escola superior de tecnologia e gestão
instituto politécnico de leiria

Projeto

Mestrado em Engenharia Eletrotécnica

AGV PARA AMBIENTE INTERIOR

Cristiano da Silva Justino

Leiria, Março de 2016

Projeto

Mestrado em Engenharia Eletrotécnica

AGV PARA AMBIENTE INTERIOR

Projeto

Cristiano da Silva Justino

Projeto de Mestrado realizado sob a orientação do Doutor Carlos Fernando Couceiro de Sousa Neves, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria e coorientação do Doutor Hugo Filipe Costelha de Castro, Professor da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria.

Leiria, Março de 2016

Esta página foi intencionalmente deixada em branco

Dedicatória

Para quem sabe esperar, tudo vem a tempo.

Clément Marot

Ao meu pai

Esta página foi intencionalmente deixada em branco

Agradecimentos

Este projeto simboliza o término de um ciclo: ciclo de estudo, ciclo da vida, ciclo de aprendizagem. A vida é feita deles, como é, também, pelas pessoas que cruzam o nosso caminho. Na verdade, a vida é feita de pessoas.

O curso de mestrado que agora concluo é um pequeno trajeto que me propus fazer com o intuito de aprender, dar um passo mais no caminho que é a vida.

Agradeço aos meus orientadores, Professor Doutor Carlos Neves e Professor Doutor Hugo Filipe Costelha de Castro, por todo o apoio, dedicação, paciência e ensinamentos durante o período de execução do projeto.

A todos os colegas de curso, em especial aos que partilharam longas horas de trabalho no laboratório de robótica, deixo também uma palavra de agradecimento, pelo contributo que deram em cada momento; sem esquecer os docentes do curso e colaboradores do Instituto Politécnico de Leiria que me auxiliaram em algum momento.

Por fim, agradecer a todos os meus amigos e família, em especial a minha mãe, por tudo quanto acrescentam à minha vida, cuja existência é essencial para o equilíbrio como ser humano.

Ao Universo, e a todos…

Gratidão!

Esta página foi intencionalmente deixada em branco

Resumo

O Nomad200 foi um dos robôs mais populares usados para a investigação em robótica móvel. Face à antiguidade destes robôs, a maioria da utilização dos mesmos hoje em dia implicou uma renovação do seu *hardware*. Este projeto de mestrado teve como motivação a reabilitação de um robô NOMAD200 obsoleto, tendo como objetivo a implementação de um robô capaz de navegar autonomamente em ambiente interior.

Foram estudados todos os componentes do robô e reutilizou-se todo o *hardware* funcional do robô, nomeadamente sensores, atuadores e estrutura mecânica. Foi desenvolvido *software* de baixo nível para controlar o *hardware* recuperado, bem como os novos componentes de *hardware* que foram integrados. A comunicação com o PC de alto nível é feito usando o protocolo *rosserial*. O sistema base de *software* de alto-nível do robô é o ROS, versão *Indigo*.

O sistema desenvolvido foi testado e a versão atual do robô permite realizar navegação com a utilização de um laser e odometria.

Palavras-chave: robótica móvel, Nomad 200, Synchro-drive, AGV

Esta página foi intencionalmente deixada em branco

Abstract

The Nomad 200 was among the most popular platforms used for research in mobile robotics. Given their old age, the use of these robots nowadays implied the rehabilitation of its hardware. This project's aim was the rehabilitation of an obsolete NOMAD200 robot, which goal is the implementation of an autonomous robot in an indoor environment.

All the robot's components were studied and all the functional *hardware* was reused, namely, sensors, actuators and mechanical structure. Low level software was developed to control all the recovered hardware and the new integrated *hardware*. Communication with the high-level computer is done using the *rosserial* protocol. The high-level software is based on ROS (Robot Operating System), Indigo version.

The developed system was tested in a real scenario, with the current version allowing for navigating based on a LASER range finder and odometry.

Keywords: mobile robotics, Nomad 200, Synchro-drive, AGV, Control

Esta página foi intencionalmente deixada em branco

Lista de figuras

Lista de tabelas

Esta página foi intencionalmente deixada em branco

Lista de siglas

Esta página foi intencionalmente deixada em branco

Índice

1. Introdução

A robótica móvel é, sem dúvida, uma área de estudo em grande crescimento na investigação científica, implementação industrial e comercial, ou mesmo em ambiente educativo, onde cada vez mais cedo os jovens têm acesso a equipamentos para trabalhar com pequenos robôs. De cariz interdisciplinar, abrange um leque diversificado de conhecimentos, como a eletrónica, a mecânica e também a informática, requerendo a assimilação de experiências em várias vertentes.

Na década de 90 assistiu-se a um crescimento no estudo desta área. Para isso contribuiu o aparecimento de soluções comerciais de robôs prontos a usar, com ferramentas para o seu estudo em meio académico. Um exemplo disso é a empresa norte-americana Nomadic Technologies, Inc, que desenvolveu e comercializou três robôs com essas características, o Nomad200, o Scout, o SuperScout, e o XR4000.

O Nomad200 foi considerado o melhor robô desta empresa [1] e é das plataformas mais utilizadas na investigação em robótica desde o seu lançamento no mercado [2]. Em várias instituições de investigação em robótica por todo o mundo existem robôs Nomad200 que são usados como plataforma de base para investigação e desenvolvimento de algoritmos. Exemplos disso são estudos em criação de mapas em tempo real e navegação [3] [4] [5] realizados até ao ano 2000, ano em que a Nomadic deixou de operar em robótica, após a aquisição desta empresa e alteração do ramo de atividade. Pela inexistência de suporte técnico e documentação, gradualmente estes robôs foram ficando obsoletos pela dificuldade em atualizar a capacidade processamento dos robôs.

Todavia, continuam a ser um recurso utilizado para desenvolvimento da robótica em diversos institutos e universidades que realizam investigação em Inteligência Artificial [6] [7], *software* para controlar robôs móveis e dotá-los de sensores [8] ou processamento de imagem aplicado a tarefas em robótica [9] [10]. Face à antiguidade destes robôs, a maioria da utilização dos mesmos hoje em dia implicou uma renovação do seu *hardware* [11].

1.1. *Motivação*

Este projeto de mestrado teve como motivação a reabilitação de um robô NOMAD200 obsoleto, tendo como objetivo a implementação de um robô capaz de navegar autonomamente em ambiente interior. Este trabalho aborda a utilização de robôs móveis na execução de tarefas de forma autónoma. São tidos em conta ambientes onde o robô tem de interagir com pessoas.

Posteriormente poderão ser adicionadas funcionalidades específicas, recorrendo a *software* de alto nível desenvolvido para cada aplicação, sendo a criação de um robô-guia um objetivo para um próximo projeto do Departamento de Engenharia Eletrotécnica da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Leiria (ESTG/IPLeiria).

1.2. *Objetivos do projeto*

Este projeto parte da necessidade e da oportunidade de reabilitar o robô móvel NOMAD 200 existente no laboratório de Robótica ESTG/IPLeiria, fabricado nos Estados Unidos da América pela Nomadic Technologies, Inc, e adquirido em 1998. Dada a sua idade e o rápido desenvolvimento tecnológico das últimas décadas, revelava-se imperativa a reabilitação do controlo deste robô. O objetivo é a sua utilização como robô guia ou de transporte. Pretende-se reutilizar todo o *hardware* funcional do robô, nomeadamente no que respeita a sensores, atuadores e estrutura mecânica. O robô deverá permitir, no futuro, a utilização de trabalhos desenvolvidos anteriormente no Laboratório [12] e adicionando um sensor LASER de distância para melhoria do desempenho do sistema de localização e de navegação. O sistema base de *software* de suporte ao robô será o ROS, versão *Indigo*.

Assim, como etapas gerais do projeto definiram-se inicialmente: Estudo do NOMAD 200 e avaliação do estado de conservação dos seus componentes; Projeto e desenvolvimento do *hardware* de controlo motriz; Implementação do algoritmo de controlo motriz, incluindo odometria; Desenvolvimento e implementação de uma interface amigável para o utilizador. No final do projeto o robô deve estar operacional e apto a, futuramente, integrar sistemas desenvolvidos na ESTG/IPLeiria para localização e navegação autónoma, com integração da câmara e do LASER.

O código desenvolvido no presente projeto é disponibilizado publicamente no github, em https://github.com/ipleiria-robotics/. Os ficheiros acessíveis contêm código do ROS/PC, ChipKIT e dos módulos QEI.

1.3. *Organização do documento*

O presente capítulo, de caráter introdutório, explica os objetivos do projeto, bem como a motivação para a sua execução. O capítulo 2 descreve o trabalho desenvolvido no estudo do robô existente inicialmente, com a análise aprofundada dos vários componentes do robô e do seu estado de funcionamento. Os trabalhos seguintes consistem na recuperação do robô cujo teor é descrito no capítulo 3, desde todo o *hardware* até ao *software*. Relativamente aos testes realizados e respetivos resultados, no capítulo 4 é feita a explicação das várias etapas com foco nos itens mais relevantes. Por fim, no capítulo 5 são apresentadas conclusões e linhas orientadoras para trabalhos futuros.

2. O robô móvel Nomad200

No presente capítulo é realizada uma análise ao robô Nomad200 do laboratório de robótica da ESTG/IPLeiria, indicando de forma detalhada o estado de conservação dos seus componentes à data de início deste trabalho, nomeadamente no que diz respeito a sensores, atuadores, controladores e estrutura mecânica.

Trata-se de um robô móvel, não holonómico [13], com configuração 'Syncho-Drive' (*synchronous drive robot*), Figura 1, com três graus de liberdade. O robô possui três rodas que funcionam de forma sincronizada, ligadas por um sistema mecânico de polias e correias, tanto em movimento de translação como em movimento de rotação. Apresenta como vantagem o facto de reduzir o escorregamento, proporcionando melhores estimativas odométricas; no entanto, a sua característica menos positiva é ser mecanicamente menos robusto, quando comparado com montagens mais simples como a diferencial, necessitando de realinhamentos e calibrações frequentes.

Figura 1 - O robô Nomad200.

A estrutura mecânica do Nomad200 baseia-se em duas partes acopladas por um rotor: a parte inferior é denominada de 'base' e a superior de 'torreta'. Na base do robô estão alojadas as placas de controlo de baixo nível cuja interligação com o processamento de baixo nível existente na torreta é feita por um conjunto de 24 condutores. Na Figura 2 é apresentado um diagrama de blocos estrutural que permite uma visualização geral da arquitetura do robô em termos de *hardware*.

A velocidade linear máxima indicada é de 24"/s, o que é equivalente a cerca de 0,61m/s, e como velocidade de rotação máxima o fabricante refere 60º/s, ou seja 1,047rad/s.

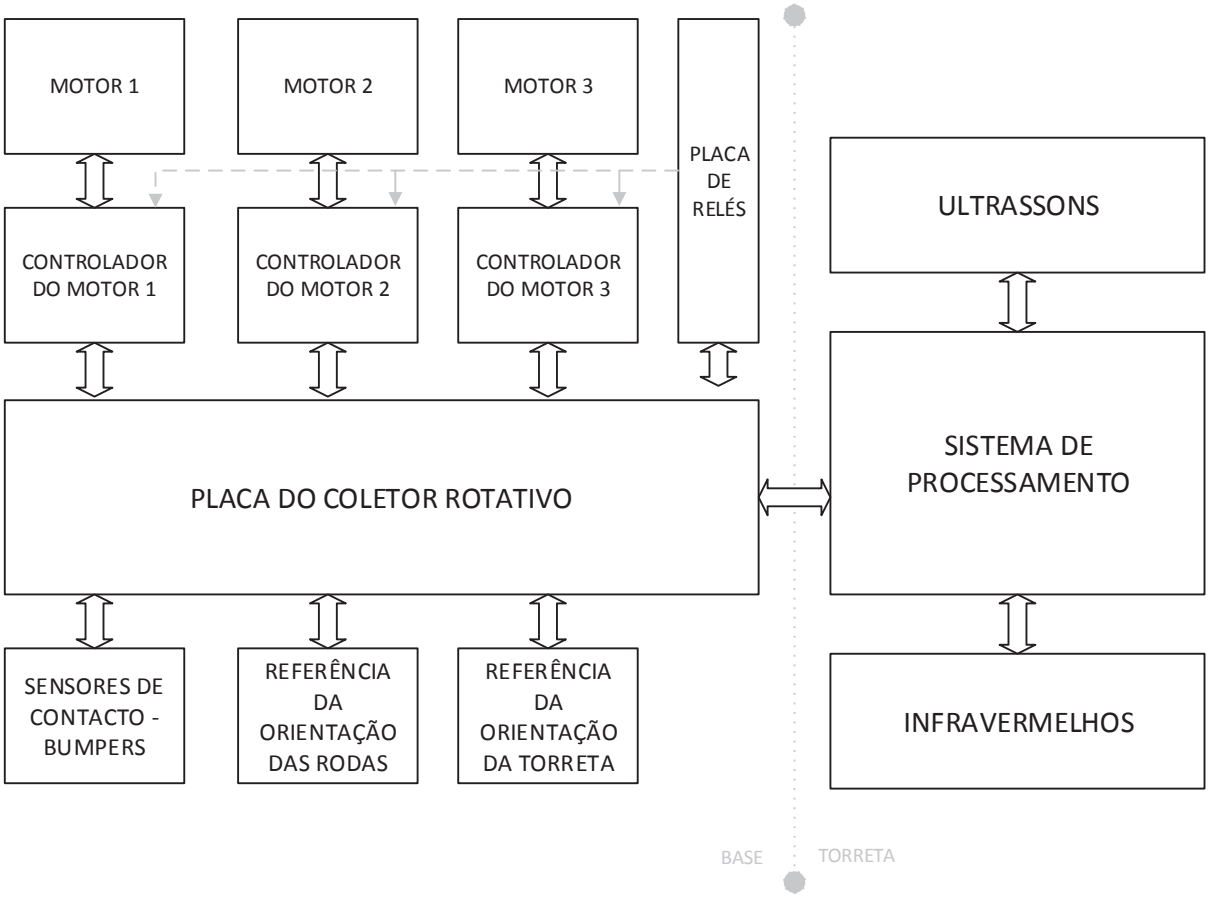


Figura 2 - Diagrama estrutural do robô.

Na base estão localizados os compartimentos para três baterias de 12V/12Ah, os controladores dos motores, os motores e os três conjuntos de correias: sincronismo de rotação das rodas, sincronismo de orientação das rodas e rotação da torreta. A cobrir todo este equipamento eletrónico e mecânico está uma grande saia metálica, Figura 3, onde estão instalados dois anéis de borracha, para-choques, com sensores de contacto – *bumpers*. Esta

saia pode ser içada e travada com um batente específico, permitindo aceder a todo o equipamento localizado na base, incluindo as baterias.

Figura 3 - Saia metálica da base do Nomad200.

Na torreta estão alojados vários componentes do robô, como sejam: um par de baterias, computadores, placas de processamento de sinais, placa de síntese de voz, placa de rede de dados e dois conjuntos de sensores para deteção de obstáculos e medição de distâncias, com as respetivas placas de controlo.

2.1. *Placa de relés*

A placa de relés existente no robô é denominada, na documentação original,"*Relay Board*", Figura 4, e aloja um relé (K1) controlado pela CPU que permite o corte geral da alimentação 24V, exclusiva ao funcionamento dos motores. É nesta placa que são ligadas as três baterias (12V/12AH) da parte inferior do robô, das quais duas são ligadas em série para obtenção da tensão de funcionamento dos motores e a terceira bateria destina-se à alimentação da CPU – em conjunto com duas outras baterias alojadas na parte superior do robô. É possível ver que nesta placa não vêm implementadas todas as funcionalidades disponíveis originalmente [14], eventualmente por se tratar de funções opcionais. Pela análise da placa obtém-se o esquema detalhado apresentado no Anexo A.

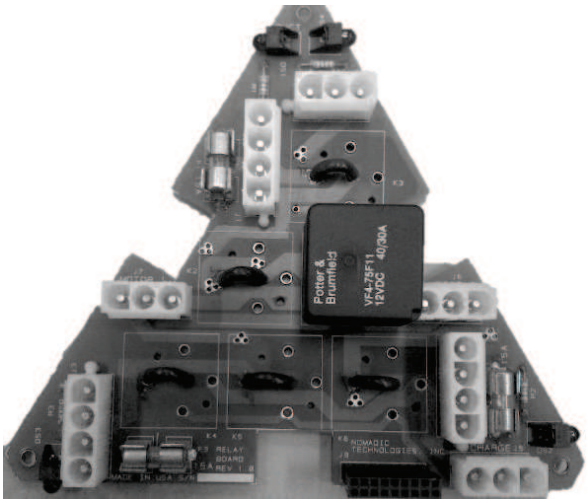


Figura 4 - Placa de relés - Relay Board.

Além de fusíveis de proteção para cada bateria, existe também um *LED* indicador da correta alimentação. Esta placa está ligada à placa do coletor rotativo, analisada de seguida, por um conector (J9) de 18 terminais.

2.2. *Placa do coletor rotativo*

A placa do coletor rotativo, originalmente denominada "*Slipring Board*", realiza a interface alto nível - baixo nível, recebendo os sinais de controlo da parte superior do Nomad200 e ligando os vários dispositivos de baixo nível existentes na parte inferior do robô (Figura 5). Como já referido anteriormente, está também ligada à placa de relés por um conector de 18 terminais.

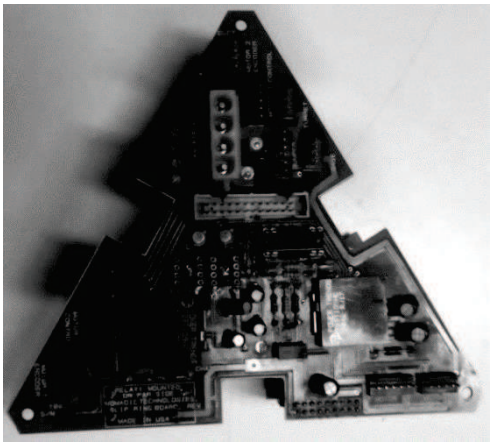


Figura 5 - Placa do coletor rotativo.

Por este conector entram as tensões 12V e 24V, alimentando dois reguladores de tensão de 5V (1,5A). Os dois reguladores são dedicados: um para tarefas de controlo, referido como "Chassi", e outro para alimentação dos controladores dos motores. Assim, verifica-se a existência de alimentações separadas para circuitos de potência e controlo.

A verificação do nível de carga das baterias para os 24V é realizada por intermédio de um amplificador de isolamento (Burr-Brown ISO122) que consiste em dois amplificadores onde o sinal é transmitido através de uma barreira capacitiva, com alimentações independentes, conforme o esquema representado na Figura 6. Para obter os valores de tensão negativa recorre-se a dois conversores de tensão LMC7660.

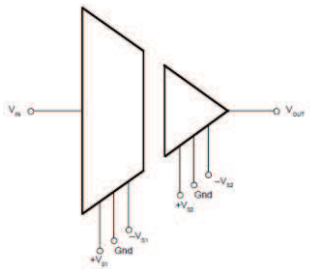


Figura 6 - Amplificador de isolamento.

Nesta placa são ligados os sinais de posição zero da torreta, posição zero da orientação das rodas do robô, os sinais dos sensores *bumper*, e os sinais de controlo dos motores com os respetivos codificadores.

É a partir do conector central que se estabelece a ligação com a CPU existente na torreta, sendo o seu diagrama de terminais apresentado na Figura 7.

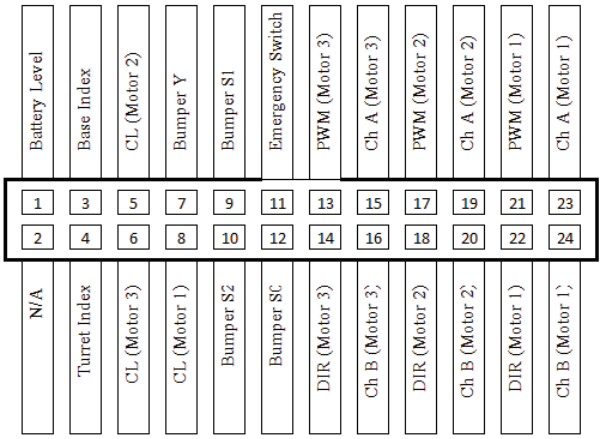


Figura 7 - Diagrama de terminais do coletor rotativo.

2.3. *Controladores dos motores*

Os controladores de motores utilizados são da *Nomadic*, denominados SERVO-AMP (Figura 8). Não existindo qualquer referência a estes controladores no manual do robô, o trabalho passou por investigar a documentação referente a trabalhos já realizados, no sentido de obter informação que pudesse contribuir para o entendimento do seu funcionamento. Além disso, foi imprescindível analisar detalhadamente cada um dos três controladores.

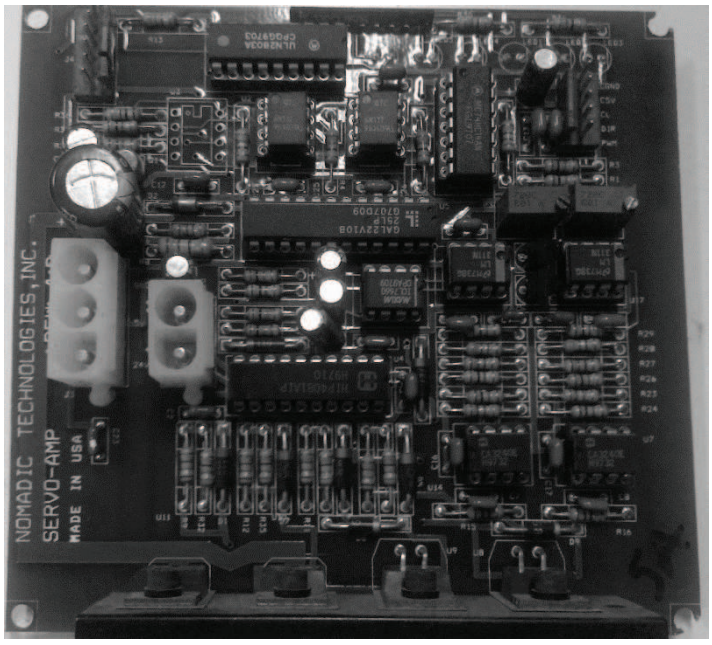


Figura 8 - Controlador de motor SERVO-AMP.

Os sinais de controlo que são provenientes da CPU descem à placa do coletor rotativo que os encaminha para cada um dos controladores dos motores. No conector J1 são recebidos os sinais de PWM, DIR e CL, além da alimentação de controlo C5V e CGND. Cada um destes sinais é invertido e opto-isolado, com a utilização de dois circuitos integrados: *Schmitt Trigger* inversor (MC74HC14A) e opto-acoplador(6N137). A necessidade de entender o funcionamento do controlador para a sua possível utilização levou à análise pormenorizada do circuito e permitiu o levantamento do respetivo esquema (Anexo B)

Foram realizados testes aos controladores, aplicando um sinal em PWM e um sinal de direção. Para os testes utilizou-se um sinal PWM com uma frequência de 23.4kHz [2], com fator de ciclo variável. Verificou-se que os controladores não estavam operacionais: apesar dos sinais chegarem à entrada das PAL, não se observaram variações nas suas saídas e, consequentemente, não geraram rotação nos motores. Verificou-se um sobreaquecimento deste componente em dois dos três controladores. Ainda se tentou fazer uma leitura destes semicondutores para eventual diagnóstico, mas, por ser um componente obsoleto (GAL22V10, descontinuadas desde 2010 pela *Lattice – Semiconductor Corporation* [15]) tal não foi possível. Conclui-se que dos três controladores nenhum estava em funcionamento.

2.4. *Motores*

O Nomad200 dispõe de três motores para controlo de cada um dos seus três graus de liberdade. São servo motores DC, da *Pittman*, com tensão nominal de 24V. O motor para o movimento de translação (velocidade linear) do robô tem uma caixa de redução de 4:1, referência GM14606C831-R1, do qual foi obtida a ficha técnica contactando o fabricante. Realizados os testes ao motor, comprovou-se o seu bom funcionamento.

O movimento de rotação do robô (velocidade angular) é conseguido utilizando um servo motor DC, referência GM9434J173-R1, com caixa de redução de 19.7:1. Todos os dados técnicos deste modelo encontram-se disponíveis na página *web* do fabricante[1]. O movimento de rotação da torreta é realizado com um servo motor DC igual. Na Tabela 1 são apresentadas as principais características dos motores presentes no Nomad200.

[1]www.pittmannet.com

		Motor 1 Translação	Motor 2 Rotação das Rodas	Motor 3 Rotação da Torreta
Característica	**Unid.**	GM14606C831-R1	GM9434J173-R1	GM9434J173-R1
Tensão Nominal	V	24	24	24
Corrente em Vazio	A	0,26	0,16	0,16
Binário Carga	Nm	0,25775	0,0431	0,0431
Velocidade em Vazio	rpm	3216	6151	6151
Relação da caixa		4:1	19,7:1	19,7:1

Tabela 1 - Características principais dos motores.

Os testes realizados comprovam que o motor de rotação do robô estava funcional, porém o último motor apresentava problemas. Após algumas diligências no sentido da sua eventual reparação, tal não foi possível, impedindo assim, a sua reutilização.

Para determinar o consumo real dos motores foram realizados ensaios com o rotor bloqueado e com o sistema mecânico do robô de rodas em vazio. Os valores medidos são apresentados na Tabela 2:

Motor	Rotor bloqueado	Sistema mecânico
GM14606C831-R1	> 12 A	2,2 A
GM9434J173-R1	5,11 A	0,5 A

Tabela 2 - Correntes dos motores.

A fonte de alimentação disponível para o teste com rotor bloqueado para o motor de 4:1 apenas permitiu medir até 12A, sendo o consumo máximo teórico de 28 A, segundo o fabricante.

Ambos os motores estão equipados com codificadores (*encoders*) incrementais de 500 pulsos por volta, HEDS-9100 – A00 da *Avago Technologies* (ver Figura 9), com saída dos canais A e B e alimentação de a 5V.

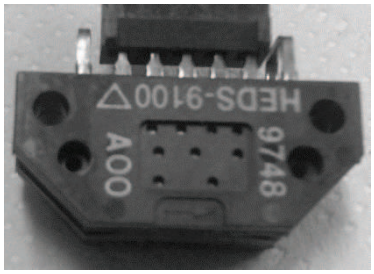
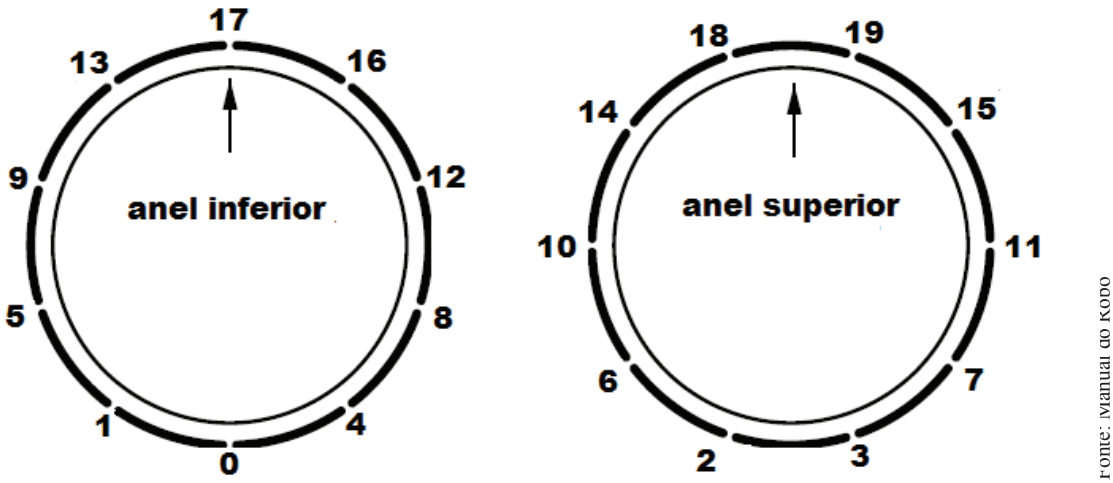


Figura 9 - Encoder HEDS-9100.

2.5. *Sensores de contacto - Bumpers*

O Nomad200 vem equipado com sensores de contacto instalados nos anéis de borracha (para-choques). No manual do robô são denominados *bumpers*, os quais estão dispostos em dois anéis de dez sensores, incorporados num revestimento de borracha, como mostra a Figura 10. Estes anéis de borracha apresentavam deficiências mecânicas nas extremidades, resultando na má fixação dos anéis na saia metálica. Consequentemente os contactos por vezes não eram atuados. No interior da estrutura metálica de suporte está colocada uma placa de circuito impresso com um circuito de controlo, onde ligam os sensores (Figura 11), que depois interliga com a placa central da base do robô.



Fonte: Manual do Robô

Figura 10 - Distribuição dos contactos – Bumpers.

O estudo desta placa de controlo permitiu obter o respetivo esquema elétrico e também perceber o seu funcionamento para posterior utilização (Anexo C).

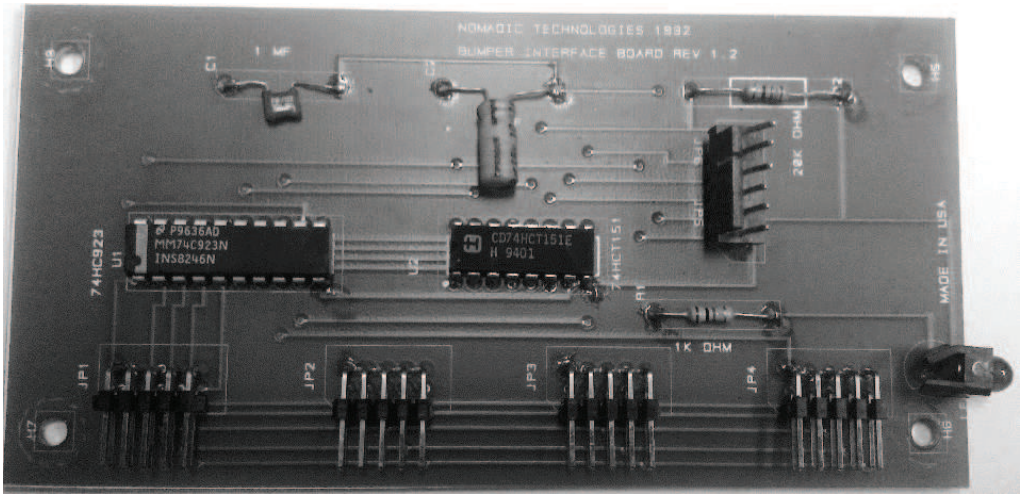


Figura 11 - Controlador de contactos – Bumpers.

A análise desta placa evidencia a existência de um circuito composto por um codificador para teclado matricial, de 20 contactos, com saída de "*Data Available*", e um multiplexador de oito entradas. As entradas I0 a I4 recebem os dados do codificador (Data Out de A a E) e a entrada I7 recebe o sinal "*Data Available*".

S2	S1	S0	Y
L	L	L	*Data Out A*
L	L	H	*Data Out B*
L	H	L	*Data Out C*
L	H	H	*Data Out D*
H	L	L	*Data Out E*
H	L	H	*N/A*
H	H	L	*N/A*
H	H	H	*Data Available*

Tabela 3 - Tabela de verdade do multiplexador.

A saída Y do multiplexador é enviada ao controlo de baixo nível do robô, que deverá manter o multiplexador com as entradas a S0 a S2 asseridas a valor lógico alto, de forma a obter na saída deste componente o sinal de "*Data Available*" do codificador. Isto permite saber quando um dos contactos foi pressionado, podendo assim reagir de imediato, por exemplo, pela imobilização imediata do robô, para de seguida fazer um varrimento dos *bumpers* e saber qual deles foi atuado. A leitura de cada sensor é realizada mediante a combinação lógica dos sinais S0 a S2 aplicados ao multiplexador 74HCT151N (Tabela 3), em que cada combinação permite ler a informação "*Data Out*" do codificador, de A a E, sendo que cada combinação é única e corresponde a um contacto numerado conforme indicado no Anexo D. Este circuito é alimentado a partir dos 5V do *chassi*.

2.6. *Contactos de referência*

O Nomad 200 tem um contacto instalado para *reset*/referência da orientação das rodas do robô e outro para *reset*/referência da orientação da torreta, Figura 12. A placa designa-se "*Slotted Switch Board REV1.0",* e o circuito é baseado num comutador fotológico OPB960, da "*TT electronics*".

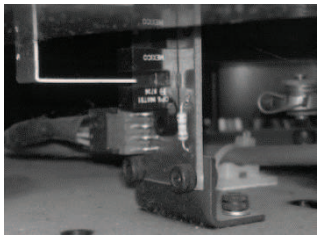


Figura 12 - Placa de leitura de posição de referência.

A saída é asserida a zero, sendo, portanto, um sinal de nível lógico baixo que indica a posição zero das rotações do robô, posição 'Home'.

2.7. *Sensores de ultrassons*

Para o controlo dos sensores de ultrassons é usada uma placa de circuito impresso de fabrico da *Nomadic Technologies* denominada de "*Sonar Ranging Board (VER 3.0)*", no manual referida como "*Sensus 200*" (ver Figura 13). Com alimentação a 12V e um regulador de tensão L7805C, o circuito funciona a 5V e consiste num sistema multiplexador de 16 canais, e um módulo controlador de sensor de ultrassons, denominado"*6500 Ranging Module*", da *SensComp, Inc*.

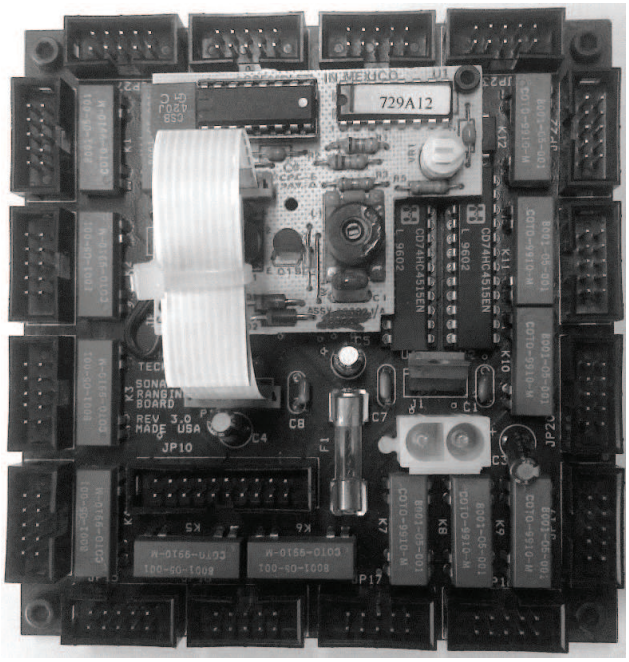


Figura 13 - Controlador dos sensores ultrassons.

A seleção do sensor que efetua a leitura é conseguida pelos sinais provenientes do sistema de processamento, que são recebidos através do conector JP10, cujo diagrama de terminais é apresentado na Figura 14 (desenho esquemático simplificado em Anexo E).

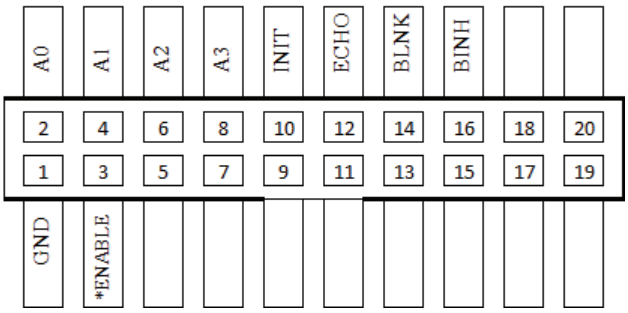


Figura 14 - Diagrama de terminais da controladora dos sensores ultrassons.

Verificou-se que a placa tinha em falta o circuito integrado de controlo do sensor (TL851 - *sonar ranging control*). Reposto este semicondutor, todas as seleções possíveis do multiplexador foram testadas, verificando-se o seu bom funcionamento. Foram também testados todos os sensores (Figura 15), recorrendo ao método de teste do fabricante [16], *SensComp* e, à exceção de um sensor que se encontra danificado mecanicamente, todos os sensores funcionaram na perfeição. O tempo necessário para cada medição é de 2,38ms, no mínimo, e pode demorar até 38,2ms, o que corresponde aos intervalos de medição de 0,4m e 6,5m, respetivamente.

Figura 15 - Transdutor de ultrassons.

2.8. *Sensores de infravermelhos*

A placa "*Sensus 300*", ilustrada na Figura 16, é denominada por "*A/D Board Rev1.0*" no manual do robô, tendo como função controlar os sensores de infravermelhos.

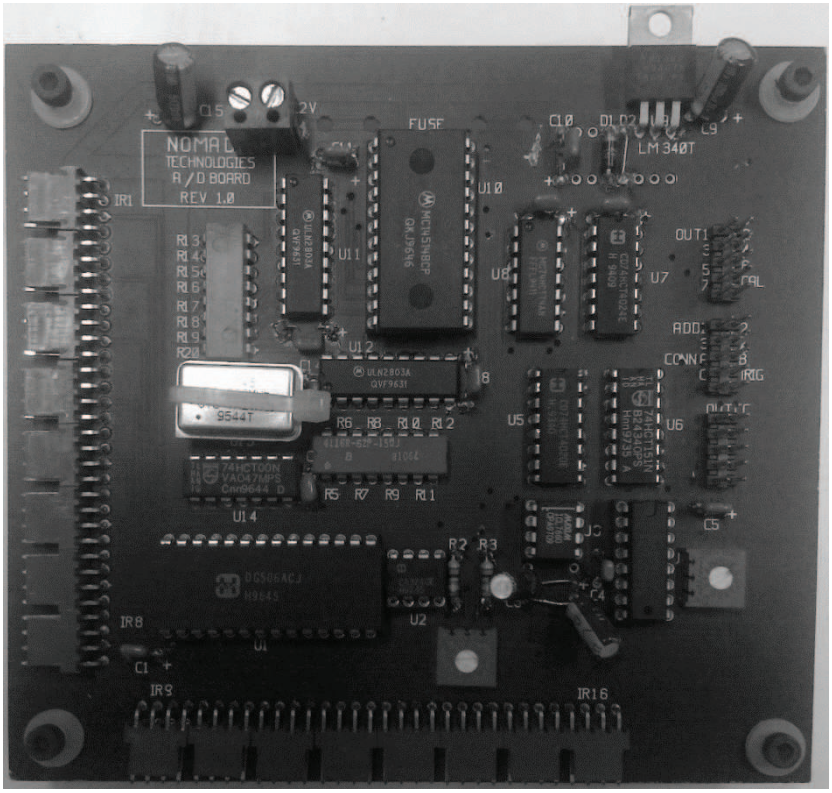


Figura 16 - Controladora dos sensores infravermelhos.

Cada sensor é um conjunto composto por um foto-díodo (ao centro) e dois *LED* de infravermelhos, montados num suporte com um conector de 5 terminais, Figura 17.

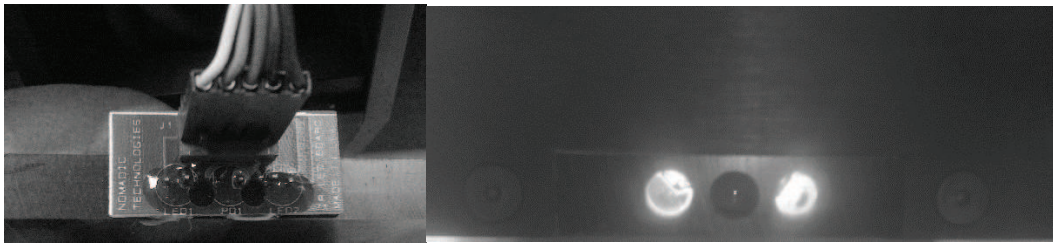


Figura 17 - Sensor de infravermelhos.

A placa foi analisada de forma detalhada, no sentido de perceber o seu funcionamento, para se poder realizar o seu controlo. O esquema elétrico completo encontra-se em Anexo F. O diagrama simplificado é apresentado na Figura 18.

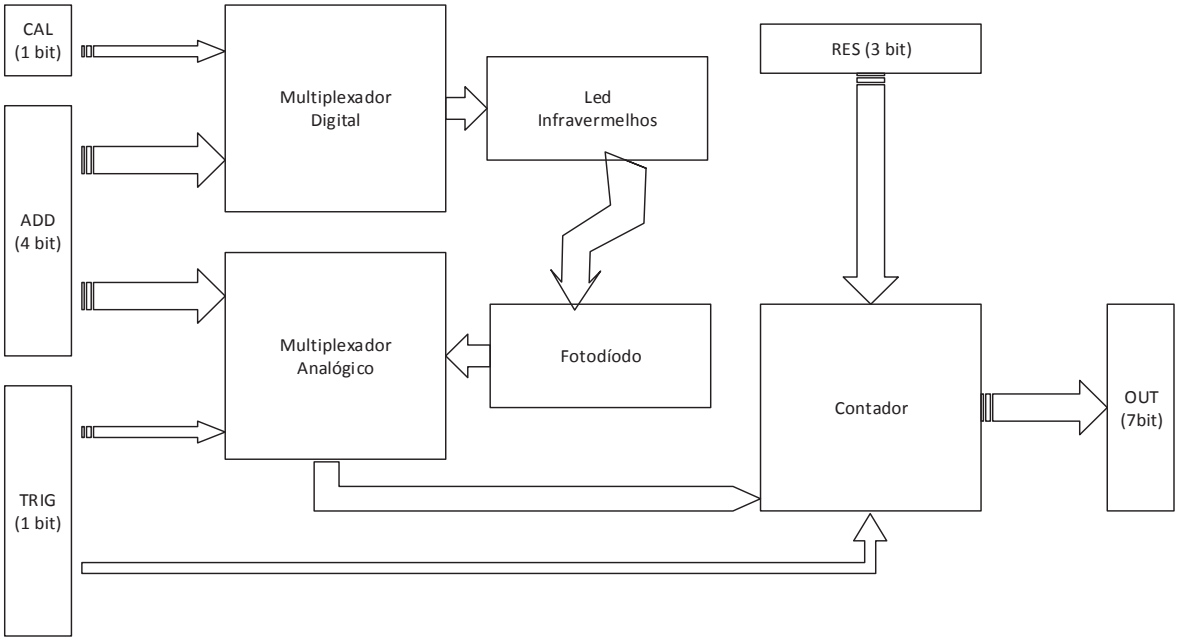


Figura 18 - Diagrama simplificado da placa de controlo dos sensores infravermelhos.

A entrada ADD é de 4 bits, utilizados para multiplexar os 16 sensores do robô. Esta multiplexagem (MC14514) é realizada para os *LED* emissores de infravermelhos, bem como para os foto-díodos, utilizando um multiplexador analógico (DG506A). Existem ainda 3 bits que permitem definir a resolução das leituras efetuadas.

O sinal de TRIG permite iniciar o ciclo de leitura, sendo que deve ser realizada uma primeira leitura com os *LED* apagados, lendo o nível de infravermelhos do ambiente. De seguida é feita uma leitura com os LED acesos, asserindo o sinal CAL, para obter o reflexo de infravermelhos que são proporcionais à distância medida. Seguidamente, subtraem-se os dois valores para obter o valor de contagem que representa a distância medida, cujo valor em metros será obtido aplicando uma expressão de conversão, consoante a resolução utilizada. A saída do circuito é composta por 7 bits e é necessário realizar a contagem de pulsos, de forma a contabilizar o número total, proporcional à distância ao obstáculo mais próximo.

Asserindo o sinal de TRIG (disparo) é iniciado o ciclo de leitura, durante o qual fica asserido o sinal disponível "OUT CC". Neste tempo, o valor de tensão do foto-díodo é amplificado e convertido em frequência com uso de um VCO (74LS629), incrementando um contador de 14 etapas (74HC4020). Os 8 bits mais significativos da saída deste contador são ligados na entrada do multiplexador (74HCT151) que permite a seleção da resolução. Ao colocar a saída com valor lógico alto, este multiplexador faz parar a contagem de outro contador (74HC4024), de 7 etapas, que iniciou a contagem no momento de disparo. Terminado o ciclo de leitura, o sinal "OUT CC" fica com nível lógico zero, para que o sinal TRIG possa voltar a ser colocado a nível lógico zero.

Para perceber o resultado obtido, foram realizados ensaios com um analisador lógico com um ambiente "fechado", igual para todos os testes: um pequeno túnel branco, com um obstáculo de cor branca colocado a determinadas distâncias – 40cm, 30cm, 20cm, 10cm e 4cm. Para cada distância foram efetuadas leituras com emissores ligados e com emissores desligados e, para cada uma das condições, foram realizadas três leituras diferentes. Como o oscilador existente na placa é de 50 MHz, e dado o analisador lógico utilizado ter amostragem máxima a 24MHz, foi necessário alterar o oscilador para 10MHz de forma a conseguir uma correta amostragem dos sinais. Dos 240 ficheiros de dados (csv) exportados do *software* do analisador lógico, foi compilada a informação para obter uma relação entre distâncias medidas e tempo de leitura para cada resolução. No gráfico, Figura 19, são apresentados os resultados compilados, para a resolução '100' (quatro, numa escala de zero a sete, resolução mínima e máxima, respetivamente).

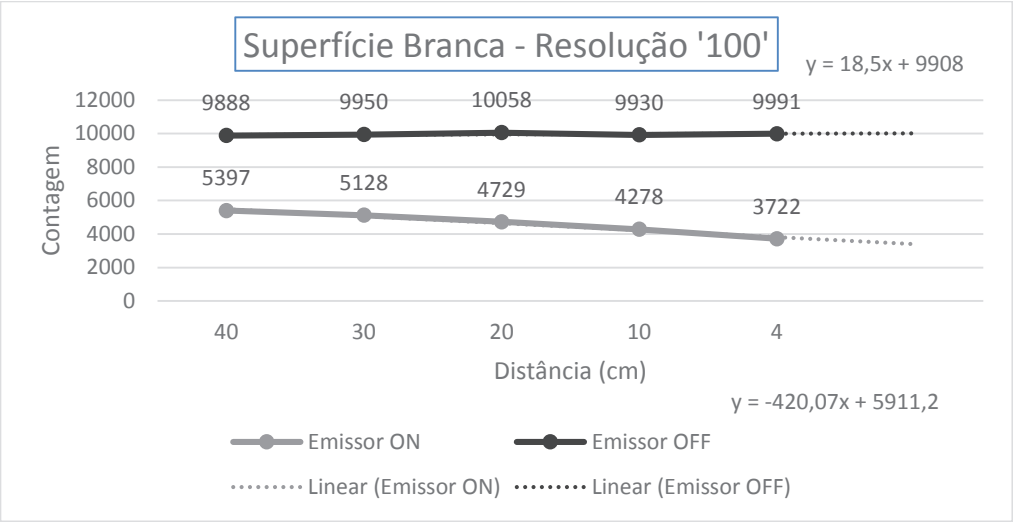


Figura 19 - Leituras de sensor de infravermelho.

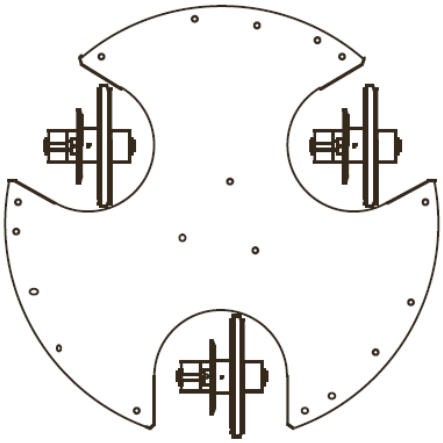
No Anexo G são apresentados os dados recolhidos para todas as resoluções e condições dos ensaios realizados. Foi possível testar e aferir do bom funcionamento de todos os sensores. No capítulo 4 são descritos os dados relativos aos testes realizados com os sensores de infravermelhos.

2.9. *Baterias*

O robô móvel em análise tem como fonte de energia dois conjuntos de baterias que alimentam dois circuitos distintos. Os motores e seus controladores são alimentados a 24V por duas baterias, 12V/12Ah, com dimensões aproximadas de 100 x 151 x 98 [mm], que se encontram alojadas na base do robô e que são ligadas em série através da placa de relés abordada na Secção 2.1 . É também nesta placa que é ligada a terceira bateria, com as mesmas características das anteriores, que alimenta a CPU em conjunto com outras duas baterias alojadas na torreta, 12V 17Ah com dimensões aproximadas de 167 x 181 x 72 [mm]. A alimentação a 12V é interligada da base para a torreta através de um anel de escovas, sendo concentrada numa placa (‘*Power Distribution Board’*) e depois distribuída através da placa de alimentação, denominada ‘*Power Management Board’*. Este conjunto de baterias fornece corrente para a CPU e para os circuitos sensoriais existentes na torreta. Segundo indica o manual, cada conjunto de baterias permite a utilização do robô durante aproximadamente 7 horas, com 288 Wh para os motores e 552 Wh para os computadores e sensores, totalizando 840Wh.

2.10. *Rodas*

O robô é suportado por três rodas idênticas dispostas de forma simétrica, como mostra a Figura 20. Estas funcionam de forma solidária pela sua ligação mecânica composta por engrenagens e correias. A energia é, assim, transmitida de igual forma, desde o motor até cada uma das rodas.



Fonte: Manual do Robô

Figura 20 - Disposição das rodas do robô.

Todas as rodas são constituídas por discos metálicos de 14mm de espessura e 115mm de diâmetro, aproximadamente, e cada um deles possuem um anel de material plástico totalizando assim um diâmetro de 124,6mm, Figura 21. Os referidos anéis encontravam-se deformados pela permanência na mesma posição, suportando o peso de todo o robô por largos períodos de tempo.

Figura 21 - Roda metálica com anel plástico.

2.11. *Sistema de processamento*

O processamento no *Nomad200* era feito originalmente com a utilização de um computador baseado num par de processadores Intel Pentium MMX, com ligações em barramento ISA, e com controladores Motorola, MC68HC11F1 a 16MHz para a interface dos sensores. O controlo dos motores é realizado pela utilização de um processador Motorola 68008/ASIC para controlo de três eixos. Todos os periféricos presentes utilizavam o barramento ISA.

O robô vem equipado com uma placa para realizar *text-to-speech (TTS)*, *DoubleTalk PC* da *RC Systems*, com um microprocessador 80C188EB de 10MHz [17], com dois conetores áudio para ligação do altifalante e respetivo regulador de volume.

Para comunicar com computadores externos ao robô, era disponibilizada uma placa de rede sem fios, fabricada pela *Proxim*, utilizando um protocolo proprietário denominado *RangeLAN2*, com largura de banda para transmissão de dados a 1.6Mbit/s [18]. A antena era externa e colocada no topo do robô.

2.12. *Modelo Cinemático*

Na presente secção é abordada a cinemática do robô tendo em conta a sua estrutura mecânica. O sistema é representado assumindo que o robô é rígido, que as rodas não são deformáveis e que se movem num plano horizontal.

Arbitrariamente, é atribuído um referencial base fixado no plano *b*, enquanto outro referencial é fixado no robô móvel *m*. A pose do robô pode ser descrita em função das coordenadas x,y da origem P do referencial móvel e pelo ângulo de orientação θ do referencial móvel, ambos relativamente ao referencial de base com origem em O, Figura 22.

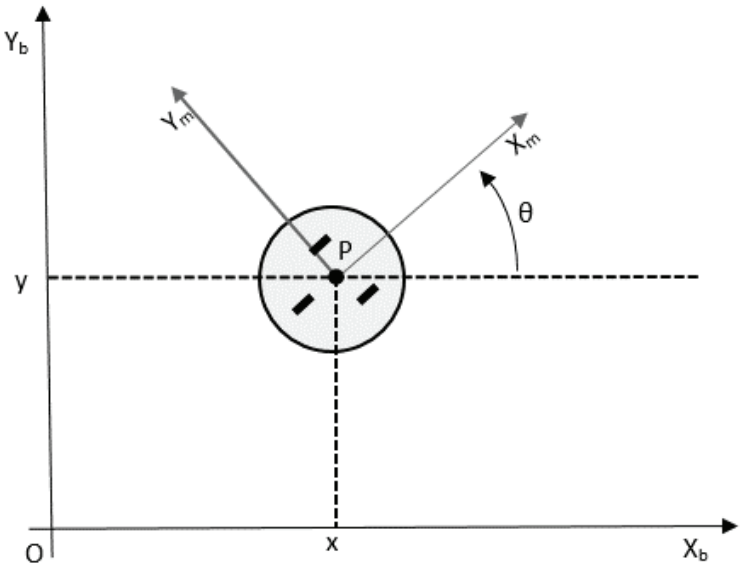


Figura 22 - Referenciais e pose do robô.

Desta forma, a pose do robô é expressa pelo vetor

$$q = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$$

e a matriz de rotação que expressa a orientação do referencial de base relativamente ao referencial móvel é

$$R(\vartheta) = \begin{bmatrix} \cos\vartheta & \sin\vartheta & 0 \\ -\sin\vartheta & \cos\vartheta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

É assumido que o plano de cada roda, durante o movimento, permanece vertical e que a sua rotação acontece em torno do seu eixo, no plano horizontal, sem atrito. É também assumido que o ponto de contacto das rodas com o pavimento é reduzido a um único ponto do plano, sem escorregamento [19].

$$\dot{q} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\vartheta} \end{bmatrix} = \begin{bmatrix} \sin\vartheta \\ \cos\vartheta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega$$

Portanto, o robô em estudo permite realiza o controlo independente nas componentes linear e angular da velocidade.

3. Recuperação do Nomad200

Neste capítulo são abordadas as várias questões relacionadas com o trabalho de reabilitação do robô: elementos mecânicos, estruturação dos circuitos, aspetos técnicos específicos e, também, todo o *firmware* e *software* desenvolvido.

Tendo em conta o *hardware* passível de ser reutilizado, foi definida uma nova estrutura para prosseguir com a recuperação do Nomad200. Na Figura 23 é apresentado o diagrama funcional do AGV.

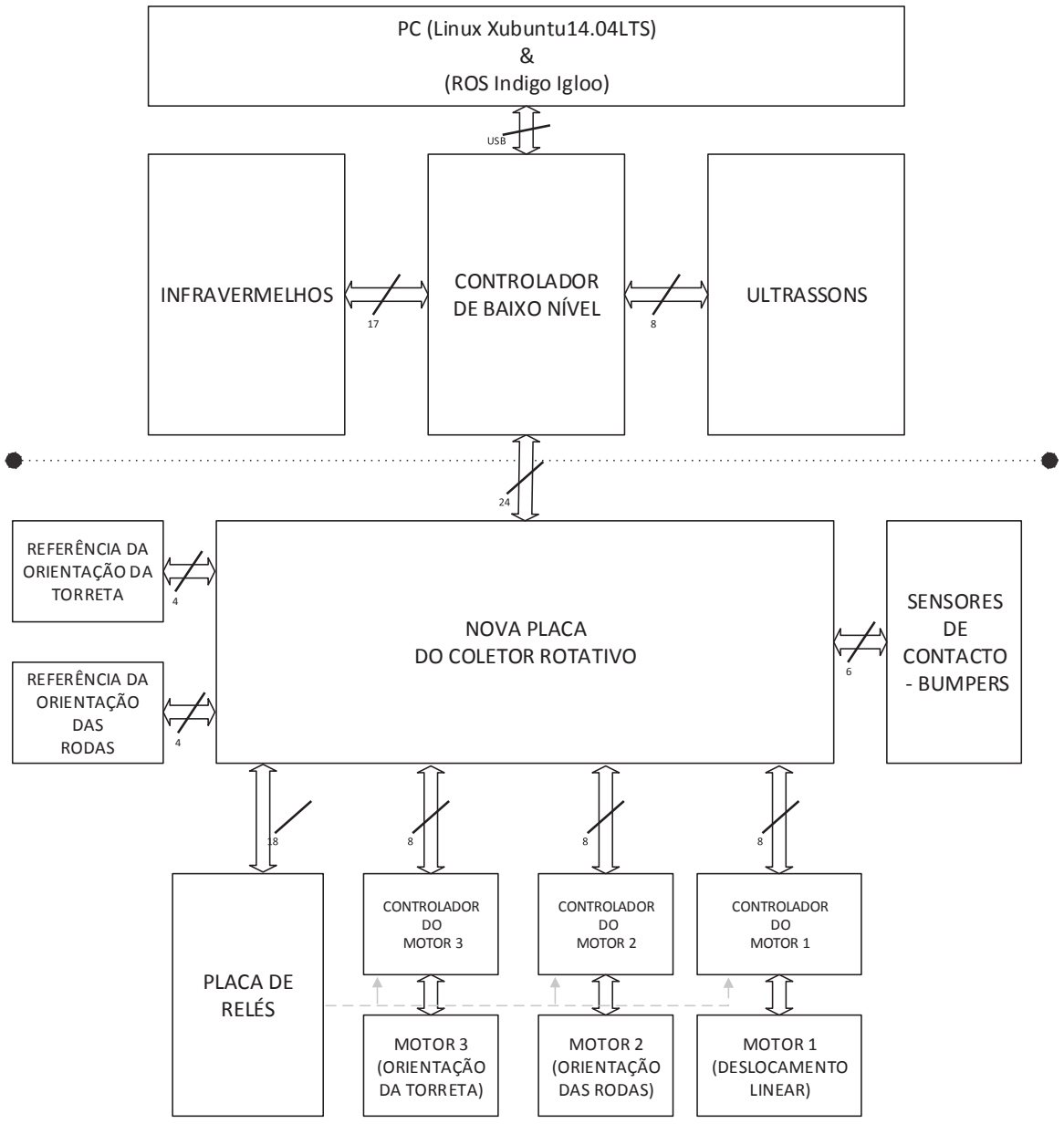


Figura 23 - Diagrama funcional do AGV.

Para o novo AGV é utilizado todo o *hardware* funcional, começando pela estrutura mecânica (*chassi*), rodas, motores, sensores e cablagem instalados. Conforme analisado no capítulo anterior, foram necessárias reparações menores, como a dos anéis plásticos de revestimento das rodas que se encontravam deformados. Estes anéis foram substituídos por três novos anéis, Figura 24, de material equivalente e, após realizado o alinhamento das rodas, o sistema mecânico para o movimento do robô ficou apto para o correto funcionamento.

Figura 24 - Roda metálica com novo anel plástico.

Também os anéis de borracha dos sensores de contacto – *bumpers* – instalados na saia do robô foram reparados por apresentarem deficiências mecânicas, corrigindo as extremidades e a respetiva fixação ao robô. Foi substituído o *LED* da placa e conferido o bom funcionamento deste conjunto de sensores, sendo a sua reutilização possível. No que diz respeito aos sensores ultrassom, procedeu-se no sentido de substituir o que se encontrava danificado, mas tal não foi possível até ao término deste projeto, por razões logísticas e burocráticas. Os sensores de infravermelhos, conforme referido no capítulo anterior, estão todos operacionais e são usados com a respetiva placa de controlo.

3.1. *Placa de relés*

Verifica-se que esta placa tem circuitos para relés que se encontram ponteados (*shunt*), que podem ser instalados para realizar o carregamento das baterias sem haver necessidade de as retirar da base do robô. Aplicando 12V no terminal 1 do conector "*Charge*", os relés K2 a K6 são atuados, realizando a comutação que isola as baterias do restante circuito, permitindo assim fornecer 12V a cada bateria através do terminal 3 do conector "*Charge*".

Assim, foram instalados os relés K2 a K6 e testada a placa para verificar o funcionamento do carregamento de baterias no local. Após confirmar o seu funcionamento correto, foi desenvolvida uma pequena placa - denominada "*Charger*" - para a ligação da fonte de tensão, com o objetivo de carregar as baterias com o robô ligado a uma fonte de tensão externa de 12V, Figura 25. Para tal, a placa inclui botões que auxiliam no início do carregamento, sendo possível ver o seu esquema no Anexo H.

Figura 25 - Placa 'Charger' montada no robô.

Esta funcionalidade fica acessível por um pequeno painel, instalado entre a base e a torreta, onde estão alojados os dois botões de pressão, os dois *LED* e o conector para a alimentação a 12V.

3.2. *Controladores de motores*

Os controladores do Nomad200, conforme indicado no segundo capítulo, não estão funcionais e, sendo impossível recuperar os dispositivos lógicos PAL que processam os sinais de entrada, considerou-se que a melhor solução para otimizar o tempo de

desenvolvimento do trabalho neste robô seria substituir os controladores, adquirindo um controlador robusto e fiável que permitisse controlar cada motor.

Foram comparados vários controladores e, considerando as características técnicas, preço e dimensões — optou-se pelo controlador da *'Maxon'*, modelo '*ESCON Module 50/5*', Figura 26, configurável e parametrizável por porta USB, possibilitando a visualização do estado do controlador em tempo real com o *software* gráfico do fabricante.

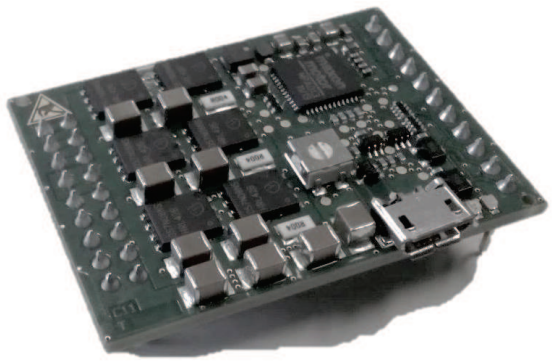


Figura 26 - Controlador 'Maxon - ESCON Module 50/5.

Este controlador em entradas para sinais de codificadores incrementais, sensores de efeito de *Hall*, duas entradas digitais, duas entradas analógicas e duas saídas analógicas. Por fim, possibilita configurar como entrada ou saída dois sinais adicionais. No presente projeto são utilizadas todas as entradas e saídas digitais, saídas analógicas e as configurações são para uso do controlador com motor DC com codificadores incrementais, descritos na Secção 2.4. Os dados de configuração para os controladores podem ser consultados no Anexo I.

O controlador é de dimensões reduzidas (43.18 x 31.75 x 5.8 [mm]); no entanto, é necessário criar uma placa mãe (*motherboard*) para o alojar, provendo-a de conectores para a ligação do motor e componentes de filtragem. O fabricante deste controlador tem também uma versão pronta a usar, mais cara, que se revela incompatível com o robô em questão, tendo em conta as suas grandes dimensões. Assim, foi desenhada e produzida a placa mãe, tendo em conta as especificações da '*Maxon'* [20], apresentada na Figura 27. O seu desenho foi feito para que as dimensões sejam idênticas aos controladores originais da '*Nomadic'* de forma a aproveitar os pontos de fixação no *chassi* e o espaço disponível.

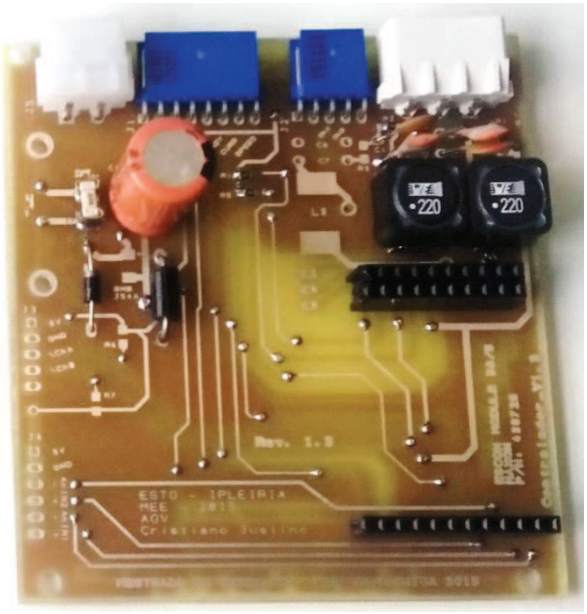


Figura 27 - Placa mãe para o controlador Maxon.

Esta placa mãe está desenhada para utilizar todas as funcionalidades do controlador escolhido em possíveis futuras utilizações no IPLeiria, sendo no presente projeto usado para controlo de motores DC com codificadores incrementais. Com o desenho criado, Anexo J, para este controlador pretendeu-se otimizar o espaço existente no robô e deixar os controladores acessíveis, sendo possível visualizar os *LED* de indicação de estado e também conectar o cabo USB para parametrização e monitorização dos controladores.

3.3. *Nova Placa do coletor rotativo*

Pela utilização de novos controladores com diferentes sinais de controlo foi imperativo introduzir alterações no que concerne à placa que recebe os sinais vindos da torreta. A nova placa tem, à partida, dois requisitos fundamentais: manter o número de sinais que ligam à torreta e manter as dimensões da placa original. Assim, foi desenhada a placa com o mesmo formato triangular, tendo sido necessário garantir a mesma posição do conector para a '*relay-board'* e os furos de fixação no *chassi*. Também foram mantidos o conector de sinais central e o conector de alimentação 12V próximos da posição original, para possibilitar a utilização dos cabos originais. Na Figura 28 é apesentada a placa desenvolvida já com os componentes e conectores soldados.

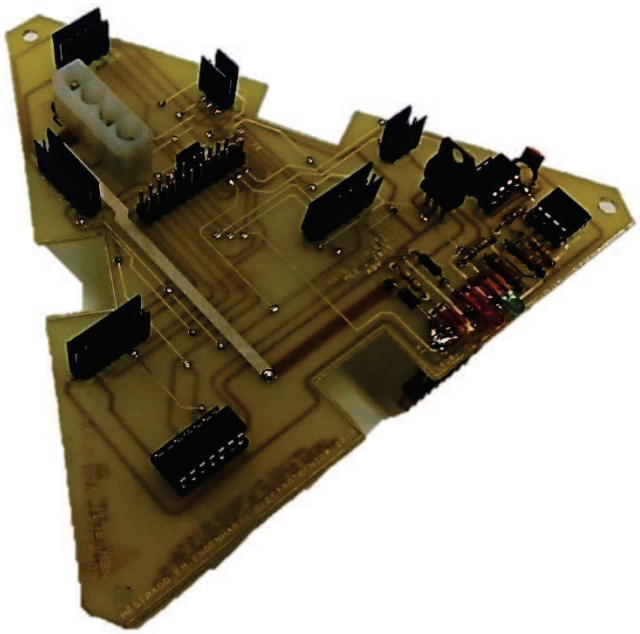


Figura 28 - Nova placa do coletor rotativo.

Tal como na configuração original, esta placa de circuito impresso na base do robô agrega um conjunto de sinais e realiza a interface com o controlador de baixo nível existente na torreta. Para possibilitar uma verificação da correta alimentação dos circuitos, esta placa dispõe de três *LED* de cores diferentes para cada tensão de alimentação usada no robô. Por conseguinte, pode ser feito, externa e visualmente, um primeiro diagnóstico do seu bom funcionamento. As cores dos *LED* usados para cada valor de tensão são apresentadas na Tabela 4.

LED	**TENSÃO [V]**
Vermelho	24
Amarelo	12
Verde	5

Tabela 4 - LED de alimentação.

O nível de tensão de 24V vem a esta placa apenas para a sinalização da sua presença no sistema e para medição do seu nível de tensão, dado que os controladores dos motores recebem esta alimentação diretamente da *'relay-board'*. É também nesta placa que entra a tensão de 12V para alimentar alguns dispositivos, assim como um conversor DC-DC para a obtenção de 5V necessária para alguns circuitos de controlo do robô. O conversor escolhido é um '*murata OKI-78SR*' que pode fornecer até 1,5A. Para proteção contra inversão da polaridade das baterias, esta placa contém díodos na entrada das duas tensões de alimentação.

Para a ativação dos circuitos de potência é necessário atracar o relé K1 da *'relay-board'*, sendo este controlo feito a partir de um sinal de 5V (*'Activate'*), recorrendo a um par de transístores de sinal, BC547, e média potência, TIP32. Também nesta placa está implementado um circuito de condicionamento de sinal com opto-isolamento para a leitura do nível de tensão da série de baterias (24V). Isto permite obter um valor proporcional da tensão medida condicionada para os 3.3V do controlador de baixo nível, permitindo a sua monitorização (a detalhar na Secção 3.5.2).

Resta referir a existência de um circuito integrado com portas lógicas que fazem um 'e' lógico dos sinais '*ready'* dos três controladores. Esta operação revela-se necessária dado que não existem condutores em número suficiente para o envio dos sinais em separado até à torreta. O desenho esquemático da placa, bem como os cálculos feitos para o dimensionamento do circuito aqui referido, podem ser consultados no Anexo K.

3.4. *Motores*

No acionamento dos dois graus de liberdade principais, velocidades linear e angular, são usados os dois motores originais que se encontram completamente operacionais, conforme abordado no capítulo anterior. Para o terceiro grau de liberdade, que consiste na rotação da torreta, foi necessário substituir o motor danificado. Pelo elevado custo de um motor igual, para essa substituição foi usado um motor, existente no laboratório de robótica, do mesmo fabricante que, pelas suas características elétricas e mecânicas, é aplicável ao AGV em desenvolvimento.

O motor usado é o GM14902 ('*Pittman – LO-COG'*) com tensão de alimentação de 30,3V e caixa de desmultiplicação de 5,9:1. O codificador que equipa esta montagem é de 512 pulsos por volta, 'HEDS-9100 – I00' da '*Avago Technologies'*, semelhante aos originalmente instalados neste robô.

Para o correto funcionamento dos três motores, os controladores são configurados tendo em conta as características elétricas e mecânicas de cada motor.

3.5. *Sistema de processamento*

O processamento necessário para o controlo do robô, conforme apresentado genericamente no início deste capítulo, é realizado a dois níveis.

Para o controlo de alto nível, é usado um PC portátil com sistema operativo *Linux*[2] e o *Robot Operating System (ROS)*[3], ambos softwares de código aberto (*open-source*).

O *ROS* é um meta-sistema operativo criado para o desenvolvimento de aplicações de robótica, contendo um vasto conjunto de bibliotecas e ferramentas, desenvolvido de forma a permitir uma utilização modular e mais facilmente portável para diferentes robôs.

O controlo de baixo nível é realizado mediante um conjunto de três microcontroladores: dois microprocessadores de sinal de 16 bits, módulos QEI, e um microcontrolador de 32 bits, módulo *ChipKIT*™. Na Figura 29 mostra-se o diagrama que permite visualizar de que forma se interligam os diversos dispositivos— no qual o 'Motor 1' é para velocidade linear, o 'Motor 2' para a velocidade angular e, por fim, o 'Motor 3' destina-se ao controlo da rotação da torreta.

[2] *Xubuntu 14.04LTS*
[3] *ROS - Indigo Igloo*

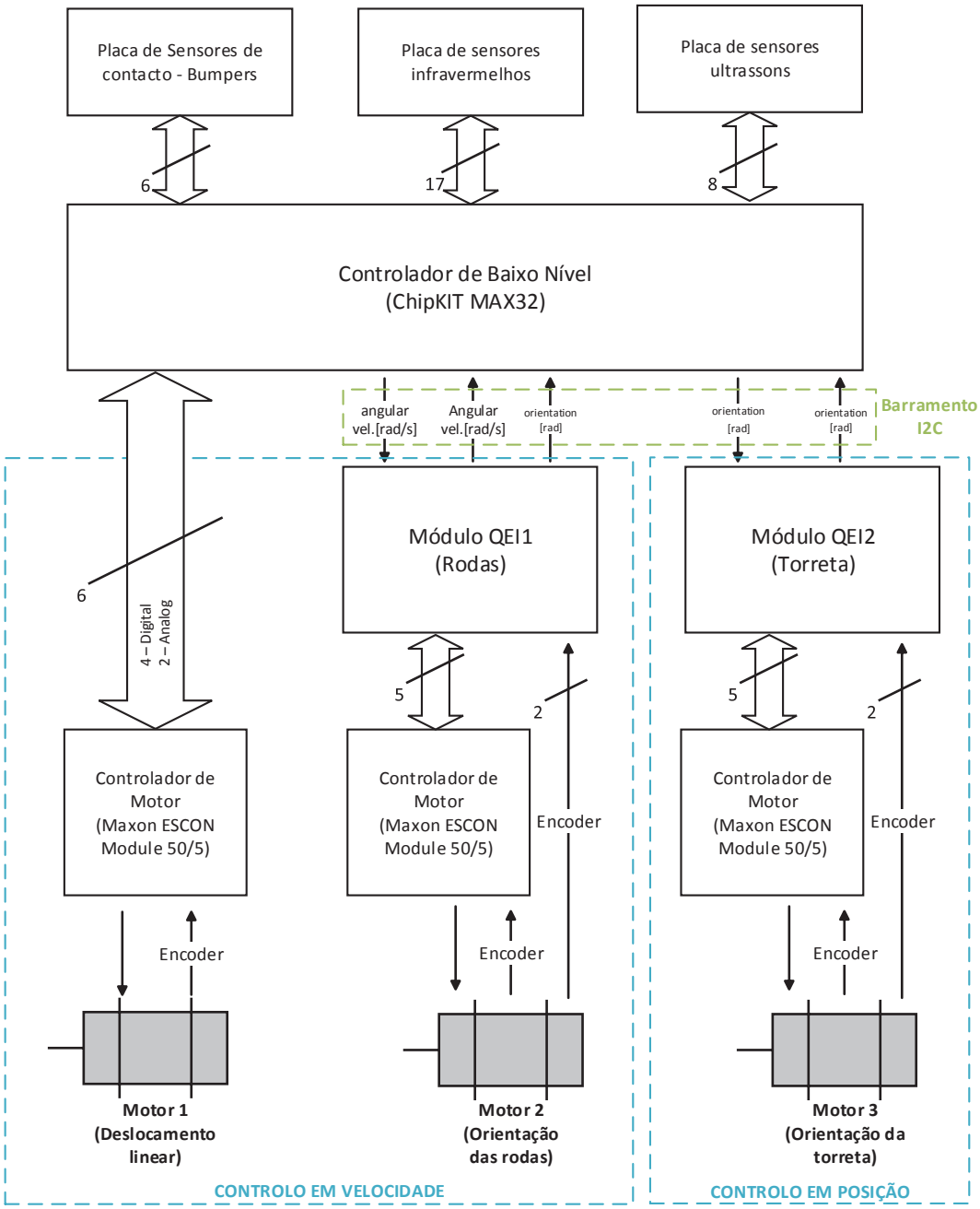


Figura 29 - Diagrama do controlo de baixo nível do AGV.

Para a conexão de todos os sinais do robô, foi criada uma placa (Anexo L) para acoplar ao *ChipKIT*, habitualmente designada como *Shield,* com dois conectores para encaixe dos módulos QEI e para a ligação dos conectores dos vários periféricos (Figura 30).

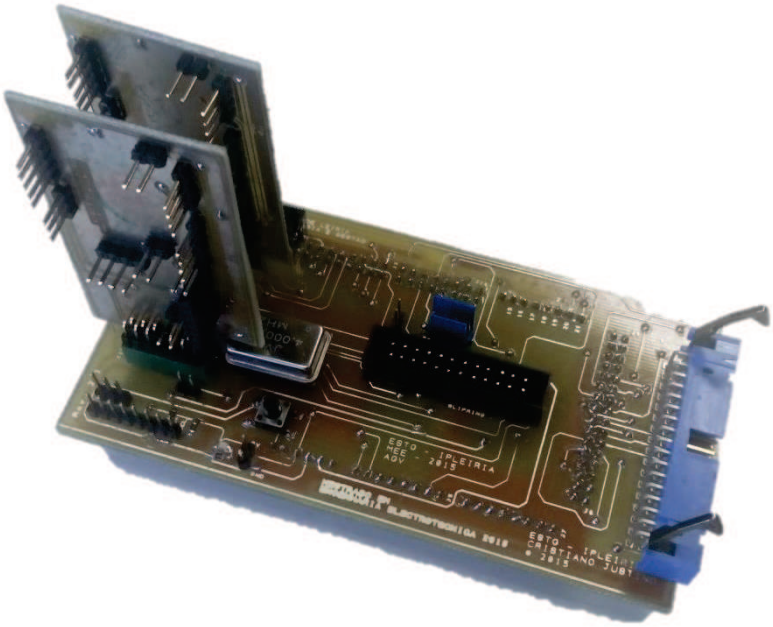


Figura 30 - Controlador de baixo nível.

Para controlar de forma correta os movimentos do robô, é necessário saber a orientação das rodas e da torreta. Os controladores selecionados para o projeto são controladores de velocidade, revelando-se imprescindível criar controladores de posição para cada um destes dois graus de liberdade. Com esse propósito, foi desenvolvido um módulo específico que possibilita este tipo de controlo, recorrendo a um microcontrolador de sinal com módulo de codificador em quadratura, designado de *Quadrature Encoder Interface* (QEI). O seu desenho esquemático encontra-se no Anexo M.

3.5.1. Módulos QEI

O desenvolvimento do módulo em questão tem como propósito controlar a posição e velocidade de um grau de liberdade do robô e medir a sua posição: QEI1 – rotação das rodas (velocidade angular); QEI2 – rotação da torreta. Apesar das diferenças no controlo dos dois graus de liberdade, o *hardware* do módulo é igual para ambos, sendo depois criado *firmware* para cada um deles tendo em conta as especificidades de cada conjunto motor – sistema mecânico. Para realizar este controlo foi escolhido um controlador dedicado, ficando a seu cargo o controlo de posição, medição de velocidade e o controlo do motor através do uso de um módulo PWM. A escolha recaiu sobre microcontroladores de sinal digital da família dsPIC30F (*Digital Signal Controllers*) da *Microchip* [21]. Os controladores desta família

têm arquitetura de 16 bits e caracterizam-se pelo seu alto desempenho. O primeiro controlador testado foi o dsPIC30F3010, Figura 31, que possui diversos módulos periféricos dedicados, que o tornam num dispositivo de grande capacidade de processamento, sendo de realçar para o presente projeto: um módulo de PWM dedicado com seis canais; um módulo de comunicação dedicado, usado para o protocolo I2C; e um módulo QEI também dedicado. Este módulo QEI tem várias características que o tornam versátil, como as entradas dos dois sinais de codificador incremental e outra de pulso de posição de referência, o contador bidirecional de 16 bits, os modos de medição em 2x ou 4x, os filtros de ruído digital programáveis para todas as entradas, a interrupção para posição e o *reset* de contador automático por *overflow* ou *rollover*.

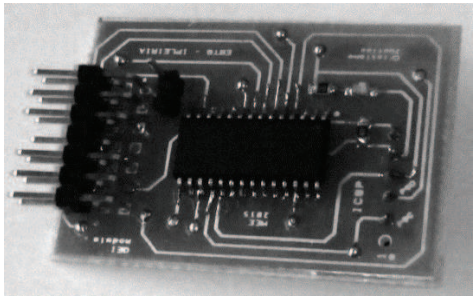


Figura 31 - Módulo QEI com dsPIC30F3010.

Escolhido o processador base, começa-se pela comunicação deste módulo com o *ChipKIT*, tendo-se optado pelo protocolo I2C por este necessitar de poucos recursos e estar presente em ambos os controladores. Foi definido que o *ChipKIT* será sempre o dispositivo mestre (*Master Device*), sendo assim o controlo do barramento assumido por si. Qualquer transmissão de dados é iniciada pelo mestre, quer seja pelo envio de dados a um ou mais dispositivos, quer seja pelo pedido de dados a um dispositivo escravo (*Slave Device*). Portanto, o dsPIC é o dispositivo escravo.

Assim, o *ChipKIT* escreve para o módulo QEI usando o barramento I2C. O código escrito para realizar esta tarefa teve por base as bibliotecas disponibilizadas pela *Microchip*, criadas para rotinas de escrita em memórias EEPROM do mesmo fabricante. Foi possível verificar que o dispositivo mestre executava corretamente a comunicação. Porém, o dispositivo escravo, dsPIC, não a detetava. Neste ponto, surgem dificuldades que levam a processos morosos no sentido de resolver o problema: ao iniciar a comunicação o módulo I2C do dsPIC não faz disparar a respetiva interrupção. Após vários testes e pesquisas em páginas *web* dedicadas, foi possível encontrar no fórum da *Microchip* [22] a referência a um problema

detetado neste módulo: em modo *Slave,* o *buffer* de receção é preenchido mesmo quando o endereço não corresponde ao configurado para o dispositivo, provocando *overflow* do registo de estado do I2C. Este *overflow* inibe a asserção da *flag* da interrupção de receção I2C. Consequentemente, a rotina de interrupção não é chamada, não havendo leitura dos dados. Levando a cabo o teste do mesmo código, usando um controlador diferente, no caso um dsPIC30F3011, o resultado foi o esperado inicialmente, concretizando a comunicação entre os dois dispositivos.

Posto isto, foi redesenhada a placa do módulo QEI para integração deste processador, semelhante ao escolhido inicialmente, contendo alguns recursos adicionais, produzido em encapsulamentos de 40 ou 44 terminais (neste projeto foi utilizada a versão TQFP de 44 terminais - Figura 32).

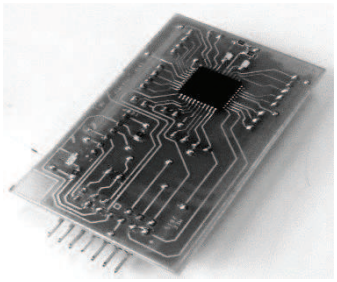


Figura 32 - Módulo QEI com dsPIC30F3011.

Este módulo está concebido para permitir realizar a sua programação de forma simples, com a inclusão de um conector *ICSP* de 6 terminais, conectando o programador ICD3 da *Microchip*. Nesta placa de circuito impresso existem três *LED*: um vermelho que nos indica a correta alimentação a 5V e dois verdes controláveis por *software* pela utilização dos bits 2 e 3 da porta D do controlador. A conexão com a *Shield* é realizada mediante um conector de 14 terminais a 90º, em duas filas, em que um dos terminais centrais é retirado para possibilitar o bloqueio contra inversão da posição da placa. Além de todos os sinais necessários ao controlo, restam dois terminais de reserva para eventuais necessidades futuras. Também com esse intuito, o módulo disponibiliza o acesso às entradas e saídas do controlador – presentemente não utilizadas – com conectores localizados nas extremidades da placa de circuito impresso.

O sinal de referência recebido do robô, além da sua normal ligação ao terminal específico do periférico QEI, está ligado de forma a realizar uma interrupção no código, RE8 – INT0.

Verificou-se esta necessidade aquando da realização de testes durante o desenvolvimento do *firmware* para os acionamentos com os novos controladores de motores. Esta situação deve-se ao facto de o módulo QEI ter como função o controlo de posição do motor, permitindo fazer um *reset* ao contador de impulsos sempre que o *codificador* do motor assira o sinal de referência. No caso do sistema mecânico do AGV, o sinal de referência está acoplado na extremidade do sistema de transmissão, ao invés de o acoplamento ser feito no motor. Assim, para a medição de uma volta completa na rotação das rodas ou da torreta, o motor dá várias voltas, o que provoca o *overflow* do contador, fazendo o *reset* do contador. Para o correto controlo do contador, a deteção do *overflow* é usada para incrementar o contador expandido. O *reset* total do contador é conseguido recorrendo a uma interrupção externa onde é ligado o sinal de referência do sistema mecânico do robô.

3.5.2. Módulo *ChipKIT*

O controlador principal para o baixo nível é um controlador PIC32, incorporado na placa de desenvolvimento *ChipKIT*™, criada pela *Digilent* e pela *Microchip*, sistema compatível com a disseminada plataforma *Arduino*™ [23]. O modelo que se revela necessário para este projeto é o *MAX32*™, Figura 33, de dimensões idênticas ao *Arduino Mega2560*, mas disponibilizando 83 entradas e saídas. Possui todas as vantagens desta conhecida plataforma, sendo as principais diferenças de relevar: o processador de 32bits com oscilador a 80MHz, a memória RAM de 128K, a memória Flash de 512K, ter suporte para vários periféricos como I2C, SPI, UART e Ethernet, entre outros. Embora opere a 3.3V, este dispositivo é compatível com os 5V habituais em todas as entradas e saídas digitais. Para as entradas e saídas analógicas deve-se ter em conta esta particularidade, a fim de evitar danos irreversíveis no microcontrolador.

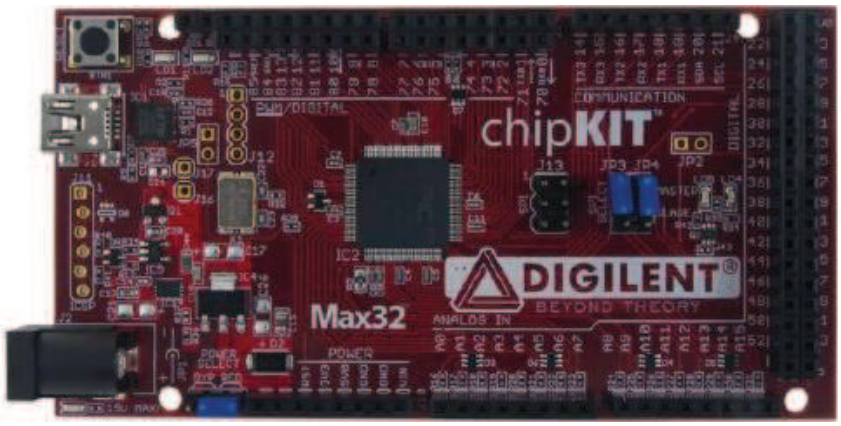


Figura 33 - Placa de desenvolvimento ChipKIT Max32.

Além de cinco saídas com modulação PWM, este dispositivo tem cinco interrupções externas que são de extrema importância no controlo dos vários componentes do nosso robô.

Na placa acoplada ao *ChipKIT*, ligam todos os conectores do robô, nomeadamente os provenientes das placas de ultrassom, infravermelhos e coletor rotativo. É importante agregar a informação relativa a todos os sinais, bem como as especificidades de alguns deles (como os que devem ser interrupção externa ou analógicos, por exemplo) para o desenho da *Shield*. Os sinais a controlar nesta placa, entradas e saídas, são apresentados no ANEXO N.

A *Shield* desenvolvida dispõe de um *LED* indicador de alimentação (*Power LED*) e um botão de *reset*, conforme demonstra a Figura 34. É possível verificar também a existência de um conector de reserva para acesso ao barramento I2C. O oscilador incluído não é presentemente usado, pelo facto de os controladores dos módulos QEI incorporarem oscilador no seu encapsulamento.

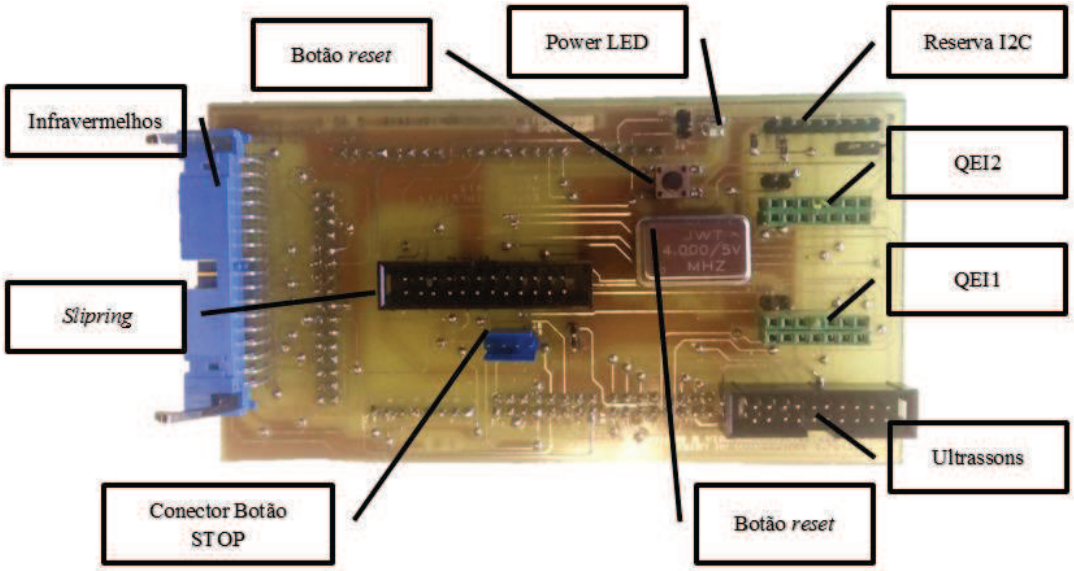


Figura 34 - Shield para o ChipKIT.

Todos os sinais que requerem a utilização de interrupção externa têm de ser direcionados para os terminais específicos para o efeito. Assim, são usados quatro sinais digitais para interrupção do *Max32*, ficando definidos com o sinal de *bumper* como prioridade máxima; de seguida os sinais relativos aos infravermelhos e por fim o sinal de eco (*Echo*) do ultrassom. Esta escolha prende-se com o evento relacionado com cada sinal. Quando ocorrer um evento no sinal de *bumper*, será indicador de choque do robô com algum obstáculo. Os infravermelhos são os que possibilitam a leitura de curtas distâncias, sendo por isso de prioridade intermédia. Os sensores ultrassom permitem medir distâncias maiores, não sendo possível medir dentro da gama de valores dos infravermelhos, o que nos leva a colocar o sinal para este evento como menor prioridade. Estas atribuições são apresentadas na Tabela 5.

ChipKIT Pin	Interrupt	Sinal
2	INT1	Bumper_Y
7	INT2	IR_DONE
21	INT3	IR_OVERFLOW
20	INT4	SONAR_ECHO

Tabela 5 - Interrupções externas.

Numa primeira fase, pretendia-se realizar todo o controlo de baixo nível apenas com o *ChipKIT*, caso o processamento fosse suficientemente rápido. Ao longo do trabalho de desenvolvimento foram detetadas algumas dificuldades relacionadas com a necessidade de

realizar controlo de posição em dois graus de liberdade. Tomando a decisão de usar um controlador QEI externo, permitiria realizar o controlo de posição e simultaneamente reduzir a quantidade de informação a processar pelo ChipKIT. Consequentemente passou-se a gerar o sinal PWM também no referido módulo, ficando a comunicação entre os módulos a requerer apenas 2 ligações para o barramento I2C.. Para implementar estas alterações, a *Shield* foi redesenhada tendo em conta os terminais dedicados à comunicação I2C e as ligações dos vários periféricos com ligações em terminais associados a interrupções externas no *ChipKIT* e a escolha dos conectores adequados a cada elemento.

Considerando as funções específicas de determinados terminais disponíveis, houve a necessidade de proceder a alterações à plataforma original do ChipKIT. O barramento principal de I2C (I2C1) é disponibilizado nos terminais 20 (SCL) e 21 (SDA), cuja utilização é requerida para uma interrupção externa, já mencionada acima. Consultando a ficha técnica do controlador PIC32, verifica-se a existência de outros periféricos I2C, não referenciados na documentação do *ChipKIT*. Após alguns testes, confirma-se que é possível utilizar o segundo periférico de I2C (I2C2), sendo, porém, necessário proceder a duas alterações: uma em *hardware* e outra em *software*. Estas alterações estão documentadas no Anexo O.

O controlo da velocidade linear é realizado no *ChipKIT* com uma saída em PWM, configurada no terminal 10, no qual é obtido um sinal de frequência de 500Hz. Conforme a especificação técnica do controlador, o sinal PWM utilizado pode estar na gama de 10Hz a 5kHz, sendo impreterível que o fator-de-ciclo varie de 10% a 90%, com resolução de 0.1%, o que corresponde a velocidades zero e máxima, respetivamente, de acordo com a documentação técnica dos controladores *Maxon* dos motores. Para leitura do valor de velocidade linear instantânea, recorre-se a uma saída analógica do controlador de velocidade, Motor1, parametrizado para que a sua saída tenha um valor de tensão proporcional à velocidade do motor. Em situação de velocidade zero, a saída tem uma tensão de 1,65V e a variação é linear como ilustra a Figura 35.

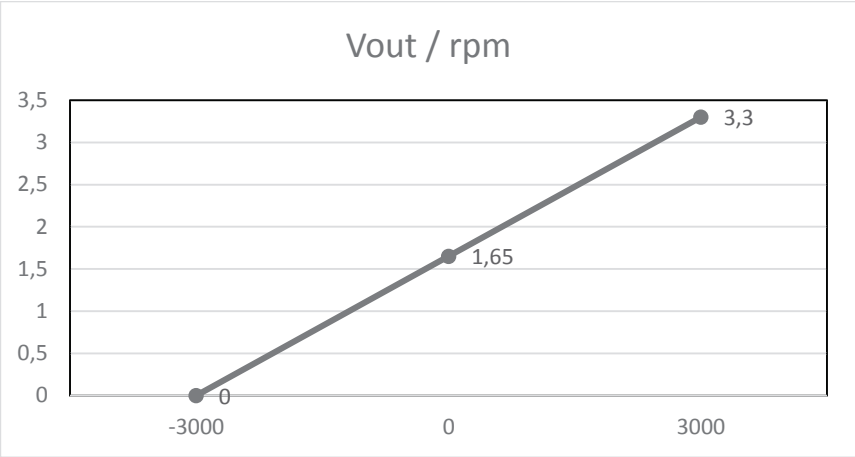


Figura 35 - Relação do feedback Vout/rpm do controlador do Motor 1.

3.6. *Firmware*

O controlo do *hardware*, abordado no ponto 3.5, é realizado por *software* desenvolvido especificamente para estes microcontroladores, denominado *firmware*. Pelas características dos diferentes microcontroladores utilizados, os ambientes de desenvolvimento são também diversos. Para os dois módulos QEI referidos anteriormente, são analisadas as diferenças do programa desenvolvido para cada um deles, tendo em conta a especificidade do grau de liberdade a atuar. Para estes módulos, foi usado o "MPLAB X IDE v3.00" com o compilador adequado para os controladores de 16bit da *Microchip*®, "MPLAB XC16 C Compiler", e ainda o programador e depurador "MPLAB® ICD 3 In-Circuit Debugger System", Figura 36. Este possibilita a programação dos módulos com os processadores já montados nas respetivas placas de circuito impresso, mediante a utilização de um conector "ICSP – In Circuit Serial Programming" [24]. São utilizadas diversas funções genéricas da família dsPIC [25] e também outras mais específicas da família dsPIC30F que serão documentadas ao longo da explicação do programa desenvolvido.

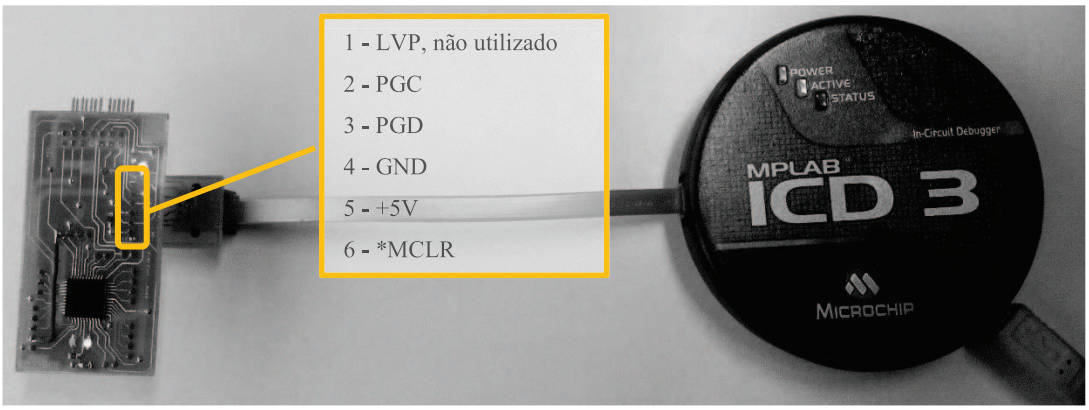


Figura 36 - Programador MPLAB® ICD 3 conectado ao módulo QEI.

Os dois módulos de controlo, para as rodas e torreta, estão implementados com *hardware* idêntico, como mencionado na Secção 3.5.1 acima, e os respetivos códigos foram desenvolvidos procurando manter a mesma base. A inicialização dos periféricos dedicados é realizada de igual forma, com funções semelhantes para todas as tarefas a realizar por cada módulo. As principais diferenças a destacar são o controlo em velocidade, realizado no módulo QEI1 – Rodas, e o controlo em posição, no módulo QEI2 – Torreta.

Como referido anteriormente, a comunicação entre os módulos e o *ChipKIT* é realizada utilizando o protocolo I2C, cujas mensagens mantêm o formato, apesar dos diferentes comprimentos. O seu funcionamento será explicado na secção seguinte.

Por fim, é abordado o desenvolvimento do código para o controlador PIC32, *ChipKIT* Max32 referido anteriormente, para o qual foi usada a plataforma disponibilizada pelo fabricante, MPIDE versão 0023.

3.6.1. Comunicação I2C

No respeitante à comunicação entre módulos, o barramento I2C funciona sempre com o *ChipKIT* como dispositivo mestre e os módulos QEI como escravos, em modo '*Single Master*'. Portanto, qualquer comunicação é iniciada pelo dispositivo mestre, o qual pode enviar instruções ou pedidos de informação a qualquer dos outros módulos. Foi definido um conjunto de três mensagens para envio de comandos e uma mensagem de pedido de dados. Considerando que a gestão do barramento é feita pelo periférico dedicado, como sinais de '*Start*', '*Stop*' e '*Acknowledge*', por exemplo, destacam-se os *bytes* considerados para formar as mensagens.

Uma mensagem de comando é encabeçada pelo endereço do dispositivo (sete *bits* e o *bit* de leitura/escrita), seguido de um *byte* que indica a instrução e, por fim, um valor numérico. A estrutura da mensagem para o módulo QEI1 e apresentada na Figura 37, onde são também indicados os valores aplicáveis para cada campo.

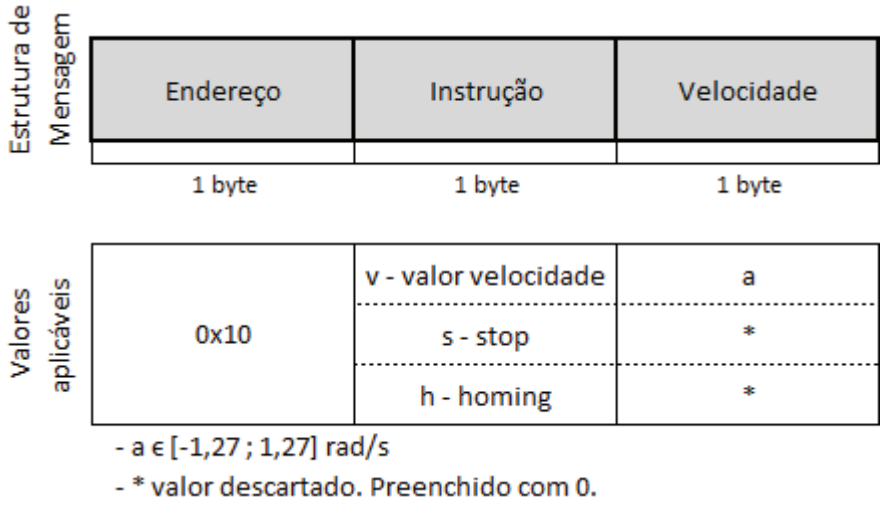


Figura 37 - Estrutura da mensagem I2C para o módulo QEI1 - Rodas.

A estrutura da mensagem para a torreta é semelhante, sendo o valor numérico ajustado para a gama de rotação da torreta, conforme mostra a Figura 38.

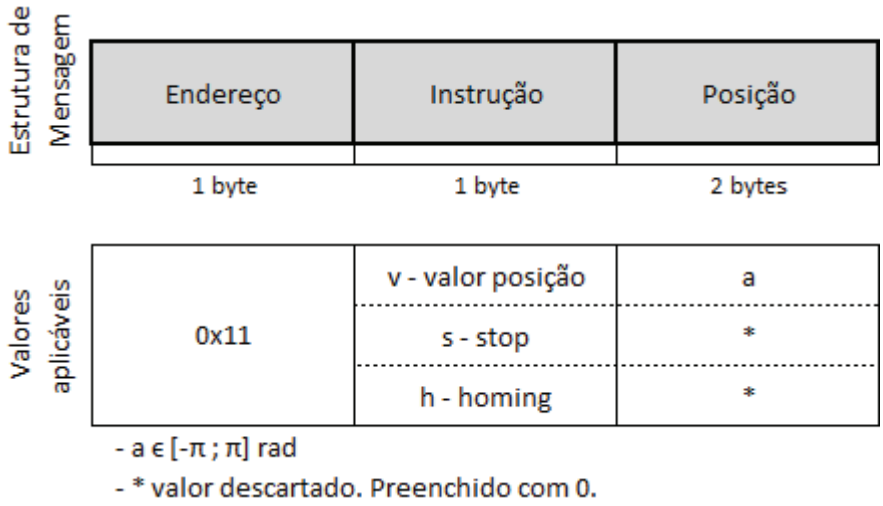


Figura 38 - Estrutura da mensagem I2C para o módulo QEI2 - Torreta.

O comando de '*Homing'* permite movimentar as rodas ou a torreta para a orientação de zero radianos, posição de referência '*Home'*. Caso seja iniciado um ciclo de '*Homing'*, todas as mensagens de comando serão descartadas até que se finalize essa ação.

Noutras situações, tratando-se de um comando de velocidade, o valor numérico recebido é validado e usado para o cálculo da ação de controlo necessária com vista à obtenção do comportamento requerido. As expressões são específicas de cada módulo por terem em conta a cinemática do robô e serão abordadas na Secção 3.6.2 e na Secção 3.6.3. No caso de a interrupção ocorrer para um pedido de informação, a mensagem de resposta é composta por um *byte* indicativo do estado do módulo, seguido pelos valores numéricos adequados ao grau de liberdade a controlar. O *byte* para o estado do módulo pode representar a execução de um ciclo de '*Homing'* — em que a mensagem se inicia com o byte 'H', e é preenchida com zeros nos restantes campos— ou expressar o estado de rotação ou repouso — representado por 'R' no início da mensagem, e preenchida com os valores, desse instante, para cada módulo: orientação e velocidade angular das rodas; orientação da torreta. A execução do código que implementa as ações descritas é esquematizada no fluxograma apresentado em Anexo P.

3.6.2. Módulo QEI1 – Rodas

Nesta secção é abordado o desenvolvimento do código para o módulo QEI1 que permite controlar, em velocidade, a rotação das rodas do robô pelo uso do periférico dedicado para saída em PWM. De salientar que o periférico QEI é inicializado para o modo 2x, significando que a contagem é feita a cada flanco ascendente e descendente do canal A, sendo o canal B é usado para determinar o sentido do contador. Adicionalmente, é realizado o cálculo da orientação das rodas. O programa consiste na utilização de vários periféricos, aos quais corresponde um bloco de código, função, que é chamado por um evento associado a cada um desses periféricos. No programa principal são inicializados esses periféricos, ficando depois a correr em ciclo infinito, conforme demonstra o fluxograma do programa apresentado na Figura 39.

Na função de inicialização I2C, void I2CInit(void), é configurado o periférico do microcontrolador, ativando as *flags* necessárias e atribuído o endereço ao dispositivo, 0x10, funcionando com endereços em modo de 7 bit. Todas as comunicações I2C são realizadas em modo 'full speed', a 400 kbit/s, conseguida pela alteração da configuração original do ChipKIT.

Seguidamente, é chamada a função para inicialização do periférico PWM, onde se configuram os respetivos registos de controlo para uma saída com frequência de 5kHz, dentro da gama de frequência de sinal do controlador do motor, e ainda é alterado o valor do

registo para que a saída fique com um fator-de-ciclo de 10%. Este periférico recorre a um temporizador de 15 bits, com frequência parametrizável e possui vários modos de operação.

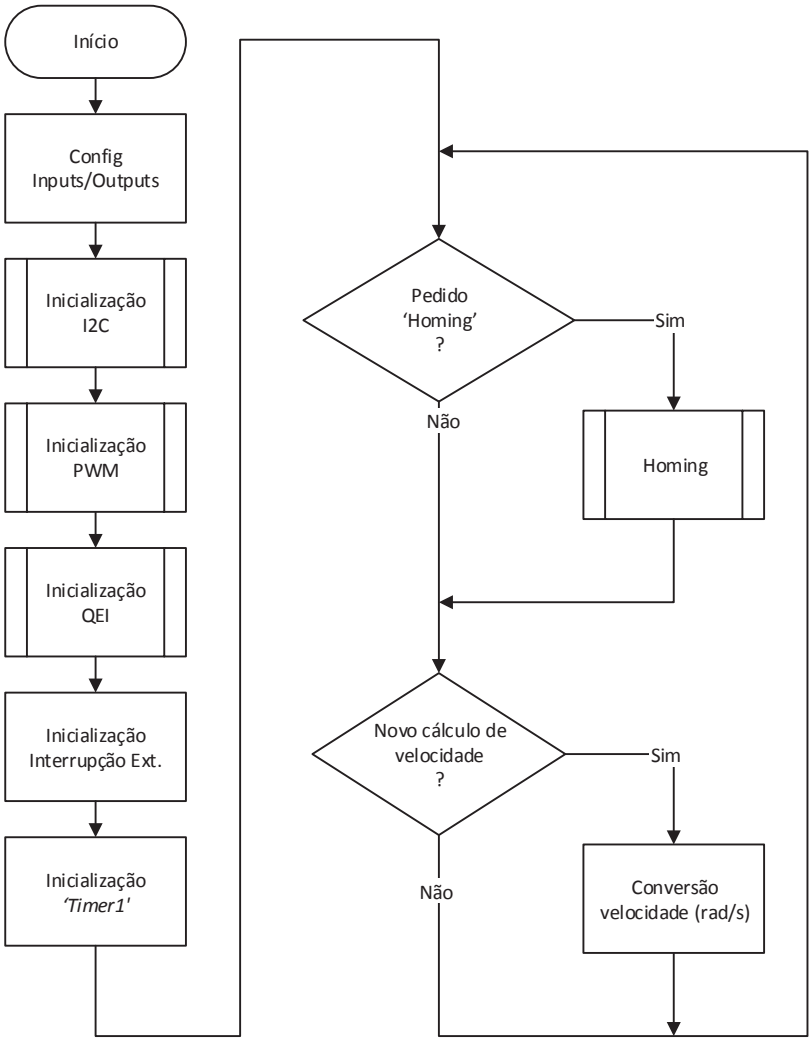


Figura 39 - Fluxograma do programa principal do módulo QEI1.

Para a implementação do sinal PWM, o módulo QEI possui um registo de 15bit (PTPER) para definir o período. O valor deste registo determina o período do sinal, e consequentemente a sua frequência. Para inicializar o registo com o valor adequado procede-se ao cálculo conforme a documentação acerca do controlo de motores em PWM [26] e ficha técnica do controlador [21].

$$T_{CY} = {}^{1}/_{MIPS * 10^{6}}$$

$$PTPER = \frac{F_{CY}}{F_{PWM} * (PTMR\ Prescaler)} - 1$$

$$MIPS = {(F_{OSC} * PLL)}/{4}$$

O sinal de relógio interno é configurado para uma frequência de 7,37MHz, com PLL de 8x, totalizando 58,96MHz. Pela expressão anterior obtém-se MIPS=14.74, e consequentemente $F_{CY} = 14.74x10^6$.

Resulta assim:

$$PTPER = \frac{14.74 \times 10^6}{5 \times 10^3} - 1 = 2947$$

Com PTPER = 2947 é obtida a frequência de 5kHz — a máxima admissível na entrada do controlador; valores superiores para PTPER traduzem-se em frequências inferiores — sendo o ajuste do fator-de-ciclo feito mediante a alteração do registo PDC1. Este é inicializado com o valor 580, o que corresponde a um fator-de-ciclo de cerca de 10%, como apresentado na Figura 40.

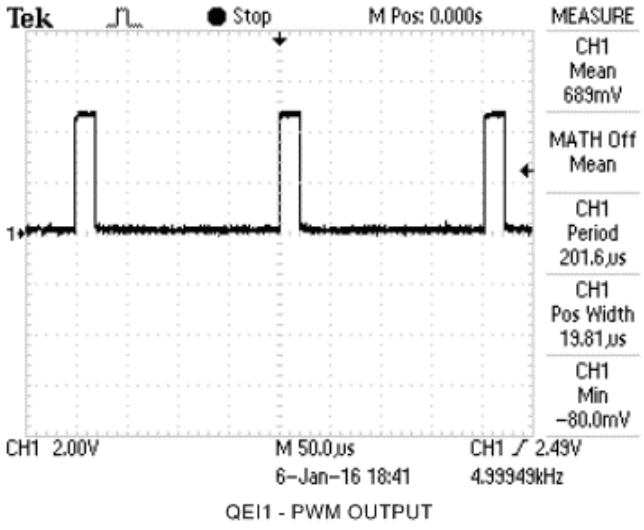


Figura 40 - Gráfico da saída de PWM do módulo QEI com fator-de-ciclo de 10%.

Após a receção do valor de velocidade angular, este é usado no cálculo do fator-de-ciclo de PWM adequado para que o acionamento produza o movimento desejado. Assim, a expressão que relaciona as duas grandezas foi obtida com ensaios práticos ao robô e é dada por:

$$DC_{PWM} = \omega_{rodas} \cdot 68{,}19 + 10 \quad [\%]$$

Para obter a saída de PWM com o fator-de-ciclo pretendido, podemos relacionar o seu valor com o do registo PDC1. Assim, considerando que PDC1 = 2947 corresponde a 50% de fator-de-ciclo, obtemos:

$$PDC1 = \omega_{rodas} \cdot 4019 + 589 \quad , \epsilon \ [\,589\,;\,5412]$$

Portanto, o valor de PDC1 é calculado diretamente a partir da velocidade angular pretendida, sendo apenas verificado que o seu valor se encontra no intervalo adequado, [-1,20 ; 1,20] rad/s, para o correto funcionamento do acionamento. Esta operação é realizada no momento de receção da velocidade, na interrupção I2C descrita na Secção 3.6.1.

O periférico dedicado QEI é configurável para vários modos de operação [27], sendo para isso utilizados os registos de configuração e de filtragem de entradas, *QEICON – Control/Status Register* e *DFLTCON – Digital Filter Control Register*, respetivamente. O valor máximo de contagem de pulsos é definido pelo registo MAXCNT —*Maximum Count Register*, cujo valor é comparado com o valor instantâneo da contagem de posição POSCNT — *Position Count Register*, ambos contadores de 16 bits. Portanto, a contagem máxima é 65535, o que se revela insuficiente para o sistema mecânico em questão. A leitura dos sinais de codificador é feita no motor; e o sinal indicador de referência está presente no extremo oposto do sistema mecânico. Para contornar esta situação, foi criado um contador expandido (*overflow* ou *rollover*) que é incrementado ou decrementado conforme o sentido de contagem. Desta forma é possível expandir o contador até ao valor necessário para uma volta completa das rodas, isto é, 271000.

Como referido na Secção 3.5.1 acima, o sinal de referência está ligado de forma a realizar uma interrupção no código, INT0, sendo alterado o flanco de deteção utilizado, consoante o sentido de rotação: flanco descendente em rotação com sentido horário e flanco ascendente no sentido contrário.

A última inicialização feita neste programa é uma interrupção por temporizador, *Timer1*, usado para realizar o cálculo da velocidade angular, considerando que esta corresponde a um número de incrementos num período de tempo fixo [28]. O período desta interrupção tem de ser inferior ao tempo necessário para completar meia volta à velocidade máxima. A velocidade nominal do motor deste grau de liberdade é 6151 rpm, pelo que foi usado para o cálculo 6500 rpm para ter uma margem de segurança.

$$T_{INTERRUPÇÃO} = \frac{60}{2*v_{máx}} = \frac{60}{2*6500} = 4{,}62 \times 10^{-3}\ [\text{s}]$$

Calculado o intervalo para a interrupção, é necessário determinar o valor a escrever no registo de configuração do período, TPER1:

$$TPER1 = \frac{T_{INTERRUPÇÃO}}{T_{CY} \times (Timer\ Prescale)} = \frac{4{,}62 \times 10^{-3}}{67{,}84 \times 10^{-9} \times 64} = 1006$$

Finalizadas todas as configurações, o funcionamento do módulo ocorre por eventos de funções associadas às interrupções já configuradas: QEI, INT0, Timer1 (descrito no parágrafo anterior) e I2C (descrito na Secção 3.6.1).

Nativamente, o periférico QEI pode gerar eventos de interrupção em diversas circunstâncias. No entanto, a configuração presente neste módulo apenas permite a ocorrência do evento pelo transbordo do contador de posição, por *overflow* ou *rollover*, o que possibilita a atualização do contador expandido, como representado no fluxograma da Figura 41.

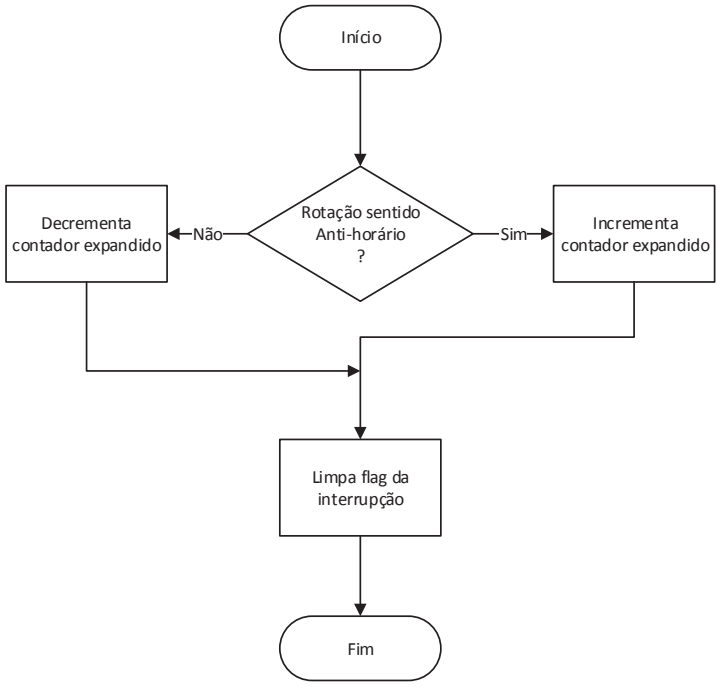


Figura 41 - Fluxograma da função de interrupção QEI.

O contador expandido é também manipulado na interrupção externa, INT0, que ocorre na deteção da posição de referência, originalmente denominado '*Home'*, altura em que se procede ao *reset* do contador tendo em conta o sentido da rotação. Na execução desta função

de interrupção verifica-se se está em execução um ciclo '*Homing'* que é aqui finalizada, provocando a paragem do movimento.

3.6.3. Módulo QEI2 – Torreta

O presente módulo tem como finalidade controlar a rotação da torreta, realizando o controlo em posição. Esta é a principal diferença comparativamente com o módulo apresentado na secção anterior, sendo o restante *firmware* dos dois módulos QEI semelhante, como referido anteriormente.

As inicializações dos periféricos dedicados usados no módulo são idênticas, sendo dispensável o temporizador, Timer1, por não ser necessário realizar o cálculo de velocidade, dado que o controlo é em posição. Consequentemente, o ciclo infinito do programa principal apenas verifica o eventual pedido de '*Homing'*, Figura 42.

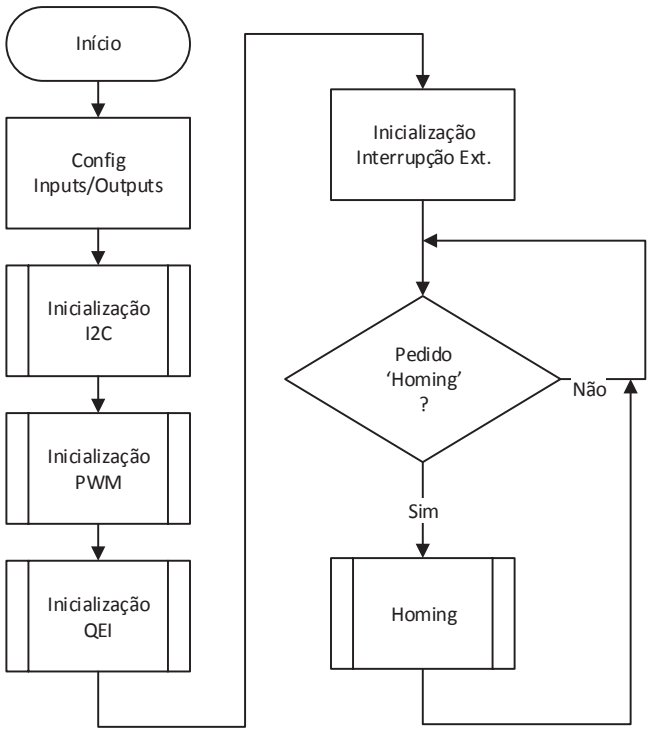


Figura 42 - Fluxograma do programa principal do módulo QEI2.

Na inicialização da comunicação I2C, é configurado o periférico do microcontrolador atribuindo o endereço ao dispositivo, 0x11, funcionando com endereços em modo de 7 bit.

A inicialização do periférico PWM é feita exatamente da mesma forma que a explicada na secção anterior. O valor do registo PTPER = 2947 determina a frequência de 5kHz, sendo o ajuste do fator-de-ciclo feito mediante a alteração do registo PDC1, inicializado com o valor 580 que corresponde a um fator-de-ciclo de cerca de 10%. Para o sistema mecânico do acionamento em questão, a expressão que relaciona o fator-de-ciclo (DC_{PWM}) e a velocidade angular ($\omega_{torreta}$) foi obtida após a realização de ensaios ao robô, resultando na equação:

$$DC_{PWM} = \omega_{torreta} \cdot 18{,}29 + 10 \quad [\%]$$

Relacionando o fator-de-ciclo e o valor do registo PDC1, tendo em conta que PDC1 = 2947 corresponde a 50% de fator-de-ciclo, vem:

$$PDC1 = \omega_{torreta} \cdot 1078 + 589 \;\; , \epsilon \; [\,589\,;\,4427]$$

O movimento da torreta é iniciado na função 'NewPositionCommand', cujo fluxograma é apresentado na Figura 43, onde o registo PDC1 é alterado de forma a criar a rotação para a posição pretendida. Para tal, foi definida uma função para que a velocidade angular seja adequada à amplitude da rotação necessária, tendo por base a informação do módulo QEI.

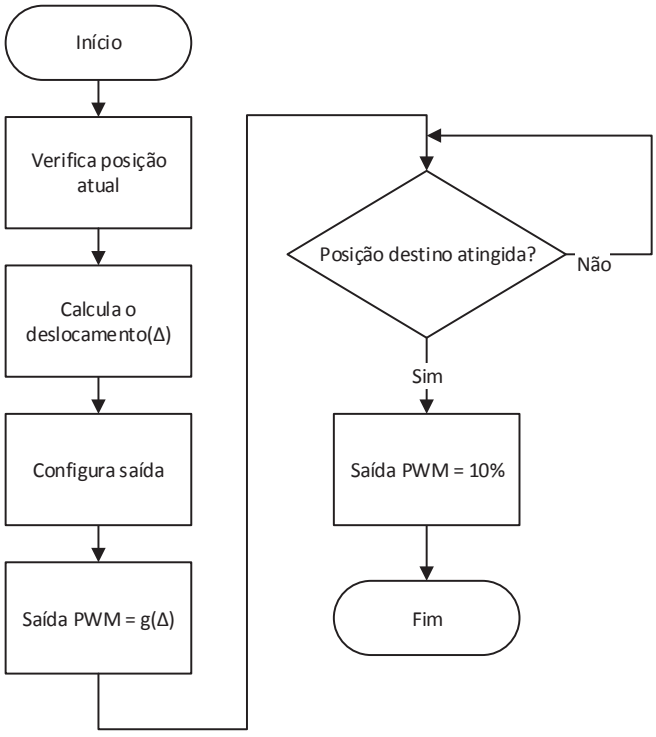


Figura 43 - Fluxograma da função de rotação da torreta.

No que diz respeito ao periférico QEI, há a relevar uma única diferença, que tem que ver com a contagem de uma volta completa. Para este conjunto motor — sistema mecânico o número de pulsos é de 83600, o que corresponde ao intervalo [0 ; 2π], valores de orientação da torreta usados no software do módulo. Externamente, todos os valores de entrada e saída do módulo estão convertidos para o intervalo [-π ; π]. A relação entre o valor de contagem (linha ponteada), a conversão em radianos usada internamente (linha em traço-ponto) e o valor de orientação de saída (linha contínua) são esquematizados na Figura 44.

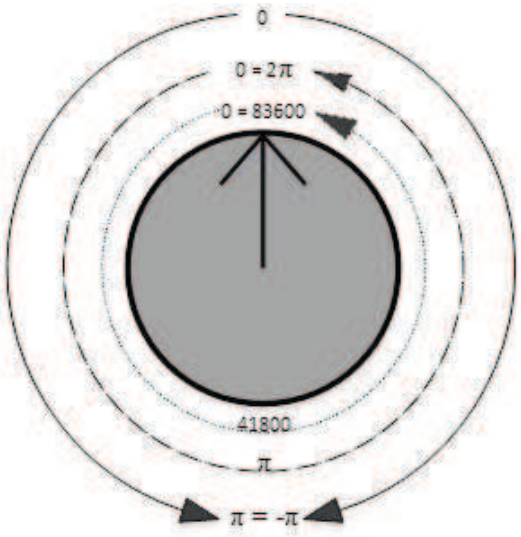


Figura 44 - Orientação da torreta: relação de valores.

Com a relação aqui esquematizada, é calculada a amplitude de rotação necessária, com a qual se determina a velocidade de rotação da torreta. É assim possível que movimentos de maior amplitude possam ser executados com maior velocidade, diminuindo o tempo de rotação. Na Figura 45 é apresentado gráfico da velocidade angular (ω) em função da amplitude do movimento (Δ).

O módulo descrito na presente Secção tem também implementado um contador expandido. A manipulação deste contador é similar ao apresentado na secção anterior, na rotina de interrupção externa, INT0, que ocorre na deteção da posição de referência, e onde se procede ao *reset* do contador tendo em conta o sentido da rotação. É na execução desta função de interrupção que se finaliza um ciclo '*Homing'*, caso seja verificada a sua execução, provocando a paragem do movimento.

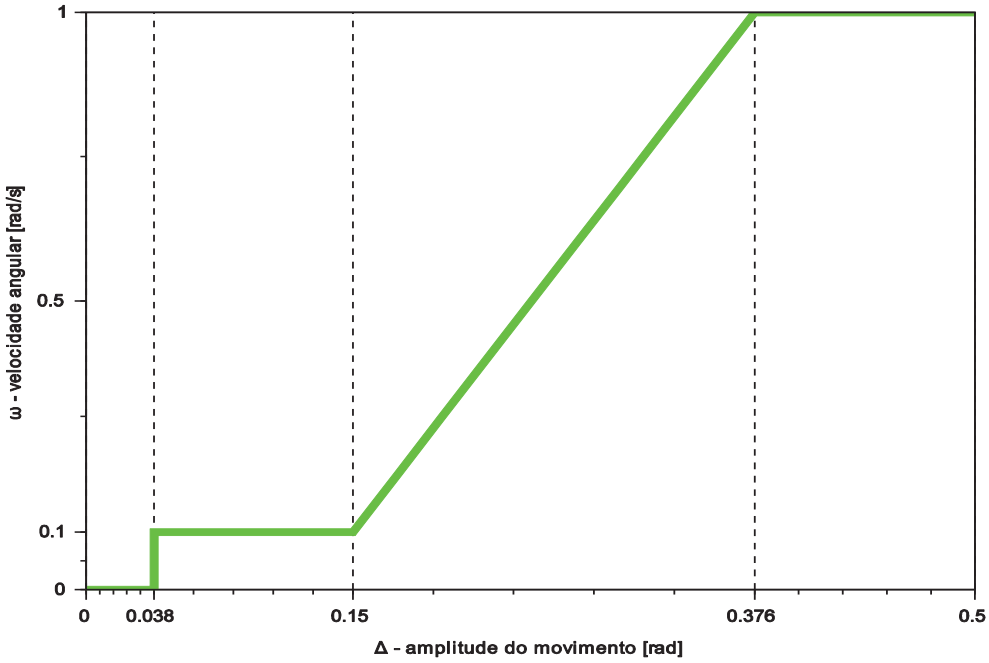


Figura 45 - Gráfico velocidade angular (ω) / amplitude do movimento (Δ).

3.6.4. Módulo *ChipKIT*

Nesta secção é abordado o programa desenvolvido para o ChipKIT, que implementa o controlador principal de baixo nível do robô, como enunciado no início do presente capítulo. Para a conexão com o computador é utilizado o protocolo '*rosserial*' [29] que permite enviar mensagens para o ROS, cuja descrição é feita na Secção 3.7 deste documento. Para melhor entendimento das tarefas realizadas no *ChipKIT*, importa destacar a utilização de mensagens e serviços do ROS para controlar o robô e para envio das respostas ao computador de alto nível, cuja estrutura e funcionamento será detalhada no capítulo 3.7.1.

Primeiramente, são incluídos os ficheiros de bibliotecas necessários ao funcionamento do programa, nomeadamente: *plib (periféricos PIC32)*, *wire (*protocolo I2C*)*, *ros (*protocolo *rosserial)* e os relativos ao pacote "agv_ipleiria". Seguidamente, é feita a atribuição de todos os sinais de entrada e saída aos vários terminais e respetivos nomes; a descrição detalhada destes sinais/terminais é apresentado no Anexo N. Concluídas todas as inicializações, o programa entra no *loop* em que são geridas as publicações das mensagens do estado do robô (*'Status')*, dados dos sensores (*'Sonar_Ranges'*, *'Infrared_Ranges'* e *'Bumper_Ranges'*), bem como o início de leitura de sensores ultrassom e sensores infravermelhos, conforme apresentado no fluxograma da Figura 46. As restantes funções do robô são executadas por

eventos que podem ocorrer por interrupção externa ou por serviços e mensagens do *rosserial*. Seguidamente, são abordadas as características mais relevantes do seu funcionamento.

- **Bumper**

A função é chamada pela interrupção externa um (INT1), e ocorre quando é premido um dos contactos *bumper*, referido na Secção 2.5, desabilitando os controladores de motores e provocando a paragem imediata do robô. Com o robô imobilizado, é realizada a verificação do contacto que gerou o evento com vista a obter a posição do choque relativamente ao referencial da base do robô. Esta informação é publicada no tópico de mensagem ROS '*Bumper_Ranges*' e, por fim, as saídas são ativadas para ficarem prontas para nova leitura. O fluxograma desta função é apresentado na Figura 46.

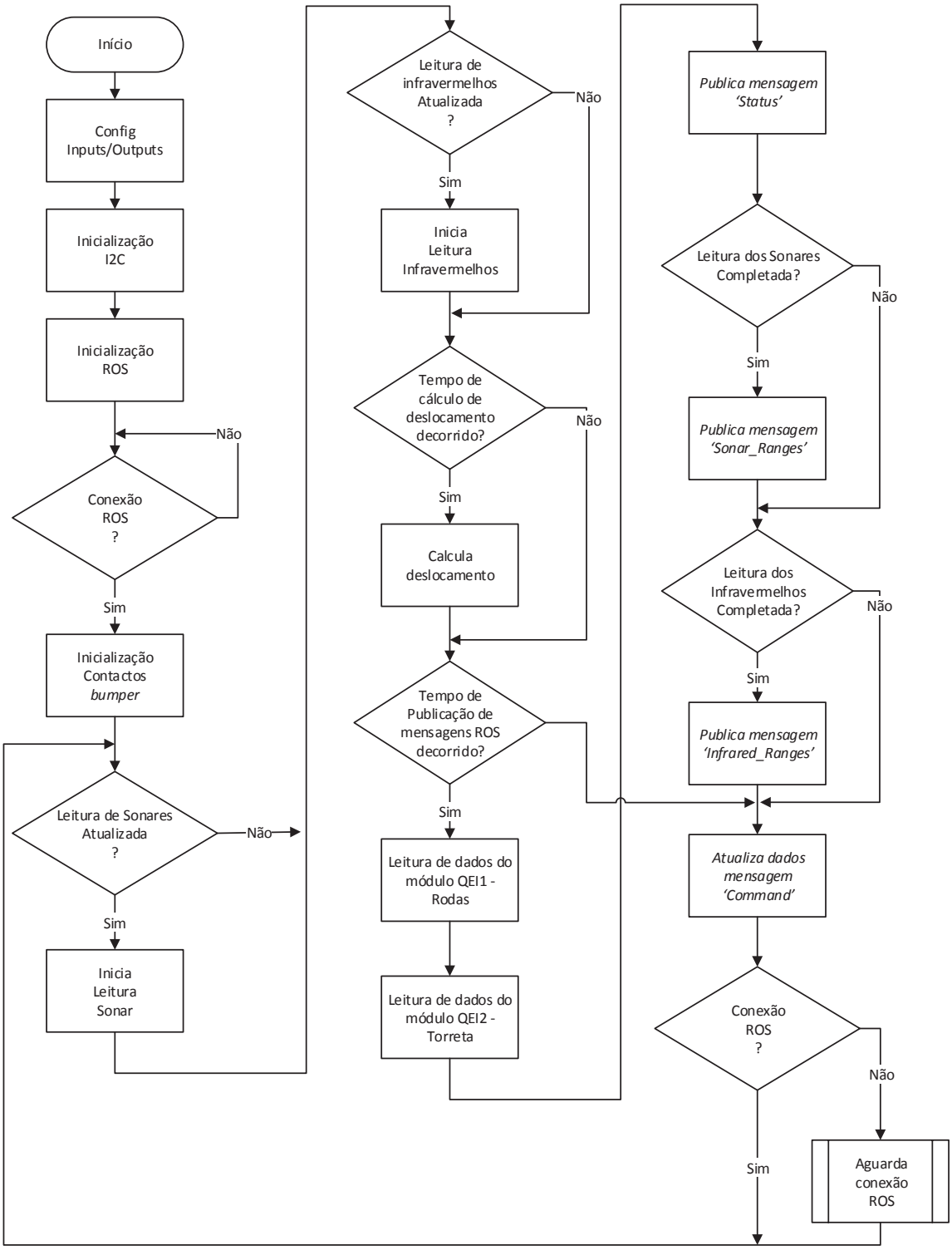


Figura 46 - Fluxograma do programa principal do módulo ChipKIT.

- **Infravermelhos**

De acordo com o analisado na Secção 0, para o funcionamento do controlador de infravermelhos estão implementadas três funções, uma para iniciar a leitura e duas para eventos de interrupção. A função de início da leitura começa por selecionar o sensor, com a sequência de leitura conforme apresentado na Figura 47. Faz o disparo asserindo a saída 'TRIG' e ativa as interrupções para deteção de *overflow* e fim de leitura ("OUT CC"), interrupt3 e interrupt2, respetivamente. A deteção de *overflow* é realizada com o evento de interrupção e faz apenas incrementar um contador em que cada incremento corresponde ao valor de 127. Quando é finalizada a leitura, ocorre o evento de interrupção, interrupt2, em cuja função se somam o resultado final dos 7 bits e o valor de *overflow* incrementado ao longo do ciclo de leitura. O valor de contagem é convertido em distância, em metros, segundo a expressão obtida pelos ensaios realizados para cada resolução do controlador. Por fim, são preenchidos os campos da mensagem ROS para os infravermelhos. Os dados relativos aos ensaios, seus resultados e fluxogramas destas funções são apresentados no capítulo 4.

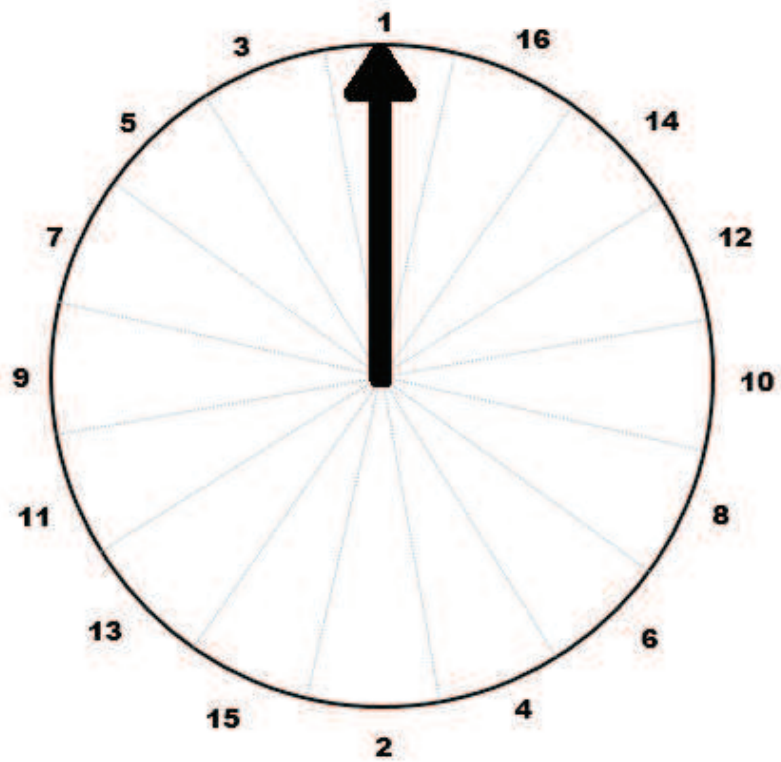


Figura 47 - Sequência de leitura dos sensores.

- **Ultrassons**

O controlo dos sensores ultrassom é realizado com a implementação de duas funções: a primeira habilita o controlador e provoca a emissão do ultrassom, a segunda função é executada por evento de interrupção externa, interrupt4, e onde é calculada a distância ao obstáculo, sendo esse valor colocado na mensagem ROS respetiva. A multiplexagem dos 16 sensores é feita na sequência indicada na Figura 47. Para melhor entendimento destas funções, são apresentados os seus fluxogramas no Anexo Q.

- **Mensagem de comando**

A cada atualização de dados do ROS, caso exista alguma mensagem de comando em espera, é chamada a função '*commandmessage'*, cuja execução permite receber os valores de velocidades, linear e angular, e posição da torreta, para de seguida alterar o comportamento do robô para os novos valores recebidos.

- **Serviço de configuração principal**

O serviço '*Agv_main*' permite ao utilizador enviar pedidos de configuração ao robô e obter a resposta acerca da configuração presente no momento. Na receção de um serviço no *ChipKIT* chama a função '*MainCallback'* que processa a informação e responde adequadamente. A utilização deste serviço é necessária para comutar a alimentação de potência, habilitar e desabilitar os controladores de motores, definir a frequência de atualização de dados de estado do robô e, ainda, realizar o posicionamento na posição índex das rodas e da torreta.

- **Serviço de configuração de sensores**

Quando ocorre a receção de um pedido de serviço '*Agv_sensors*', é chamada a função '*SensorsCallback'* que permite receber no controlo de baixo nível as configurações pretendidas relativas aos sensores enviadas pelo utilizador. É possível, para cada tipo de sensor, saber a configuração atual, além dos diversos pontos configuráveis para cada tipo de sensor. Os sensores infravermelhos possibilitam a sua ativação ou desativação, seleção da resolução e a ativação/desativação de cada sensor individualmente. No caso de se tratar de um pedido para configuração dos sensores ultrassom, é permitida a configuração dos 16

sensores individualmente e também a ativação/desativação de todos os sensores simultaneamente. Para os sensores de contacto *bumper* apenas é possível ativar ou desativar o seu funcionamento. O diagrama de fluxo que representa o funcionamento desta função é apresentado na Figura 48.

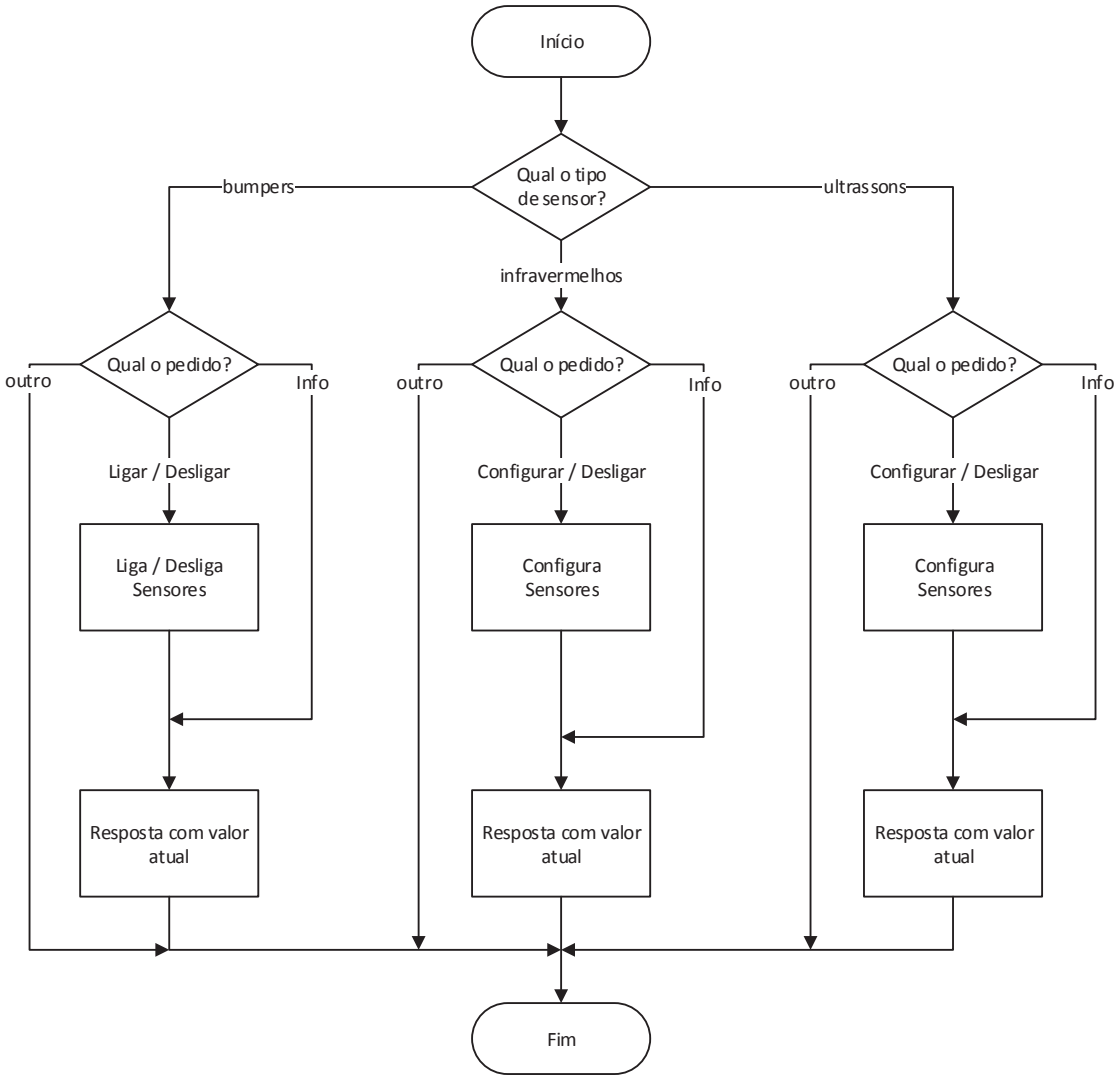


Figura 48 - Fluxograma da função SensorsCallback , serviço de sensores.

Caso a conexão com o controlador de alto nível se perca, ocorre a imobilização imediata do robô nos três graus de liberdade e são desabilitados os controladores dos motores, ficando a aguardar que se restabeleça o contacto. Depois de reatada a comunicação, o controlador de alto nível deve habilitar de novo os controladores com um pedido de serviço '*Agv_main*'.

3.7. *Software*

No presente subcapítulo são abordadas todas as questões subjacentes ao desenvolvimento do *software* de alto nível, com especial ênfase no *Robot Operating System (ROS)*.

Para o código desenvolvido teve-se em conta a sua construção, e foram criadas mensagens para os vários tipos de sensores, e na Figura 49 mostra-se o diagrama geral da comunicação ROS implementada.

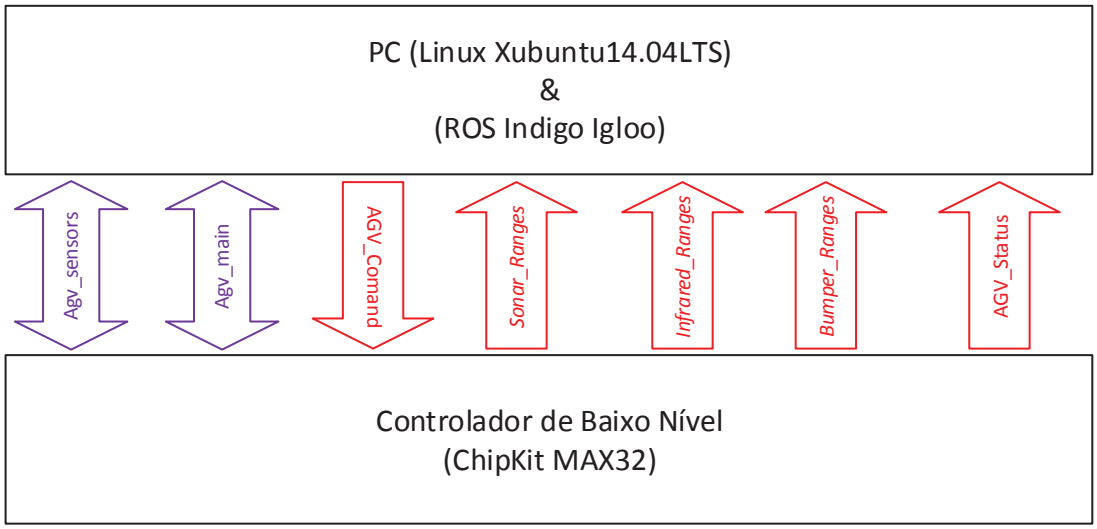


Figura 49 - Diagrama geral da comunicação ROS implementada.

3.7.1. ROS

O ROS é um meta-sistema operativo de código aberto (*open-source*) criado para aplicação específica em robótica que proporciona a abstração de *hardware*, controlo de dispositivos de baixo-nível, gestão de mensagens entre processos e controlo de pacotes. Também dispõe de ferramentas e bibliotecas para a criação, escrita e execução de código entre diferentes plataformas robóticas, sendo um sistema modular. Pelas várias características apresentadas, facilita a criação de *software* robusto e de uso genérico, com o intuito de simplificar tarefas e ambientes para robôs, permitindo a colaboração e desenvolvimento continuado de diversas soluções entre as várias instituições que integram esta comunidade de partilha [30] [31] [32]. A organização de *software* de código ROS é feita por pacotes, em que cada pacote contém bibliotecas, executáveis, *scripts* e outros ficheiros. Para melhor entendimento acerca dos

termos específicos do sistema operativo, são aqui abordados os mais importantes na execução do *software* desenvolvido no presente projeto:

- Nós (‘*Nodes’)*: Um nó é um executável que utiliza o ROS para comunicar com outros nós;

- Mensagens (‘*Messages’)*: As mensagens são o tipo de dados do ROS na subscrição ou publicação de um tópico, para troca de informação entre nós;

- Tópicos (‘*Topics’)*: Os tópicos são como barramentos onde os nós podem publicar mensagens ou subscrever um tópico para receber mensagens. Cada ligação a um tópico é unidirecional, podendo ser do tipo subscritores (ligações de leitura) ou publicadores (ligações de escrita);

- Serviços (‘*Services’)*: Os serviços são um conjunto de mensagens que permite comunicação bidirecional, com resposta a um pedido [33].

Pretende-se a utilização do ROS com vista à implementação de um robô com comunicação de alto nível padronizada para que a sua programação seja realizada de forma simples e intuitiva. Foi criado um pacote (*package*) específico para o AGV, denominado ‘agv_ipleiria’, contendo as mensagens e serviços adequados. Seguidamente são analisadas as mensagens e serviços criados para o referido pacote.

- **AgvCommand**

A mensagem ‘AgvCommand’ foi criada com a finalidade de permitir o envio de comandos ao robô, nomeadamente no que diz respeito às informações de cada grau de liberdade. Nesse sentido, a mensagem contém, para além do cabeçalho padrão do ROS, três valores *float32* com a posição da torreta e as velocidades linear e angular.

- **AgvStatus**

Usando a mensagem ‘AgvStatus’ o robô envia para o controlador de alto nível todas as informações do seu estado atual. Os primeiros dados da mensagem são o cabeçalho padrão do ROS e três variáveis binárias. Estas variáveis indicam o estado da alimentação aos motores – ‘Motor_power’ -, a informação relativa aos controladores de motores, prontos ou em falha, - ‘Motor_ready’ - e ainda a indicação da ocorrência de uma colisão – ‘Collision’.

Todos os dados seguintes são do tipo *float32* (valor em vírgula flutuante, em variável de 32 bits), com a informação relativa à tensão das baterias da alimentação de motores, '*Battery'*, e à odometria: velocidades linear e angular, pose (x e y), e orientação do robô e da torreta.

- **RangeArray**

Para as informações dos sensores foi criada uma mensagem 'RangeArray', baseada na mensagem *standard* do ROS '*Range'*, em que para o campo 'radiation_type' foi adicionado o valor para os sensores *bumper*. Um campo relativo à resolução dos sensores infravermelhos foi criado com uma variável do tipo inteiro. Todas as restantes variáveis são do tipo float32, como na mensagem *standard*, e o valor de distância 'range' foi alterado para criar um vetor de 16 valores, correspondendo às leituras de cada sensor.

Criado o tipo de mensagem específico para os sensores do AGV, na inicialização do robô são criados três tópicos com este tipo de mensagem: '*Sonar_Ranges'*, '*Infrared_Ranges'* e '*Bumper_Ranges'*, conforme apresentado na Secção 3.6.4. Os dados publicados pelo robô em *'Sonar_Ranges'* deixam com valor nulo a variável *'ir_res'* por não ser aplicável para este conjunto de sensores. Relativamente aos dados publicados no tópico *'Bumper_Ranges'*, a variável *'radiation_type'* tem o valor dois e no vetor ranges apenas são preenchidos os valores de posição zero e um, correspondendo ao ângulo de orientação da deteção detetada e o número do sensor atuado, respetivamente. Na inicialização do dispositivo, o vetor ranges é preenchido com o valor 99, no segundo elemento, o que indica um estado de "não-colisão". Quando ocorre uma colisão, neste elemento do vetor é colocado o número do sensor atuado, com um número entre 0 e 19, sendo de imediato publicado no tópico, conforme descrito no subcapítulo anterior.

- **Serviço de configuração principal**

O serviço '*Agv_main*' permite ao utilizador enviar informação de configuração ao robô e obter a resposta acerca da configuração presente no momento. Como define a especificação relativa a '*Services'*, a mensagem tem duas partes: envio de dados e receção da confirmação. A mensagem criada tem quatro variáveis para cada uma dessas partes. Todas estas variáveis são do tipo inteiro de 8 bits: 'Motor_power', 'Motor_enable', 'Homing' e 'Update_freq'.

A variável ‘Update_freq’ permite definir a frequência de atualização de dados ROS, cujo valor padrão é 50Hz. A utilização das variáveis ‘Motor_power’ e ‘Motor_enable’ é feita com os valores zero, um e dois; que correspondem a desativar, ativar e consulta, respetivamente. Assim, na receção de um serviço no *ChipKIT* chama a função ‘*MainCallback’* que processa a informação e responde adequadamente. A utilização deste serviço é necessária para comutar a alimentação de potência, habilitar/desabilitar os controladores de motores, definir a frequência de atualização de dados de estado do robô e, ainda, realizar o posicionamento na posição de referência das rodas e da torreta.

- **Serviço de configuração de sensores**

Quando ocorre a receção de um pedido de serviço ‘*Agv_sensors*’, é chamada a função ‘*SensorsCallback’* que permite receber, no controlo de baixo nível, as configurações pretendidas relativas aos sensores enviadas pelo utilizador. É possível, para cada tipo de sensor, saber a configuração atual, além dos diversos pontos configuráveis para cada tipo de sensor. Os sensores infravermelhos possibilitam a sua ativação ou desativação, seleção da resolução e a ativação/desativação de cada sensor individualmente. No caso de se tratar de um pedido para configuração dos sensores ultrassom, é permitida a configuração dos 16 sensores individualmente e também a ativação/desativação de todos os sensores simultaneamente. Para os sensores de contacto *bumper* apenas é possível ativar ou desativar o seu funcionamento.

4. Testes e Resultados

No desenvolvimento do presente projeto houve a necessidade de realizar diversos testes, desde uma fase inicial para estudo do estado de conservação e funcionamento do robô até aos testes finais para verificação da operacionalidade do sistema concluído.

Os testes iniciais foram relatados no capítulo 2, onde se descreve o estado de conservação do robô antes da sua recuperação, permitindo perceber cada teste realizado com o intuito de avaliar as várias componentes do robô. Fundamentalmente, esses testes permitiram obter informação para a decisão acerca dos componentes a reutilizar e os que revelaram necessidade de substituição, bem como permitiram perceber o funcionamento dos componentes por forma a realizar o seu controlo numa fase mais avançado do projeto.

O trabalho foi sendo desenvolvido de forma modular, transversalmente a todo o *hardware* e *software*. Após estar concluída a implementação do *software* para controlo dos vários componentes de baixo nível, foi desenvolvido o código para a comunicação do controlador baixo nível com o computador de alto nível.

Após terminados os módulos relativos ao controlo de movimento linear e angular das rodas, foram realizados testes no sentido de aferir do bom funcionamento de odometria e realização de ajustes que se revelassem necessários. Na Tabela 6 é apresentada informação relativa a algumas das medições feitas para testes de odometria. A distância calculada é resultado do processamento realizado pelo ChipKIT, com posterior envio por *rosserial* para o PC, e a medição da distância real resulta da utilização de fita métrica, com marcação do ponto de início e fim do movimento realizado pelo robô.

Distância calculada [m]	Distância medida [m]	Diferença [m]	Erro (%)
5,5	5,51	0,01	0,182%
5,97	5,9	-0,07	-1,173%
6,3	6,16	-0,14	-2,222%
6,63	6,55	-0,08	-1,207%
7,53	7,51	-0,02	-0,266%
8,7	8,85	0,15	1,724%
9,97	9,87	-0,10	-1,003%
9,98	9,92	-0,06	-0,601%
10,01	9,93	-0,08	-0,799%
10,02	9,95	-0,07	-0,699%

Tabela 6 - Medições de odometria.

A análise dos dados permite perceber que existe um erro importante na odometria, de dimensão considerável e inconstante. Este comportamento deve-se à forma como é feita a realimentação do controlador de motor. A saída analógica do controlador está configurada para fornecer um valor de tensão proporcional à velocidade do motor. A tensão de funcionamento do ChipKIT é de 3,3V, o que obriga a que a saída analógica esteja nessa gama. O motor na sua rotação máxima coloca o robô à velocidade linear de 0,60m/s, com uma gama de 6000 rpm, [-3000;3000] rpm, e que corresponde à saída de [0;3,3] V. Esta relação determina que fica disponível a gama de variação de 1,65V para 3000 rpm. Pelas características do sistema mecânico, verificam-se variações consideráveis na relação da velocidade do motor e a velocidade que é possível imprimir ao robô, essas variações traduzem-se nos resultados das medições de odometria apresentados.

Conforme descrito no capítulo anterior, o trabalho foi realizado para o funcionamento com sistema ROS, *rosserial*, cuja implementação em ChipKIT foi criada para este projeto. Isto permitiu realizar a comunicação entre os dispositivos. Porém surgiram dificuldades com a parametrização da frequência de envio de dados a partir do ChipKIT, dado que, para realizar esta tarefa com uma frequência exata, há necessidade de criar uma interrupção por temporizador (Timer), que nativamente é possível com as bibliotecas de código da plataforma MPIDE utilizada. A utilização de qualquer Timer revelou provocar o bloqueio do código ROS, ficando parado indefinidamente, por incompatibilidade de utilização do Timer com as outras funções.

O pacote de código ROS desenvolvido tem mensagens e serviços próprios, através dos quais são enviadas as informações para o PC de alto nível. A informação de odometria é enviada na mensagem ‘Status’, conforme já descrito na Secção 3.7.1. A abordagem inicial foi feita com o intuito de manter a mensagem de odometria padrão do ROS, ‘Odometry’. Os testes realizados permitiram verificar que a quantidade de campos que esta mensagem contém, não possibilita uma rápida execução do programa no ChipKIT. Por este facto houve a necessidade de criar na mensagem ‘Status’, bem como na mensagem ‘Command’, campos com a informação relativa à Pose e Odometria do robô.

Ainda com o código ROS no *ChipKIT*, surgiram outros problemas aquando da realização dos testes incorporando todos os módulos de *software* desenvolvido. Refere-se em particular o problema das interrupções externas utilizadas para o controlo de sensores (ultrassom, infravermelhos e *bumper*) cujo funcionamento provoca perda de sincronização com o ROS,

o que implica que o sistema operativo acabe por dar erro por dados recebidos erroneamente. Com o intuito de perceber o impacto destas interrupções na comunicação *rosserial* foram realizados testes apenas com o controlo dos sensores ultrassom e o tópico de mensagem ROS para os mesmos sensores que revelaram o bom funcionamento enquanto as interrupções externas estavam abaixo de 4,6Hz. Em medição real, a frequência a que ocorrem as interrupções está diretamente relacionada com a distância a medir: quanto mais próximo de um obstáculo, maior a frequência de disparo da interrupção. No caso das leituras de sensores infravermelhos, o problema torna-se mais grave, pela quantidade de disparos de interrupção necessária para cada leitura. Para a situação menos favorável, resolução máxima, a leitura pode ter valores de contagem próximos de 50000 pulsos, o que, para os 7 bits de saída, equivale a cerca de 400 interrupções, em cerca de 4ms, para uma única leitura. Para a leitura total dos 16 sensores podem ser necessárias 32 leituras, em 128ms, o que totaliza a possibilidade de 12800 interrupções. Pelo descrito se verifica a impossibilidade da utilização simultânea das interrupções externas e a comunicação *rosserial* no mesmo microcontrolador ChipKIT, para as frequências indicadas.

Foram realizados testes aos sensores de infravermelhos e ultrassons para a verificação do seu funcionamento já com controlo em alto nível. Essas medições foram acompanhadas de leituras realizadas com um sensor laser, do tipo '*range finder*', Hokuyo URG-04LX-UG01, para efeitos comparativos. Este sensor permite leituras com ângulo de 240º e alcance de 4m [34]. Na Figura 50 é apresentado o gráfico com os valores medidos pelo conjunto de sensores infravermelhos, para curta distância, linha amarela. Igualmente são mostrados os valores obtidos com o sensor '*range finder*', laser, para a mesma situação de teste (linha azul).

A distância medida usando os sensores infravermelhos, conforme descrito na Secção 2.8, é calculada com base numa equação linear, demonstrando um desempenho aceitável. Estes sensores permitem realizar medições de distâncias até 0,6m. Os tempos de medição são apresentados no Anexo G.

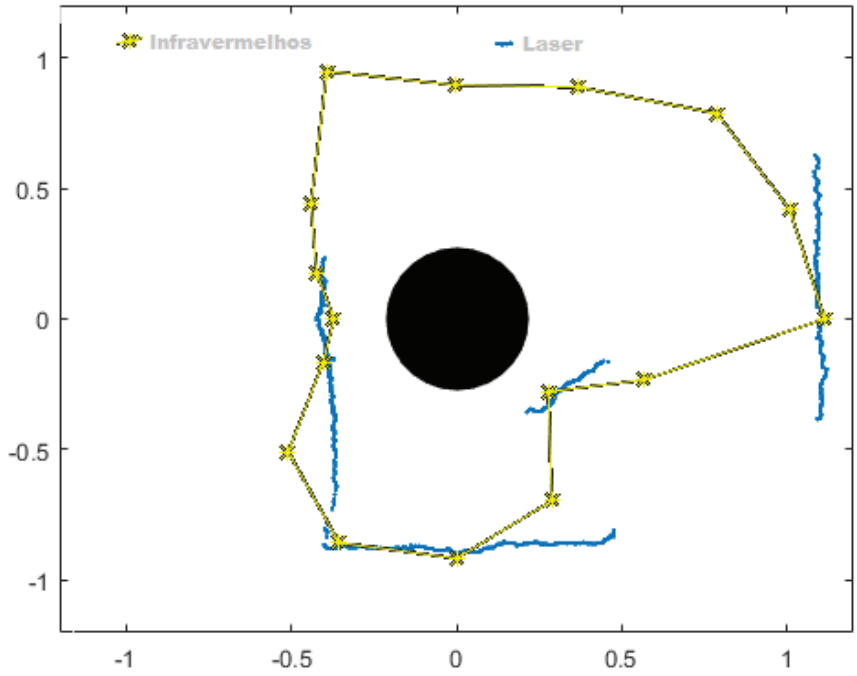


Figura 50 - Leitura de distâncias curtas com sensores de infravermelhos e laser.

Na Figura 51 pode ver-se o gráfico com os valores medidos pelo conjunto de sensores ultrassom, para distâncias superiores a 0,40m, assinaladas a amarelo. Para as mesmas condições de teste, a leitura executada com o sensor '*range finder*', resulta na representação gráfica mostrada a azul.

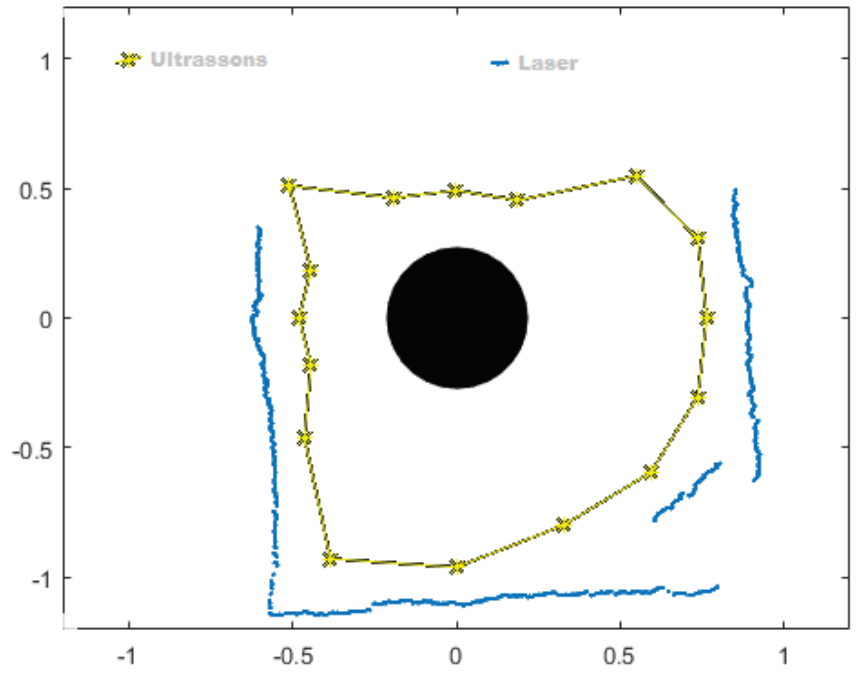


Figura 51 - Leitura de distâncias com sensores ultrassom e laser.

O robô foi testado com um programa simples para realizar movimentos pela sala, com desvio de obstáculos com rotações aleatórias. A deteção dos elementos em torno do robô foi realizada mediante a utilização do laser Hokuyo anteriormente referido.

5. Conclusões e Trabalhos Futuros

No presente, e último, capítulo são apresentadas as conclusões acerca do desenvolvimento do trabalho proposto para este projeto. Adicionalmente, são indicadas algumas linhas orientadoras para trabalhos a realizar futuramente com o intuito de melhorar o robô, colmatando as falhas detetadas na fase de testes do presente projeto.

Para este trabalho foi necessário proceder a um estudo extensivo do robô Nomad200, inoperacional e obsoleto, sendo o mote para a sua recuperação. Esse estudo permitiu saber o estado de conservação dos seus componentes, nomeadamente no que diz respeito a sensores, atuadores, controladores e estrutura mecânica. Esta tarefa revelou-se particularmente trabalhosa e morosa, tendo sido necessário compilar a pouca, e dispersa, informação existente, com grande componente de trabalho habitualmente designado de engenharia inversa. Os esquemas dos circuitos eletrónicos apresentados neste relatório são fruto deste trabalho.

Consequentemente, os passos seguintes consistiram no desenvolvimento de *hardware* para integrar o robô, aproveitando os componentes em bom estado de funcionamento. Em termos de programação, esta foi desenvolvida de forma modular, permitindo perceber o bom funcionamento de cada parte. No que respeita ao *firmware*, programação do hardware de baixo-nível, importa referir que se tornou computacionalmente pesado pela necessidade de utilização de diversas interrupções externas para controlar os diversos sistemas recuperados – sensores – , bem como para controlar os atuadores selecionadas para o projeto. Relativamente ao desenvolvimento do *software* de alto nível, nomeadamente os pacotes de mensagens ROS, com as particularidades inerentes a este sistema para robôs, a integração das várias componentes desenvolvidas demonstrou a existência de incompatibilidades, pela necessidade de tempos de execução de código precisos, cuja origem se centra na quebra de comunicação alto-nível – baixo-nível.

Conforme foi descrito no capítulo 4, verificam-se variações consideráveis na medição da velocidade linear, com erros de odometria. Para melhorar a medição da velocidade linear do robô deverá ter-se em conta, para trabalhos a realizar futuramente, a inclusão de um módulo QEI adicional para realizar o cálculo da velocidade

As questões relacionadas com a incompatibilidade do uso simultâneo dos sensores (de ultrassom e de infravermelhos) e do *rosserial*, são solucionáveis retirando a computação de ambas no mesmo microcontrolador. É recomendável a alteração do sistema de processamento, adicionando um ChipKIT para controlar os sensores e tratar a respetiva informação de forma dedicada. Esta separação poderia permitir, a comprovar em testes específicos, realizar as medições com os sensores, e passando o conjunto de informação já processada para o microcontrolador responsável pela comunicação com o alto nível. Ficando um ChipKIT com a tarefa de agregar toda a informação e enviar para o ROS, a correr no PC, e recebendo os comandos, poderia permitir o uso de mensagens padrão para enviar a informação da pose e odometria do robô, a comprovar mediante testes práticos.

A distância medida usando os sensores de infravermelhos, demonstra um desempenho aceitável com o cálculo por equação linear. Para melhores resultados neste cálculo, um ponto a melhorar no futuro é a alteração das expressões, de forma a obter uma expressão polinomial que deverá representar o sistema de forma mais exata.

Posto isto, o robô tem implementadas todo o controlo que se pretendia inicialmente, com as limitações descritas, estando operacional para algumas tarefas e estão já identificados diversos pontos de melhorias a introduzir futuramente.

Bibliografia

[1] D. Austin e H. I. Christensen, "DRobot - Nomadics Robot Software and Hardware Support," Agosto 2001. [Online]. Available: http://drobot.sourceforge.net/. [Acedido em 20 10 2015].

[2] A. Chopra, M. Obsniuk e M. R. Jenkin, "The Nomad 200 and the Nomad SuperScout: Reverse," em *Proceedings of the 3rd Canadian Conference on Computer and Robot Vision (CRV'06)*, 2006.

[3] G. Oriolo, G. Ulivi e M. Vendittelli, "Real-Time Map Building and Navigation for Autonomous Robots in Unknown Environments," em *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS*, Rome, Italy, 1998.

[4] U. Nehmzow, "Map Building through Self-Organisation for Robot Navigation," University of Manchester, Manchester, UK, 2000.

[5] U. Nehmzow e C. Owen, "Robot navigation in the real world: Experiments with Manchester's FortyTwo in unmodified, large environments," University of Manchester, Manchester, UK, 2000.

[6] P. Kopacek e N. Chivarov, "Modular Approach in Projecting of Intelligent Mobile Robots," Institute for Handling Devices and Robotics, Vienna University of Technology,, Vienna, Austria, 2002.

[7] J. Martínez-Gómez, R. Marfil, L. V. Calderita, J. P. Bandera, L. J. Manso, A. Bandera, A. Romero-Garcés e P. Bustos, "Toward Social Cognition in Robotics: Extracting and Internalizing Meaning from Perception," em *XV Workshop of Physical Agents*, León, Spain, 2014.

[8] J. Carlson e R. R. Murphy, "Use of Dempster-Shafer Conflict Metric to Detect Interpretation Inconsistency," University of South Florida, Tampa - Florida, USA, 2005.

[9] E. Amir e P. Maynard-Zhang, "Logic-based subsumption architecture," www.elsevier.com/locate/artint, 2001.

[10] G. Bianco e P. Fiorini, "Visual avoidance of moving obstacles based on vector field disturbances," Universitá di Verona, Verona Italy, 2001.

[11] E. Andrade, F. Caetano, J. Quaresma, L. Perdigoto e L. Conde Bento, "Rehabilitating a Nomad SuperScout Robot," em *Proceedings of the 10th International Conference on Mobile Robots and Competitions*, Leiria, Portugal, 2010.

[12] P. F. F. Alves, "Localização e Navegação de um Robô Móvel em Ambiente Interior," Leiria, Portugal, 2012.

[13] Nomadic Technologies, *Nomad 200 Hardware Manual,* Mountain View, CA 94043: Nomadic Technologies, Inc, 1997.

[14] M. Lindstrom, "Nomadic 200 Hardware Guide by Mattias Lindstrom," 19 setembro 2001. [Online]. Available: http://www.nada.kth.se/~mattiasl/nomad200/. [Acedido em setembro 2014].

[15] Lattice Semiconductor Corporation, *GAL®22V10 Device Datasheet,* Hillsboro, Oregon, 2010.

[16] SensComp, Inc, *Application Note 2004-3: Verifying 6500 or Smart Sensor Operation,* SensComp, Inc, 2004.

[17] RC Systems, *DoubleTalk PC/104 - Hardware Manual,* Everett, WA - USA, 1995.

[18] Proxim, Inc., *RangeLAN2/ISA - Wireless LAN Adapter for ISA Bus Computers - User's guide,* Mountain View, CA - USA, 1997.

[19] A. Gattupalli, "MODELLING, SIMULATION AND PLANNING OF A SYNCHRONOUS DRIVE ROBOT ON UNEVEN TERRAIN," Robotics Research Centre, International Institute of Information Technology, Hyderabad - 500 032, INDIA, 2012.

[20] maxon motor ag, *ESCON Module 50/5 - Servo Controller - P/N 438725 - Hardware Reference (rel4734),* 2014 .

[21] Microchip Technology Inc., *dsPIC30F3010/3011 datasheet - DS70141C,* 2006.

[22] Microchip, "Microchip Forum," 21 Janeiro 2008. [Online]. Available: http://www.microchip.com/forums/m308687.aspx. [Acedido em 9 Dezembro 2015].

[23] Digilent, Inc., *chipKIT™ Max32™ Board Reference Manual,* Pullman, WA 99163, January 27, 2015.

[24] Microchip Technology Inc., *In-Circuit Serial Programming™ (ICSP™),* 1997.

[25] Microchip Technology Inc., *dsPIC® Language Tools Libraries - DS51456B,* 2004.

[26] Microchip Technology Inc., *Section 15. Motor Control PWM - DS70062E,* 2007.

[27] Microchip Technology Inc., *Section 16. Quadrature Enconder Interface (QEI) - DS70063D,* 2010.

[28] M. T. Inc. e J. Zambada, *GS002 - Measuring Speed and Position with the QEI Module - DS93002A,* 2005.

[29] Open Source Robotics Foundation, "rosserial - ROS Wiki," Open Source Robotics Foundation, 20 11 2015. [Online]. Available: http://wiki.ros.org/rosserial. [Acedido em 25 03 2016].

[30] Open Source Robotics Foundation, "ROS / Introduction," Open Source Robotics Foundation, 22 05 2014. [Online]. Available: http://wiki.ros.org/ROS/Introduction. [Acedido em 25 01 2016].

[31] Open Source Robotics Foundation, "ROS Documentation," [Online]. Available: http://wiki.ros.org/. [Acedido em 28 01 2016].

[32] Open Source Robotics Foundation, "Questions - ROS Answers," [Online]. Available: http://answers.ros.org/questions/. [Acedido em 28 01 2016].

[33] "ROS - Concept," Open Source Robotics Foundation, [Online]. Available: http://wiki.ros.org/ROS/Concepts. [Acedido em 30 01 2016].

[34] Hokuyo Automatic Co., LTD, *Scanning Laser Range Finder - URG-04LX-UG01 (Simple-URG) - Specifications.*

[35] B. Barshan, "Directional Processing of Ultrasonic Arc Maps and its Comparison with Existing Techniques," *The International Journal of Robotics Research,* 2007.

[36] S. Banerjee, “A Comparative Study Of Underwater Robot Path Planning Algorithms For Adaptive Sampling In A Network Of Sensors,” University of Nebraska - Lincoln, 2014.

[37] A. Nizzoli, “Progettazione E Realizzazione Di Un Sistema Di Asservimento Visivo Per Un Robot Manipolatore,” Università degli Studi di Parma, Parma, Italy, 2006.

[38] J. Almeida, A. Almeida e R. Araújo, “Tracking Multiple Moving Objects for Mobile Robotics Navigation,” University of Coimbra, Coimbra, Portugal, 2005.

[39] N. L. Doh, H. Choset e W. K. Chung, “Relative localization using path odometry information,” Springer Science+Business Media, LLC, 2006.

[40] S. Aydin, I. Kilic e H. Temeltas, “Using Linde Buzo Gray Clustering Neural Networks for Solving the Motion Equations of a Mobile Robot,” King Fahd University of Petroleum and Minerals, Dhahran, Saudi Arabia, 2011.

[41] L. Yenilmez e H. Temeltas, “A new approach to map building by sensor data fusion: sequential principal component-SPC method,” Springer-Verlag London Limited, 2006.

[42] F. Tièche e H. Hügli, *From topological knowledge to geometrical map,* Neuchâtel, Switzerland: Pergamon, 1999.

[43] T. Koshizen, *The architecture of a Gaussian mixture Bayes (GMB) robot position estimation system,* Canberra, Australia: Elsevier, 2001.

[44] W. L. Xu, *A virtual target approach for resolving the limit cycle problem in navigation of a fuzzy behaviour-based mobile robot,* Palmerston North, New Zealand: Elsevier, 2000.

[45] S. Aydin e H. Temeltas, *Fuzzy-differential evolution algorithm for planning time-optimal trajectories of a unicycle mobile robot on a predefined path,* Istanbul, Turkey, 2003.

[46] C. Silva e B. Ribeiro, *Navigating mobile robots with a modular neural architecture,* Coimbra, Portugal, 2003.

[47] M. Mucientes, R. Alcalá, J. Alcalá-Fdez e J. Casillas, *Learning Weighted Linguistic Rules to Control an Autonomous Robot,* Santiago de Compostela, Spain: Wiley Periodicals, Inc., 2009.

[48] U. Nehmzow, *Quantitative analysis of robot–environment interaction—towards "scientific mobile robotics",* Wivenhoe Park, Colchester - UK: Elsevier Science B.V., 2003.

[49] C. C. d. Wit, B. Siciliano e G. Bastin, Theory of Robot Control, London: Springer - Verlag, 1997.

Esta página foi intencionalmente deixada em branco

Anexos