UNIVERSIDADE FEDERAL DO CEARÁ
CENTRO DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA DE TELEINFORMÁTICA
ÁREA: SINAIS E SISTEMAS

MODELO DE NAVEGAÇÃO PARA ROBÔS MÓVEIS BASEADO EM REDES DE PETRI COLORIDAS

ÍTALO JÁDER LOIOLA BATISTA

Fortaleza
2008

MODELO DE NAVEGAÇÃO PARA ROBÔS MÓVEIS BASEADO EM REDES DE PETRI COLORIDAS

Dissertação de mestrado submetida à Coordenadoria do Curso de Pós-Graduação em Engenharia de Teleinformática da UFC como requisito para obtenção do título de Mestre em Engenharia de Teleinformática.

Orientador: Prof. Giovanni Cordeiro Barroso, Dr.

Co-Orientador: Prof. Antonio Themoteo Varela, Dr.

FORTALEZA

2008

Modelo de navegação para robôs móveis baseado em redes de petri coloridas

ÍTALO JÁDER LOIOLA BATISTA

Dissertação aprovada em 30 de janeiro de 2008.

_____
Prof. Dr. Giovanni Cordeiro Barroso - UFC
(Orientador)

_____
Prof. Dr. Antonio Themoteo Varela - CEFETCE
(Co-Orientador)

_____
Prof. Dr. José Tarcisio Costa Filho - UFC
(Membro da Banca Examinadora)

_____
Prof. Dr. Pedro Fernandes Ribeiro Neto - UERN
(Membro da Banca Examinadora)

_____
Prof. Dr. José Marques Soares - CEFETCE
(Membro da Banca Examinadora)

A Deus, por esta oportunidade.

AGRADECIMENTOS

A Deus, pela vida e por todas as coisas que me tem dado até hoje.

Aos professores Giovanni Cordeiro Barroso e Antonio Themoteo Varela pelo interesse, incentivo e excelente orientação.

Ao professor José Tarcísio Costa Filho pelos incentivos e trocas de idéias que auxiliaram a elaboração deste trabalho.

Ao professor Walter Fetter Lages por ter me orientado durante o estágio no Laboratório de Sistemas de Controle, Automação e Robótica da UFRGS.

Aos colegas de trabalho do ITTI, pela convivência.

À minha família pelo incentivo e apoio em todos os momentos, e.

À FUNCAP pela provisão da bolsa de mestrado.

RESUMO

Sistemas de navegação autônomos devem ser capazes de definir uma seqüência de ações a serem tomadas por robôs móveis, dotados de um conjunto limitado de sensores, quando expostos a um ambiente externo suposto desconhecido e tendo que atender simultaneamente a um elenco de objetivos previamente especificados. O interesse científico no estudo de sistemas de navegação em ambientes desconhecidos e sujeitos ao atendimento de múltiplos objetivos é motivado basicamente pelo seu evidente potencial em aplicações industriais e pelo fato de demandarem a implementação de estratégias de soluções complexas. Este trabalho apresenta a modelagem de um sistema de navegação para robôs moveis por meio de Redes de Petri Coloridas. O modelo apresentado consegue simular várias situações, tais como a representação do mundo em volta do robô, interação com o ambiente, planejamento de trajetória, localização do robô e análise das baterias, bem como servir de base para implementação em um robô móvel real e otimização do sistema.

ABSTRACT

Systems of autonomous navigation must be able to define a sequence of actions to be taken by mobile robots endowed with a set of limited sensors, while exposed to an unknown environment and having to serve simultaneously a set of objectives previously specified. The scientific interest in the study of systems of navigation in unknown environment which are subject of serving to several objectives is motivated basically by its evident potential in industrial applications and by the fact of demanding the implementation of complex solutions strategies. This research presents the modeling of a navigation system for mobile robots through coloured Petri nets. The model presented here can simulate several situations, such as: the representation of the world around the robot, interaction with the environment, trajectory planning, robot location, battery analysis, as well as how to serve as a basis for implementation in a real mobile robot and optimization of the system.

SUMÁRIO

LISTA DE TABELAS

LISTA DE FIGURAS

CAPITULO I

1. Introdução

Os robôs representam hoje uma realidade presente no nosso cotidiano, sendo encontrados nos mais diversos ambientes, desde as águas glaciais das calotas polares até em outros planetas do sistema solar.

Construir um robô com características humanas é uma tarefa difícil. Seres humanos possuem características muito difíceis de serem modeladas. Nosso sistema de visão é apurado, nossa reação a uma situação inesperada é quase que instantânea, nossos movimentos são coordenados inconscientemente para manter o equilíbrio do corpo. Nosso sistema de controle de energia opera de forma bastante otimizada.

Mesmo a realização de uma tarefa simples envolve diversas etapas, como a captura das informações relevantes do ambiente através dos sensores, o processamento e interpretação dos dados pelos algoritmos e a execução de ações através dos atuadores. Neste escopo, a tarefa de navegação consiste na extração da topologia do ambiente de forma a permitir o planejamento de uma trajetória entre dois pontos quaisquer de forma segura e eficiente.

A autonomia de um sistema robótico representa o grau de dependência humana que ele necessita para realizar suas missões e, encontra-se intimamente ligado à sua capacidade sensorial, ao ambiente onde está imerso, ao grau de desenvolvimento dos algoritmos e à sua capacidade em tomar decisões próprias.

Um dos problemas de engenharia mais complexos, desafiadores e propícios à pesquisa de sistemas autônomos é a navegação autônoma de robôs. O problema consiste, basicamente, em desenvolver mecanismos de tomada de decisão para um ou mais robôs móveis, dispostos em um ambiente arbitrário junto ao qual devem atuar de forma autônoma, visando cumprir certas tarefas. Muito embora a navegação de robôs possa ser descrita com base nesta breve definição, existem muitos aspectos de projeto envolvidos: configurações do ambiente, modelo do robô, elenco de tarefas e critérios de desempenho. Sendo assim, o desenvolvimento de sistemas de navegação autônomos – SNA, envolve desafios extremamente complexos para o trabalho de projeto (Piere, 2002).

A complexidade do projeto fica clara quando os aspectos nele envolvidos são esmiuçados. Primeiramente, o ambiente, além de arbitrário, pode apresentar uma conformação não-estruturada e dinâmica, sendo que o robô desconhece, a priori, sua topologia. O robô interage com o ambiente apenas por meio de seus sensores e atuadores, que, como é sabido, são freqüentemente de alcance limitado e sujeitos a ruído. Além de navegar pelo ambiente sem nenhum auxílio externo, o robô deve executar certas tarefas pré-determinadas, possivelmente conflitantes entre si e muitas vezes de execução simultânea. As tarefas podem ser as mais variadas como: desvio de obstáculos, busca de alvos, coleta de objetos, recarga de energia, mapeamento de território, prospecção, etc (Cazangi, 2004).

Particularmente, neste trabalho, o problema é caracterizado da seguinte forma: o ambiente desconhecido possui obstáculos, o robô possui rota pré-determinada e a navegação do robô deve se dar visando à conclusão da rota. Estas duas tarefas de execução concomitantes são claramente conflitantes, requerendo a emergência de mecanismos de coordenação de objetivos para tomada de decisão.

As redes de Petri (Murata, 1989) e suas extensões constituem uma classe de modelos conceituais utilizada na modelagem de diversos tipos de sistemas computacionais paralelos. Por exemplo, têm-se usado redes de Petri para modelar protocolos de comunicação, processos de manufatura e arquiteturas de computadores. Tais modelos possibilitam a representação de sistemas paralelos, os quais podem então ser simulados, por exemplo, nos níveis funcional e lógico, incluindo ou não aspectos de temporização, com um grau razoável de simplicidade.

As redes de Petri têm demonstrado ser eficientes na representação de determinados aspectos de sistemas a eventos discretos. Este fato deve-se, em grande parte, à existência de um conjunto de técnicas para análise estrutural e dinâmica de redes de Petri, permitindo a validação formal de importantes propriedades de um modelo, tais como inexistência de *livelocks* e *deadlocks*, reiniciabilidade do sistema modelado (após a realização de um conjunto de tarefas), limitabilidade dos recursos modelados, dentre outras.

Em resumo, redes de Petri e suas extensões representam um amplo campo de estudo, o qual pode ser explorado no que se refere à área de robótica, buscando-se soluções alternativas no tocante à modelagem, simulação, análise e síntese de tais sistemas robóticos (Jensen, 1997). Tendo em vista todas as vantagens das abordagens baseadas em Redes de Petri, principalmente relacionadas ao poder de análise, este trabalho propõe e desenvolve um sistema de navegação autônomo para robôs móveis baseado em Redes de Petri Coloridas.

1.1 Objetivos do trabalho

Um dos objetivos deste trabalho é modelar, analisar e implementar um Sistema de Navegação Autônomo para guiar um robô móvel por ambientes desconhecidos. Por sua vez, para modelar essa autonomia foram utilizadas as Redes de Petri coloridas, que constituem um formalismo poderoso para essa finalidade.

O sistema de navegação é reativo, ou seja, será estabelecida uma tarefa e o robô não possuirá conhecimentos incorporados a priori, e utilizará apenas estímulos instantâneos capturados do ambiente. O robô é incumbido da tarefa de chegar a um destino pré-estabelecido através de um ambiente desconhecido.

Também é objetivo deste trabalho desenvolver um robô móvel no qual será implementado o modelo de navegação proposto. Este ambiente (robô e modelo de navegação) será então testado e avaliado experimentalmente. O robô deve ser capaz de interagir coerentemente com seu mundo, sendo capaz de recuperar descrições espaciais úteis, usando informações adquiridas pelos sensores e utilizando eficientemente essas descrições com o objetivo de realizar uma tarefa específica. Uma missão típica seria a partida de uma base, a exploração de um ambiente sem colisões com obstáculos e o retorno à base. Ao retornar à base o robô deverá ser capaz de realizar operações de atracação com capacidade para recarregar suas baterias.

1.2 Motivação

Uma das motivações mais importantes para a pesquisa de sistemas de navegação autônomos é a enorme gama de aplicações reais possíveis. Talvez os produtos mais notáveis até hoje obtidos neste domínio tenham sido os robôs *Sojourner, Spirit e Opp*ortunity, usados pela NASA (Cazangi, 2004). O primeiro robô, em 1996, e os outros dois, em 2003, foram enviados a Marte para exploração da superfície do planeta visando coleta de dados. Seus sistemas de navegação, apesar de apenas parcialmente autônomos, são uma mostra das potencialidades da pesquisa em navegação autônoma.

Contudo, ao contrário do que se possa pensar, as aplicações de sistemas autônomos inteligentes não se restringem apenas a projetos futuristas de exploração espacial.

O potencial prático destes sistemas e equipamentos abrange desde tarefas corriqueiras até mesmo situações de extrema periculosidade.

A principal barreira para a criação de robôs cada vez mais avançados está no software. A parte física (hardware), com a tecnologia disponível, pode-se criar robôs poderosos, com diversas capacidades, e relativamente baratos (Borenstein, 1996). Através da ligação de chips de memória e processamento em paralelo é possível atingir elevada capacidade de processamento.

Apesar de haver um grande número de pesquisas em sistemas de navegação, ainda não existe uma solução robusta, capaz de realizar as tarefas exigidas no mundo real com eficiência e flexibilidade.

Entre as aplicações mais comuns para robôs móveis, estão tarefas como vigilância, limpeza de superfícies, auxílio a deficientes físicos, aplicações médicas e transporte de materiais. Os casos de trabalhos que envolvem situações de risco ao ser humano também são variados, como no desarmamento de minas, manutenção de material radioativo, limpeza de encanamentos de esgoto, prospecção submarina, sensoriamento remoto, auxílio em resgates e combate a incêndios.

1.3 Organização do documento

Esta dissertação está organizada como segue:

No Capítulo II é apresentada a teoria que fundamenta este trabalho, os sistemas de navegação autônoma, controle de robôs móveis, Redes de Petri e vários aspectos relacionados às suas aplicações.

Os detalhes de implementação do robô móvel experimental desenvolvido são descritos no Capítulo III.

O modelo proposto, simulações, experimentos realizados e os resultados obtidos são discutidos e analisados no Capítulo IV.

Finalmente, no Capítulo V, apresenta-se uma conclusão do resultado dos trabalhos desenvolvidos nesta dissertação e as propostas para trabalhos futuros.

CAPITULO II

2. Fundamentos teóricos relacionados a robôs móveis - navegação, controle e modelagem

2.1 Robôs móveis autônomos

Karel Capek foi quem utilizou pela primeira vez a palavra robô ao escrever a peça teatral R.U.R. (*Rossum´s Universal Robots*). Esta peça tinha como personagens dispositivos mecânicos com aparência humana que, por serem desprovidos de qualquer tipo de sensibilidade, podiam realizar apenas operações rotineiras de forma automática.

Karel Capek, para designar estes dispositivos, escolheu inicialmente “Labori”. No entanto, como o termo não lhe agradava, acabou por adaptar a palavra Checa “robota”, que significa trabalhador “a força”. O termo “robótica”, que se refere ao estudo e uso de robôs, foi introduzido pelo cientista e escritor Isaac Asimov.

Na sociedade atual é crescente a necessidade de se realizar tarefas com eficiência e precisão. Paralelamente, existem tarefas a serem realizadas em lugares onde a presença humana se torna difícil, arriscada e até mesmo impossível, como é o caso do fundo do mar ou a imensidão do espaço. Para realizar essas tarefas, se faz cada vez mais necessária a presença de robôs, que realizam essas tarefas sem riscos à vida humana e com eficiência tipicamente superiores às alcançadas pelo homem. A robótica é a área da ciência que se preocupa com o desenvolvimento de tais dispositivos. Em particular, o desenvolvimento de robôs móveis autônomos é o que diretamente mais contribui para a realização de atividades até então inviáveis ao ser humano. Exemplos destas atividades incluem a exploração da superfície de outros planetas, a remoção de minas terrestres e submarinas, a limpeza de acidentes nucleares, a explosão de vulcões, entre outras.

Um robô móvel pode ser definido como um veículo provido de um sistema sensorial e um sistema atuador, gerenciado por uma arquitetura de controle capaz de executar missões definidas pelo usuário. Em geral, um robô móvel autônomo apresenta fonte de energia própria e depende de computadores de bordo para a execução de missões. Tais missões consistem de uma série de instruções pré-programados e modificáveis em tempo-real por meio de dados ou informações provenientes dos sensores do veículo (Borenstein, 1996).

Fundamentalmente, todo robô móvel autônomo deve desempenhar um conjunto comum de funções que requerem o uso de dispositivos de entrada (sensores) para detectar sua orientação e um conjunto de dispositivos de saída como controles de superfície ou propulsores que ajustem ou mantenham a orientação. No centro deste sistema está o controlador do veículo, representado, em sua na maioria, por microcomputadores ou microcontroladores. Um conjunto mínimo de dispositivos de saída deve incluir algum tipo de controle móvel de superfície e um sistema de propulsão. Entretanto, muitos controladores de robôs móveis autônomos terão em adicional ao conjunto de dispositivos de entrada e saída, um conjunto de dispositivos específicos necessários para satisfazer às exigências do sistema ou missão especifica. Tais dispositivos podem ser sonares, sensores infra-vermelho, sistemas de posicionamento via satélite (GPS), etc (Borenstein, 1996).

É já comum encontrar em aplicações industriais sistemas semi-autônomos os quais são normalmente utilizados em operações de transporte de peças ou componentes em linha de montagem. Os mais comuns são os conhecidos AGVs (*Automated Guided Vehicle*) que são plataformas móveis que se deslocam seguindo um cabo metálico, uma faixa refletora pintada no pavimento ou outro tipo de guia, sendo o primeiro o mais comum. As trajetórias são estáticas e uma vez definidas basta "desenhá-las", instalando o fio metálico para que estas possam ser executadas. O seguimento é efetuado usando sensores que detectam o afastamento do robô em relação ao fio e com essa informação é feita a correção dos movimentos.

Os AGVs apresentam, no entanto grandes limitações devido ao fato de serem guiados. Uma das desvantagens é a incapacidade de se desviar de quaisquer obstáculos que possam existir no seu caminho, nestes casos a única solução é a de parar, dar um sinal qualquer de alarme e esperar que o obstáculo seja retirado do seu caminho. Uma outra desvantagem é a necessidade de instalar novos fios quando se pretende adicionar ou modificar uma trajetória (Menezes, 1999).

Perante essas limitações torna-se evidente a necessidade de se criar sistemas com um maior grau de autonomia. O aumento do grau de autonomia passa pelo desenvolvimento de novas técnicas, nomeadamente para a geração automática de trajetórias e para a detecção e caracterização de obstáculos.

2.2 Navegação autônoma

Navegação é a ciência ou tecnologia de encontrar posição, o curso e a distância percorrida por um veículo qualquer. Com isso pode-se resumir o sistema de navegação em três questões: Onde estou? Para onde vou? Como irei até lá? Como muitos destes problemas são comuns a todas as entidades de movimentação, ou seja, homens com ou sem veículos, animais e veículos autônomos, e tendo como base que tanto homens quanto animais despendem uma grande parte de sua infância para aprender o básico sobre navegação, existem diversas formas envolvendo a solução desses problemas (Borenstein, 1996).

Conseguir desenvolver métodos de navegação autônoma é um dos principais objetivos da investigação atual em robótica móvel, uma vez que sem esta capacidade as plataformas móveis vêem a sua utilização limitada a sistemas guiados ou telecomandados (Borenstein, 1996).

Os sistemas guiados, onde as plataformas móveis seguem fio ou marcas no pavimento, têm sido bastante utilizados em linhas de produção para o transporte de peças ou produtos. No entanto, dado que os trajetos são pré-fixados através da instalação dos guias referidos, o aparecimento de um obstáculo na trajetória constitui um problema, uma vez que apesar deste poder ser detectado pela plataforma (com auxílio de sensores apropriados), esta não possui, normalmente, capacidade de ultrapassá-lo. Outra desvantagem destes sistemas tem a ver com a constante alteração da disposição das linhas de montagens em muitas das fábricas atuais. A utilização destas plataformas requeria que os "guias" fossem alterados sempre que a disposição da linha de montagem fosse alterada o que é, em muitos casos impossível, visto que estes se encontram enterrados no pavimento.

No que diz respeito aos sistemas telecomandados tem-se verificado a necessidade de dotá-los de certo grau de autonomia de forma a possibilitar a sua utilização quando existem grandes atrasos nas comunicações ou para reduzir o "stress" do operador. Um exemplo dessa necessidade foi a recente missão a Marte da plataforma *Sojourner*, que se a mesma não dispusesse de certo grau de autonomia seria completamente impossível a sua utilização, dado que as comunicações apresentavam um atraso bastante elevado e uma largura de banda extremamente reduzida (Cazangi, 2004).

Apesar das tarefas de navegação parecerem simples, dada a facilidade com que os seres humanos as executam, a criação de sistemas capazes de executá-las está longe de ser um problema trivial devido à complexidade das operações envolvidas (Cazangi, 2004).

Para melhor perceber quais os mecanismos requeridos para conduzir uma plataforma entre dois pontos arbitrários, analisam-se as diferentes fases necessárias: A primeira fase é a identificação das posições de partida e destino, feito isso é necessário estabelecer um caminho que una as duas posições. Para o planejamento de uma trajetória que conduza o veículo até a posição objetivo é necessário existir um "mapa" onde possam ser identificadas as zonas livres (sem obstáculos) e onde o veículo caiba. Definido o caminho é necessário converter os diversos segmentos que o compõe em primitivas de comando da plataforma e executá-las de forma seqüencial.

A definição de um caminho pode ser feita com base em primitivas geométricas (tais como: segmentos de reta) que são unidas através de pontos de passagem bem definidos, desde que o ambiente seja também descrito através de primitivas geométricas num espaço cartesiano. Por outro lado, se a descrição do ambiente é feita com base em informação topológica, o planejamento do caminho implica na definição de um conjunto de ações de navegação e percepção. Esta última abordagem é a normalmente utilizada por nós próprios quando descrevemos o caminho entre quaisquer dois sítios, sendo as descrições geométricas utilizadas apenas em alguns casos particulares (Borenstein, 1996).

Até aqui não se tem mais que sistemas guiados uma vez que o planejamento das trajetórias também antecede a instalação dos fios ou das pinturas de guiamento, e as saídas dos sensores que detectam os desvios em relação ao guia são utilizadas para gerar os comandos para o sistema de tração.

Basicamente o que se pretende introduzir de novo é um modelo de um SNA por RPC para um ambiente desconhecido, com a capacidade de reação à presença de qualquer tipo de obstáculos. Para satisfazer este requisito é necessário utilizar sistemas sensoriais que permitam não só detectar a presença de obstáculos, mas também descrevê-los pelo menos parcialmente de forma a permitir a ultrapassagem dos mesmos.

2.2.1 Problema de navegação

Dada a seguinte colocação:

"Dado um ambiente conhecido ou não, qualquer que seja o ponto de partida onde se encontre o robô este deve ser capaz de atingir um ponto destino que lhe seja dado."

Um robô móvel capaz de resolver este problema simples necessita de (Borenstein, 1996):

1. **Mapas** – Dispor de uma representação do ambiente à qual chamamos habitualmente mapa que poderá ser fornecido "a priori" ou construído pelo próprio robô após ter explorado o ambiente inicialmente desconhecido.

2. **Planejamento** – Utilizando mapas, estabelecer uma trajetória entre o ponto de partida e o ponto de chegada.

3. **Sistemas de Controle** – Seguir a trajetória planejada convertendo o caminho em comandos de movimento e verificar a sua execução. Deverá ainda evitar a colisão com quaisquer obstáculos inesperados e, sempre que possível, contorná-los de forma a regressar à execução da trajetória. Sempre que se verificar que não é possível transpor um obstáculo é necessário invocar novamente o planejamento para obter uma trajetória alternativa que conduza ao objetivo.

4. **Localização** – A maioria dos robôs utilizam para localização os sistemas odométricos para estimar a posição do robô em cada instante, estes, devido a fatores como o escorregamento das rodas ou variação do diâmetro das mesmas, vão acumulando erros ao longo da execução de uma trajetória. Por estas razões é freqüente utilizar sistemas de localização complementares destinados a compensar os erros de odometria ou dos sistemas inerciais, como câmeras e marcas no ambiente.

5. **Percepção** – Sistemas sensoriais que permitam identificar características no ambiente que rodeia o robô e que uma vez detectadas as suas posições relativas ao robô possam ser utilizadas pelo sistema de localização, ou obstáculos cuja posição relativa deve ser enviada ao sistema de controle para evitar a colisão. São os sistemas sensoriais que permitem ao robô a sua adaptação a alterações no ambiente conhecido, sendo por isso peças fundamentais para se obter a autonomia desejada.

2.2.2 Navegação reativa, planejada e híbrida

Para o robô navegar em ambientes, detectar e desviar de obstáculos, localizar-se e planejar sua trajetória, um certo grau de autonomia é requerido. Também certa autonomia é exigida quando o robô deve reagir com rapidez às mudanças dinâmicas do ambiente. As arquiteturas de programação de robôs são também fundamentais para a realização de um sistema mais autônomo.

Assim, a execução de movimentos pelos robôs móveis pode ser dividida em três abordagens: reativa, planejada e híbrida.

a) Navegação Reativa

Para não colidir com eventuais obstáculos, os deslocamentos executados pelo robô dependem da informação sobre o ambiente que o rodeia, adquirida através de sistemas sensoriais como mostrado na Figura 2.1. Na arquitetura reativa, o robô navega com base em comportamentos, que são atitudes do robô face à dinâmica do ambiente. Com esse tipo de arquitetura um robô navega sem que seja sabido qual é a possível trajetória do robô a ser realizada durante a navegação, pois esta depende de qual(is) comportamento(s) irá(ão) atuar.

A navegação reativa é considerada uma navegação de baixo nível, em que o robô não tem nenhum conhecimento sobre o ambiente (não contém um mapa), nem informação sobre a posição final a atingir.

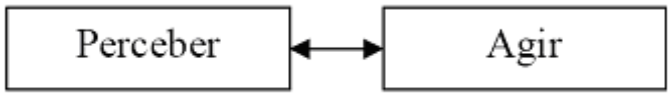

Figura 2.1: Ciclo sentir – agir.

A abordagem reativa, também conhecida como arquitetura comportamental, é dividida nos diversos comportamentos que se deseja para o veículo. Normalmente, uma missão é descrita como uma seqüência de fases com um conjunto de comportamentos ativos. Os comportamentos reagem continuamente aos estímulos detectados pelo sistema de percepção. O comportamento global do veículo surge da combinação dos comportamentos elementares ativos.

Esta arquitetura foi introduzida por Rodney Brooks (Brooks, 1986), a qual é caracterizada por rejeitar, ou minimizar as informações de estados internos, bem como as informações sobre um modelo do mundo. Em sua forma original, os comportamentos podiam suprimir uns aos outros, sem respeitar hierarquias, o que pode levar a um comportamento errático. Um avanço nesta proposta é a arquitetura de fusão de comportamentos reativos, cujo diagrama é mostrado na Figura 2.2. Nesta arquitetura, existe um módulo de fusão e um módulo supervisor, que são responsáveis por avaliar os sensores e decidir, entre os comportamentos aplicáveis em dado momento, qual terá o controle do robô. Sua vantagem é que apenas um comportamento, em geral pequeno, é ativado por vez e nenhum modelo do mundo é criado e mantido, reduzindo assim o tempo de processamento.

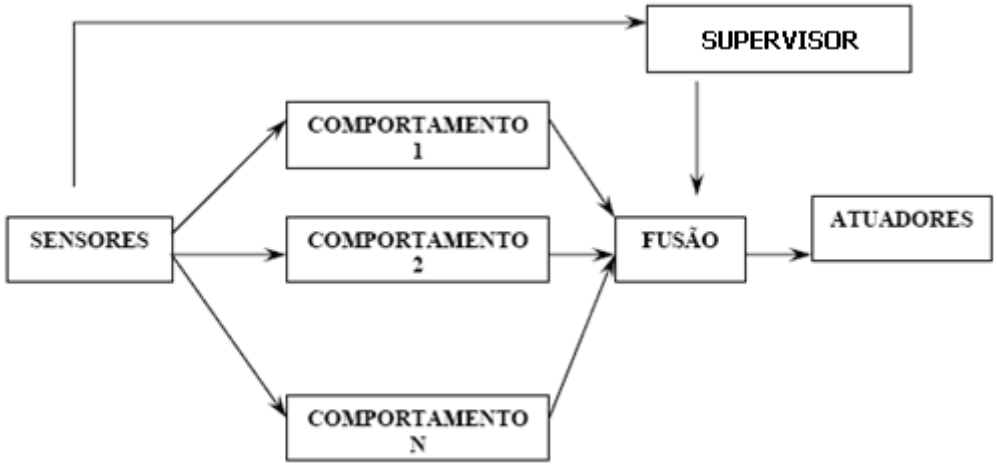

Figura 2.2: Diagrama da arquitetura reativa de fusão de comportamentos.

Um exemplo deste paradigma encontra-se no trabalho de Mataric (1990), que criou um robô chamado Toto, que tinha a missão de navegar por um ambiente inicialmente desconhecido, utilizando para isso, um sistema baseado em comportamento (Brooks, 1986). O objetivo era investigar como o método se comportaria nesta tarefa especifica de um robô real. O robô de Mataric segue paredes e trafega em apenas dois sentidos. Quando se depara com um mundo dinâmico, o sistema se comporta de forma satisfatória, porque ele atualiza o grafo que representa o ambiente. Finalmente, as regras contidas nos comportamentos de baixo nível tratam os obstáculos móveis de forma autônoma, evitando colisões. Entretanto, sua incapacidade de se localizar em espaços abertos, e seu método de navegação, que não garante encontrar o alvo, tornam sua utilização pouco recomendável para tarefas do mundo real.

b) Navegação Planejada

Os deslocamentos a executar pelo robô foram previamente estabelecidos com base numa configuração conhecida do ambiente. Na navegação de robôs baseada na arquitetura planejada, ou clássica, todas as tarefas requeridas para a navegação são baseadas principalmente em um processo de planejamento e em um modelo do ambiente de execução da missão (Figura 2.3). Através do modelo do ambiente, a missão é planejada e um sistema de controle executa o plano de ação obtido. Arquiteturas desenvolvidas segundo esta abordagem geralmente apresentam comportamento previsível, porém não se adaptam com facilidade a ambientes muito dinâmicos, nos quais as mudanças ocorrem rapidamente. De acordo com essa arquitetura, algoritmos de planejamento de trajetória, auto-localização e detecção de obstáculos, são implementados para a realização dessas tarefas.

A abordagem planejada é caracterizada por construir um modelo do mundo constantemente utilizado pelo robô para realizar suas tarefas. O processamento é realizado em ciclos, chamados ciclos sentir-modelar-planejar-agir, apresentado na Figura 2.3. Em cada ciclo, o robô obtém dados do ambiente através de seus sensores, realiza um processamento nestes dados brutos, transformando-os em informação útil para atualizar seu modelo de mundo. Então, é feito um planejamento das próximas ações, e a próxima ação imediata é enviada aos atuadores (Thrun e Bucken, 1996).

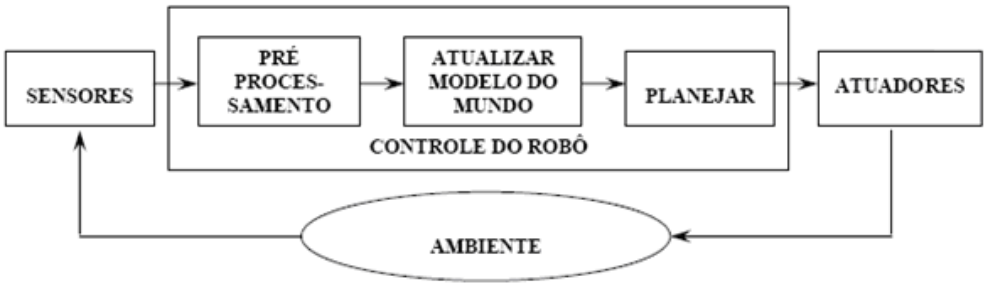

Figura 2.3: Ciclo sentir – modelar – planejar – agir.

Os métodos específicos, baseados na arquitetura planejada apresentam uma série de dificuldades. A primeira delas é que estes métodos não conseguem tratar ambientes dinâmicos, p.e., mapas não modelam obstáculos móveis. A modelagem de obstáculos semi-móveis, como portas que podem estar abertas ou fechadas, é um pouco mais fácil (Thrun e Bucken, 1996).

Modelar obstáculos móveis como seres humanos é um problema em aberto em navegação. Uma segunda dificuldade é sobre a sintonia dos sensores, que se não são perfeitos, pelo menos devem ser estáveis, ou seja, não sofrer deterioração no desempenho à medida que o tempo passa. Isto é um requisito pesado para qualquer peça mecânica. A opção é utilizar modelos para calibração de sensores que são ajustados enquanto o robô está operando, que é uma área de pesquisa ativa.

c) Navegação Hibrida

Os movimentos planejados são adaptados ao ambiente através da utilização de informação sensorial. Estas arquiteturas são as que possuem maior número de implementações. Elas unem características das arquiteturas planejadas e reativas (Figura 2.4).

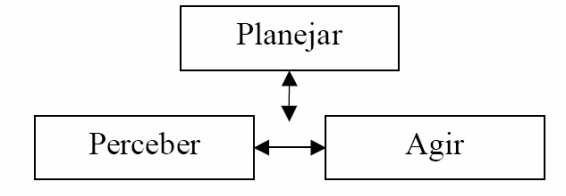

Figura 2.4: Ciclo sentir – planejar se der tempo – agir.

As duas abordagens anteriores apresentam vantagens e desvantagens. Enquanto que o paradigma reativo resulta em sistemas mais simples e rápidos, ele tem menor poder de realizar inferências e trabalhar com conhecimentos adicionais. Já os sistemas que seguem a arquitetura planejada dispõem de muito conhecimento, mas são lentos e têm grande dependência de sensores e atuadores bem calibrados, o que é muito difícil de se obter em situações reais.

A abordagem híbrida procura combinar os pontos fortes dos dois paradigmas para oferecer uma solução melhor do que cada um deles separadamente (Cazangi, 2004).

2.3 Métodos para controle de robôs móveis

Os robôs móveis são dotados de um sistema de controle de movimento (Figura 2.5) que os tornam capazes de navegar através do seu ambiente de trabalho, interagindo com este ambiente na realização de tarefas pelo uso de recursos próprios de sensoriamento e tomada de decisão. O controle de movimento pode ser definido como um sistema de integração tecnológica resultante da teoria de controle, eletrônica de potência, e controle por microcomputador para obter precisão no controle de torque, velocidade, e/ou posição do sistema mecânico responsável pela locomoção do robô (motores, sistema hidráulico, sistema pneumático, rodas, esteiras, pernas) (Borenstein, 1996). Todos os movimentos realizados pelo robô são obtidos através do controle do acionamento dos motores de tração. Servomotores equipados com encoder de alta resolução têm tornado o sistema de controle de movimento incremental a melhor abordagem tecnológica para obter precisão e controle do sistema de acionamento. Em geral, para robôs pequenos, como é o caso deste trabalho, é possível utilizar controladores convencionais do tipo PID (Lages, 1998).

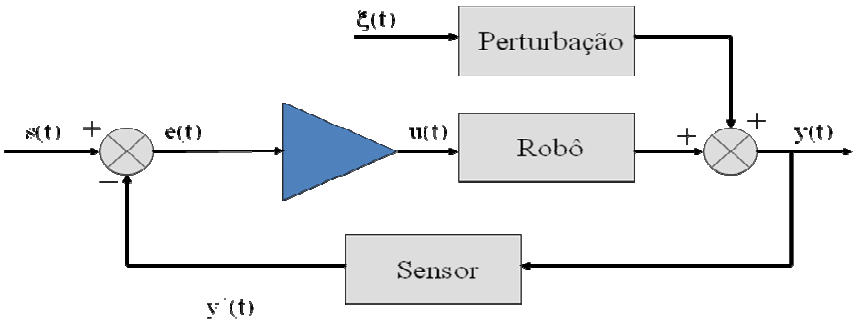

Figura 2.5: Sistema de controle básico para um robô móvel.

Um robô móvel é um sistema não holonômico, isto é, existem restrições impostas ao movimento não integráveis, ou seja, as restrições não podem ser escritas como derivadas temporais de funções das coordenas generalizadas. Embora a dinâmica desses sistemas tenha sido bastante estudada, apenas no final da década de 90 que começaram a surgir estudos sobre o controle destes sistemas. É importante ressaltar que algoritmos padrões para controle de manipuladores robóticos sem restrições cinemáticas não são aplicáveis no caso de robôs móveis (Lages, 1998).

Existem dois tipos principais de modelos utilizados para descrever o comportamento de robôs móveis: cinemático e dinâmico. Em Campion (1995) são apresentados o modelo cinemático e o dinâmico de um robô com acionamento diferencial. A mesma formulação se aplica para o robô móvel a ser utilizado como estudo de caso neste trabalho.

O modelo cinemático descreve o robô em função da velocidade e orientação das rodas, enquanto o modelo dinâmico (inclui as equações do movimento de Newton e Euler, bem como o modelo dinâmico dos atuadores) descreve o robô em função das forcas generalizadas aplicadas pelos atuadores, por exemplo torque nas rodas.

Considera-se que o robô é construído de um corpo rígido e de rodas não deformáveis e movimenta-se em um plano horizontal. O contato entre as rodas e o plano é pontual. O movimento do robô é realizado através de atuadores que fornecem torque para a rotação das rodas em uma configuração diferencial. O acionamento do robô se dá por aplicação da tensão de armadura nos motores acoplados às rodas fixas do mesmo. Tal acionamento pode ser considerado como aplicação de torque nas rodas, levando-se em consideração que as variações na corrente de armadura dos motores têm um comportamento muito mais rápido que as variações na velocidade angular dos eixos do motor (e conseqüentemente das rodas) e, portanto podem ser desprezadas.

Podem-se classificar as metodologias de controle em malha fechada para robôs móveis em basicamente três tipos: estabilização em um ponto, rastreamento de trajetória e seguimento de caminhos (Kuhne, 2005).

a) Estabilização em um ponto

O objetivo é fazer com que o robô estabilize em um ponto fixo do espaço de configuração, ou seja, fazer com que

$$x(k) - x_{ref} = 0 \tag{2.1}$$

em que $x_{ref} = [x_{ref}\, y_{ref}\, \theta_{ref}]^T$ a trajetória de referência (posição e orientação).

Para um sistema linear invariante no tempo, se os autovalores instáveis são controláveis, um ponto de equilíbrio pode ser assintoticamente estabilizado por uma realimentação estática, suave e invariante no tempo. Entretanto, para sistemas não lineares e com restrições não holonômicas, isto não é mais possível (Brockett, 1983). Consequentemente, ferramentas lineares antes utilizadas não podem mais ser consideradas, nem localmente. Neste caso, usualmente, leis de controle variantes no tempo ou não suaves são utilizadas a fim de transpor as restrições de Brocket (Brockett, 1983).

b) Rastreamento de trajetória

Este problema é bem parecido com o de estabilizar o robô em um ponto, a diferença é que agora o objetivo é fazer com que o robô siga uma trajetória de referência pré-determinada e variante no tempo, que pode ser formulado então como fazer com que

$$x(k) - x_{ref}(k) = 0, \qquad x_{ref}(k) = \begin{bmatrix} X_{ref}(k) \\ Y_{ref}(k) \\ \theta_{ref}(k) \end{bmatrix} \tag{2.2}$$

em que $x_{ref}(k)$ é a trajetória de referência.

Nota-se que a expressão (2.2) é bem semelhante à expressão (2.1). A diferença é que agora o ponto onde o robô deve convergir move-se ao longo do tempo. Assim, comumente associa-se à trajetória de referência *virtual*, que possui o mesmo modelo cinemático do robô a ser controlado.

$$\begin{cases} x_{ref}(k+1) = x_{ref}(k) + u_{ref}(k)\cos\theta_{ref}(k)T \\ y_{ref}(k+1) = y_{ref}(k) + u_{ref}(k)\sin\theta_{ref}(k)T \\ \theta_{ref}(k+1) = \theta_{ref}(k) + w_{ref}(k)T \end{cases} \tag{2.3}$$

ou em uma forma mais compacta,

$$x_{ref}(k+1) = f(x_{ref}(k), u_{ref}(k)) \tag{2.4}$$

em que $u_{ref}(k) = [u_{ref}(k) \;\; w_{ref}(k)]^T$ são entradas de controle de referência.

Assume-se então que, para entradas de referência não nulas, deseja-se calcular uma lei de controle que faça com que o erro entre o robô e a referência seja nula. Este problema será abordado neste trabalho.

c) Seguimentos de caminhos

Neste caso, à semelhança do caso anterior, também deseja-se que o robô convirja para uma trajetória de referência, mas geralmente, este problema é menos restritivo, pois há uma especificação temporal para que esta convergência seja alcançada. Ainda, usualmente considera-se que a velocidade linear é mantida constante e a convergência é obtida apenas através da velocidade angular.

A seguir serão apresentadas algumas técnicas de controle de robôs móveis não holonômicos: controle variante no tempo, controle não suave e controle Preditivo Baseado em Modelo (MPC)

a) Controle variante no tempo

Leis de controle variantes no tempo possuem a desvantagem de apresentar baixas taxas de convergência e trajetórias altamente oscilatórias (Campiom, 1995), o que, em uma aplicação real, pode se tornar até não factível, dependendo das amplitudes e taxas de variação das entradas de controle.

Considera-se o sistema

$$\begin{cases} \dot{x} = u \;\; \cos\theta \\ \dot{y} = u \;\; \text{sen}\,\theta \\ \dot{\theta} = w \end{cases}$$

em que $[x \quad y \quad \theta]^T$ representa a configuração do robô em relação ao espaço de configuração e $[u \quad w]^T$ são as velocidades linear e angular, respectivamente, conforme o modelo cinemático em Campion (1995).

Em Sanson (1990) utiliza uma lei de controle variante no tempo para estabilizar o sistema assintoticamente. Fazendo uma transformação de coordenadas da base cartesiana (x, y, θ) para uma base variante no tempo (z_1, z_2, z_3). Para este caso, conseguiu-se apenas minimizar um pouco os problemas de movimento altamente oscilatório e um tempo de acomodação elevado.

Em Tell (1995) utiliza uma transformação canônica do sistema para a forma encadeada. A vantagem é que a lei de controle pode ser facilmente generalizada para vários tipos de sistemas não holonômicos. Assim, novamente as características das leis de controle variante no tempo são observadas: trajetórias de estado altamente oscilatórias e um elevado tempo de acomodação.

b) Controle não suave

As leis de controle não suave podem ser subdivididas em leis de controle por partes ou de modos deslizantes. A vantagem do controle não suave é que o mesmo pode superar as desvantagens comumente associadas ao controle variante no tempo, ou seja, baixa taxa de convergência e trajetórias de estado oscilatórias (Lages, 1998).

Em Lages (1998) o modelo cinemático é transformado em coordenadas polares através da uma transformação descontínua. Assim, o sistema na base polar de coordenadas torna-se descontínuo, sendo possível assim estabilizar o sistema em um ponto através de uma lei de controle suave e invariante no tempo, então, através de uma determinada função de Lyapunov chega-se a uma lei de controle. Esta lei, quando transformada para a base original de coordenadas cartesianas, torna-se descontínua. Os resultados mostram que esta lei de controle gera uma trajetória de estado bastante suave e alta taxa de convergência.

Em Sordalen (1993) também utiliza uma transformação descontínua de coordenadas. A lei de controle determinada estabiliza o robô em um ponto exponencialmente. Da mesma forma que em (Lages, 1998), esta lei de controle também gera uma trajetória bastante suave, sem movimentos oscilatórios desnecessários e com uma taxa de convergência bastante elevada.

c) Controle Preditivo Baseado em Modelo (MPC)

O MPC tem como principal característica a utilização de um modelo para prever o comportamento do sistema dentro de um determinado horizonte de tempo futuro. Uma função de custo, válida dentro deste horizonte, é minimizada com relação a certas variáveis de decisão, respeitando-se um conjunto de restrições impostas. Obtém-se assim, para cada instante atual, uma seqüência de controle ótima com relação à função de custo. Embora a predição e a otimização sejam feitas dentro de toda a extensão do horizonte, apenas a ação de controle atual é efetivamente aplicada ao sistema. Uma desvantagem do MPC refere-se ao esforço computacional necessário para o cálculo da lei de controle, já que um problema de otimização precisa ser resolvido *on-line*, em cada instante de amostragem (Kuhne, 2005).

Em Kuhne (2005) são apresentados diversos algoritmos de controle preditivo baseados em modelo para o controle de um robô móvel dotado de rodas e com restrições não holonômicas. Através de extensivos resultados experimentais, é mostrada a eficácia da técnica, onde foi feita uma análise comparativa com algumas leis clássicas de controle de robôs móveis.

2.4 Redes de Petri (RP)

As redes de Petri foram criadas a partir da tese de doutorado do pesquisador alemão Carl Adam Petri, intitulada "*Kommunication mit Automaten*" ("Comunicação com Autômatos"). Desde o princípio, as redes de Petri (RP) e suas extensões objetivaram a modelagem de sistemas com componentes concorrentes.

As primeiras aplicações de RP aconteceram em 1968, no projeto norte-americano *Information System Theory*, da ADR (*Applied Data Research*, Inc.). Muito da teoria inicial, da notação e da representação de RP foi desenvolvido neste projeto e foi publicado no seu relatório final. Este trabalho ressaltou como RP poderiam ser aplicadas na análise e modelagem de sistemas com componentes concorrentes (Peterson, 1981).

As aplicações de RP aumentaram consideravelmente na década de 1980, com o surgimento das chamadas Redes de Petri de alto nível (por exemplo, as *Coloridas e as Numéricas*). Tais redes acrescentaram uma grande força descritiva ao processo de modelagem, através do uso de fichas com identidade e, conseqüentemente, do uso de conjuntos de fichas. Desta forma, RP atingiram outras áreas, como automação de escritórios, bancos de dados, inteligência artificial e sistemas de informação.

A consolidação de RP se deu na década de 1990, fomentada pela segunda versão das redes Coloridas, desenvolvidas por Kurt Jensen, da *Århus Universitet* (Dinamarca). Além de trabalharem com fichas diferenciáveis, tais redes apresentam tratamento de aspectos temporais. Ainda naquela década, vários pesquisadores demonstraram a eficiência e potencial de RP na modelagem de sistemas a eventos discretos. Este fato deve-se, em grande parte, à existência de um conjunto de técnicas para análise estrutural e dinâmica de RP, que permitem a validação formal de importantes propriedades de um modelo: vivacidade, segurança, inexistência de conflitos, dentre outras.

2.4.1 Composição básica de Redes de Petri

Uma RP é representada por um grafo (pode ser representada também matematicamente por meio de matrizes), sendo composta de elementos de dois tipos: lugares, indicados por nós elípticos ou circulares, e transições, indicadas por nós retangulares ou em forma de barra. Arcos orientados ligam lugares às transições ou vice-versa, nunca entre elementos do mesmo conjunto. Portanto, de acordo com o exposto, uma RP é um grafo orientado bipartido. Na Figura 2.6 é fornecida a composição básica de uma RP sem considerar, por enquanto, qualquer interpretação ou regra de funcionamento dada à mesma.

Figura 2.6: Redes de Petri equivalentes: 2 lugares e 2 transições.

Em relação a uma transição, os arcos são classificados em arcos de entrada, indicados pela seta chegando à transição (retângulo na Figura 2.6), e arcos de saída, indicados pela seta saindo da transição. Analogamente, classificam-se os lugares como sendo de entrada ou de saída.

De acordo com o tipo da rede, lugares e transições recebem conotações diferentes. Na maioria das interpretações de redes, os lugares representam elementos do tipo estado e as transições representam elementos do tipo ação.

2.4.2 Características e vantagens das Redes de Petri

Em Jensen (1997) destacam-se as seguintes características e vantagens desta técnica:

- Representam à dinâmica e a estrutura do sistema modelado segundo o nível de detalhamento desejado;
- Identificam estados e ações de modo claro e explícito, facilitando com isto a monitoração do sistema em tempo real.
- Têm a capacidade para representar de forma natural as características dos Sistemas a Eventos Discretos - SED, (sincronização, assincronismo, concorrência, causalidade, conflito, compartilhamento de recursos, etc.).

- Associam elementos de diferentes significados numa mesma representação, ou segundo o propósito do modelo (avaliação de desempenho, implementação do controle, etc).
- Oferecem um formalismo gráfico que permite a documentação e monitoração do sistema, facilitando assim o diálogo entre projetista e as pessoas que participam no processo de projeto ou de análise do comportamento do sistema (projetista, operador, gerente, etc.).
- Constituem-se como uma teoria muito bem fundamentada para a verificação de propriedades qualitativas (sem restrições no tempo).
- Possuem uma semântica formal e precisa que permite que o mesmo modelo possa ser utilizado tanto para ánalise de propriedades comportamentais (ánalise quantitativa e/ou qualitativa) e avaliação do desempenho, assim como para a construção de simuladores a eventos discretos e controladores (para implementar ou gerar códigos para controle de sistemas).
- Incorporam conceitos de modelagem do tipo refinamento ("*top down*") e do tipo composição modular ("*bottom up*") através de técnicas como: modularização, reutilização e refinamento.

2.5 Redes de Petri Coloridas (RPC)

RPCs consistem em uma técnica formal com embasamento matemático para descrição de sistemas, especialmente apropriadas para modelagem de sistemas a eventos discretos (Jensen, 1997). Dentre suas principais características, podem-se citar:

- Possuem representação gráfica;
- São de fácil aprendizado;
- Funcionam como linguagem de comunicação entre especialistas de diversas áreas;
- Permitem descrição dos aspectos estáticos e dinâmicos do sistema a ser representado;
- Usufruem do formalismo matemático, o que permite uso de ferramentas de análise;
- Disponibilidade de técnicas de análise.

As RPCs são extensões das RPs, adicionando-lhes recursos para a definição e manipulação de tipos de dados. Historicamente, convencionou-se designar os tipos em RPC como "cores", em contraste com as redes originalmente propostas, nas quais todas as fichas

são iguais, ou seja, sem cor. De um ponto de vista abstrato, as RPCs estão para as redes de Petri, assim como as linguagens de programação estão para a linguagem de máquina.

Para introduzir as RPCs, na Figura 2.7 é apresentado informalmente um exemplo de uma RPC com os seus respectivos elementos gráficos e suas descrições.

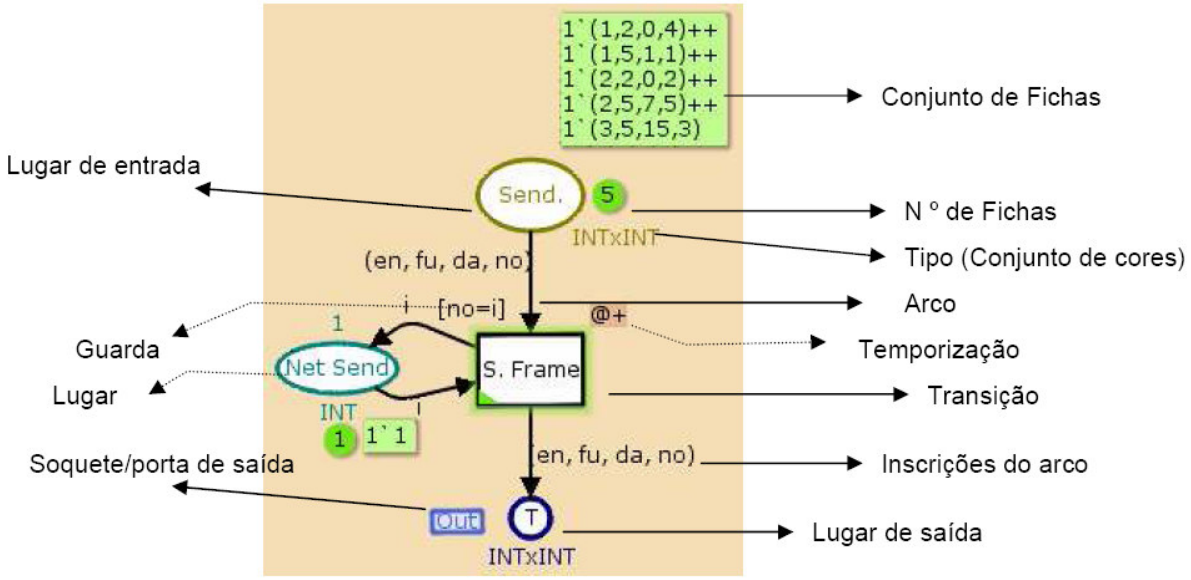

Figura 2.7: Representação de uma Rede de Petri Colorida.

Os elementos que constituem esse modelo serão explicados posteriormente, à medida que forem introduzidos os conceitos relacionados.

- **Muti-conjuntos**

Para a definição de RPC é necessário o conceito de multi-conjuntos de elementos. Um multi-conjunto é uma entidade matemática semelhante ao conjunto, exceto pelo fato de que múltiplas ocorrências de um mesmo elemento são permitidas. Também são denominados como *bags*, suítes ou coleções.

- **Cores**

Como mencionado anteriormente, dentro do contexto de RPC o termo cor é utilizado como sinônimo de tipo de dados. Formalmente, um conjunto de cor C é um conjunto não vazio, possivelmente com infinitos elementos. Uma RPC possui um conjunto finito não vazio de cores $\sum$. No modelo da Figura 2.7 existe apenas um conjunto de cor, do tipo INTxINT.

- **Linguagem de Anotação dos Arcos**

De forma análoga ao que ocorre com linguagens de programação em geral, para a definição das cores de especificação e para a manipulação de seus elementos é necessária a utilização de uma linguagem com uma sintaxe bem definida. A escolha dessa linguagem não altera a semântica e a expressividade da técnica. No entanto, a linguagem impacta de forma decisiva no desempenho das técnicas de análise, e procura-se utilizar linguagens que possuam semântica bem definida. Ao propor as RPC Jensen utiliza a linguagem CPN-ML, a qual é derivada da linguagem Standard Meta-Language (SML), uma linguagem funcional e fortemente tipada. SML é uma sintaxe concreta para o Cálculo Lambda Tipado. Isso implica que é possível definir todos os tipos de funções matemáticas computáveis (Jensen, 1997).

- **Elementos Estruturais**

Uma RPC é formada por elementos que representam os aspectos estruturais e elementos que definem os aspectos dinâmicos. Os elementos estruturais definem componentes estáticos do sistema, e são eles: os lugares, as transições e os arcos.

Em geral, os lugares modelam os elementos passivos do sistema, tais como condições, recursos, repositórios de dados, etc. A cada lugar é associado um conjunto de cores. Um lugar pode conter um multi-conjunto de elementos de um mesmo conjunto. Graficamente, lugares são representados por elipses ou círculos e o nome do conjunto de cores associado é posicionado próximo ao respectivo lugar (Figura 2.7). Na Figura 2.7, pode-se observar os seguintes lugares: Send, Net Send, T.

Em geral, as transições modelam os elementos ativos do sistema, tais como eventos, ações e atividades, etc. Graficamente, uma transição é representada por um retângulo. Na Figura 2.7, é exemplo de transição: S. Frame.

Os arcos representam os relacionamentos entre os elementos do sistema. Um arco pode relacionar:

- Um lugar p a uma transição t. Neste caso, o arco (p,t) é um arco de entrada. Analogamente, p é um lugar de entrada de t e t é uma transição de saída de p. Os arcos de entrada de um modelo podem representar pré-condições, valores de entrada, recursos necessários, etc. No modelo da Figura 2.7, um exemplo de arco de entrada é o arco que liga Send a S. Frame.
- Uma transição t a um lugar p. Neste caso, o arco (t,p) é um arco de saída. Analogamente, p é um lugar de saída de t e t é uma transição de entrada de p. Os arcos de saída de um modelo podem representar pós-condições, valores de saída, recursos

produzidos, etc. No modelo da Figura 2.7, um exemplo de arco de saída é o arco que liga S. Frame a T.

- **Elementos Dinâmicos**

Os elementos dinâmicos de um modelo são conceitos diretamente relacionados à execução da rede, e são eles: anotações de arcos, expressões de guarda e expressões de inicialização.

Uma *anotação de arco* é uma expressão que, quando avaliada, representa um multi-conjunto de elementos do conjunto de cores associado ao lugar ligado a esse arco e é utilizada para quantificar e qualificar a relação entre os lugares e as transições. Anotações de arcos podem ser formadas por valores constantes, variáveis, funções, etc. Graficamente, a anotação é posicionada próxima ao respectivo arco. No modelo da Figura 2.7, por exemplo, a anotação de arco de entrada que liga S. Frame a T é (en, fu, da, no).

Expressões de guarda são associadas às transições e, quando avaliadas, representam valores booleanos. Em geral, as expressões de guarda utilizam as variáveis dos arcos de entrada e saída da transição. Uma expressão de guarda é a condição que deve ser satisfeita para que a transição possa ocorrer. Caso a expressão de guarda não seja explicitamente declarada, assume-se o valor verdadeiro. Graficamente, a expressão de guarda é posicionada, entre colchetes, próxima à respectiva transição. Deve-se observar que no modelo da Figura 2.7, não existem expressões de guarda.

Expressões de inicialização são valores (fichas) associados aos lugares que representam um multi-conjunto de elementos do conjunto de cores associado ao respectivo lugar. Uma expressão de inicialização informa o conteúdo de um lugar quando o sistema é inicializado (estado ou marcação inicial). Quando não especificada, a expressão de inicialização assume como valor *defaut*, o multi-conjunto vazio. Graficamente, a expressão de inicialização é posicionada próxima ao respectivo lugar em itálico. No modelo da Figura 2.7, por exemplo, a expressão de inicialização do lugar *Net Send* é 1.

2.5.1 Definição formal

Uma RPC é uma tupla CPN = ($\sum$, P, T, A, N, C, G, E, I), em que (Jensen, 1997):

(i) $\sum$ é um conjunto finito não vazio de tipos, também chamados de ***conjuntos de cores***;

(ii) P é um conjunto finito de ***lugares***;

(iii) T é um conjunto finito de ***transições***;

$$P \cap T = \Phi$$

(iv) A é um conjunto finito de ***arcos***, de forma que:

$$P \cap A = T \cap A = \Phi$$

(v) N é uma função de ***nós***, tal que:

$$N : A \rightarrow P \times T \cup T \times P$$

(vi) C é uma função de ***coloração***, tal que:

$$C : P \rightarrow \sum$$

(vii) G é uma função de ***guarda***, que associa a cada $t \subset T$ uma expressão do tipo booleano;

(viii) E é uma função de ***anotação de arcos*** que associa a cada $\alpha \subset A$ uma expressão do tipo lugar relacionado;

(ix) I é uma função de ***inicialização*** que associa a cada $\rho \subset P$ uma expressão do tipo C(p).

2.5.2 Execução de uma Rede de Petri Colorida

Em geral, quando os aspectos dinâmicos da especificação são importantes para a análise de propriedades, é necessário que a semântica seja estabelecida pela definição das regras de como a especificação pode ser simulada. Dada uma seqüência de entradas, a simulação consiste em calcular a seqüência de configurações nas quais a especificação resultará.

No contexto de RPC, a simulação da especificação é controlada pela regra de execução, a qual se baseia nos conceitos de fichas e marcações (ambos relacionados às configurações), ligações e disparos (ambos relacionados às transições entre configurações).

Uma ficha (do inglês, *token*) é um item mínimo em uma especificação baseada em RPC. Uma ficha está associada a um lugar da rede e deve ser um elemento do conjunto de cores associado a esse lugar. Portanto, um lugar pode estar associado a um multi-conjunto de fichas de seu respectivo tipo.

Uma marcação é caracterizada pelo conteúdo de todos os lugares da rede. Formalmente, uma marcação m é uma função que mapeia o lugar p da rede em um multi-conjunto de elementos de seu respectivo conjunto de cores associado. Dessa forma, uma marcação é uma configuração particular de uma RPC. A marcação inicial M_0 é o conteúdo dos

lugares no estado inicial do sistema, e é calculada pela avaliação da expressão de inicialização de cada lugar.

A execução de uma RPC é definida pelo relacionamento entre as transições e os lugares, representado pelos arcos, ou mais precisamente, por suas anotações. No entanto, como dito anteriormente, uma anotação de arco é uma expressão que pode conter valores constantes, operadores, funções e variáveis. Na presença de variáveis, a anotação de arco somente representará um multi-conjunto de elementos concreto se valores forem atribuídos a essas variáveis. Assim, sendo v uma variável do conjunto de cores C, para que uma anotação de arco que envolve algum cálculo com v seja utilizada é necessário que um elemento de C seja ligado (do inglês, bound) a v. Uma ligação é formada por uma transição t e uma associação de cada variável do modelo a um valor do tipo apropriado. Além disso, a avaliação da expressão de guarda t com relação a essa associação de valores deve resultar no valor *booleano verdadeiro.*

2.5.3 Técnicas de análise e validação

As técnicas de análise e validação para RPCs são, em geral, adaptações de técnicas utilizadas para RPs. Desta forma, para análise e validação de uma RPC, pode-se empregar simulação, grafo de ocorrência e cálculo de invariantes (Jensen, 1997).

A simulação consiste em executar a rede, interativa e automaticamente, e avaliar o comportamento do sistema modelado, possibilitando visualizar o resultado de uma seqüência de eventos. A simulação automática também permite investigar questões sobre o desempenho do sistema, tais como a identificação de gargalos (Jensen, 1997).

Neste trabalho é abordada a técnica de simulação para a análise da RPC proposta.

2.6 Redes de Petri para robótica

As aplicações de redes de Petri na modelagem e em controle de sistemas da área de robótica são um dos focos da pesquisa acadêmica nas últimas décadas. Este fato é percebido facilmente na robótica industrial, como por exemplo, em Sistemas Flexíveis de Manufatura (FMS), e no desenvolvimento de robôs autônomos para ambientes não-estruturados. Carvalho (2005) e Milutinovic e Lima (2002) propõem um Modelo de Tarefas Robóticas de um robô, usando RPs através do estabelecimento de um sistema que faça avaliações qualitativa e quantitativa das tarefas desse robô. Na mesma linha, Lima (1998) usou RPs Interpretadas e Sava e Alla (2001) RPs híbridas para modelagem de Tarefas

Robóticas. Mellado e Canepa (2003) usaram uma RP com três níveis, dividiram o sistema do robô móvel em três tipos de rede: a rede do ambiente, a rede do agente e a rede do objeto. Assim, o nível do ambiente descreve o ambiente dos robôs, o nível do agente modela o comportamento geral dos robôs móveis e o nível do objeto representa características específicas de um dado robô (saber missões, tarefas e mapas rodoviários).

As redes de Petri têm em geral aplicação em todas as atividades da robótica, modelagem e simulação, incluindo a especificação, validação e geração do código para o controle dos robôs (Montano, 2000). O formalismo da rede de Petri pode fornecer as exigências de segurança de sistemas em tempo real. Isto torna o projeto desses sistemas bem mais simples, reduzindo o tempo e os custos de desenvolvimento.

Em ambientes não-estruturados é impossível modelar completamente o ambiente que os robôs móveis navegam e trabalham, eventos assíncronos requerem um modelo que incorpore a informação do sensor durante a execução de uma tarefa (Hale, 1999).

Os sistemas de controle dos robôs podem ser modelados através das RPs devido as suas facilidades e benefícios com técnicas de validação (Caloini e Magnani, 1998). No entanto, embora o uso de RPs na área de robótica seja extenso, após uma revisão exaustiva da literatura, como visto anteriormente, foram encontrados poucos trabalhos que tratam da aplicação das Redes de Petri ao problema de navegação de robôs e nada se achou para o mesmo problema utilizando as Redes de Petri Coloridas.

Assim, neste trabalho é proposta a modelagem de um sistema de navegação reativo para robôs móveis baseado em Redes de Petri Coloridas. Diversos aspectos serão testados e validados por um extenso conjunto de experimentos de navegação com um robô móvel real.

CAPÍTULO III

3. O robô móvel PISA G1 – aspectos de implementação

3.1 Características do robô móvel

A tarefa definida para o robô móvel é tão somente navegar pelo ambiente, dada uma rota para se atingir um alvo especifico. O robô é posicionado arbitrariamente, e não deve colidir contra obstáculos. O ambiente de navegação é bidimensional.

Assume-se um robô móvel tal como descrito no diagrama de blocos da Figura 3.1. Sensores de obstáculos, colisão e captura, provêem as informações utilizadas pelo sistema de navegação. Os sensores de obstáculos estão dispostos à frente do robô. O sensor de colisão ("*on-off*") detecta as colisões contra obstáculos, colocados como "pára-choques" na frente e na traseira do mesmo. Os sensores captam estímulos provenientes do ambiente e os transmitem para o sistema de navegação. A cada instante o sistema de navegação recebe os sinais captados pelos sensores e determina o ângulo de ajuste para correção da direção de movimento do robô móvel. Após o ajuste de direção de movimento, o robô avança um passo na rota pré-estabelecida.

Na Figura 3.2 é mostrada uma foto do robô móvel experimental PISA G1 desenvolvido como parte deste trabalho em parceria entre laboratórios de pesquisa do Departamento de Engenharia de Teleinformática - DETI da Universidade Federal do Ceará - UFC e do Instituto de Telemática - ITTI do Centro Federal de Educação Tecnológica do Ceará - CEFETCE. Este robô é composto por uma série de periféricos, dentre eles sonares baseados em ultra-som, sensores de infra-vermelho, ambos utilizados neste trabalho para detecção de obstáculos e medição de distâncias para alimentar o sistema de navegação do robô. O sensor de ultra-som está fixado a um servo motor, o que permite uma varredura de aproximadamente 180º e coleta de informações das reflexões das ondas sonoras nos objetos, formando assim uma "fotografia sônica" do ambiente. É incorporado ao robô um sistema de comunicação sem fio, conferindo assim uma maior flexibilidade no tratamento das informações obtidas pelos sensores, com um propósito de monitoramento. O circuito de controle é baseado em microcontroladores PIC18F452 de fabricação da Microchip. Quanto à sua geometria, ele possui forma cilíndrica com 0,80m de altura e 0,30m de raio.

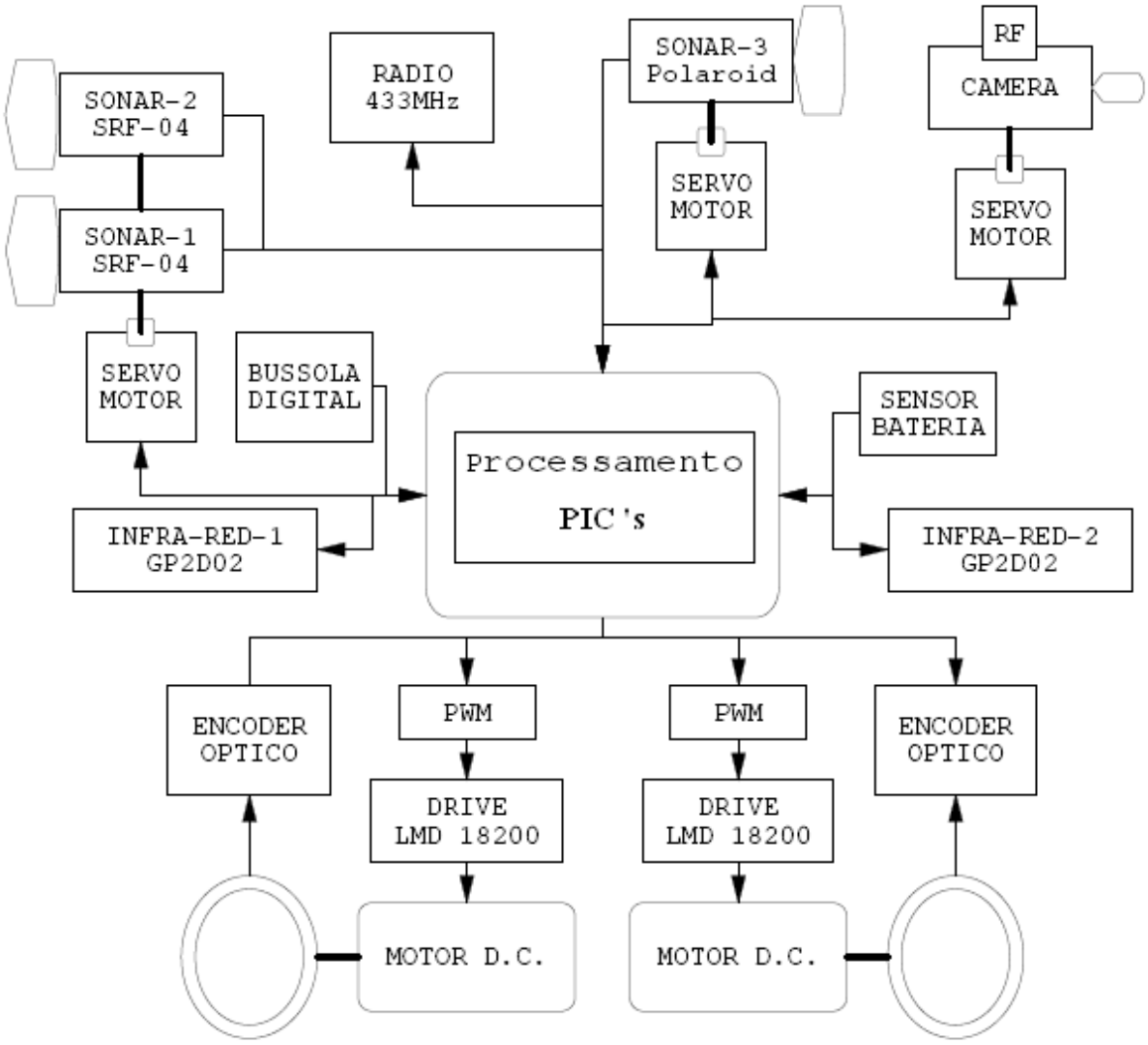

Figura 3.1: Diagrama de blocos do robô móvel.

A locomoção desta plataforma baseia-se na utilização de duas rodas motrizes (dois motores de corrente contínua acoplados diretamente às rodas de tração do robô), comandadas de forma independentes, e outras duas rodas livres apenas para apoio. Isto permite não só controlar a velocidade da plataforma, mas também a direção de movimentos através da aplicação de velocidades diferentes a cada roda. A obtenção de movimentos retilíneos ou circulares depende da aplicação de velocidades iguais ou diferentes às rodas e da geometria do próprio veiculo, que neste caso é circular (não holonômico). Esta configuração é bastante utilizada por robôs móveis em diversas aplicações na literatura especializada (Borenstein, 1996).

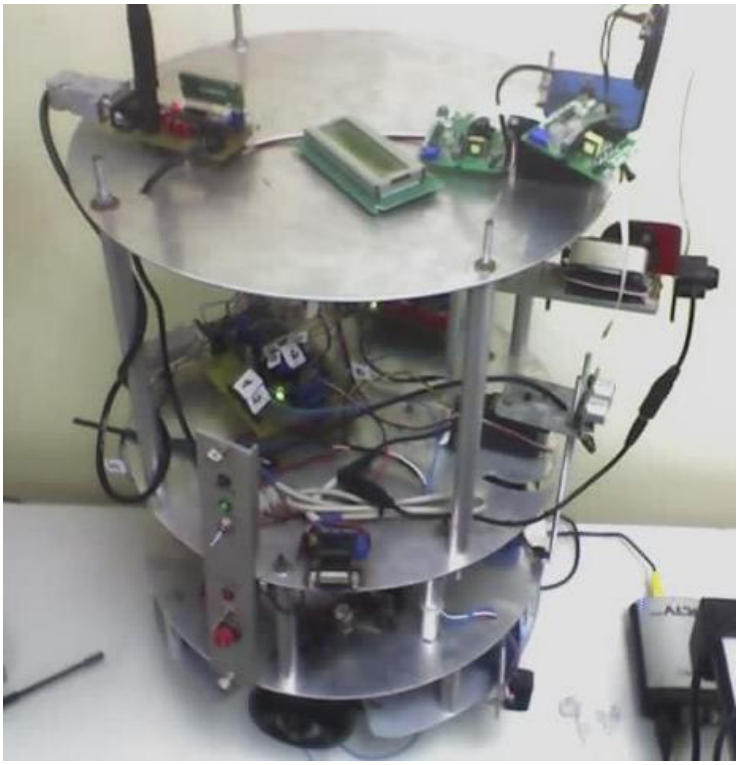

Figura 3.2: Robô móvel experimental PISA G1.

O controle de posição e velocidade da plataforma utiliza um sistema de estimação odométrico que por sua vez utiliza como entradas dois codificadores ópticos incrementais ligados a cada uma das rodas motrizes.

Embora o robô disponha de um sistema de aquisição de imagens, o mesmo não foi utilizado neste trabalho, pois os métodos de visão computacional necessários para tal são relativamente complexos, merecendo um espaço exclusivo para que se possa dar o tratamento adequado a este problema.

3.1.1 Arquitetura de Hardware e Software

A arquitetura é composta pelo arranjo, pela comunicação e pelo fluxo de dados entre as partes componentes. Uma arquitetura deve facilitar o desenvolvimento de sistemas robóticos provendo restrições benéficas no projeto e na implementação da aplicação desejada, servindo inclusive para a integração de módulos que sejam desenvolvidos independentemente (Coste e Simmons, 2000).

As arquiteturas podem ser classificadas (de acordo com a organização dos módulos interconectados e pela forma de processamento desses módulos) em centralizadas e distribuídas. No caso da arquitetura centralizada, um processador central único executa todo o

processamento, agregando, portanto, um melhor custo benefício que a arquitetura distribuída, por utilizar um único processador ao invés de vários. Por outro lado, a arquitetura centralizada oferece uma confiabilidade menor que a arquitetura distribuída, uma vez que uma falha no processador central compromete todo o sistema.

A arquitetura distribuída é baseada em sistemas sensoriais e controladores distribuídos simples, interligados em uma rede de sensores e atuadores, na qual nenhum sistema individual está em nível hierárquico superior a qualquer outro. Desta forma, o sistema se torna robusto, confiável, flexível, extensível e mais tolerante a falhas do que na abordagem centralizada. A falha de um subsistema não implica necessariamente na falha do sistema global (Coste, 2000).

Embora arquiteturas distribuídas tenham a vantagem da distribuição da carga computacional e não sofra das limitações impostas às arquiteturas centralizadas quanto ao processamento, há ainda limitações associadas, como: fluxo de comunicação maior e problemas de sincronização.

Algumas características da maioria das aplicações robóticas, tais como alta taxa de amostragem, grande número de sensores e atuadores de natureza diversa e necessidade de alta taxa de comunicação de dados, levam à necessidade de distribuir o processamento, justificando, portanto a utilização de uma arquitetura distribuída (Coste, 2000).

Nesse sentido, é proposta uma arquitetura de hardware e software distribuída que possua as seguintes características:

- Ser capaz de lidar com um grande número de sensores e atuadores diversos distribuídos;
- Ser implementada numa plataforma de baixo custo;
- Permitir modularização, padronização e reuso das aplicações;
- Possuir um alto nível de sincronização entre as trocas de mensagens entre os nodos da arquitetura distribuída.

3.1.1.1 Hardware

Devido ao elevado número de sensores e atuadores a serem integrados ao robô optou-se por especificar uma arquitetura de hardware distribuída.

De acordo com esta proposta, os sensores e atuadores são controlados de forma independente através de microcontroladores dedicados. Sendo assim, cada motor e seu sensor

de posição associado possuem uma placa microcontrolada dedicada para amostragem do sensor e acionamento do respectivo motor.

Foi especificada uma arquitetura mestre-escravo, onde uma das placas microcontroladas supervisiona todos os outros módulos controladores de sensores e atuadores, comunicando-se com os mesmos através de uma rede I2C (Philips, 2007).

O protocolo de comunicação I2C foi originalmente desenvolvido pela Philips em meados de 1996. Atualmente este protocolo está amplamente difundido e interconecta uma ampla gama de dispositivos eletrônicos. Dentre estes, encontramos vários dispositivos de controle inteligente, normalmente microcontroladores e microprocessadores assim como outros circuitos de uso geral, como *drivers* LCD, portas de I/O, memórias RAM e EEPROM ou conversores de dados. O I2C obtém destaque em sistemas de automação por ter uma proposta de alta velocidade, baixo custo e por possuir um excelente conjunto de definições em termos de protocolo (Philips, 2007).

Muitas vantagens podem ser atribuídas ao protocolo I2C. Destacam-se entre elas (Philips, 2007):

- Organização funcional em blocos, providenciando um simples diagrama esquemático final.
- Não há necessidade dos projetistas desenvolverem interfaces. Todos os dispositivos integram as interfaces "*on-chip*", o que aumenta a agilidade no desenvolvimento. Endereçamento e protocolo de transferência de dados totalmente definido via software. Possibilidade de inclusão ou exclusão de dispositivos no barramento sem afetá-lo ou a outros dispositivos conectados a este.
- Diagnóstico de falhas extremamente simples. O mau funcionamento é imediatamente detectado.
- Desenvolvimento simplificado do software através do uso de bibliotecas e módulos de software reutilizáveis.
- Facilidade no desenvolvimento de placas de circuito impresso, devido à quantidade de interconexões.

Características Gerais do Barramento I2C (Philips, 2007):

- Suporta qualquer tecnologia de produção.

- Duas vias de comunicação: serial data (SDA) e serial *clock* (SCL), ambas bidirecionais, conectadas ao positivo da fonte de alimentação através de um resistor de *pull-up*. Enquanto o barramento está livre, ambas as linhas ficam em nível lógico alto.
- A taxa de transferência máxima é de 100kbit/s no modo padrão (*standart*), ou 400kbit/s no modo rápido (*fastmode*).
- Informação de *carry* entre dispositivos conectados.
- Todo dispositivo possui um endereço único no barramento, independente de sua natureza.
- Qualquer dispositivo conectado pode operar como transmissor ou receptor. Claro que isso depende da natureza do dispositivo - um LCD não vai operar como transmissor, assim como um teclado não operará como receptor. Independente disto, qualquer dispositivo endereçado é chamado de escravo (*slave*).
- O número de interfaces conectadas é dependente da capacitância máxima do barramento, que é de 400pF.

Na Figura 3.3 é mostrado um esquema da arquitetura proposta.

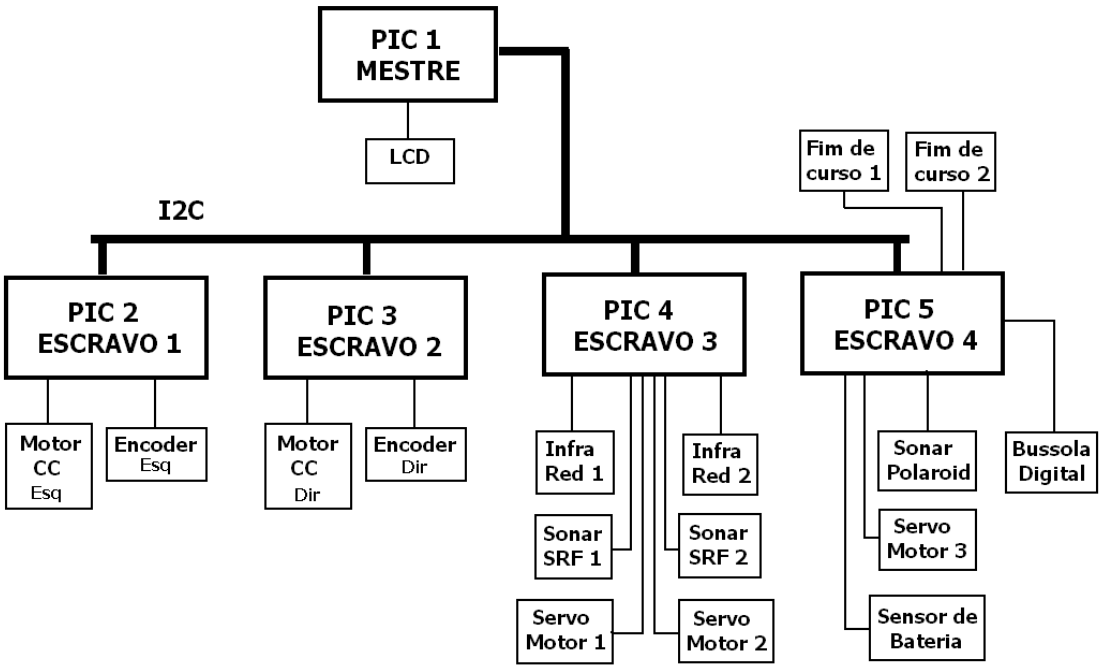

Figura 3.3: Arquitetura de Hardware do Robô.

De acordo com a arquitetura proposta foram projetados módulos microcontrolados embarcados para tratar os sonares e os atuadores, baseado em um microcontrolador de baixo custo.

3.1.1.2 Software

A arquitetura de software foi desenvolvida levando-se em consideração as definições, características e os protocolos de comunicação empregados na arquitetura de hardware.

Há duas formas de comunicação entre os módulos do sistema. A primeira trata da comunicação mestre-escravo entre os módulos microcontrolados através de uma rede I2C de transmissão de dados, a segunda foca na comunicação entre o PC (usuário) e o microcontrolador mestre.

Devido à arquitetura ser distribuída, os processos que agem paralelamente disputam os mesmos recursos de hardware e sistemas operacionais, ao mesmo tempo em que, pela própria definição de programa concorrente, cooperam para o êxito do sistema como um todo. Os processos compartilham recursos e se comunicam entre si, seja através do uso de variáveis (memórias) compartilhadas, seja através da troca de mensagens (Coste, 2000). Em ambos os casos, é necessária a utilização de um mecanismo de sincronização, que pode ser a primitiva de exclusão mútua ou semáforos, por exemplo.

A comunicação entre os módulos microcontrolados através da rede I2C é realizada através de uma arquitetura mestre-escravo, ou seja, os microcontroladores distribuídos, que desempenham a função de escravos, respondem às requisições feitas pelo microcontrolador mestre.

Localmente, os módulos microcontrolados, possuem embarcados softwares específicos que lêem os sensores de posição, controlam e acionam os motores CC (Corrente Continua), medem e acionam os sonares, recebem referências de velocidade e posição etc, que são disponibilizados em áreas (memórias) compartilhadas nos próprios controladores, de modo que quando solicitadas possam ser enviadas ao controle mestre ou possam ser utilizadas no processamento local do controlador.

Na Figura 3.4 é apresentado o formato dos quadros que trafegam na rede I2C entre os módulos microcontrolados.

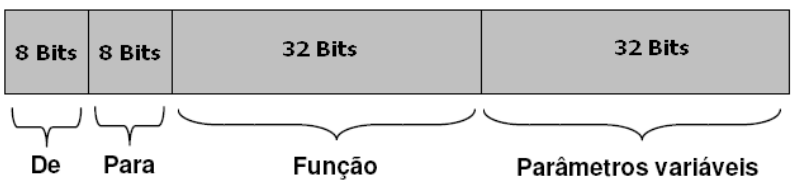

Figura 3.4: Quadros que trafegam na rede I2C.

Entre algumas funções disponíveis oriundas do controle mestre e que possui como destino os controladores escravos podem-se listar: requisição de referências (velocidade ou posição), acionamento do sonar, configuração de um controlador, requisição de medição, etc. Os controladores distribuídos sempre respondem à requisição solicitada num quadro com este mesmo formato.

3.1.2 Modos de Locomoção

A locomoção da plataforma assenta na utilização de duas rodas fixas que estando ligadas a dois motores podem ser controlados de forma independente, e por duas rodas do tipo castor. O controle de movimentos é então conseguido definindo as velocidades de cada uma das rodas fixas.

O software de controle do robô permite definir em cada instante a velocidade linear v_1 e a velocidade angular ω a serem aplicadas ao robô.

Como se pode ver na Figura 3.5 a velocidade linear v_1 correspondente à velocidade tangencial do ponto P situado a meio do eixo que une as duas rodas motrizes e a velocidade angular correspondente à velocidade angular de qualquer ponto do robô.

Utilizando o conceito de centro de rotação instantâneo, que permite descrever o movimento de um corpo rígido em cada instante como uma rotação pura em torno de um ponto no espaço (note que o centro de rotação instantâneo vai variando ao longo da trajetória do corpo rígido, sendo fixo apenas nos casos em que estamos em presença de uma rotação pura). Obtém-se da relação entre velocidade angular e velocidade linear a distancia r do ponto P ao centro de rotação instantâneo (C_{RI}). Uma vez que todos os pontos sobre o robô têm a mesma velocidade angular, tem-se a seguinte relação para as velocidades da roda esquerda (v_e) e da roda direita (v_d):

$$\frac{v_e}{r_e} = \frac{v_d}{r_d}$$

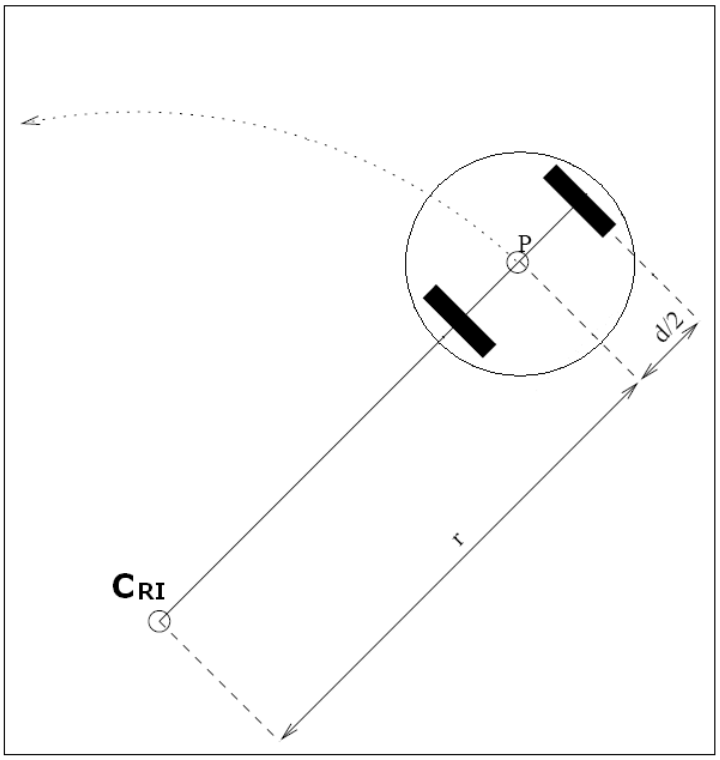

Figura 3.5: Rotação da plataforma em torno do ponto ICR.

em que r_e e r_d correspondem respectivamente as distâncias da roda esquerda e da roda direita ao ponto C_{RI}.

Por outro lado se o ponto P está situado exatamente ao meio das duas rodas motrizes então a distância do ponto P a cada uma das rodas será $d/2$ e obtém-se facilmente a velocidade linear de cada uma das rodas,

$$v_e = \omega \times (r - d/2)$$
$$v_d = \omega \times (r + d/2)$$

em que d é um vetor coincidente com o eixo de rotação das rodas e que parte da roda esquerda para a roda direita e r é o vetor que une o ponto C_{RI} ao ponto P. Obtidos estes valores é possível calcular a velocidade angular de cada uma das rodas em torno do seu eixo de rotação

$$w_e = V_e / R$$
$$w_d = d_d / R,$$

em que R é o raio das rodas motrizes.

3.1.3 Estimação de Posição

A forma mais utilizada para se obter a posição de um robô móvel é através da estimação da mesma com base no conhecimento da posição inicial, no caminho percorrido e no perfil de velocidade utilizado ao longo desse mesmo caminho. Este tipo de localização é muitas vezes referido na literatura pela expressão inglesa "*dead reckoning*" que derivou de "*deduced reckoning*" (Borenstein, 1996) ou simplesmente por odometria. Este é um dos métodos mais simples e mais utilizados, sendo na maior parte das vezes implementado acoplando codificadores ópticos incrementais ao eixo das rodas motrizes. Estes sensores permitem detectar qualquer deslocamento de cada uma destas rodas. A seguir é mostrado como é feita a estimação de posição com base nos incrementos obtidos dos sensores.

Em primeiro lugar é necessário dispor, para cada uma das rodas, de um fator de conversão entre dois impulsos obtidos do codificador e o deslocamento linear associado. Sendo C_m esse fator de conversão, D_m o diâmetro da roda (mm), C_e o número de impulsos emitidos pelo codificador durante uma revolução completa e n o fator de redução entre o eixo da roda e o eixo a que está ligado o codificador, temos

$$C_m = \frac{\pi D_m}{n C_e}$$

Sendo N_e e N_d o número de impulsos contados desde a última amostragem para cada uma das rodas, pode-se calcular a variação incremental na posição destas.

$$\Delta U_{e,i} = C_m N_{e,i}$$
$$\Delta U_{d,i} = C_m N_{d,i}$$

em que $\Delta U_{e,i}$ e $\Delta U_{d,i}$ correspondem respectivamente as variações incrementais na posição da roda esquerda e da roda direita.

Nota-se que $N_{e,i}$ e $N_{d,i}$ podem tomar valores positivos ou negativos se as rodas rodarem no sentido direto ou inverso, respectivamente e de acordo com o que for convencionado. Assim, os valores $\Delta U_{e,i}$ e $\Delta U_{d,i}$ corresponderão a deslocamentos lineares "para frente" ou "para trás".

Utilizando estes valores, o deslocamento do centro do robô (Ponto P) é dado por

$$\Delta U_i = \frac{\Delta U_d + \Delta U_e}{2}$$

A variação incremental da orientação do robô será

$$\Delta\theta_i = \frac{\Delta U_d - \Delta U_e}{b}$$

em que $b = \|d\|$ é a distância entre os pontos de contato de cada uma das rodas com o solo.

A orientação (relativa) do robô é calculada por

$$\theta_i = \theta_{i-1} + \Delta\theta_i$$

sendo a posição (relativa) do ponto central dada por

$$x_i = x_{i-1} + \Delta U_i \cos\theta_i$$
$$y_i = y_{i-1} + \Delta U_i \operatorname{sen}\theta_i$$

Este método tem como vantagem a simplicidade, sendo por isso bastante utilizado em robôs móveis terrestres. No entanto, dado o seu funcionamento em malha aberta, os valores de posição fornecidos estão sujeitos a erros cumulativos que advêm de fatores tais como o escorregamento das rodas, a variação do centro de curvatura ao longo da superfície de contato das rodas com o chão (deslocamento do ponto de contato) e mesmo da variação do diâmetro das rodas com a carga do veículo.

3.2 Arquitetura de controle

A organização da arquitetura hierárquica de controle desenvolvida pode ser observada na Figura 3.6. São três níveis de controle: o nível sistema de navegação (Nível 2), o nível controle de trajetória (Nível 1), e o nível controle de velocidade (Nível 0).

- O nível 0 gera informações sobre o deslocamento angular de cada roda através dos pulsos gerados pelo codificador óptico correspondente, de forma a obter a localização do robô no sistema de coordenadas X-Y e sua respectiva orientação. Após fazer o processamento desses dados, o nível 2 fornece ao nível 1 informações sobre o deslocamento linear ortogonal e o desvio angular.
- O nível 1, com base nas variáveis de erro de trajetória, realimenta o controlador de velocidade com novas velocidades de referência, de forma a fazer com que o robô convirja para a trajetória e mantenha a velocidade no ponto P constante.
- O nível 2, com base nos dados da odometria, determina trajetórias através dos parâmetros passados para o nível 1, por meio da velocidade do ponto P e das

coordenadas polares dos pontos x_d e y_d de destino desejados, que são o comprimento do segmento de reta a ser percorrido e o ângulo de orientação.

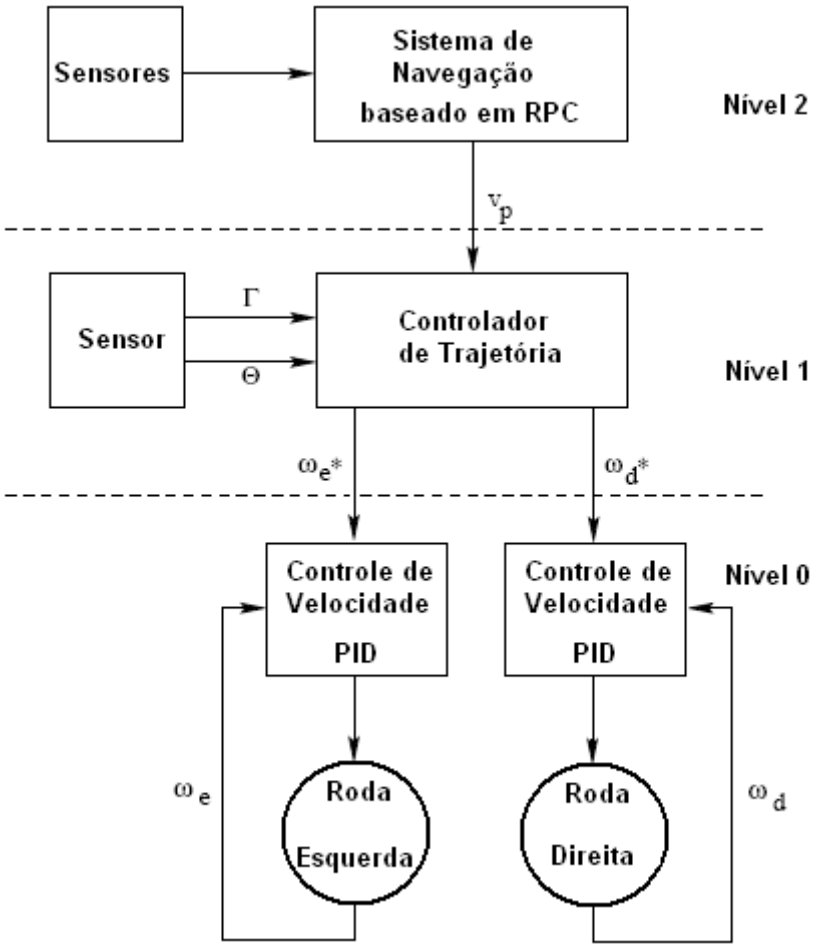

Figura 3.6: Hierarquia de controle.

3.3 Controle de trajetória

Em robôs móveis com rodas, controladores de trajetória atuam sobre o sistema de propulsão para fazer o veiculo seguir uma trajetória de referência. A trajetória pode ser representada por curvas livres (i.e., Curvas de Bézier), ou sob forma de estruturas geométricas (i.e., segmentos de arcos, retas) (Sordalem, 1983) que são mais facilmente tratáveis do ponto de vista computacional.

Para o rastreamento de trajetórias, as abordagens existentes empregam ou o paradigma controle seqüencial de postura ou o paradigma de rastreamento de caminho. No controle seqüencial de postura, a cada passo de amostragem, a postura desejada para o robô é

definida como um ponto sobre a trajetória de referência, que é parametrizada pelo tempo. Nesse caso, o robô estará seguindo a trajetória pré-determinada, amostrada sob forma seqüencial de posturas. No rastreamento de trajetória, não existe controle sobre a posição do robô, mas sobre variáveis que definem o erro de trajetória, que são função da posição atual e da estrutura geométrica sendo rastreada (Borges et. al., 2003).

No que diz respeito aos controladores de trajetória, com controladores dinâmicos, é necessário obter experimentalmente o modelo dinâmico do robô, cujos parâmetros variam a cada vez que ocorrem mudanças estruturais no veículo ou variação no seu peso, o que é comum com veículos de transporte de materiais. Para esses casos, Colbaugh (2000) apresenta um controlador robusto a incertezas no modelo dinâmico. Por outro lado, com controle cinemático, velocidades de referência são determinadas para cada eixo de tração e disponibilizadas para serem aplicadas por controladores de velocidade. Com relação a esta modalidade de controle, cita-se, por exemplo, realimentação não-linear de estados, controle adaptativo, lógica nebulosa e estrutura variável.

Nas últimas décadas foram propostas várias técnicas de controle de trajetória para veículos autônomos e robôs. Sabe-se que em geral o sistema de controle de velocidade é decisivo para o bom desempenho e para a estabilidade do sistema de controle de trajetória. Já está bem consolidado na literatura, que controladores PID não garantem a estabilidade para veículos não-holonômicos (Borenstein, 1996), dentre os quais estão os robôs de tração diferencial.

Em Borges et. al. (2003), os autores apresentaram um controlador proporcional de trajetórias baseado na cinemática de robôs a tração diferencial. Os parâmetros do controlador são relacionados diretamente com algumas medidas de performance desejadas, de forma que pouco ajuste era necessário. Apesar da simplicidade da estrutura, seu desempenho foi considerado satisfatório em experimentos com uma plataforma real. Seguindo a mesma estrutura de projeto, neste trabalho utiliza-se de um controlador proporcional, também baseado apenas na cinemática de robôs a tração diferencial, haja vista que o objetivo deste trabalho não é projetar controladores para robôs móveis, e sim apresentar um controle de velocidade e um controle de trajetória que possa funcionar de maneira satisfatória, e assim validar um modelo de navegação baseado em Redes de Petri Coloridas.

3.3.1 Erro de trajetória

Considera-se a classe de robôs com sistema de locomoção diferencial, onde em Campion (Campiom, 1995), é apresentado o modelo cinemático e dinâmico que descreve o comportamento do mesmo. Na Figura 3.7 observa-se o robô móvel com sistema de tração diferencial e variáveis de erro de trajetória. Define-se $C : X \times Y$ o sistema de coordenadas no qual a tarefa de navegação é definida, e $C_R : X_R \times Y_R$ o sistema de coordenadas do robô. Por conveniência, a origem de C_R está sobre o ponto de referência P, localizado no centro das rodas de tração, com o eixo Y_R coincidindo com os eixos das rodas. Define-se ainda r como sendo o raio das rodas de tração, consideradas idênticas, b sendo a distância entre as rodas de tração, e υ_d e υ_e sendo as velocidades das rodas direita e esquerda, respectivamente, que conjuntamente formam $v(t) = [\upsilon_d(t), \upsilon_e(t)]^T$. O vetor $\xi = [x_p, y_p, \phi]^T$ descreve a configuração cartesiana do robô, i.e., sua posição (x_p, y_p) e sua orientação ϕ (i.e., o ângulo entre os eixos X_R, X).

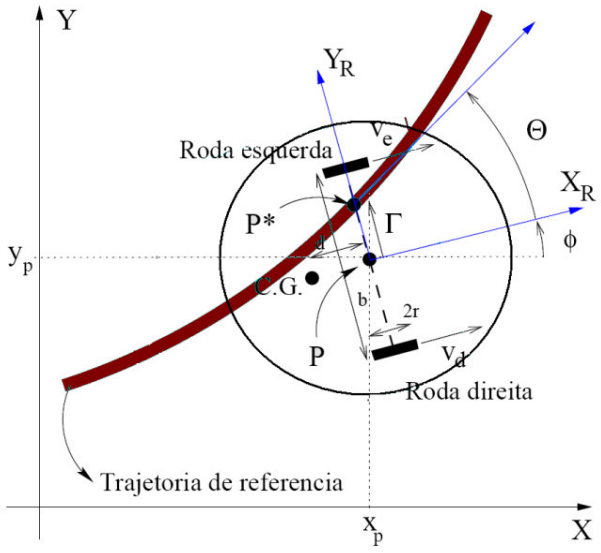

Figura 3.7: Robô móvel com tração diferencial e variáveis de erro de trajetória.

No contexto desse trabalho, considera-se o paradigma de rastreamento de caminho onde o erro de rastreamento é a variável a ser minimizada pelo controlador. Considera-se ainda que a trajetória desejada possa ser definida (ou aproximada) por um segmento de reta ou por um arco. O erro de trajetória é dado em termos de duas variáveis: o deslocamento

ortogonal Γ e o desvio angular Θ, como mostrado na Figura 3.7. Define-se $P^* = (x^*, y^*)$ como o ponto de intersecção entre a trajetória e uma reta imaginária coincidente com o eixo das rodas. Γ representa a distância $P - P^*$, sendo negativo se P^* está sobre o lado direito de P (i.e., na direção da roda direita).

O desvio angular Θ é o ângulo entre o vetor tangente à trajetória sobre P^* e o vetor de direção do veículo. Considerando-se a trajetória de referência como sendo dada por um segmento de reta, o erro de rastreamento é dado por (Borges et. al., 2000):

$$\Theta = -\phi(t), \tag{3.1}$$

$$\Gamma(t) = \frac{y_p(t)}{\cos\phi(t)}. \tag{3.2}$$

No caso de arcos de raio de curvatura R_C, considera-se que o robô tenha alcançado regime permanente, apresentando apenas erro residual em Γ, que pode ser dado por

$$\Gamma(t) = \frac{\upsilon_p(t)}{\Delta\omega}\frac{b}{2r} - R, \tag{3.3}$$

com $\Delta\omega$ sendo o diferencial de velocidade angular das rodas motrizes. Entende-se assim que o robô esteja realizando uma curva de raio $R_c = -(\Gamma_c + R)$.

As velocidades angulares da roda esquerda e da direita, ω_e e ω_d são dadas por

$$\omega_e = \frac{\upsilon_p}{r} - R\Delta\omega(\Gamma(t), \theta(t)), \tag{3.4}$$

$$\omega_d = \frac{\upsilon_p}{r} - R\Delta\omega(\Gamma(t), \theta(t)) \tag{3.5}$$

3.3.2 Estratégia de controle

A estratégia de controle empregada nesse trabalho, que inicialmente foi proposta em Borges et. al. (2000), permite manter a velocidade do veículo constante, i.e., $\upsilon(t) = \upsilon_p$. Em aplicações de transporte de materiais, υ_p pode ser especificado pelo navegador e, de modo geral, pode variar de zero a um valor máximo, em função do material transportado.

Assim, uma das atribuições do navegador é a determinação de υ_p, conforme a tarefa em execução. Para se garantir que a velocidade de navegação seja $\upsilon(t) = \upsilon_p$, considera-se

$$\upsilon_d(t) = \upsilon_p + r \cdot \Delta\omega(t), \tag{3.6}$$

$$\upsilon_e(t) = \upsilon_p + r \cdot \Delta\omega(t), \tag{3.7}$$

com $\Delta\omega(t)$ sendo a variável de controle, que corresponde ao diferencial de velocidade angular do sistema de tração. Observa-se que, tratando-se de controle cinemático, os reguladores propostos devem determinar velocidades desejadas para as rodas motrizes, sob a forma (3.6)-(3.7). No entanto, são necessários controladores de velocidade eficientes para as rodas motrizes, pois de seu bom desempenho depende a redução de efeitos ligados à dinâmica do robô. Neste trabalho, os controladores de velocidade das rodas são feitos por controladores convencionais do tipo PID, que serão tratados adiante.

3.3.3 Controlador proporcional (CP)

Em Borges et. al. (2003) foi proposto um controlador proporcional segundo a seguinte lei de controle:

$$\Delta\omega(t) = K_\Gamma \cdot \Gamma(t) + K_\Theta \cdot \Theta(t), \tag{3.8}$$

fazendo com que, considerando-se as equações. (3.1) e (3.2) e $\varepsilon(t) = \Gamma(t)\cos\Theta(t)$, o comportamento do erro de trajetória seja regido por

$$\dot{\varepsilon}(t) = \upsilon_p \sin\Theta(t), \tag{3.9}$$

$$\dot{\Theta}(t) = -\frac{2r}{b}\Delta\omega(t). \tag{3.10}$$

Para o sistema descrito por (3.9)-(3.10), pode-se demonstrar que o atrator (0; 0) é estável no sentido de *Lyapunov* com K_Θ e K_Γ sendo constantes positivas. Os ganhos K_Θ e K_Γ do controlador são determinados analiticamente de forma a se obter um controlador *deadbeat* quando do rastreamento de segmentos de retas, e apresentar um erro Γ_c constante no segmento de arcos:

$$K_\Theta = \sqrt{\frac{2\upsilon_p K_\Gamma b}{r}}, \tag{3.11}$$

$$K_{\Gamma} = \frac{2\upsilon_p b}{r((R + 2\Gamma_c)^2 - R^2} \tag{3.12}$$

Este controlador proporcional de trajetória tem a vantagem de ser relativamente simples, ter um bom desempenho e ainda ter um custo computacional pequeno, podendo ser implementado em microcontroladores de baixo custo (Borges et. al. 2000).

Para trajetórias virtuais, pode-se aproximar qualquer curva suave por uma soma de segmentos de retas, de forma que pode ser utilizada uma estratégia de controle de trajetória usando somente segmentos de retas. Neste ponto, pode-se notar que o controlador garante que o robô sempre convergirá para a trajetória e terá um erro nulo em regime permanente para o caso de retas.

Entretanto, caso ocorra um problema na medição das variáveis Γ e θ, o controlador não conseguirá manter o robô na trajetória desejada, mostrando a importância de se ter um subsistema de controle de velocidade e odometria bem ajustados.

3.4 Controle de velocidade

Todos os movimentos realizados pelo robô são obtidos através do controle do acionamento dos motores de tração. Servo motores equipados com encoder incremental de alta resolução tem tornado o sistema de controle de movimento a melhor abordagem tecnológica para obter precisão e controle do sistema de acionamento. Existem várias abordagens para o controle de velocidade, que vão desde a abordagem do controlador proporcional até abordagens mais complexas, tais como o controle adaptativo. Contudo, o objetivo aqui não é fazer um estudo sobre qual estratégia é a melhor, e sim apresentar uma estratégia que garanta um bom tempo de resposta e erro nulo em regime permanente.

Em geral, para robôs pequenos, como é aqui o caso, é possível utilizar controladores PID convencionais (Lages, 1998) para se construir servos de velocidade ou posição para cada roda, pois estes laços de controle internos são rápidos o suficiente para o sistema apresentar o comportamento descrito pelo modelo cinemático e dinâmico do robô móvel.

Neste sentido, as características do controlador de velocidade precisam ser eficientes frente a perturbações (quando o robô esta realizando uma tarefa, como carregar um

objeto) e variações dos parâmetros (a massa e o centro de gravidade do robô podem se modificar).

Neste contexto, um controlador PID bem sintonizado pode fornecer uma escala bastante ampla de controle de velocidade com baixa ondulação no torque aplicado para girar ou direcionar as rodas de um robô móvel.

3.4.1 Controlador PID

Certas estruturas de controle definem um controlador como um dispositivo que modela a ação de controle, operando sobre o sinal de erro e(t), conforme ilustrado na Figura 3.8.

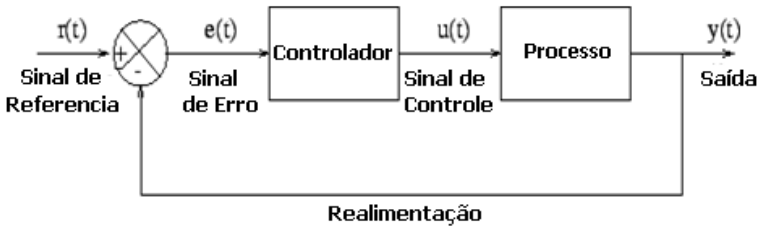

Figura 3.8: Controlador.

A implementação de sistemas com a finalidade de controle requer, geralmente, a introdução de blocos que alteram as funções originais existentes no sistema. Esses requisitos podem advir de critérios de estabilização, de modificação de transitórios, de critérios de otimização, etc. Uma forma de conseguir estas alterações de desempenho do sistema é através das realimentações (Ogata, 1993).

A combinação das ações proporcional, integral e derivativa para gerar um só sinal de controle, dá origem ao que chamamos de controlador proporcional-integral-derivativo ou simplesmente PID, O objetivo é aproveitar as características particulares de cada uma destas ações a fim de se obter uma melhora significativa do comportamento transitório e em regime permanente do sistema controlado. A característica básica destes controladores, que os torna muito populares, é o fato de que, quando adequadamente ajustados, eles geralmente conduzem o sistema a um desempenho satisfatório (Hermely, 1996). O sinal de controle gerado pelo controlador PID é assim genericamente dado como:

$$u(t) = K(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt}) \tag{3.13}$$

Desta forma têm-se três parâmetros de sintonia no controlador: o ganho proporcional K (ação proporcional), o tempo integral T_i (ação integral) e o tempo derivativo T_d (ação derivativa).

Apesar de termos a disponibilidade das três ações básicas, dependendo da aplicação não será necessária a utilização de uma ou mais destas ações. Por exemplo, em uma planta do tipo 1 (i.e. apresentando um pólo na origem) a utilização da ação integral não se fará necessária se o objetivo de controle for o de seguir, com erro nulo, um sinal de referência constante. Basicamente têm-se 4 configurações possíveis de controladores a partir de uma estrutura PID: proporcional (P), proporcional-integral (PI), proporcional-derivativo (PD) e proporcional-integral-derivativo (PID)

O controlador PID combina as vantagens dos controladores PI e PD. A ação integral está diretamente ligada à precisão do sistema sendo responsável pelo erro nulo em regime permanente. O efeito desestabilizador do controlador PI é contrabalançado pela ação derivativa que tende a aumentar a estabilidade relativa do sistema, ao mesmo tempo que torna a resposta do sistema mais rápida devido ao seu efeito antecipatório.

A função de transferência do controlador PID é dada por:

$$G_{pid} = \frac{u(s)}{r(s)} = K(1 + \frac{1}{T_{is}} + \frac{spTd}{s+p}) = \frac{K(s^2 + \frac{1+T_dT_i}{T_i}s + \frac{p+T_ip}{T_i}}{s(s+p)} \tag{3.14}$$

É importante ressaltar que a equação (3.13) e a função de transferência (3.14) constituem-se na versão clássica do controlador PID. Outras versões e variações existem, mas a filosofia de funcionamento, a partir da combinação dos efeitos das três ações básicas, é a mesma.

3.4.2 Sintonia do controlador PID

Vários métodos de sintonia de controladores PID são conhecidos e utilizados na prática de sistemas de controle. Cada um destes métodos requer algum tipo de informação sobre a dinâmica do processo a ser controlado e a natureza desta informação é que caracteriza cada um destes métodos. A fim de obter um método prático de ajuste, deve ser possível obter estas informações a partir de ensaios simples sobre o processo, ao mesmo tempo em que estas informações devem ser suficientes para possibilitar um ajuste adequado do controlador. Logo,

a quantidade adequada de informação a ser obtida do processo deve ser selecionada de forma a obter um compromisso entre simplicidade e desempenho do controlador.

O método mais bem sucedido na prática industrial de ajuste de controladores PID é o método de *Ziegler-Nichols*. Eles propuseram métodos para a determinação dos valores do ganho proporcional K_p, do tempo integral T_i e do tempo derivativo T_d, com base nas características da resposta transitória de um dado processo. Tal determinação dos parâmetros do controlador PID pode ser feita a partir da resposta no domínio do tempo do processo a ser controlado, para uma dada entrada, caso não se tenha um modelo matemático (Ogata, 1993).

Os métodos de *Ziegler-Nichols* foram introduzidos já em 1942 e hoje são considerados clássicos. Estes métodos continuam a ser largamente aplicados até hoje, mesmo em sua forma original, mas mais costumeiramente em alguma forma modificada. Os dois métodos básicos de ajuste de *Ziegler-Nichols* visam obter uma mesma resposta pré-especificada para o sistema em malha fechada, e diferem no que diz respeito à natureza da informação sobre a dinâmica do processo que é exigida por cada um deles.

O método da resposta ao salto, ou método do domínio do tempo, requer o conhecimento de duas grandezas que caracterizam a resposta ao salto de um processo. Já o método da realimentação por relé, ou método do período crítico, exige o conhecimento de duas grandezas características da resposta em freqüência do processo. Uma vez obtidas estas informações, basta recorrer a fórmulas extremamente simples para calcular os ganhos do controlador. Estas fórmulas foram determinadas de maneira empírica por meio de ensaios de processos industriais típicos. As fórmulas originalmente propostas por Ziegler e Nichols fornecem uma resposta que foi posteriormente considerada insatisfatória. Diferentes fórmulas foram então propostas com base nos mesmos ensaios, obtendo-se melhor desempenho (Ogata, 1993).

3.4.3 Discretização dos controladores PID contínuos

A implementação do controlador PID pode ser feita fazendo-se aproximações numéricas das derivadas e da integral que aparecem na lei de controle. Desta forma, é possível descrever cada uma das ações por uma equação de recorrência. As equações de recorrência descrevem as operação matemáticas a serem programadas no microcontrolador ou no microcomputador onde será implementado o PID digital, esse processo é descrito em seguida.

Um controlador ideal PID no domínio do tempo é dado pela seguinte equação (Souza e Costa Filho, 2001):

$$u(t) = K_p\left[e(t) + \frac{1}{T_i}\int_0^t e(t)dt + T_0\frac{de(t)}{dt} \right] \tag{3.15}$$

Sendo,

K_p = ganho proporcional;

T_i = tempo integral;

T_d = tempo derivativo.

Para um tempo de amostragem pequeno, a equação (3.15) pode ser discretizada para a obtenção da equação à diferença correspondente. A derivada é substituída por uma diferença de primeira ordem (primeira diferença) e a integral por uma soma (Souza e Costa Filho, 2001). Para a integração retangular obtém-se:

$$u(k) = K_p\left\{ e(k) + \frac{T_0}{T_i}\sum_{i=1}^{k} e(i-1)\frac{T_d}{T_0}\left[e(k) - e(k-1)\right] \right\} \tag{3.16}$$

A equação (3.16) determina um algoritmo de controle digital tipo PID não recursivo, pois para determinar u(k) todos os valores passados de e(k) têm que ser armazenados. Para a programação em processadores digitais, a forma recursiva é a mais adequada. Isto significa que o cálculo do controle num instante, u(k) é baseado no valor anterior u(k-1) mais termos corretores. Para obter a forma recursiva faz-se

$$u(k-1) = K_p\left\{ e(k-1) + \frac{T_0}{T_i}\sum_{i=1}^{k-1} e(i-1)\frac{T_d}{T_0}\left[e(k-1) - e(k-2)\right] \right\} \tag{3.17}$$

subtraindo (3.16) de (3.17) obtém-se:

$$u(k) - u(k-1) = K_p e(k) - K_p e(k-1) + K_p\frac{T_0}{T_i}e(k-1) + K_p\left[e(k) + e(k-2) - 2e(k-1)\right] \tag{3.18}$$

ou, equivalentemente

$$u(k) - u(k-1) = [K_p + K_p\frac{T_d}{T_0}]e(k) + [K_p\frac{T_0}{T_i} - 2k_p\frac{T_d}{T_0} - K_p]e(k-1) + K_p\frac{T_d}{T_0}e(k-2) \tag{3.19}$$

ou seja

$$u(k) - u(k-1) = q_0 e(k) + q_1 e(k-1) + q_2 e(k-2) \tag{3.20}$$

em que os parâmetros qo ,q1 e q2 são dados por

$$q_0 = k_p[1 + \frac{T_d}{T_0}]$$

$$q_1 = -k_p[1 + 2\frac{T_d}{T_0} - \frac{T_0}{T_i}] \qquad (3.21)$$

$$q_2 = k_p \frac{T_d}{T_0}$$

CAPÍTULO IV

4. Modelo de navegação, simulação e resultados experimentais

4.1 Modelo de Navegação

Para facilitar a definição e manipulação da RPC proposta, a mesma está organizada em páginas RPC. Uma página pode ser considerada como uma RPC simples, cuja semântica depende de outras páginas e/ou superpáginas. A associação entre uma página e suas subpáginas é feita por meio de transições hierárquicas e lugares portas. Uma transição hierárquica é associada a outra página. Cada lugar ligado a uma transição hierárquica é um lugar-porta. A cada lugar-porta, associado a uma transição hierárquica, deve-se associar um lugar compatível (associado ao mesmo tipo de dado) na subpágina (denominado, lugar-soquete). Um diagrama da arquitetura da RPC proposta pode ser visto na Figura 3.9. Nesta figura pode-se perceber que algumas páginas estão no mesmo nível (i.e. “PRINCIPAL”, “MAPEAMENTO”, “ODOMETRIA”, …) e “COLETA SENSOR” e “NAVEGA” são subpáginas de “PRINCIPAL”.

A semântica da RPC hierárquica é a mesma de uma rede não hierárquica, fazendo-se com que cada par de lugares-porta/soquete se comportem como se fossem o mesmo lugar. Em outras palavras, o conteúdo de ambos os lugares será sempre o mesmo. Isso significa que, sempre que uma ficha é adicionada a qualquer um dos lugares, uma ficha igual é adicionada ao respectivo par. Analogamente, sempre que uma ficha é removida de qualquer dos lugares, uma ficha igual é removida do respectivo par.

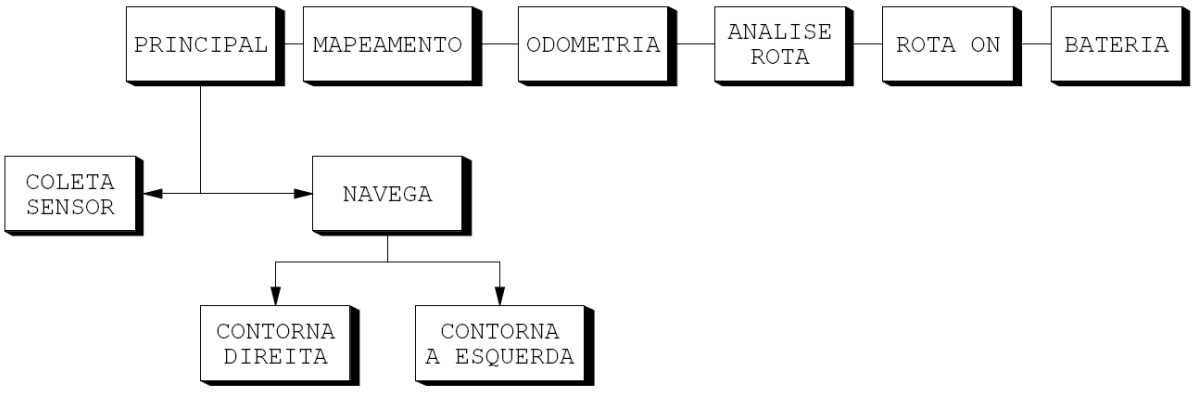

Figura 4.1: Hierarquia da RPC proposta.

O modelo RPC de navegação proposto é formado pelos seguintes módulos:

- **Principal:** responsável pelo gerenciamento da rede, coordenando os demais módulos, coleta informações através dos sensores, constrói uma representação do mundo em volta do robô através da fusão dos dados coletados pelos sensores, e então planeja as ações.
- **Coleta Sensor:** faz a leitura do ambiente utilizando os sensores embarcados no robô.
- **Navega:** modela a interação do robô com o ambiente fazendo uma representação do ambiente, possibilitando ao robô coordenar suas ações no sentido de desempenhar sua tarefa.
- **Contorna a Direita e Contorna a Esquerda:** para uma dada rota, esse módulo é solicitado quando um planejamento de trajetória tende à direita ou à esquerda, respectivamente.
- **Mapeamento:** faz a geração da trajetória a ser rastreada pelo robô.
- **Odometria:** determina a localização do robô por meio da observação e integração consecutiva do movimento das rodas
- **Analise Rota:** observa se a rota inicialmente carregada está sendo executada corretamente durante todos os movimentos do robô. Permite a representação do ambiente, construída a partir das leituras obtidas pelos sensores e é utilizada para o planejamento de trajetória.
- **Rota On:** analisa se a tarefa ou determinada trajetória foi concluída com sucesso.
- **Bateria:** permite o uso otimizado das baterias do robô, observando o deslocamento do robô e indicando quando e o mesmo deve voltar a uma base de apoio para recarga.

4.2 Simulação da RPC

A página principal do modelo é apresentada na Figura 4.2. Para a modelagem, analise e simulação da RPC proposta utilizou-se a ferramenta *CPN Tools*, que se trata de um programa computacional, executado nos ambientes Windows e Linux, para modelagem, análise e validação de RPCs. A página de declarações referente ao modelo de navegação proposto é apresentada no Apêndice A.

A rede foi executada de forma exaustiva, a fim de avaliar o comportamento dinâmico da navegação de um robô móvel. Inicialmente o robô encontra-se em espera e uma rota simples é aplicada à RPC, formada por seis sub-rotas. Este estado é modelado pela

presença das seis fichas no lugar “Repouso” da RPC: 1(1,65,"SubRota-1 Frente"), 1(2,73,"SubRota-2 Direita"), 1(3,88,"SubRota-3 Esquerda"), 1(4,56,"SubRota-4 Direita"), 1(5,74,"SubRota-5 Frente"), 1(6,62,"SubRota-6 Esquerda"), como mostrado na Figura 4.2. A transição circundada de verde indica que a mesma está habilitada e pode ser disparada.

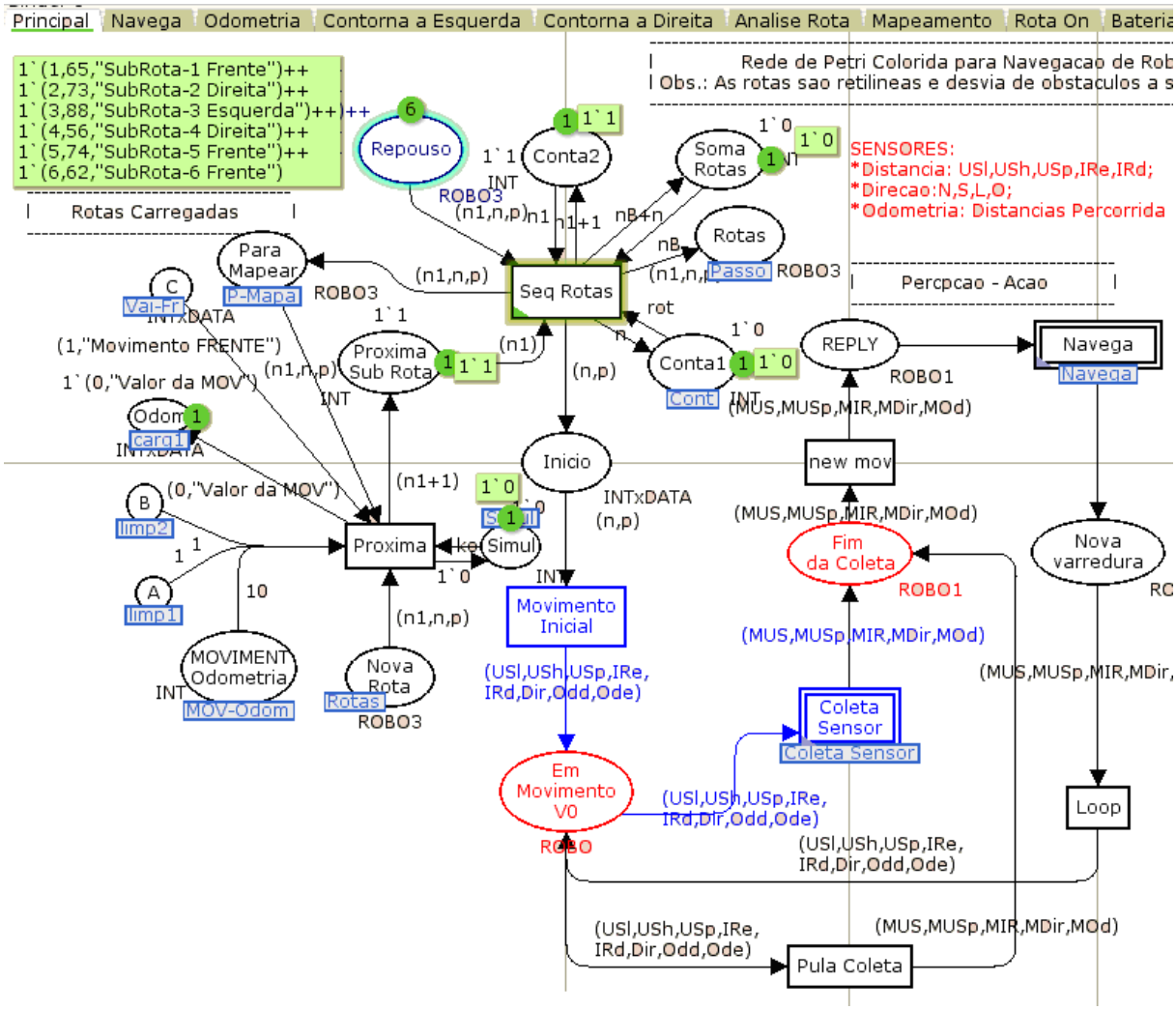

Figura 4.2: módulo “Principal” da RPC proposta.

O disparo da transição “Seg-Rotas” modela o início de uma rota. A sub-rota 1 inicia com o robô seguindo em linha reta, isto é feito através do disparo da transição “Movimento Inicial”, a partir de então, inicia-se o ciclo de navegação do robô. A primeira seção deste ciclo é mostrada na Figura 4.3 e corresponde ao “Sensoriamento”. Isto ocorre, pois para iniciar o processo de tomada de decisão o robô deve primeiramente situar-se no espaço em que se encontra. Nesta seção, cada sensor de obstáculo registra a distância do robô com relação ao obstáculo em sua direção, estas distâncias são valores aleatórios que foram

modeladas através de uma função randômica da própria ferramenta de modelagem, por exemplo, o sensor de Ultra-Som usa a função: UltraSomm = int with 1..70, esta função gera números aleatórios inteiros entre 1 e 70. A partir da agregação das informações de todos os sensores, o robô pode tomar decisões e então se locomover.

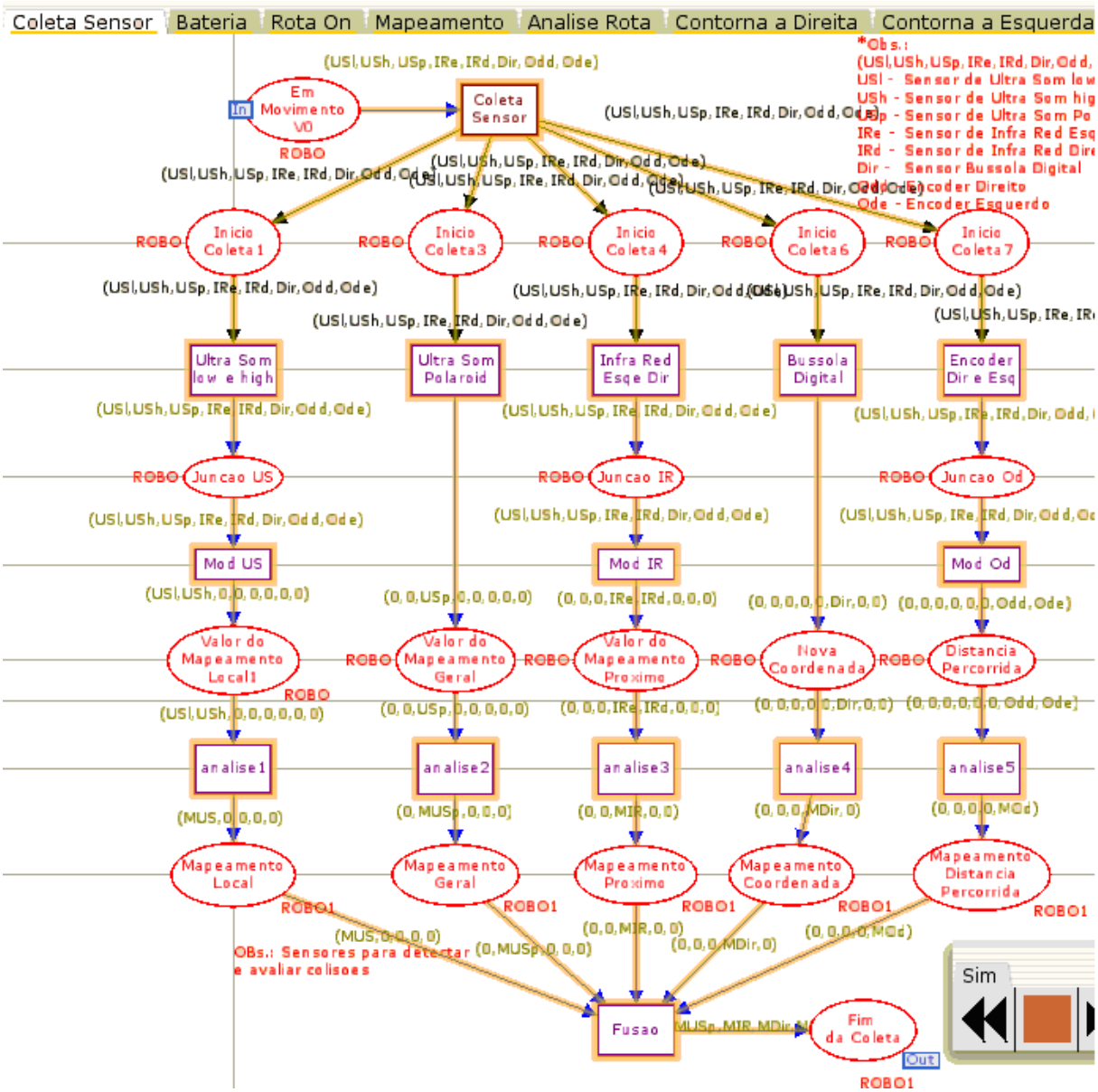

Figura 4.3: módulo “Coleta Sensor”.

Após a seção de “Sensoriamento” o robô inicia a “Navegação” (Figura 4.4) onde o controlador autônomo utiliza os dados provindos dos sensores do robô para produzir uma nova direção de navegação.

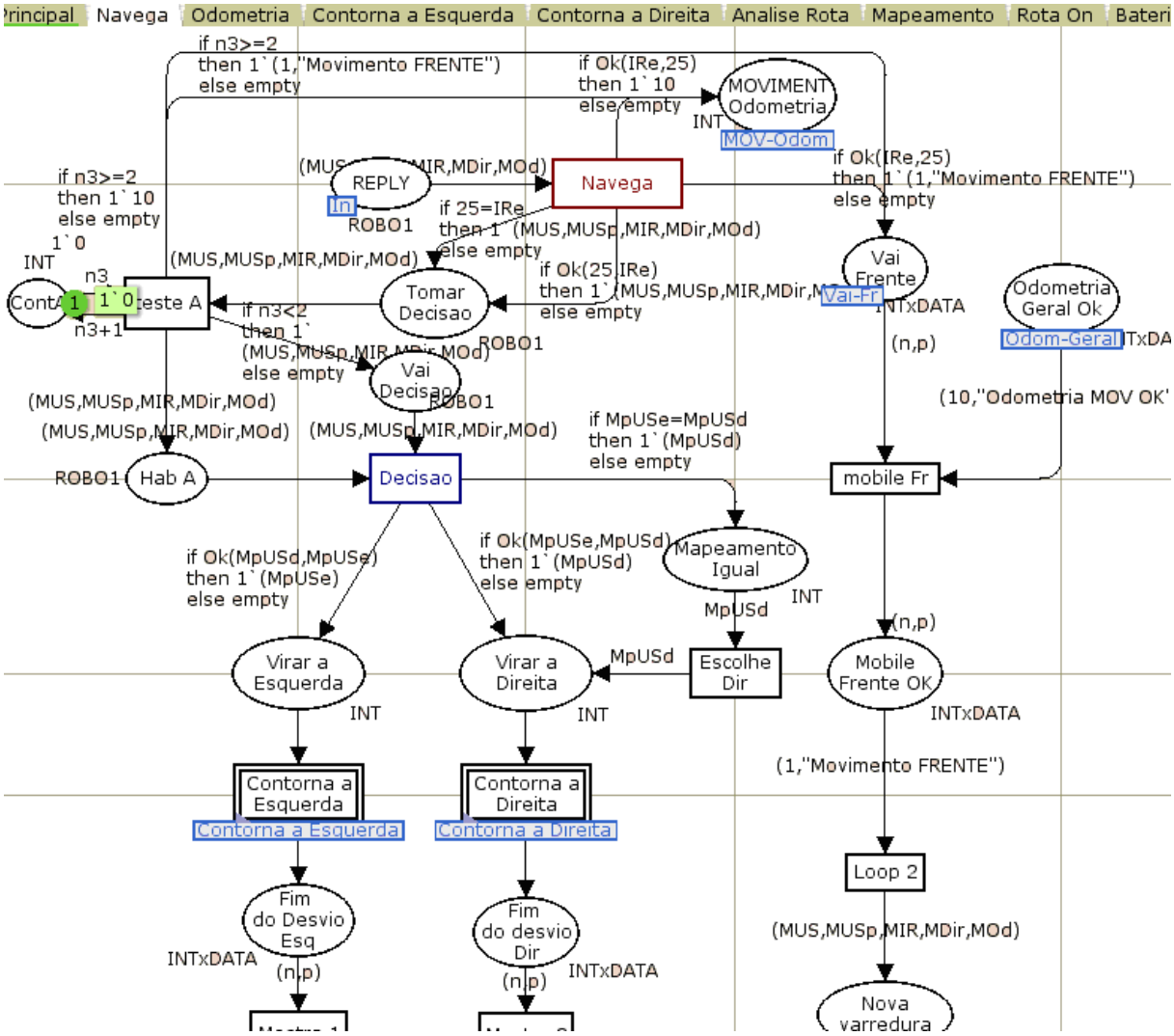

Figura 4.4: módulo “Navegação”.

Desde que um obstáculo é detectado (inclusão de uma ficha no lugar “Tomar Decisão”), o comportamento de evitar obstáculo ocorre com o disparo da transição “Decisão”, como pode ser observado na Figura 4.4. A partir daí ocorre um planejamento para contornar o obstáculo à direita ou à esquerda (Figura 4.5) que vai depender de um novo sensoriamento. Se não, se nenhum obstáculo for encontrado o robô continua seu trajeto pré-estabelecido. Nesta subpágina (Figura 4.5) pode-se acompanhar passo a passo o robô contornando o obstáculo de forma seqüencial, inicialmente o robô aproxima-se 10 centímetros do obstáculo (lugar “Aprox 10cm MovE”), dado que a especificação da distância crítica para que o robô tome a decisão de contornar obstáculo é de 25 centímetros, em seguida faz um giro de 90 graus (lugar “Rotacionado 90 graus a esquerda”), após o giro o mesmo segue em frente até perceber que chegou à extremidade do obstáculo (lugar “Caminha E a esquerda”), daí desfaz o giro (lugar “Rotacionado 90 graus a direita”). Para continuar o movimento se faz necessária uma nova

varredura do ambiente, já que neste momento o robô encontra-se à esquerda do obstáculo e precisa saber o quanto terá que percorrer ao lado do obstáculo. Continuando esta análise pode-se perceber que o robô contornará todo o obstáculo. Ressalta-se que todos estes movimentos do robô estão sendo mapeados no módulo "Mapeamento" que será descrito à frente. A modelagem do módulo "Contorna a Direita" é similar ao módulo "Contorna a Esquerda", a seqüência de disparos é de forma análoga, mudando apenas as direções de movimento.

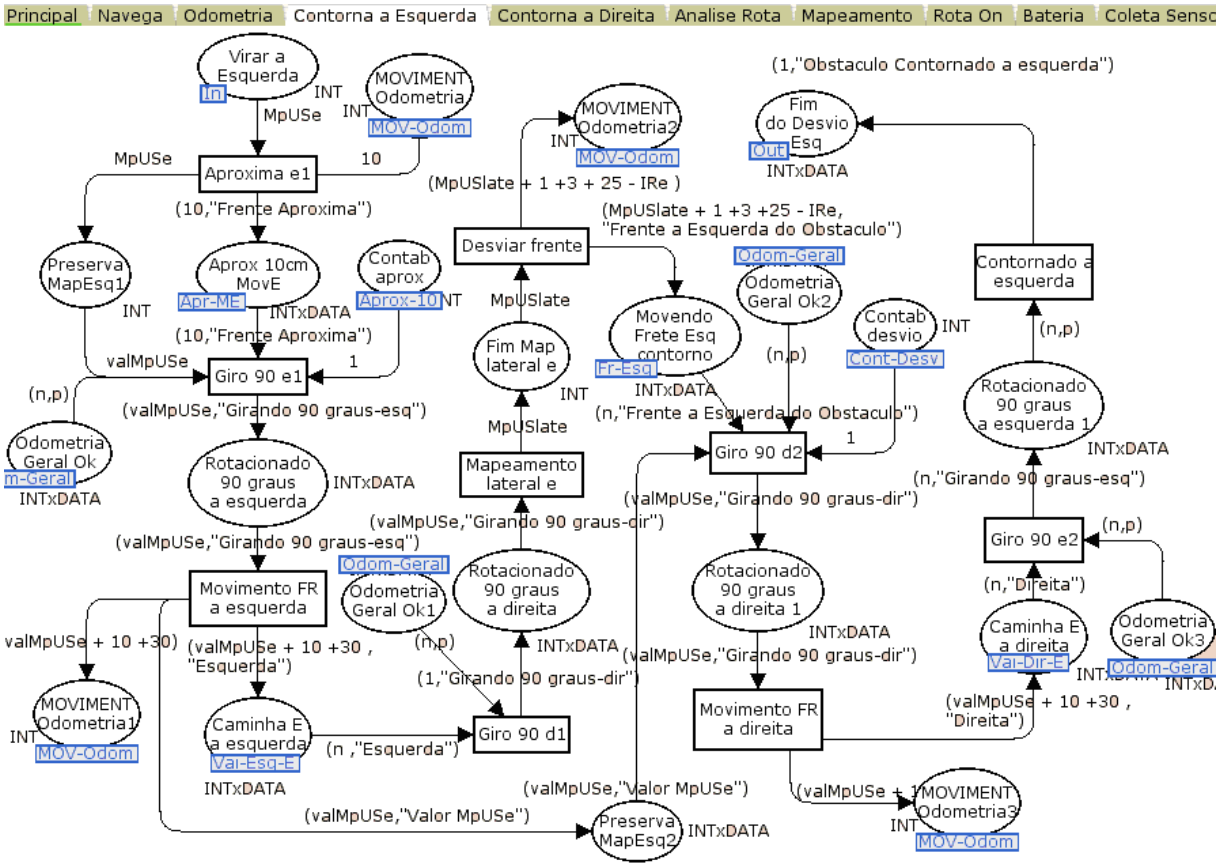

Figura 4.5: módulo "Contorna a Esquerda".

Terminada a seção de "Navegação", o sistema inicia a seção de "Atualização", onde a posição do robô é alterada no ambiente. O robô então avança na direção estabelecida pela seção de Navegação.

Na seção de "Atualização" destaca-se o módulo de "Odometria", que é responsável pela simulação do controlador de movimentos e a localização do robô (Figura 4.6). Neste módulo pode-se observar que no lugar "Acumulador Movimento" acontece a soma de todos os deslocamentos de posição do robô. Esta informação juntamente com a carga da bateria é utilizada pelo módulo "Bateria", que trata da otimização do uso das baterias do robô.

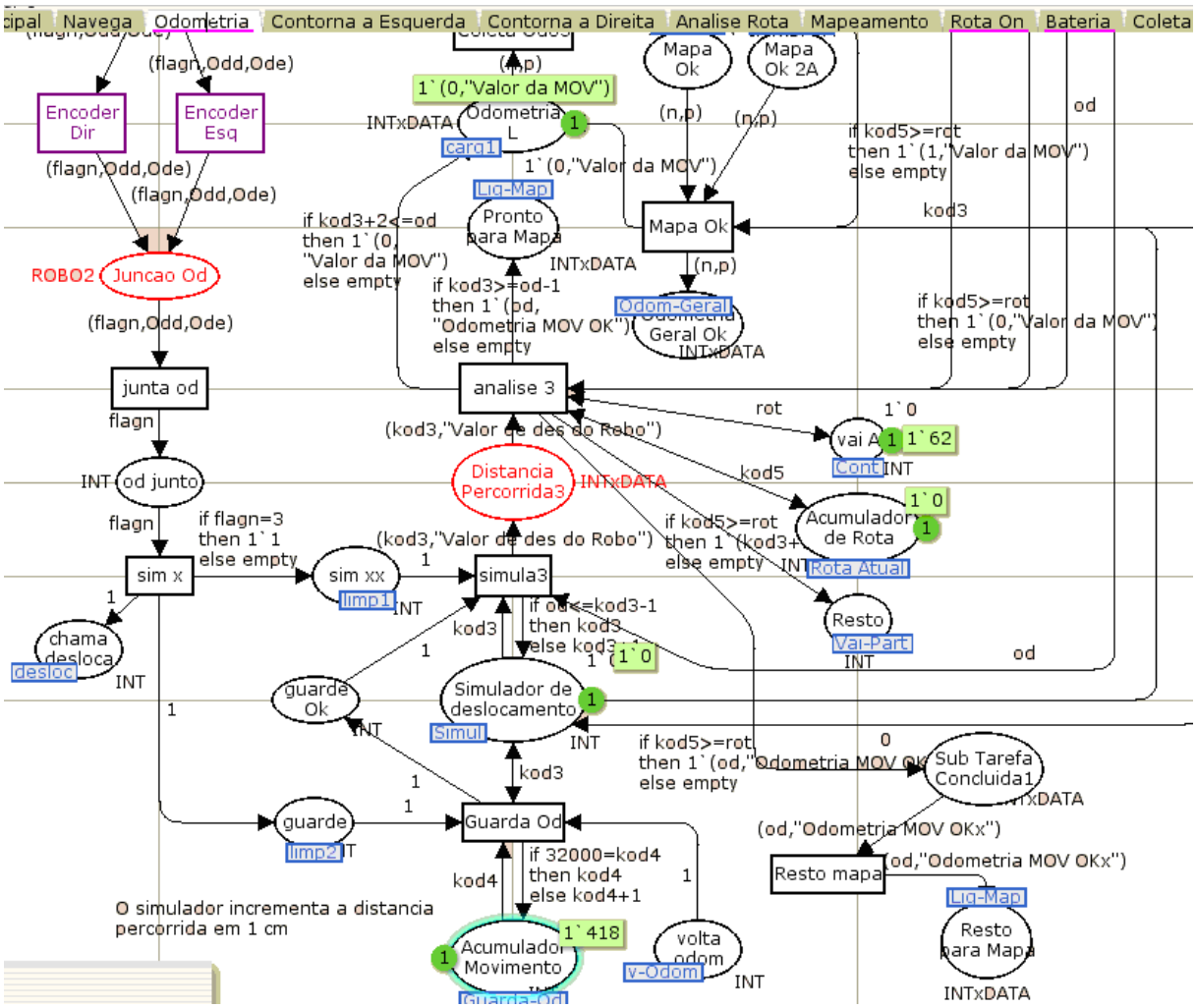

Figura 4.6: módulo “Odometria”.

Quando a rede está sendo executada, a simulação de movimento ocorre a cada 1 centímetro. Então a cada centímetro que o robô se movimenta, os módulos “Rota On” e “Analise Rotas”, são acionados respectivamente. O módulo “Rotas On” (Figura 4.7) analisa se a tarefa ou determinada trajetória foi concluída com sucesso, por meio das transições: “x1”, “x2”, “x3”, “x4” e “x fr”, no qual comparam a trajetória desejada (lugar “Acumulador de Rota”) com a trajetória que está sendo executada (lugares “Vai Frente” “Aprox 10cm MovE” “Aprox 10cm MovD” “Movendo Frete Dir Contorno” “Movendo Frete Esq contorno”).

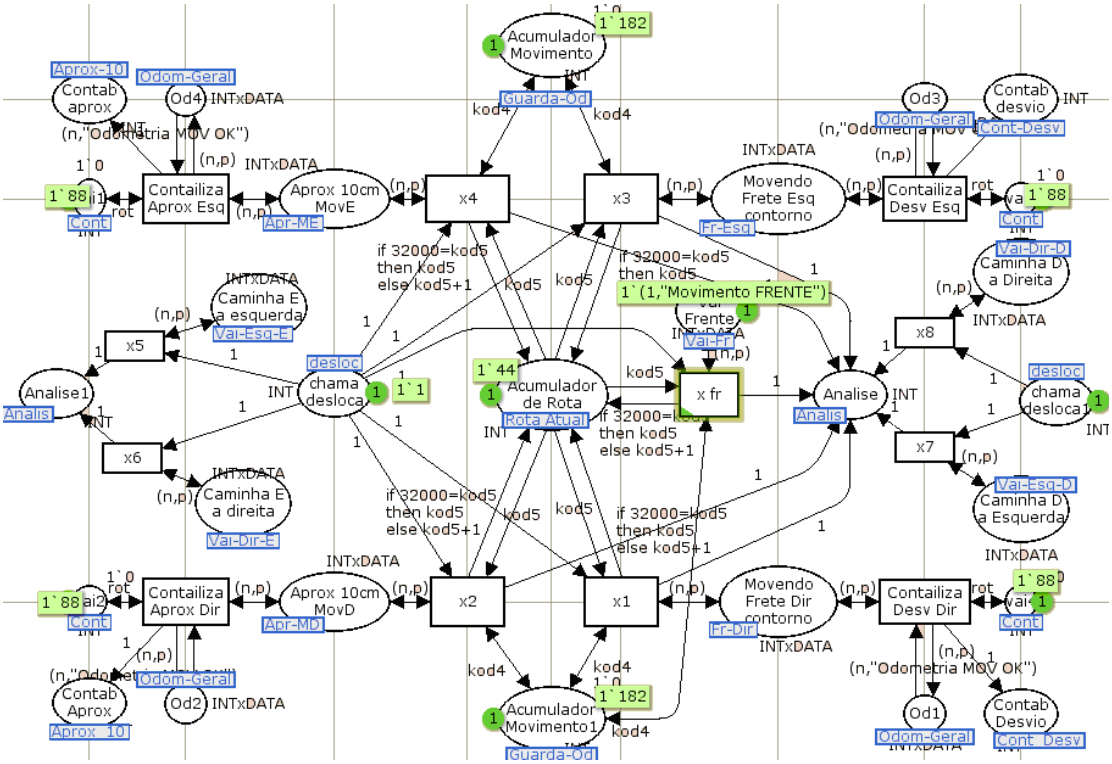

Figura 4.7: módulo “Rota On”.

O módulo “Analise Rota” apresentado na Figura 4.8, trata de observar se as sub-rotas inicialmente carregadas estão sendo executadas corretamente. No lugar “vai” encontra-se a sub-rota que está sendo executada e o lugar “Analise” representa o estado de execução em que se encontra esta sub-rota e, quando a mesma for concluída, uma ficha é adicionada no lugar “Nova Rota” que vai habilitar a transição “Proxima” (Figura 4.2). Logo que esta transição for disparada, a transição “Seq Rotas” estará novamente habilitada. Assim acontecerá todo o processo descrito anteriormente para uma nova sub-rota.

Por exemplo, em um dos testes de simulação, ao final da execução de 4297 passos da rede, a tarefa pré-estabelecida foi concluída e a rede ficou no estado de aguardar que uma nova rota fosse carregada, enquanto que no módulo de “Mapeamento”, observaram-se as 45 fichas do lugar “Guarda Mapa” da rede, as quais representam incrementos de posição da trajetória seguida pelo robô, ou seja, um rastreamento completo do robô, como pode ser visto na Figura 4.9. Ao longo de toda a simulação da rede, diversos dados considerados importantes para a análise do desempenho do robô podem ser coletados.

Os resultados apontam que o uso do formalismo proposto é capaz de determinar políticas de controle, de ánalise e de identificação de conflitos com facilidade antes da implementação do sistema.

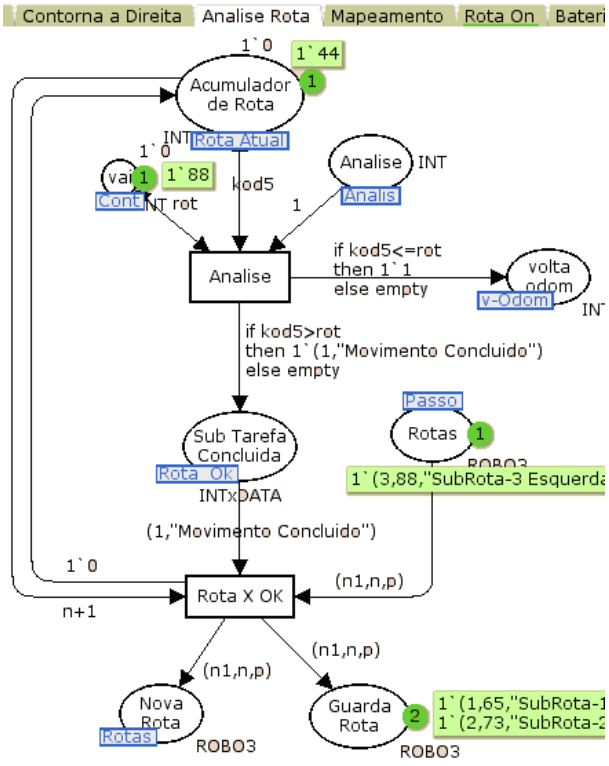

Figura 4.8: módulo “Analise Rota”.

Na simulação mostrada na Figura 4.9, observa-se que não foi encontrado nenhum obstáculo na roda pré-estabelecida, isto é representado pela marcação do lugar “Guarda Mapa”, que possui fichas com quatro elementos, por exemplo, a ficha 1(1,65,"Frente","SubRota-1 Frente"), onde o primeiro elemento denota a ordem dos movimentos do robô, o segundo o quanto o robô se deslocou em centímetros, o terceiro a direção tomada em cada movimento e o quarto indica qual das sub-rotas, inicialmente carregadas (Figura 4.2), foi executada. De posse desta marcação, analisando-se o terceiro elemento, nota-se que existem apenas direções do tipo “frente”, ou seja, o robô manteve-se na rota durante todo o percurso, já em casos em que obstáculos são encontrados, este mesmo elemento trás informações do tipo: a direita, a esquerda, contornando a direita do obstáculo e contornando a esquerda do obstáculo, com isso podemos dizer quantos e aonde foi encontrado cada obstáculo, o tamanho, como foi contornado, etc. Outra forma de observar a presença de obstáculos na rota seguida pelo robô é por meio da marcação do lugar “Acumulador Movimento” presente no Módulo “Odometria” (Figura 4.6). Esta marcação apresenta o valor dos deslocamentos do robô em centímetro, então, se este valor for maior que a rota pré-estabelecida, indica que ocorreu um desvio na rota.

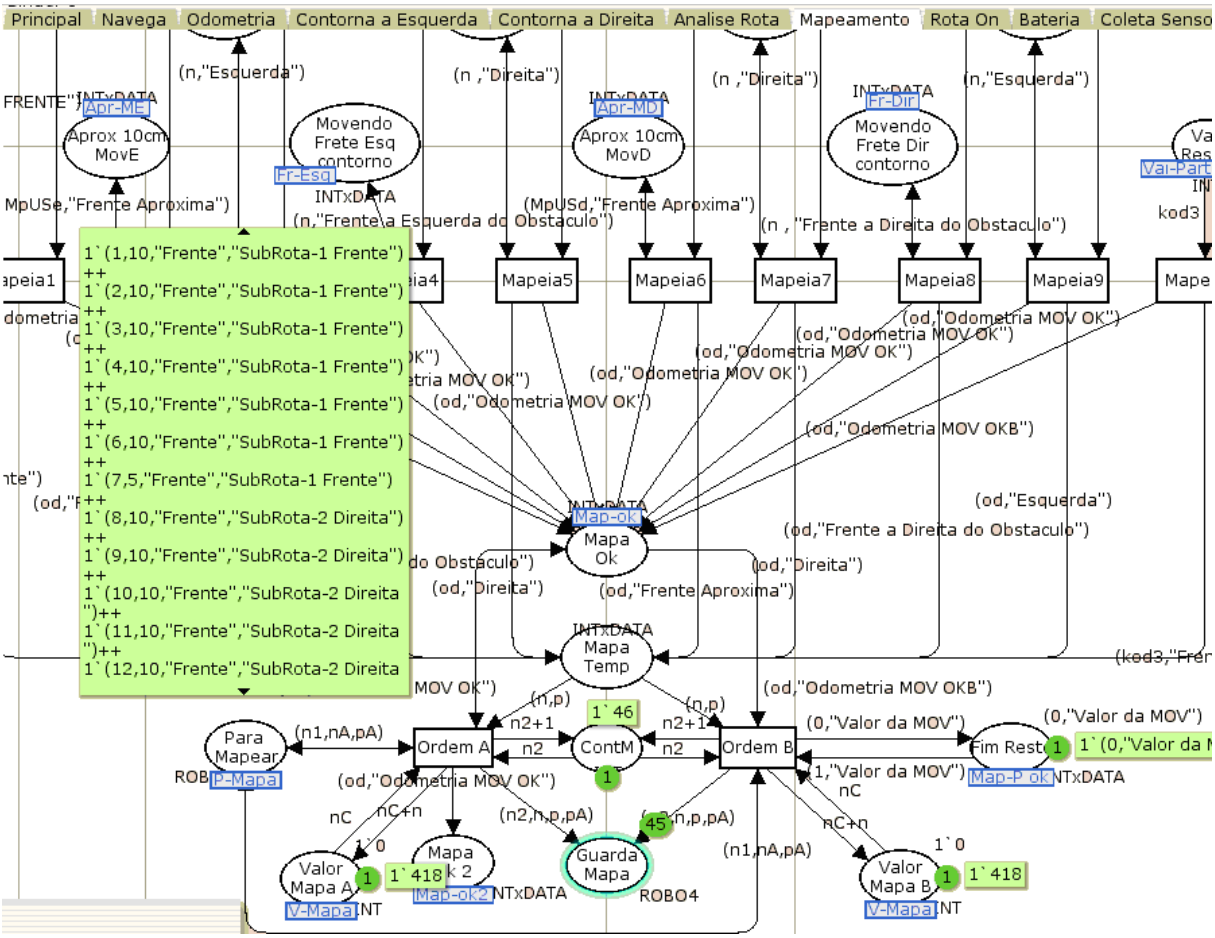

Figura 4.9: módulo “Mapeamento”.

Foram realizados vários testes de simulação contendo rotas diferenciadas, com e sem obstáculos e, em todos eles o robô realiza a sua tarefa, ou seja, alcança o destino especificado e retorna para um estado em que aguarda uma nova rota. Ao longo de cada simulação, foi possível extrair dados como nível de tensão da bateria, número de obstáculos contornados, tomadas de decisão, distância percorrida (rastreamento de posição), conforme o exemplo apresentado anteriormente. Observa-se que o robô é autônomo, pois o mesmo é capaz de tomar decisões para desviar de obstáculos no desvio de obstáculos e retornar a base quanto a tensão da bateria esta abaixo do nível estabelecido.

4.3 Controle digital de velocidade dos motores CC

O controle digital é de grande utilidade em sistemas de automação e controle. Ele é usado como instrumento para exercer o controle de posição ou o controle de velocidade ou

de ambos. A representação esquemática de um motor DC de fluxo constante e excitado por armadura é apresentada na Figura 4.10 (Souza e Costa Filho, 2001).

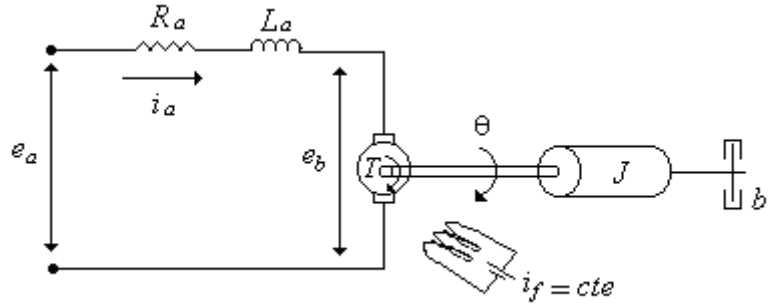

Figura 4.10: Representação Esquemática do Motor DC de Fluxo Constante

A função de transferência simplificada do motor, desprezando o atraso elétrico em relação ao atraso mecânico, é vista na equação (4.1).

$$F(s) = \frac{W(s)}{E(s)} = \frac{K}{R_a Js + R_a b + KK_b} \tag{4.1}$$

Sendo:

E(s) a tensão de armadura no plano s;

K a constante de armadura;

Ra a resistência da armadura;

W(s) a velocidade angular do eixo do motor no plano s;

J a inércia em relação ao eixo do motor e de sua carga;

b o coeficiente de atrito que atua sobre o eixo.

Na equação 4.1, temos outra forma de escrever a função de transferência.

$$F(s) = \frac{W(s)}{E_a(s)} = \frac{K_m}{T_m s + 1} \tag{4.1}$$

Na equação 4.1, aplicando-se s = 0, tem-se o ganho do motor em regime permanente que é visto na equação (4.2). Pode-se escrever a função de transferência do motor

como na equação (4.3). Onde τ é a constante de tempo do sistema, ou precisamente, τ é o tempo que a saída leva pra atingir 66,6% do valor em regime permanente.

$$G = \frac{K_m}{R_a b + k^2} \tag{4.2}$$

$$F(s) = \frac{G}{\tau s + 1} \tag{4.3}$$

Utilizando o robô móvel experimental PISA G1, é executado um programa que aplica um degrau na entrada e recolhe dados da saída, pode-se visualizar a resposta do motor CC de tração do robô ao degrau de amplitude 10,4 Volts, na Figura 4.11.

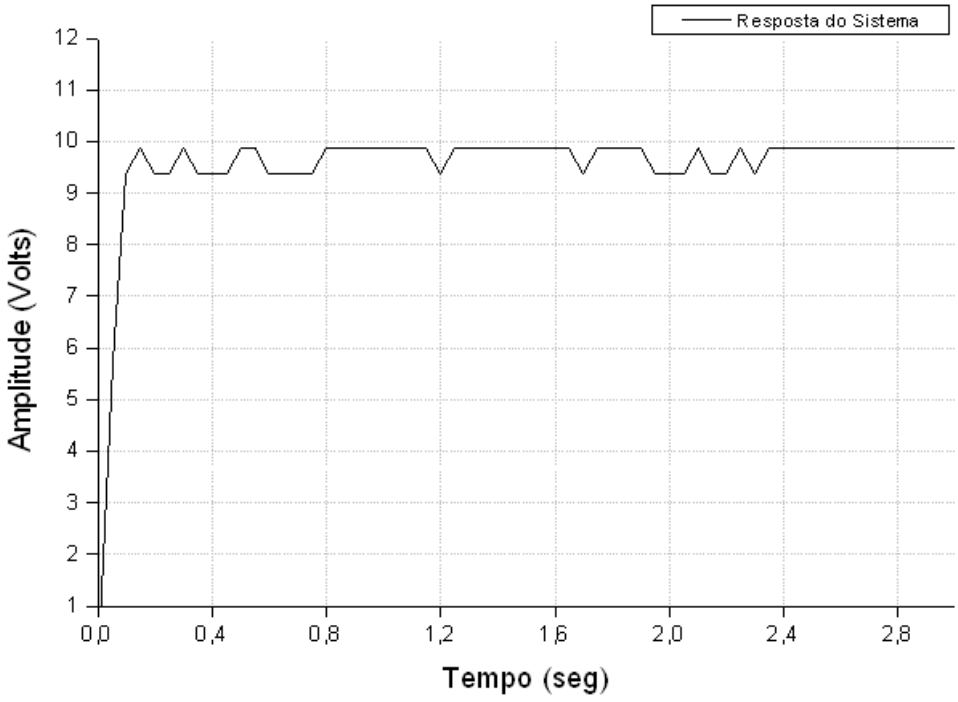

Figura 4.11: Resposta do motor CC ao degrau de amplitude 10,4 Volts.

Na Figura 4.11 é mostrado que a constante de tempo do motor é aproximadamente 0,12seg e o ganho do motor é aproximadamente 0,942, que é obtido da divisão entre o valor da amplitude do degrau, 10,4V, pelo valor em regime 9,8V.

Aplicando os valores de G e τ na equação 4.3, resulta na função de transferência vista na equação 4.4. Na Figura 4.12 é mostrada a resposta contínua simulada, usando o MATLAB, com a função de transferência vista na equação 4.4.

$$F(s) = \frac{0{,}942}{0{,}12s + 1} \tag{4.4}$$

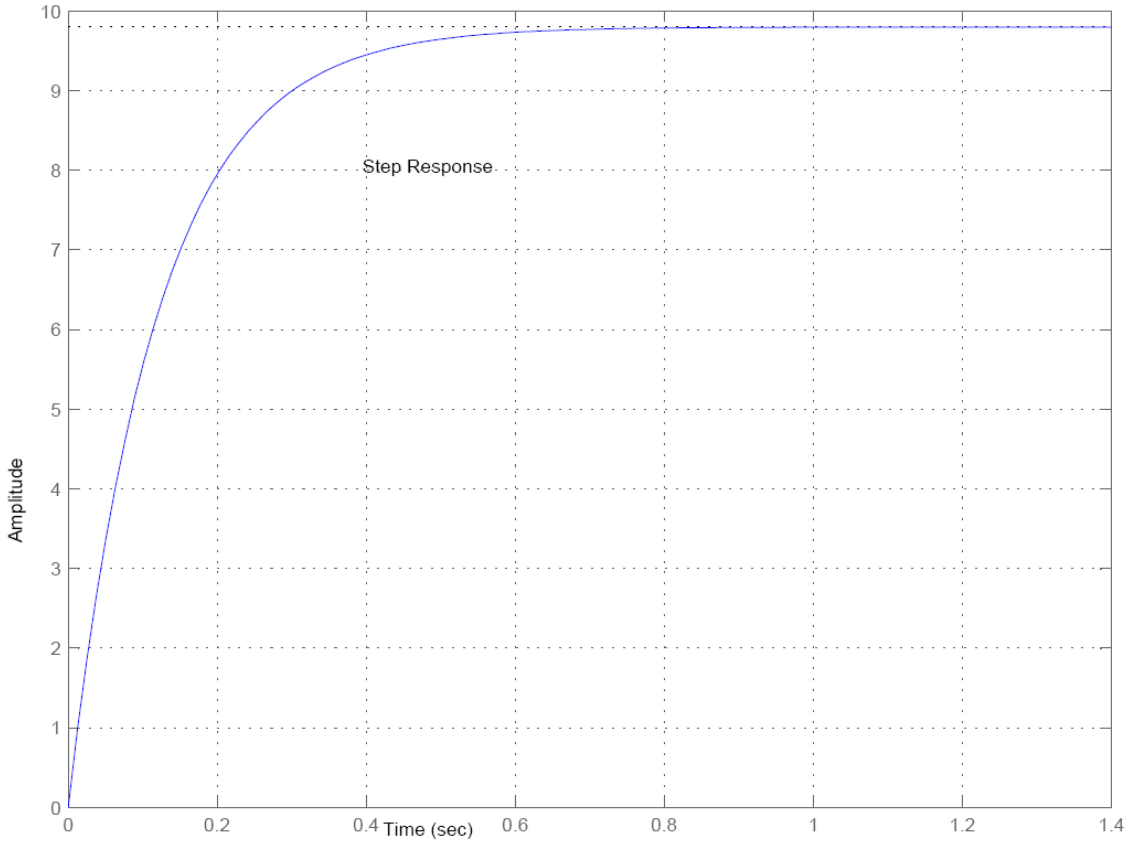

Figura 4.12: Resposta contínua simulada.

Percebe-se que a resposta discreta corresponde à resposta contínua, então o modelo de função de transferência para o motor CC em questão é consistente.

4.3.1 Controle digital

A determinação dos parâmetros do controlador é dada como se segue. A partir da Figura 4.12, têm-se os seguintes dados:

- Tempo de atraso: 0.01 segundos (valor teórico)
- Constante de tempo: 0.045 segundos

Aplicando o método de *Ziegler-Nichols*, chega-se aos seguintes valores para K_p, T_d e T_i para o controlador PID, apresentados na Tabela 4.1.

Tabela 4.1: Parâmetros do controlador PID projetados pelo método de Ziegler-Nichols para o controle de velocidade do motor CC.

Tipo de controlador	**Parâmetros**		
	K_p	T_i	T_d
PID	14,4	0,02	0,005

Na função de transferência mostrada na equação 4.4 não está presente o tempo de atraso do processo. Por aferição, o tempo morto é em torno de 0,01seg. Assim a função de transferência simplificada do processo é mostrada na equação 4.5.

$$\frac{Y(s)}{X(s)} = \frac{0{,}942e^{-0{,}01s}}{0{,}12s+1} \tag{4.5}$$

De acordo com o método de *Ziegler-Nichols*, considerando T = 0,0452seg, G = 0,942 e d = 0,01seg, os parâmetros do controlador PID são dados por:

$$\begin{aligned} K_p &= 1{,}2\frac{T}{d} = 5{,}4 \\ T_i &= 2d = 0{,}02 \\ T_d &= 0{,}5d = 0{,}005 \end{aligned} \tag{4.6}$$

Assim, a função de transferência do controlador é:

$$G(s) = 5{,}4\left[1 + \frac{1}{0{,}02s} + 0{,}005s\right]$$

$$G(s) = 0{,}027\left(\frac{(s+100)^2}{s}\right) \tag{4.7}$$

Na Figura 4.13 é mostrada a resposta simulada ao degrau do processo, utilizando o controlador PID obtido pelo método de *Ziegler-Nichols*. A tabela 4.2 mostra o desempenho do controlador PID no processo de controle do motor CC.

Tabela 4.2: Desempenho do processo controlado PID, pelo método de Ziegler-Nichols.

Tipo de controlador	**Desempenho**		
	Tempo de acomodação	**Sobre-sinal máximo**	**Erro de regime permanente**
PID	2 Seg.	5%	0

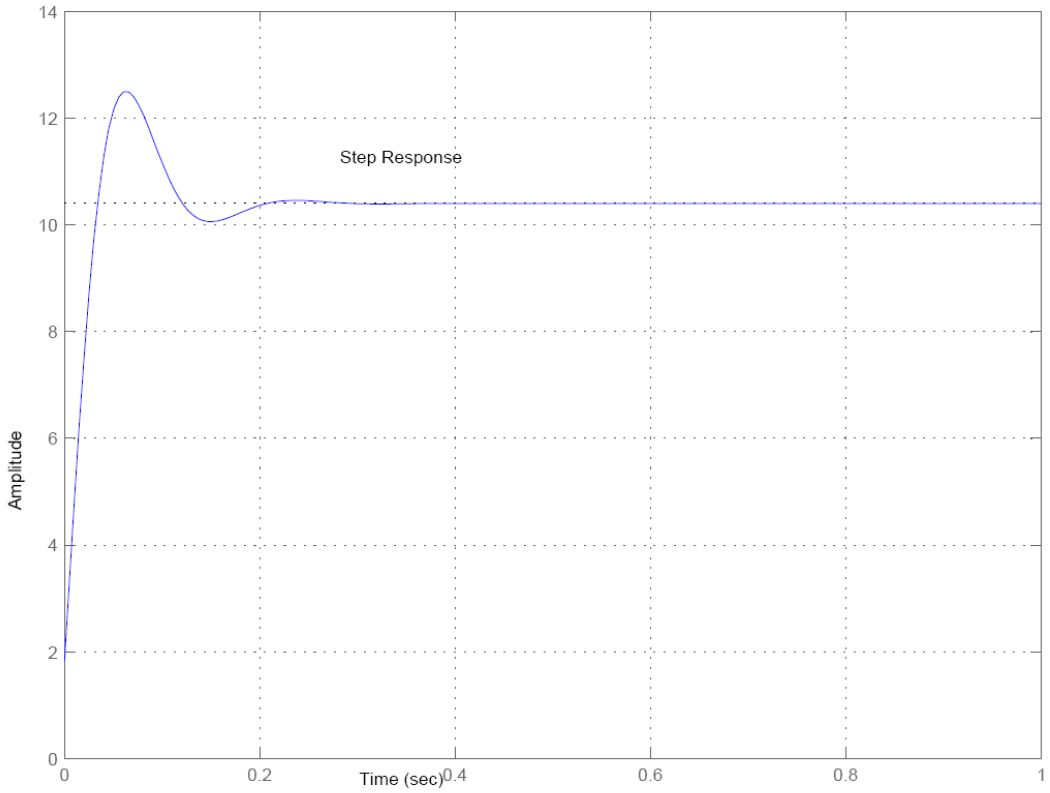

Figura 4.13: Desempenho do controlador PID obtido pelo método de Ziegler-Nichols.

4.3.2 Discretização do controlador PID

De acordo com a seção anterior tem-se que, K_p=5,4, T_i=0.02seg, T_d=0,050seg, assim os parâmetros do controlador PID discreto serão:

$$q_0 = 5{,}1$$
$$q_1 = 7{,}02 \tag{4.8}$$
$$q_2 = 0{,}54$$

Deste modo, de acordo com a equação 3.20, o algoritmo de controle PID é:

$$u(k) - u(k-1) = 5{,}1e(k) + 7{,}02e(k-1) + 0{,}54e(k-2) \tag{4.9}$$

Na prática, com os valores de q_0, q_1 e q_2, vistos na equação 4.9, ocorre saturação do atuador, pois esse algoritmo requer tensões de u(k) acima da fornecida pela bateria. Neste caso usou-se método de tentativa e erro tomando como referência os valores da equação 4.9. O melhor resultado, não ocorrendo saturação do atuador e o desempenho do controlador sendo satisfatório, é o apresentado na equação 4.10.

$$u(k) - u(k-1) = 2{,}5e(k) - 2{,}8e(k-1) + 0{,}5e(k-2) \tag{4.10}$$

A resposta do sistema, velocidade do motor CC, é vista na figura 4.14.

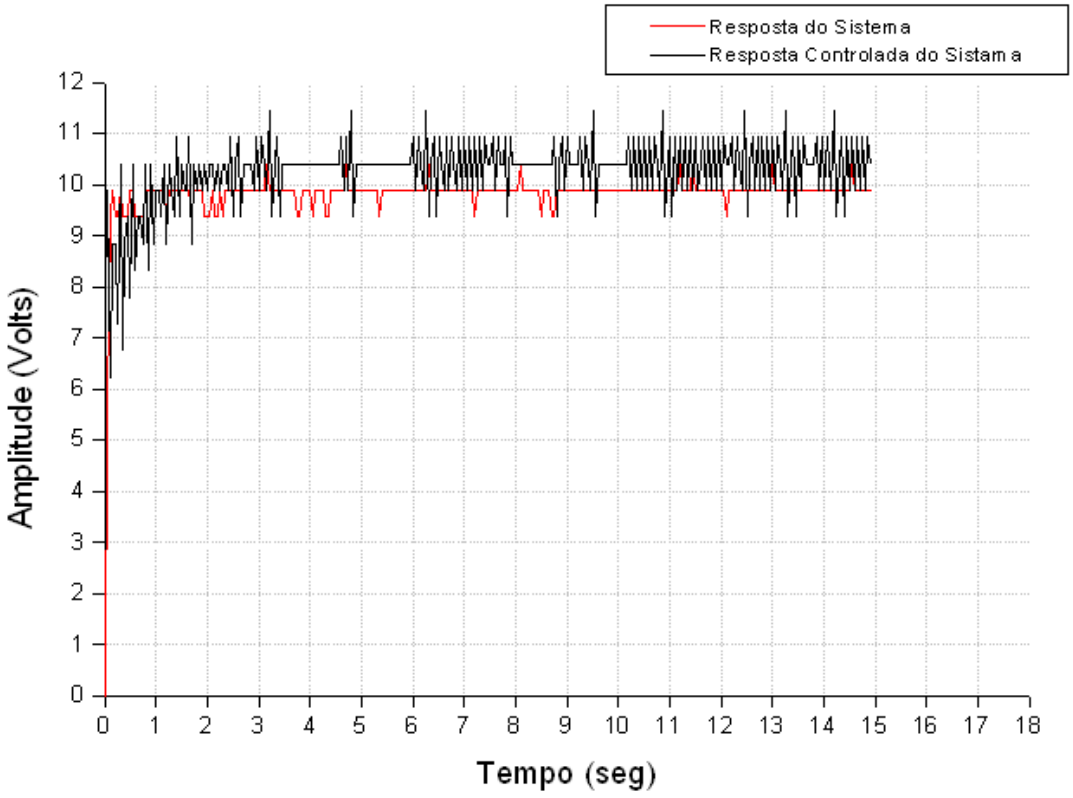

Figura 4.14: Resposta do sistema utilizando o algoritmo PID discreto.

Na Tabela 4.3 é mostrado o desempenho do processo controlado via algoritmo PID discreto.

Tabela 4.3: Desempenho do processo controlado via algoritmo PID discreto.

Tipo de controlador	**Desempenho**		
	Tempo de acomodação	**Sobre-sinal máximo**	**Erro de regime permanente**
PID	2 Seg.	5%	4,5%

4.4 Navegação com o robô real: experimentos e resultados

Para validação do modelo de navegação apresentado nesta dissertação, foram realizados alguns experimentos. Para tal, foi utilizada a plataforma experimental PISA G1, já descrita.

4.4.1 Experimento 1 – Navegação sem obstáculos

Na Figura 4.15 é ilustrada uma execução do experimento relativo à navegação em um ambiente sem obstáculos, onde é mostrado o robô nas posições inicial e final, um gráfico das trajetórias descritas pelo robô durante experimento. Alguns pontos de controle sobre a trajetória de referencia são indicados pelas letras A, B, C, D e E, sendo composta de sete segmentos de reta. Ao robô foi designada a tarefa de navegar até o ponto de coordenadas (1300; 1000), partindo do ponto (0; 0). A posição inicial é definida de forma que o erro seja dado por $\Gamma = \Gamma_0$ e $\Theta = 0$. Neste teste, os obstáculos inicialmente fornecidos ao robô foram às paredes que limitam o ambiente.

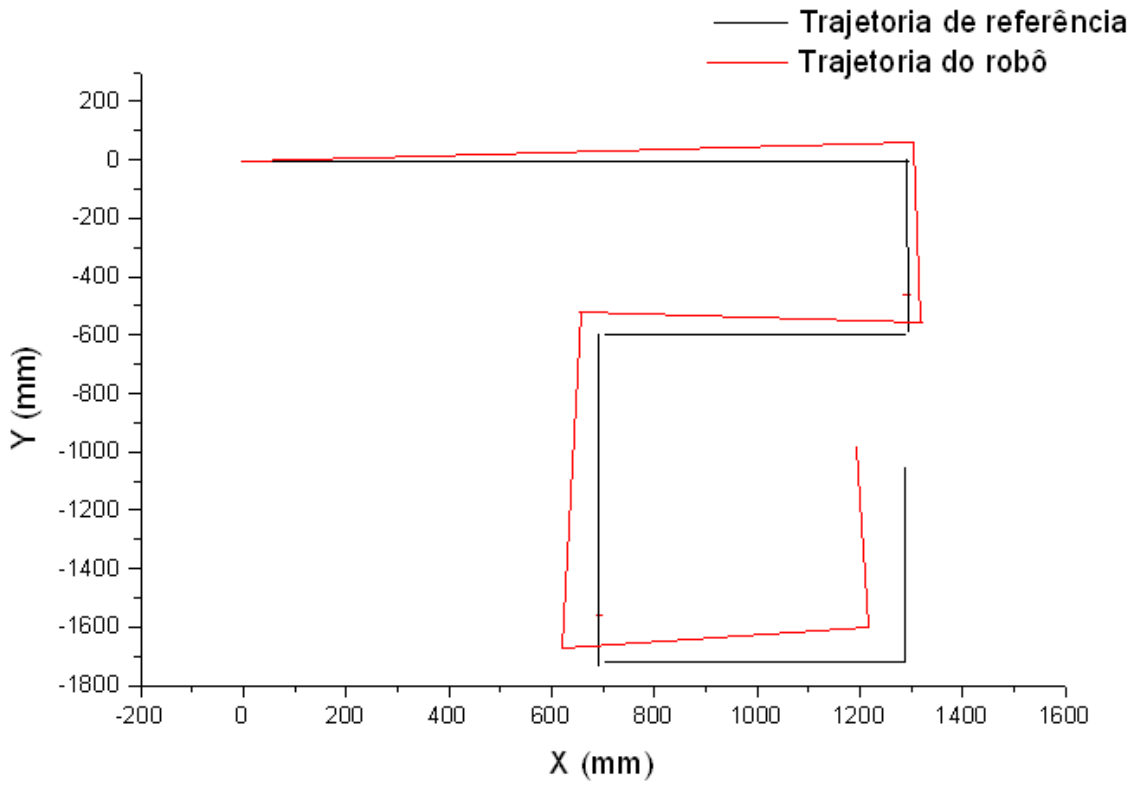

Figura 4.15: Experimento de navegação sem obstáculos.

4.4.2 Experimento 2 – Navegação com obstáculos

Este experimento mostra o comportamento do robô quando submetido à mesma configuração do ambiente utilizado no experimento anterior, diferenciando-se apenas em relação ao acréscimo de dois obstáculos cilíndricos de raio 10 cm situados em (600mm; 0mm), e (700mm; -1200mm), respectivamente. O layout do ambiente de trabalho correspondente a este experimento é representado na Figura 4.16, juntamente com a trajetória que o robô descreve.

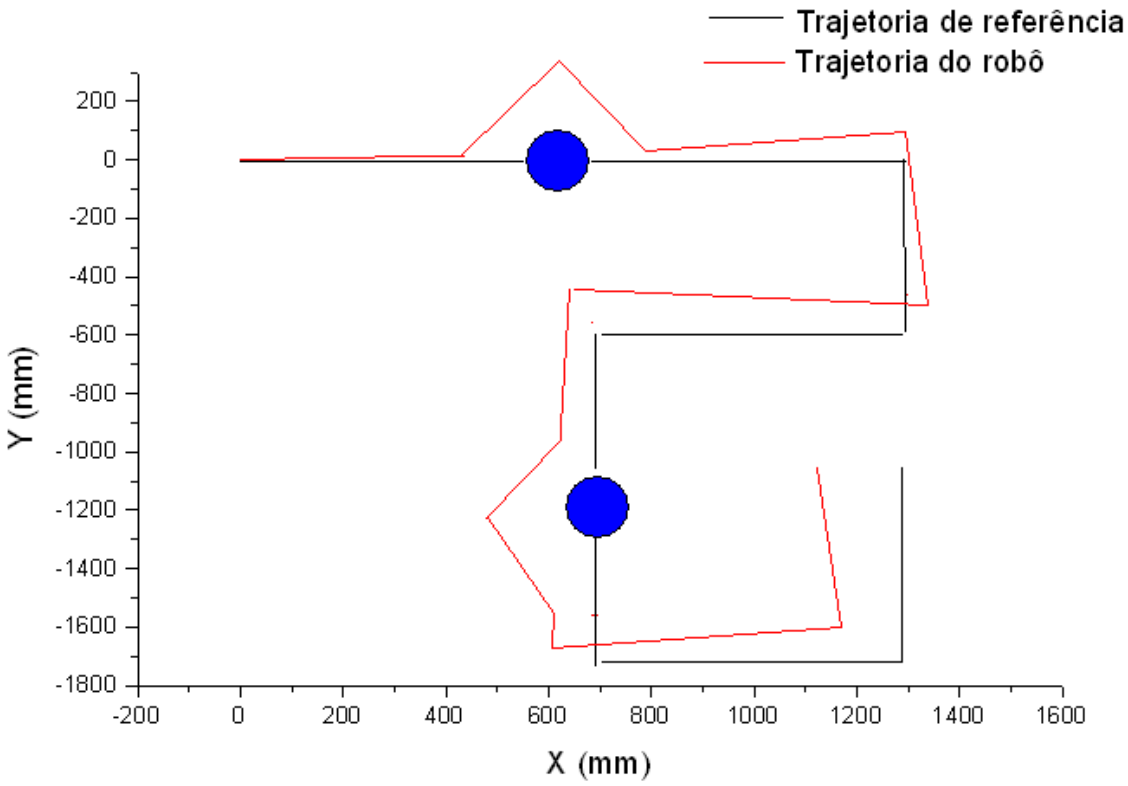

Figura 4.16: Experimento de navegação com obstáculos.

4.5 Discussão dos resultados experimentais de navegação

Os resultados obtidos foram bastante satisfatórios, sendo que o robô adquiriu assim a capacidade de locomover-se autonomamente por entre os obstáculos detectados pelo uso de seus sonares. Apenas alguns problemas na detecção dos obstáculos foram identificados, como já era esperado, dada a natureza do método utilizado (uso de sonares). Notou-se também que a performance de tempo como um todo ficou prejudicada, uma vez

que, com o programa utilizado para detecção de obstáculos, é necessário que o robô interrompa seu movimento para efetuar a identificação dos obstáculos através dos sonares.

No que tange ao alcance da posição desejada e planejamento de trajetória o resultado foi satisfatório. Contudo, o tempo que o robô levou para atingir o seu objetivo foi relativamente elevado, isso se deveu ao grande número de vezes que ele precisou parar para detectar novos obstáculos. Percebeu-se também em outros testes, com o método implementado, que o robô às vezes não tenta passar em locais pelos quais ele poderia, este problema se deve a imprecisão dos sonares e a folga utilizada no raio do robô informado ao programa, necessária como medida de segurança em decorrência do primeiro problema.

Erros de odometria foram encontrados, sendo os mesmos "cumulativos" para um dado experimento. Em alguns casos, pontos de destino não eram alcançáveis, pois, de acordo com a odometria do robô, eles estavam distante do alvo. Uma fusão de sinais provenientes dos encoders e dos sonares talvez pudesse ser utilizada para obtenção de dados mais confiáveis.

Os erros, mostrados na Figura 4.15 e na Figura 4.16, se devem principalmente a erros relativos na odometria, praticamente inexistentes no modelo simulado, mas constantes na plataforma móvel. Além disso, a baixa precisão dos sonares influencia no instante em que um obstáculo é detectado, podendo gerar, portanto, reorientações do robô em pontos diferentes. No entanto, mesmo com as dificuldades encontradas, as trajetórias descritas foram consideradas adequadas e de acordo com o objetivo proposto neste trabalho.

CAPITULO V

5. Conclusão

Navegar em um ambiente desconhecido é uma tarefa extremamente complexa, salvo quando o ambiente de atuação do robô é limitado e cuidadosamente controlado, situação ideal que geralmente não acontece na maioria das aplicações realísticas.

Neste trabalho foi apresentado um modelo para navegação de robôs móveis baseado em RPC, tendo basicamente como objetivo guiar, sem qualquer auxílio externo, um robô móvel por ambientes arbitrários e inicialmente desconhecidos, cumprindo simultaneamente duas tarefas: evitar colisões em obstáculos e chegar ao destino pré-estabelecido. O modelo foi analisado e validado através de simulação, na qual mostrou que o robô é capaz de realizar uma rota pré-estabelecida mesmo na presença de obstáculos. Assim, o modelo RPC se comportou de forma eficiente e várias situações práticas no funcionamento de um SNA foram simuladas, utilizando-se o modelo proposto.

O SNA baseado em RPC e seus diversos aspectos foram também testados e validados por um extenso conjunto de experimentos de navegação com um robô móvel real. Os experimentos visaram demonstrar as virtudes do sistema e detectar deficiências. O SNA é reativo, isto é, o sistema atua com base em informações instantâneas capturadas pelos sensores e não com base em dados incorporados previamente.

A arquitetura do sistema de controle do robô móvel experimental PISA G1 está organizada de forma hierárquica, em três níveis de controle: sistema de navegação, controle de trajetória e o nível controle de velocidade. Quanto ao controle de velocidade implementado (do tipo PID), por si só já possibilitou manter o robô em linha reta. Já o controle de trajetória se mostrou suficiente para manter o robô dentro de uma trajetória pré-definida, possibilitando enviar o robô de uma forma mais eficiente para qualquer ponto definido a partir de sua localização atual, sendo o erro dado basicamente pela odometria. Durante a navegação do robô, um dos problemas encontrados foi referente ao uso dos sonares para detecção de obstáculos, dada a sua precisão inerente. Entretanto, o sistema desenvolvido é suficientemente genérico, de forma que possa ser utilizado juntamente com outros métodos de detecção de obstáculos, como através sensores de laser, câmeras de vídeo e etc.

A arquitetura de controle implementada apresentou um bom desempenho no robô móvel experimental PISA G1. Além disso, esta plataforma se mostrou excelente para a implementação de novas técnicas de controle e sensoriamento.

Trabalhos futuros

Dando seqüência ao trabalho realizado, são pertinentes as seguintes propostas para trabalhos futuros:

- Desenvolvimento de um sistema de sensoriamento mais completo, usando acelerômetros, cinturão ultra-sônico e etc, fazendo então uma abordagem de fusão sensorial.
- Desenvolvimento de um novo hardware de controle baseado em processadores mais rápidos, como o Processadores Digitais de Sinal - DSP;
- Implementação de um controlador de trajetórias mais robusto, tal como controladores não-linear;
- Utilizar as Redes de Petri Coloridas, para modelar SNAs mais complexos.
- No tocante ao sistema de navegação, a hibridização do SNA desenvolvido neste trabalho com as técnicas de Inteligência Artificiais - IA, tendo em vista que as técnicas de IA apresentam propriedades muito interessantes de aprendizagem, adaptação e auto-organização.
- Como o elenco de experimentos realizados e descritos ao longo do texto não foi exaustivo, planeja-se a realização de uma variedade maior de experimentos, especialmente no tocante aos experimentos reais, visando basicamente expor o SNA a novas situações e abrindo espaço para a navegação envolvendo múltiplos robôs, com cada um deles representando um obstáculo móvel para os demais e contando com a possibilidade de interação dos robôs, mais especificamente troca de mensagens.

Referências

Borenstein, H. R., et al., 1996. **Navigation Móbile Robots: Systems and Techniques**, AK Peters.

Borges, G., Lima, A., Deep, G., 2000. **Design of an Output Feedback Trajectory Controller for an Automated Guided Vehicle**. XIII Congresso Brasileiro de Automática.

Borges, G., Lima, A., Deep, G., 2003. **Controladores Cinemáticos de Trajetória para Robôs Móveis com Tração Diferencial**. VI Simpósio Brasileiro de Automação Inteligente.

Brockett, R. W., 1983. **Asymptotic Stability and Feedback Stabilization.** In Differential Geometric Control Theory, Brockett, Millman & Sussman. Eds, Boston – MA: Birkhauser, 1983, p. 1983, p. 181-208.

Brooks, R., 1986. **A robust layered control system for a mobile robot**. IEEE Journal of Robotics and Automation 2, 1 (Mar. 1986), 14.23.

Caloini, A. et al., 1998, **A Technique for Designing Robotic Control Systems based on Petri Nets**, IEEE Transactions on Control Systems Technology, Vol. 6, No. 1, pp. 72-86.

Campion, G.; Bastin, G. e D'Andréa-Novel, B., 1995. **Structural Properties and classification of Kinematic and Dynamical Models of Wheeled Mobile Robots.** IEEE Transactions on Robotics and Automation, Vol. 12, No. 1, pp. 47-62.

Carvalho, H. J. R. et al., 2005, **Modeling of a control architecture for a mini-robot navigation using petri nets**, Proceedings of the 18th International Congress of Mechanical Engineering. Ouro Preto, MG, Brazil.

Cazangi, R. R., 2004. **Uma Proposta Evolutiva para Controle Inteligente em Navegação Autônoma de Robôs**, Tese de Mestrado, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, SP.

Colbaugh, R., 2000. **Tracking control of uncertain dynamic nonholonomic system and its application to wheeled mobile robots**, IEEE Transaction on Robotics and Automation, Vol. 16, No. 6, pp. 870-874.

Coste-Maniére, E. e Simmons, R., 2000. **Architecture, the backbone of robotic systems**. IEEE International Conference on Robotics and Automation.

Hale, R.D., Rokonuzzaman, M., Gosine, R.G., 1999, **Control of Mobile Robots in Unstructured Environments Using Discrete Event Modeling**, Proceedings of SPIE - The International Society for Optical Engineering, Vol. 3838, Boston, MA, USA, pp. 275-282.

Hermely, E. M., 1996, **Controle por Computador de Sistemas Dinâmicos.** São Paulo: Edgard Blucher.

Jensen, K., 1997. **Coloured petri net: Basic concepts**, Berlin: Springer 2nd edition.

Kuhne, F., 2005. **Controle Preditivo de Robôs Móveis Não Holonômicos**, Dissertação de Mestrado, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS.

Lages, W. F., 1998. **Controle e Estimação de Posição e Orientação de Robôs Móveis**, Tese de Doutorado, Instituto Tecnológico de Aeronáutica , São José dos Campos, SP.

Lima, P. et al., 1998, **Petri Nets for Modeling and Coordination of Robotic Tasks**, Proceedings of the IEEE: International Conference on Systems, Man and Cybernetics, Vol. 1, pp. 190-195.

Mataric, M., 1990. **Integration of representation into goal-driven behavior-based robots**. IEEE Transactions on Robotics and Automation 8, 3 (June 1992), 304.312.

Mellado, E.L., Canepa, H.A., 2003, **A Three-level Net Formalism for the Modeling of Multiple Mobile Robot Systems**, Proceedings of IEEE International Conference on Systems, Man and Cybernetic, Vol. 3, Washington, DC, USA, pp. 2733-2738.

Menezes, P. J. C., 1999. **Navegação de Robôs Móveis**, Dissertação de Mestrado, Universidade de Coimbra, Coimbra, Portugal.

Milutinovic, D. and Lima, P., 2002, **Petri Net Models of Robotic Tasks**, Proceedings of the 2002 IEEE: International Conference on Robotics and Automation, Vol. 4, Washington, USA, pp. 4059-4064.

Montano, L. et al., 2000, **Using the Time Petri Net Formalism for Specification, Validation and Code Generation in Robot-control Applications**, The International Journal of Robotics Research, Vol. 19, No. 1, pp. 59-76.

Murata, T., 1989. **Petri nets: Properties, analysis and applications**. Proceedings of the IEEE, v. 77, n. 4, p. 541-580.

Ogata, K., 1993, **Engenharia de Controle Moderno.** 2. Ed. São Paulo: Makron Books.

Peterson, J. L., **Petri Net Theory and the Modeling of Systems**, Prentice-Hall, N.J., 1981, ISBN 0-13-661983-5.

Philips, 2007. **I2C-bus specification and user manual**, *Document* UM10204.

Piere, E. R., 2002. **Curso de Robótica Móvel**. Apostila, UFSC – Universidade Federal de Santa Catarina.

Reisig, W., 1985. **Petri Nets, An Introduction**, EATCS, Monographs on Theoretical Computer Science, W.Brauer, G. Rozenberg, A. Salomaa (Eds.), Springer Verlag, Berlin.

Sanson, C., 1990, **Velocity and Torque Feedback Control of a Nonholonomic,** Proceedings of the International Workshop in Adaptive and Nonlinear Control: Issues in robotics, Grenoble, France.

Sava, A.T., Alla. H., 2001, **Combining Hybrid Petri Nets and Hybrid Automata**, IEEE Transactions on Robotics and Automation, Vol. 17, No. 5, pp. 670-678.

Sordalem, O., 1983. **Exponential control law for a móbile robots: Extension to path fallowing**, IEEE Transactions on Robotics and Automation, Vol. 9, No. 6, pp. 837-842.

Sordalen, O. J., 1993, **Feedback Control of Nonholonomic Mobile Robots,** Tese (Dr. ing.) – The Norwegian Institute of Tecnology.

Souza C. P. e Costa Filho, J. T., 2001, **Controle por Computador: Desenvolvendo Sistema de Aquisição de Dados para PC.** São Luis: EDUFMA.

Tell, A. R., 1995, **Nonholonomic Control Systems: from Steering to Stabilization with sinusoids,** International Journal of Control, [S.1], v.62, n.4, p. 849-870.

Thrun, S., e Bucken, A., 1996. **Integrating grid-based and topological maps for móbile robot navigation**. In Procedings of the Thirteenth National Conference on Artificial Intelligence.

Apêndice A

1. Declarações do modelo de navegação por RPC.

```
Standard declarations
  colset E = with e;
  colset INT = int;
  colset BOOL = bool;
  colset STRING = string;
  colset DATA = string;
  colset UltraSomm = int with 1..40;
  colset UltraSomg = int with 1..70;
  colset InfraRed = int with 26..40;
  colset Bussola = int with 1..4;
  colset Odometria = int with 1..99;
  colset MUltraSom = int with 1..10;
  colset MUltra = int with 1..20;
  colset MInfraRed = int with 1..10;
  colset MBussola = int with 1..4;
  colset MOdometria = int with 1..99;
  colset Ten0 = int with 0..10;
  colset Ten1 = int with 0..10;
  colset mov = int;
  colset MapUS = int with 1..10;
  colset MapIR = int with 1..10;
  colset teste = int;
declara
  var k1,k2,k3,k4,k5 :INT;
  var USl,USh:UltraSomm;
  var USp : UltraSomg;
  var IRe,IRd : InfraRed;
  var Dir : Bussola;
  var Odd,Ode : Odometria;
  var MUS:MUltraSom;
  var MUSp : MUltra;
  var MIR : MInfraRed;
  var MDir : MBussola;
  var MOd : MOdometria;
  var MpUSe, MpUSd, MpUSlate, MpUSlatd : MapUS;
  var MpIRe, MpIRd : MapIR;
  colset ROBO=product INT*INT*INT*INT*INT*INT*INT*INT;
  colset ROBO1=product INT*INT*INT*INT*INT;
  colset ROBO2=product INT*INT*INT;
  colset ROBO3=product INT*INT*DATA;
  colset ROBO4=product INT*INT*DATA*DATA;
  colset INTxDATA = product INT * DATA;
  var n,n1,n2,n3,nA,nB,nC, k,dir,esq,od: INT;
  var kod1,kod2,kod3,kod4,kod5: INT;
  var flagn,flag1,flag2,flag3: INT;
  var mov,Odm,rot,Aprox,map: INT;
  var p,pA, str,lmap: DATA;
  var s: Ten0;
  var r: Ten1;
  var u: INT;
  var valMpUSe: INT;
  var valMpUSd: INT;
  fun Ok(s:Ten0,r:Ten1) = (r<s);
  val Seg1 = 65;
  val Seg2 = 73;
  val Seg3 = 88;
  val Seg4 = 56;
  val Seg5 = 74;
  val Seg6 = 62;
  val RotaR = 90;
```