



COPPE/UFRJ

**ARQUITETURA HÍBRIDA E CONTROLE DE MISSÃO DE UM ROBÔ
MÓVEL GUIADO POR TRILHOS**

Renan Salles de Freitas

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Ramon Romankevicius

Rio de Janeiro

Maro de 2015

ARQUITETURA HÍBRIDA E CONTROLE DE MISSÃO DE UM ROBÔ
MÓVEL GUIADO POR TRILHOS

Renan Salles de Freitas

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO
ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE
ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE
JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A
OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA
ELÉTRICA.

Aprovada por:

Prof. Ramon Romankevicius, D.Sc.

Prof. , D.Sc.

Prof. , Ph.D.

Prof. , D.Sc.

RIO DE JANEIRO, RJ – BRASIL

MARO DE 2015

de Freitas, Renan Salles

Arquitetura híbrida e controle de missão de um robô móvel guiado por trilhos/Renan Salles de Freitas. – Rio de Janeiro: UFRJ/COPPE, 2015.

XII, 46 p.: il.; 29,7cm.

Orientador: Ramon Romankevicius

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2015.

Referências Bibliográficas: p. 42 – 45.

1. Controle de Missão. 2. Arquitetura robótica.
3. Máquina de estado. I. Romankevicius, Ramon.
- II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

Agradecimentos

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

ARQUITETURA HÍBRIDA E CONTROLE DE MISSÃO DE UM ROBÔ
MÓVEL GUIADO POR TRILHOS

Renan Salles de Freitas

Maro/2015

Orientador: Ramon Romankevicius

Programa: Engenharia Elétrica

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DORIS MISSION CONTROL

Renan Salles de Freitas

March/2015

Advisor: Ramon Romankevicius

Department: Electrical Engineering

Sumário

Lista de Figuras	x
Lista de Tabelas	xii
1 Introdução	1
1.1 Motivação	1
1.2 Objetivo	1
1.3 Metodologia	1
1.4 Organização da tese	1
2 Revisão Bibliográfica	2
2.1 Paradigma hierárquico/deliberativo	4
2.1.1 Robôs deliberativos	5
2.1.2 Arquiteturas deliberativas	5
2.1.3 Análise crítica	14
2.2 Paradigma reativo	16
2.2.1 Robôs reativos	19
2.2.2 Arquiteturas reativas	19
2.2.3 Análise crítica	26
2.3 Paradigma híbrido ou deliberativo/reactivo	26
2.3.1 Robôs híbridos	28
2.3.2 Arquitetura híbrida	28
2.3.3 Análise crítica	37
2.4 Robotic Development Environments	38
2.4.1 ROS	38

3 Arquitetura proposta	39
3.1 Robô DORIS	39
3.1.1 Funcionalidades	39
3.2 Arquitetura híbrida em três camadas	39
3.2.1 Camada planejador	39
3.2.2 Camada executiva	39
3.2.3 Camada funcional	39
4 Resultados e Discussões	40
4.1 Metodologia para avaliação do Método	40
4.2 Validação da rotina implementada	40
5 Conclusões	41
Referências Bibliográficas	42
A Código Fonte	46

Lista de Figuras

2.1	Arquitetura hierárquica tradicional	4
2.2	Shakey robot	6
2.3	HILARE	6
2.4	Manipulador robótico atual	7
2.5	Arquitetura de Albus para sistemas deliberativos.	9
2.6	Arquitetura NASREM.	10
2.7	Arquitetura de Saridis	12
2.8	Arquitetura do Veículo MARIUS	15
2.9	Comparativo da arquitetura deliberativa com o ser humano.	16
2.10	Projeto da google 2011- para o desenvolvimento de um carro autônomo.	16
2.11	Arquitetura para sistema de controle de robôs móveis por Brooks . .	18
2.12	A tartaruga de Grey Walter.	19
2.13	Veículos de Braitenberg.	20
2.14	Robô comercial Roomba da iRobot.	20
2.15	Camadas de controle de Brooks	22
2.16	Nível 0 e 1 de controle do sistema	23
2.17	Módulo básico da arquitetura de subsunção (AFSM).	24
2.18	Analogia de sistemas reativos com o ser humano.	27
2.19	Robô George.	28
2.20	O AUV Phoenix.	29
2.21	Robô Curiosity, o robô da NASA explorador do planeta Marte. . . .	29
2.22	Arquitetura AuRA.	33
2.23	Arquitetura do Robô Phoenix de Healey	35
2.24	Arquitetura do Robô Stanley	36
2.25	Arquitetura CLARAty	37

2.26 Analogia de sistemas reativos com o ser humano.	38
--	----

Lista de Tabelas

Capítulo 1

Introdução

1.1 Motivação

1.2 Objetivo

1.3 Metodologia

1.4 Organização da tese

Capítulo 2

Revisão Bibliográfica

A modelagem de robôs de acordo com suas principais funcionalidades e o desenvolvimento de novas arquiteturas são o âmago no estudo de controle de missão de robôs móveis. Dessa forma, arquitetura robótica e controle de missão são conceitos relacionados, e, portanto, o cerne desta pesquisa bibliográfica.

Neste capítulo, são apresentados os fundamentos teóricos necessários para o entendimento desta dissertação. O objetivo deste levantamento bibliográfico é apresentar alguns dos principais trabalhos e pesquisas científicos sobre arquiteturas e sistemas de controle de missão de robôs móveis. Por fim, o leitor é direcionado para os ambientes de software desenvolvidos para robôs autônomos (*Robotic Development Environments - RDEs*).

O conceito de arquitetura para robôs é definido de diferentes formas na literatura. Em [1], arquitetura de robô é relacionada com arquitetura de software, em uma adaptação à arquitetura de computadores de [2], e definida como: arquitetura de robô é a disciplina dedicada ao projeto de robôs altamente específicos e individuais a partir de uma coleção de blocos comuns de softwares. Em [3], a definição aborda sistemas de controle: uma arquitetura fornece uma maneira principal de organizar um sistema de controle, contudo, a arquitetura também impõe restrições sobre a forma como o problema de controle pode ser resolvido. Já em [4], o autor tenta associar a arquitetura de software com os componentes de hardware (processadores) para compor a arquitetura robótica.

O sistema robótico é composto por diversos elementos de hardware e software que são interdependentes e necessários para o funcionamento do sistema. Como o

propósito deste trabalho é o desenvolvimento de uma arquitetura robótica para o DORIS, sendo considerados os aspectos físicos, lógicos e a aplicabilidade do robô, entende-se que uma arquitetura para robô móvel descreve uma maneira de se construir o controle inteligente do robô, os módulos do sistema, como estes módulos interagem entre si e seus elementos de hardware associados, visando sua aplicação. A evolução das arquiteturas apresentadas nesta revisão bibliográfica mostra que os elementos de hardware assumem um papel de grande importância durante o desenvolvimento dessas arquiteturas, por exemplo como um fator limitador, assim como a aplicação e o meio em que o robô está inserido.

De acordo com [5], os componentes básicos de uma arquitetura para robôs são classificados em três grupos: *Percepção*, que envolve as atividades de interpretação e integração dos sensores; *Planejamento* de tarefas, sincronização, e o monitoramento da execução de todas as atividades do robô; *Atuação*, que envolve as atividades de execução dos movimentos, ações do robô e controle dos atuadores.

As três primeiras seções desta pesquisa bibliográfica abordam os paradigmas da robótica, isto é, as três arquiteturas de operação de um sistema robótico: paradigma hierárquico/deliberativo (SPA - *Sense, Plan and Act*); paradigma reativo; e paradigma híbrido deliberativo/reactivo. As seções apresentam e exemplificam as arquiteturas pela ótica de diversos autores, e são comparadas e analisadas.

O controle de missão (*Mission Control System - MCS*) ou planejamento de missão (*Mission Planning*) ou planejamento de tarefas (*Task Planning*) de robôs faz parte da arquitetura robótica, e pode ser desenvolvido para os três tipos de arquiteturas. Em [6], o conceito é bem introduzido como: controle de missão é um sistema que permite ao operador definir as missões de um veículo em linguagem de alto nível; provê ferramentas adequadas para converter planos em Progamas de Missões que podem ser verificados e executados em tempo real; e permite ao operador saber o estado da missão enquanto esta é executada, e modificá-la se for necessário. Em [7], o conceito de planejamento de missão é ampliado para múltiplos robôs: planejamento de missão é o processo de determinar o que cada robô deve fazer para alcançar, de uma maneira conjunta, os objetivos da missão, em um ambiente dinâmico.

Neste trabalho, o controle de missão de robôs é o componente da arquitetura robótica que organiza e executa todas as tarefas do robô de maneira ótima, exerce

o papel de traduzir os comandos de missão do usuário ao robô, provê feedback ao operador (*execution monitoring*), e contém as diretivas do robô. Como faz parte da arquitetura, as seções que seguem buscam, em cada arquitetura, destacar de forma exemplificada alguns controles de missão.

A robô DORIS é um robô móvel com aplicação de inspeção em um ambiente não muito dinâmico. Apresenta missões semelhantes a um AUV, apesar de sua dinâmica ser bem mais simplificada, já que este se move ao longo de um trilho. Entretanto, como a aplicação (inspeção) é igual, os hardwares são equivalentes por ser um robô móvel com vários sensores, e alguns desafios são comuns, é de se esperar que a arquitetura robótica possa apresentar muitas semelhanças. Desta forma, durante a apresentação das arquiteturas, será sempre destacada uma arquitetura em AUV.

2.1 Paradigma hierárquico/deliberativo

Em meados do século XX, são realizados os primeiros estudos de robôs autônomos, juntamente com o aparecimento da Inteligência Artificial (IA). Em um sistema robótico, a IA clássica consiste em um modelo centralizado que coleta informações usando sensores, cria um modelo do ambiente, planeja o próximo movimento e executa a ação. São sistemas do tipo *Sense, Plan and act* (SPA). Essa arquitetura de controle é clássica e tem abordagem *top-down* (hierárquica), como na decomposição tradicional, figura 2.1.

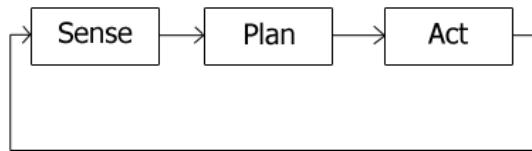


Figura 2.1: Arquitetura hierárquica tradicional.

De acordo com Marvin Minsky, uma máquina (robô) deveria tender a criar, por si só, um modelo abstrato do ambiente em que está inserido (define-se *mundo*). Caso fosse dado uma tarefa, ela primeiro poderia explorar soluções dentro de seu modelo abstrato e, então, experimentá-las externamente. Seria como realizar uma simulação

interna e, caso funcionasse, realizá-la no mundo real.

2.1.1 Robôs deliberativos

Entre 1966 e 1972, Charles Rosen e Nils Nilsson da Universidade de Stanford criaram o Shakey, primeiro robô móvel autônomo (figura 2.2). Foi desenvolvida uma inteligência artificial, *problem solver*, chamada STRIPS. Este sistema é um planejador de trajetórias que armazena as informações do ambiente (mapas e obstáculos) de maneira simbólica, e se dada uma tarefa de deslocamento (*goto*), é realizada uma busca lógica pelo sistema.

Em 1977, começou a ser desenvolvido o projeto HILARE (figura 2.3), no Laboratoire d'Automatique et d'Analyse des Systèmes (LAAS), Toulouse, France. O robô possuía sensores como câmera, ultrassons e laser para medir distância, sendo possível atualizar o seu mundo com acurácia. Seu mundo era representado por modelos geométricos e um modelo relacional que expressava a conectividade dos quartos e corredores (simbólico) [8].

Também em 1977, o Stanford Cart foi criado por Moravec para navegação e desvio de obstáculos [9]. Os obstáculos eram identificados pelo robô durante a operação e representados em seu mundo interno como esferas. O robô possuía uma segunda representação do mundo, simbólica por grafos.

Em 1969, Victor Scheinman, Universidade de Stanford [10], inventou o primeiro manipulador robótico totalmente elétrico de seis elos e com solução completa e integrada de cinemática inversa. Isto é, dado um ponto qualquer pertencente ao espaço de trabalho do manipulador, este calcula o ângulo das juntas de forma que o efetuador alcance o ponto especificado. Isso permitiu que o manipulador percorresse trajetórias arbitrárias. Até os dias atuais, 2015, é ampla a utilização de manipuladores industriais. A sofisticação destes sistemas já possibilita que estes armazenem todo o conhecimento do mundo e executem tarefas autônomas (figura 2.4).

2.1.2 Arquiteturas deliberativas

As arquiteturas deliberativas são sistemas hierárquicos com a lógica *SPA*. São utilizados em sistemas robóticos até hoje, quando a aplicação favorece seu uso e o poder

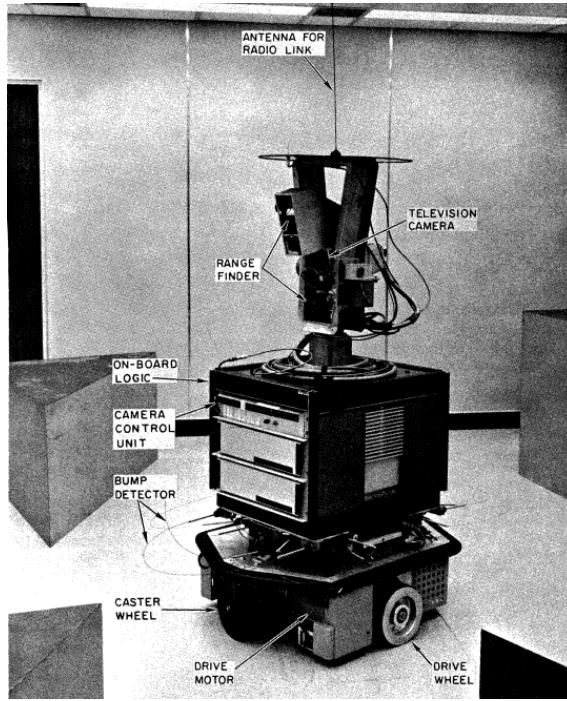


Figura 2.2: Shakey robot

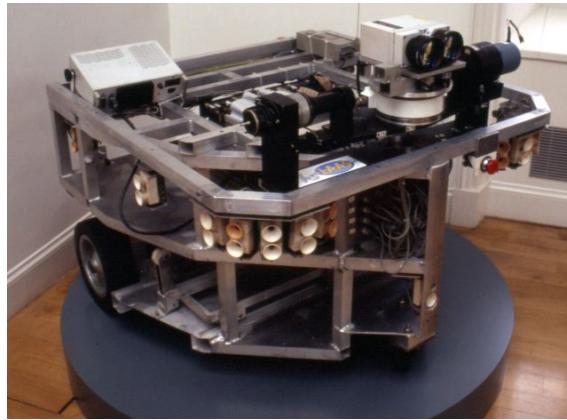


Figura 2.3: HILARE

computacional não é uma restrição. Destacam-se os modelos de Albus, NASREM, o *Intelligent Mobile Robot System*, MARIUS (AUV).

Modelo de Albus

Albus foi o pioneiro e mais influente autor de teorias em arquiteturas deliberativas, [11]. Sua grande contribuição foi a formalização e definição de diversos termos amplamente utilizados em automação e controle. Dentre outros, destaca-se o teorema de que há quatro sistemas que compõem a inteligência: processamento de sensores,



Figura 2.4: Manipulador robótico atual

modelo do mundo, geração de comportamentos e julgamento de valor. As entradas desses elementos são os sensores e suas saídas são os atuadores:

- Atuadores: as saídas de um sistema inteligente são produzidas por atuadores, como mover, posicionar braços, pernas, mãos, olhos e etc. Os atuadores naturais são os músculos e as glândulas, já os atuadores de máquinas são motores, pistões e válvulas.
- Sensores: são as entradas de um sistema inteligente, como sensores de força, torque, posição, velocidade, vibração, acústico, gases, temperatura e muitos outros. Monitoram o mundo e o estado interno do sistema, e provê dados ao sistema de processamento sensorial.
- Processamento sensorial: sistema que compara novas observações com a expectativa interna do modelo do mundo. Integra e armazena as diferenças e semelhanças encontradas, a fim de reconhecer padrões, objetos e relações no mundo.
- Modelo do mundo: é a melhor estimativa que o sistema inteligente possui do mundo, e atualizado pelo processamento sensorial. É um banco de dados com todo o conhecimento do mundo e contém uma capacidade de simulação que

gera expectativas e predições. O modelo do mundo pode prover informações do passado, presente e prevê estados futuros. Os dados são importantes para: o gerador de comportamentos escolher o plano adequado para execução das ações; o processamento sensorial fazer correlações, comparação de modelos, e reconhecimento de objetos, estados e eventos; e o sistema de julgamento de valor computar valores de custo, benefício, risco, incerteza, importância e outros.

- Julgamento de valor: este é o sistema que determina o que é bom ou ruim, importante ou trivial, certo ou improvável. Computa custos, riscos e benefícios de situações observadas e atividades planejadas.
- Gerador de comportamentos: elemento que seleciona objetivos e planos, executa e monitora ações, e modifica planos existentes quando alguma situação do mundo exigir. Tarefas são decompostas em subtarefas, e subtarefas são sequências de objetivos. A ordem lógica de funcionamento é: o gerador de comportamentos cria planos, o modelo do mundo predita o resultado do plano, e o julgamento de valor avalia os resultados. O gerador de comportamento seleciona o plano com a avaliação mais alta.

As relações entre os elementos do sistema inteligente estão representados na figura 2.5. Esses elementos e suas relações possibilitaram a criação de diversas arquiteturas.

Vale ressaltar que, nesta arquitetura, o *gerador de comportamentos* faz o papel do controle de missão, porém não de maneira completa, já que a interação com o usuário ainda é precária.

NASREM

O NASREM [12] foi uma arquitetura utilizada pela NASA e possuía uma arquitetura com seis níveis de funcionalidade (figura 2.6):

1. Servo: provê o controle dos atuadores do robô (posição, velocidade e etc).
2. Primitiva: determina as primitivas de movimento para gerar trajetórias suaves.
3. Movimento elementar: define e planeja trajetórias livre de colisões.

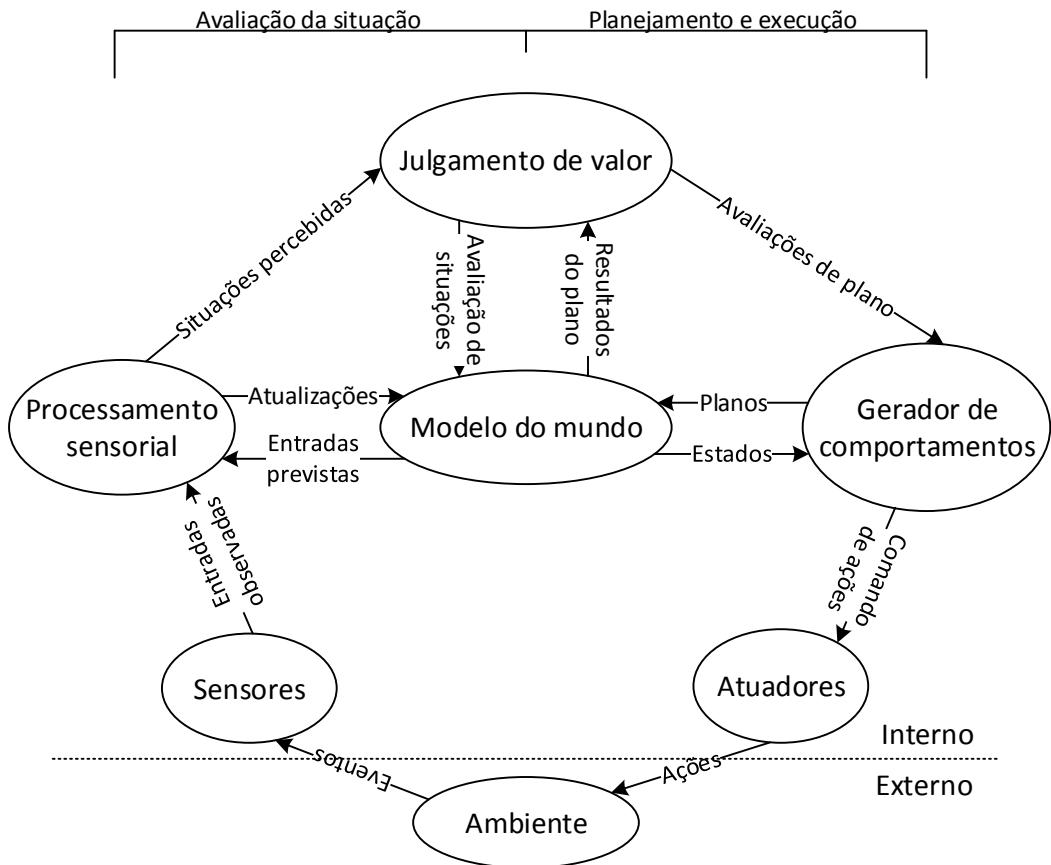


Figura 2.5: Arquitetura de Albus para sistemas deliberativos.

4. Tarefa: converte ações desejadas de um objeto em sequências de movimentos elementares.
5. Compartimento de serviços: converte ações de grupos de objetos em tarefas de um objeto.
6. Missão: decompõe o plano de missão em alto nível em compartimento de serviços.

Vale ressaltar que, no modelo NASREM, o operador tem acesso a qualquer nível hierárquico do robô e pode tomar o controle do robô para si, além de poder substituir as entradas de sensores, modelo do mundo e outros. Dessa forma, o nível de autonomia do robô pode ser desenvolvido de forma incremental.

A arquitetura hierárquica proposta em NASREM permite modularidade e propõe uma metodologia de software.

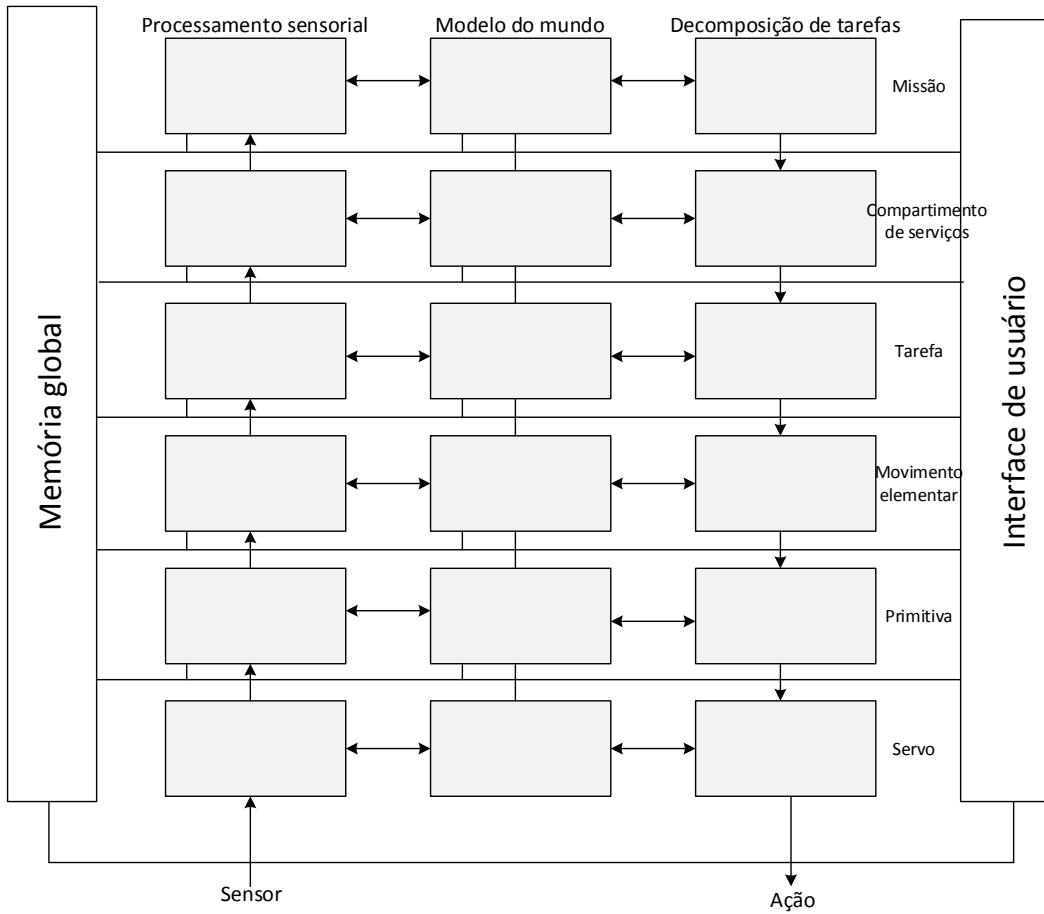


Figura 2.6: Arquitetura NASREM.

Intelligent Mobile Robot System

Em 1991, Saridis [13] cria o *Intelligent Mobile Robot System* (IMRS) baseado na teoria de inteligência hierárquica de controle [14]. Saridis utiliza redes de Petri como módulos básicos da arquitetura para traduzir os comandos gerados pelo nível de organização em algo comprehensível para o nível de execução.

O IMRS possui a seguinte arquitetura (figura 2.7):

- Nível organizacional (organizador de tarefas): gera tarefas de movimentação de alto nível.
- Nível de coordenação: funciona como uma interface entre o nível organizacional e o de execução. O nível é composto por um remetente e alguns coordenadores. O remetente recebe o plano da tarefa do organizador, decompõe a tarefa em ações de controle e remete aos coordenadores. Os coordenadores

traduzem os comandos de controle em instruções de operação e transmite ao nível de execução.

- Nível de execução: executa a instrução proveniente do nível de coordenação e reporta seus resultados a ele.

O nível de coordenação do IMRS é composto por um remetente (*Dispatcher*) e três coordenadores: sistema de visão (VS), desvio de obstáculo e controle de rastreamento (OATC), e planejamento de trajetórias (PP). Com o modelo de redes de Petri não é possível implementar o esquema de linguagem de decisão para descrever a tradução de tarefas entre remetente e coordenadores. Portanto, os *Petri Net Transducers* (PNTs) foram introduzidos como tradutores de linguagem (protocolo): $PNT = (N, \Sigma, \Delta, \sigma, \mu, F)$. Onde:

- A rede de Petri $N = (P, T, I, O)$, P lugares, T transições, função de entrada I , função de saída O , é o controle da tradução;
- μ é o estado inicial de N ;
- Σ é o alfabeto de entrada, representa tarefas de entradas;
- Δ é o alfabeto de saída, representa tarefas de saída;
- σ especifica, para uma dada tarefa de entrada, as transições em N e as subtarefas de saída que podem ser usadas na tarefa;
- F é o estado final. Indica o fim da tradução da tarefa;

Os quatro PNT's são combinados para realizarem a tradução de tarefas no Nível de Coordenação: remetente, sistema de visão, desvio de obstáculo e controle de rastreamento, e planejamento de trajetórias.

A modelagem do sistema utilizando redes de Petri proporcionaram algumas funcionalidades essenciais em uma arquitetura robótica: a capacidade de executar duas tarefas simultaneamente, por exemplo, movimentação e planejamento de trajetórias; e o *Input Semaphore*, que impede um processo de ser executado até outro ser finalizado.

Saridis salienta os benefícios das PNT:

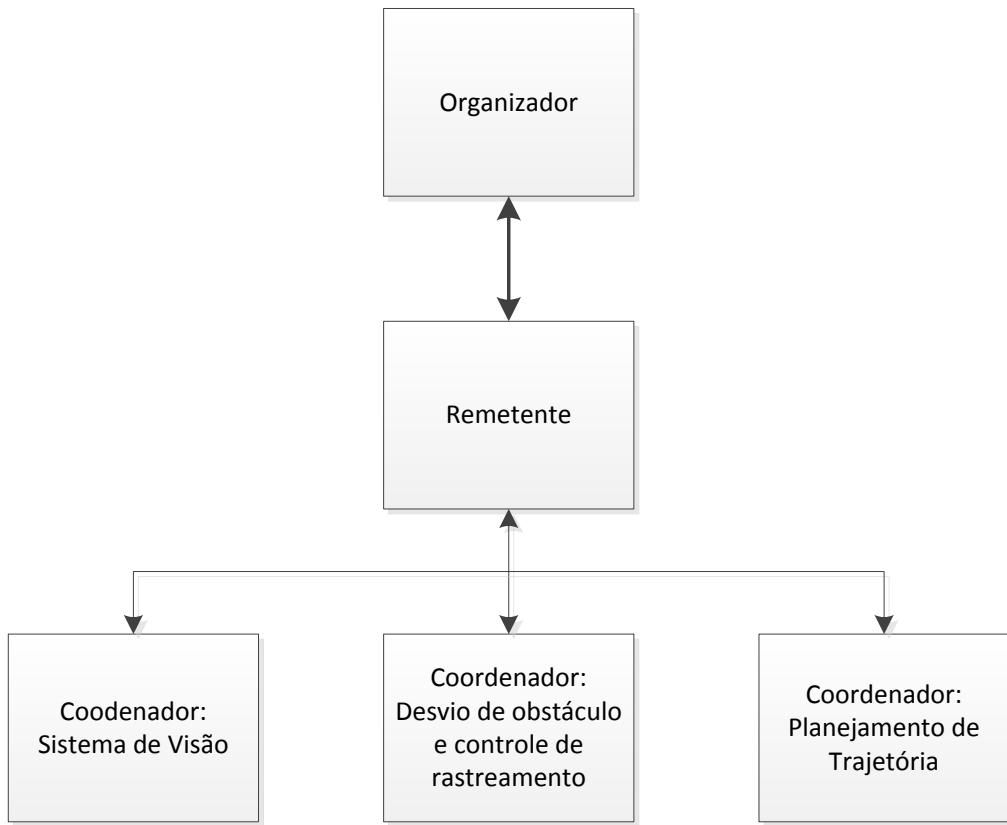


Figura 2.7: Arquitetura de Saridis

- Redes de Petri podem ser usadas como módulos básicos para sistemas de controle de missão de robôs móveis.
- A comunicação e conexão de módulos são eficientes entre redes de Petri.
- Controle e mecanismo de comunicação para coordenação de tarefas de um robô móvel podem ser realizados com redes de Petri.

A arquitetura de Saridis é uma contribuição importante por criar um nível organizacional, separando o nível do desenvolvedor de baixo nível e um nível de alto nível para um operador (usuário). Além disso, as redes de Petri assumem um importante papel como módulo básico de controle para seu sistema IMRS. As redes de Petri foram originalmente introduzidas para descrever as comunicações de máquinas de estado finito (FSM), possibilitando flexibilidade e robustez, e é provado que redes de Petri são uma excelente ferramenta para modelagem de sistemas, sobretudo quando

envolvem tarefas conflitantes ou simultâneas [15].

MARIUS

Em 1996, Silva et al. [16] (Institute for Systems and Robotics, Lisboa) projetaram, desenvolveram e testaram um sistema de controle de missão para o MARIUS, robô autônomo submarino. O trabalho de Silva introduz novos e importantes conceitos chave para o MCS: Tarefa do Sistema (*System Task*), Primitiva do Veículo (*Vechicle Primitive*), Procedimento de Missão (*Mission Procedure*) e Programa de Missão (*Mission Program*). Além disso, a arquitetura do veículo, por ser um AUV, possui sistemas e interconexões semelhantes ao robô DORIS, estudo desta dissertação.

Tarefa do Sistema (ST): é a especificação paramétrica de uma classe de algoritmos ou processos que implementam uma funcionalidade básica em um robô. Requer a implementação de dois módulos: *i*) um *módulo Funcional* que contém um determinado algoritmo e processo, e transfere dados com outras Tarefas do Sistema e dispositivos físicos; *ii*) um *módulo Comando*, máquina de estado finito, que recebe comandos externos, produz mensagens de saída, e controla a seleção de algoritmos, processos, e caminhos dos dados para/de módulos Funcionais.

A arquitetura do MARIUS é descrita abaixo, figura 2.8:

- *Vehicle Support System* (VSS) - Controla a distribuição de energia aos hardwares instalados no veículo, monitora consumo de energia e detecta falhas de hardware, podendo enviar comandos de emergência.
- *Actuator Control System* (ACS) - Controla a velocidade de rotação dos pulsadores e posição dos ailerions e lemes. Os *Set Points* dos atuadores são dados pelo *Vehicle Guidance and Control System* (VGCS) e os dados dos atuadores são transmitidos para o *Mission Control System*.
- *Navigation System* (NS) - Estima posição linear e velocidade do veículo, orientação e velocidade angular. O sistema funde informações do *Positioning System (Long Baseline unit)* e *Motion Sensor Integration System*, o qual inclui diversos sensores. As saídas do NS são entradas do VGCS, e enviadas ao MCS.

- *Vehicle Guidance and Control System* (VGCS) - Recebe como entrada as trajetórias de referência pelo MCS, e os dados de navegação do NS. Suas saídas são *Set Points* para velocidade de rotação e outros atuadores do ACS, tal que o veículo siga a trajetória desejada mesmo com incertezas e distúrbios.
- *Communication System* (COMS) - Controla o link bidirecional usado pelo operador para passar missões ao MCS, e pelo veículo para passar status de missão ou estados do veículo.
- *Environmental Inspection System* (EIS) - Coleta dados do ambiente com diversos sensores (inclusive câmeras), como temperatura, pressão, pH. É controlado pelo MCS.
- *Data Logging System* (DLS) - Adquire e armazena dados do veículo.
- *Mission Control System* (MCS) - Sequencia e sincroniza a execução das tarefas básicas do veículo para uma determinada missão e provê a recuperação em caso de falhas.

Como pode ser visto, a arquitetura é dividida em um nível organizacional, um coordenador e um nível funcional.

Silva desenvolveu os softwares CORAL e ATOL para implementação das VPs e STs em redes de Petri de forma que um usuário final, como um operador, pudesse facilmente criar seus MPs. A grande contribuição

2.1.3 Análise crítica

A abordagem deliberativa simula, de certa forma, o processo de planejamento e tomada de decisão do ser humano. Há um núcleo (cérebro) que processa todos os dados sensoriais e armazena o mundo, isto é, o ambiente em que o robô está inserido, de maneira simbólica, geométrica ou outros tipos de mapeamento. Além disso, o núcleo planeja todas as ações para uma determinada tarefa, consultando sua ideia de mundo intensivamente. Há, também, sensores que enviam suas novas informações periodicamente para o núcleo, (órgãos receptivos: visão, olfato, etc), atualizando o mundo. E há atuadores (músculos) necessários para a realização das tarefas (figura 2.9).

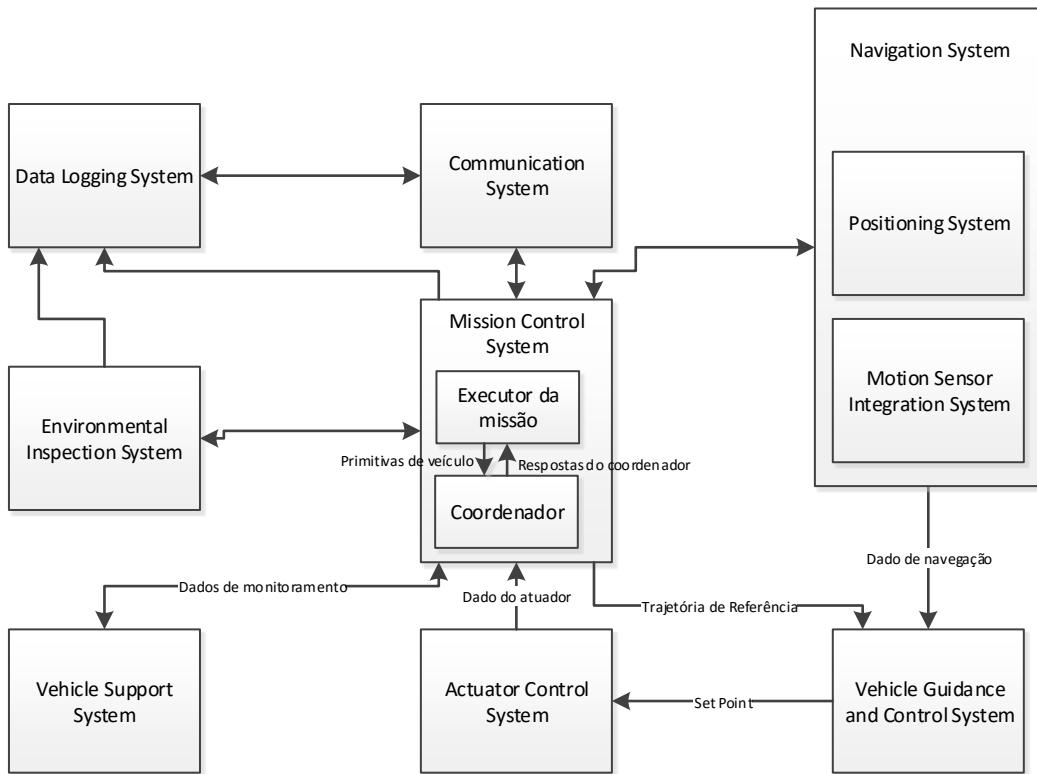


Figura 2.8: Arquitetura do Veículo MARIUS

É fácil observar que a arquitetura deliberativa é dependente do modelo de mundo armazenado e suas atualizações periódicas. Portanto, a utilização da abordagem deliberativa em ambientes extremamente dinâmicos pode ser muito custosa devido às atualizações e ao replanejamento. Além disso, é fácil observar que a arquitetura SPA dificulta a criação de sistemas em tempo real eficientes. Dessa forma, robôs móveis em ambientes muito dinâmicos, como o carro autônomo da google (figura 2.10), não são aplicações favoráveis para esta arquitetura.

O controle de missão é presente em arquiteturas deliberativas, mas de maneira primitiva, normalmente possuindo apenas as funcionalidades de: quebrar as missões em tarefas menores que o robô possa executar, gerenciar as tarefas, e monitoramento interno. Entretando, peca no desenvolvimento da interface com o usuário, a possibilidade de alteração de tarefas em tempo de execução, e feedback ao operador em tempo real.



Figura 2.9: Comparativo da arquitetura deliberativa com o ser humano.



Figura 2.10: Projeto da google 2011- para o desenvolvimento de um carro autônomo.

2.2 Paradigma reativo

Sistemas de arquitetura reativa também são chamados de sistemas baseados em comportamentos. Os robôs são programados para agir através de ativação de uma coleção de comportamentos primitivos de baixo nível. De acordo com [17], as principais características de sistemas puramente reativos são:

- Comportamentos são como elementos construtivos: são um par sensor-motor, onde o sensor provê informação necessária para o motor executar uma ação reativa, como desvio de obstáculo, atrair-se a objetivos, escapar de predadores e etc.
- Não há criação ou manutenção precisa do modelo do mundo. Os sistemas reagem ao estímulo do mundo, extremamente útil para mundos dinâmicos e hostis.

- Comportamentos de animais são normalmente utilizados para modelar esses sistemas.

Dessa forma, controle reativo é uma técnica que une percepção e ação, tipicamente no contexto de comportamentos motores, para produzir respostas robóticas em tempo real em mundos dinâmicos e não estruturados.

Em 1986, um dos primeiros estudos em sistemas reativos foi desenvolvido por Rodney Brooks [4]. Este estudo é a base para diversos trabalhos atuais que envolvem robôs reativos móveis. Os desafios de robôs autônomos apontados por Brooks e que ainda ilustram os problemas da atualidade são: *múltiplos objetivos, múltiplos sensores, robustez e extensibilidade*. De acordo com Brooks, esses desafios não são suportados pela arquitetura tradicional (paradigma hierárquico) de um sistema de controle.

Os múltiplos objetivos de robôs móveis podem:

- Ser conflitantes: por exemplo, um robô pode estar tentando alcançar um determinado ponto no espaço, porém evitando obstáculos locais.
- Ter relações de prioridade: por exemplo, um robô que inspeciona trilhos de trêm deve sair dos trilhos ao ouvir o sinal de um trêm chegando, mesmo se estiver finalizando a operação.
- Ser dependentes: objetivos de *alto nível* englobam diversos objetivos de *baixo nível*. No caso do exemplo acima, o robô que sai do trilho para evitar o trêm deve se manter equilibrado para não cair. Artigos recentes, como em [6] separam esses objetivos em *tarefas* (objetivos de *alto nível*) e *primitivas do veículo*.

Robôs são normalmente providos de múltiplos sensores e suas diversas informações podem ser redundantes, conflitantes ou complementares, podendo ser utilizadas para uma mesma tarefa do robô. Por exemplo, encoders para odometria e câmeras fixas ao robô podem ser utilizados para localização, de forma que se complementem. Os sensores podem apresentar erros ou resultados conflitantes, portanto a fusão da informação de múltiplos sensores, a determinação de seus graus de confiabilidade e em quais tarefas devem ser considerados são decisões que o robô deve saber fazer.

Um robô deve ser robusto, isto é, em caso de falha de um sensor, o robô deve se adaptar e utilizar os outros sensores que ainda funcionam para realizar as tarefas. Ou em caso de alterações no ambiente, o robô deve ser capaz de cumprir determinadas funções essenciais.

A extensibilidade constitui em acrescentar mais sensores e, portanto, aumentar a capacidade do robô, sendo possível a execução de novas tarefas. Porém, esta afirmação é normalmente criticada por alguns autores já que existe um limite imposto pelo processamento do robô, já que um novo hardware (sensor) é adicionado, mas o processamento (computador) não é substituído e sua capacidade não é aumentada.

Brooks propõe um dos primeiros sistemas baseados em comportamentos, uma arquitetura que tem como objetivo descentralizar a tomada de decisão de um modelo central, como pode ser visto na figura 2.11. O autor comenta que essa decomposição conduz a uma arquitetura radicalmente diferente para sistema de controle de robôs móveis em estratégias de implementação a nível de hardware e com grandes vantagens em robustez, desenvolvimento e teste.

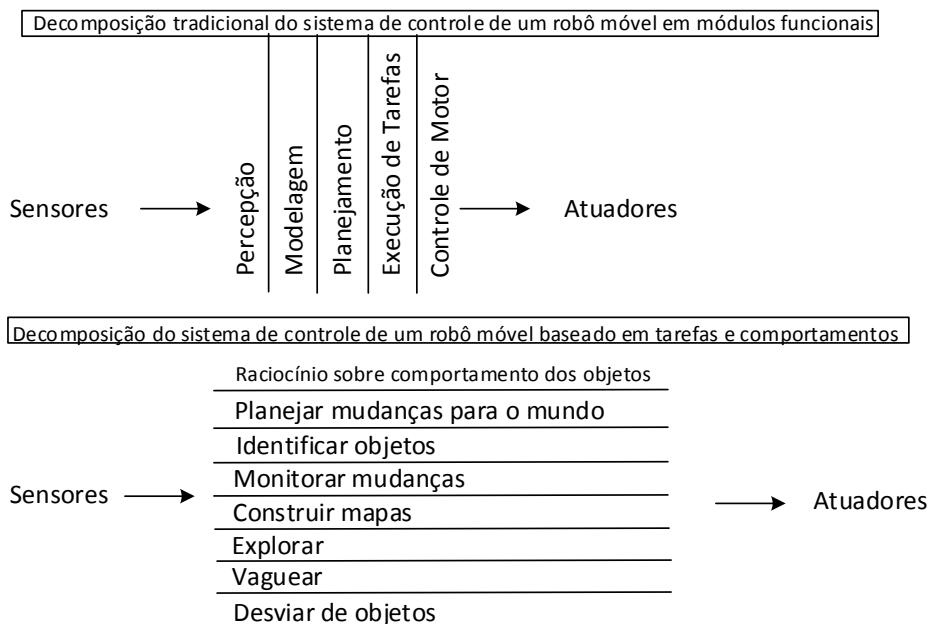


Figura 2.11: Arquitetura para sistema de controle de robôs móveis por Brooks

2.2.1 Robôs reativos

Antes mesmo de Brooks, ou seja, antes da formalização de toda a teoria em sistemas reativos, simples robôs eram criados na lógica de controle reativo e de comportamentos. Em 1953, por exemplo, Grey Walter [18] desenvolveu uma “tartaruga” elétrica capaz de se movimentar pelo ambiente, evitando luz intensa (“ameaças”) e atraída por certos objetivos. Vale observar a característica de perceber baixo nível de bateria e procurar uma estação de recarga, comportamento que se sobrepõe aos outros. Os comportamentos são simples e não há representação abstrata do mundo: dirigir-se para luz fraca; fugir de luz forte; e evitar obstáculos (figura 2.12).

Em 1984, veículos simples e puramente reativos com os pares clássicos sensor-motor foram desenvolvidos pelo psicólogo Braitenberg [19], a fim de simular sentimentos, como covardia, agressividade e outros (figura 2.13).

Em 2002 até os dias atuais, o robô iRobot Roomba ganha destaque comercial e executa uma simples tarefa doméstica: limpar o chão. Em sua arquitetura, o robô Roomba possui apenas algumas funções reativas, como esquivar-se e locomover-se, e, em suas versões antigas, foi constatado que não possui o modelo do mundo, mapa, dentro de si [20] (figura 2.14).

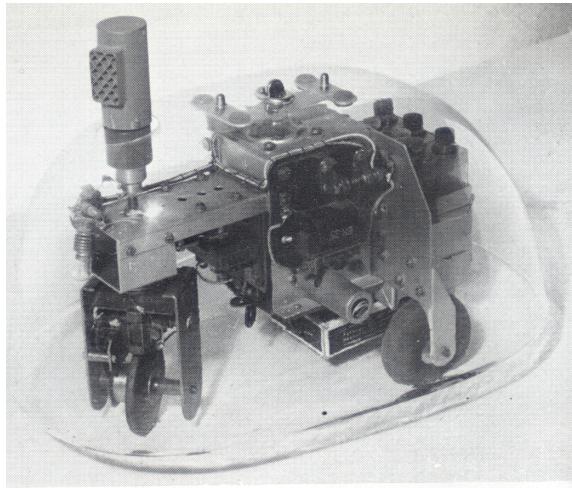


Figura 2.12: A tartaruga de Grey Walter.

2.2.2 Arquiteturas reativas

As arquiteturas reativas foram formalmente introduzidas em 1986 por Roodney Brooks. O roboticista, futuro fundador da empresa iRobot, vivenciou uma época de

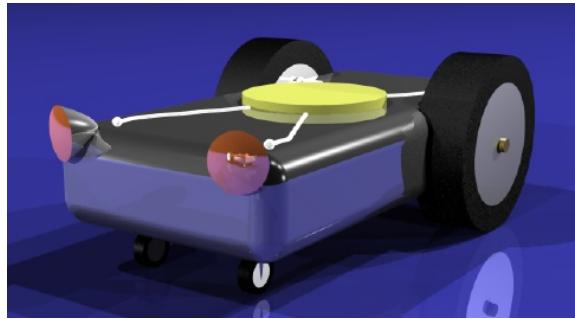


Figura 2.13: Veículos de Braitenberg.



Figura 2.14: Robô comercial Roomba da iRobot.

processadores lentos e de alto custo, dificultando o uso de diversos sensores e a armazenagem do modelo do mundo no robô. Simulações de tarefas, atualizações do mundo e análise de sensores como câmeras eram extremamente complexas para robôs e impossíveis de serem executadas em tempo real. Brooks vislumbrou como solução o processamento paralelo, evitar o uso de um modelo do mundo e criar módulos puramente reativos, pares sensor-motor, que juntos compõem o robô.

O sistema de controle de arquiteturas reativas utilizam informações locais do meio, obtidas pelos sensores do robô. Estas são simplificadamente tratadas, de forma que a ação ao estímulo é tomada rapidamente e, assim, os robôs reativos podem responder de forma mais rápida a variações do ambiente. A arquitetura define como a informação é mapeada em uma ação e como é feita a coordenação dos pares estímulo-ação (comportamentos reativos).

De acordo com Arkin [1], há duas classes predominantes para a função de coordenação: competitiva e cooperativa.

Um conflito ocorre quando dois ou mais comportamentos estão ativos ao mesmo

tempo e possuem respostas diferentes. Nesse caso, a coordenação de forma competitiva entra em ação, escolhendo um dos comportamentos para a ação final do robô. A prioridade de um comportamento sobre os demais pode ser definida explicitamente em uma hierarquia entre os comportamentos, ou pode haver uma votação por uma ação, ou outros métodos. O exemplo de sistema reativo com coordenação do tipo competitiva é o desenvolvido por Brooks, arquitetura de subsunção.

Na arquitetura cooperativa, a ação do robô é a fusão da resposta de todos os comportamentos ativos. A arquitetura reativa esquema motor de Arkin é um exemplo claro deste tipo de coordenação, onde cada comportamento influencia o movimento do robô por meio de um vetor de força artificial e a ação resultante é determinada pela soma vetoria de todos os vetores de força.

Arquitetura de subsunção

A arquitetura de subsunção é, neste trabalho, destacada, exemplificada e minuciosamente comentada, já que sua lógica será o cerne da implementação da camada reativa da DORIS.

Na figura 2.11, Brooks define *Níveis de competência*, que são classes de comportamentos desejados para o robô sobre todos os ambientes que ele pode encontrar. As classes definidas por Brooks são:

0. Evitar contato com objetos (estacionários ou móveis);
1. Vaguear sem rumo e sem bater em objetos;
2. Explorar o ambiente utilizando sensores, definir lugares alcançáveis, e seguir rumo em suas direções;
3. Construir um mini-mapa do ambiente e planejar trajetórias de um lugar para outro;
4. Observar mudanças no ambiente;
5. Raciocinar sobre o ambiente em termos de objetos identificáveis e realizar tarefas relacionadas a certos objetos;
6. Formular e executar planos que envolvam mudar o estado do ambiente como desejado;

7. Raciocinar sobre o comportamento de objetos no ambiente e modificar planos quando necessário;

Cada nível de competência inclui, como subconjunto, os níveis de competência anteriores.

Após a decomposição na nova arquitetura, Brooks define as *Camadas de Controle*, correspondentes a cada nível de competência. A ideia dessa abordagem é adicionar camadas de controle a níveis de competências superiores sem precisar alterar a camada do nível inferior. Inicia-se, portanto, com a camada de controle para o nível zero de competência, esta será testada e não mais alterada. Após, é criada a camada de nível 1, capaz de examinar os dados da camada de nível 0 e injetar dados nas interfaces internas deste nível, suprimindo seu trânsito de dados, figura 2.15.

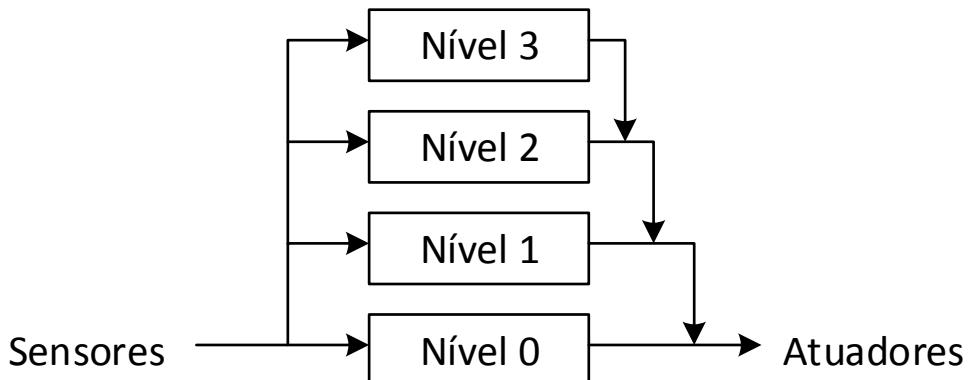


Figura 2.15: Camadas de controle de Brooks

A camada de controle nível zero deve garantir que o robô não entre em contato com outros objetos, estacionários ou móveis. Portanto, o robô deve desviar de objetos que se aproximam ou parar se houver um objeto fixo em sua trajetória. A camada de controle nível 1, combinada a camada de controle nível 0, permite que o robô vagueie sem colisões. A figura 2.16 mostra o sistema de controle aumentado pelo nível da camada 1.

A estrutura das camadas de controle foram construídas por um conjunto de pequenos processadores que enviam mensagens uns para os outros. Cada processador

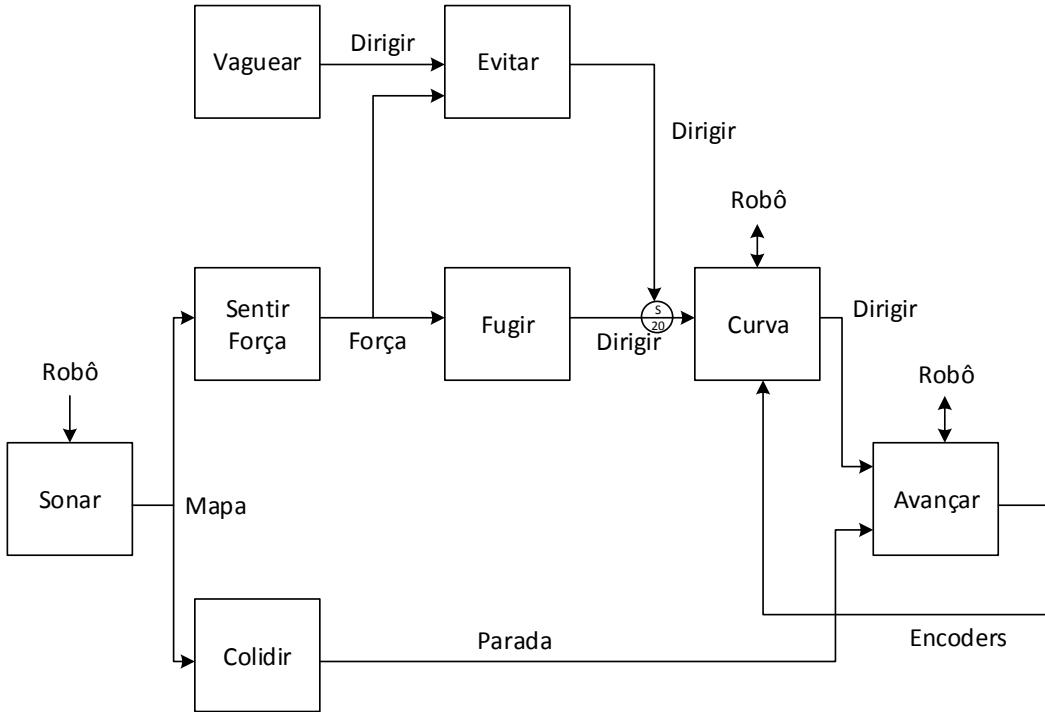


Figura 2.16: Nível 0 e 1 de controle do sistema

é uma máquina de estado finito. A nova arquitetura e essa nova estrutura de camadas com eventos discretos foram a base de diversos sistemas de controle de missão da atualidade. A fim de melhorar o entendimento desse sistema criado por Brooks, serão apresentados dois níveis de seu controle em uma aplicação de robô móvel.

A nova arquitetura de Brooks é robusta, permite interações dinâmicas, é flexível para integrar novas funcionalidades, em camadas superiores, e fácil para implementar e debugar. Brooks ainda associa sistemas de eventos discretos no controle de robôs autônomos, utilizando como módulos básicos máquinas de estados finitos (FSM - *Finite State Machine*). Como a conexão de diversas FSM's não é uma FSM, a solução encontrada por Brooks foi acrescentar inibidores e supressores em suas FSM, chamando-as AFSM (*Augmented Finite State Machine* ou máquina de estado aumentada, figura 2.17). Porém, o modelo hierárquico dos níveis cria uma certa inflexibilidade nos níveis inferiores.

Apesar dos pontos positivos, a arquitetura de controle de um sistema baseado em comportamento apresenta problemas com escala e contextualização (*situatedness*).

O problema com escala é resultado das interconexões, que podem crescer de

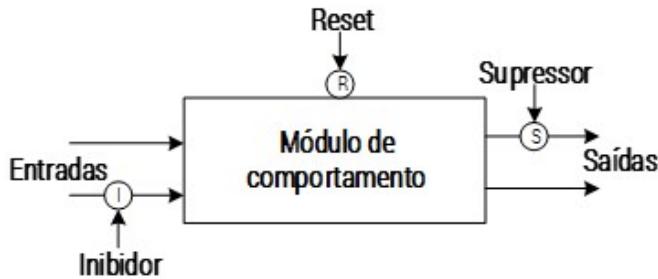


Figura 2.17: Módulo básico da arquitetura de subsunção (AFSM).

maneira factorial em relação ao número de comportamentos. A contextualização é um problema de sistemas de subsunção, onde subsistemas são incluídos em um sistema mais amplo. Como cada comportamento é uma FSM, operando concorrentemente com as outras, o comportamento corrente é o único estado no qual o veículo como um todo funcionará, apesar de todos os estados estarem sendo executados.

Apenas em 1996, Bellingham e Consi [21] propuseram um controle por camadas (*Layered Control*) a fim de resolver o problema de escala. Há um módulo de processamento de sensores que disponibiliza os dados aos comportamentos, resolvendo o problema de interconexões, já que antes as camadas superiores deveriam se conectar às inferiores para obter os dados dos sensores. Porém, Bellingham não resolveu o problema utilizando uma arquitetura do tipo subsunção: as camadas são associadas com um número de prioridade, de forma que saídas conflitantes são resolvidas com esse número. Esta solução pode gerar uma certa inflexibilidade, visto que a adição de um novo comportamento pode alterar toda a estrutura de prioridades previamente estabelecida.

A solução para o problema de contextualidade só foi resolvida em 2000 por Bennet [22]. No *State Configured Layered Control* (SCLC), múltiplos conjuntos de comportamentos simples são escolhidos de uma biblioteca de comportamentos e são executados em cada fase da missão. A vantagem é o número reduzido de comportamentos executados ao mesmo tempo. Esta é uma estratégia comum em arquiteturas de controle híbrido.

O AUV Eric Desenvolvido pelo Key Center Robotics Laboratory, University of Technology, Sydney [23], o AUV Eric segue a arquitetura de subsunção desenvolvida por Brooks com alguns avanços. Foram desenvolvidos três níveis de competência,

que descrevem a capacidade do sistema: nível de preservação, nível de exploração e nível de socialização. O nível de preservação realiza o desvio de obstáculos, a exploração executa navegação de alto nível, e a socialização habilita o comportamento de seguir objetos.

Arquitetura Esquema Motor

Em 1987, Arkin [24] utiliza a teoria de esquemas (psicologia) proposta por [25] para desenvolver sua função de coordenação cooperativa para sistemas reativos, chamado de Esquema Motor. Neste sistema, as respostas dos comportamentos aos estímulos são representadas por vetores (magnitude e orientação) e a coordenação é alcançada pela adição dos vetores, produzindo um vetor resultante. Não há hierarquia pré-definida entre comportamentos, todos os comportamentos ativos contribuem para a saída do sistema com sua resposta individual (vetor) e ganho associado. O ganho do vetor (ou peso) é um parâmetro para dar flexibilidade, possibilidade de aprendizado e adaptação do robô caso este não seja fixo.

Em uma tarefa de navegação, é fácil imaginar como funciona o método do Esquema Motor. São definidos alguns esquemas motor (comportamentos), como mover-se em direção ao objetivo, evitar obstáculos, desviar, escapar, e outros, e cada comportamento responde com um vetor que representa velocidade (magnitude) e direção (orientação) que o robô deve seguir. A soma dos vetores resulta na direção e velocidade final do robô. Ao perceber um obstáculo, por exemplo, os vetores do comportamento *desvio de obstáculos* possuem magnitude superior aos outros e orientação para fora do obstáculo (vetor de repulsão), de forma que o robô executa o desvio, em vez de colidir.

Assim como a arquitetura de subsunção, no Esquema Motor, os esquemas agem de maneira distribuída, paralela e são modulares. O sistema apresenta vantagem em relação à arquitetura de camadas por sua dinâmica, já que os esquemas podem ser instanciados e desinstaciados a qualquer momento, e fácil reconfiguração. Apesar do importante e atraente resultado em tarefas de navegação, os esquemas motores dominaram apenas esse nicho da robótica e, mesmo neste nicho, outras tarefas normalmente não são executadas com esta arquitetura.

2.2.3 Análise crítica

Se a abordagem deliberativa simulava o processo de planejamento e tomada de decisão do ser humano, os sistemas de arquitetura reativa simulam outra importante função da medula espinhal, pertencente ao nosso sistema nervoso central: o circuito reflexivo, ou arco reflexo. O reflexo é uma resposta involuntária rápida, consciente ou não, originado de um estímulo externo e realizada antes mesmo de o cérebro tomar conhecimento do estímulo periférico.

Dessa forma, não há planejamento, não há modelo de mundo, apenas uma reação ao estímulo. Esse comportamento é extremamente importante e necessário para a sobrevivência do ser humano. Por exemplo, quando encostamos a mão em uma panela quente temos a reação imediata de retirar a mão sem intervenção do cérebro, sem replanejamento, cujo processamento levaria tempo suficiente para causar danos severos. É fácil, portanto, perceber que tais comportamentos devem ser necessário também em robôs.

Robôs simples, como por exemplo um robô para limpar trilhos de trêm, ou robôs para limpar o chão, podem ter o custo reduzido e executar extraordinariamente bem suas tarefas sem a necessidade de planejamento e modelos complexos do mundo. Durante a execução de sua tarefa, um robô limpador de trilhos só precisa saber que é necessário sair do trilho ao avistar um trêm, e para isso basta um par estímulo-motor e uma supressão de tarefa (no caso, a tarefa de continuar limpando o trilho).

Não é comum na literatura encontrarmos controle de missão em arquiteturas reativas e é comum alguns autores se referirem a esses robôs como *non-taskable systems*, isto é, sistemas que não é possível atribuir tarefas. Não há interface com o usuário, mas sim com o programador, que deve alterar comportamentos a fim de gerar uma nova tarefa ou aplicação.

A tabela 2.1 mostra a comparação das arquiteturas analisadas até agora.

2.3 Paradigma híbrido ou deliberativo/reactivo

Arquiteturas deliberativas e reativas apresentam vantagens e desvantagens e que muitas vezes se opõe. Por exemplo, é muito ineficiente utilizar uma arquitetura deliberativa em um ambiente extremamente dinâmico, assim como é ineficiente uti-

Tabela 2.1: Tabela comparativa de arquitetura deliberativa e reativa

Arquiteturas deliberativas	Arquiteturas reativas
Modelo interno completo e preciso	Informações sensoriais locais
Planejamento	Ações pré-definidas às informações sensoriais
Maior flexibilidade na definição de tarefas e objetivos	Sistemas mais dedicados às tarefas e problemas específicos
Resposta lenta às mudanças no ambiente	Resposta rápida às mudanças do ambiente

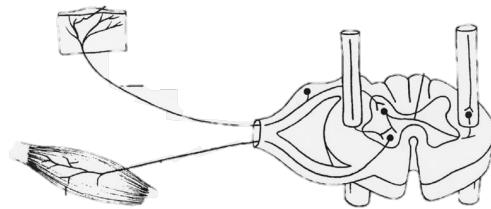


Figura 2.18: Analogia de sistemas reativos com o ser humano.

lizar arquitetura reativa em uma linha de montagem (manipuladores industriais). É natural pensar que integrando, de alguma forma, as duas arquiteturas, formando uma arquitetura híbrida, será possível absorver as vantagens de ambas.

Atualmente, os robôs com arquitetura híbrida predominam. Em muitas aplicações de robôs móveis mais complexas, fica claro que formas de conhecimento do mundo na arquitetura robótica permitem que a navegação dos robôs seja mais flexível, eficiente e geral. A arquitetura híbrida tenta combinar os métodos simbólicos da IA e seu uso de representação abstrata do modelo do mundo, mas mantém o objetivo de prover robustez, resposta em tempo real e flexibilidade dos sistemas puramente reativos. Arquiteturas híbridas podem permitir a reconfiguração de controles reativos baseado no conhecimento do mundo através da sua capacidade de raciocinar sobre os componentes comportamentais subjacentes.

O principal problema no paradigma híbrido é em como desenvolver uma metodologia unificada de arquiteturas que garantem um sistema capaz de executar planos de uma maneira robusta, como a arquitetura reativa, e, ao mesmo tempo,

ter um entendimento de alto nível da natureza do mundo e um modelo da intenção do usuário.

2.3.1 Robôs híbridos

Entre 1986 e 1987 [26], Arkin foi o primeiro a criar uma arquitetura híbrida e implementar em um robô. O objetivo do robô era reconhecer o ambiente de trabalho, planejar uma trajetória e navegar até o ponto desejado, evitando obstáculos estáteis e móveis (figura 2.19).

Em 1996, o AUV com arquitetura híbrida Phoenix, desenvolvido em Naval Post-graduated School, Monterey, foi testado em Moss Landing California [27], figura 2.20.

Desde 2003 com o Spirit até os dias atuais com Opportunity e Curiosity (figura 2.21), a NASA vem desenvolvendo robôs de alta tecnologia, com grande robustez de hardware e software.

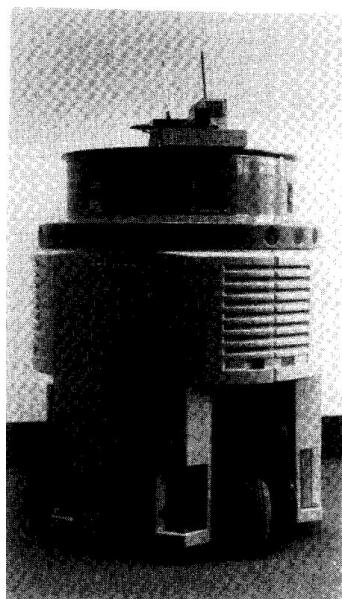


Figura 2.19: Robô George.

2.3.2 Arquitetura híbrida

Em [1], Arkin aponta quatro possíveis estratégias para o projeto de arquiteturas híbridas:



Figura 2.20: O AUV Phoenix.



Figura 2.21: Robô Curiosity, o robô da NASA explorador do planeta Marte.

- Seleção: o planejador é visto como um configurador. O planejador determina a composição de comportamentos e parâmetros usados durante a execução. O planejador pode reconfigurá-los quando necessário devido a falhas no sistema.
- Conselho: o planejador é visto como um aconselhador. O planejador sugere mudanças que o controle reativo pode ou não usar.
- Adaptação: o planejador é visto como um adaptador. O planejador continuamente altera os componentes reativos ativos de acordo com as mudanças nas condições do mundo e requisitos de tarefas.
- Adiamento: o planejador é visto como um recurso para ser usado em último caso. Neste caso, os planos só são elaborados quando necessários.

Em [28], Murphy afirma que apesar de arquiteturas híbridas variarem em como a função deliberativa (o planejador) atua no sistema, os componentes implementados em uma arquitetura híbrida são semelhantes, possuindo geralmente os seguintes módulos:

- Sequenciador: agente que gera o conjunto de comportamentos a serem usados para completar uma subtarefa, e determina qualquer sequência e condições

de ativação. A sequência é normalmente representada como uma rede de dependências ou FSMs, mas o sequenciador gera essa estrutura ou a dinamicamente a adapta.

- Gerenciador de recursos: agente que aloca recursos aos comportamentos. Este agente está diretamente ligado aos sensores do robô, podendo verificar qual o sensor é mais adequado para cada situação do robô.
- Cartógrafo: agente responsável por criar, armazenar e atualizar o mapa, modelo do mundo.
- Planejador de missão: agente que interage com o usuário, traduz a mensagem do usuário para os termos do robô, e contrói um plano de missão. Por exemplo, um robô “assistente” poderia receber o comando “Robô, traga um xícara com café”. O planejador de missão interpreta o comando de forma que primeiro o robô deve procurar uma xícara, servir café (supondo que este já esteja pronto) e, então, levá-lo até o usuário.
- Monitor de desempenho e solucionador de problemas: é o agente que permite ao robô perceber se está progredindo no cumprimento de sua tarefa.

Neste trabalho, são destacadas arquitetura híbridas Autonomous Robot Architecture (AuRA), AUV Phoenix, CLARAty e Stanley.

AuRA

A Autonomous Robot Architecture (AuRA) foi a primeira arquitetura híbrida, desenvolvida em 1986 por Arkin [24]. De acordo com Murphy [28], AuRA pertence ao estilo de arquitetura híbrida *administradora* (*Managerial*), reconhecida por sua decomposição similar a um gerenciamento de negócios. Há agentes superiores que realizam planejamento em alto nível e passam o plano a subordinados, os quais refinam os planos e coletam os recursos, e, então, estes são passados ao último nível de agentes, os comportamentos reativos. De acordo com Arkin, AuRA é um projeto de arquitetura com estratégia de Seleção.

AuRA é composto por cinco subsistemas, dos quais dois pertencem à porção deliberativa: Planejador e Cartógrafo, o subsistema de sensores, subsistema de geren-

ciador de comportamentos (gerenciador esquema motor) e o subsistema de controle homeostático.

O planejador é responsável pela missão e o planejamento de tarefas e é subdividido em três componentes: planejador da missão, navegador e piloto e seus módulos são executados sequencialmente, tornando-se mais específico e detalhado. A lógica é hierárquica: o planejador de missão envia trechos da missão ao navegador, o qual envia trechos de trajetória para o piloto, que determina ações ao controlador de baixo nível. O planejador de missão também funciona como interface ao usuário. A utilização do mapa interno do robô por cada módulo é diferente, enquanto o planejador usa o mapa global, o piloto recebe informações locais. Vale observar que, quando o modelo do mundo é atualizado, muitas vezes não há necessidade de o planejador atualizar toda a missão e recomeçar o ciclo de planejamento, o piloto pode recalcular a trajetória local. Por exemplo, o navegador recebe as tarefas: “vá até a montanha até a torre da água, siga o caminho até o acampamento”, e o piloto recebe a primeira subtarefa “vá até a montanha ae a torre da água” e as informações necessárias para gerar os comportamentos.

O Cartógrafo encapsula todo o mapa e tem a funcionalidade de receber mapas a priori (operador pode fornecer um mapa inicial). Os três componentes do planejador interagem com o Cartógrafo para obter a trajetória a seguir, quebrada em segmentos.

O subsistema do AuRA que gerencia comportamentos utiliza a arquitetura reativa esquema motor. O esquema motor, como visto na subseção 2.2.2, representa cada ação em campos potenciais (vetores) e a resposta do sistema é a soma dos vetores.

O controle homeostático, quinto subsistema, está no meio do caminho entre os sistemas deliberativos e reativos. Sua função é modificar a relação entre comportamentos, modificando os ganhos do vetores (subseção 2.2.2). Considere o seguinte exemplo, para melhor entendimento deste subsistema: um veículo (robô) operando, no planeta Marte em um ambiente rochoso, têm como tarefa remover fisicamente amostras de rochas de várias localidades pelo planeta e levá-las a um veículo de retorno, o qual tem uma data fixa de lançamento. O robô é provido com ganhos padrões em seus comportamentos, os quais produzem uma execução conservadora da operação, por exemplo garantindo que o robô esteja sempre 2 metros de distância

dos obstáculos em seu caminho. No começo da missão, a execução conservadora parece razoável, mas agora considere próximo à data limite de decolagem do veículo de retorno: se o robô estiver próximo ao veículo de decolagem, ele deveria cortar caminhos, reduzindo sua margem de evitar obstáculos para realizar a entrega, sacrificando sua própria existência para a missão.

AuRA poderia integrar o controle homeostático à parte deliberativa da arquitetura, mas foi motivado pela biologia a colocá-lo em outra área, o qual chama de subconsciente, o qual em animais modifica sempre a área consciente em resposta a necessidades internas. A figura 2.22 mostra os cinco subsistemas da arquitetura AuRA. A tabela 2.2 resume a arquitetura.

Tabela 2.2: Resumo da arquitetura AuRA

Sequenciador	Navegador, Piloto
Gerenciador de recursos	Gerenciador esquema Motor
Cartógrafo	Cartógrafo
Planejador de missão	Planejador de Missão
Monitor de desempenho	Piloto, Navegador, Planejador de missão

Arquitetura de três camadas

De acordo com Murphy, esta arquitetura pertence ao estilo Estado-hierárquico por ser organizado em três camadas baseadas no estado de conhecimento: Planejador, Executivo e Funcional.

O planejador é uma camada composta pelo planejador de missão e o cartógrafo, gerando objetivos e planos estratégicos. Esses objetivos são repassados para a camada intermediária, chamada Executivo.

A camada Executivo utiliza técnicas de planejamento reativo para selecionar um conjunto de comportamentos de uma biblioteca, e desenvolve uma rede de tarefa especificando a sequência de execução para os comportamentos a um subobjetivo particular. De acordo com Murphy, o Executor seria a camada responsável por sequenciar e monitorar o desempenho de uma arquitetura híbrida.

Os comportamentos selecionados para uma tarefa pelo Executor formam a camada Funcional. A tabela 2.3 resume os componentes desta arquitetura híbrida.

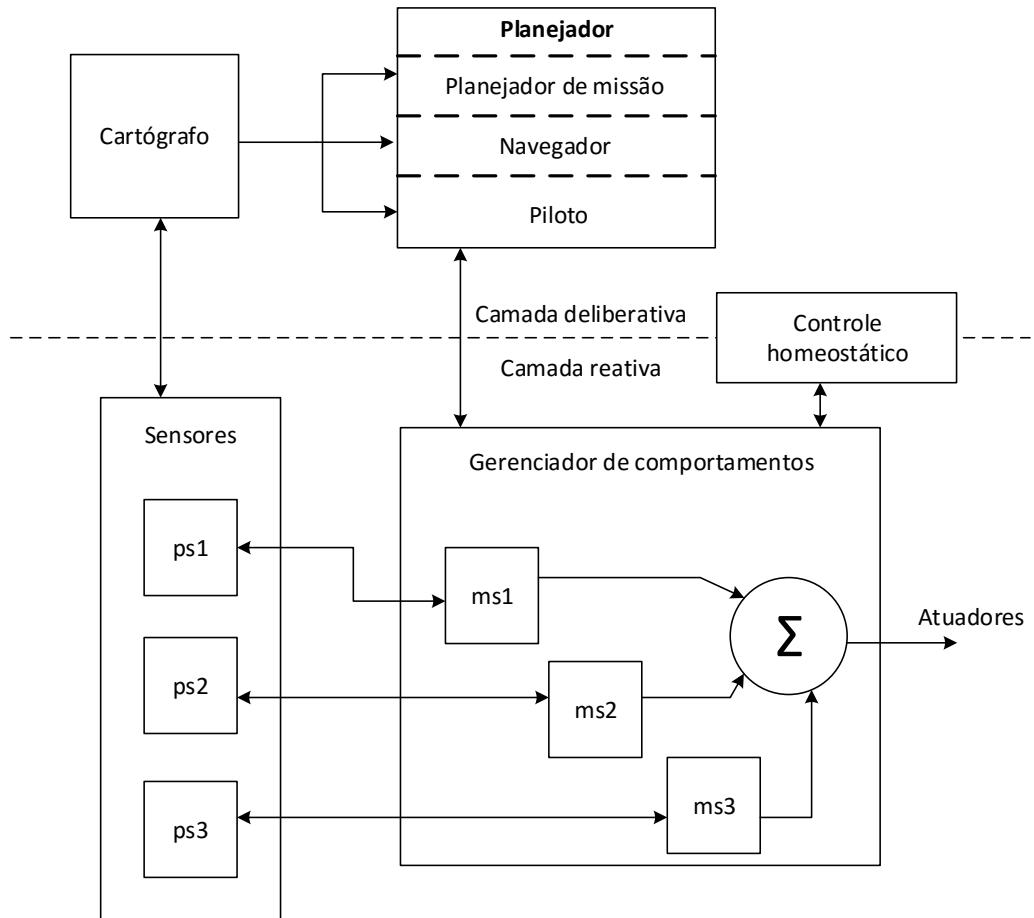


Figura 2.22: Arquitetura AuRA.

A arquitetura do AUV Phoenix foi destacada por ser um AUV e, portanto, ser uma aplicação semelhante ao robô em destaque desta dissertação. São famosas as arquiteturas Atlantis [29], sua evolução CLARAty [30], ambas desenvolvidas pela NASA, e o robô Stanley (Stanford).

Tabela 2.3: Resumo da arquitetura de três camadas

Sequenciador	Executor
Gerenciador de recursos	Executor
Cartógrafo	Planejador
Planejador de missão	Planejador
Monitor de desempenho	Planejador

AUV Phoenix Em 1996, Healey [31], California, aplica técnicas de controle híbrido em seu trabalho de desenvolvimento do AUV Phoenix. Haley propõe uma arquitetura de software híbrida com três níveis organizacionais e hierárquicos. Há um aumento de inteligência entre as camadas, do reflexivo, para o procedural, para o deliberativo figura 2.23:

- **Estratégico (planejador)**: utiliza Prolog como linguagem de controle de missão. Desenvolve os comandos que levam o veículo a executar determinada missão.
- **Tático (executivo)**: funções na linguagem C que faz interface com os predicados de Prolog e retorna variáveis booleanas (*true, false*). Este nível funciona de maneira assíncrona e retém os dados da missão, além de se comunicar com o nível de execução.
- **Funcional**: controlador em tempo real do veículo, usando mensagens assíncronas. Opera os atuadores do veículo.

O controle híbrido será responsável tanto pela movimentação do veículo, contínuo e síncrono, quanto pela sequência lógica das fases das missões, eventos discreto com transições assíncronas. A arquitetura incorpora detecção de erros e procedimentos de recuperação. O módulo reativo é desenvolvido pela arquitetura de subsunção.

Stanley Em 2006, Sebastian Thrun [32] ganhou o desafio DARPA, uma competição de veículos autônomos organizada pelo governo dos Estados Unidos a fim de promover o desenvolvimento na área de direção autônoma.

O robô Stanley foi desenvolvido pela Stanford University. É um veículo (Volkswagen Touareg R5) Fora de estrada (*Off Road*) autônomo de alta velocidade, que ganhou o desafio de se locomover pela acidentada região do deserto Mojave por mais 132 milhas sem a interferência de um ser humano. Em grande parte, o desafio DARPA é uma competição de software, pois um motorista equipado com um carro apropriado consegue atravessar o deserto sem muitas dificuldades.

O software desenvolvido é capaz de adquirir dados de sensores, construir modelos do mundo e tomar decisões de direção a uma velocidade de 60 km/h. Neste robô, foi utilizada a arquitetura híbrida de três camadas, como em Atlantis (figura 2.24)

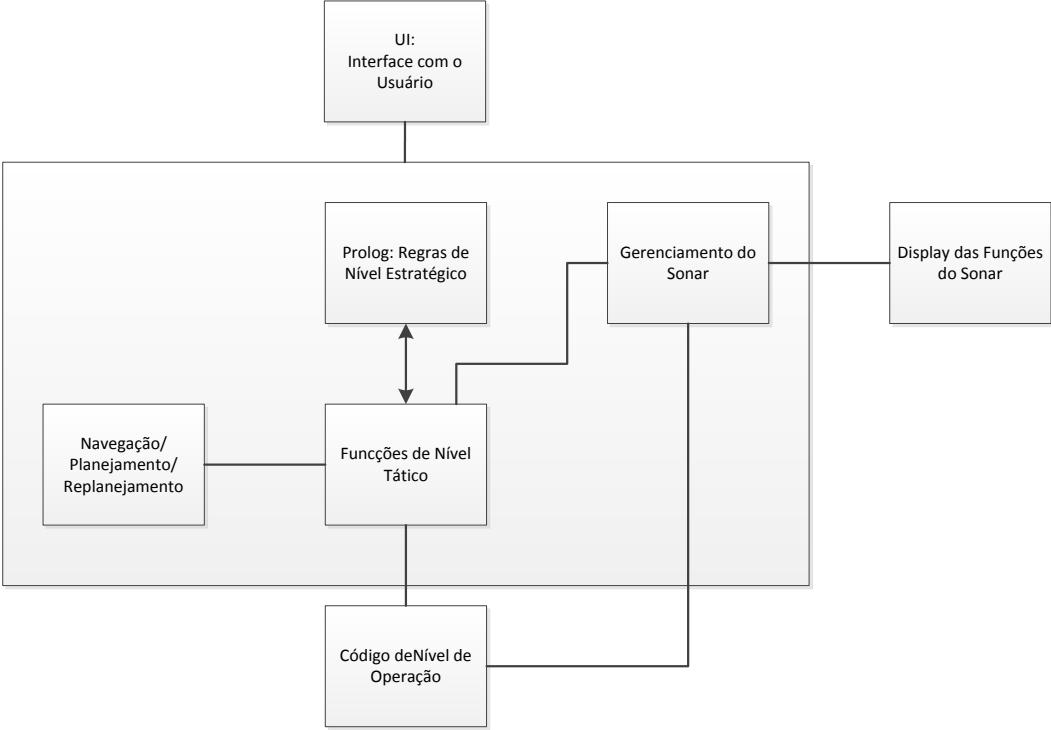


Figura 2.23: Arquitetura do Robô Phoenix de Healey

CLARAty Em 2001, Volpe [30] desenvolve uma arquitetura híbrida de duas camadas: Coupled Layer Autonomous Robot Architecture (CLARAty), como uma evolução da arquitetura Atlantis, arquitetura de três camadas. De acordo com Volpe, a arquitetura de três camadas apresenta algumas desvantagens:

- As responsabilidades e tamanho de cada nível é subjetivo ao criador da arquitetura. Portanto, há arquiteturas em que a camada funcional é dominante, outras em que a camada de planejamento domina.
- O acesso entre a mais alta camada da hierarquia (planejador) à camada de nível inferior (funcional) é restrita. Apesar de isso ser desejado durante execução, isto separa o planejador de informações da funcionalidade do sistema durante o planejamento. Uma consequência é que planejadores normalmente carregam seus modelos do sistema, que não são derivados (sentido de classe derivada, em programação como C++) diretamente do nível Funcional. Esta repetição de modelos pode gerar certas inconsistências.
- Cada camada pode ter sua própria hierarquia com granularidade variada. A

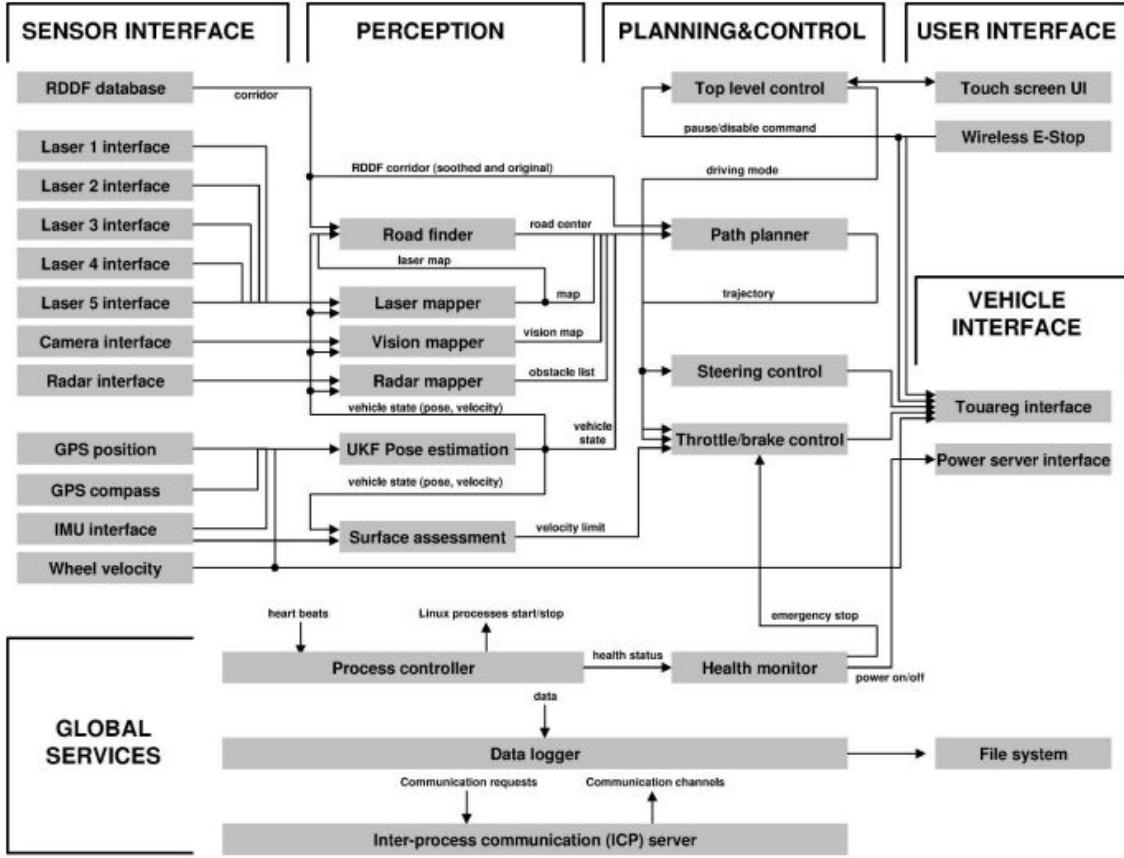


Figura 2.24: Arquitetura do Robô Stanley

camada funcional é composta por vários subsistemas aninhados, o executivo tem árvores lógicas para coordená-los, e o planejador tem diversas linhas de tempo e horizontes.

De acordo com Volper, a estrutura CLARAty tem duas vantagens principais: representação explícita da granularidade das camadas (em uma representação 3D); e a mistura das técnicas declarativas e processuais para a tomada de decisões (figura 2.25).

A camada funcional é uma interface com todo o sistema de hardware e suas capacidades. É um software orientado a objeto, obtendo assim modularidade de hardware, e estruturação apropriada de software para usar as propriedades de herança, característica extremamente importante para o nível funcional.

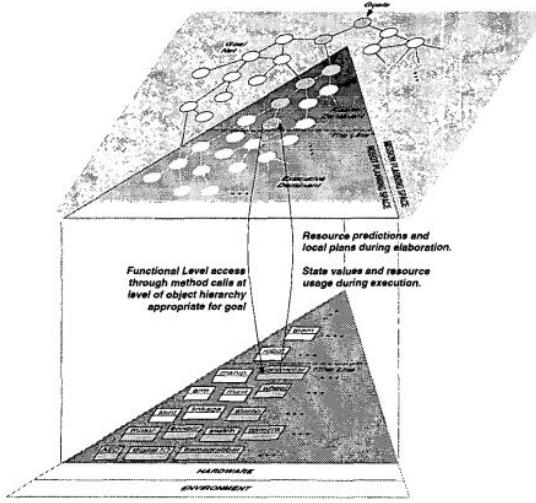


Figura 2.25: Arquitetura CLARAty

2.3.3 Análise crítica

Continuando a analogia de robôs com o funcionamento de animais e, especificamente o ser humano, a arquitetura híbrida visa juntar as duas abordagens anteriores, deliberativa e reativa, e de forma inteligente aproveitar as vantagens de ambas as arquiteturas e aproximar cada vez mais o funcionamento das máquinas com o homem. Nesta arquitetura final, está presente tanto o planejamento exercido pelo cérebro, quanto a camada reflexiva da medula, buscando tornar o robô uma entidade completa.

A grande dificuldade está em como será realizada essa fusão de arquiteturas, qual será a maneira ótima e que garanta essas vantagens. A abordagem híbrida evolui a cada ano e novos sistemas, como robôs aeroespaciais e carros com direção autônoma, provam que esta arquitetura é eficiente.

A arquitetura híbrida é modular já que grande maioria é dividida em camadas, as quais são subdivididas em módulos. O escopo de aplicação é enorme se comparada às outras arquiteturas, como já foi apontado: veículo aeroespacial, carros autônomos, AUV e outras, e como é possível partitionar a arquitetura, pode-se utilizar apenas a área deliberativa ou a reativa e englobar o escopo das duas arquiteturas anteriores. Além disso, a arquitetura híbrida provê robustez, já que apresenta monitor de desempenho e é capaz de adaptação e reconfiguração.

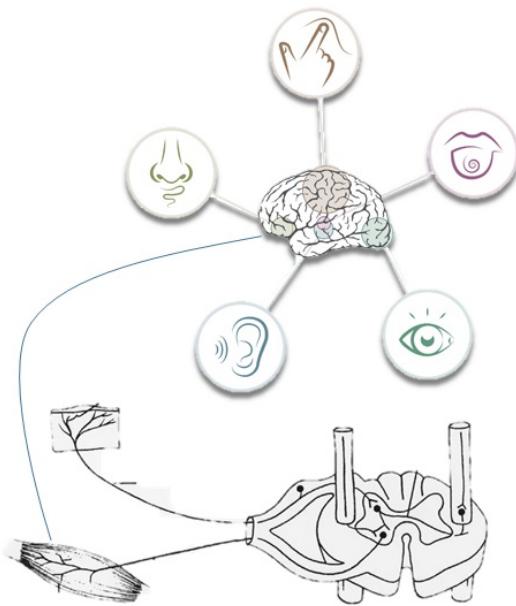


Figura 2.26: Analogia de sistemas reativos com o ser humano.

2.4 Robotic Development Environments

2.4.1 ROS

Capítulo 3

Arquitetura proposta

3.1 Robô DORIS

3.1.1 Funcionalidades

3.2 Arquitetura híbrida em três camadas

3.2.1 Camada planejador

3.2.2 Camada executiva

3.2.3 Camada funcional

Capítulo 4

Resultados e Discussões

4.1 Metodologia para avaliação do Método

4.2 Validação da rotina implementada

Capítulo 5

Conclusões

Referências Bibliográficas

- [1] ARKIN, R. C., *Behavior-based robotics*. MIT press, 1998.
- [2] STONE, H. S., *Introduction to computer architecture*. Sra, 1980.
- [3] MATARIC, M. J., “Behavior-based control: Main properties and implications”. In: *Proceedings, IEEE International Conference on Robotics and Automation, Workshop on Architectures for Intelligent Control Systems*, pp. 46–54, 1992.
- [4] BROOKS, R. A., “A robust layered control system for a mobile robot”, *Robotics and Automation, IEEE Journal of*, v. 2, n. 1, pp. 14–23, 1986.
- [5] SIEGWART, R., NOURBAKHSH, I. R., “Autonomous mobile robots”, *Massachusetts Institute of Technology*, 2004.
- [6] FRYXELL, D., OLIVEIRA, P., PASCOAL, A., et al., “Navigation, guidance and control of AUVs: an application to the MARIUS vehicle”, *Control Engineering Practice*, v. 4, n. 3, pp. 401–409, 1996.
- [7] BRUMITT, B. L., STENTZ, A., “Dynamic mission planning for multiple mobile robots”. In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, v. 3, pp. 2396–2401, 1996.
- [8] NORELIS, F., CHATILA, R. G., “Control of mobile robot actions”. In: *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*, pp. 701–707, 1989.
- [9] MORAVEC, H. P., “TOWARDS AUTOMATIC VISUAL OBSTACLE AVOIDANCE”. In: *International Conference on Artificial Intelligence (5th: 1977: Massachusetts Institute of Technology)*, 1977.

- [10] SCHEINMAN, V. D., *Design of a computer controlled manipulator.*, Tech. rep., DTIC Document, 1969.
- [11] ALBUS, J. S., “Outline for a theory of intelligence”, *Systems, Man and Cybernetics, IEEE Transactions on*, v. 21, n. 3, pp. 473–509, 1991.
- [12] ALBUS, J. S., MCCAIN, H. G., LUMIA, R., *NASA/NBS standard reference model for telerobot control system architecture (NASREM)*. National Institute of Standards and Technology Gaithersburg, MD, 1989.
- [13] WANG, F. Y., KYRIAKOPOULOS, K. J., TSOLKAS, A., et al., “A Petri-net coordination model for an intelligent mobile robot”, *Systems, Man and Cybernetics, IEEE Transactions on*, v. 21, n. 4, pp. 777–789, 1991.
- [14] SARIDIS, G. N., VALAVANIS, K. P., “Analytical design of intelligent machines”, *Automatica*, v. 24, n. 2, pp. 123–133, 1988.
- [15] MURATA, T., “Petri nets: Properties, analysis and applications”, *Proceedings of the IEEE*, v. 77, n. 4, pp. 541–580, 1989.
- [16] OLIVEIRA, P., PASCOAL, A., SILVA, V., et al., “Design, development, and testing at sea of the mission control system for the MARIUS autonomous underwater vehicle”, *Oceans MTS/IEEE*, 1996.
- [17] ARKIN, R., “Reactive Robotic Systems Ronald C. Arkin College of Computing Georgia Institute of Technology Atlanta, Georgia”, 1995.
- [18] HOLLAND, O., “Grey Walter: the pioneer of real artificial life”. In: *Proc. 5th Int. Workshop Synthesis Simulation Living Syst*, pp. 34–41, 1997.
- [19] BRAITENBERG, V., *Vehicles: Experiments in synthetic psychology*. MIT press, 1986.
- [20] TRIBELHORN, B., DODDS, Z., “Evaluating the Roomba: A low-cost, ubiquitous platform for robotics research and education”. In: *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1393–1399, 2007.

- [21] BELLINGHAM, J., GOODEY, C., CONSI, T., et al., “A second generation survey AUV”. In: *Autonomous Underwater Vehicle Technology, 1994. AUV'94., Proceedings of the 1994 Symposium on*, pp. 148–155, 1994.
- [22] BENNETT, A., LEONARD, J. J., OTHERS, “A behavior-based approach to adaptive feature detection and following with autonomous underwater vehicles”, *Oceanic Engineering, IEEE Journal of*, v. 25, n. 2, pp. 213–226, 2000.
- [23] BOSWELL, A., LEANEY, J., “Using the subsumption architecture in an autonomous underwater robot: Expostulations, extensions and experiences”, *Proceedings of the IARP 2nd Workshop on: Mobile Robots for Subsea Environments*, pp. 95–106, 1994.
- [24] ARKIN, R. C., RISEMAN, E. M., HANSON, A. R., “AuRA: An architecture for vision-based robot navigation”. In: *proceedings of the DARPA Image Understanding Workshop*, pp. 417–431, 1987.
- [25] ARBIB, M. A., “Schema theory”, *The Encyclopedia of Artificial Intelligence*, v. 2, pp. 1427–1443, 1992.
- [26] ARKIN, R. C., MURPHY, R., PEARSON, M., et al., “Mobile robot docking operations in a manufacturing environment: Progress in visual perceptual strategies”. In: *Proc. IEEE International Workshop on Intelligent Robots and Systems*, v. 89, pp. 147–154, 1989.
- [27] BRUTZMAN, D., BURNS, M., CAMPBELL, M., et al., “NPS Phoenix AUV software integration and in-water testing”. In: *Autonomous Underwater Vehicle Technology, 1996. AUV'96., Proceedings of the 1996 Symposium on*, pp. 99–108, 1996.
- [28] MURPHY, R., *Introduction to AI robotics*. MIT press, 2000.
- [29] GAT, E., “Reliable goal-directed reactive control of autonomous mobile robots”, 1991.

- [30] VOLPE, R., NESNAS, I., ESTLIN, T., et al., “The CLARAty architecture for robotic autonomy”. In: *Aerospace Conference, 2001, IEEE Proceedings.*, v. 1, pp. 1–121, 2001.
- [31] HEALEY, A., MARCO, D., MCGHEE, R. B., “Autonomous underwater vehicle control coordination using a tri-level hybrid software architecture”. In: *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, v. 3, pp. 2149–2159, 1996.
- [32] MONTEMERLO, M., THRUN, S., DAHLKAMP, H., et al., “Winning the DARPA Grand Challenge with an AI robot”. In: *Proceedings of the national conference on artificial intelligence*, v. 21, n. 1, p. 982, 2006.
- [33] MARTIN, S. C., WHITCOMB, L. L., YOERGER, D., et al., “A mission controller for high level control of autonomous and semi-autonomous underwater vehicles”. In: *OCEANS 2006*, pp. 1–6, 2006.
- [34] ARNOLDI, W. E., “The principle of minimized iteration in the solution of the matrix eigenvalue problem”, *Quart. Appl. Math.*, v. 9, pp. 17–29, 1951.
- [35] LANCZOS, C., “An iteration method for the solution of the eigenvalue problem of linear differential and integral operators”, *J. Research Nat. Bur. Standards*, v. 45, pp. 255–282, 1950.
- [36] ELFES, A., TALUKDAR, S. N., *A Distributed Control System for the CMU Rover.*, Tech. rep., DTIC Document, 1983.

Apêndice A

Código Fonte