UNIVERSIDADE DE COIMBRA

FACULDADE DE CIÊNCIAS E TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES

ISROBOT:

DESENVOLVIMENTO E INTEGRAÇÃO DE SISTEMAS E MÓDULOS PARA CONDUÇÃO AUTÓNOMA

Luís Filipe Rodrigues Alves

Coimbra, 2008

Universidade de Coimbra

Faculdade de Ciências e Tecnologia

Departamento de Engenharia Electrotécnica e de Computadores

Mestrado em Engenharia Electrotécnica e de Computadores

ISRobot:

Desenvolvimento e Integração de Sistemas e Módulos para Condução Autónoma

Luís Filipe Rodrigues Alves

	Júri:
Presidente:	**Urbano Nunes**
Orientador:	**Urbano Nunes**
Vogais:	**João Barreto**
	Paulo Peixoto

Coimbra, Setembro de 2008

ISRobot:

Desenvolvimento e Integração de Sistemas e Módulos para Condução Autónoma

Dissertação Submetida ao Departamento de Engenharia Electrotécnica da Faculdade de Ciências e Tecnologia da Universidade de Coimbra satisfazendo parcialmente os requisitos para a obtenção do grau de Mestre em Engenharia Electrotécnica e de Computadores, área de especialização em Automação e Informática Industrial

Submetida por:

Luís Filipe Rodrigues Alves

Sob orientação de:

Urbano José Carreira Nunes

Ana Cristina Lopes

Coimbra, Setembro de 2008

“A vida é-nos dada para realizarmos grandes coisas.”

Agradecimentos

Durante o decorrer deste projecto várias pessoas tiveram um contributo para a evolução do mesmo, sem o qual este não teria sido possível. Sendo assim, expresso aqui a minha homenagem de gratidão

Ao Prof. Doutor Urbano Nunes, meu orientador, agradeço a oportunidade que me deu para desenvolver este projecto, assim como a orientação e apoio prestados.

Ao ISR (Instituto de Sistemas e Robótica) quero agradecer os meios técnicos disponibilizados os quais foram indispensáveis para a realização de todo este trabalho. À FCT (Fundação para a Ciência e Tecnologia) agradeço também pelo facto de através do projecto *MTDTS04: Detecção e Seguimento de Múltiplos Alvos em Ambientes Exteriores Semi-estruturados usando Laser e Visão* (POSC/EEA/SRI/58279/2004) ter possibilitado a realização deste subprojecto assim como a concessão duma bolsa de iniciação à investigação, sem a qual não teria sido possível a realização desta tese.

Dentro do ISR quero agradecer aos outros investigadores que me acompanharam e me incentivaram no decurso deste projecto nomeadamente: Ana Cristina, Gabriel Pires, Pedro Sousa e Rodrigo Maia. Também agradeço aos meus colegas de laboratório pelos momentos bem passados e camaradagem e em especial ao Filipe Apóstolo e Luís Vaz que dedicaram parte do seu tempo a este projecto aquando do FNR (Festival Nacional de Robótica) este ano em Aveiro no qual participamos na prova de Condução Autónoma.

Agradeço também ao Pedro Dias, Samuel Oliveira e seu orientador Prof. Luís Almeida do Departamento de Informática do IPT (Instituto Politécnico de Tomar) pela disponibilidade e ajuda prestada na área da visão artificial com os quais tive o privilegio de trocar ideias e soluções antes e durante o evento do FNR.

Por fim fica um muito obrigado à minha família por ter incutido em mim uma vontade de vencer e me ter depositado uma grande confiança fazendo sacrifícios para que tal sonho se realiza-se.

Resumo

Este projecto consiste no desenvolvimento de um conjunto de módulos de hardware/software, visando a construção de uma arquitectura distribuída de navegação de um robô móvel para integração de uma plataforma de software. Esta plataforma de software foi desenvolvida a pensar na autonomia do robô utilizando um PC embutido no mesmo. Os módulos de hardware destinam-se ao controlo e interface com o sistema motriz, o sistema sensorial, e o sistema de alimentação do robô.

Os módulos de hardware/software têm que estar munidos de capacidade de comunicação com a plataforma de software. É nesta plataforma de software que se vai encontrar o algoritmo responsável pela tomada de decisões em função da informação recolhida pelos vários elementos sensoriais, efectuando cálculos e enviando a informação relativa às velocidades pretendidas para os módulos de hardware responsáveis pelo controlo motriz do robô. Os módulos de hardware/software têm que ter a capacidade de transformar a informação enviada pela camada superior num efectivo e correcto sinal de comando para controlo de tracção dos motores.

O objectivo final é conseguir dotar um robô das capacidades necessárias para percorrer um percurso em tudo semelhante a uma estrada comum – Condução Autónoma – sendo este capaz não só fazer o percurso no menor tempo possível, como o de se desviar de obstáculos presentes na via, ou até mesmo detectar a presença de uma passadeira para peões parando ou seguindo em função do estado do semáforo.

Palavras-Chave:

Arquitectura Distribuída; Comunicação CAN; Módulos de Hardware/Software; Condução Autónoma.

Abstract

This project consists in the development of hardware/software modules aiming at the construction of distributed navigation architecture for a mobile robot to be integrated in a software platform already developed. This software platform was conceived focusing the robot mobility and is based on an embedded PC. The hardware modules are responsible for the control and interface to the motion system, the sensory system, and the power supply system.

The hardware/software modules have the capacity to communicate with the software platform that is responsible for taking the decisions in function of the sensory information and sending of the speed commands for the motor control modules of the robot. The hardware/software module should convert the command signal into a motion control action.

The final goal is to develop a robot that is able to track a trajectory similar to a common road – Autonomous Driving – in the shortest possible time, avoiding obstacles and respecting the commands issued by the traffic light.

Key words:

Distributed architecture; CAN communication; Modules of Hardware/Software; Autonomous Driving.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

AGV – Veículos Guiados Automaticamente do inglês *Autonomous Guided Vehicle*

ACR – Arquitectura de Controlo do Robô

CAN – *Controller Area Network*

DAC – Conversor Digital-Analógico do inglês *Digital-to-Analog Converter*

DARPA – *Defense Advanced Research Projects Agency*

DEE – Departamento de Engenharia Electrotécnica

DES – Sistema Embebido Distribuído do inglês *Distributed Embedded Systems*

FCT – Fundação para a Ciência e Tecnologia

FNR – Festival Nacional de Robótica

IPT – Instituto Politécnico de Tomar

IC – Circuito Integrado do inglês *Integrated Circuit*

ISR-UC – Instituto de Sistemas e Robótica - Pólo de Coimbra

ISRobot – Robô Diferencial para Condução Autónoma

IV – Infra-Vermelho

PC – *Personal Computer*

PCB – *Printed Circuit Board*

PI – Proporcional - Integral do inglês *Proportional-Integral*

PIC – *Programmable Interface Controller*

SPI – *Serial Peripheral Interface*

UBM – Universidade de Bundeswehr Munich

UC – Universidade de Coimbra

Glossário

Barramento – Designa o meio físico através do qual comunicam dois sistemas informáticos; conjunto de condutores que permite a troca de sinais eléctricos entre dois ou mais dispositivos electrónicos.

CAN (*Controller Area Network*) – Protocolo de Comunicação, desenvolvido por Robert Bosch, que permite a troca de informação entre vários dispositivos. O protocolo CAN prevê identificadores de mensagens que facilitam o controlo do fluxo de dados. Como características principais, podemos citar um controlo de alto nível na detecção/correcção de erros, grande flexibilidade na topologia e arranjo da rede e baixa latência na comunicação.

CPU (Unidade central de processamento do inglês *Central Processing Unit*) - Parte de um computador que interpreta e executa as instruções contidas em software. Na maioria das CPU's, essa tarefa é dividida entre uma unidade de controlo que dirige o fluxo do programa e uma ou mais unidades de execução que executam operações em dados.

DAC (*Digital to Analog Coverter*) – Conversor de sinais digitais em sinais analógicos.

IC (Integrated Circuit) – Termo Inglês que significa Circuito Integrado. É um dispositivo micro electrónico que integra transístores e outros componentes interligados capazes de desempenhar várias funções. Apresenta dimensões extremamente reduzidas, sendo os componentes formados em pastilhas de material semicondutor.

IV (infra-vermelhos) – Conjunto de emissor e receptor de raios infra vermelhos com o objectivo de detectar a presença de meios absorventes ou reflectores desses raios.

PIC - Os PIC's são uma família de microcontroladores fabricados pela Microchip® Technology, que processam dados de 8, 16 ou 32 bits, com extensa variedade de modelos e periféricos internos. Estes dispositivos electrónicos tem capacidades semelhantes às de um microprocessador, contendo dispositivos periféricos (ADC's, PWM, Contadores, etc …) já integrados, controlados por um microprocessador interno.

Ultra-sons - Dispositivos para detecção de obstáculos físicos, através da emissão de ultra-sons. Consiste num sistema capaz de emitir ondas sonoras e captar os seus ecos, permitindo assim verificar a distância aos obstáculos através da medição do tempo entre a emissão do som e a recepção do seu eco.

SPI (*Serial Peripheral Interface*) – Protocolo de comunicação série, para transferência de dados com elevado tráfego e para distâncias curtas.

Capítulo 1

1. Introdução

Neste capítulo é apresentada a introdução ao trabalho realizado no âmbito do Mestrado em Engenharia Electrotécnica e de Computadores, área de especialização em Automação, fazendo o seu enquadramento e identificando os objectivos. Inicialmente expõe-se a motivação que originou este trabalho, seguindo-se a apresentação dos objectivos e grau de concretização. Finaliza-se com a apresentação da estrutura da tese.

1.1. Motivação

A área dos robôs/veículos móveis inteligentes semi-autónomos/autónomos tem vindo a ser alvo de interesse por um alargado leque de investigadores por todo o mundo. Esse interesse tem-se manifestado em aplicações diversas tais como: indústria automóvel, transportes públicos, industriais, militares e robôs centrados no ser humano onde por exemplo se destacam os robôs que assistem pessoas com necessidades especiais.

No que toca à indústria automóvel os sistemas activos de segurança tornaram-se familiares, uma vez que estes são amplamente comercializados. Os carros actuais integram sistemas mencionados na Tabela 1.1. A evolução destes sistemas activos de segurança de auxílio parcial à condução está listada na Tabela 1.1 (Mendes, 2004).

Tabela 1.1: Evolução dos sistemas activos de segurança

Ano	Sistema de Segurança
1978	Sistema anti-derrapagem (ABS)
1982	Controlo de velocidade de cruzeiro (CC)
1989	Controlo de tracção (TCS)
1990	Sistema de controlo anti-patinagem (ASR)
1995	Controlo de estabilidade (ESP)
1997	Controlo de distância de parqueamento (PDC)
1998	Controlo activo da velocidade de cruzeiro (ACC)

Estes sistemas continuam a evoluir em termos de investigação, com o objectivo de chegar a soluções ainda mais inovadoras (inteligentes e seguras) tais como: manutenção na faixa de rodagem e condução autónoma. Prova disso é o projecto europeu *Prometheus* e *Cybercars* e o projecto americano *Darpa* que são referidos no capítulo 2.1 abaixo. Sendo estes projectos direccionados para ambientes exteriores e que se integram na área da indústria automóvel, transportes públicos e "militares" respectivamente.

No entanto, também existem projectos direccionados para ambientes interiores e que incorporam conceitos semelhantes aos mencionados anteriormente, no que diz respeito à condução autónoma. Tais projectos passam pelo transporte de materiais dentro unidades fabris, chamados vulgarmente por *Autonomous Guided Vehicle* (AGV). Existem ainda projectos que têm por objectivo auxiliar pessoas com necessidades ao nível da mobilidade, sendo um deles a *RobChair*[1]. Projecto este com vários contributos e objectivos, sendo alguns mais vocacionados nomeadamente para auxílio à pessoa com mobilidade limitada (Pires, et al., IAV'98), (Pires, et al., AMC'98), (Solea, et al., ICINCO'08). No entanto, esta plataforma tem sido também usada para testes de outros algoritmos passíveis de serem incorporados em veículos. Esses algoritmos são: seguimento de trajectórias, planeamento de trajectórias e detecção e classificação de obstáculos e espaço envolvente (Mendes, et al., 2004) (Premebida, et al., Robótica'05) (Solea, et al., Controlo'06) (Lopes, et al., ETFA'07) (Sousa, et al., ISIE'07) (Sousa, et al., ETFA'07) (Solea, et al., 2007) .

[1] RobChair - acrónimo dos termos em inglês *Robot* e *Wheelchair*

Os algoritmos desenvolvidos têm que ter em conta o meio envolvente, seja ele ambiente exterior ou interior semi-estruturado ou não estruturado. É necessário portanto a percepção do meio envolvente através dos mais variados sensores.
No entanto, todos estes sistemas independentemente da dimensão e da complexidade, têm por base uma infra-estrutura que possibilita a locomoção (estrutura mecânica e eléctrica). É sobre esta estrutura que se aplicará a plataforma de software capaz de dotar o sistema de determinados graus de inteligência e autonomia.

1.2. Âmbito da Tese

Os algoritmos orientados aos sistemas robóticos autónomos após serem idealizados e devidamente simulados em diversos ambientes tais como o MatLab ou Player/Stage necessitam de ser devidamente testados em ambientes real. Para tal é necessário ter uma plataforma que interaja com o mundo e possibilite esses testes. Uma plataforma móvel passível de ser controlada por meio de software, ou seja, que tenha uma camada de abstracção que faça o interface entre o computador onde temos o software de alto nível e a parte de actuação da plataforma. Para além de conseguir analisar o mundo que o rodeia através dos mais variados sensores, o robô deve conseguir interagir com o mundo de forma eficiente e em tempo útil (sistemas de tempo real).

A plataforma RobChair referida anteriormente é um robô de tracção diferencial, com a configuração de uma cadeira de rodas, a qual já passou por diferentes fases de concepção inclusive ao nível do hardware. Tem também servido como plataforma de testes a diversos algoritmos como foi referido na secção anterior. No entanto esta plataforma tem uma dimensão e peso considerável, não permitindo o seu transporte de forma trivial e exigindo um espaço considerável no teste dos algoritmos.

Para facilitar as experiências e testes de diversas metodologias de controlo e seguimento de trajectória, decidiu-se construir outra plataforma com sistema de tracção semelhante. Sendo esta de tamanho e peso mais reduzido possibilita ainda o uso em competições de robótica móvel como por exemplo no Festival Nacional de Robótica, na prova de Condução Autónoma (Robótica, 2008).

O projecto e construção de um novo robô (plataforma móvel) implicam a análise da interligação entre os diversos sistemas que o compõem. Exige a escolha correcta dos actuadores, dos sensores, dos sistemas de processamento e do sistema de alimentação

para este ser autónomo. A forma como todos estes sistemas comunicam entre si, para dar acções de comando e para recolha de informação, é um dos ponto-chave de todo o sucesso do sistema.

Aos canais de comunicação que interligam os vários módulos de processamento com os módulos dos sistemas sensoriais e de actuação, dá-se o nome de Arquitectura de Controlo do Robô (ACR). A análise da ACR pode ser feita segundo dois planos: o de software e o de hardware, sendo que um é dependente do outro. Com o aparecimento dos sistemas de tempo real, as arquitecturas de software são construídas de forma modular (sistema de processamento central - tipicamente o PC). No entanto ao nível do hardware, a opção dominante durante muito tempo era que tanto os sistemas sensoriais como de actuação convergiam todos para a mesma unidade de processamento. Esta opção trazia os inconvenientes de uma cablagem volumosa e complexa, não permitindo a extensibilidade do sistema uma vez que não é modular.

Inspirado em grande parte na indústria automóvel - onde a cablagem devido à sua dimensão, é uma parte dispendiosa e crítica - onde são usados sistemas DES (*Distributed Embedded Systems*), tem-se vindo a desenvolver cada vez mais robôs com este tipo de arquitectura, tanto ao nível de hardware como software.

1.3. Objectivos do Trabalho

No decurso do trabalho desenvolvido no âmbito do projecto RobChair (Maia, 2004), em que essa plataforma foi dotada de um sistema DES de forma a providenciá-la de novas capacidades e melhor desempenho, verificou-se a necessidade de redefinir toda a parte de processamento distribuído que está interligada por um barramento de comunicação com o sistema central de processamento. Essa remodelação foi efectuada de forma a existir determinismo e sincronização na troca de informação, definindo-se também o escalonamento das tarefas de cada módulo da arquitectura distribuída. Estas alterações foram projectadas (Sousa, et al., ETFA'07) para ir de encontro ao trabalho posteriormente realizado, nomeadamente a incorporação de um sistema modular de tempo real na unidade central de processamento (Vaz, 2008).

Com vista à prova de Condução Autónoma, realizada todos os anos no contexto do Festival Nacional de Robótica, e visto que a plataforma RobChair não obedece às

limitações impostas pelo regulamento da prova (Robótica, 2008), decidiu-se desenvolver uma outra plataforma móvel.

1.4. Trabalho Realizado

Este trabalho apresenta de forma detalhada a concepção e implementação de um sistema DES, baseado num barramento de comunicação CAN (*Controller Area Network*) e unidades de processamento distribuídas baseadas em micro controladores PICs (*Programmable Interface Controller*), para integração num sistema de tempo real. Tal trabalho foi publicado em conferência internacional (Sousa, et al., ETFA'07).

A sua validação foi feita inicialmente na plataforma robótica Robchair já existente, e posteriormente na plataforma ISRobot. Plataforma esta que foi construída de raiz, sendo mais portável e obedecendo às limitações de tamanho impostas no regulamento referido na secção anterior. Esta é baseada também numa filosofia de sistema distribuído (software e hardware), à semelhança da outra. Toda a sua concepção, tal como, escolha e dimensionamento de todos os sistemas que a constituem é descrito ao pormenor em capítulos desta dissertação. São descritos todos os módulos de hardware construídos para interacção com a parte de actuação e recolha de informação por parte dos sistemas sensoriais.

1.5. Organização da Tese

A dissertação aqui apresentada apresenta-se dividida em várias partes, sendo em cada capítulo feita uma abordagem a uma determinada parte específica da evolução do trabalho desenvolvido.

No Capítulo 1 é apresentada a motivação do trabalho e o seu âmbito. São apresentados também os objectivos delineados e a sua realização.

O Capítulo 2 apresenta o estado da arte nas áreas que envolvem a Condução Autónoma de Veículos e de modo mais particular a sua aplicabilidade na prova de Condução Autónoma que se realiza durante o Festival Nacional de Robótica.

No Capítulo 3 é apresentada uma filosofia distribuída de controlo e interacção com sistemas robotizados nos quais é utilizado um barramento de comunicação de forma a tornar o sistema modular e flexível.

O Capítulo 4 apresenta os diversos componentes escolhidos e seu dimensionamento, que por sua vez fazem parte da nova estrutura robotizada.

No Capítulo 5 são apresentados alguns algoritmos que foram incorporados no PC da plataforma de forma a torná-la autónoma.

No Capítulo 6 são relatados os testes realizados com ambas as plataformas que comprovam a operacionalidade do sistema distribuído de interface com o hardware que ambas possuem.

No Capítulo 7 são apresentadas as conclusões do trabalho e apontam-se alguns caminhos para a progressão do mesmo. São também apontados de um modo genérico alguns algoritmos que são necessários para poder tirar proveito de todas as capacidades desta nova plataforma e com vista à participação na prova de Condução Autónoma.

Capítulo 2

2. Estado da Arte

2.1 Condução Autónoma 8

Para termos um robô/veículo autónomo aplicado à condução, este tem de ser capaz de interagir com o mundo de forma autónoma, recolhendo informação do mesmo e evitando situações perigosas para pessoas, para si próprio e outros objectos. Sendo portanto essencial o conhecimento da sua posição e a envolvente que o rodeia, para poder navegar, de forma segura em pisos irregulares, pouco estruturados e de visibilidade variável.

A partir sensivelmente de 1980 iniciou-se a investigação nos automóveis inteligentes, tendo a Universidade de Bundeswehr Munich (UBM) construído o primeiro veículo robô. Iniciou-se também o projecto Europeu *Prometheus* com o objectivo da construção de automóveis inteligentes. No início da década de 90 algumas universidades americanas iniciaram projectos de condução autónoma, tendo por exemplo a Universidade Carnegie Mellon em 1995 apresentado um veículo com capacidade de movimentação em estrada de forma autónoma. Também nesta altura a UBM adapta um Mercedes Classe S que consegue percorrer de forma autónoma a distância de ida e volta de Munich até Odence (Dinamarca). (Behringer, et al., 1996)

Tem-se portanto vindo a apostar cada vez mais nesta área, ao ponto da *Defense Advanced Research Projects Agency* (DARPA), promover um concurso para testes das novas tecnologias aplicadas no contexto da condução autónoma em ambientes difíceis e não estruturados com vista à aceleração da investigação em veículos terrestres autónomos. Tendo promovido em 2004 e 2005 o intitulado *DARPA Grand Challenge* e em 2007 o *DARPA Urban Challenge.*

Também a nível nacional, se tem realizado anualmente desde 2001 o Festival Nacional de Robótica (FNR). Este visa promover desenvolvimentos técnicos e científicos na área da Robótica Móvel e áreas afins (electrónica, mecânica, programação, visão por computador, inteligência artificial, navegação, controlo, etc) através de um problema

motivador, a ser resolvido por diferentes grupos de investigadores e estudantes. Procura-se ainda difundir a Ciência e a Tecnologia junto do público em geral e dos jovens em particular, nomeadamente motivando estes últimos para a aprendizagem experimental da Ciência. Este evento consiste em três actividades paralelas: uma competição de robótica móvel, um encontro científico e ainda várias demonstrações de outros robôs móveis e/ou autónomos. Este festival é composto por diversas provas diferentes, desde Futebol Robótico (diversos géneros, tais como *RoboCup*), Dança Júnior, Seguimento de Linha e também uma prova de Condução Autónoma. Sendo esta última a simulação de uma estrada em condições mais restritas, e controladas, mas com o objectivo de procurar reduzir o tempo de percurso através de um bom controlo de seguimento de pista, paredes do túnel, execução das ordens dadas pelos semáforos, contorno de obstáculos e obras na pista e estacionamento em parque, bem como, evitando penalizações (quando uma ou mais rodas saiem fora da pista, o túnel é tocado, a direcção ou a ordem de paragem ditada pelos semáforos não é cumprida.

2.1. Condução Autónoma

Existiram e existem neste momento diversos projectos/parcerias, que têm em vista a investigação e desenvolvimento de sistemas robotizados terrestres com a particularidade de serem completamente autónomos/inteligentes. Além disso existem também alguns eventos que visam promover e impulsionar o desenvolvimento das tecnologias necessárias para aqueles fins.

Tendo o Festival Nacional de Robótica o objectivo de promover desenvolvimentos técnicos e científicos na área da Robótica, tal é conseguido graças à oportunidade que este oferece, nomeadamente com a existência de um encontro científico no qual são apresentados trabalhos na área da robótica móvel, em grande parte trabalhos integrados nas provas das quais o FNR é composto. Dentro do Festival Nacional de Robótica, existe uma prova de Condução Autónoma em que é usada uma pista, semelhante a uma estrada, que visa obrigar os participantes a desenvolver um bom sistema de condução com características similares às necessárias a um automóvel autónomo num meio semi-estruturado, e cujas regras e especificações são descritas no anexo.

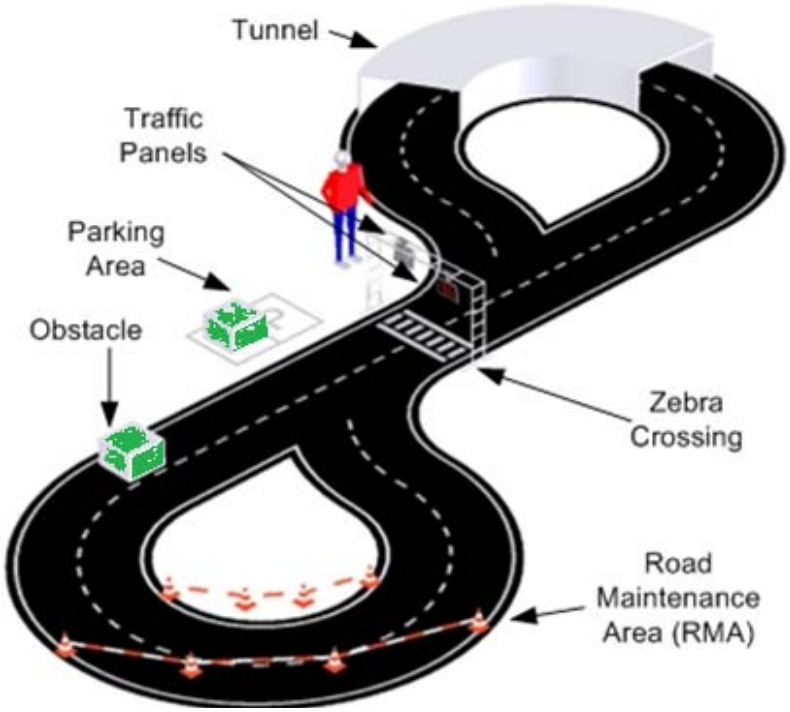


Figura 2.1: Pista de condução autónoma

Tendo em conta os vários desafios da prova, é necessário contemplar diversos tipos de sensores para a aquisição de forma correcta dos diversos elementos que esta integra. A técnica de seguimento de linhas e seguimento baseado em odometria são as opções mais fáceis, mas que não são suficientes. Havendo a necessidade de aplicação de técnicas mais complexas, tais como, reconhecimento de objectos, ou seja, identificação de passadeira, túnel, obstáculo, semáforos e zona de obras que são essenciais à realização da prova. A Tabela 2.1 apresenta exemplos de sensores utilizados nas plataformas robóticas de concorrentes, assim como alguns itens de comparação.

Tabela 2.1: Comparação entre sensores usados nas plataformas robóticas

Tipo de Sensor	Campo de Visão	Alcance	Iluminação	Custo Hardware	Complexidade Algoritmo
Câmara Convencional	médio	médio	Passivo, precisa de luz ambiente	baixo	grande
Câmara Grande Angular	grande	pequeno	Passivo, precisa de luz ambiente	médio	grande
IR proximidade	pequeno	médio	Activo, trabalho no escuro	baixo	grande
Ultra Sons	médio	médio	Activo, trabalha no escuro	baixo	grande
LASER scanner	grande	grande	Activo, trabalha no escuro	grande	baixo

- Pista Simples

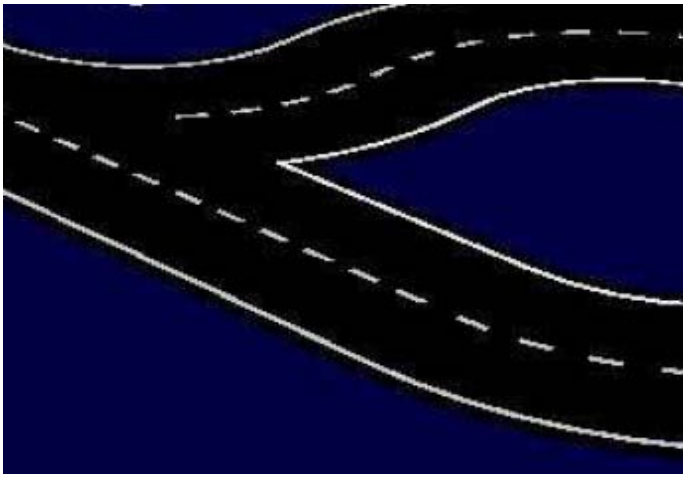


Figura 2.2: Traçado da pista

Tabela 2.2: Métodos para detecção de pista

Artigo	Técnica	Contexto	Descrição	Observações
ROTA (Azevedo, et al., Abril de 2007)	Seguimento da linha da direita da pista	Festival de Robótica	É baseado no erro de trajectória calculado a determinada distância à frente do mesmo.	A correcção da trajectória é feita por um algoritmo PD
ATLAS III (Cancela, et al., Abril de 2005)	Obtenção de um quadrilátero com os extremos da pista dentro da imagem	Festival de Robótica	Processo toma diferentes abordagens consoante os cantos do quadrilátero encontrados.	Se não detectar nenhum ponto o processo não toma nenhuma decisão e segue a direcção da última análise
VERSA Robot (André, et al., 2006)	Identificação e balizamento das linhas da pista baseado na orientação e centro de massa	Festival de Robótica	Método simplificado de Tsai, para localização dos objectos	A posição absoluta do robot no mundo é determinada através das medições da posição do robot relativamente às faixas de rodagem, seguidas de um processo de rotação e translação.
GOLD (1998) (Bertozzi, et al., Jan. 1998)	*Threshold* adaptativo da diferença dos pixéis	Pista de largura fixa e plana	Operação em frames isolados. Ampliação morfológica	Assume traçado na estrada no escuro. Alguma robustez com oclusões
Ma et al. (2000) (Ma, et al., Sep. 2000)	Probabilidade baseado no gradiente da imagem	Pista circular e plana	Operação em frames isolados. Fusão dos dados de visão com os do radar	Projectado para detecção de estradas rurais (borda desnivelada)

A Tabela 2.2 apresenta técnicas usadas para fazer o seguimento de pista presente na Figura 2.2. Na maior parte dos casos o seguimento da pista é feito recorrendo exclusivamente à visão.

- Passadeira

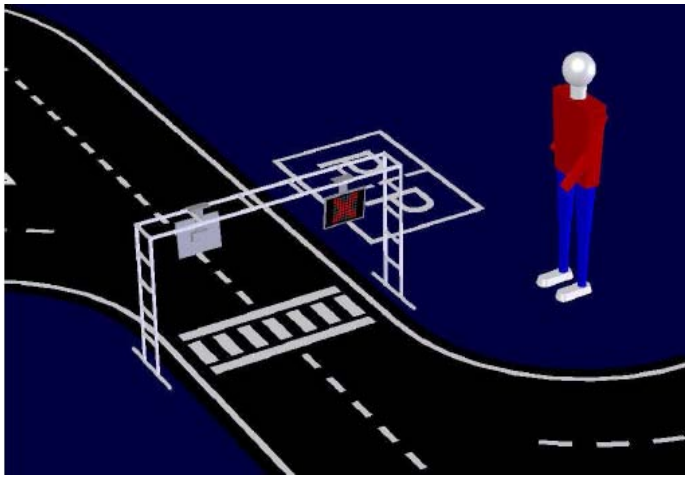


Figura 2.3: Passadeira existente na pista

As técnicas para reconhecimento da passadeira recorrem normalmente à visão por computador. Uma possibilidade consiste numa busca por uma linha grande transversal à pista, ou por uma sucessão de linhas compreendidas pelas linhas laterais da pista. A imobilização do robô junto da mesma pode ser feita recorrendo apenas a visão caso o ângulo de visão o permita ou então recorrendo a sensores de IV presentes na frente do robô, possibilitando uma exacta imobilização junto da passadeira. É usual a redução da velocidade do robô na proximidade da passadeira para a correcta identificação da informação presente no painel sinaléctico.

- Túnel

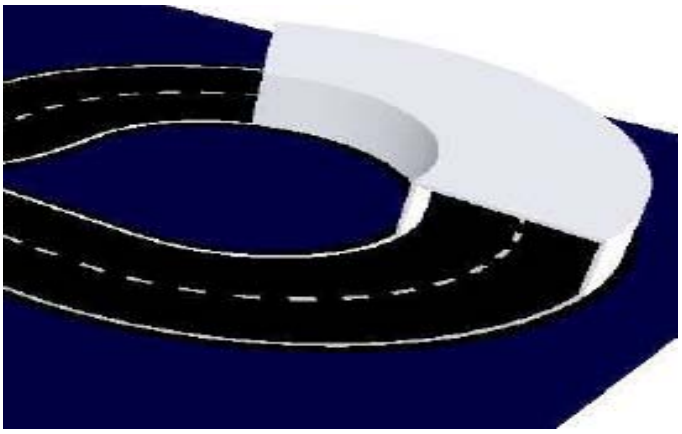


Figura 2.4: Túnel presente na pista

No caso da área da pista composta por um túnel é possível a utilização de diversas técnicas de orientação. Tais técnicas passam pela utilização de visão ou leitura da distância de cada lado do robô, através de IV, laser ou ultra-sons. No caso da visão os lados do túnel branco podem facilmente passar por linhas delimitadoras da pista, podendo apenas ser necessário um ajuste em termos de luminosidade através de luz própria ou no valor de *threshold* de binarização das imagens. No caso do uso do laser ou ultra-sons podemos medir a distância do robô às paredes do túnel. Com base nisso à que manter o robô a uma distância igual de ambos os lados do túnel.

- Obstáculo

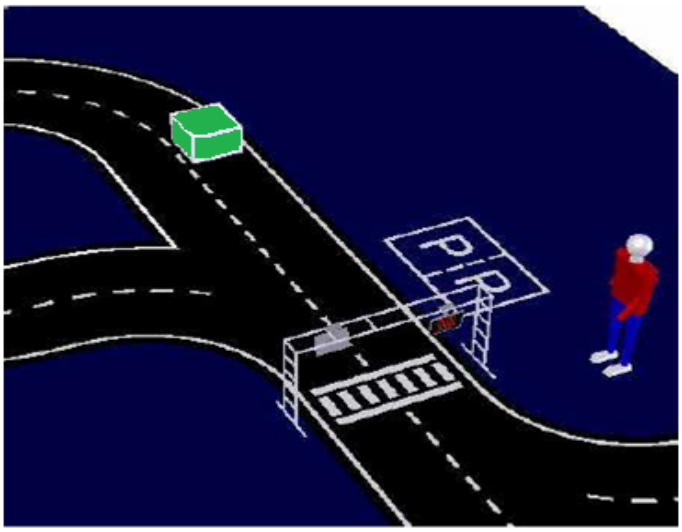


Figura 2.5: Obstáculo na pista

O obstáculo apresenta cor verde e é de dimensões tal que permitem a sua fácil identificação através do sistema de visão, mediante uma segmentação do obstáculo, ou mais uma vez recorrendo à leitura de distâncias em torno do robô, de forma a detectá-lo. O contorno do obstáculo por parte do robô é feito mediante um desvio para a outra faixa de rodagem da pista, visto o obstáculo estar a ocupar apenas uma faixa como é ilustrado na Figura 2.5.

- Semáforos (Painel Sinaléctico)

Figura 2.6: Semáforos de sinalização possíveis

Tabela 2.3: Métodos para detecção dos símbolos no semáforo

Artigo	Técnica	Contexto	Descrição	Observações
ROTA (Azevedo, et al., Abril de 2007)	Identificação de cor e forma	Festival de Robótica	É feito inicialmente uma separação por cor e posteriormente a forma é analisada com base em características morfológicas e posteriormente determinado com base em algumas regras	É aproveitado o facto das cores presentes no semáforo não inclui o azul, e o meio envolvente do mesmo (cor branca) tem presente a componente azul da cor
ATLAS III (Cancela, et al., Abril de 2005)	Identificação de cor e forma	Festival de Robótica	O fundo é anulado com base na saturação (S), pois o semáforo apresenta cores mais saturadas (mais puras), e depois com base em filtros de cor (vermelho e verde) com base em valores de (H) é identificado a cor presente, sendo que a cor amarela passa nos dois filtros. De seguida é avaliada a forma com base na comparação do centróide da forma e o correspondente ao rectângulo mínimo que envolva a forma	Ao contrário do modelo RGB, o modelo matemático da cor HSV, permite quantificar a cor numa única variável, o *Hue* (H) que traduz a tonalidade, assim como o (S) a saturação e o (V) a intensidade da luz
VERSA Robot (André, et al., 2006)	Após segmentação do sinal, análise da cor coincidente com o centro de massa	Festival de Robótica	Segmentação e identificação do objecto na imagem que demonstre o adequado tamanho	Afinação das gamas de valores (RGB) que corresponde a cada cor passível de aparecer no semáforo (vermelho, verde, amarelo)
An Active Vision System for Real-Time Traffic Sign Recognition (Jun, et al.)	Detecção de candidatos sinais baseado em cor e intensidade	Sinais de trânsito em condições reais no dia-a-dia	Após identificação de candidato é feito zoom do mesmo e é feita uma comparação com padrões existentes (*pattern matching*)	A detecção de candidatos é feita recorrendo a diversos valores de *threshold* por forma a não perder nenhum candidato, mas tendo em conta a análise dos valores da cor da imagem (YUV color space)
Road Sign Recognition (Dmitry, et al.)	Segmentação por cor. Classificação por forma.	Sinais de trânsito em condições reais no dia-a-dia	Após classificação por cor é analisada a forma (circular, triangular, rectangular) e posteriormente a forma (tipo de informação) no centro do mesmo (análise radial do centro para a extremidade como o olho humano)	As imagens recolhidas em RGB, foram transformadas para o espaço CIE XYZ e depois para LCH (*lighteness, Chroma, Hue*) através do modelo CIECAM97

No caso dos semáforos isto é uma tarefa que normalmente se utiliza apenas visão. As técnicas passíveis de serem usadas para identificação dos diversos símbolos que podem aparecer no semáforo, baseiam-se na cor e/ou forma do símbolo como se verifica na Tabela 2.3. Ao todo são cinco possíveis símbolos para identificar como se observa na Figura 2.6. Pelo facto de esses símbolos terem forma e cores diferentes, pode ser utilizado uma das duas técnicas, no entanto ou usar ambas as técnicas uma irá complementar a outra tornando o algoritmo de identificação do símbolo mais robusto.

Os símbolos tem uma particularidade que consiste em não possuírem a componente azul. As suas cores são vermelho, verde e amarelo (presença do vermelho e verde em simultâneo), ao contrário da luz natural envolvente em que a cor azul está presente de forma bem perceptível, tornando assim mais fácil a isolação do semáforo do resto da envolvente presente na imagem captada (modelo RGB).

A identificação de cor pode ser feita também convertendo a imagem obtida para o modelo matemático de cor conhecido como HSV que permite quantificar a cor numa única variável. Ao contrário do modelo RGB, o *Hue*(H) traduz a tonalidade, o (S) a saturação e o (V) a intensidade de luz. A análise de cor pode ser feita verificando o valor destas variáveis. A envolvente do semáforo é anulada no caso em que a saturação dos pixéis da imagem seja relativamente baixa, implicando a extracção apenas dos pixéis com cores mais saturadas. Cabendo a comparação de cores pela análise do valor de (H) e filtrando desta forma a cor vermelha e verde com filtros com determinado gama de valores de (H), sendo o amarelo detectado por passar nos dois filtros. Esta técnica não é suficiente pois existem símbolos diferentes com a mesma cor, mas com forma diferente. É portanto indispensável a análise da forma do símbolo.

Essa análise pode passar por modelações distintas, tais como, a comparação do envelope desse símbolo com a geometria dos possíveis candidatos de correlação, poderá ser também analisada as dimensões dum rectângulo mínimo contendo o símbolo e comparar qual o lado maior e de que lado se encontra o centróide do símbolo. É possível, mediante a análise do centróide e da área real do símbolo em comparação com a área do envelope, fazer a identificação inequívoca do símbolo presente.

- Zona Obras

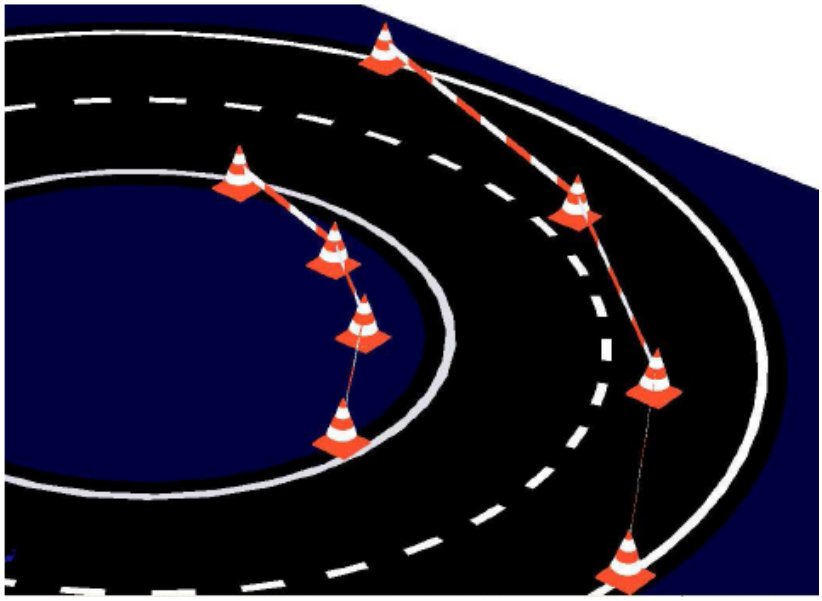


Figura 2.7: Sinalização de zona de obras

No caso da zona de obras, esta pode ser identificada de forma semelhante ao túnel e obstáculo, ou seja esta zona é delimitada por cones e uma fita (altura predefinida) que os interliga ao longo de toda esta zona como ilustra a Figura 2.7. Deste modo, com um sensor de distâncias é possível fazer o seu seguimento à semelhança do túnel. Através de visão também é possível a identificação dos cones. A identificação da fita poderá não ser tão trivial dadas as condições de aquisição da câmara (ângulo). Sendo os cones de cor laranja é possível identificá-los por exemplo com uma procura na imagem de forma radial, detectando a transição da cor preta ou branca para laranja.

- Parque

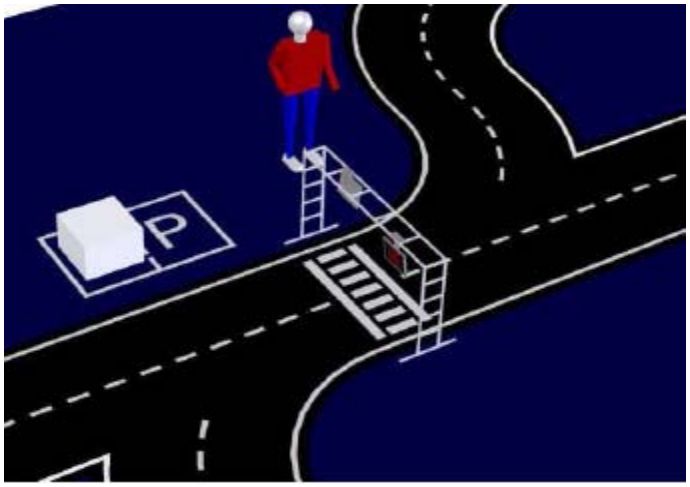


Figura 2.8: Parque parcialmente ocupado

A identificação do parque pode ser realizada tendo por base a procura de um rectângulo fechado com duas zonas distintas (dois quadrados com a letra P), estando um deles possivelmente ocupado pelo obstáculo já descrito anteriormente como se observa pela Figura 2.8. Devido a este motivo é necessário o recurso a visão, com o auxílio de uma medida da distância do robô ao obstáculo de forma a evitar o contacto.

Após a análise de todas estas particularidades da pista, conclui-se que é indispensável um sistema de visão para os semáforos, e para o seguimento da pista e consequente identificação dos diversos constituintes da pista. A inclusão de outros sensores como laser, ultra-sons ou infravermelhos é complementar ao sistema de visão na análise de objectos com relevo e na determinação da localização destes em relação ao robô, sendo o uso destes sensores mais simples para a detecção de objectos do que recorrendo a visão. O laser é aquele que apresenta um maior potencial para estas tarefas. Da análise da informação do laser conjuntamente com a visão é possível a detecção e localização de grande parte dos obstáculos de forma mais eficiente (simples e rápida).

Das soluções apresentadas nem todas chegaram a ser implementadas sendo uma base de trabalho para o futuro, e que como se verifica pela Tabela 2.2 e Tabela 2.3 são usadas pela maioria dos outros robôs que participam na prova.

Capítulo 3

3. Arquitectura Distribuída

Neste capítulo apresentam-se os vários componentes que constituem a arquitectura distribuída, assim como as suas principais características.

Esta arquitectura foi inicialmente idealizada e implementada na plataforma RobChair. Depois de testada foi exportada para a nova plataforma ISRobot, comprovando a flexibilidade e modularidade deste tipo de arquitectura.

Dadas as duas plataformas não serem constituídas pelos mesmos dispositivos de hardware, também os módulos de interface que interagem com o hardware vai ter particularidades diferentes.

3.1. Módulo de Processamento "PIC_Base"

Nesta secção apresentam-se as características do PIC utilizado no módulo de hardware PIC_Base, módulo este que por sua vez é usado nos nós de actuação e aquisição do sistema de controlo. O uso dos microcontroladores, neste caso PIC, traz muitas vantagens para este tipo de sistemas onde se pretende uma evolução constante, requerendo-se flexibilidade e modularidade. Estes oferecem inúmeras capacidades, de tratamento, análise e comunicação de dados/sinais, e uma panóplia de periféricos integrados que evita assim muita electrónica externa. Este facto confere mais flexibilidade às aplicações que integram, recaindo sobre este tipo de dispositivo a escolha para o desenvolvimento dos vários componentes que constituem o sistema.

Os microcontroladores utilizados neste sistema são da família PIC18FXX8, sendo o modelo utilizado neste trabalho o PIC18F258, o qual integra diversos periféricos, como comunicação via RS232, CAN, SPI, ou mesmo ADCs e geradores de sinais de PWM.

A "*PIC_Base*" é um módulo de hardware utilizado para fazer o interface entre os PICs e os vários módulos de hardware desenvolvidos. Este módulo de interface tem várias capacidades, das quais se destaca a possibilidade de programação do PIC através do programador disponível da Microchip(R) que é o MPLAB ICD 2(R). Este programador é utilizado na fase inicial para programação do *bootfirmware* de cada PIC que incorpora a sua identidade inequívoca no barramento de CAN para posterior acesso. Para além da possibilidade de programação, o módulo PIC_Base permite a ligação fácil a todos os portos por meio de uma ficha de 20 pinos e dispõe dos *transceivers* de ligação ao RS232 e CAN. A comunicação CAN é também usada para a programação dos microcontroladores recorrendo ao programa "canbootmngr_v2" que vai interagir com o *bootfirmware* previamente colocado no PIC. A identificação dos módulos é feita tendo em conta dois factores. Caso o módulo tenha funções semelhantes é agrupado no mesmo grupo, ou seja, a sua identificação é feita pelo grupo a que pertence e ordem que ocupa no grupo como descrito nas Tabelas 3.1 e 3.2.

Tabela 3.1: ID's utilizados pelo programa "canbootmngr_v2 para a plataforma RobChair"

CanBootManager	Group	PIC
PDriveNode	1	1 (Left) / 2 (Right)
EncoderNode	2	1 (Left) / 2 (Right)
JoystickNode	4	1
TriggerNode	6	1

Tabela 3.2: ID's utilizados pelo programa "canbootmngr_v2 para a plataforma ISRobot"

CanBootManager	Group	PIC
IRNode	1	1
USNode	1	2
PDriveEncoderNode	2	1 (Left) / 2 (Right)
ControlNode	4	1
TriggerNode	6	1

O programa “canbootmngr_v2” é usado essencialmente para programação das funcionalidades de cada PIC via CAN, permitindo desta forma programar todos os PICs sem ter de conectá-los um a um ao programador da Microchip. Para além de programar é possível transmitir uma ordem aos PIC´s para saírem do estado *“Boot Loader Mode”* e passarem para *“Idle Mode”* através do programa “canbootmngr_v2”. O comando completo para programar um determinado PIC com o ficheiro “.hex” criado pelo compilador, neste caso o TriggerNode é o seguinte: “./canbootmngr_v2 -g 6 -p 1 -d -f TriggerNode.hex”. A interpretação do comando significa fazer o *download* do código para o PIC com a identificação expressa.

Os microcontroladores possuem três modos de operação:*“Boot Loader Mode”*, *“Idle Mode”* e *“Running Mode*” como se exemplifica na Figura 3.1. O *“Boot Loader Mode”* é o estado inicial em que é permitida a sua reprogramação. Passando para *“Idle Mode”* este fica num estado de espera. Em *“Running Mode*” este efectua o desejado controlo do processo. A Figura 3.1 apresenta as transições de estado possíveis.

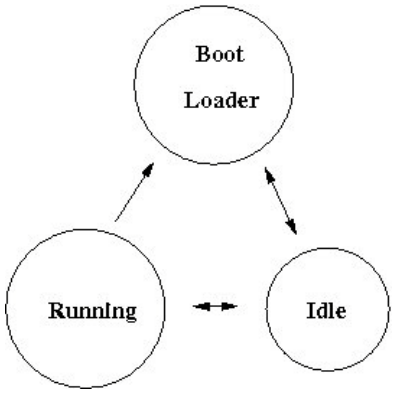


Figura 3.1: Modos de operação

3.2. Detalhes da Comunicação CAN

O CAN assenta num barramento de comunicação série do tipo multi-emissor. A taxa de transmissão deste barramento pode ascender a 1Mbit/s.

O protocolo foi desenvolvido pela companhia alemã *Robert Bosch GmbH* em meados da década de 1980, para uso na indústria automóvel, encontrando-se documentado nas normas ISO2 11898 para aplicações de elevada velocidade e ISO 11519-2 para aplicações de baixa velocidade. Actualmente, é largamente utilizado em automação industrial e em aplicações de controlo.

Uma das características deste protocolo é permitir uma excelente detecção de erros e respectivo tratamento, o que lhe confere uma boa operacionalidade em ambientes ruidosos. Neste barramento, as mensagens transmitidas não contêm o endereço do nodo emissor nem do nodo receptor. Em vez disso, o conteúdo de cada mensagem é rotulado com um identificador que é único em toda a rede. Todos os nodos da rede recebem a mensagem enviada e cada nodo decide, em função do identificador, se a mensagem é ou não relevante para o nodo em questão. Se a mensagem for relevante será processada, caso contrária será ignorada. O identificador determina também a prioridade da mensagem. Quanto mais baixo for o valor numérico do identificador, maior será a prioridade da mensagem associada. Numa situação em que dois ou mais nodos tentem transmitir simultaneamente, uma técnica não destrutiva garante que as mensagens são transmitidas pela sua ordem de prioridades e que nenhuma mensagem é perdida.

A estrutura global de uma mensagem segundo o protocolo CAN é ilustrada na Figura 3.2. Para que os vários dispositivos troquem a informação correctamente sob o protocolo CAN não pode existir nodos na rede com o mesmo identificador (ID). Ora tendo o ID da mensagem 11bits, este foi subdividido em três partes para contemplar a informação respeitante não apenas ao destino como também a origem e a função desejada para com os dados presentes na mensagem.

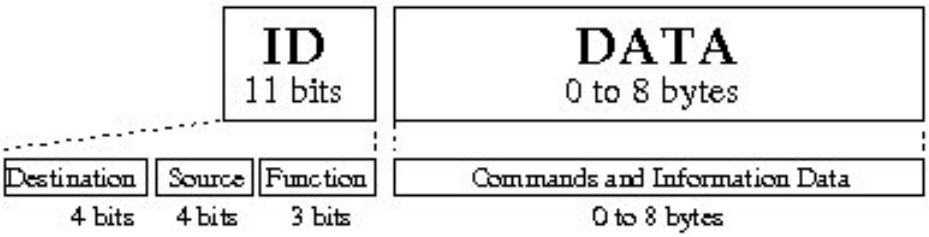


Figura 3.2: Protocolo CAN

Os ID dos módulos estão descritos nas Tabelas 3.3 e 3.4, no caso da função esta é dependente do módulo em questão sendo portanto apresentada na secção do módulo a que respeita, assim como o campo de dados, cujo tamanho também irá depender dessa mesma função.

Tabela 3.3: ID's de "Destination/Source" usados no protocolo CAN na plataforma **RobChair**

ID		Módulo
0		PC
1		*Right PDrive*
2		*Left PDrive*
3		*Both PDrive*
4		*Right Encoder*
5		*Left Encoder*
6		*Both Encoder*
10		*Joystick*
15		*syncMCU*

Tabela 3.4: ID's de "Destination/Source" usados no protocolo CAN na plataforma **ISRobot**

ID		Módulo
0		PC
1		*UltraSound*
2		*InfraRed*
4		*Right PDriveEncoder*
5		*Left PDriveEncoder*
6		*Both PDriveEncoder*
15		*syncMCU*

Como se poderá verificar mais à frente, existem duas funções comuns a todos os módulos que são "*Turn Node OFF → Idle Mode*" e "*Turn Node ON → Running Mode*", as quais correspondem aos IDs "0" e "1" respectivamente.

3.3. Plataforma RobChair & ISRobot

As Figs 3.4 e 3.5 mostram os vários constituintes e suas interligações, das plataformas RobChair e ISRobot.

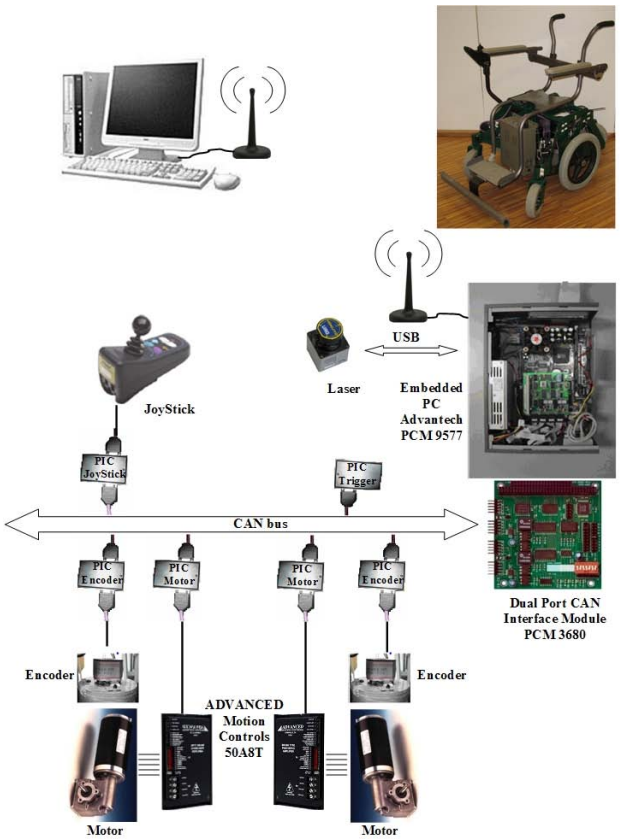


Figura 3.3: Estrutura de hardware de aquisição, controlo e actuação da RobChair

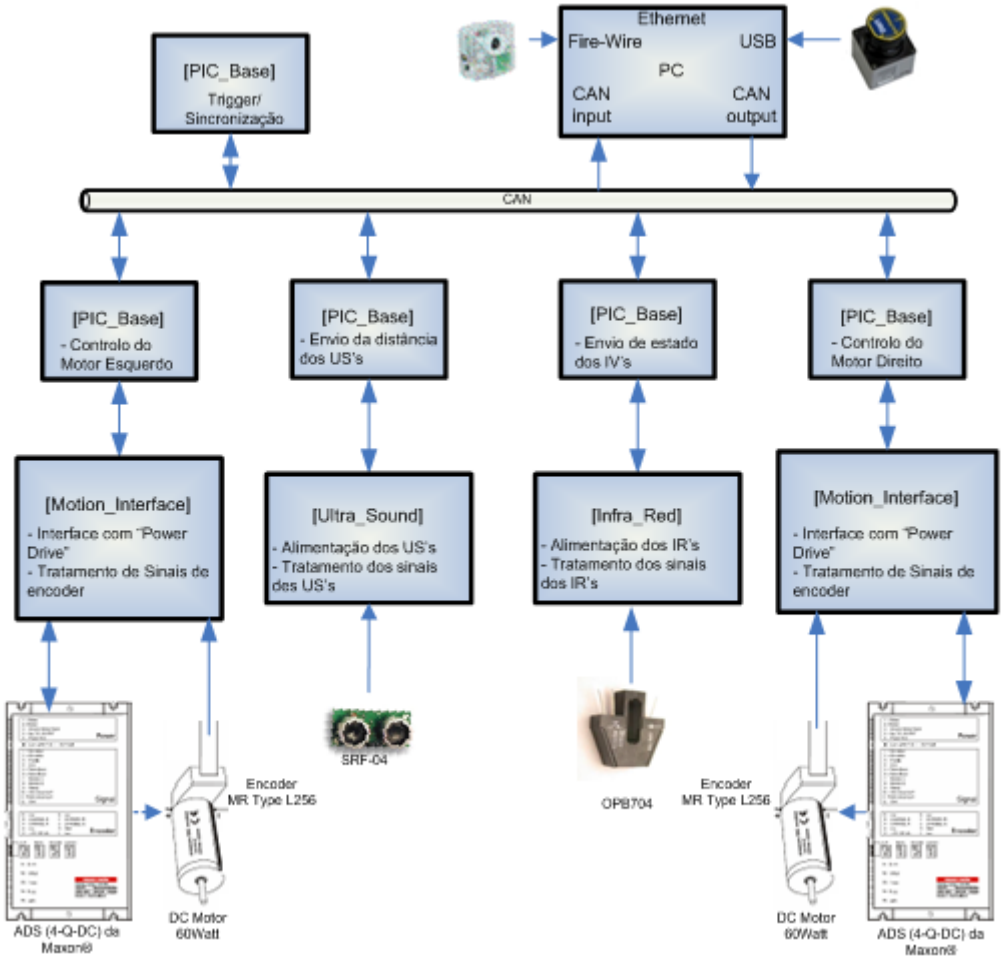


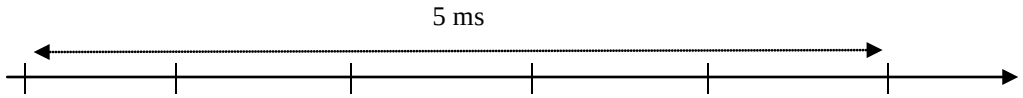
Figura 3.4: Estrutura de hardware de aquisição, controlo e actuação do ISRobot

Como é perceptível, toda a informação circula entre os vários PICs e o PC através de um barramento de CAN. O PC funciona como nó supervisor/planeador recebendo e transmitindo informação aos PICs e com ordens de aquisição e actuação.

Estas ordens obedecem a condicionantes temporais, e estão devidamente planificadas como se observa nos diagramas temporais que são apresentados nas Tabelas 3.5 e 3.6, existindo portanto uma sequência das várias acções a serem executadas no sistema. Estas operações temporais repetem-se de forma cíclica com um período de 5ms, através das mensagens *SyncMCU* que o PIC Trigger envia de forma periódica e de 1ms sincronizando todas as acções do sistema.

No caso da RobChair esta é passível de controlo com o joystick efectuando um controlo manual e sem a necessidade de sensores. No entanto esta também possui algoritmos para seguimento de trajectória baseado em mapas predefinidos (*off-line map*).

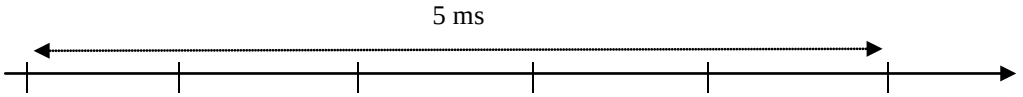
Tabela 3.5: Diagrama temporal de acções da RobChair



Pic Trigger	SyncMCU (N=1)	SyncMCU (N=2)	SyncMCU (N=3)	SyncMCU (N=4)	SyncMCU (N=5)	SyncMCU (N=1)
Pic Motor	Motor Actuation				Receive Wheels Velocity Command	Motor Actuation
Pic Encoder			Read/Send Wheels Velocity Receive Wheels Velocity Reference	Calc Wheels Velocity Command	Send Command	
Pic Joystick	Read Position	Send Position				Read Position
Embedded PC		Receive Position Calc Wheels Velocity Reference	Send Wheels Velocity Reference			

No caso do ISRobot este não permite o controlo manual através de joystick, em vez disso possui diversos sensores que lhe permite ter a noção do caminho a percorrer recorrendo a visão (*on-line map*) e outros sensores.

Tabela 3.6: Diagrama temporal de acções do ISRobot



Pic Trigger	SyncMCU (N=1)	SyncMCU (N=2)	SyncMCU (N=3)	SyncMCU (N=4)	SyncMCU (N=5)	SyncMCU (N=1)
Pic Motion	Motor Actuation				Receive Wheels Velocity Reference	Motor Actuation
Pic Infra Red	Read State	Send State				Read State
Pic Ultra Sound	Read Distance	Send Distance				Read Distance
Embedded PC		Receive Sensor Information	Calc Wheels Velocity Reference	Send Wheels Velocity Reference		

3.3.1. Módulo "Trigger"

Em termos de funções de CAN para este módulo - presente em ambas as plataformas -, este apresenta as seguintes:

Tabela 3.7: ID's das funções presentes no módulo Trigger

ID		Função
0		*Turn Node OFF*
1		*Turn Node ON*
15		*Synchronize Nodes*

A função "Synchronize All Nodes", é usada para sincronizar a malha de controlo do sistema. Sendo o tempo do ciclo de controlo de 5ms, é enviada uma mensagem de sincronismo a cada 1ms, com a indicação da fase do ciclo em que o sistema se encontra, para que sejam efectuadas todas as acções dos diversos módulos de uma forma cíclica e sincronizada. Tal mensagem é elaborada de acordo com o apresentado na Figura 3.5.

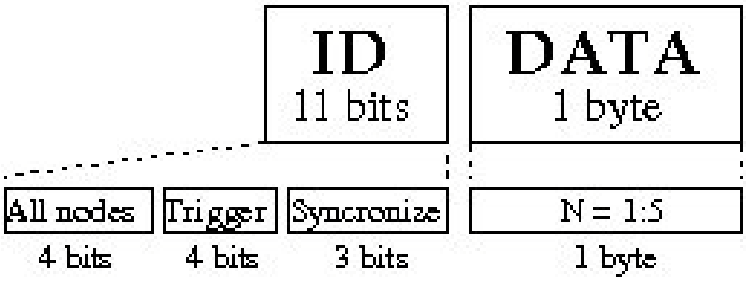


Figura 3.5: Mensagem de sincronização do sistema

3.3.2. Módulo "PDrive_Interface"

As funções de CAN para este módulo - presente na plataforma RobChair -, são identificadas na Tabela 3.8.

Tabela 3.8: ID's das funções presentes no módulo *PDrive*

ID		Função
0		*Turn Node OFF*
1		*Turn Node ON*
2		*Set DAC Command*
3		*Set Control Mode*
4		*Data from Motor*

A função "*Set DAC Command*", é usada para definir o valor a transmitir ao DAC que por sua vez irá corresponder a uma velocidade do motor, sendo esse valor composto por 2 bytes (DAC de 10 bits de resolução).

No entanto, existem dois modos de funcionamento relativos a comandos para o motor, designadamente: modo directo e modo de velocidade. Estes são comutados pela função "*Set Control Mode*". No primeiro caso o comando do motor pode vir directamente do joystick por exemplo. Neste caso o sistema de velocidade é em malha aberta. No segundo caso existe um controlador PI no módulo *EncoderNode* que irá assegurar controlo de velocidade de acordo com a referência (comando de velocidade), sendo portanto os comandos do motor enviados por este módulo e não pelo PC ou joystick. Os dados da mensagem irão ser "0" ou "1", de acordo com o modo de funcionamento desejado respectivamente.

Os valores monitorizados pelo módulo totalizam seis bytes, em que dois bytes correspondem ao valor lido pelo ADC, seguido de mais dois bytes referentes à corrente do motor e por fim mais dois bytes correspondendo ao valor de comando recebido pelo módulo. Estes dados são transmitidos sobre a função "*Data from Motor*".

O fluxograma presente na Figura 3.6 mostra resumidamente a estrutura do código desenvolvido para o PIC responsável pelo envio de comandos para o motor, o qual irá corresponder a uma determinada velocidade.

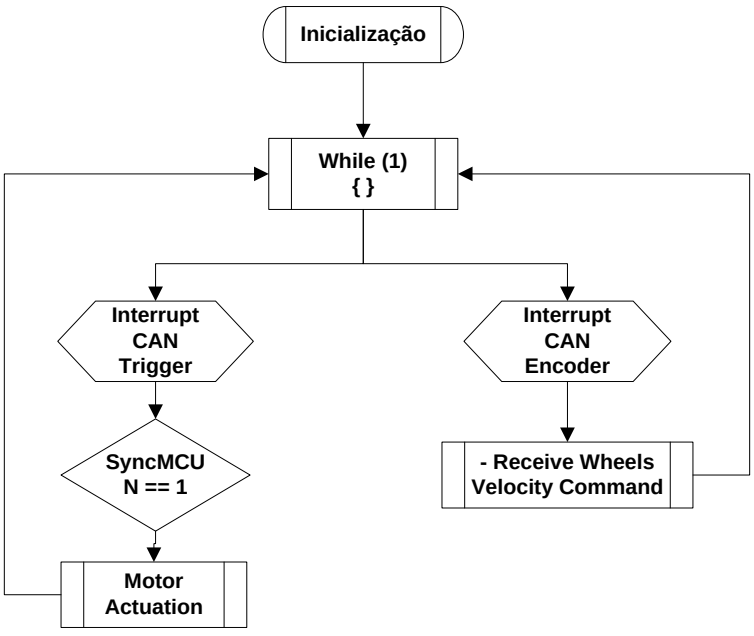


Figura 3.6: Fluxograma do código implementado no *PDrive Node*

Como é constatado pelo fluxograma presente na Figura 3.6, o *PDrive Node* recebe pelo barramento de CAN o comando desejado para o controlo do motor. Esse comando é concretizado em acção sobre o motor quando o *PDrive* recebe uma mensagem de sincronismo sinalizando o instante N=1 do ciclo.

3.3.3. Módulo "Encoder_Interface"

As funções de CAN para este módulo - presente na plataforma RobChair -, são identificadas na Tabela 3.9.

Tabela 3.9: ID's das funções presentes no módulo *Encoder*

ID		Função
0		*Turn Node OFF*
1		*Turn Node ON*
2		*Set Velocity Value*
3		*Set Control Mode*
4		*Set PI Control Values*
5		*Reset Encoder Information*
6		*Data from Encoder*

A função "*Set Velocity Value*", é usada para definir o valor de referência da velocidade desejada para o motor, sendo esta controlada com base no controlador PI presente neste módulo. Se este estiver em modo de velocidade tal é possível através da função "*Set Control Mode*"como descrito no *PDriveNode*. Para defenição de tal velocidade são usados dois bytes de dados
É também possível alterar os valores do controlador, sendo para tal necessário quatro bytes para o ganho proporcional e outros quatro para o ganho integral. Estes valores são inteiros que devem estar referenciados às milésimas de unidade.

Os valores monitorizados pelo módulo totalizam seis bytes, em que quatro bytes correspondem ao valor da posição do motor, seguido de mais dois bytes referentes à velocidade do motor. Estes são os dados transmitidos sobre a função "*Data from Encoder*". No entanto, o valor da posição pode ser reinicializado, sendo para tal usada a função "*Reset Encoder Information*".

O fluxograma presente na Figura 3.7 mostra resumidamente a estrutura do código desenvolvido para o PIC responsável pela recepção de comandos de referência para a

velocidade do motor. Note-se que a leitura da velocidade real é também realizada por este módulo o que possibilita a aplicação do controlador PI da velocidade.

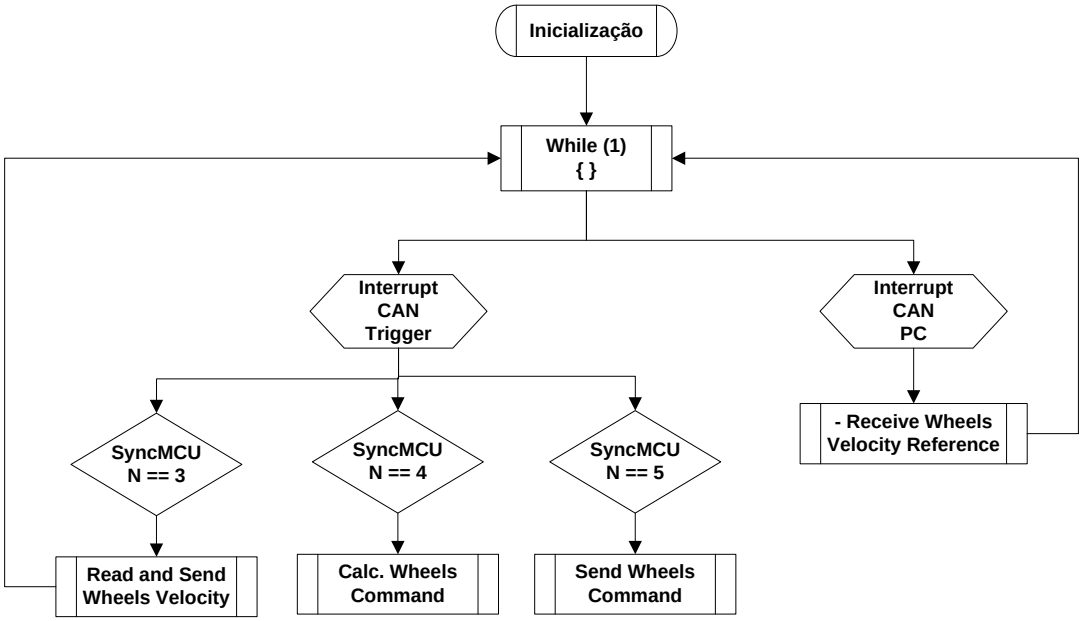


Figura 3.7: Fluxograma do código implementado no *Encoder Node*

Como é constatado pelo fluxograma presente na Figura 3.7, o *Encoder Node* recebe pelo barramento de CAN da camada de alto nível, a referência de velocidade para controlo do processo. Efectua a leitura e envio da velocidade real, calcula o comando desejado com base na referência de velocidade recebida e envia o comando resultado do PI.

3.3.4. Módulo "Joystick_Interface"

As funções de CAN para este módulo - presente na plataforma RobChair -, são as identificadas na Tabela 3.10.

Tabela 3.10: ID's das funções presentes no módulo *Joystick*

ID		Função
0		*Turn Node OFF*
1		*Turn Node ON*
4		*Data from Joystick*

Os valores monitorizados pelo módulo totalizam seis bytes, em que os dois primeiros bytes correspondem ao valor desejado para a velocidade linear, seguido de mais dois bytes referentes ao valor desejado da velocidade angular e por fim outros dois bytes de referência do centro dos valores medidos, sendo estes os dados transmitidos sobre a função "*Data from Joystick*".

O fluxograma presente na Figura 3.8 mostra resumidamente a estrutura do código desenvolvido para o PIC responsável pelo envio dos comandos de referência dados pelo joystick.

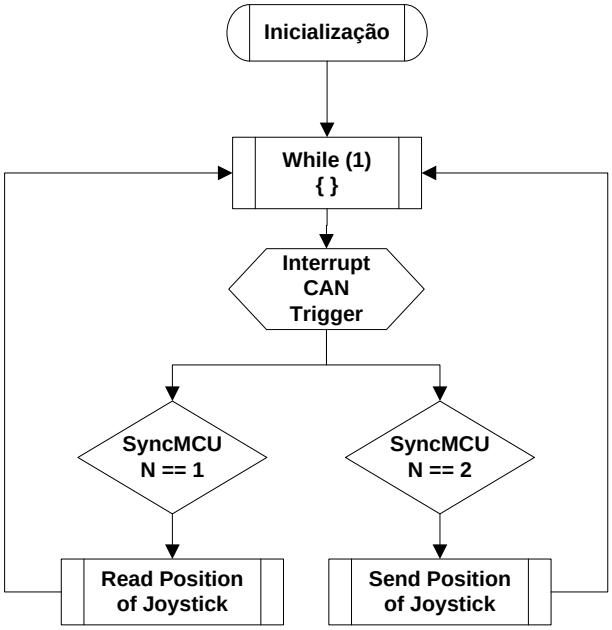


Figura 3.8: Fluxograma do código implementado no Joystick *Node*

Como é constatado pelo fluxograma presente na Figura 3.8, o Joystick envia pelo barramento de CAN para a camada de alto nível, comandos de referência da velocidade desejada para o robô de acordo com a posição do joystick.

3.3.5. Módulos de Software Presentes no PC da RobChair

O fluxograma presente na Figura 3.9 apresenta resumidamente a estrutura do código implementada no PC, para a recolha da informação proveniente do barramento de CAN.

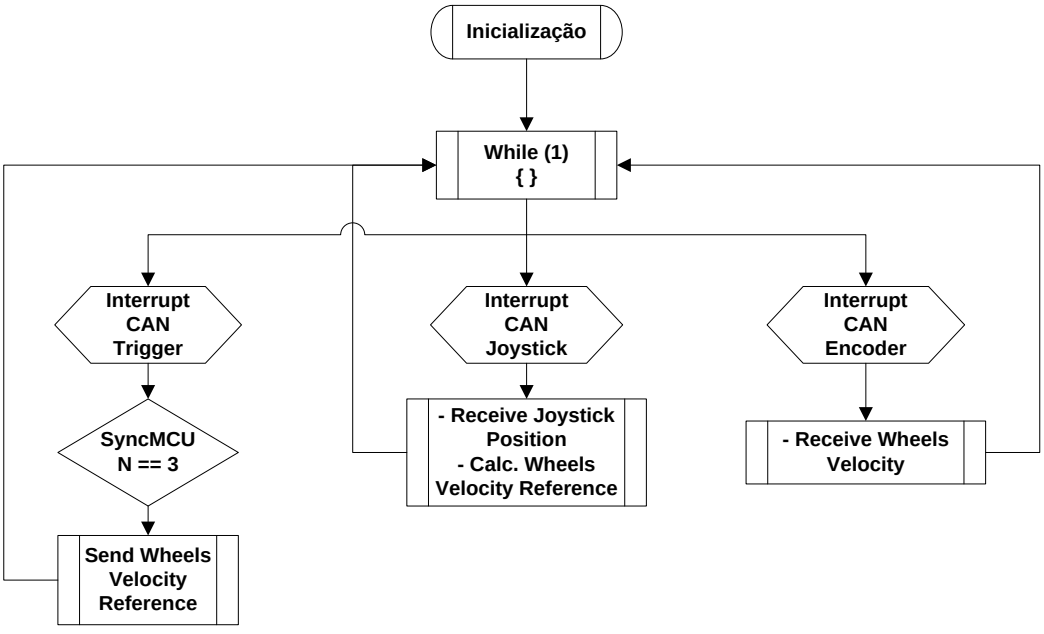


Figura 3.9: Fluxograma dos módulos de controlo no PC

Como é constatado pelo fluxograma presente na Figura 3.9, o PC recebe através do barramento de CAN a informação do processo da camada de baixo nível, sendo a recolha da informação respeitante a cargo da camada de interligação. Com base na informação do joystick é controlado a velocidade desejada para cada motor, sendo essa velocidade de referência enviada para o controlador PI.

3.3.6. Módulo "Motion_Interface"

As funções de CAN para este módulo - presente na plataforma ISRobot -, são as identificadas na Tabela 3.11.

Tabela 3.11: ID's das funções presentes no módulo *Motion*

ID		Função
0		*Turn Node OFF*
1		*Turn Node ON*
2		*Set DAC Command*
4		*Data from Motion*

A função "*Set DAC Command*", é usada para definir o valor a transmitir ao DAC que por sua vez irá corresponder a uma velocidade do motor, sendo esse valor composto por 2 bytes (DAC de 10 bits de resolução).

Os valores monitorizados pelo módulo totalizam oito bytes, em que quatro bytes correspondem ao valor acumulado dos pulsos do encoder, seguido de mais dois bytes referentes à velocidade do motor e por fim mais dois bytes correspondendo ao valor de comando recebido pelo módulo. Estes são os dados transmitidos sobre a função "*Data from Motion*".

O fluxograma presente na Figura 3.10 mostra resumidamente a estrutura do código desenvolvido para o PIC responsável pelo envio de comandos para o motor, o qual irá corresponder a uma determinada velocidade.

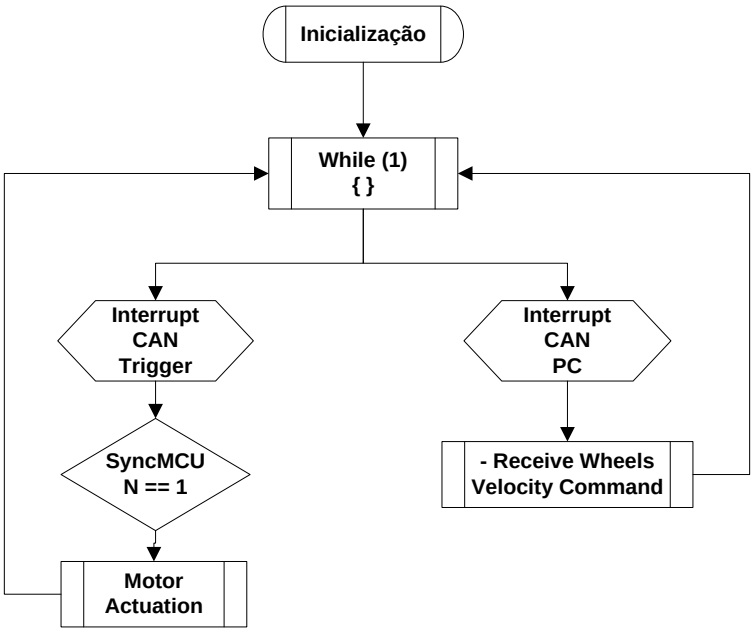


Figura 3.10: Fluxograma do código implementado no *Motion Node*

Como é constatado pelo fluxograma presente na Figura 3.10, o *Motion Node* recebe pelo barramento de CAN da camada de alto nível, comandos para controlo da velocidade do motor. Esse comando é concretizado em acção sobre o motor quando o *Motion* recebe uma mensagem de sincronismo sinalizando o instante N=1 do ciclo.

3.3.7. Módulo "Ultra_Sound_Inteface"

As funções de CAN para este módulo - presente na plataforma ISRobot -, são as identificadas na Tabela 3.12.

Tabela 3.12: ID's das funções presentes no módulo *Ultra Sound*

ID		Função
0		*Turn Node OFF*
1		*Turn Node ON*
4		*Data from US*

Os valores monitorizados pelo módulo totalizam no máximo oito bytes, em que cada byte corresponde ao valor medido por cada ultra-som, sendo estes os dados transmitidos sobre a função "*Data from US*". É possível monitorizar oito ultra-sons no máximo, sendo que se estiverem apenas dois ligados, apenas dois bytes de dados serão transmitidos.

O fluxograma presente na Figura 3.11 mostra resumidamente a estrutura do código desenvolvido para o PIC responsável pelo envio do estado dos infra-vermelhos.

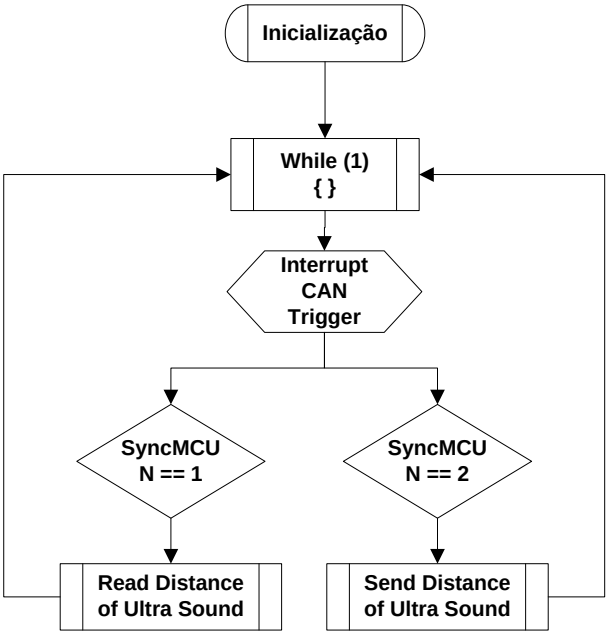


Figura 3.11: Fluxograma do código implementado no Ultra Sound *Node*

Como é constatado pelo fluxograma presente na Figura 3.11, o *Ultra Sound Node* envia pelo barramento de CAN para a camada de alto nível, o valor de distância medido por cada ultra-som de acordo com as mensagens de sincronismo.

3.3.8. Módulo "Infra_Red_Interface"

As funções de CAN para este módulo - presente na plataforma ISRobot -, são as identificadas na Tabela 3.13.

Tabela 3.13: ID's das funções presentes no módulo *Infra Red*

ID		Função
0		*Turn Node OFF*
1		*Turn Node ON*
4		*Data from Infra Red*

Os valores monitorizados pelo módulo totalizam dois bytes, sendo usados dez bits alinhados à direita, com a informação do estado de cada infravermelho sobre a função "*Data from Infra Red*".

O fluxograma presente na Figura 3.12 mostra resumidamente a estrutura do código desenvolvido para o PIC responsável pelo envio do estado dos infra-vermelhos.

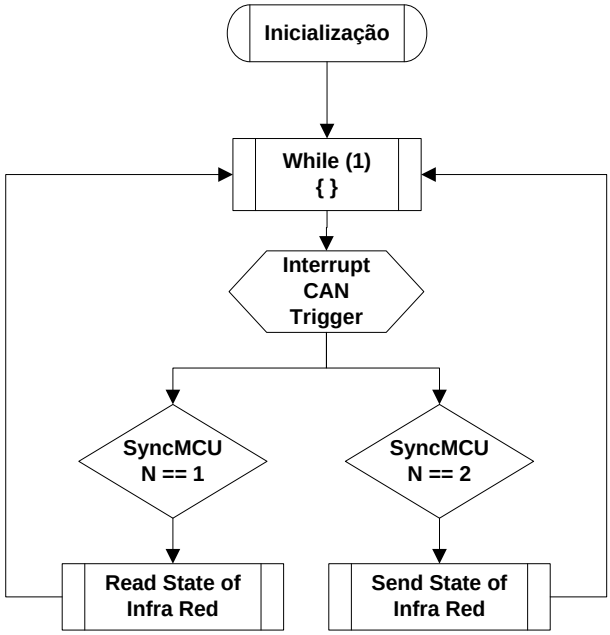


Figura 3.12: Fluxograma do código implementado no Infra Red *Node*

Como é constatado pelo fluxograma presente na Figura 3.12, o *Infra Red Node* envia pelo barramento de CAN para a camada de alto nível, o estado dos infra-vermelhos anteriormente lidos de acordo com as mensagens de sincronismo.

3.3.9. Módulos de Software Presentes no PC do ISRobot

O fluxograma presente na Figura 3.13 representa resumidamente a estrutura do código implementada no PC, para a recolha da informação proveniente do barramento de CAN.

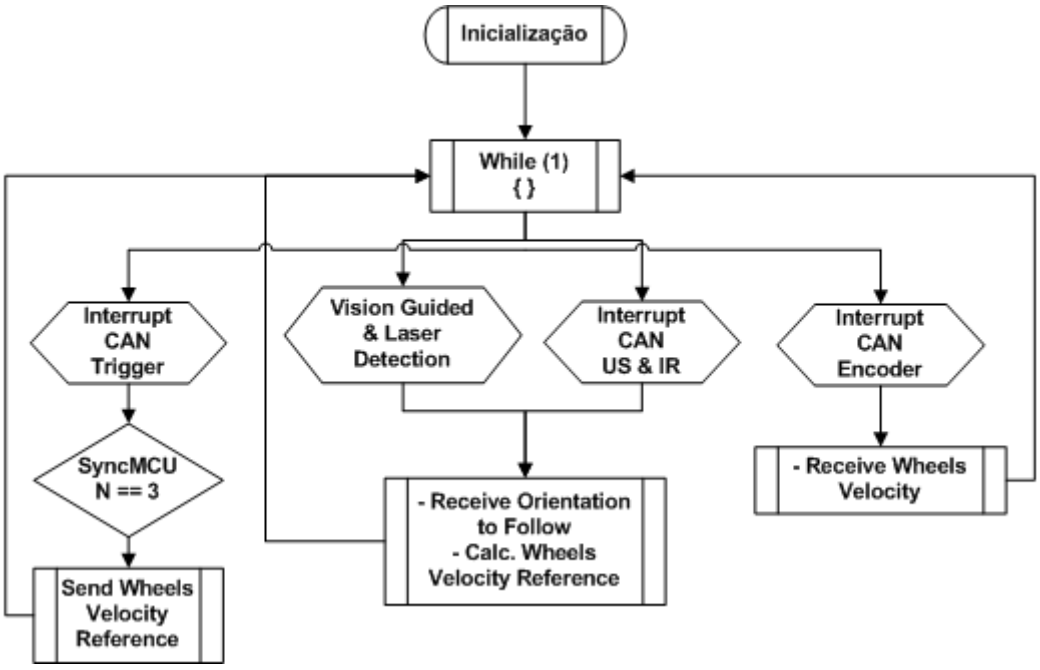


Figura 3.13: Fluxograma dos módulos de controlo no PC

Como é constatado pelo fluxograma presente na Figura 3.13, o PC recebe pelo barramento de CAN da camada de baixo nível, informação do processo o qual este tem de controlar, sendo a recolha da informação respeitante a cargo da camada de interligação. É com base na informação recolhida tanto do CAN como do laser e visão que é calculada a orientação a seguir. Com base nessa orientação é calculada a velocidade desejada para os motores e é posteriormente enviada respeitando o sincronismo e instante alocado para essa tarefa.

Capítulo 4

4. ISRobot: Concepção Estrutural

Neste capítulo são apresentados os fundamentos teóricos e os cálculos de dimensionamento necessários para a definição dos vários componentes que constituem o robô (ISRobot), apresentando as suas descrições e principais características.

4.1. Motores

Este sub-capítulo é dedicado ao estudo efectuado para a determinação dos parâmetros dos motores a utilizar, sendo apresentados os cálculos teóricos efectuados e as características pelas quais a escolha recaiu sobre o modelo apresentado.

Para dimensionamento dos motores a utilizar no robô, aplicaram-se as Leis de Newton. Pretendem-se uma velocidade (v) e uma aceleração (a) máxima de 2m/s e 2m/s^2 respectivamente, ou seja, que o robô acelere até à velocidade máxima em apenas um segundo. Sabendo que a massa do robô é aproximadamente 20 Kg, a expressão (4.1) indica a força necessária para fazer deslocar um corpo com massa (m), e aceleração (a).

$$F = m.a = 40\ N \tag{4.1}$$

Desprezando o facto do centro de massa não coincidir com o eixo das rodas, pode-se obter o binário necessário para produzir a força apresentada em (4.1), através da Equação 4.2:

$$\tau = F.r = 4\ N.m \tag{4.2}$$

Onde r é o raio das rodas do robô (r = 0,1 m). À velocidade máxima do robô a velocidade angular das rodas será:

$$\omega = \frac{V}{2.\pi.r} = 3.183\ rad/s \tag{4.3}$$

Pelo que a velocidade em r.p.m. vem dada por.

$$\omega(r.p.m.) = \frac{\omega}{2.\pi}.60 = 190{,}98\ r.p.m. \tag{4.4}$$

Desta forma partiu-se para uma solução integrada, com motor DC alimentado a 24 V acoplado a um sistema de engrenagem de redução de velocidade, permitindo desta forma ter um binário mais elevado e velocidades perto da gama pretendida. A curva velocidade/binário para o motor RE 30 (Graphite Brushes, 60Watt) da Maxon®, sobre o qual recaiu a escolha, é a apresentada na Figura 4.1.

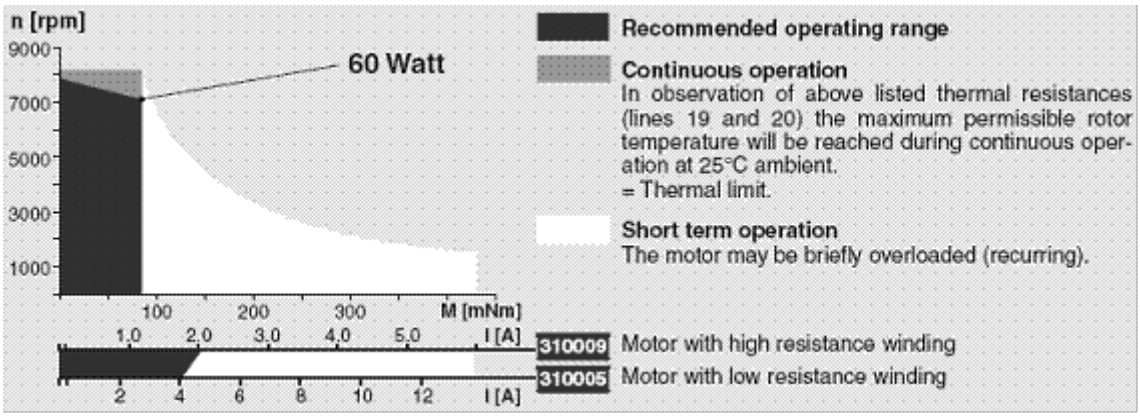


Figura 4.1: Curva característica do motor

A este motor é acoplado uma caixa de engrenagem GP32A (*Planetary Gearhead* 0,75-4,5Nm) da Maxon®, com uma redução aproximada de 1/33 (redução real 529/16), que permite desta forma, e de acordo com a curva característica do motor, que a velocidade máxima das rodas seja dada pela Equação 4.5:

$$\omega_{Rodas} = \frac{\omega_{motor}}{33} \cong \frac{7000}{33} = 212{,}12\ rpm \tag{4.5}$$

Para um binário máximo, por motor, dado pela Equação 4.6.

$$\tau_{Rodas} = \tau_{motor}.33 \cong 0{,}08 \times 33 = 2.64\ Nm \tag{4.6}$$

Como o sistema é movido com recurso a dois motores, cada um acoplado a uma das rodas, então obtém-se um binário resultante que será o dobro do apresentado na Equação 4.6, ou seja, um binário resultante total de aproximadamente *5,28 Nm*. Isto garante que o binário resultante dos motores é suficiente para as características velocidade/binário que foram apresentadas como objectivo para o robô.

4.2. Codificador Óptico

Neste sub-capítulo pretende-se especificar as características do codificador óptico ("*encoder*") utilizado de forma a fundamentar essa escolha. O codificador óptico foi seleccionado de acordo com a indicação do fabricante do motor, estando este acoplado ao motor como se mostra na Figura 4.2.

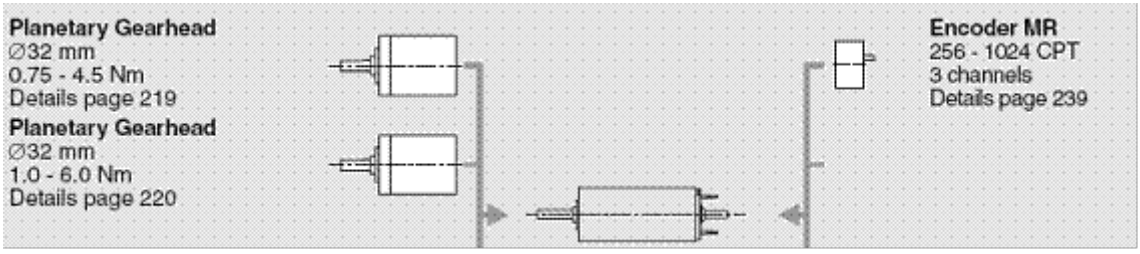


Figura 4.2: Composição motor+caixa de engrenagem+encoder

O codificador óptico seleccionado é o Encoder MR TypeL da Maxon®, tratando-se de um codificador óptico de 3 canais em modo diferencial com a resolução de 256 pulsos por volta. O canal A e B têm uma resolução de 256 pulsos por volta, encontrando-se estes dois canais em quadratura sendo o terceiro um canal de Índex que envia apenas um pulso por volta. É alimentado com uma tensão de 5V, sendo esta feita a partir do módulo de potência Servoamplifier ADS (4-Q-DC) da Maxon®, sendo as suas características principais apresentadas na próxima secção. O conector do codificador óptico tem a configuração apresentada na Figura 4.3:

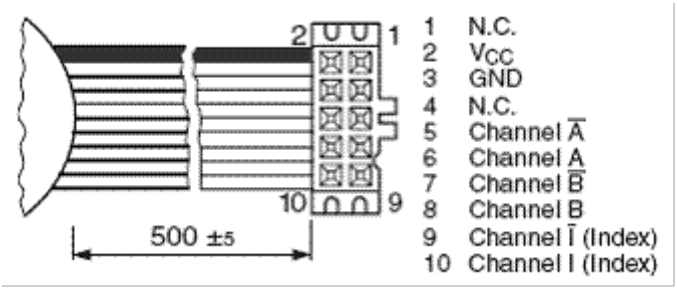


Figura 4.3: Configuração do conector do encoder

4.3. Módulo de Potência

Neste sub capítulo pretende-se essencialmente dar a conhecer o módulo de potência ("*Power Drive*") utilizado e as suas características mais importantes, sendo apresentados igualmente os motivos principais da utilização destes.

O módulo de potência utilizado é o indicado pelo fabricante do motor, à semelhança do que acontece com o codificador óptico utilizado. Desta forma utiliza-se o Servoamplifier ADS (4-Q-DC) da Maxon®, que dispõe já de 4 modos de controlo para o motor.

As características dos principais sinais de entrada e saída do módulo são apresentadas de seguida na Tabela 4.1, onde se pode encontrar informações relativas à alimentação do módulo bem como em relação aos sinais de saída.

Tabela 4.1: Características principais do módulo

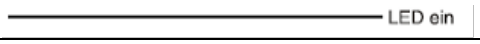
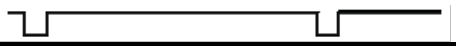
Sinal		Valor
Tensão de alimentação		12 a 50 V
Tensão máxima de Saída		0.9 * V_{CC}
Corrente máxima de Saída		10 A
Corrente continua de Saída		5 A
Frequência de comutação		50 kHz
Eficiência		95 %
Entradas	Set Value	-10 a 10 V (R_i=20kΩ)
	Enable	4 a 50 V (R_i=15kΩ)
	Sinais de Encoder	f_{Max}=100kHz (Níveis TTL)
Saídas	Monitorização de Corrente	-10 a 10 V (R_o=100Ω)
	Monitorização de Velocidade	-10 a 10 V (R_o=100Ω)
	Indicador de Estado "Ready"	Montagem em colector aberto

4.3.1. Estado do Módulo

Para facilitar o diagnóstico de eventuais problemas de funcionamento que possam ocorrer no módulo, este encontra-se munido de 2 led's que indicam o seu estado e a

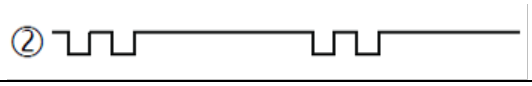
ocorrência de algum erro. O estado do módulo, bem como os erros que possam surgir no seu funcionamento podem ser então definidos de acordo com o estado dos led's descritos na Tabela 4.2 e Tabela 4.3.

Tabela 4.2: Estado do módulo de potência (led verde)

Led Verde	
Estado do Led	Estado do Módulo
	Módulo Activado
	Módulo Alimentado mas Inactivo

Como é perceptível, a cor verde do Led indica apenas se o módulo está activado ou desactivado, o que implica que só quando este está activado os sinais de tensão colocado nas entradas "+/-*Set value*" serão tidos em conta pelo módulo.

Tabela 4.3: Estado do módulo de potência (led vermelho)

Led Vermelho	
Estado do Led	Estado do Módulo
	Indicação de que a temperatura dos Transístores de Potência (Andar de Potência) foi excedida, 90ºC. Módulo desactivado.
	Indicação que a corrente máxima admitida pelo motor foi atingida, aprox.12.5A. Módulo desactivado.
	Indicação que a fonte de tensão interna não consegue atingir o valor de tensão pretendido. Módulo desactivado.
	Indicação que os sinais de entrada do codificador óptico excedeu a frequência máxima admissível, 150 KHz. Módulo desactivado.

Quando ocorrem os erros descritos anteriormente, o módulo passa ao estado de inactivo, sendo necessário reactivá-lo novamente, sempre que tal aconteça. Ou seja, não basta que o problema desapareça para ele passar ao modo activo. Se o problema persistir, quando a activação do módulo é iniciada, este coloca-se imediatamente no estado inactivo até que o problema seja resolvido.

As causas associadas ao primeiro tipo de erro são as mais variadas, podendo por exemplo dever-se a uma temperatura ambiente muito elevada, que pode ocorrer em aplicações especiais ou por existir uma má dissipação/ventilação do módulo. Pode ainda dever-se a um máximo de corrente contínua permitida atingida, situação essa que será aprofundada mais à frente quando forem abordados os potenciómetros de ajuste que o módulo possui. Essa corrente é normalmente de 5A.

4.3.2. Entradas e Saídas

Os sinais de entrada e saída deste módulo estão divididos em três tipos:

- ***Power***, onde se encontra a alimentação do módulo e as saídas de alimentação para o motor.

- ***Signal***, onde se encontram sinais de entrada de controlo, e sinais de saída indicando o estado de funcionamento do módulo.

    i. *+/- Set Value*, sinal de entrada em tensão (-10V a 10V) para controlo de sentido e velocidade do motor.

    ii. *Enable*, sinal de entrada em tensão, activa (com tensões entre 4V a 40V) a alimentação do Motor. Inactivo (com tensões de 0V a 2,5V) cortando a alimentação do motor.

    iii. *DC Tacho*, Sinal de tensão (2V a 50V), para ligação de Tacómetros, para controlo de velocidade.

    iv. *Monitor n*, Sinal de saída em tensão (-10V a 10V), equivalente à velocidade que o motor está a rodar.

    v. *Monitor I*, Sinal de saída em tensão (-10V a 10V), equivalente à corrente que o motor está a consumir.

    vi. *Ready*, Coloca a entrada ligada à massa quando o módulo se encontra a funcionar correctamente, coloca a saída em alta impedância quando o módulo não se encontra a funcionar correctamente, cujo esquema é apresentado na Figura 4.4.

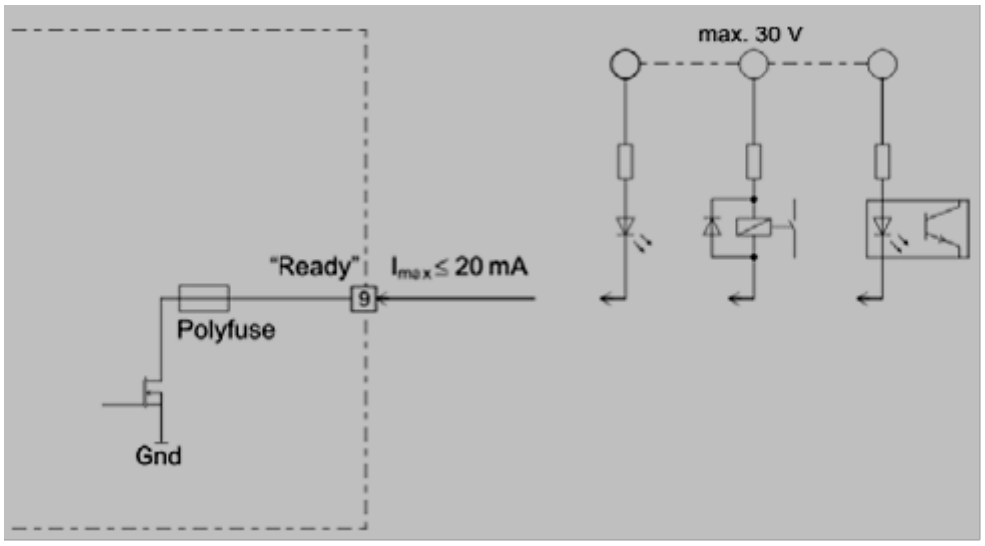
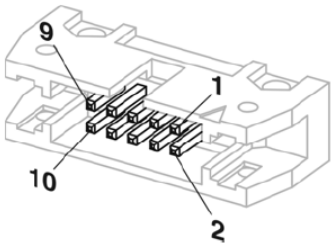


Figura 4.4: Configuração do circuito interno para o sinal "*ready*"

- **Encoder**, Sinais vindos do encoder de acordo com o esquema na Figura 4.5.



Pin configuration at "Encoder" input:

1	n.c.	Not connected
2	+5 V	+ 5 VDC max. 80 mA
3	Gnd	Ground
4	n.c.	Not connected
5	A\	Inverted Channel A
6	A	Channel A
7	B\	Inverted Channel B
8	B	Channel B
9	n.c.	Not connected
10	n.c.	Not connected

Figura 4.5: Ligações dos sinais de encoder

4.3.3. Modos de Controlo

De seguida apresentam-se os modos de controlo que se mostraram mais adequados para o controlo motriz do robô.

Os modos de controlo, disponíveis no módulo são, os seguintes:

1. Controlo de velocidade, a partir de sinais de Tacómetro.

2. Controlo de Velocidade, a partir de sinais de Encoder.

3. Controlo de velocidade por compensação de *IxR*

4. Controlo de Binário ou Corrente

A selecção do modo de controlo é feita com recurso a um conjunto de DIP-Switchs que se encontram num local de fácil acesso do exterior, sendo que as configurações de selecção são as representadas na Figura 4.6.

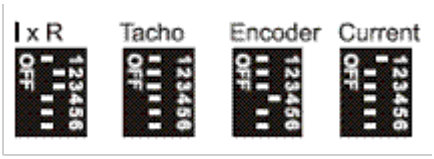
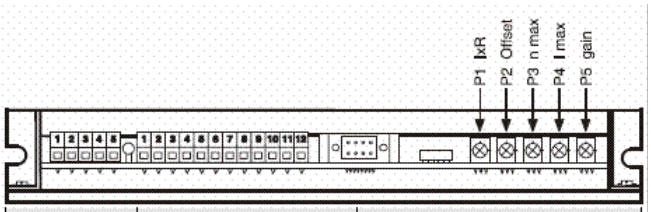


Figura 4.6: Configuração dos *DIP-Switchs* de selecção de modo

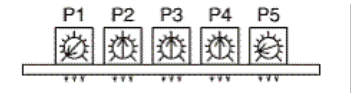
4.3.4. Potenciómetros de Ajuste de Funcionamento

Antes de abordar o modo de controlo usado há que dar algumas notas introdutórias acerca do funcionamento do módulo. Este módulo possui alguns potenciómetros que permitem fazer ajustes no seu funcionamento. Esses potenciómetros encontram-se no exterior do módulo de modo a ser fácil a sua manipulação, estando estes pré-ajustados (ver Figura 4.8) para o tipo de utilização mais comum do módulo. No quadro apresentado na Figura 4.7, são apresentadas as funções dos vários potenciómetros e qual o tipo de ajuste que estes permitem, bem como a posição dos potenciómetros tendo em conta a perspectiva lateral do módulo.



Potentiometer		Function	Turn to the left	Turn to the right
P1	IxR	IxR compensation	weak compensation	strong compensation
P2	Offset	Adjustment n = 0 / I = 0 at set value 0 V	motor turns CCW	motor turns CW
P3	n_{max}	max. speed at 10 V set value	speed slower	speed faster
P4	I_{max}	current limit	lower min. 0.5 A	higher max. 10 A
P5	gain	amplification	lower	higher

Figura 4.7: Potenciómetros de ajuste



Pre-adjustment of potentiometers		
P1	IxR	0 %
P2	Offset	50 %
P3	nmax	50 %
P4	Imax	50 %
P5	gain	10 %

Figura 4.8: Pré ajuste dos potenciómetros

Para além destes potenciómetros existe ainda a possibilidade de ajustes mais específicos, encontrando-se, no entanto, os potenciómetros respectivos no interior do módulo. A sua manipulação não é permitida sem que o módulo seja aberto.

Estes potenciómetros, P6, P7 e P8 cuja localização se apresenta na Figura 4.9, servem respectivamente para controlar o ganho em velocidade, o ganho em corrente e o valor de corrente de saída máximo em regime contínuo.

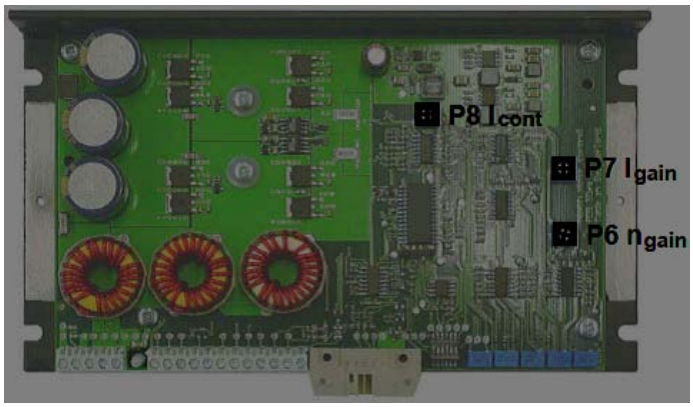


Figura 4.9: Localização dos potenciómetros

Os prés-ajustes dos potenciómetros P6 e P7, são respectivamente 25% e 40%. No que respeita ao potenciómetro P8 este serve para regular o valor de corrente em modo contínuo admissível, sendo que o valor de pico, regulado através do potenciómetro P4 será aceite durante um período de 0,1s em cada 1s. No tempo restante encontra-se limitado ao valor de corrente em modo contínuo regulado pelo potenciómetro P8, como se mostra na Figura 4.10. No entanto, este modo de controlo do valor máximo da corrente só é realizado quando o DIP switch 6 se encontra activo.

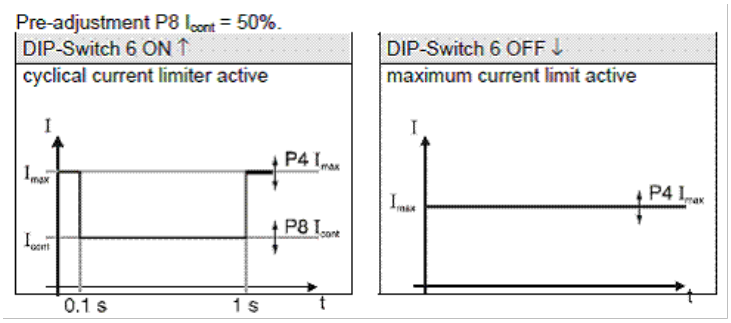


Figura 4.10: Acção do potenciómetro P8

4.3.5. Controlo em Modo de Encoder

O controlo em modo de encoder é realizado com base na informação que recebe do encoder acoplado ao motor. O valor que se pretende para a velocidade corresponde a uma dada tensão colocado na entrada "*set value*", que irá entrar no controlador de acordo com a informação que o próprio módulo de potência recebe do encoder, controlando o motor à velocidade desejada.

O controlo de velocidade é feito com base num controlador analógico que se encontra implementado em hardware, consistindo num amplificador inversor e numa montagem integradora, cuja dinâmica pode ser ajustada recorrendo à regulação dos potenciómetros de ajuste P5 e P6.

O esquema interno do módulo de potência pode ser consultado no catálogo do Servoamplifier ADS50/5 da Maxon® [10], estando de seguida na Figura 4.11 descrito de forma esquemática toda a cascata de montagens amplificadoras que implementam a malha responsável pelo controlo de velocidade do módulo.

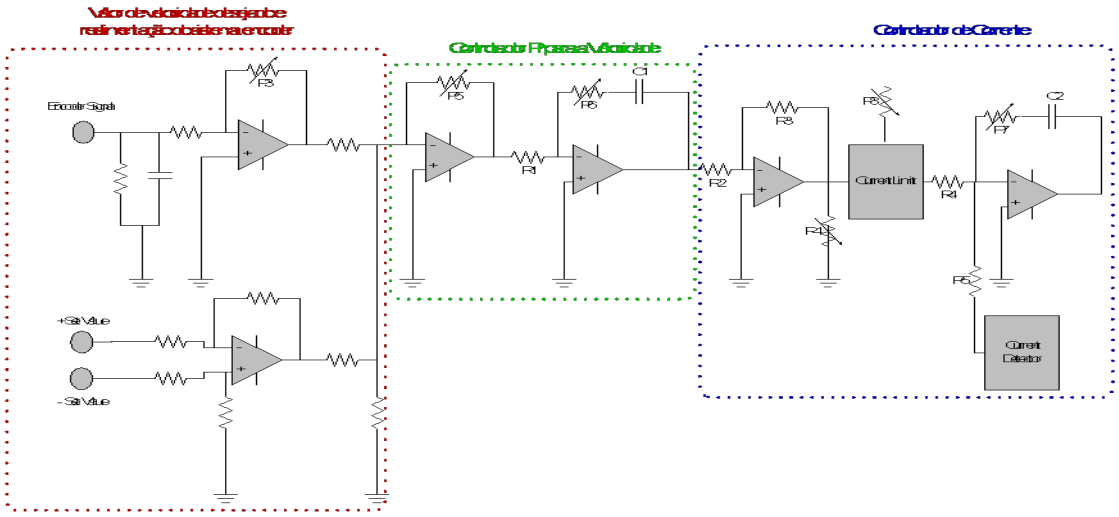


Figura 4.11: Esquema da malha de controlo em modo de encoder

A função de transferência da malha do controlador PI (Proporcional Integral) em velocidade é dada por $H_1(s)*H_2(s)$, com $H_1(s)$ a função de transferência de um amplificador inversor e $H_2(s)$ a função de transferência de um amplificador integrador como mostra a Figura 4.12.

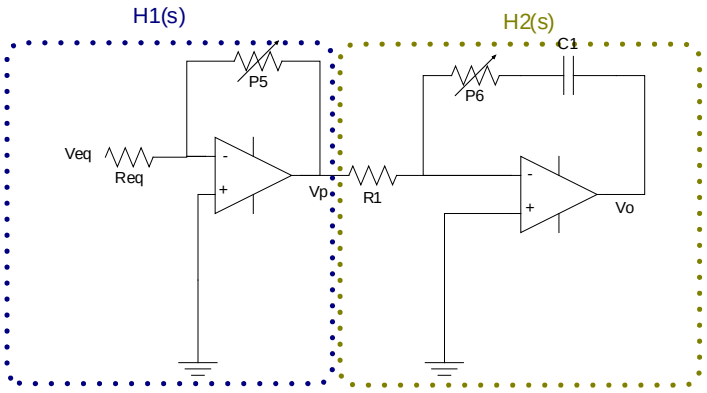


Figura 4.12: Controlador PI de velocidade do módulo

No que respeita ao amplificador inversor, considera-se que a impedância de entrada da montagem R_{eq} é elevada, ficando a sua função de transferência dada pela Equação 4.7:

$$H_1(s) = \frac{V_p}{V_{eq}} = -\frac{P_5}{R_{eq}} \tag{4.7}$$

O integrador tem a função de transferência dada pela Equação 4.8.

$$H_2(s) = \frac{P_6 + \frac{1}{sC_1}}{R_1} = -\frac{1}{R_1} \cdot \left(1 + \frac{1}{sP_6C_1}\right) \tag{4.8}$$

Variando o valor do potenciómetro P5 e o potenciómetro P6 pode-se então ajustar a sintonização do controlador PI, sendo a sua função de transferência dada por:

$$\begin{aligned} H(s) &= H_1(s) \cdot H_2(s) = \left(-\frac{P_5}{R_{eq}}\right) \cdot \left(-\frac{1}{R_1} \cdot \left(1 + \frac{1}{sP_6C_1}\right)\right) \\ \Leftrightarrow H(s) &= \overbrace{\frac{P_5}{R_{eq}R_1}}^{K_{si}} \cdot \left(1 + \frac{1}{s\underbrace{P_6C_1}_{\tau_{si}}}\right) \end{aligned} \tag{4.9}$$

O ajuste do potenciómetro P5 controla então a acção proporcional ao variar o ganho K_{si}, e o ajuste do potenciómetro P6 controla a acção integral ao variar o valor da constante de tempo τ_{si}.

4.4. Baterias

A autonomia de um robô móvel autónomo está intimamente ligada às características das baterias que suportam todo o sistema e este é, obviamente, concebido tendo em vista a minimização do consumo de energia.

Tabela 4.4: Consumos do sistema

Equipamento/Modulo	Corrente[(1)] (A)	Potência Consumida[(1)] (W)
PC	2	48
Conjunto (Motor + Power Drive)[(2)]	1,5	36
Restantes módulos de hardware	0,5	12

(1)Valores Aproximados; (2) Para os dois conjuntos

No que respeita ao consumo do sistema, são apresentados valores aproximados na Tabela 4.4 para os vários constituintes, conseguindo dessa forma chegar a um valor próximo do consumo do sistema, que se estima em 4 A/h, garantindo as duas baterias aproximadamente 2 horas de autonomia. As baterias para aplicações em robótica móvel autónoma têm que preencher alguns requisitos no que respeita ao seu tamanho e peso. Tal facto contribui por optar por baterias secas (mais leves e compactas).

Pretendendo-se um sistema distribuído simetricamente, então a opção mais certa é utilizar duas baterias, e dadas, as características dos vários módulos de hardware, a tensão nominal mais apropriada seria de 12V. Ligando estas em série obtém-se então a tensão nominal dos motores 24V.

São utilizadas as baterias da SAFT. As suas principais características apresentam-se resumidamente na Tabela 4.5:

Tabela 4.5: Características principais da bateria

Caracteristica	Valor/Especificação
Designação	VH Module D
Tensão nominal	12V
Capacidade mínima	8 Ah
Capacidade tipica	8.4 Ah

Corrente de Carga	Main	3 A
	Balancing	180 mA
	Trickle	90 mA
Duração de Carga	100%*	6 Horas
	90%	3 Horas

* - É necessário efectuar uma carga de 100% a cada 20 ciclos de carga/descarga ou de 3 em 3 meses.

O aspecto das baterias bem como uma ideia das dimensões desta podem ser observadas na Figura 4.13.

Figura 4.13: Aspecto da bateria

4.5. Módulo de Hardware *"Motion_Interface"*

Dadas as necessidades que se apresentavam ao tratamento dos sinais, o módulo *Motion_Interface* foi desenvolvido de modo a servir de interface entre o *driver* de potência (Servoamplifier ADS) e o módulo "*PIC_Base*", que faz o interface com o PIC responsável pelo controlo de velocidade de cada um dos motores.

As funções principais deste módulo são então:

1. Tratar o sinal de comando enviado pelo PIC segundo o protocolo SPI, de modo a transformar posteriormente num sinal analógico de -10V a 10V com recurso a um DAC, para colocar na entrada de comando do "drive de potência". É ainda enviado para o PIC um sinal analógico de 0V a 5V, resultado da conversão D/A, que serve de auto-regulação do sistema de controlo.

2. Tratar os sinais enviados pelo encoder (3 canais em modo diferencial, Canal A, Canal B e Canal Index), de modo a enviar para o PIC dois sinais distintos:

   2.1. Count-Up e Count-Down, a partir do Canal A e Canal B que se encontram em quadratura.

   2.2. Sinal de contagem de volta a partir do Canal Index.

3. Tratar os sinais de informação de corrente e velocidade disponíveis no drive de potência, que se trata de um sinal analógico que varia entre -10V e 10V, de modo a enviar para o PIC um sinal analógico de 0 a 5V.

4. Tratar o sinal de Status, disponível no drive de potência de potência de modo a enviar para o módulo PIC_Base.

5. Tratar o sinal de Enable a enviar da PIC_Base para o drive de potência para colocar o drive ON/OFF, remotamente.

4.5.1. Tratamento do Sinal de Comando

O sinal de comando (velocidade pretendida) é enviado pelo PC via CAN. Depois de recebido e tratado é enviado usando o protocolo SPI. Existe então a necessidade da utilização de um conversor Digital/Analógico (DAC) para a conversão desse sinal digital num sinal analógico. Utilizou-se o DAC MCP4921 (*12-bit voltage output digital-to-analog converter*). Este coloca na saída um sinal analógico VDAC, de 0V a 5V.

Este sinal tem ainda de ser tratado antes de ser enviado para o módulo de potência, uma vez que o sinal de comando a enviar tem de ser um sinal analógico de -10V a 10V. Com o objectivo de ter essa gama de valores com precisão e de igual forma em ambos os módulos que controlam cada motor, recorreu-se a uma montagem amplificadora com ganho variável para ajuste fino desse ganho.

Esse auto-ajuste do ganho do amplificador é realizado recorrendo a potenciómetros digitais, representados na Figura 4.14. P_1 e P_3 são dotados de comunicação SPI, e os seus valores são controlados pelos PIC's associados a cada um dos módulos, em função da comparação do sinal de comando à saída dos módulos com um dado sinal de referência.
A montagem escolhida é o amplificador de diferença porque apresenta uma equação simples num caso particular. Esta é apresentada na Figura 4.14, e o seu ganho é dado pelas eq. 4.13 e 4.14. A primeira é a equação genérica e a segunda um caso particular, aplicado quando se verifica $P_1=P_3$ e $R_2=R_4$.

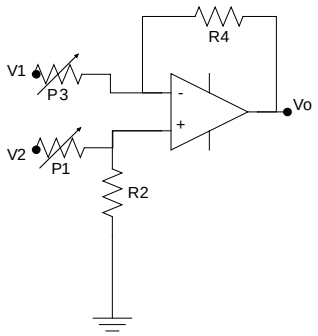


Figura 4.14: Amplificador de diferença

$$v_0 = \frac{\left(1+\frac{R_4}{P_3}\right)}{\left(1+\frac{P_1}{R_2}\right)} \cdot v_1 - \frac{R_4}{P_3} \cdot v_2 \tag{4.13}$$

$$v_0 = \frac{R_4}{P_3} \cdot (v_1 - v_2) \tag{4.14}$$

O circuito usado em simulação é apresentado na Figura 4.15 e o seu resultado pode ser observado na Figura 4.16, em que se verifica que a gama de valores pretendida é obtida. Para valores de saída do DAC entre 0V e 5V corresponde a toda a gama de entrada de referência do *PowerDrive* que é de -10V a 10V, como se verifica pela Figura 4.16 os sinais a cor verde e vermelho respectivamente.

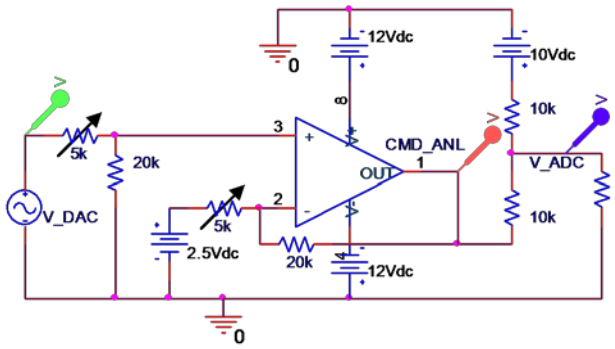


Figura 4.15: Montagem amplificadora de diferença para simulação

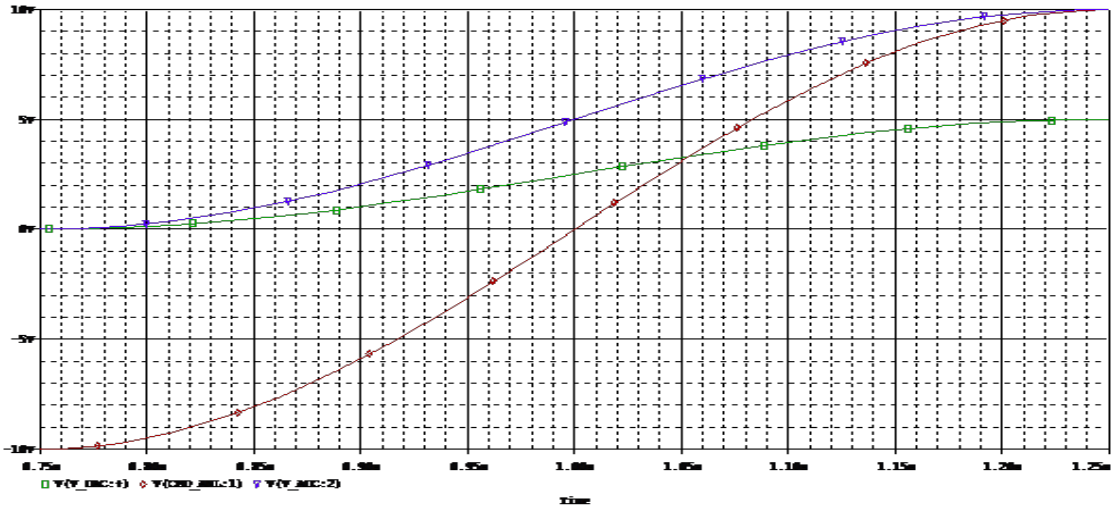


Figura 4.16: Resultado da simulação

4.5.2. Tratamento dos Sinais de Encoder

Como já foi referido o encoder possui 3 canais, Canal A, Canal B e Canal Índex. Cada um desses canais disponibiliza o sinal em modo diferencial, pelo que, existe a necessidade de transformar esse sinal diferencial em TTL. Utiliza-se para o efeito o integrado SN75179B (*differential driver and receiver pair*), como se observa na Figura 4.17.

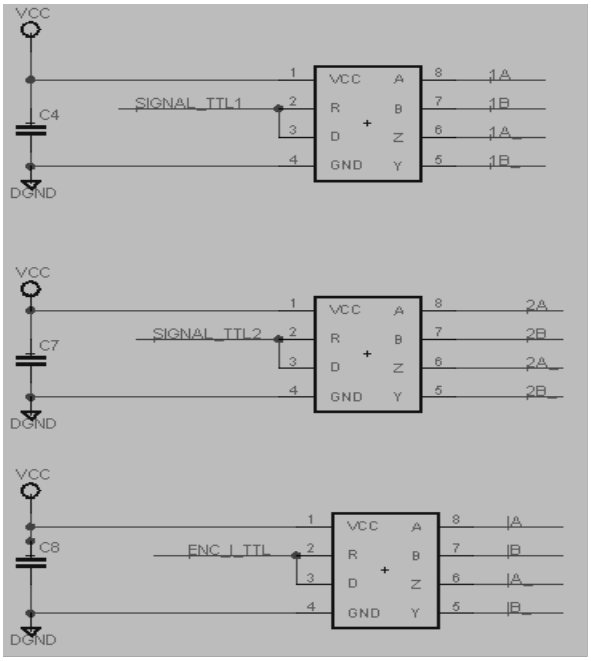


Figura 4.17: Conversão Diferencial/TTL dos sinais vindo do encoder

O sinal TTL obtido a partir do canal Índex é um sinal que é enviado para o PIC. No que respeita aos sinais TTL obtidos dos Canais A e B, estes encontram-se em quadratura, existindo a necessidade de transformar esses sinais em dois sinais distintos de *Count-UP* para enviar para o PIC. Para tal utiliza-se a montagem presente na Figura 4.18,

recorrendo à utilização de dois Flip-Flops do tipo D e duas portas NOT, cujo resultado em simulação pode ser observado na Figura 4.19 e Figura 4.20.

O objectivo é identificar se é o canal A ou o canal B que está em avanço em relação ao outro, obtendo-se portanto os sinais de *Count-UP* e *Count-Down*, respectivamente.

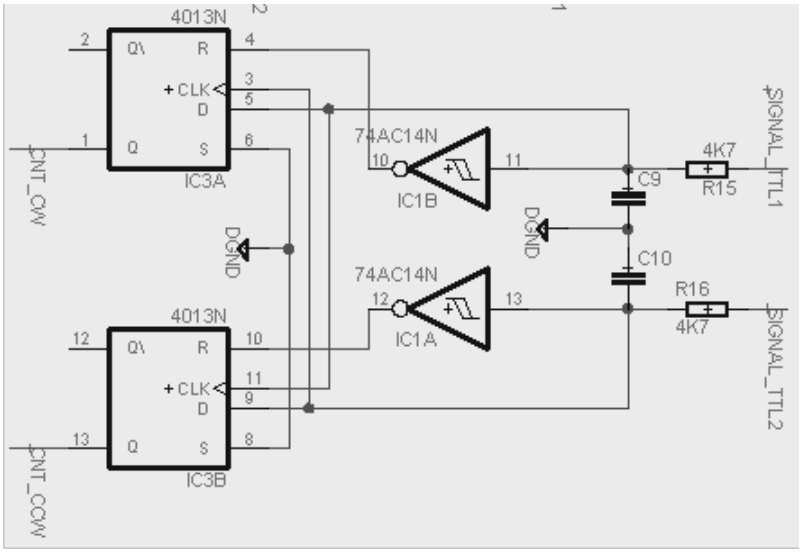


Figura 4.18: Detecção do sentido de rotação

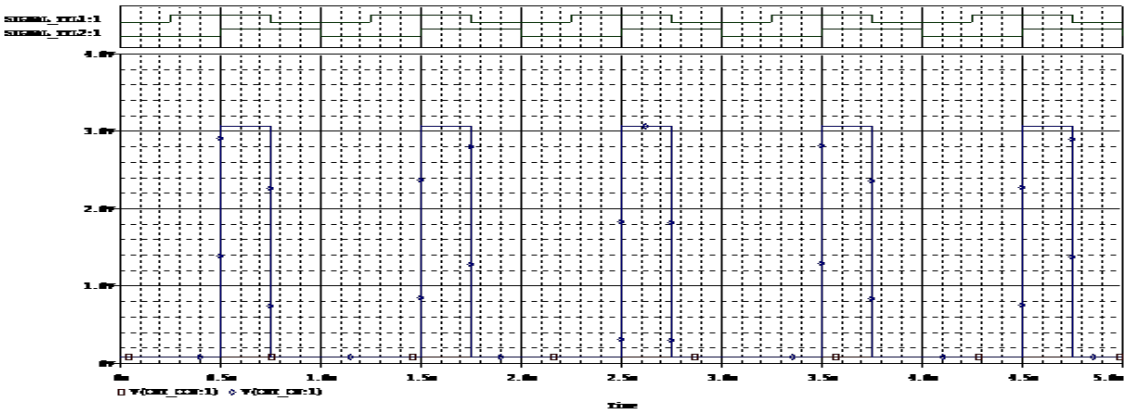


Figura 4.19: Detecção de *Count-Up*

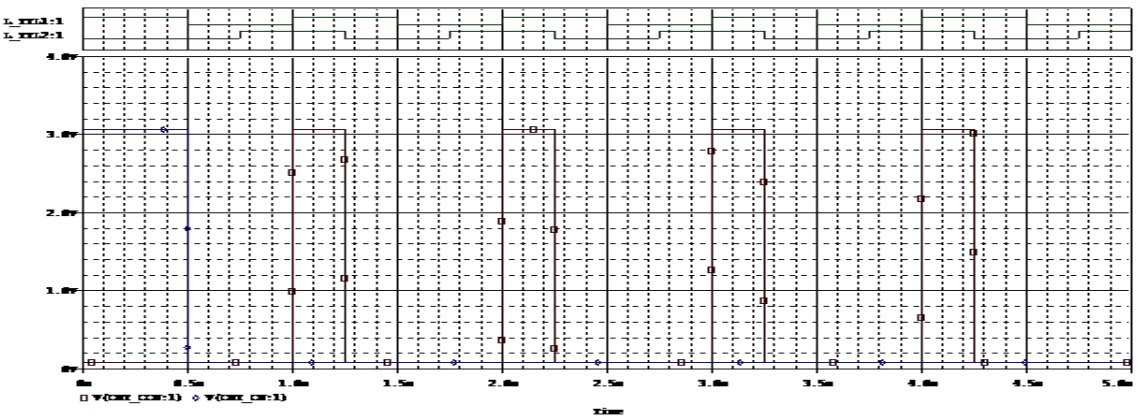


Figura 4.20: Detecção de *Count-Down*

4.6. Módulo de Hardware *"Infra_Red"*

Este módulo de hardware foi desenvolvido com o intuito de fazer de interface entre o PIC responsável pela monitorização do estado dos sensores IV. Para os sensores IV existem buffers que garantem a estabilidade dos sinais à entrada do PIC, servindo também de protecção. Este módulo encontra-se preparado para poder ligar até um máximo de dez sensores de infravermelhos.

4.6.1. Características dos IR's

Este tipo de sensores consiste num díodo emissor de infravermelhos e de um transístor fotossensível ("*Phototransistor*"). No caso do IR utilizado, OPB704 encontram-se colocados lado a lado no encapsulamento, como mostra na Figura 4.21.

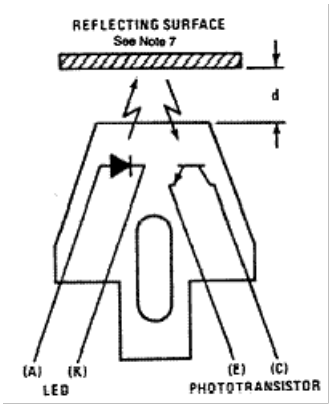


Figura 4.21: Infravermelho OPB704

A distância *d* representada é de cerca de 1,5 centímetros. Esta distância está relacionada com a superfície utilizada na pista presente no concurso, embora este valor varie em função da superfície de reflexão e da luminosidade ambiente.

4.7. Módulo de Hardware *"Ultra_Sound"*

Este módulo de hardware tem como objectivo fazer a interface dos sinais que são enviados e recebidos entre o PIC responsável por monitorizar os ultra-sons e os próprios, não possuindo quaisquer outras tarefas adicionais, tornando-se como tal um circuito bastante simples. Este módulo encontra-se preparado para poder ligar até um máximo de oito ultra-sons.

A escolha dos ultra-sons a utilizar no projecto recaiu sobre o modelo SRF-04 cujas características se adaptam bem às necessidades do projecto. As características principais do SRF-04, encontram-se descritas na Tabela 4.6.

Tabela 4.6: Características dos ultra-sons

Sinal	Unidades S.I.	Valor/Condição de funcionamento
Tensão	V	5
Corrente	mA	30 Tip. (50 Max.)
Frequência	KHz	40
Distância Max.	m	3
Distância Min.	cm	3
Disparo (Trigger)	µS	10 (Min. nível TTL)
Pulso de eco		- Sinal TTL positivo, proporcional à distância ao obstáculo
Dimensões		- 43mm x 20mm x 17mm

Tabela 4.6: Características dos Ultra-sons

Este sonar responde a um impulso de nível lógico 1 com a duração de 10 µS, que serve de sinal de disparo. Quando este sinal é recebido pelo sonar, este envia então um sinal acústico ultra-sónico e coloca a linha de eco em modo de escuta durante um período entre 100 µs a 18 ms aproximadamente. Só após 10 µs do último eco recebido se pode enviar um novo pedido de leitura. O que se referiu anteriormente constata-se no diagrama temporal representado na Figura 4.22.

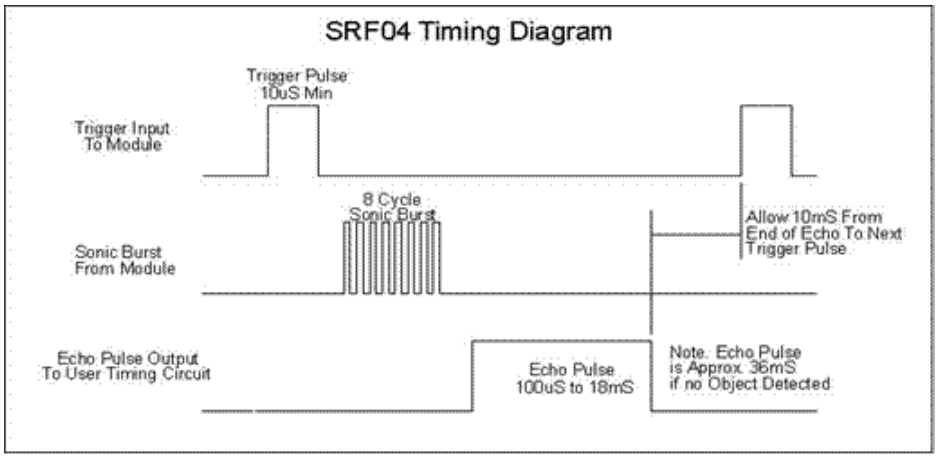


Figura 4.22: Diagrama temporal do funcionamento do SRF-04

A utilização deste tipo de sonar deve-se também à sua sensibilidade, sendo esse um factor importante para a detecção de obstáculos com áreas reduzidas, como acontece com a fita indicadora de zona de obras que tem apenas alguns centímetros de largura. No entanto, a dispersão do sinal acústico não é uniforme, embora seja especificado que o SRF-04 utiliza um ângulo de dispersão de 90º, a sua precisão melhora consideravelmente à medida que esse ângulo é reduzido, como se pode observar no diagrama de potência de radiação de dispersão acústica do SRF-04, na Figura 4.23.

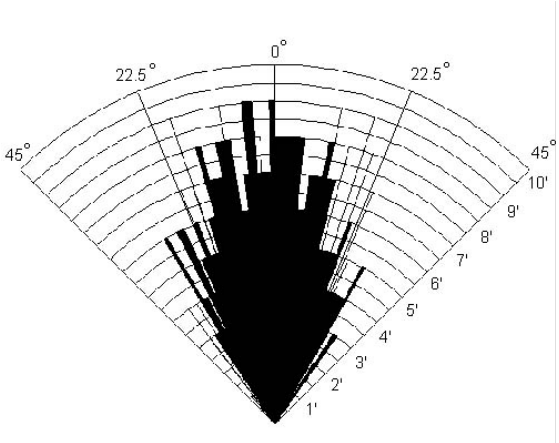


Figura 4.23: Dispersão acústica (Diagrama de potência de radiação) do SRF-04

Graças ao apresentado é comum adoptar-se a solução de usar vários sensores com diferentes orientações de forma a cobrir ângulos de dispersão na ordem de 60º, conseguindo desta forma medições mais precisas.

4.8. ***Módulo de Hardware*** *"Power_Convertion"*

Este módulo de hardware ao contrário dos outros módulos não é activo, não possui ligação com nenhum módulo "*PIC_Base*", nem com o PC. Serve apenas para centralizar e proteger todas as entradas e saídas de alimentação e gerir a fonte de energia a usar de forma electromecânica.

4.8.1. Esquema de Alimentações do Sistema

Tendo em vista a minimização da cablagem, a alimentação de todos os módulos de hardware do sistema distribuído é feita a 12V.

Dado o sistema de alimentação ser composto por duas baterias ligadas em série que alimenta os módulos de potência e respectivos motores, é no módulo "*Power_Convertion*" que se obtém os 12V. Neste módulo são obtidos os 12V

recorrendo a um conversor DC-DC, TRACO® POWER TEM 25-2412, que apresenta a capacidade de ser desligado externamente.

Dado que grande parte dos módulos necessita também de alimentação a 5V estes tem incorporado um mini conversor que satisfaz essa necessidade.

No que respeita a tensões de referência (precisão) necessárias para as montagens amplificadorasutiliza-se IC's dedicados para o efeito que garantem a não flutuação desses sinais de referência em função da carga imposta à alimentação do módulo de hardware.

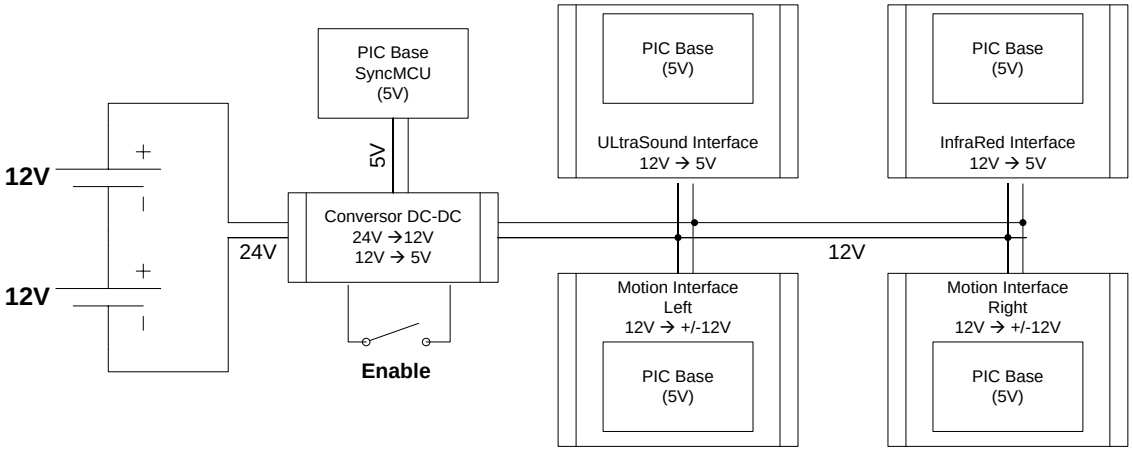


Figura 4.24: Esquematização das alimentações dos vários módulos

4.9. Estrutura de Suporte do Robô

A estrutura do Robô é constituída por uma plataforma robusta toda ela em alumínio trabalhado mecanicamente, possui duas rodas motrizes frontais de alumínio com um oringue de borracha, e uma roda livre de apoio na parte de trás. A Figura 4.25 apresenta o robô de perfil onde se verificam essas características.

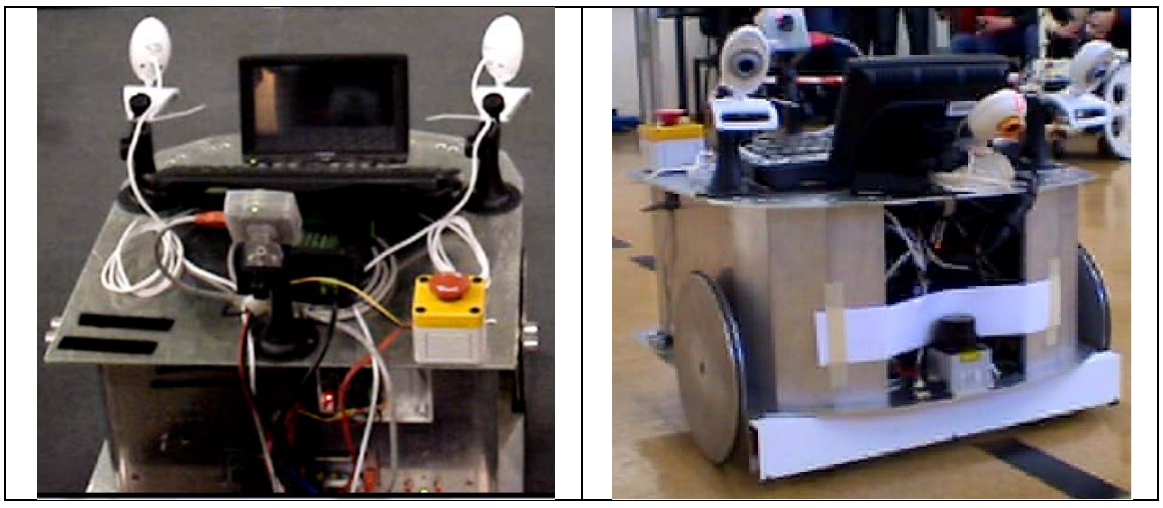


Figura 4.25: Perfil do robô

Esta plataforma foi concebida de forma a poder integrar, no seu interior, todos os componentes necessários à sua autonomia. De seguida, apresentam-se as dimensões reais da plataforma e a localização dos vários componentes que compõem a mesma.

A localização dos vários componentes como, baterias, motores, módulos de potência, placa do PC e os outros módulos de hardware, está representada na Figura 4.26. Sendo que a projecção do robô ocupa sensivelmente 0,5m por 0,5m.

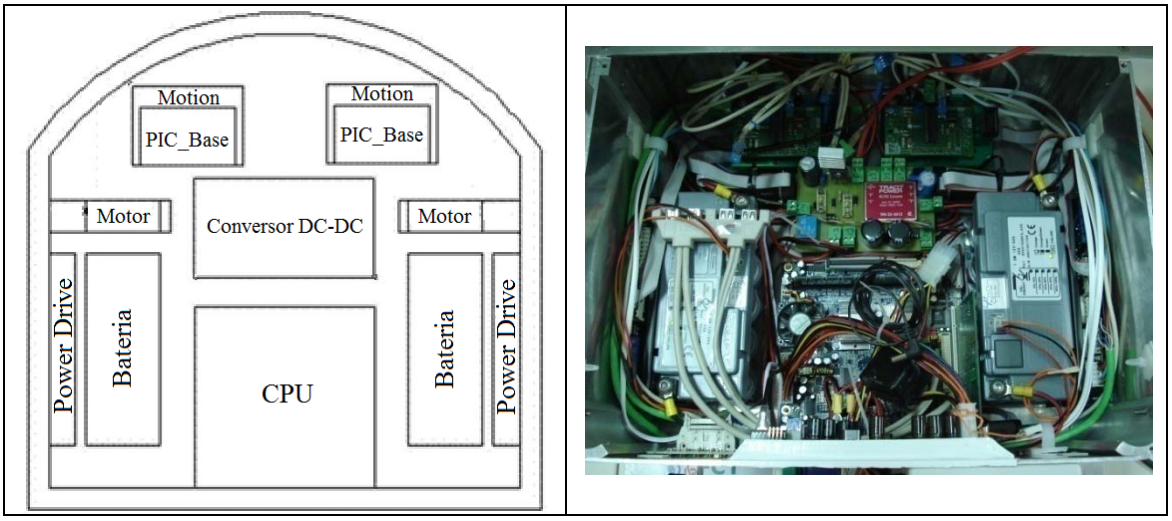


Figura 4.26: Localização do hardware dentro da estrutura

Um dos aspectos importantes, para além das dimensões do robô, é o seu peso, e a distribuição da carga (centro de massa). Esta última já se encontra com uma distribuição optimizada dado a simetria de ambos os lados do robô, sendo que o peso total do robô é aproximado em cerca de 20 kg.

4.10. Sistemas Periféricos do Robô

Como é perceptível, toda a informação circula entre os vários PIC's e o PC, que é o "cérebro" do sistema, através de um barramento CAN. Os PIC's comportam-se como escravos (*Slaves*), cumprindo as ordens do PC, que vai ser o mestre (*Master*).

Os PIC's associados aos sensores de IV e Ultra-Sons, têm como tarefa enviar o estado dos sensores periodicamente, para tomada de decisões por parte do PC. Temos portanto ligado à rede distribuída de CAN alguns sensores, que em termos de quantidade de dados é suportável para este tipo de rede.

Outros tipos de sensores tais como o Laser e a Visão são sistemas periféricos que são ligados directamente ao PC, visto usarem outro tipo de comunicação que suporta uma taxa de transferência de dados maior, que é o USB e FireWire respectivamente.

Para além destes sensores periféricos, existem alguns mecanismos de segurança inerentes ao robô tais como: um botão de emergência local para parar o sistema ou por rádio frequência (telecomando) que se revelou indispensável em fase de testes.

Associado ao PC temos os periféricos normais para interacção com o mesmo, que é um mini-tft com *touch-screen* e teclado.

Capítulo 5

5. Algoritmos Incorporados no ISRobot para a Prova de Condução Autónoma

Vão ser apresentados neste capítulo, os vários componentes e funcionalidades que foram adicionados à plataforma ISRobot com vista a dotá-la das capacidades necessárias para a prova de condução autónoma. Tal como foi apresentado no Capítulo 2 esta prova consiste em vários desafios, necessitando portanto de um sistema computacional para gerir todas as tarefas.

A plataforma robotizada possui uma arquitectura distribuída de actuação/aquisição sensorial tal como é descrito nos Capítulos 3 e 4, mas isto por si só não basta, daí que tenha sido instalado um sistema operativo de tempo real (RTAI) para gerir este sistema e tenham sido integrados outros sensores (laser e visão) com capacidades superiores aos já existentes (ultra-sons e infravermelhos). É verdade que é possível seguir uma linha com infravermelhos e detectar obstáculos com ultra-sons. No entanto usando o laser e a visão para estas tarefas é possível a implementação de acções mais flexíveis visto conseguirmos obter mais informação e maior detalhe do meio envolvente.

5.1. Sistema de Tempo Real - RTAI

Com este sistema de tempo real, o programa pode ter várias tarefas (*threads*) com periodicidade e prioridade definidas de acordo com a importância da tarefa.

Neste caso em concreto existe a tarefa principal que faz a gestão dos recursos a serem usados (AutoDrive). É esta que, de acordo com as necessidades cria outras para gerir determinados recursos em concreto. Estes recursos são o CAN, o Laser e a Visão, existindo as tarefas *CAN_Module, Laser_Module* e *Vision_Module* respectivamente. Estas tarefas têm como função gerir os respectivos recursos (dispositivos) e fazer o

tratamento dos dados provenientes dos mesmos. Como é desejado que exista um processamento paralelo entre a aquisição e processamento dos dados, ou seja, que se possa fazer uma nova aquisição de dados antes de ter terminado a análise da amostra anterior, temos outras tarefas dedicadas à aquisição dos dados.

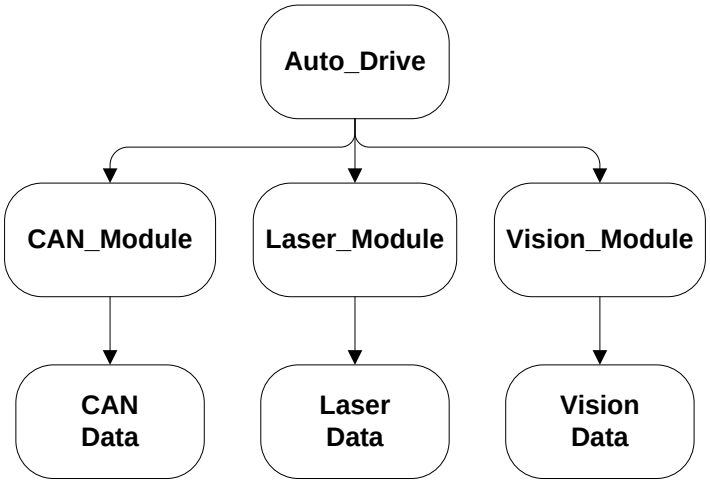


Figura 5.1: Sub-tarefas do programa principal de RTAI

5.2. Navegação com recurso a Laser e Visão

Para termos um robô/veículo autónomo aplicado à condução autónoma, este tem de interagir com o meio, sendo esse mundo complexo é usual usar-se laser e visão à semelhança do DARPA Grand Challenge.

Neste caso o laser foi usado apenas como dispositivo de segurança não tendo sido feito a segmentação e classificação de qualquer ordem. Funcionando como laser de segurança quer dizer que é definida uma zona próxima do mesmo onde caso seja detectado algo é despoletada uma acção de segurança. O alcance do mesmo é de cerca de 4m, tendo a zona de segurança sido definida nos 0.3m para imobilização do robô.

O laser usado (Hokuyo) faz varrimentos de 240º com uma resolução de 0.3516º. Esta amplitude foi subdividida em cinco zonas distintas (30º+50º+80º+50º+30º), tendo sido dado importância apenas às três zonas centrais (180º), e que permite identificar se o obstáculo se encontra na frente ou num dos lados do robô. Desta forma é possível, no caso particular da prova, decidir se o robô pode prosseguir na faixa de rodagem actual ou se tem de mudar para a outra.

Neste caso particular o objectivo do uso da visão é identificar as linhas delimitadoras das duas faixas de rodagem e respectiva linha central tracejada. Com este objectivo foram usadas duas câmaras (uma de cada lado da estrutura frontal do robô), com vista a detectar a linha mais próxima do robô, sendo que nesses caso o varrimento da imagem da câmara direita faz-se da esquerda para a direita e a outra faz no sentido contrário. A ideia com esta solução era estando o robô na faixa direita orientar-se pela linha direita continua, e caso passasse para a faixa esquerda orientar-se pela linha esquerda contínua. Devido à limitação de processamento do PC a aquisição de imagens é feita apenas numa câmara. Infelizmente esta solução não foi devidamente explorada, tendo-se optado por usar apenas uma câmara central a seguir a linha tracejado como alternativa mais simples. Neste caso o varrimento da imagem faz-se da zona central da imagem para as extremidades e segue-se a linha tracejado, tendo sido atingido uma solução minimamente aceitável tendo em conta as limitações existentes neste vasto leque de soluções passíveis de serem implementadas futuramente.

Capítulo 6

6. Resultados

Começando pela plataforma RobChair, uma vez que foi nesta plataforma que se procedeu inicialmente à implementação do DES, foram realizados alguns testes com vista à comprovação da funcionalidade do sistema distribuído desenvolvido para controlo da mesma. Inicialmente esses testes foram realizados recorrendo ao controlo da RobChair através do joystick (controlo manual), e posteriormente noutro trabalho desenvolvido por Razvan Solea, no controlo de seguimento de trajectória (Solea, et al., ICINCO'08). Em ambos os casos serviu para aperfeiçoar o sistema distribuído nomeadamente impondo determinismo nas acções realizadas pelos módulos, uma vez que se verificou que com o joystick (controlo malha aberta) não era crítico este determinismo mas no caso de controlo autónomo (controlo malha fechada) este pormenor é indispensável, sendo o determinismo obtido com base no sistema de tempo real.

Para comprovar a portabilidade deste sistema e para termos uma plataforma para a prova de Condução Autónoma como já foi referido na Introdução, foi construída a plataforma ISRobot. Esta plataforma revelou ter as características necessárias à sua função, após algumas reestruturações do sistema distribuído. À medida que se foram adicionando mais sensores e periféricos ao sistema central, verificou-se a necessidade de aumentar a capacidade de potência da fonte de alimentação do PC.

Com vista a um consumo não exagerado (maior autonomia) do sistema este foi dotado dum PC mini-ITX de baixo consumo. Obviamente que sendo de baixo consumo não se pode exigir dele capacidades de alto desempenho. No entanto apresenta capacidades suficientes para gerir o sistema distribuído (barramento de CAN) em tempo real, aquisição de dados do laser e a captura de imagem com baixa resolução. Verificou-se no entanto que caso sejam necessários algoritmos mais exigentes do ponto de vista computacional, tais como: segmentação de múltiplas imagens com resolução típica das câmaras actuais, a melhor solução será colocar um PC portátil para tratar do processamento da visão e comunicar o resultado desse processamento ao outro PC

embebido que está a correr obedecendo a requisitos de tempo real com recurso ao sistema de RTAI.

As expectativas globais traçadas – participação de forma aceitável na prova de Condução Autónoma com algoritmos de laser e visão complexos – não foram atingidas. Isto porque em certa parte tais expectativas foram colocadas num patamar demasiado elevado, tendo em conta o tempo e recursos humanos disponíveis assim como as áreas díspares que estão envolvidas. Apesar disso a plataforma foi construída, foi dotada de todo o seu sistema (actuação e sensores) e sobre a plataforma de RTAI foi feita a aquisição de laser e visão para além de todos os outros sensores que partilham o barramento de CAN. Com os dados de laser foi definida apenas uma zona se segurança para imobilização do robô em caso de choque eminente. Os dados provenientes do sistema de visão - com a finalidade de orientar o robô sobre a linha tracejada central da pista - foram filtrados e normalizados em binário usando um algoritmo simples.

Capítulo 7

7. Conclusões e Trabalho Futuro

Este projecto abrange áreas de conhecimento bastante alargadas, podendo ser sempre alvo de melhoramentos e incorporação de novas funcionalidades a qualquer momento. O objectivo final do projecto deverá estar sempre para além daquilo que já está feito, ou seja, deverá estar sempre em evolução.

No entanto o objectivo primário definido para este ano – construção de uma plataforma modular distribuída para prova de condução autónoma – foi atingido. A plataforma está apta para no futuro – continuação do projecto – ser dotada de algoritmos "inteligentes" que ira interagir com a arquitectura distribuída de interface com todo o hardware. Foi inclusive implementado um algoritmo simples de seguimento de linha baseado em visão e a definição de uma zona de segurança para imobilização com base em laser na presença de obstáculos, que comprova a operacionalidade do sistema desenvolvido.

Como trabalho futuro e neste caso com vista à prova de condução autónoma, é necessário desenvolver algoritmos de visão capazes de seguir não apenas uma linha mas sim um percurso delimitado por duas ou mais, assim como com base no laser ter a capacidade de desvio dos obstáculos.

Para terminar, deixo a ideia com que comecei esta tese. Todos os algoritmos desenvolvidos para robôs móveis têm a necessidade de ser devidamente testados em ambientes reais, necessitando para tal de uma plataforma física.

Anexo A

A. Módulos de Hardware

Para cada um dos vários módulos de hardware desenvolvidos para a arquitectura distribuída da plataforma ISRobot, é apresentado neste anexo o esquemático e respectiva placa a que deu origem.

A.1. “Motion Interface”

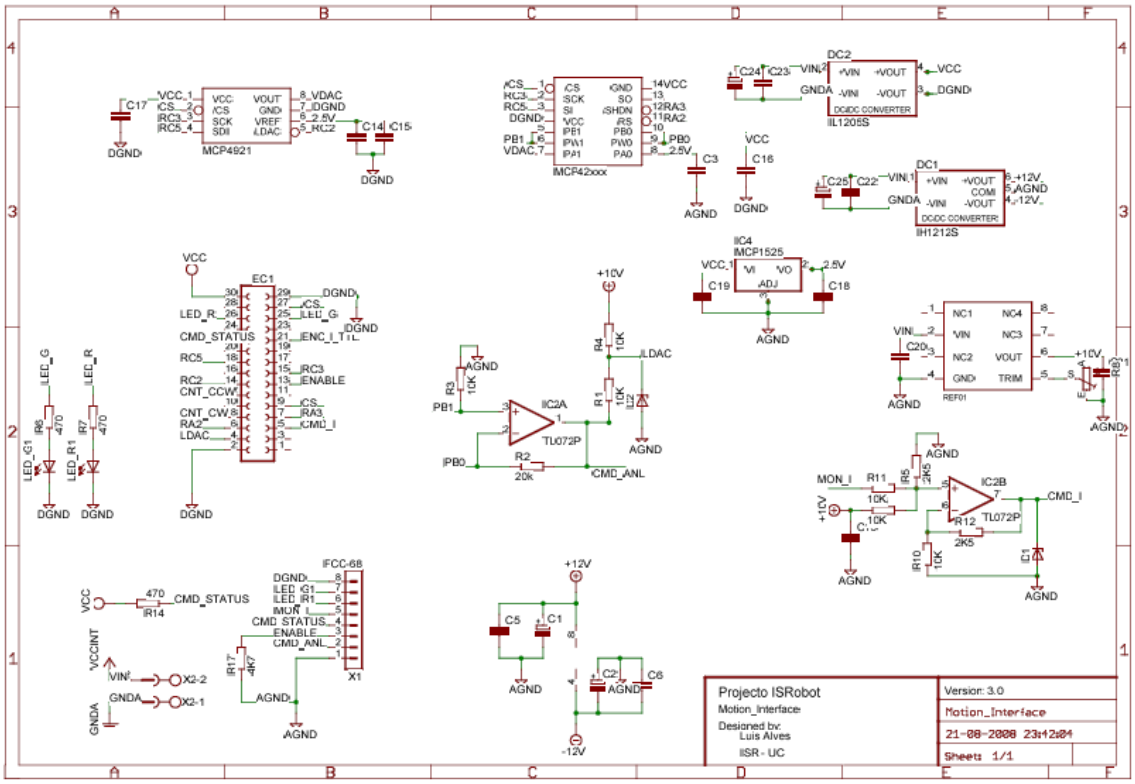


Figura A.1: Esquemático do módulo *Motion Interface (1/2)*

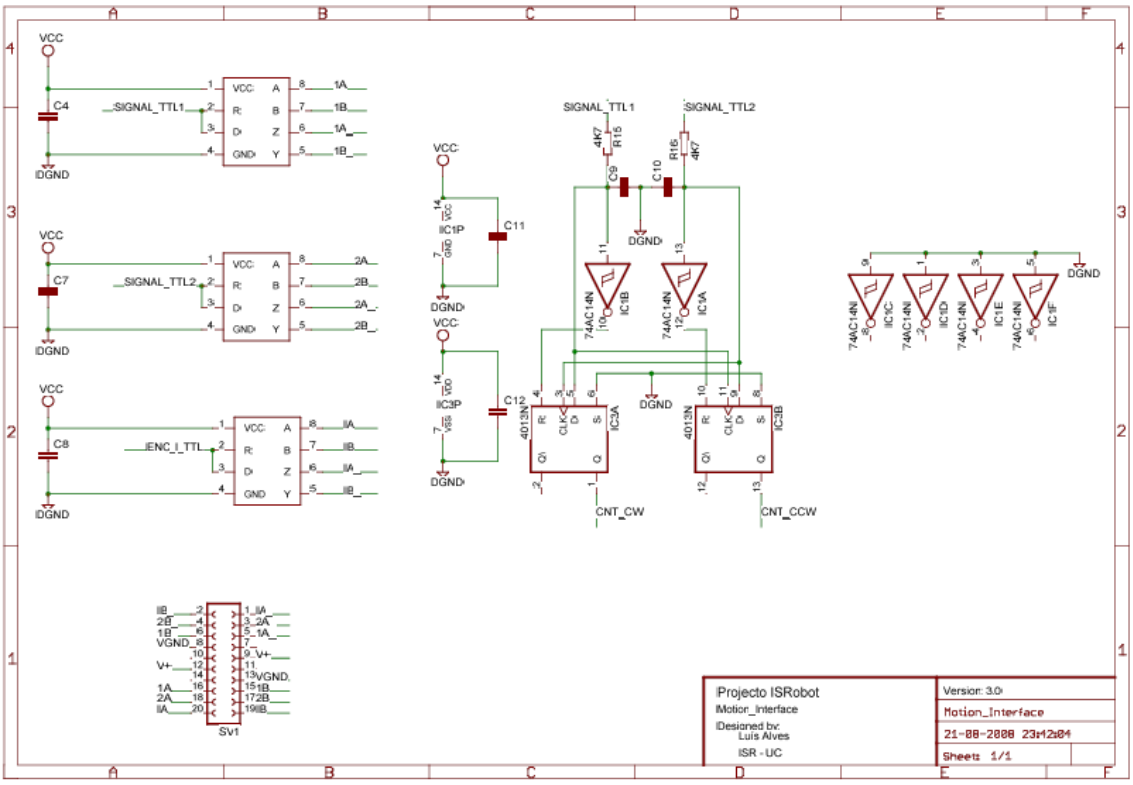


Figura A.2: Esquemático do módulo *Motion Interface (2/2)*

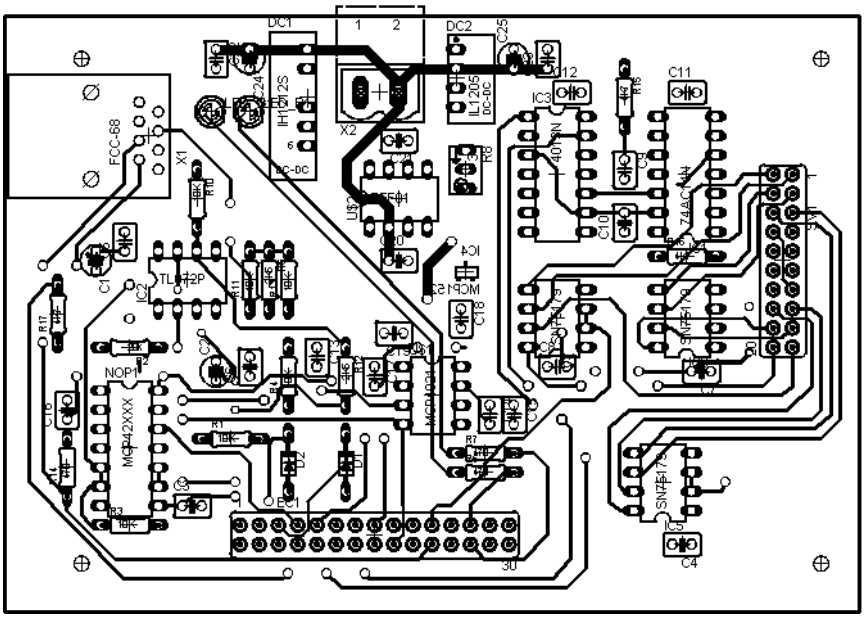


Figura A.3: Layout do PCB *Motion Interface* (top layer)

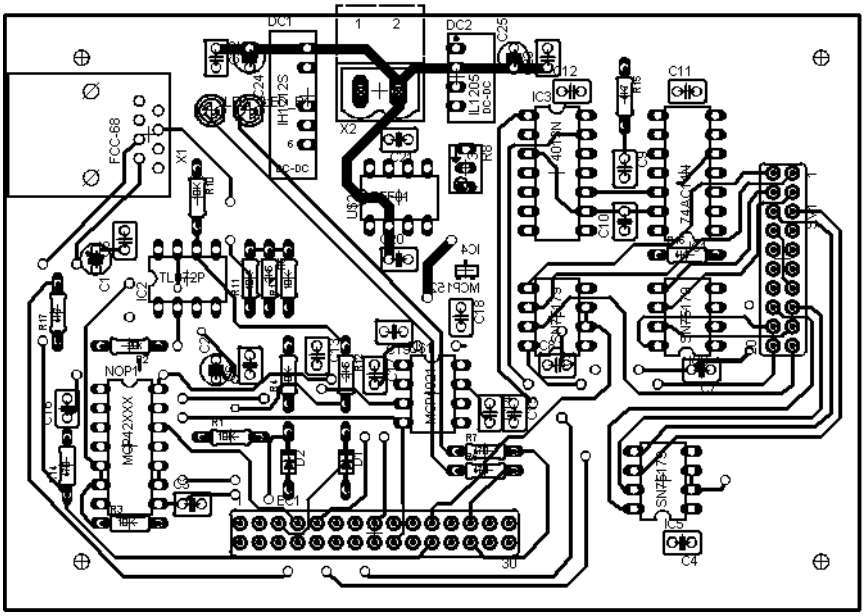


Figura A.4: Layout do PCB *Motion Interface* (bottom layer)

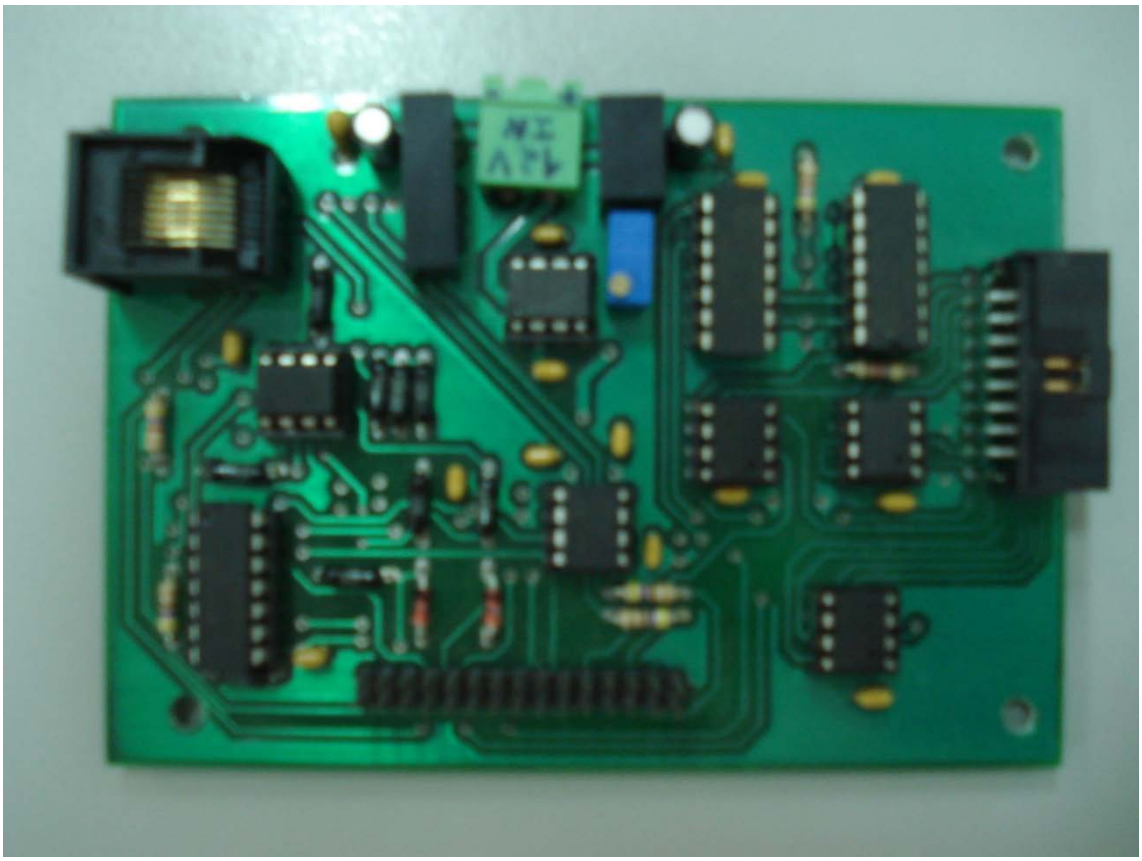


Figura A.5: PCB do módulo *Motion Interface*

A.2. *"Ultra Sound Interface"*

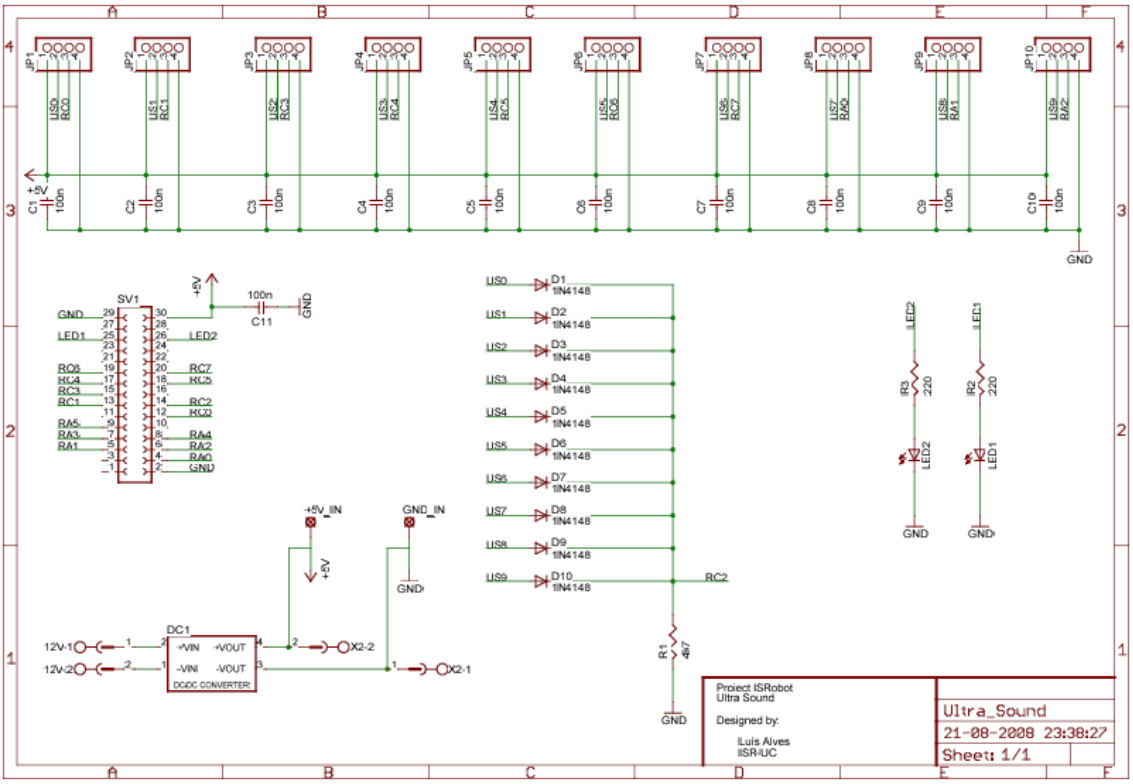


Figura A.6: Esquemático do módulo *Ultra Sound*

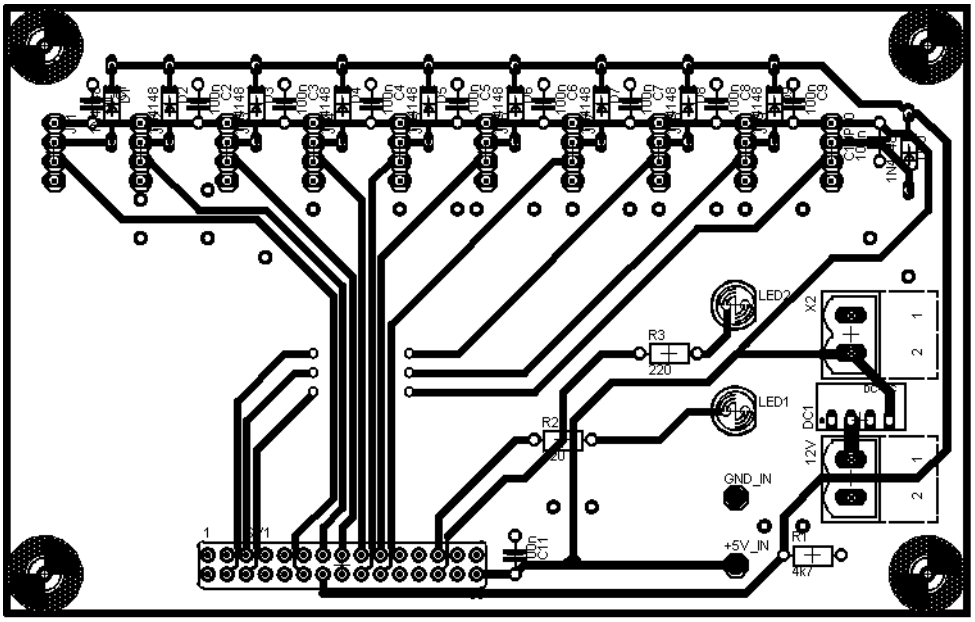


Figura A.7: Layout do PCB *Ultra Sound Interface* (bottom layer)

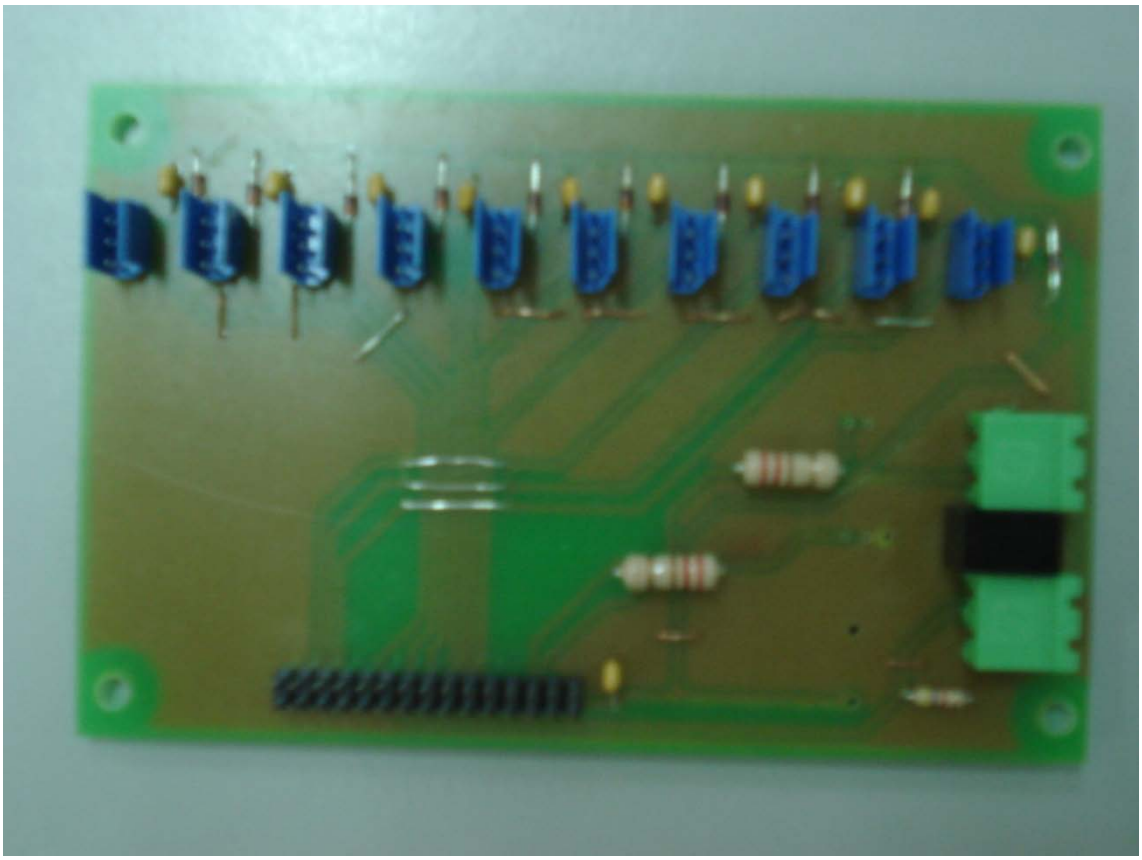


Figura A.8: PCB do módulo *Ultra Sound Interface*

A.3. "Infra Red Interface"

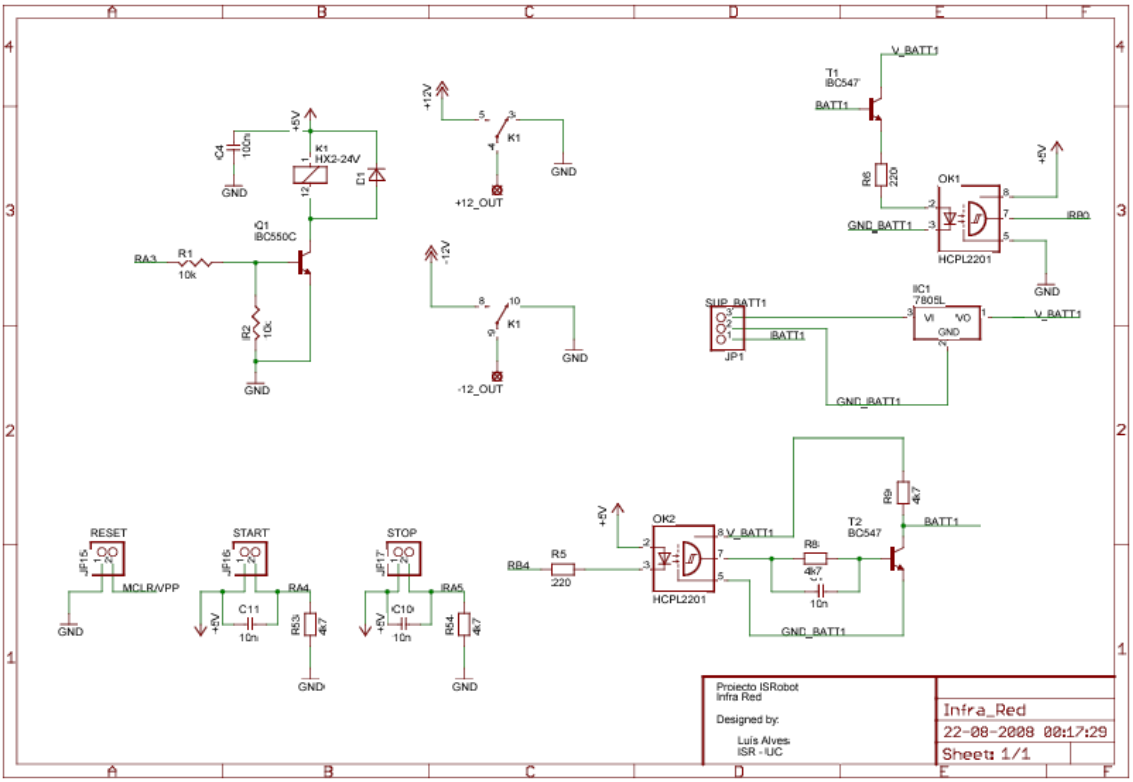


Figura A.9: Esquemático do módulo *Infra Red* (1/4)

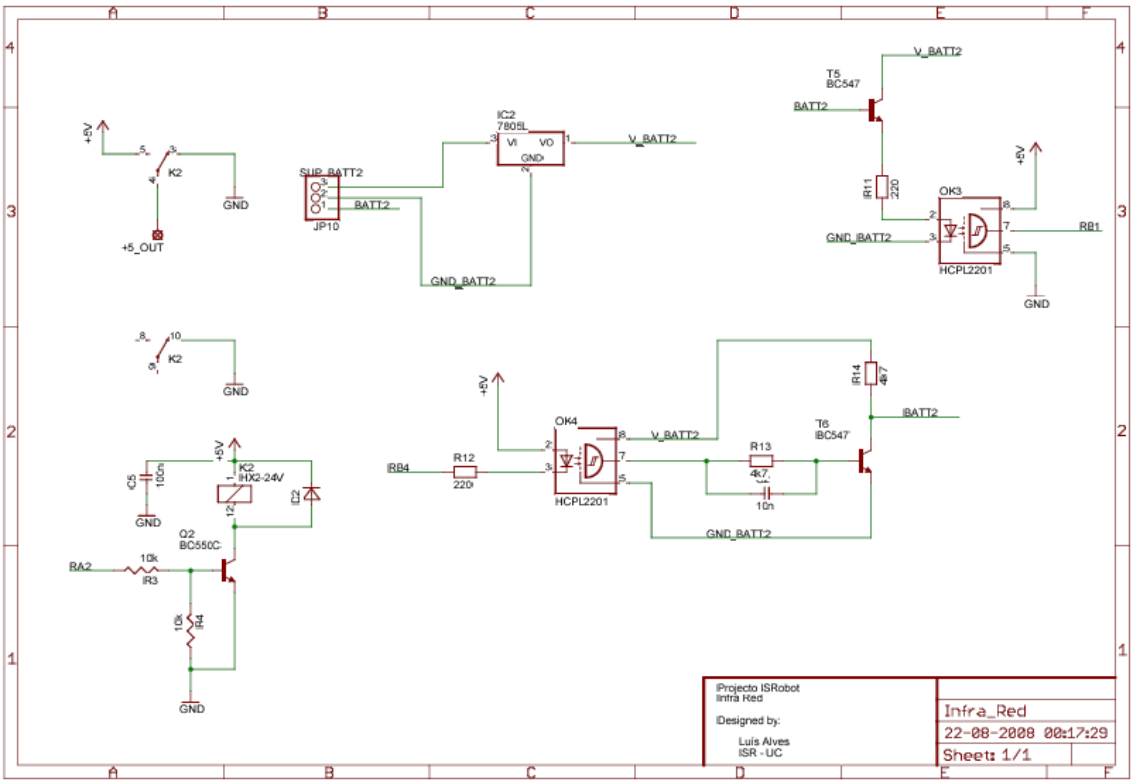


Figura A.10: Esquemático do módulo *Infra Red* (2/4)

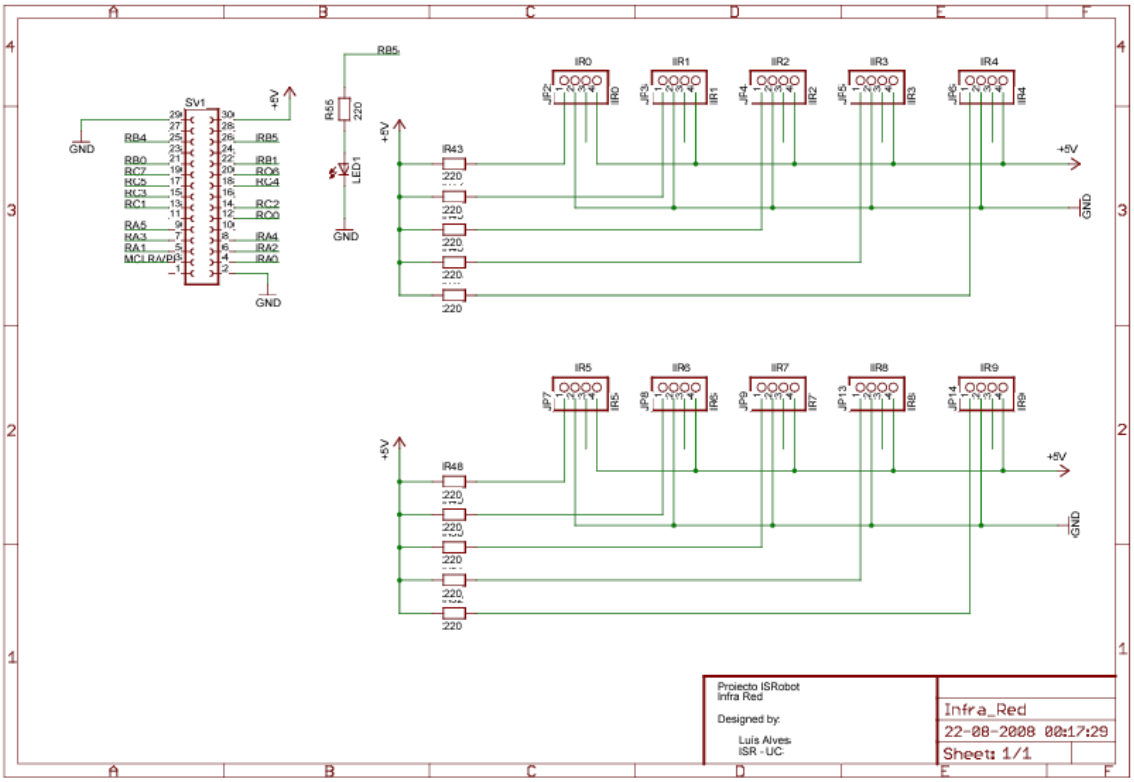


Figura A.11: Esquemático do módulo *Infra Red* (3/4)

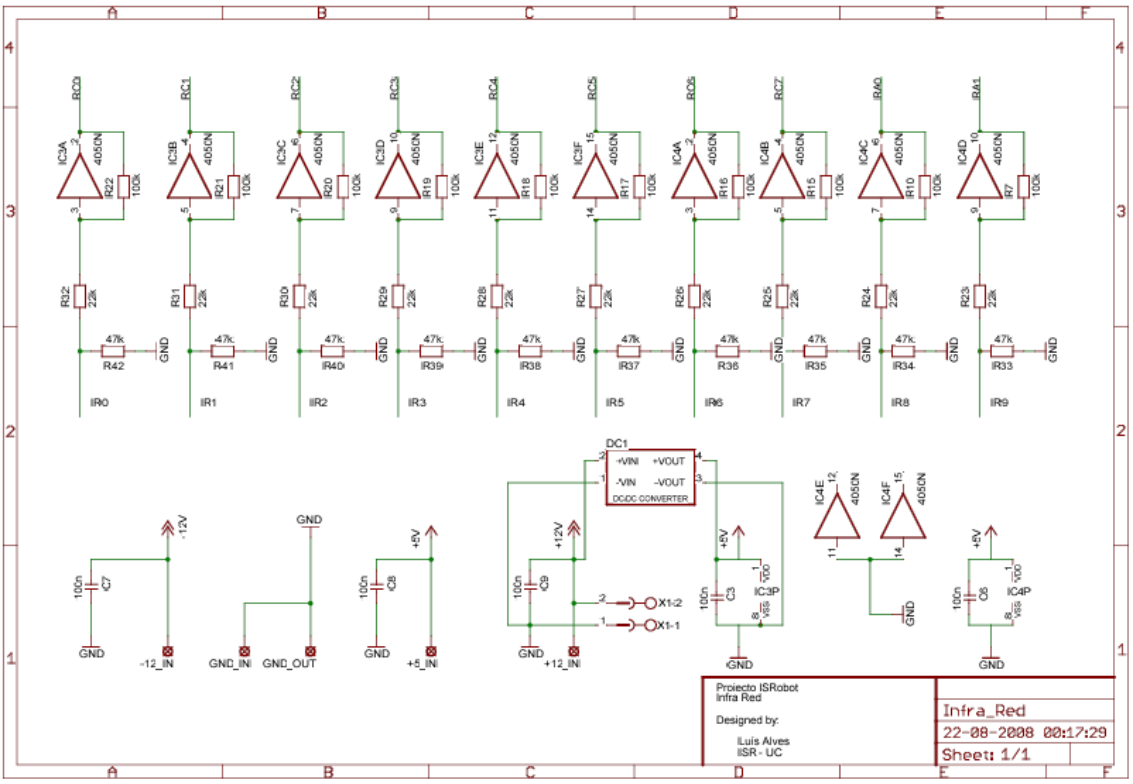


Figura A.12: Esquemático do módulo *Infra Red* (4/4)

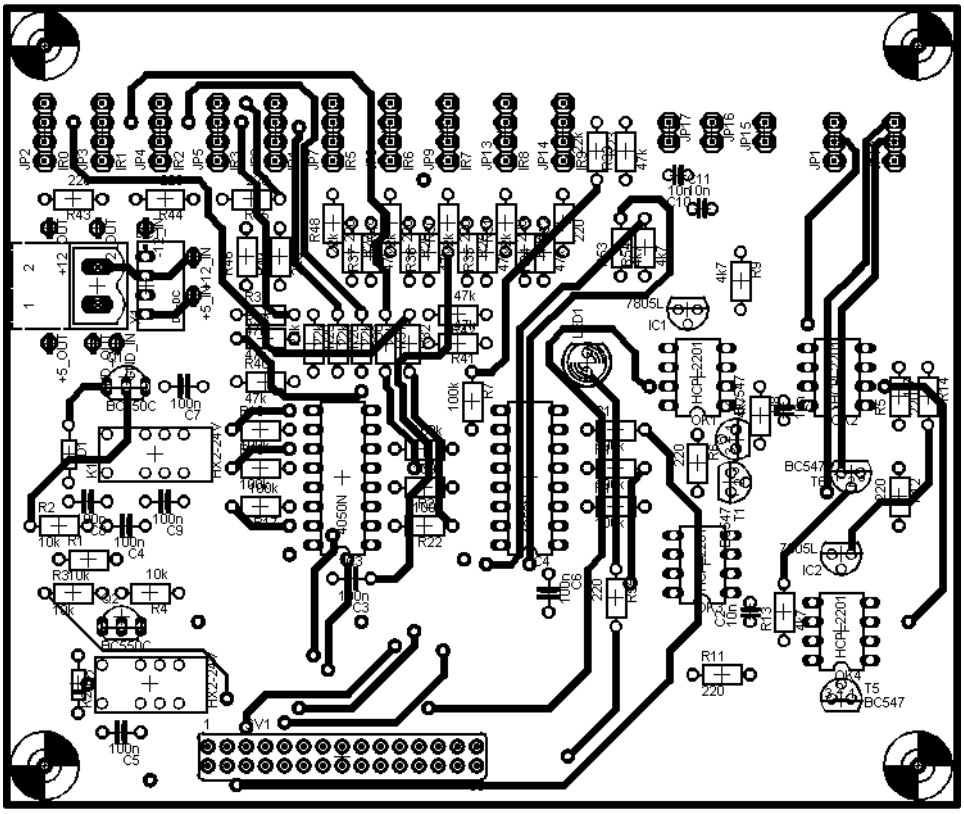


Figura A.13: Layout do PCB *Infra Red Interface* (top layer)

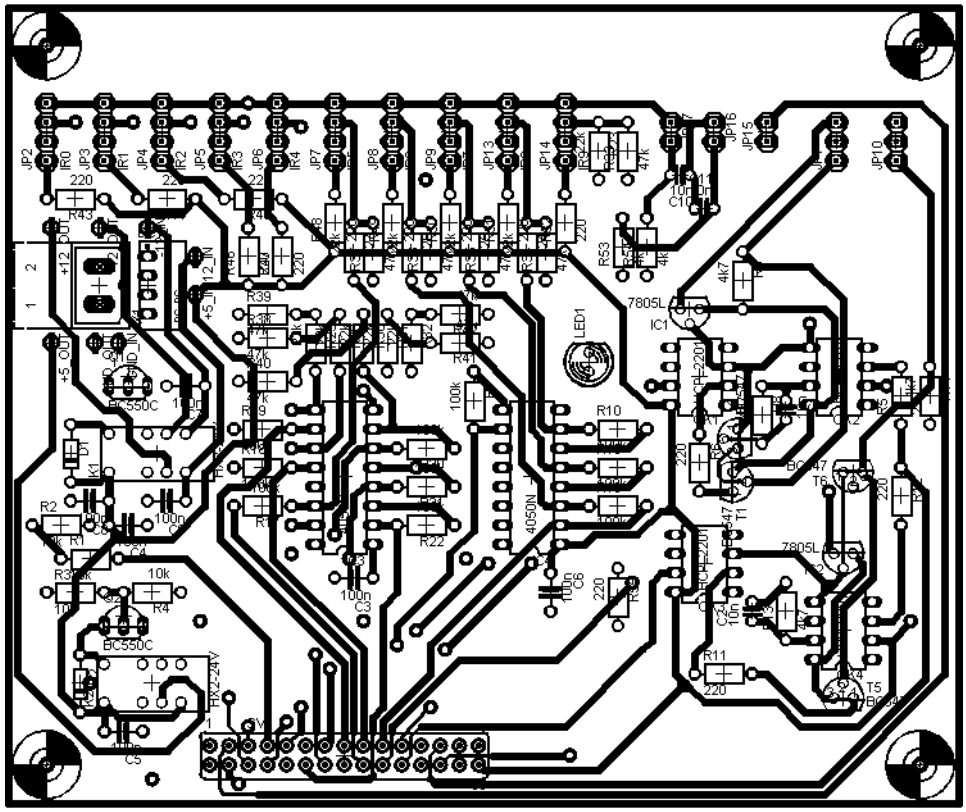


Figura A.14: Layout do PCB *Infra Red Interface* (bottom layer)

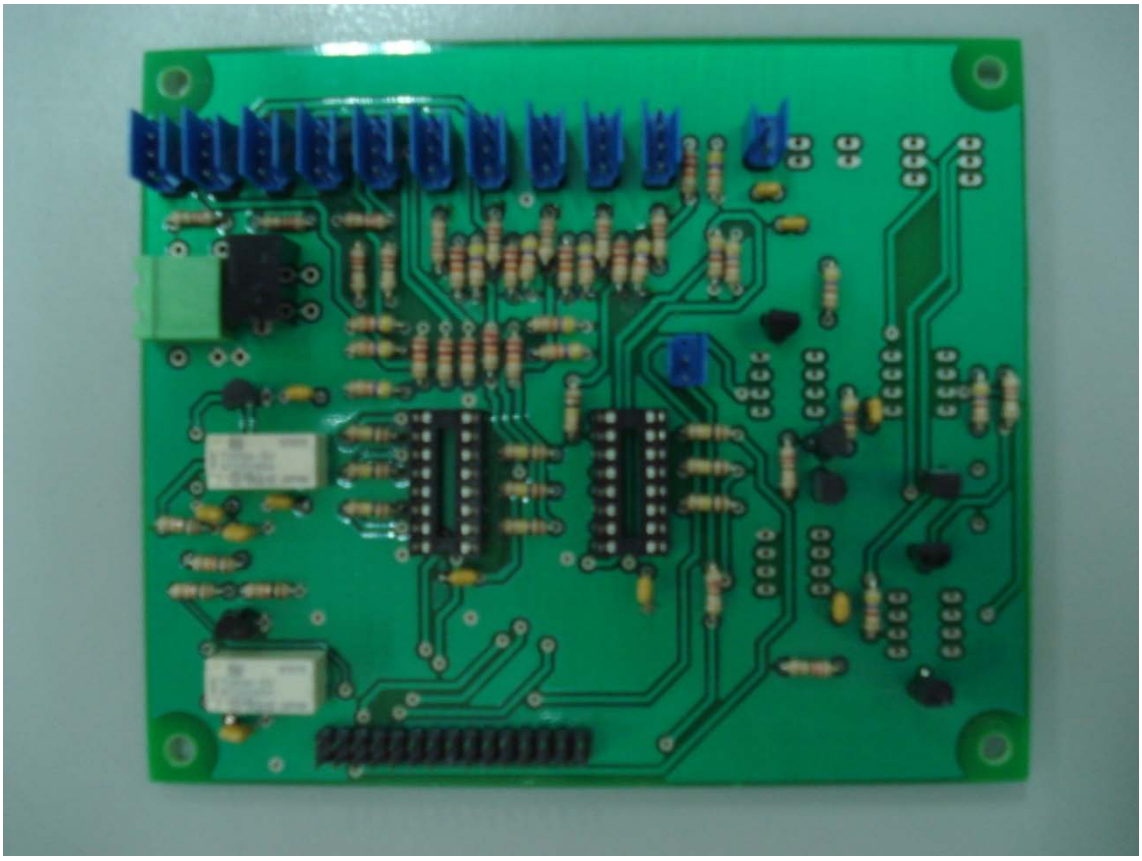


Figura A.15: PCB do módulo *Infra Red Interface*

A.4. *"Power Convertion"*

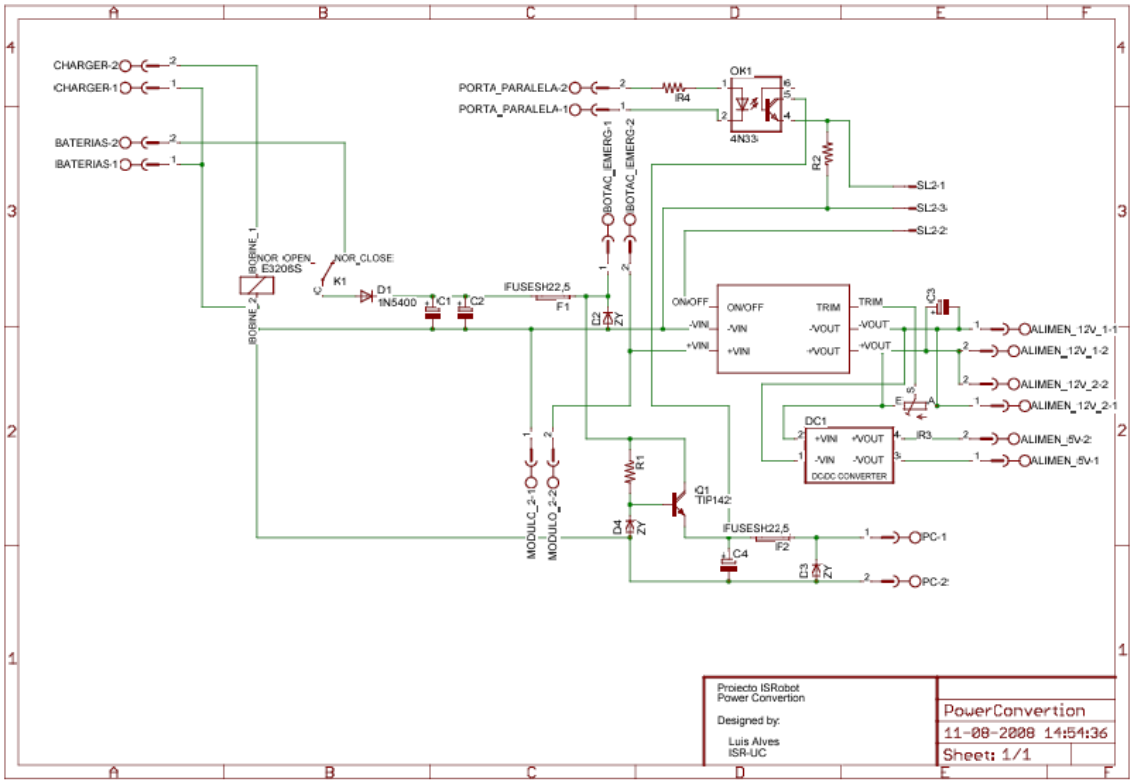


Figura A.16: Esquemático do módulo *Power Convertion*

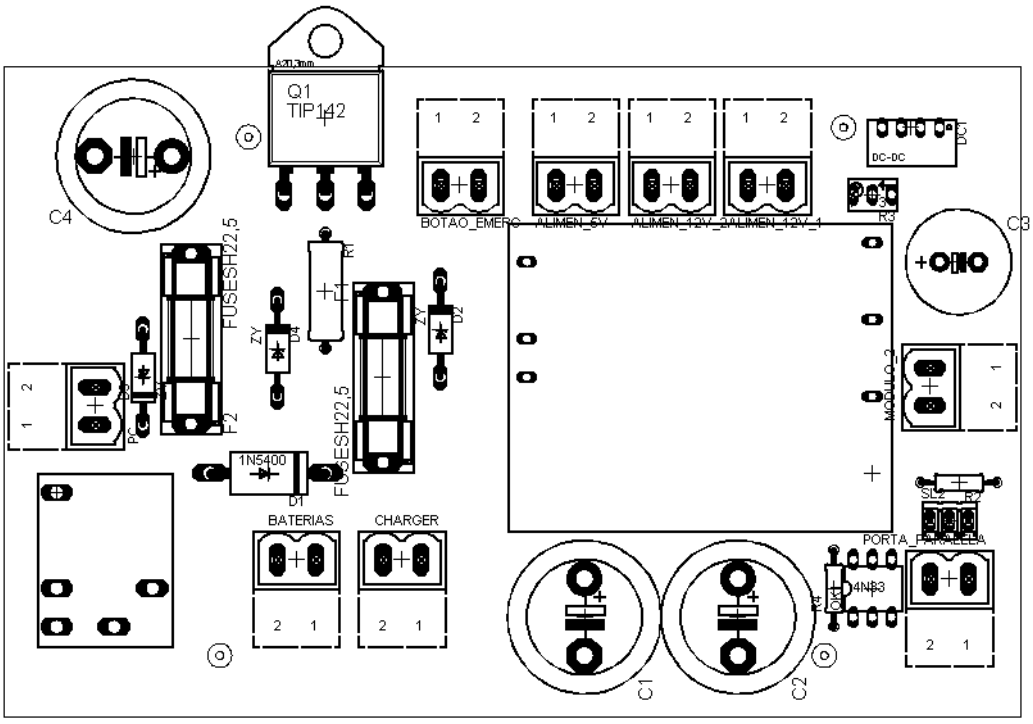


Figura A.17: Layout do PCB *Power Convertion Interface* (top layer)

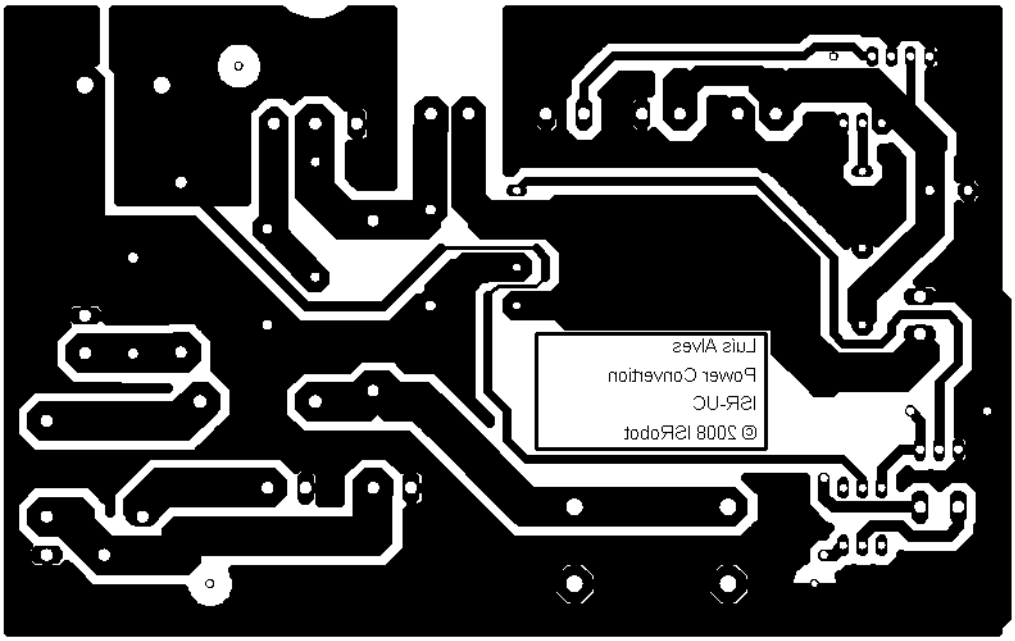


Figura A.18: Layout do PCB *Power Convertion Interface* (top layer)

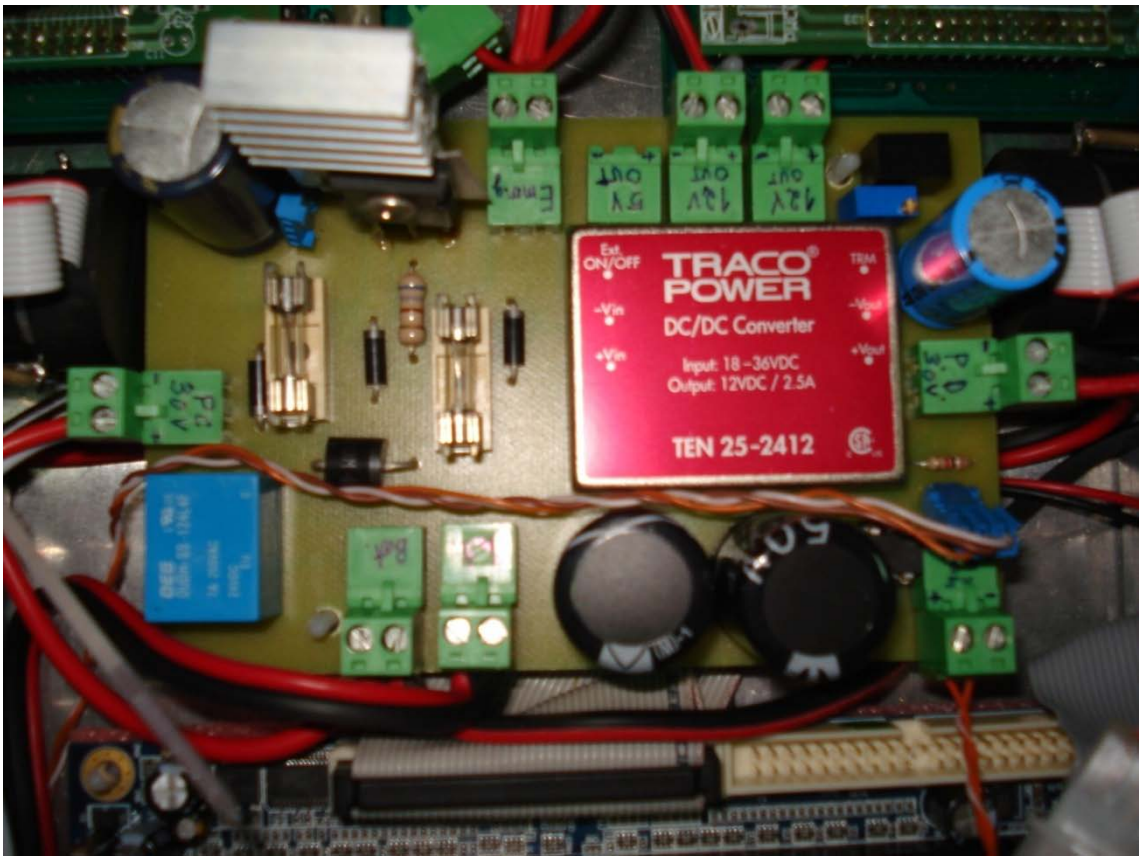


Figura A.19: PCB do módulo *Power Convertion*

Bibliografia

André e Tiago Versa Robot - Robô móvel versátil para competições em provas de robótica [Relatório]. - Porto : [s.n.], 2006.
Azevedo J. L. [et al.] Rota: a Robot for the Autonomous Driving Competion [Conferência]. - Albufeira : Robótica 2007-7th Conference on Mobile Robots and Competions, Abril de 2007.
Behringer R. e Maurer R.B.M. Results on visual road recognition for road vihicle guidance [Conferência] // Intelligent Vehicles Symposium. - [s.l.] : Proceedings of the 1996 IEEE, 1996.
Bertozzi M. e Broggi A. GOLD: A parallel real-time stereo vision system for generic obstacle and lane detection [Jornal]. - [s.l.] : IEEE Trans. Image Process, Jan. 1998. - 62-81 : Vols. 7, no. 1.
Cancela R. [et al.] ATLAS III - Um Robô com Visão Orintada para Provas de Condução Autónoma [Conferência]. - Coimbra : Robótica 2005 - Actas do Encontro Científico, Abril de 2005.
Dmitry S. e Kunbin H. Road Sign Recognition by Single Positioning of Space-Variant Sensor Window.
Jun M., Tsuyoshi K. e Yoshiaki S. An Active Vision System for Real-Time Traffic Sign Recognition. - Universidade de Osaka : [s.n.].
Lopes A. C. [et al.] An Outdoor Guidepath navigation System for AMRs based on Robust Detection of Magnetic Markers [Conferência] // 12th IEEE Conference on Emerging Technologies and Factory Automation. - ETFA'07.
Ma B., Lakshmanan S. e Hero A. O. Simultaneous detection of lane and pavement boundaries using model-based multisensor fusion [Jornal]. - [s.l.] : IEEE Trans. Intell. Transp. Syst., Sep. 2000. - 135-147 : Vols. 1, no. 5.
Maia R. Movimento de Robôs Móveis com Rodas de Tracção Diferencial: Modelação e Controlo do Sistema Motriz [Relatório]. - Coimbra : Tese de Mestrado, Universidade de Coimbra, 2004.
Mendes A., Conde Bento L. e Nunes U. Multi-target Detection and Traccking with a Laserscanner [Conferência] // Proc. of Intelligent Vehicles Symposium. - Parma, Italy : [s.n.], 2004. - pp. 796-801.
Mendes Abel Deteccção e Seguimento de Alvos com Laser Range Finder [Relatório]. - Coimbra : Tese de Mestrado, Universidade de Coimbra, 2004.
Pires G. [et al.] ROBCHAIR: A Powered Wheelchair using a Behaviour-Based Navigation [Conferência] // Proc. IEEE Int. Workshop on Advanced Motion Control (AMC). - Coimbra : [s.n.], AMC'98. - Vols. 536-541.
Pires G., Nunes U. e Almeida A. T. de ROBCHAIR – A Semi-Autonomous Wheelchair for Disabled People [Conferência] // Proc. 3rd IFAC Symposium on Intelligent Autonomous Vehicles (IAV). - Madrid : [s.n.], IAV'98. - pp. 648-652.
Premebida C. e Nunes U. Segmentation and Geometric Primitives Extraction From 2D Laser Range Data For Mobile Robot Applications [Conferência] // Actas do Encontro Científico do 5º Festival Nacional de Robótica. - Coimbra : [s.n.], Robótica'05.
Robótica Regras e Especificações Técnicas da Prova de Condução Autónoma [Relatório]. - 2008.
Solea R. e Nunes U. Robotic Wheelchair Control Considering User Comfort: Modeling and Experimental Evaluation [Conferência]. - Funchal : Fifth International Conference on Informatics in Control, Automation and Robotics (ICINCO), ICINCO'08.
Solea R. e Nunes U. Trajectory planning and sliding-mode control based trajectory-tracking for cybercars [Jornal]. - [s.l.] : Integrated Computer-aided Engineering, Int. Journal, IOS Press, 2007. - 1 : Vol. 14. - pp. 33-47.

Solea R., Nunes U. e Filipescu A. Trajectory planning and sliding mode control for WMR Trajectory-tracking and Path-following respecting human comfort travel [Conferência] // Controlo. - Lisboa : [s.n.], Controlo'06.
Sousa P. [et al.] Real-Time Architecture for Mobile Assistant Robots [Conferência] // 12th IEEE Conference on Emerging Technologies and Factory Automation (ETFA). - ETFA'07.
Sousa P., Araújo R. e Nunes U. Real-Time Labeling of Places Using Support Vector Machines [Conferência] // IEEE ISIE. - Vigo : [s.n.], ISIE'07.
Vaz L. Desenvolvimento de Módulos de Software numa arquitectura de Tempo-Real para Robots Móveis e Veículos Inteligentes [Relatório]. - Coimbra : Tese de Mestrado, Universidade de Coimbra, 2008.

Anexos Externos

Juntamente a este documento é anexo de forma externa a esta estrutura outros documentos nomeadamente:

1. **Real-Time Architecture for Mobile Assistant Robots [Artigo]**
2. Manual de utilização da RobChair [Manual Técnico]
3. Manual de utilização do ISRobot [Manual Técnico]

Real-Time Architecture for Mobile Assistant Robots

Pedro Sousa, Rui Araújo, Urbano Nunes, Luís Alves, and Ana C. Lopes
ISR-Instituto de Sistemas e Robótica
Department of Electrical and Computer Engineering,
University of Coimbra, P-3030-290 Coimbra, Portugal
{pasousa, rui, urbano, lalves, anacris}@isr.uc.pt

Abstract

Mobile robotics is a challenging research area, with produced results that were unthinkable several years ago. There exist algorithms and methods capable of performing difficult tasks such as detect/classify objects, skill learning and SLAM. From the initial design steps, the real-time software architecture of a robotic platform requires great attention. The problem is difficult, because various components, such as sensing, perception, localization, and motor control, are required to operate and interact in real-time. This makes the system a very complex one. This paper presents a real-time control architecture designed for mobile robots and intelligent vehicles. Moreover, an example of application of the control structure consisting on a system for learning to classify places, using laser range data, is reported.

1 Introduction

Mobile robotics involves a large number of research areas. Automatic control, artificial intelligence, learning systems may be found in the description of some works: NAVchair [1], SENARIO [2], etc. This multidisciplinary characteristic increases the complexity of the overall system, being difficult for researchers, with different backgrounds, to master everything. Software architectures take great importance in this area, splitting the complexity in smaller logical blocks, demystifying the platform complexity to newcomers.

In the past, hardware-related architectural aspects were dominant, whereas software-related architectural integrity, was inexistent, or was not a priority of system development. With the increasing use of software-intensive systems, this scenario has changed. In the sequence of this change, studies were made to understand, document and design better software (see, for example, [3], and [4]). The attempt to define and codify the system has also been studied. Among others, the Wright language [5], and "IEEE Recommended Practice for Architectural Description of Software" [6], are interesting tools to work with.

In mobile robotics, in the effort to better organize pieces of software, architectures were proposed for many years, from Brook's Subsumption [7] to Chochon object-based architecture [8].

At present, projects exist with the objective to create a common architectural environment, allowing the reusability of the source code between mobile platforms, such as Player/Stage [9], and Genom/Orocos [10][11].

These tools are limited in the sense of not being complaint with hard real-time systems or difficult to work with. Hard-real time capabilities are required to successfully complete autonomous missions. Key features including precision, security, and determinism depend on them.

Many groups [12], [13], use commercially available real-time operating systems such as QNX, LynxOS, and vxWorks. While these systems have good real-time capabilities, they are too expensive, miss device drivers and lack expandability.

An alternative can be taken to create a hard real-time system. This approach inserts a real-time micro kernel on top of the hardware and run the standard Linux kernel as its lowest priority task. Then, the Linux kernel,can be preempted at any time to run real-time tasks. This solution is provided by the main open-source real-time Linux projects, RTLinux, RTAI, Xenomai. Working with this kind of solution has the advantage to provide a cheaper development platform than the commercial available systems. Another advantage is the possibility to interact with or use Linux resources, if hard real-time is not required.

A common practice in real-time systems is to develop the software to make it work as a kernel module. The main disadvantage of this procedure is the impossibility to use some software libraries previously developed. A new method developed by RTAI and Xenomai projects permits to execute real-time software outside the kernel. The two projects also provide a framework (RTDM) to create real-time device drivers easily.

In this paper a real-time architecture for mobile robots, using object-oriented programming, and on-board devices connected through fieldbuses, is presented.

[1]This research was supported in part by Fundação para a Ciência e Tecnologia (FCT), under contract NCT04:POSC/EEA/SRI/58016/2004.

Figure 1. Robchair WMR platform.

In the next section, the actual hardware control structure is outlined. Section 3 describes data transfer protocols between devices. In Section 4, the software architecture existing in the platform is analyzed. Finally, an example of application is described in Section 5, followed by the conclusion in Section 6.

2 System Description

Figure 1 illustrates the Robchair, robotic wheelchair used as the experimental setup. The wheelchair is composed by two motorized rear wheels and two casters in front. There is also a fifth rear wheel connected to the back of the wheelchair with a damper used for stability. During the lifetime of the project, several electronic components have been attached to this base structure in order to improve environment perception and actuation capabilities [14], [15], [16].

Figure 2 presents a block diagram of the actual hardware control architecture. The wheelchair is powered by two 12V batteries which feed two permanent magnet DC motors with 24V input voltage and 1000 RPMS. These motors are coupled to two gearboxes with factor 1:10 (1 complete wheel revolution corresponds to 10 complete motor revolutions). With the aid of these gearboxes, each wheel may have a nominal torque of 29.3 Nm. There are two power drivers that guarantee the independent and direct control of the motors. Two encoders have been coupled to motor axis, providing the velocity feedback of each motor. The encoders are also used to obtain dead-reckoning data for integration into localization methods.

An embedded PC is responsible for giving some degree of intelligence to the robot. This computer is connected to distributed devices through fieldbuses, which will be explained in Section 3. The platform is connected to external devices through a wireless link. This connection allows the implementation of a distributed architecture, which exhibits the possibility and capability to extend our single robot to other perspectives, like multi-robot cooperation, its integration in intelligent environments, etc.

User interfaces, such as the joystick and safety emergency systems, are also present.

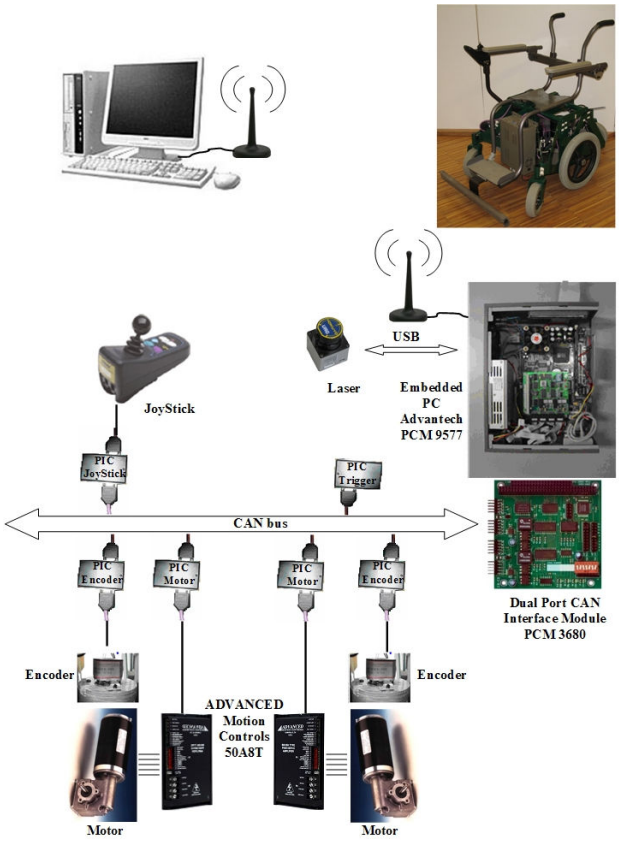


Figure 2. Robchair hardware architecture.

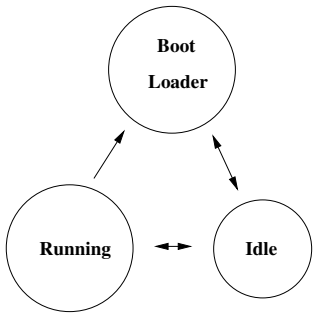


Figure 3. μC running modes.

Hokuyo URG-04LX laser range-finders [17], with 240-degree scanning capability giving 632 measures per scan, are also used in the platform. Section 5 describes a place labeling system which uses range-bearing data directly from the laser range-finder.

A real-time software architecture was developed taking advantage of the above described features, in order to implement the perception, control, and navigation methods in the wheelchair.

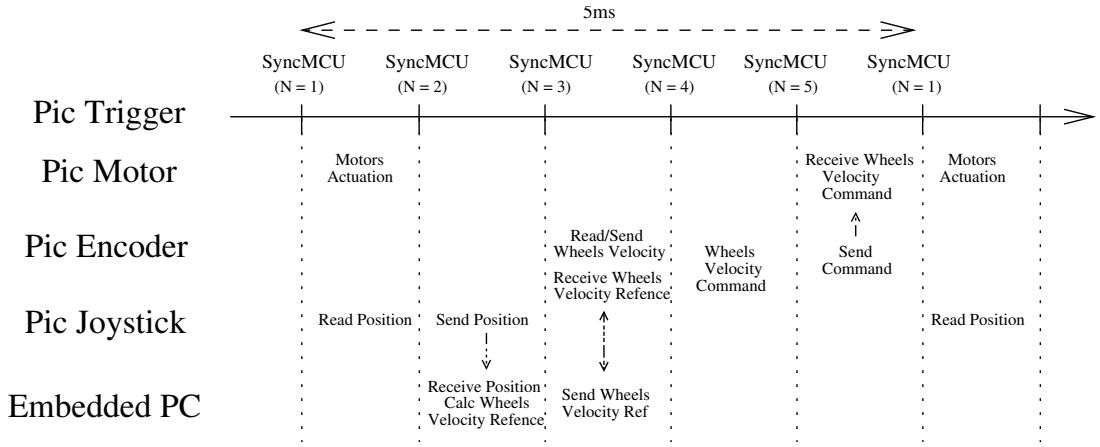


Figure 4. Messages and actions of a control cycle of 5ms.

3 Communication Protocol

As depicted in Fig. 2, the on-board devices distributed through the robot platform are connected through fieldbuses:

- Controller Area Network (CAN).
- Universal Serial Bus (USB).

The CAN is used for data transfer of small critical messages between devices, while the USB is mainly used for devices which send or receive large amounts of data. The CAN fieldbus is a deterministic protocol guaranteeing the delivery of messages (otherwise an error is reported). The main disadvantage of the CAN protocol is the low bandwidth of 1Mbit/s maximum [18]. On the other hand, the USB has a higher bandwidth, permitting 480 Mbit/s if working with devices compatible with the USB 2.0 protocol [19]. The main USB drawback is its large latency, which is inadequate for real-time systems.

All the distributed devices, connected through CAN, use a base printed circuit board, containing a microchip micro-controller (μC), as described in [20]. The devices were designed to have three operating states as illustrated in Fig. 3. The "*Boot Loader*" state is the first state to run when the μC is powered. In this state, the μC can be re-programmed without disconnecting the hardware. The "*Idle*" state is used to put the μC in a standby mode, where its normal processing is suspended. The "*Running*" state is used for controlling the device in normal operation, allowing for instance to obtain sensor data from sensor nodes and to send actuation commands to actuator nodes.

A custom communication protocol, based on the time-triggered protocol paradigm, was designed and implemented. All events are synchronized by a message, sent from a "Synchronization Micro-Controller Unit" (syncMCU), that synchronizes other Micro-Controller Units (MCUs), and defines the control loop time reference. The syncMCU message is sent, every 1 ms, by the "PIC Trigger" node, instructing the MCUs to perform data acquisition and/or actuation. Figure 4 shows the message flow along time. As shown, the syncMCU messages are triggered each 1 ms, but the control cycle time is of 5 ms. When a MCU receives the synchronization message, an internal processing is triggered, taking the corresponding action according to the predefined time table. Note that Fig. 4 only represents a single motor and encoder, being the procedure duplicated to the other motor/encoder nodes.

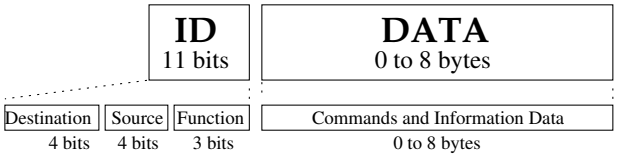


Figure 5. CAN: ID field composition.

Robchair can operate in different modes such as autonomous navigation, under human-machine shared control mode, and under human control mode using some human-machine interface such as a joystick. In each case, a specific time table is executed, being the description in Fig. 4 specific for the human-controlled mode as used in the place labeling task described in Section 5.

Each CAN message has an identifier (ID) field [18]. The ID field of each message is composed by a combination of a unique identifier of the source and destination nodes plus the requested operation. Figure 5 depicts ID field composition used in the protocol.

Devices connected to the USB are directly connected to the embedded PC and are directly accessed through the Linux operating system application programming interface (API). The on-board laser devices communicate through USB using the protocol described in [17].

4 Embedded PC Software Architecture

The Robchair has been used in testing of autonomous navigation methods, in non-structured and dynamic en-

vironments, transporting people and operating near other people and objects. Its malfunction can lead to unwanted results or accidents. Systems like this should be considered as critical system, which impose real-time constraints and fault-tolerance concerns. Fault-tolerance is not analyzed in this paper.

For this purpose, the robot must perceive and learn a model of the environment, and take decisions based on the perceived world. Robchair is designed to operate in different modes: fully autonomous mode, where the decisions are made as a result of processing the information perceived from the environment; and human-machine shared control mode, where the decisions result from a symbiosis of human and machine, and are function of the environment state.

From the control viewpoint, the robot performs many tasks related to the implementation of control loops, operating with pre-specified deadlines, that transform sensor data into actuator commands. Each task has its own sample rate, computational load and time constraint. To ensure the time execution of all tasks, a real-time system (RTAI) is used.

From the system developer viewpoint, the system should have properties such as: extensibility, reliability, maintainability, availability, security, and usability. Object oriented programming, developed with characteristics of code modularity and readability, was chosen to facilitate the attainment of the desired properties. Using a stable and standardized language it is guaranteed that all present and future developers can learn and understand the language, and consequently the architecture. Taking into account this property, good performance in computational time and in the low memory requirements (in generated applications), the C/C++ programming languages were used to code the entire system. These languages have also a big support on Linux systems which are used as the base operating system of the wheelchair.

Figure 6 shows the designed Robchair software architecture. The architecture is composed by three logical layers: the fieldbus communication layer, the low-level motion control layer, perception and navigation layer.

The bottom layer (Fieldbus communications) includes tasks that acquire data from the environment through on-board devices, and tasks which communicate to other units. Communication with on-board devices is executed through fieldbuses, as explained in Section 3. The determinism and scalability of messages are important aspects considered at this low-level layer of the architecture. In the RTAI Linux computer, for each fieldbus device there is a slot of shared memory containing the most recently received data. Semaphores are used for mutual exclusion and synchronization when accessing the slots. These data slots can be accessed by other layers.

The middle layer was designed to perform low-level motion/velocity control. This layer directly controls the devices sending them commands through the lower layer. The velocity commands sent to each individual wheel are computed by tasks running in this layer. The velocity commands are calculated taking into account the trajectory obtained from a planning module and the robot kinematics. This layer also contains a localization module that estimates the robot pose fusing dead-reckoning information and landmark detection.

The top layer (perception and navigation layer) includes higher level algorithms such as place classification, path planning, etc. At this level, as shown in Fig. 6, there is a global observer, supervising the functioning of all modules, allowing to start or stop modules. This permits the reconfiguration of the robot behavior.

This paper also describes an example of application of the real-time architecture, where the robot is controlled by joystick and, at the same time, range-bearing data is collected from laser sensors. The modules of the architecture that are used in the application are identified in blue color in Fig. 6.

Modularity and flexibility were requisites in the architecture design. In our laboratory new algorithms and methods are continuously being tested and validated. A modular system was implemented with each module being developed as a C++ class. Each class has standardized methods for base functionalities, such as initialization and termination of the modules. Module data are contained in the class, avoiding problems with corruption of global data. All modules are inherited from another class containing the RTAI API interface, creating a simple access to the real time system tools.

The communication protocols (CAN, USB, and Ethernet based) are also encapsulated into classes. This leads to improvements of device usability and readability of the source code. There is another advantage of representing devices by objects: the devices communicate through fieldbuses, where several devices are connected and communicate through a common protocol. The employed protocols are prone to change and it can be very hard to trace changes in all devices individually. In the current system, a base object was developed containing the communication protocol which is inherited by the objects communicating through that fieldbus. Using this property, changes carried out in the base communication class are automatically transmitted to the devices' objects.

5 Place Classification using Range Data

This section presents an example of application of the real-time architecture. A system designed to perform semantic classification of places is described. At the current stage, the classification system can differentiate between corridor and room, but the results are promising to expand the method.

While classifying systems, the movement is controlled by human with the joystick. Figure 7 shows the motion control block diagram. Velocity commands (v, ω) are extracted from the joystick. These commands are sent to the embedded PC, where both wheel desired velocities $(\dot{\theta}_{d1})$

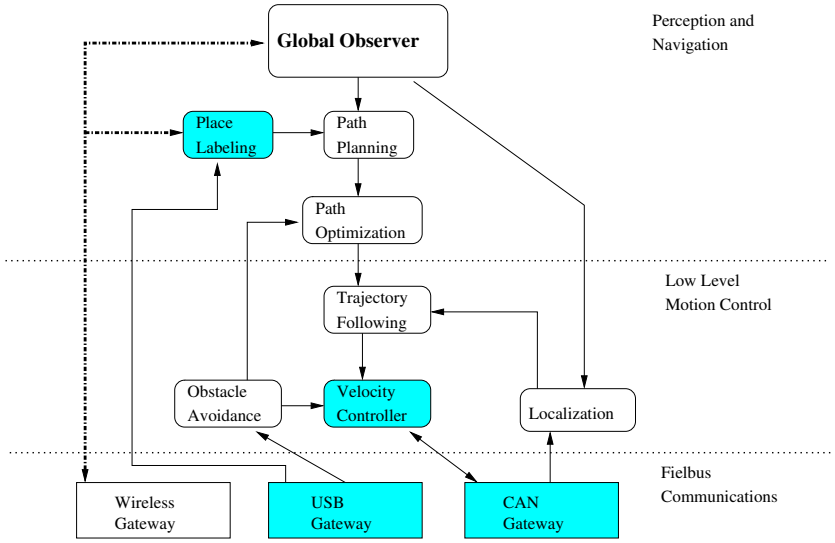


Figure 6. Complete Robchair software architecture design.

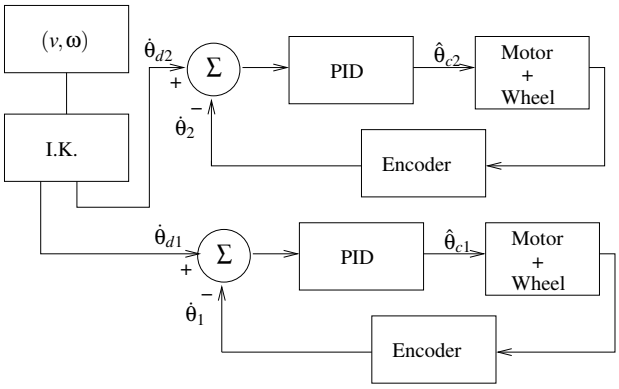


Figure 7. Place classification motion control.

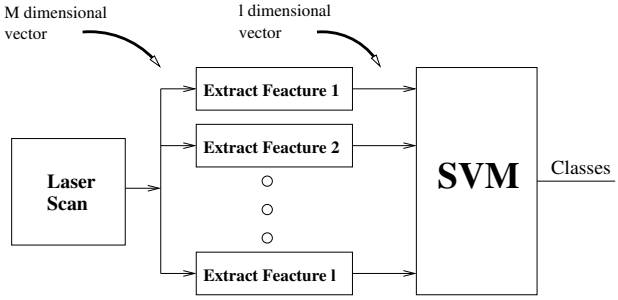


Figure 8. General SVM method overview.

and $(\dot{\theta}_{d1})$ are determined, using inverse kinematics (I.K.). These values are sent to PID controllers which produce the wheels' motor velocity commands. The real wheels' velocities are determined from wheels' encoder data.

Independently from the motion system, range-bearing data is collected from an USB-connected laser sensor, at each 500 ms. This gathered range-bearing data is used in the method described below.

The method presents two distinct phases: learning and prediction. In the learning phase, data is collected from environment areas representative of the different classes of places. Later, all collected data is processed in order to create models of the environment used for classifying places.

The method presented in this section has the logical modules presented in Fig. 8.

At the first step, collected data is processed into simple geometrical features. The extracted features are later used as inputs of a support vector machine (SVM) learning algorithm. Previous work, described in [21], focused on the internal properties of the SVM algorithm.

It will be assumed that each sensor observation $\mathbf{z} = \{\mathbf{b}_1, \ldots, \mathbf{b}_M\}$ is composed of a set of range-bearing measures $\mathbf{b}_i = (\alpha_i, d_i)$ where α_i and d_i are the bearing and range measures, respectively. Each training example for the SVM algorithm is composed by one observation $\mathbf{z}_i$ and its classification υ_i. The set of training examples is then given by

$$E = \{(\mathbf{z}_i, \upsilon_i) : \upsilon_i \in \Upsilon = \{\text{Room}, \text{Corridor}, \ldots\}\} \quad (1)$$

where Υ is the set of classes. In this paper it is assumed that the classes of the training examples are given in advance. The main goal is to develop a learning classifier of places that is able to generalize from these training examples, and that can later classify unseen places.

If the training examples of the set (1) were used directly as inputs to the SVM classifier, then all possible situations should be trained in order to attain a good classification rate. Thus, the raw data was transformed into a group of simple geometrical features from which the classification of places could be extracted. In order for the classification

system to depend only on the (x, y) position of the robot and be invariant to robot rotation, the features should be invariant to rotation. Features should also be computationally not heavy. The choice was to use a set of geometrical features often used in shape analysis [22], [23], [24], [25], [26].

Define $\mathcal{Z}$ as the set of all possible observations; i.e. observations obey $\mathbf{z} \in \mathcal{Z}$. A feature f_i is a function that takes one observation $\mathbf{z}$ as argument and transforms it to a real value $f_i(\mathbf{z})$; i.e. $f_i : \mathcal{Z} \rightarrow \mathbb{R}$.

Two methodologies were used to extract features from observations. Thus, two different sets of simple features were produced for each observation $\mathbf{z}$. The first set B was calculated directly from the sensor data in E, i.e. from the raw range measures $\mathbf{z}$. The used features are specified in Table 1.

Table 1. Set B of simple features extracted from raw beams $\mathbf{z}$

- Average difference between the length of consecutive measures.
- Standard deviation of the differences between the length of consecutive measures.
- Average difference between the length of consecutive measures considering a maximal possible value.
- Standard deviation of the differences between the length of consecutive measures considering a maximal possible value.
- Average measure length.
- Number of gaps in a scan (we consider a gap when absolute difference between consecutive measures is higher than a value.

The second set P of features is calculated from polygonal approximations, $\mathbf{P}(\mathbf{z})$, of the area covered by the observations $\mathbf{z}$. The vertices of each closed polygon $\mathbf{P}(\mathbf{z})$ correspond to the Cartesian coordinates of the end points of each range measure $\mathbf{b}_i \in \mathbf{z}$ relative to the robot, i.e.:

$$\mathbf{P}(\mathbf{z}) = \{\mathbf{v}_1, \ldots, \mathbf{v}_M\} \quad (2)$$

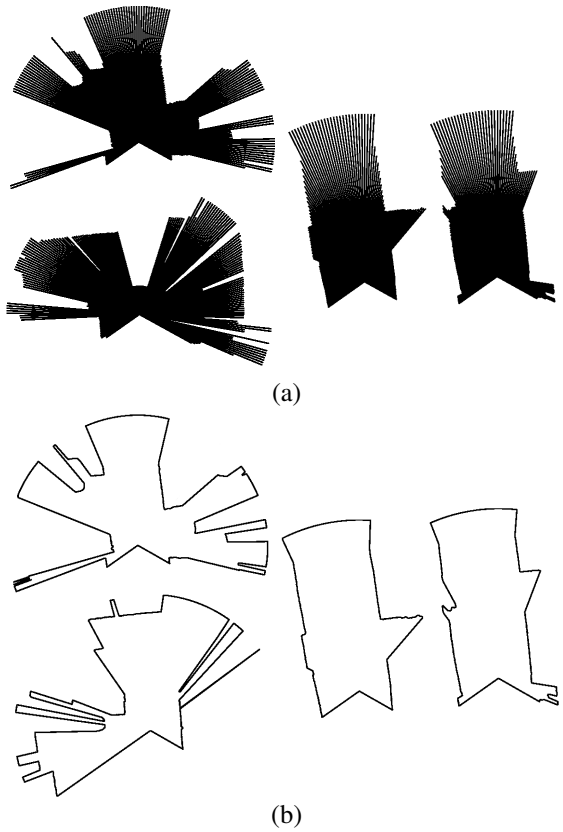
where $\mathbf{v}_i = (x_i, y_i)$, and $x_i = d_i \cos(\alpha_i)$ and $y_i = d_i \sin(\alpha_i)$. Examples of polygonal representations of laser range scans are shown in Fig. 9. The used features are described in Table 2.

Let l be the total number of features. Let all feature functions $f_i(\mathbf{z})$ be grouped into a feature mapping vector function $\mathbf{f} : \mathbb{R}^M \rightarrow \mathbb{R}^l$, with $\mathbf{f}(\mathbf{z}) = [\ f_1(\mathbf{z}) \quad \ldots \quad f_l(\mathbf{z})\ ]^T$. Then, the input data set S to the SVM which is obtained by transforming raw data, becomes:

$$S = \{(\mathbf{f}(\mathbf{z}_1), \upsilon_1), (\mathbf{f}(\mathbf{z}_2), \upsilon_2), \ldots, (\mathbf{f}(\mathbf{z}_l), \upsilon_l)\}. \quad (3)$$

Table 2. Set P of simple features extracted from closed polygon $\mathbf{P}(\mathbf{z})$ that represents raw beam $\mathbf{z}$

- Area o $\mathbf{P}(\mathbf{z})$.
- Perimeter of $\mathbf{P}(\mathbf{z})$.
- Area of $\mathbf{P}(\mathbf{z})$ divided by perimeter.
- Seven invariants calculated from the central moments of $\mathbf{P}(\mathbf{z})$.
- Normalized feature of compactness of $\mathbf{P}(\mathbf{z})$.
- Normalized feature of eccentricity of $\mathbf{P}(\mathbf{z})$.
- Form factor of $\mathbf{P}(\mathbf{z})$.
- Circularity defined as $\frac{\text{perimeter}^2}{\text{area}}$.



(a) (b)

Figure 9. (a) Examples of scans recorded in a room (left) and corridor (right). (b) Polygonal representations of the scans: room (left) and corridor (right).

where υ_i are the desired output classes, which are considered to be defined beforehand.

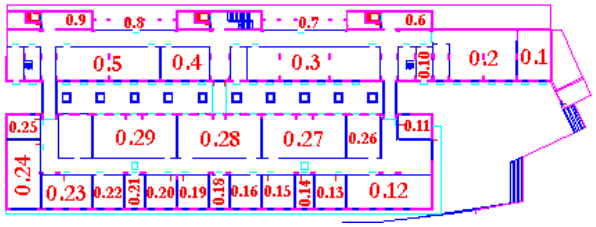


Figure 10. Map of the environment used in the experiments.

Each observation $\mathbf{z} = \{\mathbf{b}_1, \ldots, \mathbf{b}_{632}\}$ is transformed into a set of 14 simple features described previously. Thus, the feature mapping function is $\mathbf{f} : \mathbb{R}^{632} \rightarrow \mathbb{R}^{14}$.

The sensor data sets were collected in the office-like building of ISR-Coimbra (illustrated in Fig. 10). The design of the method enables it to operate independently of specific environment characteristics (both in training and classification phases), i.e. the method is prepared to operate in various environments and to be robust to environment changes. In this context, five data sets were collected in several corridors and rooms. The first data set was used for training the classifier, and the other data sets were used to test the classification system with data representing new observation situations not present in the training data. The training data set was composed of 527 sensor observations $\mathbf{z}_i$ and the corresponding place classifications υ_i. This is a relatively small training set. One of the four testing data sets corresponds to a corridor. The other three testing data sets were obtained in different rooms, and are named "Room 1"[1], "Room 2"[2], and "Room 3"[3].

In different training sessions, the SVM classifier was tested with different kernels. After obtaining the SVM model, the classifier was tested, with different data sets. Tables 3, 4, and 5 present the results obtained with the three different kernels. The best results were obtained with the Gaussian RBF kernel, where the hit ratio was always above 80% except for Room 3. These are promising results obtained with the relatively small data set used to train the SVM-based classifier. In comparison with [27], the present approach has the advantage of not requiring the design of a set of weak classifiers.

If the time taken by the method is analyzed, it can be divided into two distinct intervals. First, the time taken to extract all features listed previously is at maximum 3.18 ms. The other time, the time taken by the SVM to determine the environment class, is negligible when compared to feature computation time.

[1] equivalent to 0.3 on map
[2] equivalent to 0.16 on map
[3] equivalent to 0.5 on map

Table 3. Classification results for different indoor areas and using a sigmoid kernel

	Total samples	Correctly classified	Correct rate
Corridor	48	36	75%
Room 1	112	84	75%
Room 2	27	18	67%
Room 3	57	43	75%

Table 4. Classification results for different indoor areas and using a polynomial with degree $d = 5$ kernel

	Total samples	Correctly classified	Correct rate
Corridor	48	38	79%
Room 1	112	88	79%
Room 2	27	21	78%
Room 3	57	43	75%

Table 5. Classification results for different indoor areas and using a Gaussian RBF kernel

	Total samples	Correctly classified	Correct rate
Corridor	48	41	85%
Room 1	112	92	82%
Room 2	27	22	81%
Room 3	57	44	77%

6 Conclusion

This paper presents an architecture developed for real-time control of mobile robots and autonomous vehicles. The combination between C/C++ languages and linux/RTAI proved to be productive, stable, and with good performance. The real-time system guarantees determinism. By employing RTAI the system is developed in a Linux system, which gives us access to a rich set of resources such as network, graphics, and device I/O. The practical results attained with the architecture are very good and permit the implementation and testing of various types of algorithms such as environment mapping and learning, localization, path tracking, among others. Data transfer between modules proved also to be efficient. The development of general data transfer functions give us increasing productivity. The architecture has shown to be easily expandable. In the near future this architecture will be incorporated in other robots and vehicles.

References

[1] S. Levine, D. Bell, L. Jaros, R. Simpson, Y. Koren, and J. Borenstein, "The navchair assistive

wheelchair navigation system," *IEEE Transactions on Rehabilitation Engineering*, vol. 7, no. 4, 1999.

[2] M. Mazo, J. C. García, F. J. Rodríguez, J. U. na, J. Lázaro, and F. Espinosa, "Integral system for assisted mobility," *Inf. Sci. Inf. Comput. Sci.*, vol. 129, no. 1-4, pp. 1–15, 2000.

[3] M. Shaw and D. Garlan, *Software Architecture - Perspectives on an emerging discipline*. Prentice Hall, 1996.

[4] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice, Second Edition*. Addison-Wesley Professional, April 2003.

[5] R. Allen and D. Garlan, "A formal basis for architectural connection," *ACM Trans. Softw. Eng. Methodol.*, vol. 6, no. 3, pp. 213–249, 1997.

[6] I. A. W. Group, "IEEE std 1471-2000, recommended practice for architectural description of software-intensive systems," IEEE, Tech. Rep., 2000.

[7] R. A. Brooks, *A Robust Layered Control System For a Mobile Robot*. Cambridge, MA, USA: Massachusetts Institute of Technology, 1986.

[8] H. Chochon, "Object-oriented design of mobile robot control systems," in *The 2nd International Symposium on Experimental Robotics II*. London, UK: Springer-Verlag, 1993, pp. 317–328.

[9] B. Gerkey, R. Vaughan, K. Sty, A. Howard, G. Sukhatme, and M. Mataric, "Most valuable player: A robot device server for distributed control," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Wailea, Hawaii, October 2001.

[10] S. Fleury, M. Herrb, and R. Chatila, "Genom: a tool for the specification and the implementation of operating modules in a distributed robot architecture," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots & Systems (IROS)*, Grenoble, France, Septembre 1997, pp. 842–848.

[11] OROCOS project, "Orocos homepage." [Online]. Available: http://www.orocos.org/

[12] A. Lankenau and T. Rofer, "A versatile and safe mobility assistant," *IEEE Robotics & Automation Magazine*, vol. 8, no. 1, pp. 29–37, March 2001.

[13] E. Prassler, J. Scholz, and P. Fiorini, "A robotics wheelchair for crowded public environment," *IEEE Robotics & Automation Magazine*, vol. 8, no. 1, pp. 38–45, March 2001.

[14] G. Pires and U. Nunes, "A wheelchair steered through voice commands and assisted by a reactive fuzzy-logic controller," *International Journal of Intelligent and Robotic Systems*, vol. 34, no. 3, pp. 301–314, 2002.

[15] U. Nunes, J. Fonseca, L. Almeida, R. Araújo, and R. Maia, "Using distributed systems in real-time control of autonomous vehicles," in *Robotica*, vol. 21. Cambridge University Press, 2003, pp. 271–281.

[16] R. Maia, R. Cortesão, U. Nunes, V. Silva, and J. Fonseca, "Robust low-level motion control of WMR with stochastic active observers," in *International Conference on Advanced Robotics*, vol. 2, Coimbra, July 2003, pp. 876–882.

[17] Mori, *Range-Finder Type Laser Scanner URG-04LX Specifications*. Japan: Hokuyo Automatic Co., Ltd, July 2005.

[18] BOSH, *CAN Specification - Version 2.0*. Robert Bosch GmbH, September 1991.

[19] USB consortium, "Universal serial bus specification," USB consortium, Tech. Rep., April 2000.

[20] R. Maia, "Movimento de robôs móveis com rodas de tracção diferencial: Modelação e controlo do sistema motriz," Master's thesis, Universidade de Coimbra, 2004 (in portuguese).

[21] P. Sousa, R. Araújo, and U. Nunes, "Real-time labeling of places using support vector machines," in *2007 IEEE International Symposium on Industrial Electronics*, Vigo, Spain, June 2007.

[22] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Addison-Wesley, September 1993.

[23] Óscar Martínez Mozos, "Supervised learning of places from range data using boosting," Master's thesis, Albert-Ludwigs-Univ. Freiburg, 2004.

[24] S. Loncaric, "A survey of shape analysis techniques," *Pattern Recognition*, vol. 31, no. 8, pp. 983–1001, 1998.

[25] J. O'Rourke, *Computational Geometry in C (Second Edition)*. Cambridge University Press, September 1998.

[26] A. Young, *Handbook of Pattern Recognition and Image Processing*. Academic press, 1986.

[27] O. M. Mozos, C. Stachniss, and W. Burgard, "Supervised learning of places from range data using adaboost," in *IEEE International Conference Robotics and Automation*, Barcelona, Spain, April 2005, pp. 1742–1747.