completamente autónomo possa tomar as suas próprias decisões.

Um dos factores a definir pela arquitectura consiste em permitir o controlo de agentes apenas homogéneos ou não. A homogeneidade de uma equipa pode levar a resultados melhores em termos de controlo já que cada agente sabe perfeitamente como funciona o outro podendo por isso gerar comandos para serem executados pelos outros elementos de equipa. Por exemplo, se tivermos dois agentes iguais a realizar uma tarefa conjunta, podemos ter apenas um destes agentes a efectuar o controlo enquanto que o outro fica apenas a executar os comandos decididos. Sistemas deste tipo, que fazem o controlo do tipo *master-slave*, podem permitir uma óptima coordenação. O controlo de uma equipa de robôs homogéneos é realizada consoante as tarefas que são necessárias implementar na altura, as *core skills* de cada robô são exactamente iguais pelo que só depende da disponibilidade de cada agente. No caso dos robôs heterogéneos, as tarefas podem agora ser atribuídas consoante a capacidade do robô. Em suma o que uns robôs podem fazer podem outros não o conseguir. Esta diferença de controlo pode gerar dificuldade no caso de se ter um controlo descentralizado, já que cada robô tem que saber o que é que os outros são capazes de lhe oferecer. Arquitecturas como o caso da ALLIANCE [5] permitem o controlo de robôs heterogéneos.

Quando se pretende desenvolver uma arquitectura para controlo de sistemas distribuídos não se pode deixar o problema da interacção homem/máquina de parte. Com efeito o risco que se corre é o de ter que se alterar alguns dos conceitos de base dessa arquitectura mais tarde para que se consiga esta interacção. Quando se fala em sistemas autónomos, podemos afirmar que o sistema deveria ser completamente autónomos. No entanto, numa fase inicial, é necessário que o homem tenha controlo sobre todo o sistema, por exemplo durante a respectiva fase de desenvolvimento. Poderemos ainda pretender que as missões a executar sejam entre humanos e robôs, o que implica que tenha de existir uma interface entre eles. Actualmente os agentes ainda não são completamente autónomos quanto à definição de missões (é preciso que lhes digam o que é preciso fazer para que o façam), havendo arquitecturas que necessitam que se introduzam manualmente determinados parâmetros. Existem no entanto algumas arquitecturas que usam sistemas dos mais variados tipos para que o robô aprenda com

experiências anteriores. Assim o *learning* usado pela máquina pode em alguns casos substituir o homem neste tipo de alterações.

O problema que se levanta é o da interacção ao nível da coordenação entre a máquina e o homem, ou seja, como se coordena uma equipa formada por máquinas e humanos para executarem uma determinada tarefa. A Figura 2.9 mostra o anel de controlo em que se pode observar onde é que entra a interface e respectiva interacção do homem bem como a malha de realimentação dos robôs [44].

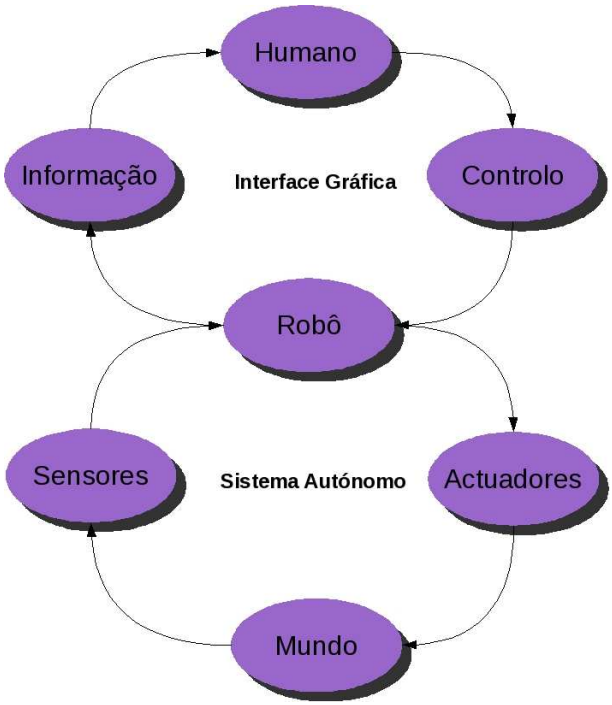


Figura 2.9: Interacção homem-robô. Adaptado de [7].

2.4 Interfaces de comando e controlo

A interacção homem-máquina ainda está longe se se tornar igual à interacção homem-homem. Existem já algumas características na interacção homem-máquina, tais como o processamento e sintetização da voz ou reconhecimento facial, que se assemelham à relação entre humanos mas ainda numa fase muito embrionária.

As interfaces de comando e controlo são produto do século XX, têm sido

usadas essencialmente para a teleróbotica, e habitualmente são dedicadas apenas a um veículo cujo grau de autonomia é nulo. Podemos encontrar estes veículos radio comandados em UAV tais como o Predator, em veículos terrestres (UGV) para inspecção de locais remotos ou catástrofe tais como o Lunokhod [21], ou em veículos subaquáticos tais como os ROV's [45]. A interacção típica entre o utilizador e o veículo pode ser facilmente esquematizada tal como se encontra na Figura 2.10.

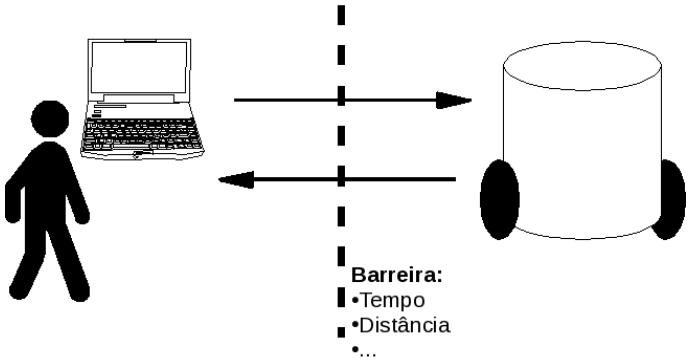


Figura 2.10: Arquitectura genérica da teleoperação de um veículo [8].

Apenas recentemente se começaram a tornar interfaces de supervisão para deixarem a parte do controlo do lado do veículo. Em termos genéricos o conceito mais comum da interacção homem-máquina rege-se conforme o esquema da Figura 2.11 [8].

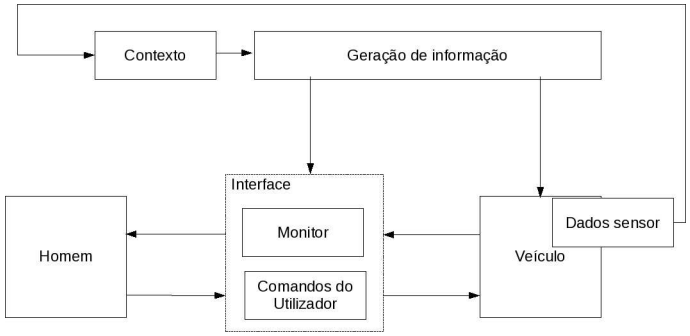


Figura 2.11: Arquitectura genérica da interacção homem-máquina. Adaptado de [8].

Existe ainda outro tipo de interfaces que não pressupõem o uso de um GUI

para controlo, pelo menos de forma directa. Os comandos podem ser realizados usando linguagem gestual, voz ou outro tipo de sinais gerados pelo corpo humano. Este tipo de interfaces ainda são muito limitadas quer no desenvolvimento quer no seu uso. Não é verdadeiramente possível usar, por exemplo, um comando de voz quando se usa um veículo num cenário de catástrofe. Quando se executa uma operação remota, em que o veículo deixa de estar em linha de vista, o *feedback* será através um computador (ou outra forma idêntica), isto se quisermos ter imagem visual do que se passa à distância, ou mesmo se quisermos saber algo sobre o estado interno no robô. Este tipo de forma de interacção pode ser útil numa fase inicial unicamente como apoio ao já existente, uso de interface gráfica, mas não será, por enquanto, crucial no comando e controlo de um qualquer equipa de robôs.

2.5 Iniciativa mista

Na Iniciativa mista o homem está sempre no ciclo de controlo, podemos considerar iniciativa mista tudo o que vai desde a simples teleoperação até à supervisão de um sistema autónomo. É assim importante que existam ferramentas que permitam ao homem intervir, interagir ou simplesmente supervisionar. Do mesmo modo é importante é que o homem faça parte da questão e que essa parte assuma uma forma que esteja previamente definida no sistema .

Torna-se assim incontornável caracterizar a que níveis o homem pode actuar, níveis esses que vão desde teleoperação passando pela teleprogramação até à autonomia completa.

Na teleoperação todos os robôs são operados directamente por um utilizador, poderá mesmo não haver absolutamente nenhum sensor a bordo. Habitualmente a complexidade deste tipo de sistemas é muito baixa e a interacção com o utilizador é muito elevada. Todos os movimentos são executados pelo operador usando para isso interfaces adequadas que poderão ir desde as simples interfaces gráficas, aos *joysticks*, a consolas ou exoesquelestos [8].

Na teleprogramação o operador manda executar manobras pré-definidas, ou vai elaborando os planos da missão a executar em tempo real. Hbitualmente sempre que o ambiente muda, ou existe algo imprevisto, o operador tem de

actuar. Neste caso os sensores já permitem a construção de manobras mais elaboradas como, por exemplo, o desvio de obstáculos. A complexidade deste tipo de sistema é média e a interacção do operador é bastante menor, apesar de ainda existir alguma.

Na autonomia completa o homem apenas tem a necessidade de supervisionar os acontecimentos. Neste caso a complexidade do sistema é extremamente elevada e a interveniência por parte do operador é nula.

A Figura 2.12 representa os diferentes tipos de iniciativa mista indicando a complexidade do sistema relativamente à interacção humana.

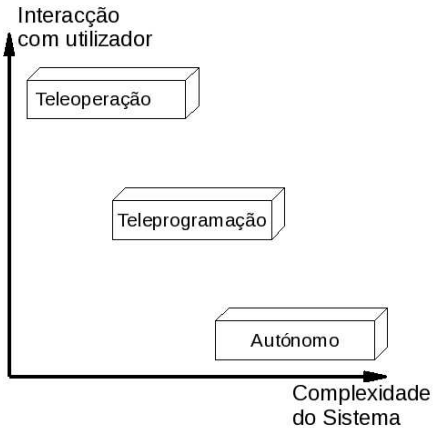


Figura 2.12: Complexidade *versus* interacção de três tipos de sistemas. Adaptado de [8].

Os cenários de aplicação da iniciativa mista são os mais variados. A aplicação deste tipo de estratégia pode ser verificada em locais como:

- Operações militares;
- Cenários de busca e salvamento;
- Aplicações industrias;
- Caracterização de cenários perigosos à distância;
- Exploração de outros planetas (p.e. Marte).

Uma das grandes plataformas de testes do uso de Iniciativa mista são as competições internacionais, competições essas que podem ser tão diferentes quando

o Robocup Rescue [20] ou o ELROB/MELROB [19]. São descritos também algumas das experiências em cenários militares [38], onde são abordados os aspectos do controlo de uma equipa de tanques não tripulados, ou de missões aéreas como o caso do MICA [37] que pretendia estudar a configuração, e reconfiguração em caso de avaria de um dos elementos de uma esquadra de aviões não tripulados em missões de reconhecimento. Existe ainda algumas abordagens a operações de desminagem [39] ou operações em zonas que sejam altamente perigosos [35]. Existem ainda cenários cuja complexidade também é relativamente elevada como o exemplo das missões a outros planetas sendo ultimamente dado um grande ênfase a Marte [40].

(Esta página foi intencionalmente deixada em branco.)

Capítulo 3

Casos de Estudo

Conteúdo

3.1 Introdução

Os casos de estudo apresentados resultam de três cenários bem distintos. São abordados três exemplos da interacção homem-máquina associados a cenários cuja autonomia dos sistemas varia consideravelmente no seu grau de autonomia. No primeiro caso, a equipa de futebol robótico ISePorto [46], temos uma

equipa de robôs cuja autonomia é bastante elevada. O segundo caso aborda uma aplicação do veículo ROAZ [42] (onde o operador fecha por vezes o ciclo de controlo) que resulta na utilização de um ASV para recolha de dados batimétricos. Por fim é apresentado um caso cuja a autonomia é muito reduzida e que exige uma grande actuação por parte do operador, o protótipo WPS.

3.2 Equipa de futebol robótico ISePorto

Interessa definir o enquadramento da equipa de futebol antes de entrarmos na aplicação em si mesma. A iniciativa ROBOCUP [47], e mais propriamente as ligas de futebol robótico, pretendem promover a robótica abordando um tema de grande actualidade - o futebol. Este projecto revela-se de grande interesse uma vez que grande parte dos algoritmos, *software* e *hardware* que são resultado destas aplicações são muitas vezes aplicadas a outros cenários. Os principais tópicos de desenvolvimento são a fusão sensorial, navegação autónoma, controlo híbrido não linear e coordenação.

Um jogo de futebol robótico tem o mínimo de intervenção. O jogo começa com a colocação em campo dos jogadores das duas equipas que, depois de estarem prontos a jogar, são ligados a um árbitro central constituído por um computador que envia os sinais aos jogadores através de uma ligação sem fios. O campo é constituído por duas balizas e um relvado verde com as marcações típicas de um campo real. Os jogadores são maioritariamente de cor preta em que cada equipa tem uma marca diferente pré-definida (magenta ou ciano). Após o árbitro dar início ao jogo, a intervenção humana passa a ser praticamente nula, existindo apenas quando se detecta uma anomalia num jogador. É ainda possível ter um treinador a funcionar num computador externo ao jogo. O treinador pode receber mensagens globais dos robôs e actuar em conformidade.

Apesar da equipa ser autónoma durante o jogo, a parametrização de alguns elementos, tais como o número de jogadores, tácticas iniciais e aplicação das regras de jogo, são sempre feitas com recurso a humanos. Há também ainda a referir que, durante o desenvolvimento dos algoritmos do jogador, é necessária a existência de ferramentas que permitam que o humano entre na malha de controlo, que efectue manobras usando a teleoperação e visualize alguns variáveis

internas da aplicação do robô [46].

Os principais requisitos para a supervisão da equipa de futebol robótico podem agrupar-se em quatro tópicos:

- Visualização modal dos dados
- Comunicação em tempo real
- Configuração das estratégias de controlo
- *Debug*, teste e avaliação de desempenho

A interface de supervisão necessita de permitir a visualização modal de todos os dados. Só se pode efectuar uma boa supervisão se se puder visualizar os dados da aplicação de várias formas e a diferentes níveis. A aplicação de supervisão, além de ter de conseguir gerir uma grande quantidade de dados, terá de os conseguir apresentar de forma a que um utilizador humano consiga retirar informação que lhe permita avaliar o estado global ou pequenos pormenores de um robô em tempo útil. É importante que seja possível ver a posição de todos os graus de liberdade (físicos) bem como o seu estado interno e como ele vê o mundo à sua volta. Estas necessidades surgem por diferentes factores, seja durante a fase de desenvolvimento em que é necessário garantir que o algoritmo produz os resultados satisfatórios, seja durante um jogo para posterior análise dos dados obtidos. Um exemplo prático é o de teste da procura de linhas do campo por parte de um robô. Depois dos cálculos feitos é necessário referenciar essa mesma linha no mundo e posteriormente perceber se o resultado é o correcto. Preferencialmente será interessante poder visualizar esta imagem em tempo real. É possível extrapolar este caso para todos os objectos do campo, nunca esquecendo que na representação tem se ser possível a distinção dos objectos por robô.

A necessidade da visualização modal dos dados, ou seja, a visualização de diferentes tipos e perspectivas da informação do robô, torna necessário ter estratégias de comunicação que permitam que tal aconteça em tempo real.

Cada veículo envia dados como posição actual, tipo de manobra, onde está a bola e ainda dados de coordenação, a *referee box* envia eventos de jogo e o treinador mensagens de coordenação. É necessário criar níveis diferentes de

dados que circulam em cada um destas linhas de transferência. Existem dados que têm de chegar em tempo útil e necessitam de confirmação de entrega, por exemplo as mensagem de coordenação. Existe outro tipo de mensagens que são enviadas periodicamente e que não há necessidade de confirmação e reenvio da mesma mas sim de uma nova actualização desse dado, por exemplo a posição actual do robô. Existe ainda outro tipo de informação que não é necessária chegar a todos os robôs, mas apenas áqueles interessados, por exemplo quando se executa uma manobra coordenada de ataque. Existem ainda dados que têm de ser enviados em *broadcast* tais como os eventos gerados pela *referee box*.

Daqui se depreende que um sistema com estas características necessita do envio de dados periódicos, não periódicos, com confirmação de entrega e com requisitos temporais bem definidos. Durante o desenvolvimento há que ter em conta ainda os *logs*. Estes dados são muito importantes, seja para visualizar em tempo real como se uma folha de cálculo se tratasse, seja para uso posterior com o recurso a análise gráfica. A referenciação dos dados adquiridos em tempo real é etiquetada temporalmente o que implica que todos os sistemas tenham de estar com os relógios sincronizados. Existe ainda outro factor importante que são os sensores de visão, este tipo de sensores precisa de funções especiais para mostrar a informação. Deverá ser possível mostrar a imagem real, bem como a sobreposição de imagens processadas. Estes dois últimos exemplos têm de permitir que seja reservada uma grande largura de banda para o uso em tempo real.

Durante a fase *debug* que é feito numa equipa de robôs surgem sempre grande quantidade de informação para posterior análise. Este tipo de informação tem de poder ser ligada e desligada consoante a utilização. Durante os jogos convém não estrangular a rede, que tipicamente é *wireless* a/b/g. Durante a fase de testes pode ser útil observar a evolução de alguns dados e determinadas variáveis.

Nas comunicações os principais aspectos/requisitos a salientar são:

- Grande quantidade de informação;
- Necessidade de preservar a rede;
- Possibilidade de diferentes saídas para a mesma informação: ficheiro, rede e uso interno;

- Dados do tipo *event-triggered* e *time-triggered*;
- Envio de dados em determinadas janelas temporais pré-definidas.

As configurações da estratégias de controlo necessitam de ser realizadas antes de cada jogo e precisam de um conjunto de comandos na interface de supervisão. As configurações neste tipos de estratégias necessitam de ser realizadas rapidamente, podendo para isso ser usado um *preview* ou simulador do que se irá passar com a configuração actual.

3.2.1 Princípio de funcionamento - *user case*

A operação da equipa de futebol robótico ISePorto, Anexo C, pode ser dividida em várias fazes: 'Setup', 'Briefing', 'Debug', 'Game', 'Debriefing' e 'Close'. Com o objectivo de se perceber cada fase torna-se importante definir as entidades envolvidas numa equipa de Futebol. São elas o *team leader* que toma decisões antes de cada jogo sobre cada jogador, o supervisor que monitoriza todos os dados durante o jogo e identifica possíveis falhas, o técnico que coloca e retira fisicamente o robô de jogo e o programador que desenvolve software para os robôs. As outras entidades serão o computador de supervisão, treinador (computador que recolhe e analisa todos dados de jogo, podendo participar fornecendo directivas aos jogadores de campo) a equipa de robôs e a *referee box*. A Figura 3.1 esquematiza as entidades envolvidas e a forma como se interligam entre si.

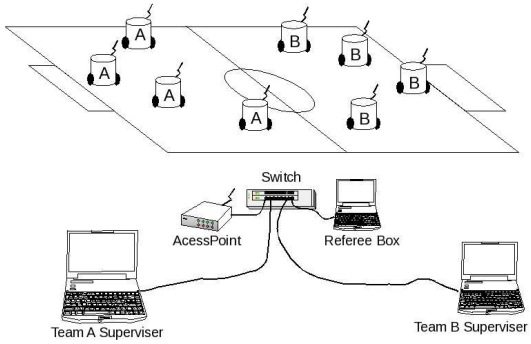


Figura 3.1: Arquitectura genérica da equipa de futebol robótico

Setup

A fase de 'Setup' não é mais do que todo o processo que se inicia no momento de desempacotar os robôs até os ligar e colocar em campo prontos a jogar. Procede-se então da seguinte forma nesta fase:

- O técnico "assembla"os robôs e monta possíveis partes que tenham sido desmontadas para o transporte. Faz verificações em todas os contactos de encaixe bem como ligações e conectores. Insere os conjuntos de baterias nos robôs.
- O técnico configura as ligações *wireless* de todos os robôs para que estes se liguem a uma rede própria que será usada por todas as equipas durante os jogos.
- O técnico monta e liga o servidor central que contem as aplicações/Sistemas operativos. O servidor também é configurado para ligar à mesma rede *wireless* dos robôs.
- O supervisor arranca a aplicação de supervisão e configuração da equipa.
- O técnico liga os robôs num botão próprio para o efeito.
- O supervisor faz desencadear uma série de testes de *software* e *hardware* remotamente para verificação do estado dos robôs.
- Quando o supervisor detecta uma falha coloca o técnico ao corrente da situação para resolução da mesma.
- Quando o supervisor verifica que os robôs estão operacionais em temos de *software* e *hardware* permite a passagem para a fase seguinte.

Briefing

O 'Briefing' é uma fase de transição onde são parametrizadas todas as opções que dependem de externos, o árbitro e a outra equipa.

- O *team leader* juntamente com o árbitro de jogo define qual a cor da equipa e que metade do campo será usada no início de jogo.

- O *team leader* notifica o supervisor da cor e que metade do campo será usada.
- O supervisor configura a cor e área de jogo da equipa através da aplicação de supervisão.
- O supervisor liga a consola ao árbitro electrónico (*referee box*) que será usado durante o jogo. Esta *referee box* tem um endereço IP e um porto de ligação que poderá variar consoante os jogos.
- O *team leader* informa o supervisor do tipo de estratégia a usar, definindo os defesas, atacantes, guarda-redes e se o treinador será usado ou não.
- O supervisor indica que tudo está configurado e pronto a passar à fase seguinte que será a de 'Game' ou de 'Debug'.

Game

A fase de 'Game' representa tudo o que se passa durante o jogo antes e após o intervalo.

- O operador indica na aplicação de supervisão que os robôs estão prontos para jogar.
- O supervisor e o *team teader* supervisionam o que se vai passando na equipa. É necessário supervisionar a posição dos robôs e respectiva confiança em campo, o *heartbeat* do robôs, a orientação de cada robô e a confirmação da recepção de comandos da *referee box*.
- O *team teader*, depois de pedir ao árbitro, pede ao técnico para retirar um robô de jogo caso seja detectada uma anomalia.
- O técnico pressiona um botão que indica que o robô fica fora de jogo caso seja necessário.
- O *team leader*, depois de pedir ao árbitro, pede ao técnico para colocar um robô em jogo caso seja necessário.
- O técnico pressiona um botão que indica que o robô fica imediatamente em jogo.

- Durante esta fase a equipa apenas reage a eventos da *referee box*. O procedimento do uso ou programação das regras são aqui omitidos para simplificar a análise da supervisão.

Debug

A fase de 'Debug' permite validar *software* ou partes de *hardware*.

- O programador pode ligar um simulador ou um *player* que representa um jogo entretanto gravado.
- O programador pode alocar uma largura de faixa que permita enviar todos os dados em tempo real e disponibilizá-los na aplicação de debug.
- Depois do 'Debug' terminado, o programador pode passar para a fase de 'Briefing' novamente ou para o 'Close'.
- O operador pode teleoperar ou teleprogramar cada robô individualmente.

Debriefing

O 'Debriefing' serve para verificação do que correu eventualmente mal.

- O supervisor copia a informação que ficou armazenada sob forma de *log* nos robôs para imediata ou posterior verificação.
- O *team teader* informa se a fase seguinte é um novo jogo e passa para 'Briefing', ou se já não há mais jogos e passa para 'Close'.

Close

O 'Close' é para as tarefas finais, tais como desligar, desmontar e embalar.

- O supervisor desliga os robôs remotamente.
- O técnico retira as baterias.
- O técnico desmonta (o necessário) e embala os robôs ou são arrumados para o próximo jogo.

3.2.2 Resultados da aplicação da equipa de futebol robótico

Como resultado final da aplicação de futebol é necessário que as ferramentas descritas permitam que a equipa jogue cumprindo todos os regulamentos e regras definidas pela ROBOCUP [47]. Entende-se nesta fase que a equipa se ligue e interprete os comandos do árbitro (*referee box*), e mediante os comandos execute as jogadas em conformidade.

3.3 Batimetria e monitorização com ASV

No processo de recolha de danos e monitorização com ASV é necessário começar por definir uma área de trabalho e o tipo de dados que serão objecto de estudo. Esta definição é feita recorrendo a um mapa. Este cenário de operação é definido no início da missão mas pode ser estendido dinamicamente caso haja necessidade. Durante uma recolha de dados típica é usada a manobra do 'corta-relva', Figura 3.2, onde se torna necessário definir as distâncias e eventualmente o raio de cada curvatura entre cada perfil.

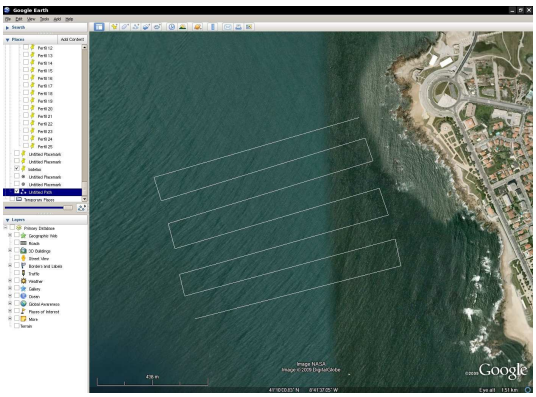


Figura 3.2: Manobra do tipo 'corta-relva' típica na recolha de dados batimétricos de um ASV no "Google Earth"®.

Os requisitos do sistema passam pela possibilidade da teleoperação, visualização modal dos dados, comunicações móveis, especificações das missões e mecanismo de teste e *debug*:

- Teleoperação e teleprogramação;

- Consola de Operação;
- Visualização modal dos dados;
- Comunicações;
- Especificações das missões;
- *Debug* e teste.

Na teleoperação é necessário o mapa do local, imagem de vídeo em tempo real e uma forma de controlo físico, tipicamente *joystick*. Tendo um ASV características de *outdoor* é importante uma georeferenciação da posição actual do mesmo com representação em cima de um mapa. A mesma interface deverá ser capaz de poder suportar a gestão de missões. À semelhança do que havia sido referido para o ISePorto, é fundamental uma boa estrutura de *logs* já que um dos objectivos principais é o de recolha de dados.

A consola de operação é um computador *sun light readable* podendo ser usado um mini computador de bolso (p.e. Nokia n770 ou n800). A aplicação de controlo deverá ser modular de tal forma que se adeqúe facilmente aos diferentes tipos de consolas de operação. A portabilidade do código também é importante visto poder ter que correr em plataformas computacionais distintas (i.e i386, ARM, etc.). Os controlos deverão ser o mais simples possível e deverão evitar os erros humanos. No caso da execução de uma missão, deverá ser possível gerar as respectivas missões *offline* recorrendo ao uso de pontos e linhas ou usando manobras compostas mais específicas que são enviadas para o veículo. Durante a execução da missão deve ser possível mudar a missão, fazer pausa, ou simplesmente abortar a missão. Deve existir também a possibilidade da rápida imobilização do veículo ou uma comutação rápida entre o controlo autónomo e teleoperação.

A comunicação entre o veículo e a consola terrestre é feita através de uma antena wireless IEEE 802.11 a b/g usando a rede A a 5GHz. Existe a necessidade de haver um sinal de *heartbeat* entre o veículo e a consola. Quando o sinal não está presente o veículo deve entrar numa manobra que o leve à última posição conhecida com *link* ou ficar simplesmente imobilizado. Os comandos enviados pela consola deverão ser repetidos até que exista um *heartbeat* por parte do

veículo. Este tipo de sinais são similares aos que são enviados pelo veículo quando um qualquer evento ocorre (i.e. mudança de manobra). A informação do estado do veículo é enviada a uma taxa pré definida. Estes dados não são reenviados caso não cheguem à consola, é sempre preferível enviar um novo sinal do que reenviar um antigo. Nesta categoria devem considerar-se todos os dados que dão origem à actualização da informação na consola.

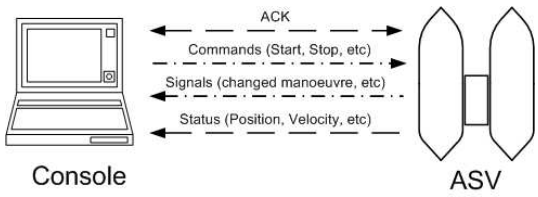


Figura 3.3: Direcção das comunicações entre a interface e o ASV

Apesar da decisão de parar ou abortar uma missão ser realizada com os dados que são constantemente actualizados na consola, é possível dividir os dados que são fornecidos pelos sensores em duas categorias, os dados que são usados para o controlo do veículo e os dados da aplicação, por exemplo batimetria. Os dados críticos devem ser mostrados de uma forma clara e sempre que possível a cores diferentes. O outro tipo de dados pode ser mostrado noutro tipo de janelas da aplicação. A informação do *link* tais como potência e actividade devem estar sempre presente. É a única forma de se poder ter alguma confiança nas manobras a executar bem como a que distância se pode ir. Deve ser fornecido um mapa do local da missão com a indicação das coordenadas de GPS.

A missão deve ser criada num interface próprio, onde deverá haver a possibilidade de a guardar e alterar. Deveram ser fornecidas ferramentas que permitam criar as missões facilmente, adicionar ao mapa pontos, linhas, poligonos, que mais tarde darão origem à missão. Depois da missão ser executada, deverá apenas poder ser abortada, parada ou em alternativa acrescentadas novas directivas à missão que está a correr.

Os dados de *log* poderão ser guardados a bordo do veículo ou enviados para a consola em tempo real. É preciso relembrar que existem dados cujas taxas são tão elevadas que não podem ser enviados em tempo real. Há que contar também com possíveis falhas na comunicações que podem levar à não integridade

de alguns dos dados. Os dispositivos de log a bordo deste tipo de veículos não devem conter partes móveis, i.e. disco duros, deverão ser utilizados, por exemplo cartões de memória.

3.3.1 Princípio de funcionamento - *user case*

Sendo o ASV, Anexo C, um veículo que funciona essencialmente associado a aplicações de dados batimétricos, torna-se necessário descrever o cenário de operação, seja ela autónoma (execução de missões) ou teleoperada. Podemos então dividir o uso do sistema de recolha de dados batimétricos e monitorização nas seguintes fases: 'Setup', 'Briefing', 'Debug', 'Pré-Mission', 'Mission', 'Pós-Mission', 'Debriefing' e 'Close'. As entidades envolvidas neste tipo de cenários são o *mission leader*, o operador e o técnico. Além do ASV para concretização da missão também é possível contar com um Barco de apoio.

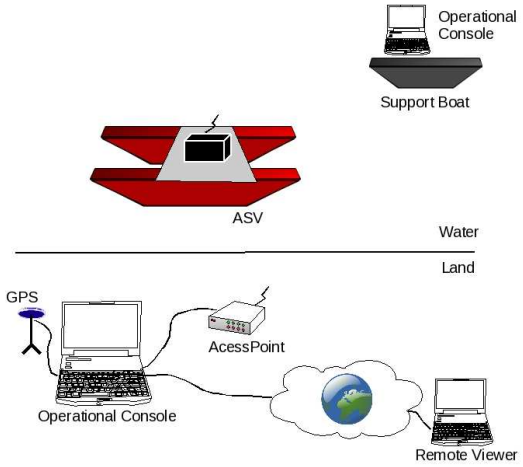


Figura 3.4: Arquitectura genérica de recolha de dados com ASV

Setup

O 'Setup' do ROAZ resulta das acções necessárias até à colocação do ASV na superfície aquática pronto a dar início à preparação da missão.

- O técnico Assembla a consola de terra e o veículo ROAZ.

- O técnico liga o sistema computacional a bordo do ROAZ. O sistema computacional ficará ligado a uma rede *Ad Hoc* que começará imediatamente a disponibilizar informação para essa mesma rede;
- O operador verifica graficamente na consola de operações que o sistema computacional a bordo do ROAZ está a funcionar, bem como cada módulo necessário para a execução da missão;
- O técnico coloca o veículo na água juntamente com o barco de apoio.
- O operador verifica os módulos de *hardware* críticos para o sucesso da missão (*link* de comunicações, *propellers*, etc);
- O operador em consonância com o *mission leader* autorizam o início da fase de definição da missão.

Briefing

O 'Briefing' é uma fase onde são parametrizadas todas as opções que dependem de elementos externos ao sistema. Em último caso serve para que seja verificado a operacionalidade de todos os módulos.

- O operador define a missão na, consola, num mapa de missões. Esta missão poderá ter sido previamente definida. O operador define a área de trabalho, bem como o tipo de percurso a efectuar.
- O operador envia a missão para o ROAZ.
- O operador notifica que está tudo operacional e configurado, podendo passar-se à fase seguinte.

Pré-Mission

Por questões de operacionalidade, na maioria da missões, o local onde ocorre a fase de 'Briefing' não corresponde ao local da área de missão e por simplicidade é necessário "levar"o veículo de um local para o outro.

- O *team leader* informa o barco de apoio, caso seja necessário, para que este reboque o veículo até o cenário de operação.

Mission

A 'Mission' representa a actividade que é realizada até que a missão definida no 'Briefing' termine (com sucesso ou por interrupção anormal).

- O operador dá início da missão a pedido do *mission leader*.
- O operador monitoriza os dados dos sistemas críticos a bordo, quer dos sensores que permitem executar a missão quer dos sensores que são usados apenas como recolha de dados.
- O operador informa o *team leader*, caso existam alarmes provenientes do ASV. Os alarmes disponíveis são do tipo: baterias fracas, inoperacionalidade funcional de alguns módulos e falha de *link* de comunicações.
- O operador verifica que existe coerência e integridade dos dados que serão o resultado da missão.
- O operador pode abortar ou parar ou fazer continuar a missão.
- O operador pode passar para modo manual permitindo a teleoperação remota ou teleprogramação.
- O operador pode alocar temporariamente uma largura de banda para visualizar os dados que estão a ser registados a bordo do ASV.
- O *mission leader* é informado pelo Operador que a missão terminou e indica a passagem para a fase seguinte.

Pós-Mission

Como o local de término da missão pode não ser o mesmo do local de 'Briefing', é necessário levar o veículo novamente para o local de recolha.

- O *team leader* informa o barco de apoio, caso seja necessário, para que este reboque o veículo até o local de recolha.

Debriefing

O 'Debriefing' serve para avaliar a missão antes de dar por terminada a missão definitivamente.

- O operador envia os dados obtidos durante a missão para a consola de operações para serem analisados.
- O *mission leader* define se é para executar uma nova missão e passa-se para o 'Briefing' ou se é para terminar e se se segue para o 'Close'.

Close

O 'Close' é o estágio do fim de missão

- O operador desliga o ASV remotamente.
- O técnico desliga as baterias.
- O técnico desmonta os componentes amovíveis sendo, de seguida, o ASV colocado no reboque.

3.3.2 Resultados da aplicação de monitorização e batimetria de um ASV

Como resultado da missão com o ASV teremos um conjunto de dados etiquetados temporalmente e georeferenciados. Este conjunto de dados pode ser passado para aplicações externas que permitam, por exemplo, a reconstrução 3D do mapeamento realizado na missão.

3.4 Levantamento de potências de sinais wifi

O Sistema "Wireless Positioning System", Anexo C, consiste numa plataforma móvel para efectuar mapas de potências de sinais de *access points wifi*, para posterior utilização na localização e posicionamento de pessoas através de "TAG"ou computador de bolso.

É possível dividir a plataforma no sistema de baixo nível e na unidade computacional que executa a interface de operação.

O sistema de baixo nível tem como objectivo a medição da velocidade e posição por hodometria. Estes valores são enviados para o sistema computacional que é responsável pela interligação de todos os dispositivos móveis PDAs

e TAGs, com o respectivo cálculo da potência do sinal *wireless* a todos os *access points* disponíveis. Depois de toda a informação fundida é enviada para a consola central a informação relevante como a posição actual, as potências e número de *Access Points*.

A interface de operação está disponível no sistema computacional do veículo ou num computador portátil ligado ao mesmo. A interface é responsável pela configuração do sistema, referenciação automática em tempo real do local de trabalho mediante um mapa fornecido e representação do estado do veículo com identificação visual da direcção e local do mesmo. Existem vários níveis de informação que são fornecidas a pedido pelo utilizador. A Figura 3.5 esquematiza o sistema na sua aplicação.

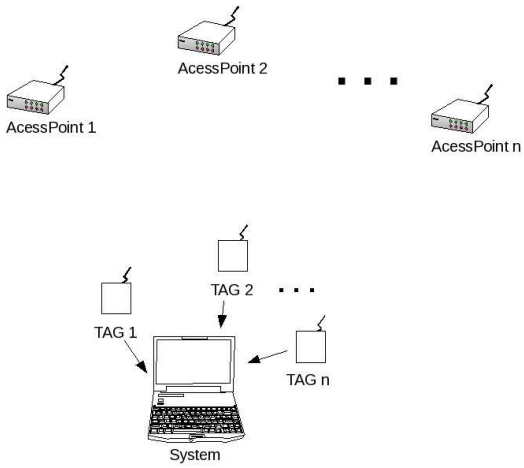


Figura 3.5: Arquitectura genérica do *Wireless Positioning System*

Neste caso começa-se por criar a missão que mais tarde será executada. A interface terá por objectivo gerir a missão como se de um projecto de software fosse, ou seja, criação de projecto, inserção de ficheiros, extracção de informação dos ficheiros, manipulação dos ficheiros e por fim guardar todo o projecto. Duas perspectivas de utilização da interface podem desde já ser definidas, a do operador que efectuará no terreno a missão e o programador que criará a missão.

No óptica do programador, depois de serem inseridos os mapas dos locais, é necessário marcar os pontos por onde a estrutura terá de passar registando os valores das potências dos *access points*. A única ferramenta disponível é a inserção de pontos no mapa que terão várias indicações visuais, por exemplo:

verde se já foram analisados, vermelho se ainda não foram analisados. Os pontos já devem estar disponíveis no mapas formando grelha com dimensões que possam ser configuráveis pelo operador. Depois de todos os pontos marcados em todos os mapas fornecem-se os ficheiros do projecto ao operador.

O operador usará a mesma interface para execução da missão, depois de lidos os ficheiros fornecidos pelo programador. Deverão estar disponíveis controlos que permitam iniciar, parar ou abortar a missão de forma clara. No mapa ver-se-á a indicação da posição actual da estrutura bem como a sua direcção depois desta ser inicializada no local. Depois de se carregar no botão de início de missão, empurra-se a estrutura até ao primeiro ponto. Quando se chega ao primeiro ponto, ficará disponível um botão adicional de inicio de *log*. Quando se termina o *log* de um ponto continua-se até outro ponto e assim sucessivamente até terminar o número de pontos no mapa.

É importante que na interface surjam diferentes tipos de informação, por exemplo o estado da bateria, o estado de cada bloco do dispositivo, de tal forma que o operador perceba se a missão está a correr conforme planeado ou não.

A interface terá no final que poder manipular os dados de *log*, gravar, alterar, apagar e enviá-los para uma base da dados.

3.4.1 Princípio de funcionamento - *user case*

Pode-se dividir o sistema de levantamento de potências de sinais wifi em cinco fases de operação, o 'Setup', 'Briefing', 'Mission', 'Debriefing' e 'Close'. As entidades envolvidas são o operador e o veículo.

Setup

- O operador retira da caixa de transporte a estrutura e "assembla"possíveis partes que tenham sido desmontadas para o transporte. Insere as baterias no sistema.
- O operador liga a plataforma computacional no veículo e executa a aplicação de configuração e execução.
- O operador verifica que todos os blocos da plataforma estão funcionais através da aplicação gráfica.

Briefing

- O operador abre os ficheiros de missão pré-definidos, ou cria uma missão nova
- O operador inicializa no mapa a posição actual do sistema.
- O operador dá início à missão através da interface gráfica.

Mission

- O operador empurra o veículo até ao ponto seguinte especificado pela missão onde se dará o inicio ao registo de valores de potências de sinais wifi;
- O veículo regista os valores de potências dos sinais wifi de todos os *access points* disponíveis nessa zona;
- Quando o *log* acaba de ser feito naquele ponto, empurra-se novamente o veículo até ao novo ponto e repete-se o procedimento até ao último ponto especificado na missão;
- O operador, durante a aquisição do sinal, deverá verificar o estado do processo a decorrer e verificar eventuais alarmes vindos do veículo.
- A interface deverá dar a informação de que todos os sinais foram registados em todos os pontos.
- O operador pode abortar, parar ou continuar uma missão a qualquer altura.

Debriefing

- O Operador remove o cartão de memória com os dados ou envia os dados para uma computador como forma de *backup*.
- O Operador define se passa para o 'Close' ou se volta ao 'Briefing' para uma nova missão.

Close

- O operador desliga remotamente a estrutura;
- O operador remove as baterias;
- O operador desmonta (o necessário) e embala a estrutura para o transporte.

3.4.2 Resultados da aplicação do levantamento de potências de sinais wifi

Como resultado desta aplicação deveremos obter um ficheiro ou uma base de dados com um conjunto de dados que permita calcular a posição de pessoas ou bens que disponham uma TAG identificadora dentro de um edifício fechado.

3.5 Características comuns em todas as aplicações

Pode dizer-se que existem muitos aspectos em comum em todas as aplicações atrás descritas. Existe um tronco comum a partir do qual é possível descrever uma arquitectura genérica para todos os ambientes. As particularidades de cada aplicação serão, depois de traçada a linha comum, discutidas caso o caso em função das funcionalidades e das características pretendidas na implementação.

- As fases de execução de todos os sistemas são muitos idênticas: 'Setup', 'Briefing', 'Mission', 'Debriefing' e 'Close'. É possível, através de uma aplicação, saber qual a etapa actual e se já é possível ou não passar para a etapa seguinte caso as condições estejam todas satisfeitas.
- As comunicações são um dos pontos comuns, a estrutura de comunicações são, em grande parte, um dos temas a considerar nos ambientes multi-robóticos. Nas comunicações há que considerar a passagem de informação entre as diferentes aplicações que correm no próprio robô bem como as interacção entre os diferentes robôs e possíveis consolas centrais.
- Todas as aplicações definem a necessidade de existência de alarmes críticos, orientados ao evento, que devem chamar a atenção do supervisor ou operador.

- Simulação das equipas de robôs. Quando a estrutura de teste e *debug* é muito grande, a melhor forma, antes de passarmos à prática, de testar o código é por intermédio da simulação evitando assim uma grande logística. A simulação deverá contemplar a dinâmica de um robô apenas bem como a de uma equipa inteira ou apenas um pequeno aspecto de um robôs como um sensor apenas.

- Aplicação de controlo deve ser modular por forma a garantir que é independente do veículo. A especificação da API é um passo importante por forma a garantir a abstracção de vários factores tais como a independência da tracção do veículo, a independência da plataforma física, em termos genéricos, e se garantirmos plataformas computacionais idênticas, garantir uma abstracção dos sensores e actuadores.

- As ferramentas de controlo e visualização de dados poderão ter um função comum, por exemplo, na apresentação dos dados. Considerando os GPS, mesmo que sejam de fabricantes e modelos diferentes e protocolos diferentes, os dados a mostrar na consola serão sempre iguais (é possível extrapolar este tipo de descrição para outro tipo de sensores).

- O grau de autonomia da aplicação está relacionado directamente com a complexidade da interface e do empenho do supervisor. A etapa de 'Briefing' é tanto mais complexa quanto mais elevada é a autonomia do sistema.

- O resultado de quase todas as aplicações, quer seja da missão ou de operação, é um conjunto elevado de dados que pode ser trabalhado posteriormente.

- As entidades envolvidas nos três casos apresentados têm funções muitos semelhantes entre si.

As características comuns apresentadas fornecem directivas na definição de uma arquitectura que possa conter itens comuns aos discutidos mas que, ao mesmo tempo, seja de fácil flexibilização e expansão para qualquer outro tipo de sistema autónomo, seja ele considerado no seu todo ou apenas numa particularidade.

Capítulo 4

Arquitectura de supervisão de sistemas autónomos

Conteúdo

4.1 Introdução

É sabido que um total controlo sobre um veículo que permita a teleoperação é uma tarefa muito complexa e requeria, se tal fosse possível, um grande treino por parte do indivíduo que o fosse teleoperar. Todavia a existência de um veículo ou múltiplos veículos que executem uma dada missão de uma forma completamente autónoma é também algo de muito complexo e não praticável actualmente. A iniciativa mista propõe neste campo a junção entre a autonomia na execução de determinadas manobras com a teleoperação para resolver os problemas que as duas abordagens teriam. Há ainda de notar que a teleoperação apenas permite o controlo de um veículo de cada vez, para a teleoperação de múltiplos veículos seria necessário o empenho de um número de indivíduos igual ao número de máquinas. A iniciativa mista pressupõe sempre a existência de um humano (ou mais) no ciclo de controlo. Todas as missões são controlados tendo em conta o elemento humano, nem que seja apenas na supervisão.

Uma das formas de medir a eficácia dos métodos apresentados consiste no recurso à medida da eficácia da execução da missão relativamente ao descuido por parte do operador [9] conforme representa o gráfico da Figura 4.1

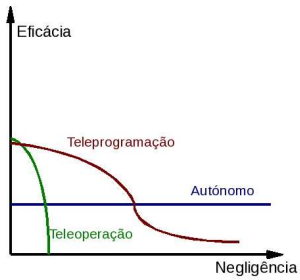


Figura 4.1: Eficácia *versus* negligência por parte do operador. Adaptado de [9].

Pode observar-se pela Figura 4.1 a diferença entre o factor humano e a ausência dele em função da atenção que esse mesmo elemento humano tem sobre o estado geral da missão.

4.2 Entidades envolvidas

É possível diferenciar as entidades envolvidas na iniciativa mista. As entidades que surgem na sistematização dos casos de estudo são as seguintes:

- O **operador** é uma entidade que deverá ter algum conhecimento, mas não necessariamente um conhecimento de desenvolvimento da aplicação. Deverá ter algum treino para executar determinadas missões. Os dados de que necessita são apenas do estado corrente de toda a missão e dos veículos. A decisão de algo crítico é sempre do operador.

- O **técnico** ajuda nas tarefas de montagem e desmontagem dos robôs e consolas de operações.

- O **programador** é alguém que pode substituir o operador, desde que não tenha que tomar decisões críticas. O tipo de dados de que necessita são, na grande maioria das vezes, de teste e debug.

- O **veículo autónomo** é algo que consegue executar determinadas manobras sem uma teleoperação total, e que pode interpretar missões que necessitem de manobras complexas baseadas nas manobras básicas.

- O **supervisor humano** é alguém que analisa um conjunto de informação de um sistema robótico/multi-robótico podendo ou não tomar decisões que afectam a execução da missão do mesmo sistema.

- O **supervisor robô** é uma peça de software que analisa um conjunto de informações de um sistema robótico/multi-robótico podendo ou não tomar decisões que afectam a execução da missão do mesmo sistema.

- A **equipa de veículos** é um conjunto de veículos autónomos que consegue executar missões que não seriam possíveis com um veículo apenas.

- O **líder da missão** é alguém que sabe toda a missão que será executada e o que fazer em cada altura ou em situações não previstas.

4.3 Interacção homem-máquina

Na interacção homem-máquina, e tendo em conta as entidades envolvidas, é possível caracterizar diferentes tipos de manipulação. Já se sabe que o operador faz parte do ciclo de controlo, no entanto podemos descrever os diferentes tipos de funções que tem, bem como o nível de empenho que ele tem que ter em cada uma delas por forma a conseguir executar a missão. Existem dois níveis de operação entre o homem e a máquina, a manipulação directa e a manipulação baseada na interacção.

4.3.1 Manipulação directa

Na manipulação directa, estamos na presença da teleoperação pura. O humano está sempre próximo do robô dando-lhe instruções. Todos os graus de liberdade do robô são controlados pelo humano, fazendo com que o robô funcione como uma extensão do próprio humano na execução de tarefas que exijam precisão. Não é necessária a existência de sensores.

4.3.2 Manipulação baseada na interacção

A manipulação baseada na interacção pressupõe que os operadores baseiem os seus comandos na análise dos valores apresentados pelos sensores que equipam o robô em causa. Existe continuamente um *feedback* dos valores gerados a partir dos sensores que auxiliam o operador na execução das manobras.

4.4 Interfaces gráficas

A forma mais comum de um humano interagir com uma máquina é a interface gráfica computacional (GUI). Existem outras formas como a voz, percepção do movimento ou mesmo através das sensações. No entanto um sistema deste género apenas iria fazer com que a especificação do problema se tornasse bem mais complexa. A interface computacional permite sem dúvida uma mais fácil monitorização e controlo ainda que se trate de equipas com diferentes robôs.

4.4.1 Desenho de uma interface gráfica

Uma interface gráfica genérica, mesmo para o desenvolvimento de aplicações computacionais que não requeiram nenhuma interacção com uma máquina a não ser aquela com a qual interagimos ao usá-la (computador pessoal), terá um aspecto idêntico à Figura 4.2.

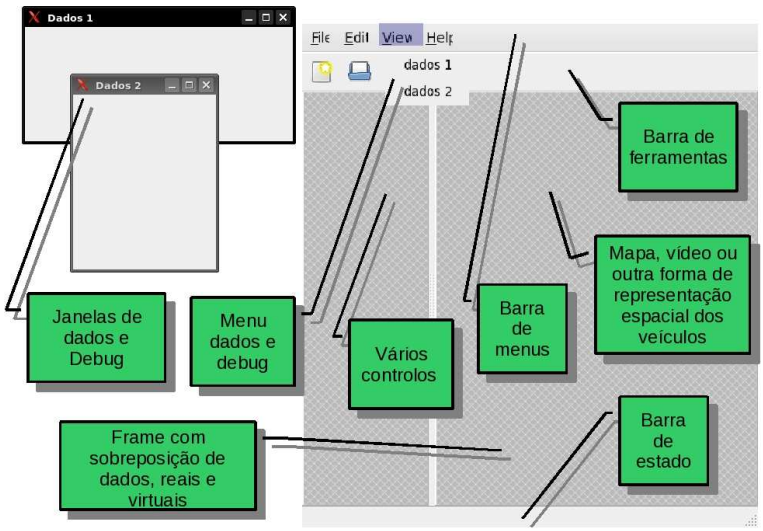
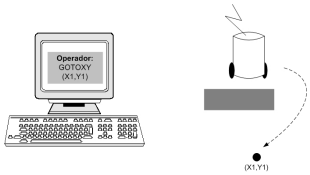


Figura 4.2: Construção de uma interface genérica aplicada ao controlo de veículos robóticos

4.4.2 Traços genéricos para interface iniciativa mista

São descritos em [48] por forma a estabelecer algumas linha gerais de trabalho para o desenvolvimento de aplicações gráficas para o uso na iniciativa mista. A maior parte destas directivas vão de encontro com as ferramentas de pensamento visual. Alguns dos principais tópicos são descritos a seguir.



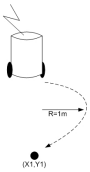
Necessidade que o robô ou a equipa possua à partida serviços automatizados que forneçam uma mais valia à teleoperação.

Ter atenção nas tarefas criticas e na forma como o utilizador é alertado.

Emprego do diálogo com o utilizador quando não há certeza de que os processos automáticos são o que o utilizador de facto pretende.

Existência de meios que permitam forçar que o veículo termine uma acção automática.

As acções e os tempos devem ser sempre controláveis através de alterações que sejam facilmente observáveis na interface.



Permitir sempre que o utilizador tenha meios de fornecer ao veículo ainda mais directivas ou parametrizações do que aquelas que estão presentes por omissão, tornando a tarefa mais precisa ou com mais graus de liberdade.

4.5 Arquitectura genérica proposta

A arquitectura proposta, conforme se pode observar na Figura 4.3, para o uso em equipas de veículos, tem como principal objectivo responder aos principais problemas enunciados nas diferentes áreas abordadas e permite ainda fornecer linhas orientadoras para a construção da interface final. O objectivo principal

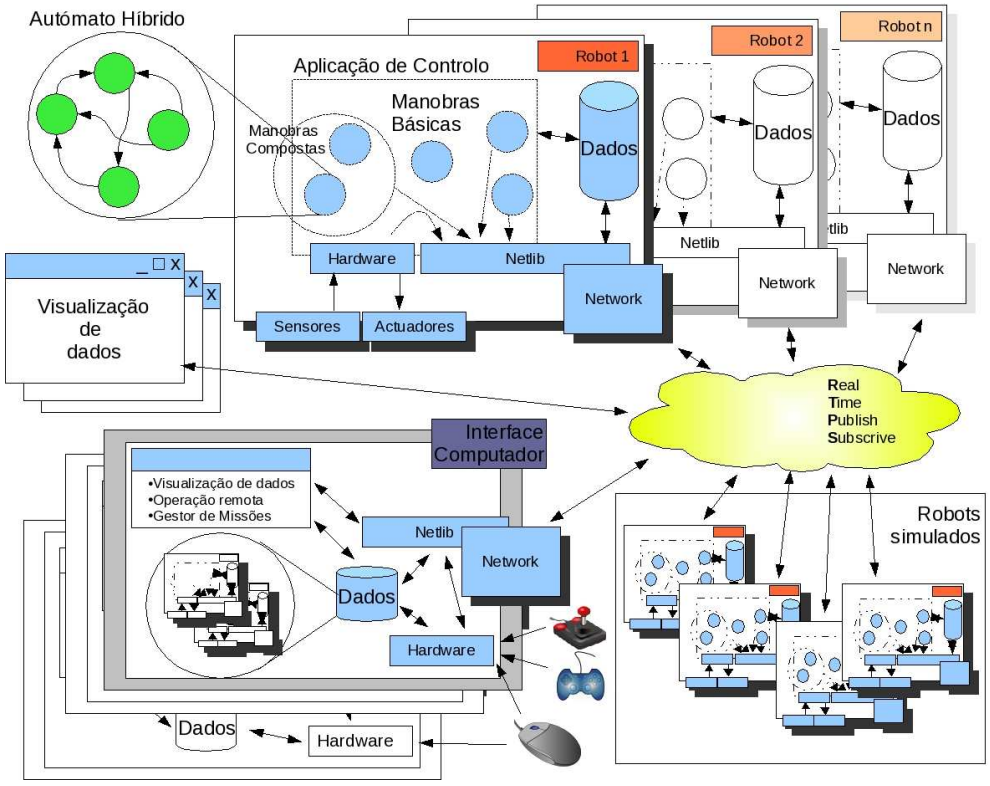


Figura 4.3: Vista geral da iniciativa mista

é o de observar na interface o estado global da equipa de robôs mas também, por exemplo, as alterações existentes numa determinada variável dentro das aplicações de controlo, sempre que desejado. É ainda pensada a forma de como uma equipa real possa interagir com uma equipa simulada, ou pelo menos parte dela, simulação essa que pode ser de um robô inteiro ou apenas de um sensor. Por fim há que acrescentar as comunicações, sendo o protocolo RTPS (*Real Time Publish Subscride*) o escolhido para estes casos. Todos os tipos de dados, com todas as suas características, que circulam neste tipo de equipas são contemplados neste protocolo.

4.5.1 Arquitectura dos robôs

A arquitectura apresenta uma forma de controlo dos robôs organizada numa rede de autómatos híbridos. O uso de sistemas híbridos permite a descrição

dos processos contínuos em conjunto com as alterações discretas dos eventos que vão sucedendo. Esta rede de autómatos fornece as manobras básicas de cada robô permitindo à aplicação de controlo seleccionar a que pretende usar ou configurar uma associação de manobras que resultam numa manobra composta bastante mais complexa. Este facto é de extrema importância quando se faz um controlo autónomo, principalmente quando o veículo tem de interpretar o cenário em que actua na execução de missões. Esta arquitectura deve abstrair-se de toda a plataforma, principalmente em termos de locomoção. A geração de mapas virtuais dos cenários deve ser idêntico em todas as aplicações, bem como a biblioteca de manobras simples ou compostas. A forma de tracção dos veículos em questão é que deve fazer com que exista um conjunto de funções específicas aos controladores/actuadores em causa.

4.5.2 Arquitectura de simulação

O problema de grandes equipas de robôs reside da sua disponibilidade para o teste de um algoritmo acarretar uma logística muito grande. A simulação pode resolver em parte este problema. A arquitectura proposta coloca os robôs simulados com a mesma interligação ao mundo que têm os robôs reais. Na arquitectura apresentada é possível que uma equipa de robôs simulados interaja com a equipa real. A API de comunicações deve estar preparada para que o fluxo de informação siga, por exemplo, para o simulador dos actuadores ou para os actuadores em si mesmo, permitindo assim o hardware no ciclo de simulação. A simulação assume assim um papel fundamental pelo que se torna incontornável o desenvolvimento de uma solução ou a utilização de simuladores genéricos tais como o Player/Stage ou CARMEN.

4.5.3 Comunicação RTPS

A comunicação recorrendo ao protocolo *Real Time Publish Subscribe* (RTPS), é aquela que permite toda a elasticidade pretendida na simulação dos diferentes componentes de um robô, quer na simulação de uma equipa inteira, quer na interacção existente entre os robôs não simulados, ou ainda no fluxo de dados interno na aplicação de controlo dos robôs.

O protocolo RTPS permite, devido aos seus princípios, diminuir facilmente

a largura de faixa utilizada, tendo assim uma maior flexibilidade em redes de veículos devido à forma inovadora de publicação/subscrição da informação.

4.5.4 Interface homem-máquina

A interface homem-máquina deverá permitir as seguintes operações:

- Visualização de dados;
- Operação Remota;
- Gestão das missões;
- Gravação de dados (*logs*).

4.5.5 Visualização de dados

A visualização de dados pode ser independente das restantes operações. Permite saber a posição geoespacial dos veículos ou outro informação genérica do que está a ser realizado por cada um. Este tipo de informação deve poder ser replicada por vários visualizadores independentes. Neste caso não há iteração possível por parte de quem visualiza esta informação. Este tipo de dados pode ser mostrado em tempo real em aplicações conhecidas tais como o "Google Maps"®.

O visualizador da interface deverá permitir mostrar a informação com diferentes graus de acesso. Tem de ser possível, pela observação da interface, saber a todo o instante o estado e posição dos veículos. A informação mais crítica para o desenrolar da missão deverá estar sempre presente se se pretende fazer *log* de um sistema inercial é necessário ter um *feedback* do bom funcionamento do mesmo durante toda a missão. Os dados menos relevantes deverão ser mostrados em janelas da interface que sejam chamadas por menus, e não apresentadas directamente na janela principal. Este tipo de dados inclui as ferramentas de *debug*, por exemplo uma janela que indique o que se passa no autómato da aplicação de controlo.

4.5.6 Operação remota

Os comandos de teleoperação devem existir, seja para *debug* da aplicação ou uso durante uma missão. A teleoperação é realizada pela interface gráfica di-

rectamente ou através de dispositivos que permitam o controlo sem olhar para o monitor tais como rato ou *joystick*.

4.5.7 Gestor de missões

A fácil integração de um gestor de missões numa arquitectura deste género é extremamente importante. A rede de autómatos híbridos da aplicação de controlo torna a integração de um gestor de missões relativamente fácil. Uma missão não é mais que um conjunto de manobras simples e/ou complexas pré-definidas até ser atingido um dado objectivo. A interface terá que fornecer as ferramentas adequadas à geração da missão no local, bem como o tipo de manobras que cada veículo consegue efectuar.

A secção seguinte aborda as funcionalidades da interface desenvolvida em função da arquitectura proposta, bem como a descrição de algumas missões efectuadas onde foram usadas as interfaces com os respectivos resultados.

4.5.8 Arquitectura de *logs*

A arquitectura de *logs* foi pensada para se poder efectuar registos da aplicação de controlo por si só ou da informação que é o objecto dessa mesma aplicação. Os sistema de registo de dados apresentado na Figura 4.4 permite criar diferentes níveis de *logs* e reencaminhá-los consoante a sua natureza e destino.

A Figura 4.5 descreve o fluxo de dados bem como a forma de o configurar. Depois dos dados gerados, estes passam por um filtro que verifica quais os dados que estão indicados para serem registados e passarem à fase seguinte. Os dados são depois encaminhados para os módulos de software correspondentes que são: a *network*, cartão de memória ou fila de mensagens para servirem como *input* para outra aplicação.

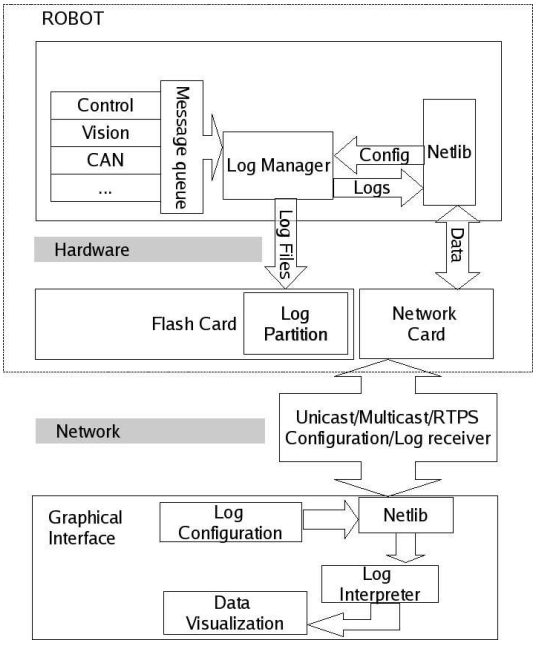


Figura 4.4: Arquitectura de *logs* e respectiva ligação à interface

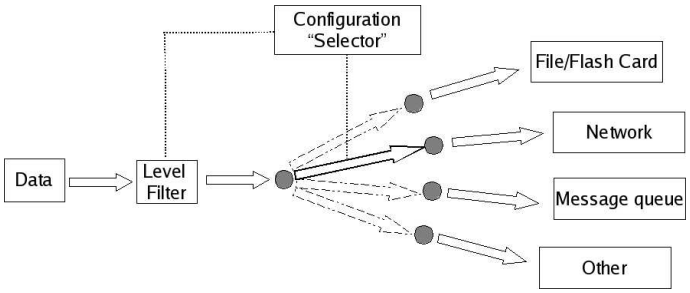


Figura 4.5: Fluxo dos dados para registo

(Esta página foi intencionalmente deixada em branco.)

Capítulo 5

Interfaces de supervisão homem-máquina

Conteúdo

5.1 Desenvolvimento das interfaces de supervisão

5.1.1 GTK+ *graphical toolkit*

Os ambientes gráficos foram criados usando a *toolkit* GTK+ [49]. O GTK+ é uma das ferramentas mais populares na criação de interfaces gráficas para o sistema operativo Linux. Todavia, desde há alguns anos a esta parte que esta ferramenta se tornou multi-plataforma, podendo correr em ambientes como o Windows®. Apesar de ser escrita completamente em linguagem C, o GTK+

fornece uma API em várias linguagens sendo as mais comuns o C, o C++, o C# e o Python. Um dos principais factores que levaram à escolha do GTK+, além de ser multi-plataforma e ter uma API para diferentes linguagens, foi o facto de ter uma licença GNU LGPL. A geração do sistema de janelas da interface foi criada, em parte, pelo software Glade [50].

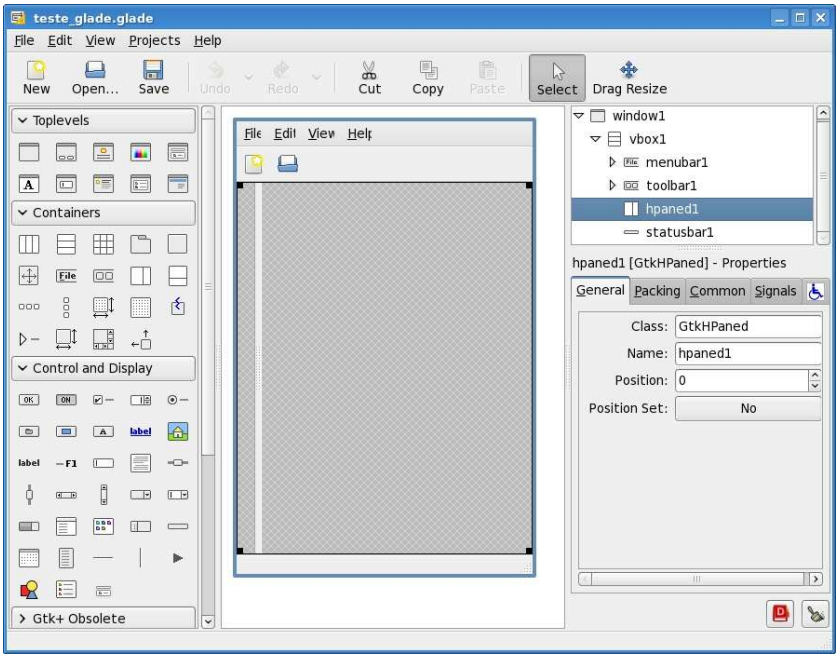


Figura 5.1: Interface de desenvolvimento Glade

O sistema operativo usado no desenvolvimento e teste das interfaces de supervisão foi a distribuição de Linux Fedora 7, sendo possível, no entanto, o uso de outra distribuição qualquer com X (ambiente gráfico) e com o ambiente de desenvolvimento GTK+.

As versões das principais bibliotecas são as seguintes:

- gcc 4.1.2
- glib 1.2.10
- Kernel 2.6.23.17-88.fc7
- GTK+ 2.0

5.1.2 Arquitectura das aplicações desenvolvidas

Fazer uma abstracção completa das aplicações para as quais seriam desenvolvidas as interfaces, iria fazer com que o produto final não fosse utilizável. Optou-se então por desenvolver um modelo de aplicação genérico e de seguida, construir pequenos blocos em que o funcionamento fosse idêntico para as várias interfaces de operação e supervisão, tais como: as comunicações, o sistema de *logs*, o sistema de alertas, etc. A parte da visualização de dados geoespaciais sofreu algumas alterações entre aplicações já que em, alguns casos, pretende-se fazer representações em coordenadas de GPS e noutros usando distâncias em metros.

De seguida serão referenciadas algumas características de funcionamento das interfaces de supervisão em função dos três casos de estudo.

5.1.3 Interface da equipa de futebol: *Ipshell*

Como a plataforma de teste é a equipa de futebol robótico é necessário ter sempre presente que a interface terá de supervisionar todos os jogadores, o treinador e o árbitro. Além disto, é indispensável que a interface consiga ter modos de operação quando se está na fase de desenvolvimento e quando se pretende configurar as estratégias de controlo. Na Figura 5.2 é possível observar o posicionamento dos robôs enquanto carregam baterias num intervalo de um jogo. Neste modo de funcionamento é possível observar perfeitamente quantos jogadores estão em jogo, quais são os suplentes e quantos estão *offline* (i.e. em reparação ou simplesmente desligados).

Ferramentas de teste, *debug* e supervisão

As ferramentas de teste e *debug* na IPShell estão maioritariamante relacionadas com a visão, o autómato de jogo, os barramentos de comunicação internos ao robô e o link de comunicações externo. Relativamente ao autómato de jogo, são disponibilizados um conjunto de opções para mudar o seu estado interno, seja, em posição no campo, papel de jogador, coordenação de jogadas e se o jogador está em jogo ou fora de jogo (por exemplo em reparação). Todo este conjunto de informação é apresentado sob forma de tabela, esta informação serve apenas para ser usada *offline* ou para gerar gráficos em tempo real (por exemplo, os

Figura 5.2: Posicionamento do robôs durante um intervalo no ROBOCUP 2006 em Bremen, Alemanha

gráficos da estimativa do erro da posição do robô). Durante um jogo, ou mesmo durante a fase de teste e *debug*, torna-se complicado observar e compreender, em tempo útil, os dados provenientes dos robôs. Como tal, adoptou-se uma forma de representar a informação mais relevante de forma gráfica de modo a que o supervisor possa identificar claramente, por exemplo, se o jogador se encontra ou não funcional, se não é usado por opção, se está apenas como suplente ou se está a cumprir um tempo fora de jogo por expulsão, Figura 5.3.

A Figura 5.3 representa a forma como é mostrada a informação sobre o robô que é independente da sua posição em campo. Na distinção do estado do jogador é usado ainda um código de cores e o seu posicionamento em determinadas zonas do campo conforme mostram a Figura 5.3 no caso da posição da bola vista por cada um.

Em termos de *debug*, pode ainda referir-se o erro de posição (confiança do robô) relativamente à sua posição actual. Na Figura 5.3 podem ver-se representadas as elipses à volta dos robôs que mostram o erro associado, em X e Y.

Com a excepção do último ano, em que a definição das balizas e cores do equipamento da equipa é realizado através de uma ferramenta controlada pelo árbitro, eram os operadores das equipas que forneciam esta indicação aos robôs. Sendo esta informação vital para o sucesso do jogo, já que, com a cor da baliza trocada poderiam surgir auto golos, dotou-se a interface com avisos gráficos bastante relevantes que deixavam de alertar o operador quando as configurações estivessem bem definidas (faixa vermelha que se vê na Figura 5.3).

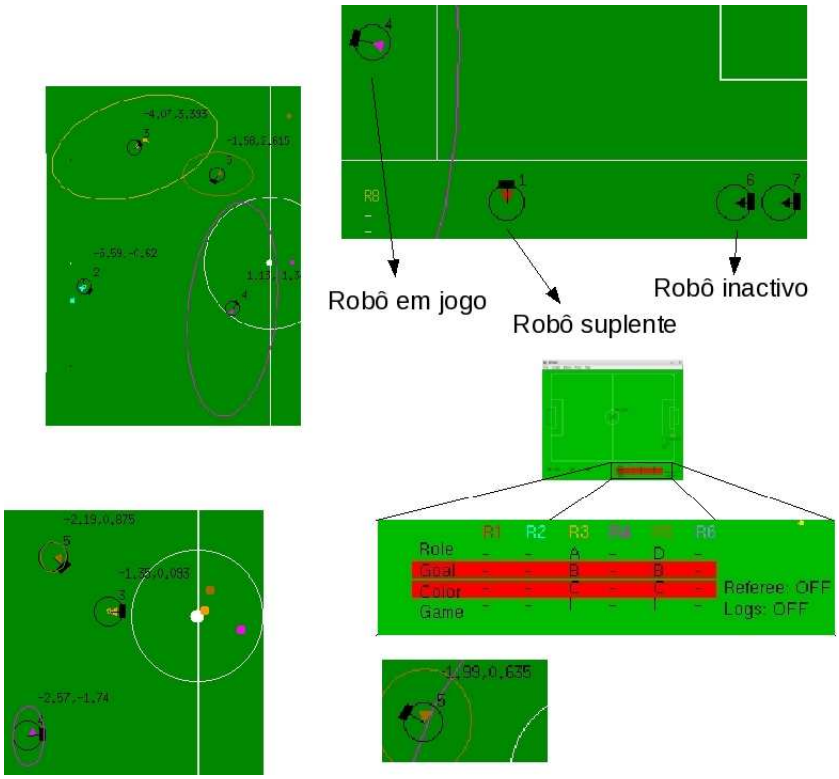


Figura 5.3: Informação visual usando código de cores

Quando é necessário monitorizar dados em tempo real respeitantes ao *hardware*, em que a informação é recebida a uma taxa muito elevada, optou-se por mostrar tabelas com actualizações constantes da informação. Estas actualizações podem ser *time triggred* ou *event triggered* conforme o tipo de dados. Um exemplo prático da necessidade de ver os dados a fluir em tempo real para detecção de algum problema é o das placas de controlo dos motores e a mo-

nitorização da temperatura por parte de quem está a desenvolver a aplicação de controlo. Neste caso ficamos com uma simples janela onde os valores são visualizados conforme se mostra na Figura 5.4.

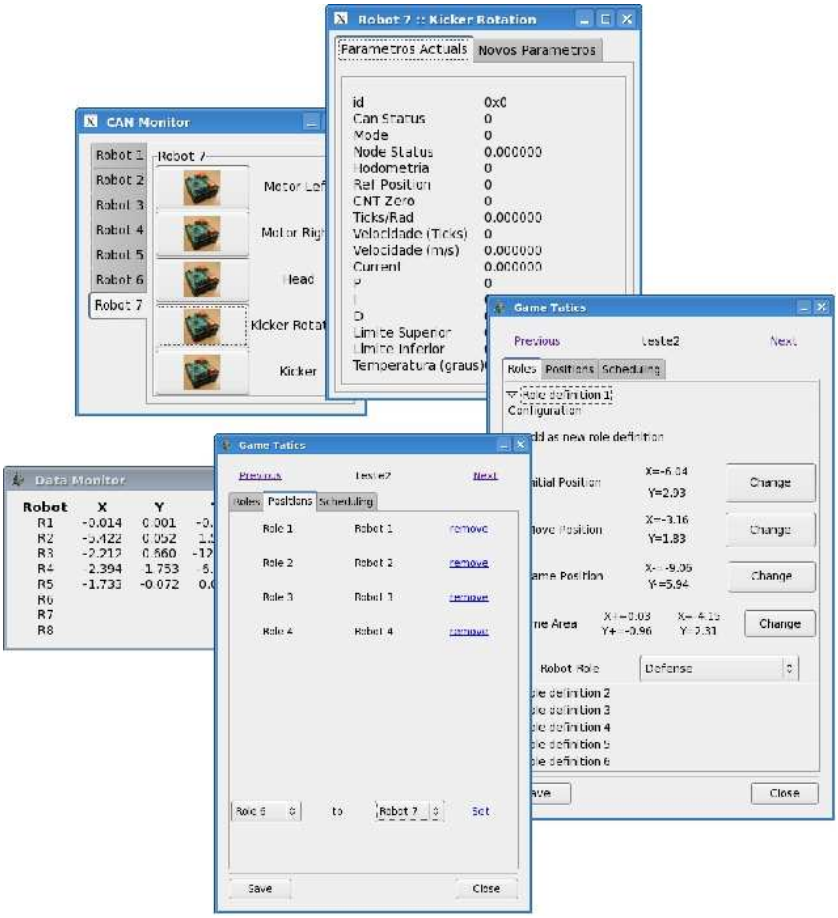


Figura 5.4: Janelas de ferramentas e debug.

Sendo a visão um dos blocos principais da equipa, também esta foi dotada de ferramentas que permitam uma interpretação mais rápida do que se está a passar. Foram implementadas as funcionalidades de visualização das imagens das câmaras dos robôs (duas), com os respectivos controlos, onde é possível ver a imagem real e a imagem processada (Figura 5.5).

Relativamente à visualização gráfica em tempo real, a interface foi dotada de opções que permitem ligar/desligar a visualização de todos os objectos que os robôs vêm, a bola, os outros elementos da equipa, os adversários, as linhas

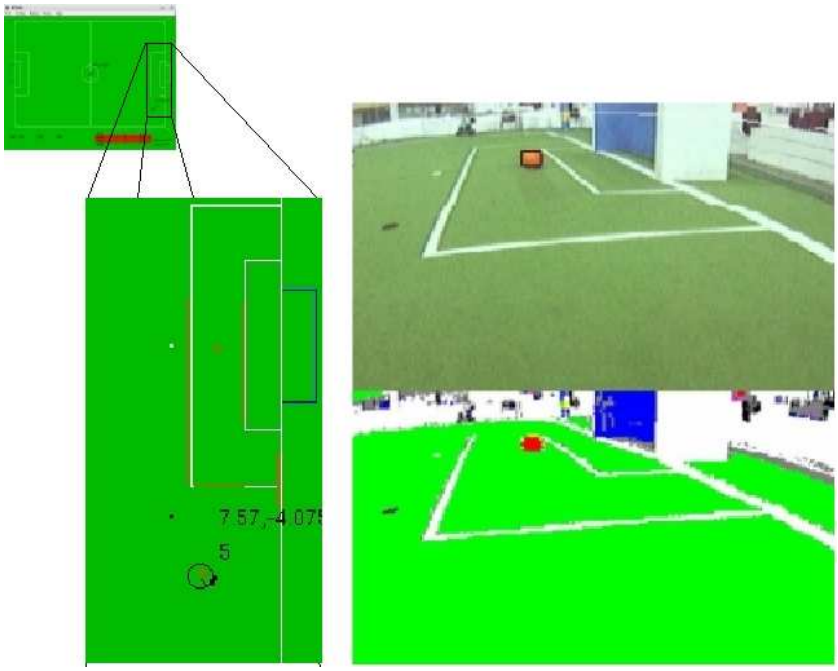


Figura 5.5: Desenho das linhas de área do robô na interface juntamente com uma imagem obtida de umas das câmaras

do campo e as balizas. Esta funcionalidade permite que o operador possa alocar um determinada largura de banda do *link* de comunicações e, em função disso, percepcionar em tempo real a calibração física do robôs. A Figura 5.5 representa o desenho em tempo real das linhas de grande e pequena área relativamente ao robô juntamente com uma imagem obtida por uma das câmaras.

Configuração da estratégia de controlo da equipa

A figura 5.6 representa a forma como é configurada a estratégia de controlo da equipa.

A configuração não é realizada directamente em função dos jogadores, é configurado em primeiro lugar os papéis que se pretendem que existam na equipa, de seguida é atribuído cada um dos papéis a um jogador que melhor se adeqúe ao papel tendo em conta a sua disponibilidade. Durante o jogo, caso o jogador seja expulso ou fique sem funcionar, o treinador, seguindo uma hierarquia configurada previamente, troca esse mesmo papel para outro jogador. A Figura 5.6

Figura 5.6: Configuração de uma jogo de futebol

mostra o aspecto da ferramenta de selecção e configuração de papéis e atribuição dos mesmos aos jogadores.

5.1.4 Interface do ROAZ

A interface de operação foi testada em cenários de aplicação reais. De seguida são descritas três das missões realizadas.

Recolha de dados no rio Tua

Durante a recolha de dados dados batimétricos no rio Tua a interface, Figura5.7, serviu de suporte à teleoperação, navegação e supervisão do ROAZ, bem como de recolha de dados batimétricos.

Testes do ROAZ com radar numa missão conjunta com a a Empresa "CriticalSoftware"

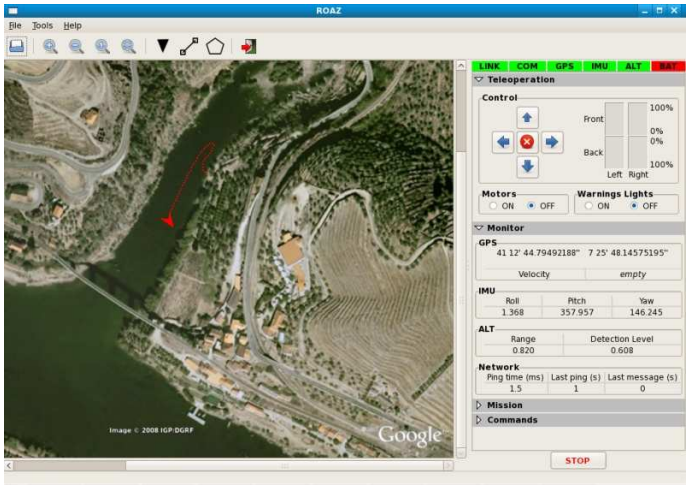


Figura 5.7: Recolha de dados do rio Tua - representação do local de passagem do ROAZ

O objectivo da missão no Porto de Leixões foi o de fazer a detecção de obstáculos no mar (outros barcos) recorrendo a um radar montado na estrutura do ROAZ. A interface (Figura 5.8) serviu mais uma vez de apoio às operações.

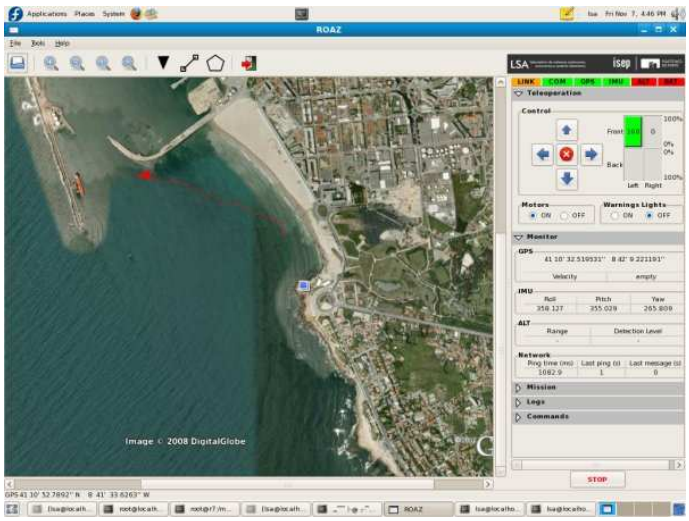


Figura 5.8: Porto de Leixões - testes com radar

Como acréscimo, e por forma a diminuir toda a logística deste tipo de

operações, foi ainda testada com um computador de bolso, o Nokia N770, Figura 5.9 e N800, Anexo D. O mapa representado na fotografia é o Porto de Leixões. A aplicação a correr neste modelo é a mesma apresentada na consola principal.

Figura 5.9: Interface a correr num Nokia N770

Recolha de dados batimétricos no Oceano Atlântico

A missão de recolha de dados na costa norte portuguesa foi feita já de uma forma autónoma por parte do ROAZ. Durante a missão o mar apresentava ondas com cerca de 1-2 metros e um período de 15 segundos. Esta missão revelou que, usando um gestor de missões, este permite tornar o erro introduzido na teleoperação bem menor. As linhas a vermelho, na Figura 5.10, mostram os perfis que serão alvo de estudo. O ROAZ tem a sua trajectória marcada a tracejado.

Na Figura 5.11 é possível observar um excerto de uma trajectória (topo) juntamente com a sua representação na interface. A interface está dotada de definição exacta de *waypoints* bem como a informação da orientação do veículo durante a missão.

O gráfico da Figura 5.12 traduz o resultado final obtido com todos os subsistemas integrados a funcionar. Os quatros pontos correspondem a uma manobra

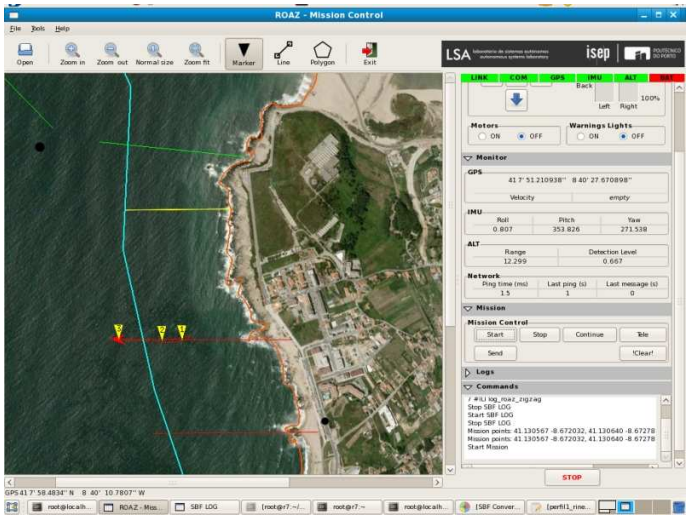


Figura 5.10: Interface com a representação de alguns perfis batimétricos

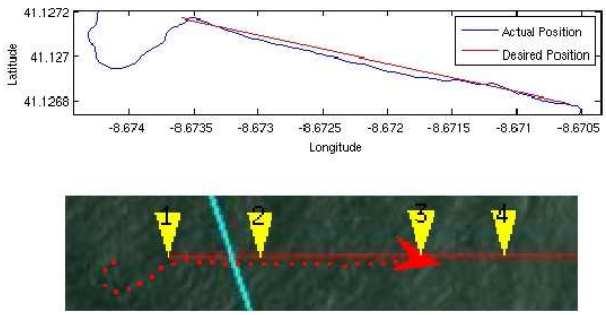


Figura 5.11: Missão autónoma com 4 *waypoints* e respectiva imagem da interface.

composta. O valor de *threshold* é de 10 metros indicados pelos círculos representados no mesmo referencial.

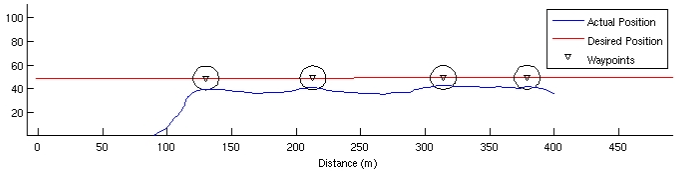


Figura 5.12: Manobra de 'Lista de *Waypoints*' com *threshold* de 10 metros

5.1.5 Interface do *wireless positioning system*

Wireless Positioning System

A interface, Figura 5.13, integra um produto que permite fazer o mapeamento de potências de sinais de *access points* dentro de edifícios, para serem posteriormente usados na localização de pessoas e objectos dentro desses mesmos edifícios.

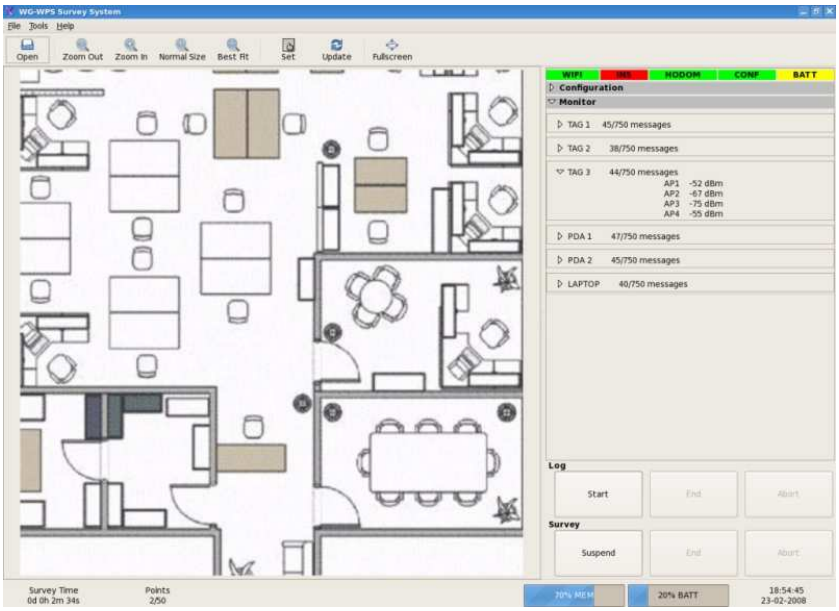


Figura 5.13: Aspecto da interface usada no *Wireless Positioning System*

5.1.6 Detalhes de implementação

A base de trabalho, em termos gráficos, é relativamente idêntica entre as aplicações do ROAZ e do *Wireless Positioning System*. Quanto ao ISePorto, apesar de bastante diferente em termos de aspecto, mantêm-se em comuns grande parte das características, das quais se podem então destacar:

- Representação dos blocos de funcionamento tipo semáforo (funciona/não funciona);
- Informação de rede actualizada para missões críticas;
- Informação genérica do estado actual do veículo na interface principal;

- Possibilidade de ver informação específica do estado actual do veículo;
- Forma de abortar toda a missão, ou a manobra actual de uma forma rápida;
- Possibilidade de registar valores importantes, que tenham altos débitos, em ficheiro;
- Sistema de alertas de problemas críticos bem visível;
- Representação colorida em função do número de veículos;

De seguida, e baseada na interface do ROAZ, Figura 5.14, podem verificar-se onde se encontram os comandos associados a cada um dos item referidos anteriormente.

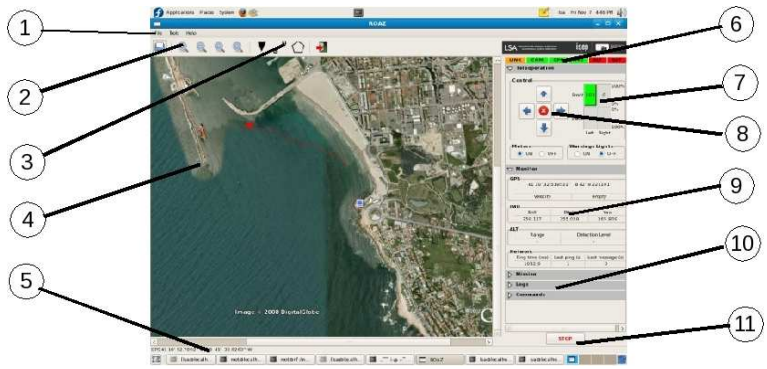


Figura 5.14: Detalhes de implementação da interface WPS/ROAZ

- 1 - Barra de Menus. Esta barra tem os menus típicos de gestão da aplicação relativamente ao Sistema Operativo, as ferramentas que possam ser integradas tais como o Gestor de Missões, opções de *debug* e acesso às janelas de dados que não são importantes durante a execução de uma missão.
- 2 - Barra de ferramentas rápidas do projecto/mapa. Este menu é um menu rápido com opções de projecto ou mapa tais como: abertura de projecto e *zoom in/zoom out* de mapa.
- 3 - Barra de ferramentas da missão. Opções de uso rápido na criação de uma missão.

- 4 - Zona de mapa de missão. Este mapa deverá georeferenciar a posição do veículo podendo ser configurável consoante a informação enviada, medidas em metros, GPS, etc. Em alternativa poderá contar com uma janela de um *link* de vídeo proveniente do veículo ou com desenho 2D/3D da atitude do veículo.

- 5 - Barra de estado. Esta barra tem como função disponibilizar pequenas mensagens não muito importantes para a missão.

- 6 - Barra de informação visual. Esta barra disponibiliza em tempo real informação de forma visual sobre o estado de alguns blocos do veículo.

- 7 - Barra de informação do veículo com informação de auxílio à teleoperação.

- 8 - Menu de teleoperação. Este bloco pode variar consoante o veículo ou não existir caso não seja necessário.

- 9 - Bloco com informação genérico de sensores. Este bloco não é necessário, no entanto pode funcionar também como apoio à teleoperação ou ao *debug* do código.

- 10 - Blocos genéricos com outro tipo de informação.

- 11 - Controlo para imobilizar o veículo.

5.1.7 Missões onde foram usadas as interfaces

As interfaces foram usadas em algumas missões realizadas em ambiente real. Relativamente à IPShell, que é usada desde o ROBOCUP 2004, ainda que não totalmente desenvolvida na altura, até à última competição, o ROBOCUP 2009 em Graz, Áustria, pode dizer-se que foi o foco central de desenvolvimento da maior parte de módulos para os outros casos de estudo.

Listam-se de seguida todas as missões onde se tem usado as interfaces de supervisão (algumas delas são competições nacionais e internacionais de robótica e outras são cenários operacionais descritos nos casos de estudo).

- ROBOCUP 2004, 2006 e 2009;

- German Open 2004, 2005, 2008;
- Festival Nacional de Robótica 2004, 2005, 2006, 2007, 2008 e 2009;
- Manobras de docagem de submarinos no Rio Douro 2006 e 2007;
- Testes e experiência no Porto de Leixões (várias missões), 2007 e 2008;
- Recolha de dados batimétricos do Rio Tua, Julho 2008;
- Protótipo de produto "Wireless Positioning System"desenvolvido para uma empresa, 2008;
- Recolha de dados batimétricos na costa norte portuguesa, Dezembro 2008.

(Esta página foi intencionalmente deixada em branco.)

Capítulo 6

Conclusões e Trabalho Futuro

Nesta dissertação apresenta-se um modelo que permitiu abordar de forma uniformizada a interacção homem-sistema autónomo. A metodologia para síntese e definição do modelo referido parte da análise das aplicações, dos diferentes estágios de desenvolvimento, bem como dos elementos estruturantes. A metodologia seguida conduziu à definição de uma arquitectura de interacção homem-robô que abrange sistemas com diferentes níveis de autonomia. Caracteriza-se e avalia-se diferentes cenários e modeliza-se com sucesso diferentes modos de interacção. Em consequência sistematiza-se os mecanismos de supervisão, de onde resultaram propostas implementadas de supervisão. De notar que no caso do futebol robótico, a supervisão foca o estado do robô e no caso da batimetria a supervisão foca o estado da missão. A ortogonalidade dos dois cenários, permitiu concluir que, no caso dos sistemas autónomos, assumem igual importância e devem ser analisadas da mesma forma, as características intrínsecas (robô) e as extrínsecas (missão) para o bom desempenho da aplicação.

Desenvolveu-se diferentes e variadas interfaces gráficas como elementos integrantes da interacção homem-sistema autónomo. Deste trabalho resultou a implementação do sistema de supervisão da equipa de futebol robótico ISePorto, e do sistema de batimetria com o veículo ROAZ.

Os resultados obtidos confirmaram que no actual nível de desenvolvimento,

os paradigmas de supervisão tendo em conta a iniciativa mista, promovem soluções mais versáteis e operacionalmente mais eficazes.

Relativamente ao trabalho futuro, é essencial contribuir para o estabelecimento de um conjunto de ferramentas de desenvolvimento e implementação baseada numa modelização única, bem como na criação de estruturas abstractas de supervisão que garantam que a introdução do homem no ciclo de controlo permita a utilização de sistemas robóticos em ambientes mais complexos sem perda de propriedades como a segurança e *liveness*.

Bibliografia

[1] R. Brooks, "A robust layered control system for a mobile robot," pp. 14–23, IEEE Journal of Robotics and Automation, 1986.

[2] J. Rosenblatt, *DAMN: A Distributed Architecture for Mobile Navigation*. PhD thesis, Carnegie Mellon University, 1997.

[3] R. Alami, R. Chatila, S. Fleury, M. Ghallab, and F. Ingr, "An architecture for autonomy," *International Journal of Robotics Research*, vol. 17, pp. 315–337, 1998.

[4] R. Simmons, T. Smith, M. B. Dias, D. Goldberg, D. Hershberger, A. Stentz, and R. Zlot, "A layered architecture for coordination of mobile robots," In Multi-Robot Systems: From Swarms to Intelligent Automata, 2002.

[5] L. E. Parker, "Allience: An architecture for fault tolerant multi-robot cooperation," pp. 220–240, IEEE Transactions on Robotics and Automation, 1998.

[6] P. Pirjanian, T. L. Huntsberger, A. Trebi-Ollennu, H. Aghazarian, H. Das, S. S. Joshi, and P. S. Schenker, "Campout: A control architecture for multi-robot planetary outposts," SPIE Conf. Sensor Fusion and Decentralized Control in Robotic Systems III, 2000.

[7] M. A. G. Jacob W. Crandall, "Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation," 2002.

[8] K.-F. Kraiss, "Advanced man machine interaction," in *Advanced Man Machine Interaction* (Springer, ed.), pp. 9–18, 2006.

[9] M. G. J. Crandall, "Measuring the intelligence of a robot and its interface," 2003.

[10] "http://path.berkeley.edu/smart-ahs/index.html," Consultada em Outubro de 2008.

[11] "http://playerstage.sourceforge.net/," Consultada em Setembro de 2009.

[12] "http://www.uppaal.com/," Consultada em Setembro de 2008.

[13] "http://embedded.eecs.berkeley.edu/research/hytech/demo.html," Consultada em Setembro de 2009.

[14] "http://www.orocos.org/," Consultada em Setembro de 2009.

[15] "http://www.ocera.org/," Consultada em Setembro de 2009.

[16] "http://marsprogram.jpl.nasa.gov/default.html," Consultada em Setembro de 2009.

[17] "http://www.darpa.mil/grandchallenge/," Consultada em Julho de 2009.

[18] S. Team, "Smart vehicle challenge problems.," vol. 14, pp. 143–148, In support of the University California Open Experimental Platform for DARPA - MoBIES, 2002.

[19] "http://www.elrob.org/," Consultada em Outubro de 2009.

[20] "http://www.robocup.org," Consultada em Setembro de 2009.

[21] "http://www.zarya.info/diaries/luna/luna17.php," Consultada em Outubro de 2009.

[22] "http://www.airforce-technology.com/projects/predator/," Consultada em Setembro de 2009.

[23] K. S. Kwok, "Research on the use of robotics in hazardous environments at sandia national laboratories," 1999.

[24] J. A. Alfredo Martins, C. Almeida, A. Figueiredo, F. Santos, D. Bento, H. Silva, and E. Silva, "Forest fire detection with a small fixed wing autonomous aerial vehicle," Proc. IAV'07 International Autonomous Vehicles Conference, 2007.

[25] E. A. P. da Silva, *Controlo de Veículos Autónomos*. PhD thesis, Faculdade de Engenharia do Porto, 2002.

[26] J. Bengtsson, K. G. Larsen, F. Larsson, P. Pettersson, and W. Yi, "Uppaal a tool suite for automatic verification of real-time systems.," In Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey), Washington, 1995.

[27] T. A. Henzinger, "The fresco project: Formal real-time software components," Keynote lecture, First Workshop on Models for Time-critical Systems (MTCS), 2000.

[28] R. Alur, A. Henzinger, F. Mang, S. Qadeer, K. Rajamani, and S. Tasiran, "Mocha: Modularity in model checking," vol. 1427, pp. 521–525, In Proceedings of the Tenth International Conference on Computer-aided Verification (CAV 1998), Lecture Notes in Computer Science, 1998.

[29] D. Simon, B. Espiau, K. Kapellos, and R. Pissard-Gibollet, "Orccad: Software engineering for real-time robotics," vol. 15, pp. 111–116, A Technical Insight, Robotica, Special issues on Languages and Software in Robotics, 1997.

[30] D. MacKenzie, R. Arkin, and J. Cameron, "Multiagent mission specification and execution," vol. 15, pp. 29–52, utonomous Robots.

[31] R. Alur, R. Grosu, Y. Hur, V. Kumar, and I. Lee, "Modular specification of hybrid systems in charon," Proceedings of the 3rd International Workshop on Hybrid Systems: Computation and Control, Pittsburgh, PA, 2000.

[32] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Times - a tool for modelling and implementation of embedded systems," pp. 460–464, In proceedings of 8th International Conference, TACAS, part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2002, 2002.

[33] P. S. Dias, G. M. Gonçalves, F. L. Pereira, J. Pinto, R. Gonçalves, and J. B. Sousa, "Neptus, command and control infrastructure for heterogeneous teams of autonomous vehicles,"

[34] A. O. Alberto Finzi, "A mixed-initiative approach to human-robot interaction in rescue scenarios," Automated Planning and Scheduling (ICAPS), 2005.

[35] D. J. Bruemmer, J. L. Marble, D. D. Dudenhoeffer, M. O. Anderson, and M. D. Mckay, "Mixed-initiative control for remote characterization of hazardous environments," In Proceedings of the Hawaii International Conference on System Sciences, Waikoloa, 2003.

[36] J. Wang and M. Lewis, "Mixed-initiative multirobot control in usar," in *Human-Robot Interaction* (I.-T. Education and A. Publishing, Vienna, eds.), pp. 405–422, 2007.

[37] M. Linegang, C. Haimson, J. MacMillan, and J. Freeman, "Human control in mixed-initiative systems: lessons from the mica-sharc program," vol. 1, pp. 436– 441, IEEE International Conference on Systems, Man and Cybernetics, 2003., 2003.

[38] E. DeKoven and A. K. Murphy, "A framework for supporting teamwork between humans and autonomous systems," Command and Control Reseach Program, 2006.

[39] D. J. Bruemmer, D. A. Few, C. W. Nielsen, and M. C. Walton, "Mixed-initiative control for collaborative countermine operations," IEEE Transactions on Robotics, 2007.

[40] J. Bresina, A. Jónsson, P. Morris, and K. Rajan, "Mixed-initiative activity planning for mars rovers," IEEE Transactions on Robotics, 2004.

[41] J. Almeida, A. Martins, E. Silva, L. Lima, C. Almeida, N. Dias, and H. Silva, "Iseporto robotic soccer team for robocup 2009:improving perception.," Robocup 2009, Team Description paper., 2009.

[42] A. Martins, H. Ferreira, C. Almeida, H. Silva, J. M. Almeida, and E. Silva, "Roaz and roaz ii autonomous surface vehicle design and implementation," International Lifesaving Congress, 2007.

[43] T. MoBIES, "Report on verification of the mobies vehicle-vehicle automotive oep problem," 2002.

[44] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Middleware for robotics: A survey," The IEEE Intl. Conf. on Robotics, Automation, and Mechatronics (RAM 2008), 2008.

[45] T. Fong and C. Thorpe, "Vehicle teleoperation interfaces," in *Autonomous Robots 11*, pp. 9–18, 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), 2001.

[46] V. Cerqueira, A. Dias, N. Matos, J. M. Almeida, A. Martins, and E. P. Silva, "Iseporto robotic soccer team: A new player generation," vol. 14, pp. 143–148, Proceedings of the Scientific Meeting of the Portuguese Robotics Open 2004, FEUP Edições, Colecção Colectâneas, 2004.

[47] ROBOCUP, "In http://www.www.robocup.org," Consultada em Setembro de 2009.

[48] E. Horvitz, "Principles of mixed-initiative user interfaces," pp. 159–166, Proceedings of CHI '99, ACM SIGCHI Conference on Human Factors in Computing Systems, 1999.

[49] "http://www.gtk.org/," Consultada em Setembro de 2009.

[50] "http://glade.gnome.org/," Consultada em Setembro de 2009.

[51] J. Kapinski and B. H. Krogh, "A new tool for verifying computer controlled systems," pp. 98–103, IEEE Conference on Computer-Aided Control System Design, 2002.

[52] Y. Hur and I. Lee, "Distributed simulation of multi-agent hybrid systems," IEEE International Symposium on Object-Oriented Real-time distributed Computing (ISORC), 2002.

[53] J. Kim and I. Lee, "Modular code generation from hybrid automata based on data dependency," In Proceedings of The 9th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2003), Washington, 2003.

[54] M. Lindahl, P. Pettersson, and W. Yi, "Formal design and analysis of a gear controller.," vol. 3, pp. 353–368, 2001.

[55] T. A. Henzinger, "Masaccio: A formal model for embedded components," vol. 1872, pp. 549–563, Proceedings of the First IFIP International Conference on Theoretical Computer Science (TCS), Lecture Notes in Computer Science, Springer-Verla, 2000.

[56] R. Grosu, T. Stauner, and M. Broy, "A modular visual model for hybrid systems," vol. 1486, pp. 75–91, Lecture Notes in Computer Science, 1998.

[57] D. Simon, B. Espiau, E. Castillo, and K. Kapellos, "Computer-aided design of a generic robot controller handling reactivity and real-time control issues," Rapport de Recherche Inria, no 1801, 1992.

[58] R. Alur, C. Courcoubetis, T. Henziger, and P. Ho, "Hybrid automaton: An algorithmic approach to the specification and verification of hybrid systems.," vol. 736, pp. 209–229, 1993.

[59] A. Desphande, *Control of Hybrid Systems*. PhD thesis, Dept. of Electrical Engineering, University of California, Berkley, 1994.

[60] A. Puri, *Theory of hybrid Systems and Discrete Event Systems*. PhD thesis, Dept. of Electrical Engineering, University of California, Berkley, 1995.

[61] B. M. S., *Studies in Hybrid Systems: Modeling, Analysis, and Control*. PhD thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, 1995.

[62] J. Lygeros, *Hierarchical Hybrid Control of Large Scale Systems*. PhD thesis, Dept. of Electrical Engineering, University of California, Berkley, 1996.

[63] W. H.S., "A class of hybrid-state continuous-time dynamic system," pp. 161–167, IEEE Transactions on Automatic Control, 1966.

[64] A. M., "Automaton decompositions and semigroup extensions, in the algebraic theory of machines, language and semigroups," pp. 37–54, M. A. Arbib, (Ed.) Academic Press, 1968.

[65] L. Tavernini, "Difrential automata and their discrete simulator, non linear analysis. theory methods and applications," pp. 665–683, 1987.

[66] A. Nerode and W. Khon, "Model hybrid systems: Automata, topologies, controlability, observability," pp. 317–356, In GROS93, 1993.

[67] P. Antsaklis, J. A. Stiver, and M. Lemon, "Hybrid systems modeling and autonomous control systems," pp. 366–392, In GROS93, 1993.

[68] T. A. Henzinger, "The theory of hybrid automata.," pp. 278–292, Proceedings of the 11th Annual Symposium on Logic in Computer Science (LICS), IEEE Computer Society Press, 1996.

[69] T. Henzinger, "The theory of hybrid automata," pp. 278 – 292, Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS '96), 1996.

[70] "http://rtg.cis.upenn.edu/mobies/charon/thermostat-old.cn," Consultada em Outubro de 2008.

[71] A. Deshpande, A. Gollu, and L. Semenzato, "Shift programming language and run-time system for dynamic networks of hybrid automata," vol. 15, pp. 29–52, PATH Report in http://www.path.eecs.berkeley.edu/shift/publications.html, 1997.

[72] "http://carmen.sourceforge.net," Consultada em Setembro de 2009.

[73] E. Kindler, "Safety and liveness properties: A survey.," 1994.

[74] B. I. Silva, K. Richeson, B. H. Krogh, and A. Chutinan, "Modeling and verification of hybrid dynamical system using checkmate," ADPM, 2000.

[75] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, and W. Yi, "Times: a tool for schedulability analysis and code generation of real-time systems," In Proceedings of the 1st International Workshop on Formal Modeling and Analysis of Timed Systems, FORMATS 2003, Marseille, France, 2003.

[76] T. Henzinger, P.-H. Ho, and H. Wong-Toi, "Hytech: A model checker for hybrid systems," pp. 110–122, Software Tools for Technology Transfer, 1997.

[77] R. Alur, T. Henzinger, and H. Ho, "Automatic symbolic verification of embedded systems," pp. 110–122, IEEE Transactions on Software Engineering, 1996.

[78] H. W.-T. Henzinger, T. A., "Using hytech to synthesize control parameters for a steam boiler. in formal methods for industrial applications: Specifying and programming the steam boiler control.," vol. 1165, pp. 265–282, Lecture Notes in Computer Science, Springer-Verlag, 1996.

[79] P.-H. Ho and H. Wong-Toi, "Automated analysis of an audio control protocol," pp. 110–122, Proceedings of the Seventh International Conference on Computer-aided Verification (CAV 1995), 1995.

[80] E. Asarin, G. Pace, G. Schneider, P.-H. Ho, H. Wong-Toi, and S. Yovine, "Speedi - a verification tool for polygonal hybrid systems," In Proceedings of Computer Aided Verification, CAV'02, Lecture Notes in Computer Science, Springer-Verlag, 2002.

[81] C. Daws, A. Olivero, S. Tripakis, and S. Yovine, "The tool kronos," pp. 23–60, In Proceedings of Hybrid Systems III, Verification and Control. Lecture Notes in Computer Science 1066, Springer-Verlag, 1996.

[82] C. Daws and S. Yovine, "Two examples of verification of multirate timed automata with kronos," n Proceedings of the 1995 IEEE Real-Time Systems Symposium, RTSS'95, Pisa, Italy, December. IEEE Computer Society Press, 1995.

[83] V. Bertin, E. Closse, M. Poize, J. Pulou, J. Sifakis, P. Venier, D. Weil, and S. Yovine, "Taxys = esterel + kronos. a tool for verifying real-time properties of embedded systems," In Proceedings of "Conference on Decision and Control, CDC'01". Orlando, December. IEEE Control Systems Society, 2001.

[84] A. Deshpande and S. Yovine, "The diadem-kronos connection: Bridging the gap between implementation and verification of hybrid systems," Systems V Workshop, Notre Dame, IN. workshop, 1997.

[85] A. Deshpande, "The diadem system for real-time dynamic event managemen," In Proc. NATO Workshop on Discrete Event and Hybrid Systems, May-June, Antalya, Turkey. Madrid, Spain,, Springer-Verlag, 1997.

[86] T. Henzinger, B. Horowitz, and C. M. Kirsch, "Giotto: A time-triggered language for embedded programming,"

[87] T. Henzinger, B. Horowitz, and C. M. Kirsch, "Embedded control systems development with giotto,"

[88] T. Henzinger, C. M. Kirsch, M. A. Sanvido, and W. Pree, "From control models to real-time code using giotto,"

[89] G. Pardo-Castellote, S. Schneider, and M. M. Hamilton, "Ndds: The real-time publish-subscribe middleware," Real-Time Innovations, Inc. Available: http://www.rti.com, 2002.

[90] G. Pardo-Castellote, "Omg data-distribution service: Architectural overview," p. 200, 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), 2003.

[91] P. Smolik, Z. Sebek, and Z. Hanzalek, "Orte-open source implementation of real-time publish-subcribe protocol," pp. 68–72, In 2nd International Workshop on RealTime lans in the Internet Age. Porto: Universidade de Porto, 2003.

[92] M. Matteucci, "Publish/subscribe middleware for robotics: requirements and state of the art," Dipartimento di Elettronica e Informazione - Politecnico di Milano, Milano Italy, 2002.

(Esta página foi intencionalmente deixada em branco.)

Apêndice A

Ferramentas de desenvolvimento

Conteúdo

A.1 Ferramentas de modelização

As diferentes linguagens para modelização dos sistemas híbridos são muitas das vezes desenvolvidas especificamente para a aplicação em causa, no entanto existem algumas que integram ou utilizam ambientes de desenvolvimento. O CheckMate [51] por exemplo usa como ambiente de desenvolvimento o Stateflow e o Simulink do MATLAB para a partir daí proceder à verificação formal. Nestes casos é usada uma aplicação gráfica para criar e editar o modelo do sistema, no entanto, existem ferramentas que permitem a descrição de uma forma textual ou ambientes que permitem o uso das duas formas a gráfica e textual como o caso do CHARON [31, 52, 53] que define *Architectural hierarchy* onde é feita a

descrição do sistema dividido por várias hierarquias, *Behaviour hierarchy* define os diferentes modos dos vários agentes assim como os diferentes sub-modos e as transições entre eles, *Discrete uptdates* são as passagens entre os vários modos e sub modos respeitando as transições especificadas e *Continuous updates* são as variáveis que variam de uma forma continua dentro de cada modo e sub modo [31]. Ou mesmo ferramentas como UPPALL [26, 54] que tem as duas formas de editar os modelos do sistema. O projecto FRESCO [27] usa o Masaccio [55] para a definição formal do sistema. Masaccio é uma ferramenta que permite a descrição de um modelo formal para sistemas híbridos dinâmicos que usa as componentes discretas e contínuas. O HYCharts consiste em dois modelos formais diferentes para definir a arquitectura e comportamento dos sistemas híbridos o HyACharts e o HySCharts [56] sendo cada um deles baseado em *hierarchical graphs* recorrendo ao uso de operadores. A função do HyACharts é a da representação visual dos blocos de estado, o HySChartes é responsável pelas transições e o fluxo de dados entre os blocos de estado. O ambiente de desenvolvimento ORCCAD [57, 29] é constituído por quatro blocos diferentes. A interface gráfica serve para se construir Robot Tasks (RT) e Robot Procedures (RP) a partir de módulos simples.

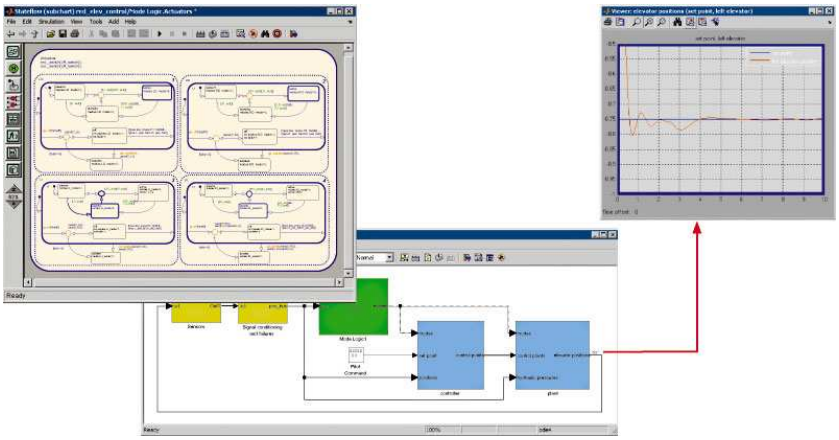


Figura A.1: Ambiente de desenvolvimento *Stateflow*

A.1.1 Formalismos para a modelização de comportamentos

Os fundamentos de sistemas discretos permitem a compreensão das ferramentas estudadas. Este tipo de sistemas permite uma maior redução da complexidade que é proporcionada pelos modelos híbridos pois permite a modelização de sistemas dinâmicos em vários níveis de abstracção. Outra razão é o aparecimento dos computadores no anel de controlo, que promove a interacção de dispositivos digitais com um mundo contínuo. Os sistemas híbridos são sistemas dinâmicos cujo comportamento é determinado pela interacção de dinâmicas contínuas e discretas. Estes sistemas tipicamente são compostos por variáveis e sinais que evoluem num domínio infinito, e variáveis que evoluem num domínio discreto. O estudo de sistemas híbridos como disciplina da área de controlo surge claramente nos anos 90 [58] [59] [60, 61, 62] em sequência de áreas predecessoras como o *sliding mode*, *switch mode control* e controlo digital. No entanto abordagens a sistemas híbridos poderão ser encontrados em anos anteriores [63] e abordagens predecessoras como em Arbib, 1968 [64] Teorias endereçando sistemas híbridos têm surgido recentemente com grande vigor e em consequência se um endereçamento cruzado entre a teoria de controlo e as ciências da computação resultou em diferentes paradigmas matemáticos para a modelização de sistemas híbridos. O trabalho desenvolvido na linha de Tavernini [65] que baseia a modelização de sistemas híbridos em *diferential automata*, Nerode e Khon [66] que fundados na teoria da composição de autómatos apresentaram uma modelo em que equações diferenciais interactuam com autómatos finitos, Antsanklis [67] desenvolveu um modelo baseado na abordagem dos sistemas de acontecimentos discretos, Alur [58] o autómato híbrido como uma extensão do *timed automata*. Com a consolidação da área surgem modelos unificados como em Desphande [59] e Anuj Puri [60, 61, 62]. Tradicionalmente a teoria de controlo baseia-se em mapas contínuos que relacionam as variações na entrada em variações na saída. A extensão desta metodologia foi utilizada no projecto de mapas analógicos-digitais. Branicky (1995) introduziu um modelo unificado, que inclui transições não controladas na dinâmica do sistema, quando este atinge determinadas fronteiras, ou controladas em resposta a uma solicitação de uma acção de controlo. A Figura A.2 representa uma autómato híbrido genérico [68].

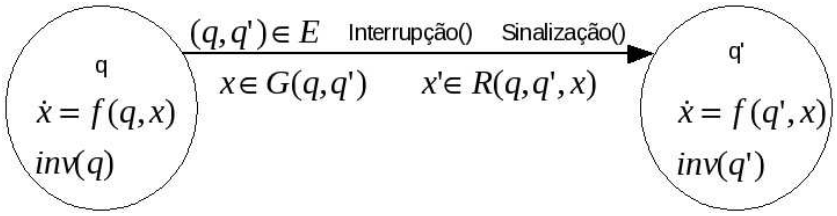


Figura A.2: Autómato híbrido genérico

Um sistema híbrido é um sistema dinâmico de tempo contínuo, caracterizado por estados discretos e transições instantâneas. Os estados discretos são caracterizados por restrições nas trajectórias contínuas identificados na Figura A.2 como inv(q). Estas restrições, designadas normalmente por invariantes, definem o espaço de estado válido para o estado corrente do sistema. As transições podem ocorrer por dois mecanismos: um externo simbolizado pela etiqueta 'interrupção' que permite a sincronização deste autómato, com outro causal da transição; o outro mecanismo de carácter aleatório ocorre quando o domínio da guarda e da invariante se interceptam. Quando o conjunto da intercepção é composto só por um ponto esta transição chama-se emergente. Quanto ocorre uma transição pode-se "emitir"um sinal por forma, a que outros autómatos possam sincronizar e efectuar o "Reset"de uma ou mais variáveis. O mecanismo de sincronização que pode ocorrer é de 1:N.

O Autómato especificado na Figura A.2 é usado essencialmente para criar um modelo matemático para a análise e implementação dos sistemas híbridos. Um autómato híbrido (H) consiste então em $H = (Q, X, Init, f, Inv, E, G, R)$ sendo cado um dos seus componentes o seguinte:

- Variáveis

Existe um número finito de variáveis onde: X é um número de variáveis contínuas

$$X = \{x_1, x_2, ..., x_n\} \tag{A.1}$$

Q é um número de variáveis discretas

$$Q = \{q_1, q_2, ..., q_n\} \tag{A.2}$$

Derivada das variáveis contínuas durante as alterações contínuas

$$\dot{X} = \{\dot{x}_1, \dot{x}_2, ..., \dot{x}_n\} \tag{A.3}$$

Valores das variáveis contínuas na conclusão das alterações discretas:

Variáveis contínuas:

$$X' = \{x'_1, x'_2, ..., x'_n\} \tag{A.4}$$

Variáveis discretas:

$$Q' = \{q'_1, q'_2, ..., q'_n\} \tag{A.5}$$

- Gráficos de controlo

Um gráfico de controlo é representado por (V, E), onde V são os chamados modos de controlo (estados discretos) e o E são alterações de controlo (dinâmica discreta)

- Condições iniciais

$Init \subseteq Q * X$ são os estados iniciais.

- Vectores de campo

$f : Q * X \rightarrow TX$ são os vectores de campo.

- Condições de invariânica

$Inv : Q \rightarrow 2^x$ atribui a cada $q \in Q$ uma lista de invariâncias.

- Condições de salto

$E \subset Q * Q$ lista de transições discretas.

- Guardas

$G : E \rightarrow 2^x$ atribui a cada $e \subset (q, q') \in E$ uma guarda

- Reset

$R : E * X \rightarrow 2^x$ atribui a cada $e \subset (q, q') \in E$ e $x \in X$ uma relação de reset.

Um exemplo simples, é o do termostato. Neste caso [69] pretende-se que a temperatura de uma sala se situe entre os 18 e os 22 graus (inclusive). Para este caso sabe-se que:

- A temperatura inicial é 20 graus.
- Com o termostato a temperatura baixa seguindo a equação: $\dot{x} = -0.1x$.
- Com o termostato temperatura sobe consoante a seguinte equação: $\dot{x} = 5 - 0.1x$.
- O controlador apenas liga ou desliga o termóstato.

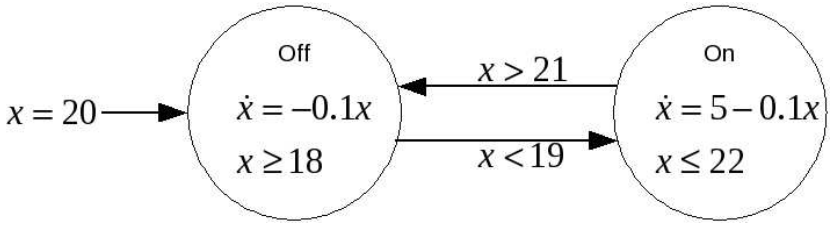


Figura A.3: Termostato representando um sistema discreto

Ficamos então com o seguinte autómato constituído por dois modos on/off cuja transição é discreta (através de saltos entre os modos) e com a dinâmica contínua de alteração da temperatura (fluxo).

- Variável contínua, $X = \{x\}, x \in \Re$ (temperatura neste caso)
- Variável discreta, $Q = \{q\}, q \in \{ON, OFF\}$
- Estados inicias, $Init = Q * X$
- Vectores de campo, $f(q,x) = \begin{cases} -0.1x \ if \ q == OFF \\ 5 - 0.1x \ if \ q == ON \end{cases}$
- Invariância, $Inv(q) = \begin{cases} \{x \in \Re : x \geqslant 18\} \ if \ q == OFF \\ \{x \in \Re : x \leqslant 22\} \ if \ q == ON \end{cases}$
- Condições de guarda, $G(e) = \begin{cases} \{x \in \Re : x = 18\} \ if \ e == (ON, OFF) \\ \{x \in \Re : x = 22\} \ if \ e == (ON, OFF) \end{cases}$
- Transições discretas, $E = \{(ON, OFF), (OFF, ON)\}$
- Reset, R(e,x)=x

As ferramentas de desenvolvimento de sistemas híbridos, que implementam os autómatos híbridos descritos anteriormente, podem dividir-se em quatro categorias:

- Modelização;
- Simulação;
- Verificação;
- Implementação.

A.2 Ferramentas de simulação

A robótica e os sistemas multi-robóticos têm-se caracterizado por permitirem a execução de tarefas em cenários reais, com robôs reais, havendo sempre uma grande interacção com todos os objectos envolventes no mundo. O desenvolvimento de um sistema robótico, apenas com um robô ou com uma equipa de robôs, levanta sempre grandes questões na fase inicial de desenvolvimento, mais ainda quando de trata de uma grande equipa de robôs. O custo total, o espaço necessário para testes, a mão de obra e o tempo necessário são uma desvantagem para a discussão da construção sistema de um sistema robótico. Toda esta logística torna menos atraente o objectivo final. Tipicamente começa-se por desenvolver um protótipo de um robô que, depois de validado, passa para uma produção em série por forma a criar a equipa de robôs. No caso da uma equipa heterogénea de robôs tudo fica ainda mais complicado. Nos sistemas com apenas um robô o problema pode surgir na colocação/disposição dos sensores ou mesmo em termos da forma mecânica do chassi. O problema da fase de testes de um sistema multi-robótico centra-se na logística. Ter uma equipa de robôs completamente funcional para a execução de uma manobra pode tornar-se numa tarefa muito complicada. O espaço é outro factor importante, nem sempre existe a disponibilidade de espaço suficiente para efectuar testes. O local especifico para a execução das manobras pode ser longínquo ou inacessível durante os testes. Encontram-se exemplos muito facilmente nos robôs que são usados na inspecção de outros planetas, que efectuam missões em locais perigosos ou em locais com ambientes imprevisíveis à partida [40, 35]. A acrescentar a este

facto, no caso de uma equipa de robôs, nem sempre é possível ter disponível toda uma equipa para efectuar os testes.

O desenvolvimento do *software*, e os testes necessários ao controlo e coordenação, também são evidentes. Não é praticável, sempre que se ajusta um diferente parâmetro, ter a equipa funcional para fazer a validação desse mesmo parâmetro. Por vezes, um pequeno ajuste necessitaria de horas de teste. Mesmo antes do sistema existir fisicamente, a validação do código de controlo e coordenação, deveria ser possível.

As ferramentas que têm vindo a ser desenvolvidas permitem, essencialmente, efectuar verificações lógicas de uma determinada arquitectura. O grande ênfase que tem sido atribuído aos sistemas multi-robóticos tem passado pelo desenvolvimento, de uma forma conceptual, de arquitecturas de controlo e coordenação. O objectivo é fazer com que seja possível realizar determinadas tarefas, recorrendo a equipas de robôs, em ambientes completamente inconstantes. Conseguir cumprir missões em ambientes que mudam constantemente, sem se ter conhecimento prévio do mundo, tem sido uma das áreas à qual se tem dado muita atenção.

A simulação de robôs tem, hoje em dia, sido alvo de grande desenvolvimento, quer pela simulação de um robô, quer pela simulação de pequenas parte do robô, seja ela, sensores, dinâmica ou *hardware*. A evolução na simulação de sistemas multi-robóticos tem passado por várias fases. A simulação pura da dinâmica do robô tem sido modificada por algo mais abrangente. As ferramentas que têm surgido [36, 11], permitem fazer a simulação de várias equipas de robô, simular sensores individualmente, simular equipas de robô virtuais com robô reais, acrescentar partes simuladas ao robô real, comunicação com todos os robô, a existência vários níveis de abstracção do *hardware* e a simulação usando o *hardware* do robô real. A simulação na robótica, permite resolver os problemas referidos acima, com a capacidade de cálculo de um simples computador já é possível fazer simulações bastante fidedignas e com grande qualidade.

O teste de toda a arquitectura através da simulação que permita o desenvolvimento de protótipos de robôs de uma forma simples, clara e de uma forma abstracta do tipo de robô e aplicação assim como o teste de múltiplos agentes sem ter que recorrer ao *hardware* específico torna-se útil quando de trata

de grandes equipas de robôs em termos de espaço-tempo. A simulação pode no entanto tornar-se complexa quando se fala de termos de simulação de sensores, actuadores e dinâmica de um veículo com grande exactidão. Não se pretende uma aproximação realista mas uma aproximação que permita tirar conclusões sobre que tipo de problemas com que nos possamos deparar na realidade. Além disso, uma simulação que permita uma reutilização do código directamente sobre a arquitectura real de controlo tornam-se imperativos. O núcleo da simulação, ao implementar o autómato de controlo usado nos veículos reais, permite uma maior versatilidade assim como a minimização do tempo de desenvolvimento. No entanto, o conjunto de ferramentas a desenvolver torna-se maior em simulação:

Em traços gerais a simulação de ambientes robóticos servem para:

- Simulação de diferentes arquitecturas para controlo e coordenação numa mesma ferramenta;
- Evitar a existência de múltiplos robôs quando se trabalha com equipas de robôs;
- Simulação de cenários aos quais é impossível ou difícil aceder;
- Resolução dos problemas construtivos dos robôs, (p.e. disposição de sensores);
- Desenvolvimento mais rápido de *software*.

Principais problemas existentes na robótica quando se usam equipas de robôs:

- Cenários aos quais é impossível ou difícil aceder: Marte, locais perigosos, etc;
- Necessidade da existência de vários robôs;
- Desenvolvimento do código fica muito dificultado quando se precisam e fazer testes, que seriam impossíveis se não fosse de uma forma simulada;
- Obrigação de construção de cenários reais.

As mais valias da simulação são:

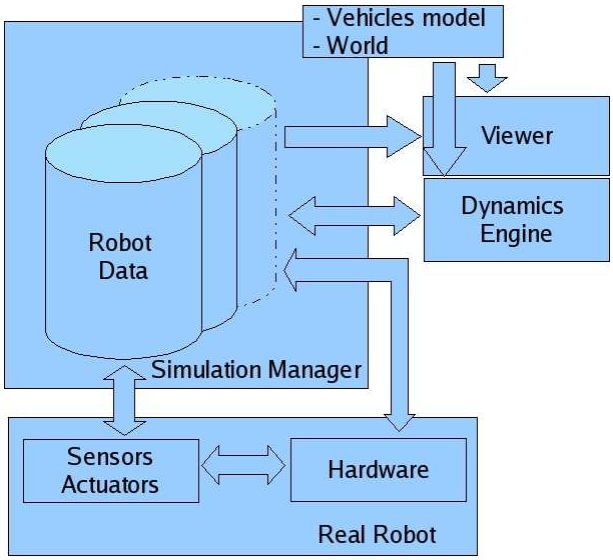


Figura A.4: Arquitectura genérica de um sistema robótico simulado

- simulação de diferentes arquitecturas para controlo e coordenação numa mesma ferramenta;
- resolução dos problemas construtivos dos robôs, (p.e. disposição de sensores);
- desenvolvimento mais rápido de *software*;
- Simulação da dinâmica de um robô;
- Simulação de múltiplos veículos;
- Comunicação entre veículos (se for possível correr o código real no robô simulado);
- *hardware in-the-loop*;
- Simulação de várias equipas numa mesma máquina.

Desde a introdução e verificação da dificuldade de implementação dos ambientes multi-robóticos começaram a aparecer *frameworks* de trabalho para a si-

mulação desde um sensor apenas ou um actuador até à equipa inteira de veículos. Algumas mais conhecidas e que apresentam um grau razoável de desenvolvimento:

- Misionlab
- Player/Stage/Gazebo
- CARMEN
- USARsim
- Webots
- TeamBotica

Na simulação de uma equipa de robôs, cada robô deve poder ter diferentes níveis de abstracção. Sabendo à partida que a dinâmica do robô é parte essencial, os sensores, actuadores, *hardware* e *software* devem poder ser vistos a diferentes patamares. É necessário que o simulador seja constituído por diferentes pequenas partes que simulam os diferentes sensores, os diferentes actuadores, a dinâmica ou mesmo que emulem o código dor robô. Essas pequenas partes têm de poder ser facilmente configuráveis já que na maioria dos casos a frequência de saída dos dados é diferente.

Alguns dos ambientes de desenvolvimento ou a combinação destes com algumas linguagens permitem simular a dinâmica de um sistema. A ferramenta CHARON [31], assim como Times [32], ou a variante do Mocha [28], o jMocha têm integrado um simulador da dinâmica do sistema. Estas aplicações podem ainda ser usados como o TeamBots (que é uma aplicação baseada em Java para a investigação relacionada com a cooperação entre múltiplos agentes) onde o simulador integrado executa o código gerado para execução, da mesma forma acontece com o Mission Lab [30].

Existem ainda linguagens orientadas simplesmente para a simulação como é o caso da linguagem SHIFT [71] que permite descrever redes dinâmicas de autómatos híbridos. Esta linguagem surgiu da necessidade de se criar uma ferramenta que suportasse sistemas híbridos dinâmicos e reconfiguráveis para uma aplicação bem definida, os veículos autónomos em auto-estradas, apesar

Algorithm 1 Exemplo de um termóstato em CHARON [70].

```
gent thermostat()
mode top = thermostatTop()

mode thermostatTop()
private analog real x;
mode on = onOff(-10000000000.0, 82.0, 100.0);
mode off = onOff(68.0, 10000000000.0, 0.0);
trans toSubMode from default to on when true do x= 73
trans fromOnToOff from on to off when x ¿ 80.0 do
trans fromOffToOn from off to on when x ¡ 70.0 do

mode onOff(real a, real b, real c)
readWrite analog real x;
inv invOnOff x ¿ a and x ¡ b
diff dOnOff d(x) == -x + c
```

de poder ser aplicado a coordenação entre veículos autónomos sejam terrestres marítimos ou aéreos. Outras como Stateflow que por si só são ferramentas de simulação de sistemas do tipo *event-driven*, permitem ainda obter soluções para aplicações em sistemas embebidos usando Stateflow Coder. Esta ferramenta está integrada no *software* MATLAB tal como o Simulink que permite simular sistemas do tipo contínuo e discreto. Com o auxílio das ferramentas associadas ao MATLAB foi desenvolvido ainda uma outra ferramenta, o CheckMate [51]. Ferramentas como o ORCCAD [57, 29] geram código para simuladores específicos, ou seja o código gerado para o simulador é diferente do usado nas aplicações.

O projecto Player/Stage/Gazebo [11] fornece dois simuladores multi-robóticos: o Stage e o Gazebo. Como os dois são compatíveis com o Player, os programas clientes podem utilizar os dois com poucas ou nenhumas alterações. A diferença principal entre estes dois simuladores é a de que o Stage foi implementado para simular uma grande equipa de robôs com uma baixa fidelidade e o Gazebo foi

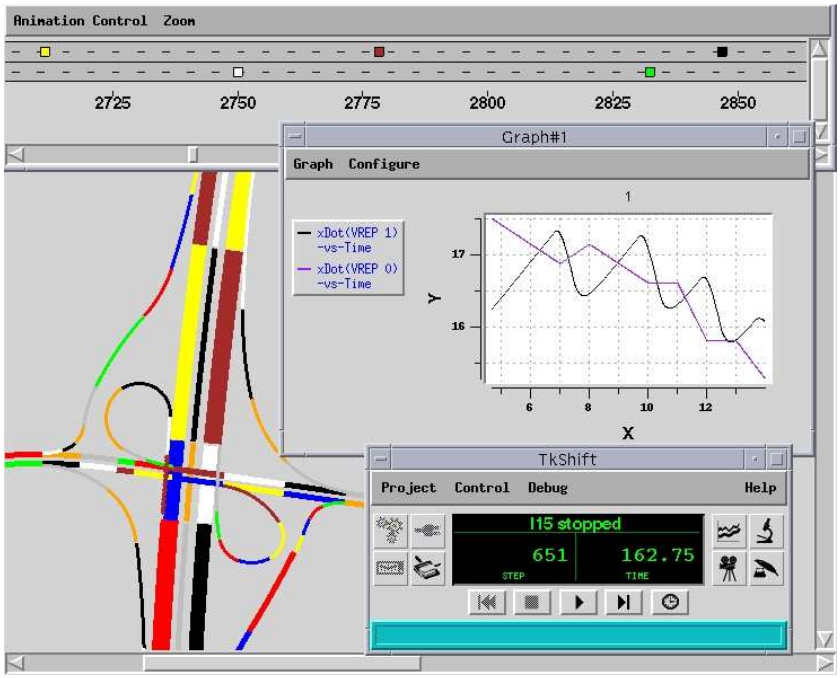


Figura A.5: Simulação SHIFT/Smart-AHS para aplicação em veículos autónomos em auto-estradas [10].

implementado para simular uma pequena equipa de robôs com grande fidelidade. Como tal os utilizadores podem alterar entre um e outro consoante as suas necessidades.

O Player fornece um mecanismo client-servidor pelos quais torna possível interagir entre com robots e sensores reais.

Gazebo é um simulador de robótica para ambientes externos. Tal como o Stage, é capaz de simular uma população de robôs, sensores e objectos, mas fá-lo de uma forma tridimensional. O Gazebo consegue gerar dados realistas de sensores e simulações físicas entre os agentes e os objectos. O Gazebo ao funcionar com o Player, permite fornecer dados de sensores simulados em vez de dados de sensores reais. Idealmente os clientes não deverão distinguir se os robôs são reais ou simulados.

O Gazebo também pode ser controlado por uma interface de baixo nível (libgazebo). Esta biblioteca permite que terceiros consigam integrar o Gazebo nos robôs que não sejam baseados no Player.

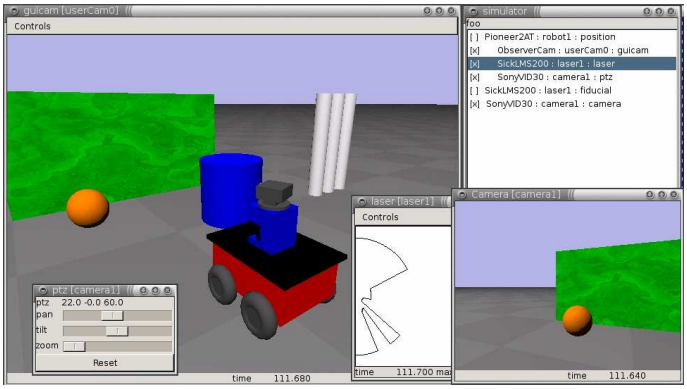


Figura A.6: Player/Stage/Gazebo [11]

Outro exemplo idêntico ao Player/Stage, é o CARMEN [72], sendo um *software* que permite fazer desenvolvimento para sistemas robóticos. Tendo uma estrutura modular tem já implementados algoritmos de localização, planeamento de trajectórias, mapeamento e desvio de obstáculos. Tem já implementado também um simulador 2D que permite simular todo o robô ou apenas sensores. À semelhança do Player/Stage o Carmen também suporta alguns tipos de *hardware* (Receptor GPS e Sick Laser).

A.3 Ferramentas de verificação

A verificação de propriedades dos sistemas híbridos, têm sido uma das principais motivações para o desenvolvimento de ferramentas que permitem fazer a verificação formal. Estas ferramentas servem essencialmente a verificação de determinadas características como a *safety* e a *liveness* [73]. Estas propriedades normalmente representam especificações fornecidas.

UPPALL consiste numa ferramenta para verificação automática das propriedades da segurança dos sistemas de tempo real [26]. É ainda realizada uma verificação sintáctica depois da descrição textual do sistema. Esta verificação sintáctica permite que se detectem erros de sintaxe de variáveis mal declaradas ou usadas de uma forma errada. Obe Checkmate [51, 74] usa o ambiente gráfico do Stateflow e Simulink para a modelização dos sistemas juntamente com dois

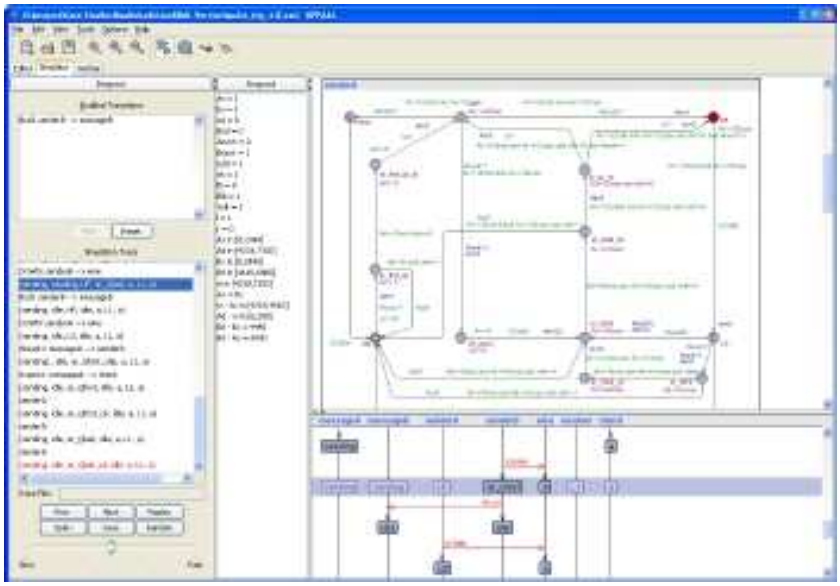


Figura A.7: Software UPPAAL [12].

comandos, um de exploração e um de verificação. O comando explore, começa por verificar se as especificações do sistema serão verdadeiras para qualquer estado inicial, e o comando *verify* verifica todas as trajectórias possíveis desde os estados iniciais até aos estados finais. Times [32, 75] faz uma análise do escalonamento, para verificar se é possível escalonar todas tarefas cumprindo as *deadlines* depois de de definirem as propriedades do sistema. Outra ferramenta para verificação dos requisitos temporais do sistema a HyTech (Hybrid TecHnology Tool) [76, 77, 78, 79] que é uma ferramenta de análise simbólica de sistemas embebidos para verificação automática dos requisitos temporais de um sistema híbrido linear. Os sistemas híbridos lineares são uma subclasse dos autómatos híbridos. O HyTech consegue, na sua análise, determinar os parâmetros que satisfazem determinadas condições temporais para um determinado sistema sendo ainda capaz de lidar com sistemas de qualquer dimensão. SPeeDI [80] tem como principal ferramenta a da verificação onde se podem definir as condições iniciais e finais de modo a que a ferramenta verifique se tal é admissível. Mocha [28] e as suas duas variantes cMocha e jMocha, cuja principal diferença é a linguagem de programação usada C e Java respectivamente, usa para a verificação dos requisitos do sistema a ATL (*Alternating Temporal Logic*) [28]. Esta ferramenta

verifica ainda a implementação seguindo a ligação existente entre os módulos de especificação e implementação. O processo de simulação em CHARON [31] inicia sempre o procedimento pela verificação formal. A ferramenta KRONOS [81, 82] foi desenvolvida com o intuito de verificar sistemas de tempo-real complexos, esta ferramenta permite a implementação ou a geração de código automático a partir de modelos de especificação. No entanto verifica se o modelo especificado por autómatos temporizados satisfaz as propriedades temporais especificadas pela fórmula da lógica temporal (TCTL). KRONOS é usado ainda juntamente com TAXYS [83] usada para validar *software* de telecomunicações de tempo-real. Existe também a ligação entre DIADEM e KRONOS [84] onde DIADEM [85] é uma plataforma para implementação de sistemas tempo real com atendimento de acontecimentos.

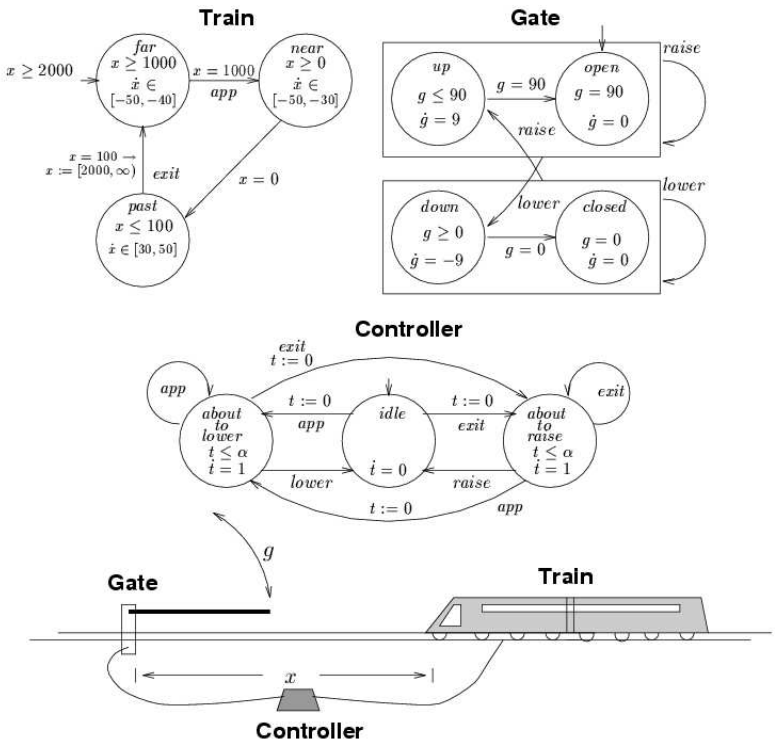


Figura A.8: Especificação de um controlador de passagens de nível por forma a garantir que sempre o comboio esteja a 10 m da passagem a barreira esteja garantidamente fechado [13].

A.4 Ferramentas para a implementação

Nem todos os ambientes de desenvolvimento possuem ferramentas que permitam a implementação ou a geração de código automático a partir de modelos de especificação. No entanto mais recentemente com o desenvolvimento sucessivo de alguns ambientes de desenvolvimento comummente usados para a simulação e verificação surgiram algumas ferramentas que permitem a implementação dos sistemas em ambientes computacionais particulares, ou ainda a gestão da implementação. TeamBots é uma ferramenta que permite criar código para aplicações em equipas de robôs com o objectivo de estudar a cooperação entre eles. O pacote de *software* foi desenvolvido na *Carnegie Mellon University* sendo usado no futebol robótico. O Mission Lab [30], é baseado na CDL (*Configuration Description Language*) seguindo a *Social Agent Theory* que diz que quer humanos quer animais possuem controladores baseados em comportamentos tornando-nos numa grande sociedade de agentes. A CDL serve para definir a inicialização e coordenação das primitivas do sistema mas não a sua implementação. A ferramenta Times (*Tool and Implementation of Embedded Systems*) [32] gera código para uma plataforma específica: LegoOS. Existem ainda ferramentas que juntam diferentes paradigmas para a implementação como o projecto FRESCO [27] que usa o Masaccio [55] como o modelo formal para a definição do sistema e o Giotto [86, 87, 88] fornece um modelo para implementação do controlo de sistemas embebidos gerando código para aplicações *hard real-time*. O Giotto baseia-se numa *time-triggered* API. O ORCCAD gera código C++ [57] que é independente da plataforma sendo mais usado em Solaris e VxWorks.

A ferramenta DIADEM [85] fornece uma estrutura de *software* orientada aos objectos para a implementação de sistemas de controlo. Possui como modelo de desenvolvimento o de uma rede dinâmica de autómatos híbridos. Permite a existência de eventos síncronos ou assíncronos bem como a reconfiguração dinâmica da propagação de eventos entre os componentes permitindo desta forma a reconfigurabilidade da rede. O código gerando é ligado a um conjunto de classes base e a um escalonador fornecido, sendo construída uma aplicação monolítica. Esta aplicação pode correr num dos sistemas operativos definidos (Linux, SunOs e QNX).

A.5 Análise às ferramentas existentes

A parte conceptual de uma arquitectura se um sistema robótico é extremamente importante, daí se dar tanto ênfase à modelização e verificação lógica das algoritmos de controlo. Este tipo de análise permite saber se estamos perante um sistema, por exemplo, tolerante a falhas, se o seu objectivo pode ou não ser atingido ou mesmo se é possível respeitar as restrições temporais do mesmo. A implementação de sistemas que respeite completamente os modelos definidos teoricamente ainda não são possíveis até porque existe muitos problemas que não entram no processo de modelização e verificação, nestes casos podemos incluir os erros de *hardware*. Muitas das vezes a própria implementação em si não corresponde ao verdadeiro modelo conceptual por falhas na construção. Daqui se percebe a razão de ainda muitos dos paradigmas serem apenas testados em simulação e não num ambiente real.

Existem ainda muitas limitações nas quais estão subjacentes algumas arquitecturas e que ainda não foram resolvidas, muitas prendem-se com a simples comunicação máquina-máquina outras porque fazem uma abstracção das máquinas em si e outras porque ainda não foram validadas já que o seu objecto de estudo não permite simulação (por exemplo exploração de outros planetas). O facto de existirem algumas arquitecturas que permitem a implementação, e apesar de não serem validadas conceptualmente, permitem já o teste e resolução de problemas específicos, no entanto, estes problemas são diferentes de arquitectura em arquitectura e nenhuma delas tenta a formalização genérica de um sistema deste género. Daqui se pode observar facilmente que ainda não existe nenhuma arquitectura que permita iniciativa mista global.

Apêndice B

Middleware e os sistemas robóticos

Conteúdo

B.1 Middleware e os sistemas robóticos

Com a introdução de maiores desafios nos sistemas que usam equipas de robôs, foram aparecendo vários *middlewares* que permitem uma maior integração de todos os componentes que equipam os sistemas robóticos discutidos até aqui. No entanto, ainda não existe nenhum que seja um dado adquirido e que funcione em qualquer sistema multi-robótico. A interacção entre o homem e o robô é deixada para segundo plano por parte dos *middlewares* aqui descritos.

O recurso a middlewares permite [44]:

- Simplificação desenvolvimento;
- Estrutura de comunicações melhorada;
- Eficiência em todos os recursos envolvidos (p.e. reutilização do código);

- Permitir uma maior abstracção do *hardware*;
- Existência em muito casos de ambientes de simulação.

Os *middlewares* apresentados de seguida não pretendem ser um apanhado de todos existentes mas algumas das abordagens existentes mais representativas e onde onde o desenvolvimento ainda se verifica bastante activo.

B.1.1 Orocos

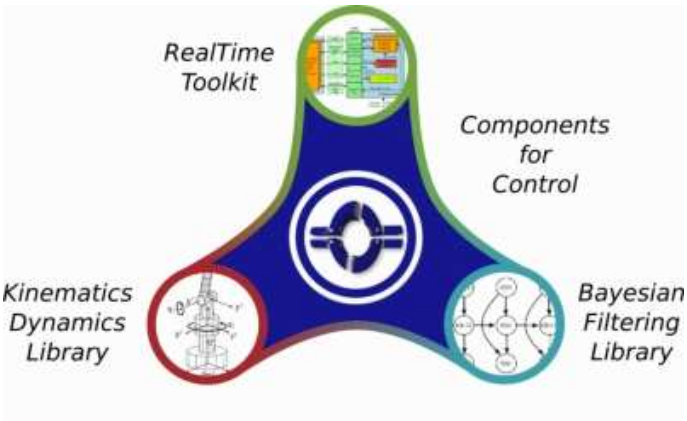


Figura B.1: Orocos [14]

O projecto *Open Robot Control Software*, Orocos [14], propõe um conjunto de componentes que permite uma maior rapidez do desenvolvimento de robôs. O Orocos é constituído pelos seguintes módulos:

- *Real-Time Toolkit* (RTT);
- *Orocos Components Library* (OCL);
- *Orocos Kinematics and Dynamics Library*;
- *Orocos Bayesian Filtering Library.*

Os módulos podem ser usados individualmente ou em conjunto nas aplicações de quem desenvolve sistemas robóticos, por si só os módulos não funcionam sozinhos, mas conseguem integrar-se e interligar-se nas aplicações onde são usados.

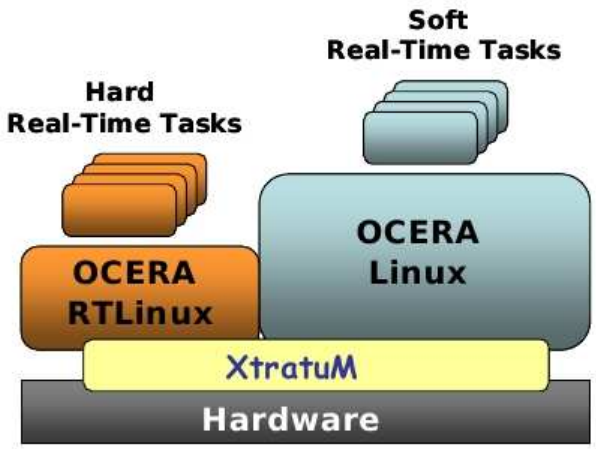


Figura B.2: Ocera [15].

B.1.2 Ocera

O Ocera, *Open Components for Embedded Real-time Applications* [15], é um projecto que se foca nos componentes de tempo real para aplicações embebidas utilizando um kernel Linux. Os componentes desenvolvidos podem agrupar-se em quatro grandes áreas:

- *Resource management*;
- *Scheduling*;
- *Fault-tolerance*;
- *Communication*.

Todas os componentes desenvolvidos têm características de tempo real, sejam *hard* ou *soft* sendo que alguns não podem ser usados separadamente, têm que ser obrigatoriamente usadas dentro da *framework* Ocera.

B.1.3 Comunicações

As comunicações mais comummente usadas nos sistemas robóticos, ou seja o fluxo de dados que é normal existir nestes sistemas, são sumariadas na Figura B.3. O tipo de dados que circulam nestes ambientes necessitam, na maior parte

dos casos, de comunicações de tempo real, redes bastante complicadas, controlo da largura de faixa, dados descentralizados, garantia de entrega dos dados, diferentes tipos de dados e alocação dinâmica de dispositivos. Em termos de implementação prática, apesar de existirem outros protocolos de comunicações, *bluetooth*, *zigbee*, ou outros protocolos com fios que tenham mecanismos de serem usados sem fio, são as redes *wireless Ethernet* que têm maior aplicação.

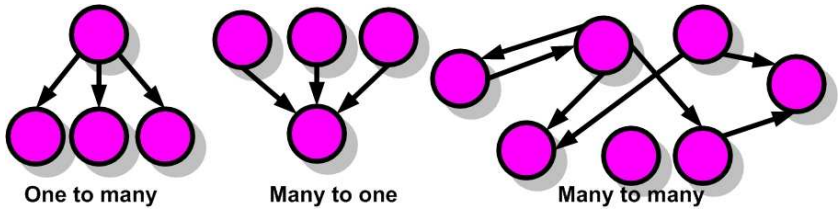


Figura B.3: Tipos de comunicações usadas em ambientes multi-robóticos

O tipo de dados que circulam nestes tipos de ambientes podem ser listados da seguinte forma:

- **Comandos** necessitam de ser entregues apenas uma vez mas de uma forma garantida e a ordem de chegada deve ser coerente com a ordem de envio;
- **Sinais** são valores que variam no tempo. Podem ser críticos no tempo, e, apesar de ser importante o reenvio em caso de falha, é ainda mais importante o envio de novos dados;
- **Eventos** são gerados sempre que é necessário iniciar manobras que exijam coordenação, é necessário que exista a garantia da entrega e que se enquadrem num espaço temporal crítico;
- **Estados** fornecem a indicação da realidade actual dos robôs. Não são críticos em termos temporais e podem ser repetidos.
- **Pedidos** são dados essencialmente de quem está a desenvolver;
- ***Logs*** (registos de dados organizados num formato conhecido) são dados que podem implicar uma grande largura de banda quando estão a ser enviados, nos *logs* o tempo não é crítico, mas a garantia de entrega de alguns e a sequência correcta de chegada pode ser importante;

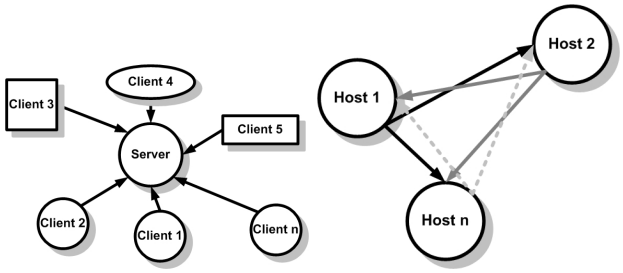


Figura B.4: Comunicação típicas cliente-servidor (esquerda) e máquina a máquina (direita)

Existe ainda o *broadcast* e *multicast* para envio de dados. Este tipo de estratégia faz com que toda a informação que seja precisa circule na rede sendo enviada a todas as máquinas, mesmo quando não é necessária ou quando apenas uma das máquinas a vai usar. Este facto faz com que o tráfego de rede seja muito elevado. Considerando estes cenários é possível verificar que existem essencialmente duas camadas de comunicação, as TCP e UDP. O uso de cada uma delas depende essencialmente da aplicação. É sabido que as redes TCP permitem a garantia de entrega de dados, mas requer um desenvolvimento mais complexo, a UDP, apesar de ser de implementação simples não garante a entrega dos dados. De notar ainda que as comunicações *ethernet* não garantem entrega de dados em tempo real nem garante a sequência de envio. A Figura B.4 mostra o tipo de arquitectura mais comum de comunicações. No caso de existir problemas no servidor, os dados permanecem inacessíveis.

Todas estas implicações levam a que sejam propostas novos paradigmas de comunicação, como ocaso RTPS [89, 90]. A arquitectura *Real Time Publish Subscribe* é orientada à informação e é implementada através de um *middleware*. A máquina que fornece os dados é responsável pela sua entrega, a Figura B.5 esquematiza o funcionamento de um destes *middlewares* [91, 92].

É possível dividir a informação, interna ou externa, em duas categorias, mensagens do tipo *event triggered* ou *time triggered*. As mensagens *time triggered* são que podem ser controladas, sendo apenas definido o tempo entre cada men-

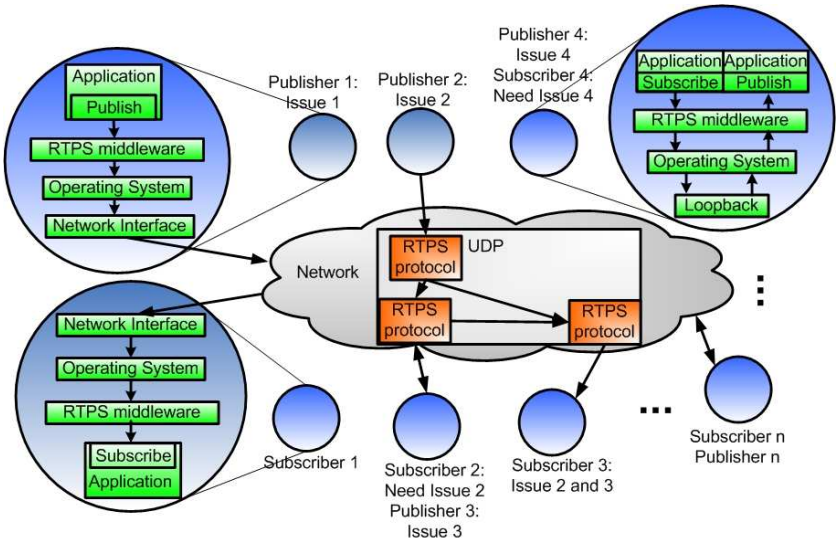


Figura B.5: Comunicações *publish-subscribe*

sagem fazendo com que o tráfego local seja determinístico. As *event triggered* são as mais problemáticas já que são enviadas para a rede sempre que exista um novo evento.

Apêndice C

Características dos veículos dos casos de estudo

Conteúdo

C.1 Características da equipa de futebol robótico ISePorto

A equipa de futebol robótico é constituída por 6 jogadores, um guarda-redes e cinco jogadores de campo. Em termos de *hardware* e *software*, os jogadores (Figura C.1) têm as seguintes características:

- Dois sensores de visão, um no topo e outro no chutador;
- Cinco graus de liberdade, tracção diferencial, câmara do topo, chutador e corpo rotativo;
- Rede CAN entre os diferente nós de controlo dos motores;

Figura C.1: Futebolista robótico do ISePorto

- *Board* computacional (industrial);
- *Pack* de 4 baterias 3600mAh/4200mAh;
- Chutador mecânico;
- Sistema operativo Linux sem X (Ambiente gráfico);
- Aplicação *multi threaded* com controlo hierárquico;
- Jogadas pré-definidas: controlo de bola, defesa, ataque, chuto à baliza;
- Possibilidade de construção de jogadas complexas baseadas nas jogadas simples;
- Biblioteca de comunicações *unicast/multicast*;
- Biblioteca de visão;
- Biblioteca dos controladores de potência.

C.2 Características do veículo ROAZ

O projecto de desenvolvimento ROAZ [42] tem como objecto de estudo a robótica marítima no desenvolvimento de veículos autónomos de superfície sendo dirigido o trabalho para dois grandes tópicos, a recolha de dados e monitorização e inspecção marítima. O programa conta já com dois veículos uma pequena embarcação catamarã ROAZ I e uma outra relativamente maior ROAZ II, ambos com uma grande variedade de sensores a bordo tais como câmaras termográficas, sonar e CTD.

As principais áreas de trabalho são a monitorização ambiental, batimetria, recolha de dados oceanográficos, busca e salvamento de pessoas e missões de segurança e reconhecimento. Todas estas tarefas consomem hoje em dia recursos humanos muito elevados. Veículos deste género com algum grau de autonomia conseguem realizar as tarefas descritas apenas com um elemento humano.

O ROAZ C.2 é constituído por:

Figura C.2: Aspecto do ROAZ II

- Embarcação catamarã de 4m;
- Sensores: câmaras de filmar (*streaming de video*), câmara termográfica, GPS, radar, *sidescan sonar* e altímetro;
- *Board* computacional;
- Dois motores de propulsão eléctricos com controlador;
- Sistema operativo Linux sem X (ambiente gráfico);
- Aplicação *multi threaded* com controlo hierárquico;
- Algumas manobras já implementadas;
- Possibilidade de execução de manobras baseadas nas manobras mais simples;

- Completamente teleoperável a partir de uma estação base.

C.3 Características da plataforma *Wireless Positioning System*

A plataforma de recolha de valores de potências de sinais *wireless* é constituída essencialmente por:

- Rodas com *encoders*;
- ARM Cortex;
- Sistema computacional Linux;
- Rede de comunicação CAN;
- Interface de uso e configuração do sistema.

Apêndice D

Características dos computadores de bolso

Principais características dos computadores de bolso utilizados como consola de operações portátil:

- Peso: 230 g
- Dimensões: 141 x 79 x 19 mm
- Ecrã táctil 800x480 com 65.536 cores
- Memória DDR RAM de 64 MB e Flash 128 MB
- Processador ARM9
- Memória: Flash 128 MB
- Sistema Operativo Internet Tablet 2005
- *Wireless LAN*: 802.11b/g