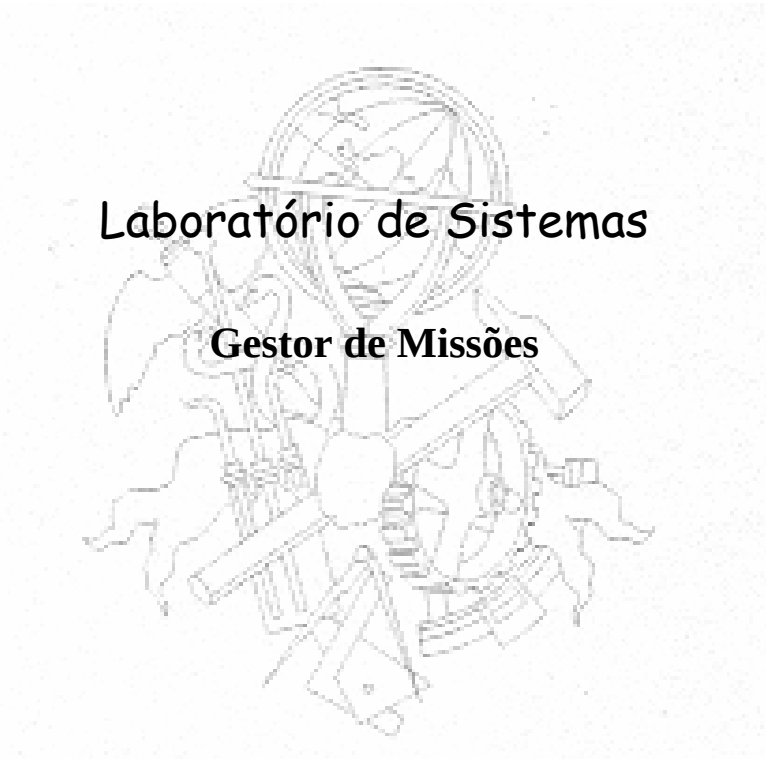
Instituto Politécnico do Porto
Instituto Superior de Engenharia
Departamento de Engenharia Electrotécnica

Engenharia Electrotécnica - Electrónica & Computadores



Laboratório de Sistemas

Gestor de Missões

José Eduardo Barbosa Leite Nunes nº1010615
Ricardo Filipe Moreira Guimarães nº1010896

Índice

Índice de Imagens

1. Introdução

A robótica móvel é uma área de pesquisa que lida com o controle de veículos autónomos ou semi-autónomos. O que diferencia a robótica móvel de outras áreas de pesquisa em robótica tais como a robótica de manipuladores, é a sua ênfase nos problemas relacionados com a operação (locomoção) em ambientes complexos de larga escala, que se modificam dinamicamente, compostos tanto de obstáculos estáticos como de obstáculos móveis. Para operar neste tipo de ambiente o robô deve ser capaz de adquirir e utilizar conhecimento sobre o ambiente, estimar uma posição dentro deste ambiente, possuir a habilidade de reconhecer obstáculos, e responder em tempo real as situações que possam ocorrer neste ambiente. Além disso, todas estas funcionalidades devem operar em conjunto. As tarefas de perceber o ambiente, se localizar no ambiente, e se mover pelo ambiente são problemas fundamentais no estudo dos robôs móveis autónomos.

O desenvolvimento de sistemas de controlo para robôs móveis autónomos tem se mostrado um grande desafio para a Inteligência Artificial até os dias actuais. Diferentes abordagens para o projecto de sistema de controlo para robôs móveis autónomos vêm sendo utilizadas em diversas áreas de pesquisa. Por muitos anos os pesquisadores de Inteligência Artificial tem construído sistemas de controlo que apresentam um comportamento inteligente, mas normalmente não no mundo real e somente em ambientes controlados. Alguns pesquisadores desenvolveram certos sistemas de controlo para serem utilizados no mundo real, mas geralmente estes sistemas são limitados e não apresentam um comportamento autónomo ou inteligente. Existem diversas aplicações possíveis para os robôs móveis. No transporte, vigilância, inspecção, limpeza de casas, exploração espacial, auxílio a deficientes físicos, entre outros. No entanto, os robôs móveis autónomos ainda não causaram muito impacto em aplicações domésticas ou industriais, principalmente devido a falta de um sistema de controlo robusto, confiável e flexível que permitiria que estes robôs operassem em ambientes dinâmicos, pouco estruturados, e habitados por seres humanos.

Por muito tempo os sistemas de controlo para robôs móveis autónomos se dividiam em duas abordagens principais: sistemas deliberativos e sistemas reactivos. Os sistemas deliberativos se baseiam fortemente em um modelo do ambiente. Eles conseguem tirar proveito do conhecimento "a priori" sobre o ambiente. Os sistemas deliberativos possuem características essenciais para a elaboração de planos. Os sistemas reactivos possibilitam um navegação robusta, bem adaptada as características do mundo real. São modulares e permitem, com a adição de novos módulos, uma fácil melhoria de seus comportamentos. Os sistemas reactivos possuem características essenciais para a execução de um plano com uma capacidade de reacção mais imediata a eventos imprevistos. A união destas duas técnicas pode produzir um sistema híbrido que possua o melhor de cada um.

1.1. Navegação

Quando nos referimos a navegação de robôs móveis autónomos, queremos descrever as técnicas que fornecem os meios para que um robô autónomo se mova de forma segura de um local a outro do ambiente. Encontrar um caminho de uma determinada posição até um destino é um dos problemas fundamentais da robótica móvel autónoma. Um algoritmo de planeamento de trajectória deve garantir um caminho até o destino, ou indicar se o destino é inacessível. Os métodos tradicionais de planeamento de trajectória assumem que o conhecimento sobre o ambiente é completo e perfeito. Utilizando modelos de mundo, um caminho livre de colisões é planejado e uma trajectória é traçada. Em uma situação mais realística o ambiente pode ser alterado com o passar do tempo. Por exemplo, um robô móvel autónomo que actua em um ambiente onde pessoas circulam deve ser capaz de lidar com a troca de posição de certos objectos (cadeiras, caixas, etc.).

1.2. Veículos Moveis

Existem diversos tipos de robôs aplicados as diferentes áreas, no caso da robótica móvel temos os seguintes veículos moveis:

- Veículo terrestre / Indoor
- Veículo terrestre / Outdoor
- Veículos de exploração espacial
- Veículos aquáticos / barcos e submarinos
- Veículos aéreos / dirigíveis, aviões e Helicópteros

2. Arquitectura do Sistema de Controlo

2.1. Objectivos

Pretende-se desenvolver um sistema de controlo de missões para veículos autónomos, no qual será efectuado o controlo das manobras a executar e o interface das mesmas com sistema de controlo e navegação de veículo.

Numa primeira fase este sistema será desenvolvido preferencialmente para um robot móvel da classe de condução autónoma para participação no festival nacional de robótica. O sistema de missões deve permitir a definição de missões complexas como as existentes nesta competição. Neste caso este sistema de controlo de missão deve ser implementado nos veículos Runner e SpeedRunner do LSA-ISEP.

2.2. Planeador

Esta parte do sistema terá como principal função o planeamento da missão a realizar pelo veículo autónomo. Isto será executado antes do veículo autónomo entrar em funcionamento, visto que normalmente existe um conhecimento prévio das tarefas necessárias a realizar para um correcto comprimento dos objectivos da missão.

Para o nosso caso definimos que o planeador irá introduzir num ficheiro, com uma organização previamente estipulada, todos os dados referentes a missão que se

pretende executar, sequência de funções que se podem chamar, valores das condições inicias no caso de serem necessárias e condições de teste, referidas mais a frente. Assim sendo no caso de existir a necessidade de se alterar uma missão não será necessário compilar o código, bastando simplesmente alterar esse ficheiro.

2.3. Missão

A missão será tudo o que o veículo autómato irá realizar ou precisará de saber para completar a sua tarefa (os seus objectivos). A execução da missão vai exigir uma tarefa de pilotagem onde se articula as diferentes modalidades de funcionamento disponíveis para o mesmo.

Cada missão será constituída por um conjunto de manobras, as quais poderão ter um diverso conjunto de acções, transições ou ocorrências, que o fará transitar para uma manobra específica.

No nosso caso, e como pensado a missão será lida segundo algumas normas a partir do ficheiro que o planeador escreveu.

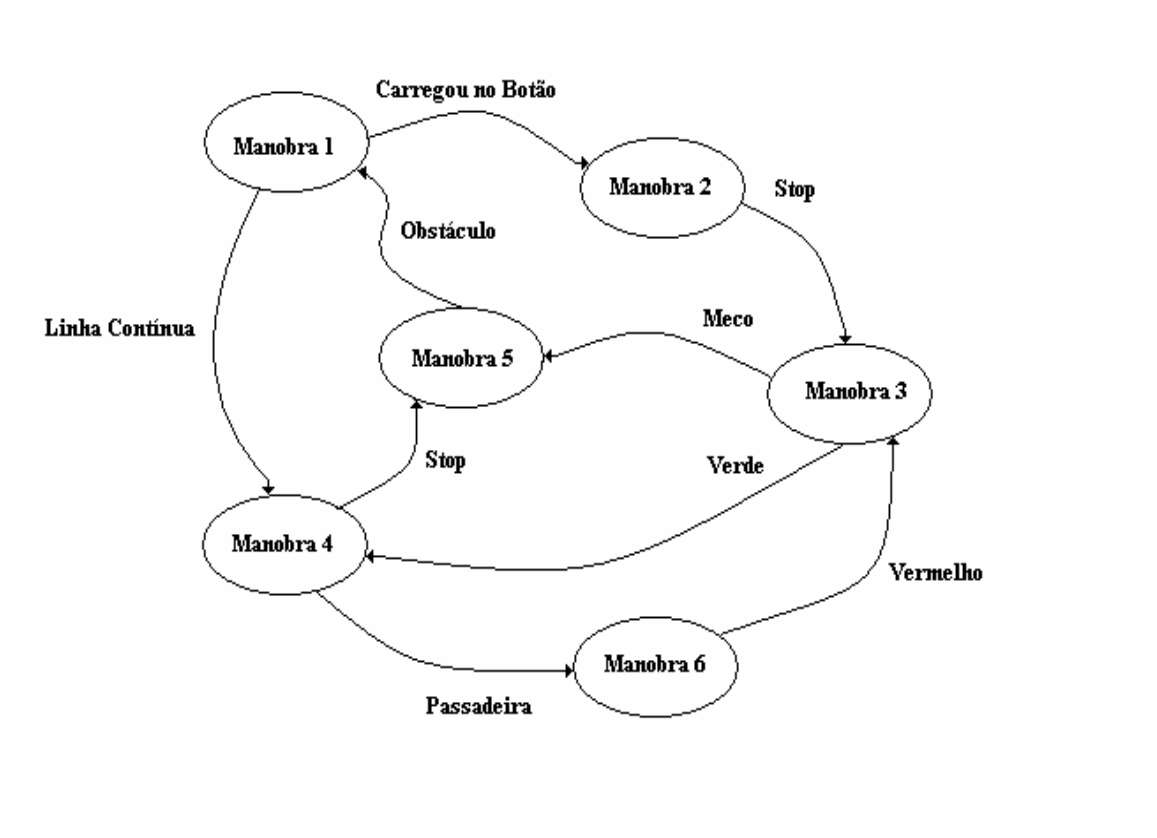


Fig 1. Grafo direccional de uma Missão

3. Projectar o Gestor de Missões

O gestor de missões fará essencialmente o controlo de tudo o que o veículo autónomo poderá realizar. O primeiro ponto a ter em consideração, e já falado, foi de onde seria lida a missão, de modo a se criar uma estrutura de simples execução no autómato. Deste modo, decidimos que o planeador iria introduzir a missão a realizar num ficheiro, pelo facto de ser mais simples, e no caso de se alterar uma missão não ser necessário alterar código, como o falado.

O veículo autónomo para a realização de uma missão, conterá diversas manobras. Cada manobra conterá acções (todas as funções para o controlo do mesmo e concretização da missão), e as transições (para transitar de manobra para manobra ou executar uma função especifica).

De seguida foi pensado em como seria possível identificar as acções e transições, que vão constituir as manobras, a partir do ficheiro, e relaciona-las com as funções para a realização das mesmas. Deste modo, foram criados registos destas, sendo esses registos identificadas por um nome, nome esse que será utilizado aquando da leitura do ficheiro de missões, para cópia destas para a manobra apropriada.

Outro ponto a ter em atenção foi o facto de um veículo autómato poder realizar diversas tarefas ao mesmo tempo, isto é, duas acções totalmente independentes e sem relação aparente (paralelismo). A estrutura do gestor de missões foi adaptada para que isso fosse possível de realizar, podendo deste modo, o gestor que estamos a criar ser utilizado em diversos veículos autómatos, desde os mais simples aos mais complexos como os veículos industriais.

4. Registar Funções Essenciais

A quando da leitura do ficheiro de missões, será necessário ter já dois registos, um com todas as acções disponíveis e outro com todas as transições que poderão existir, como o referido anteriormente. Deste modo, quando se ler num ficheiro uma determinada manobra com uma acção e suas transições, procurar-se-á nesses registos suas correspondências, não só para verificar a sua existência, como também serão utilizados para o preenchimento da estrutura do gestor de missões.

4.1. Registo de Acções

O registo de acções é essencialmente o registo de todas as funções para o controlo do veículo autómato e concretização da missão numa lista, podendo conter acções generalistas (“recta”, “curva”, “túnel”…), ou mais específicas. Esse registo será utilizado, aquando da criação de uma missão, pelo gestor de missões, que como visto anteriormente, é um conjunto de manobras com diversas acções e transições especificas para cada uma delas, para pesquisa das acções existentes e cópia das destas. Cada acção será registada com um nome específico, fornecido na altura do registo.

Cada acção irá conter ainda um diverso conjunto de apontadores para funções (para o controlo e realização da acção). Nesse conjunto de apontadores de funções poderemos encontrar alguns dos seguintes apontadores:

- Cria – Função de criação da acção (ou que criou)
- Inicializa – Função de inicialização de uma acção
- Controlo – Função que irá executar a manobra e fazer o controlo da acção
- Termina – Função que termina uma determinada acção com alguma instrução específica
- Destrói – Função para destruição da acção (retira-la da memória)

Nem todos os apontadores para as funções poderão ser necessários, poderá existir acções em que por exemplo não será necessário o apontador “Termina”, assim sendo, esse apontador será preenchido com o valor NULL, e assim sucessivamente.

As acções generalistas aquando da sua utilização pelo gestor de missões para as manobras poderão ser tornadas mais específicas, isto é, poderá existir numa missão uma manobra com uma acção que é uma cópia de uma “recta”, mas com dados mais específicos, um início, um fim, entre muitos outros dados. Para que isso seja possível, no registo das acções poderão ser acrescentadas variáveis, as quais poderão ser alteradas aquando da criação de uma manobra numa missão, para posterior inicialização de uma dada acção.

Cada variável será definida com os seguintes campos:

- Nome
- Tipo
- Default

Na variável "Nome" irá ser descrito o nome da variável a ser inserida, no campo "Tipo" será inserido que tipo de variável é podemos encontrar (inteira, float, char …), no campo "Default" será inserido um valor predefinido que no caso da variável não ser alterada, será esse mesmo valor utilizado para essa variável.

Aquando da implementação da acção, todas as variáveis serão passadas para a função de inicialização, pela ordem que foram registadas.

Esse registo irá ser efectuado numa lista ligada como o representado de seguida.

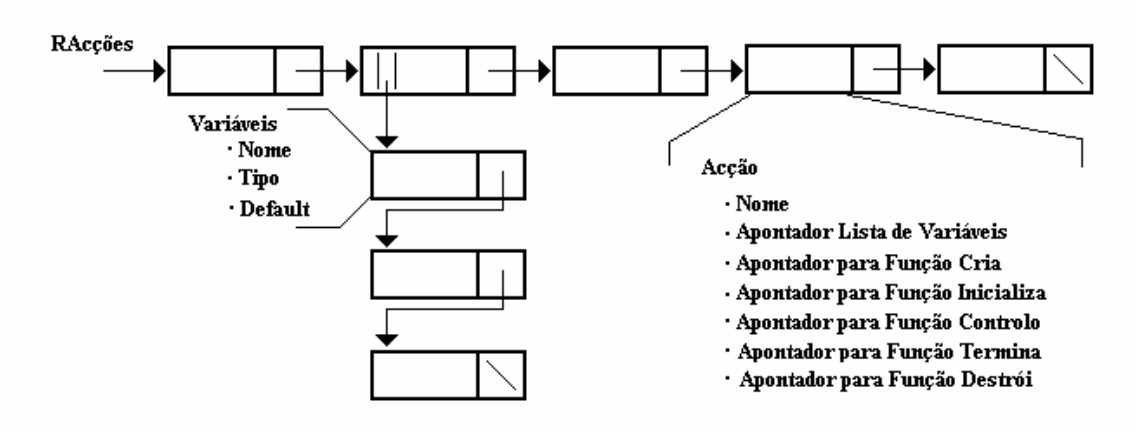


Fig 2. Lista de Acções

4.2. Registo das Transições

O registo das transições consiste no registo de todas as transições que possam ocorrer durante as manobras. Este registo de transições será também necessário, assim quando for criada uma missão, pelo gestor de missões, que como visto anteriormente, é um conjunto de manobras com diversas acções e transições especificas para cada uma delas, será pesquisada a transição pretendida, para confirmação da sua existência e cópia da mesma para uma manobra. Cada transição conterá um nome específico, dado na altura do registo da mesma.

Cada transição irá conter ainda um apontador para a função de teste da mesma (semáforo verde, semáforo vermelho, virar a direita, mecos, stop…), retornando se essa dada transição é verdadeira ou falsa. Conterá ainda uma variável, que

dependendo do estado dela, e do resultado da transição poderá mandar executar uma função especifica, que contém o seu apontador guardado na estrutura a registar.

Estados da variável "Executar":

- 0 – Não Executará nada (Default)
- 1 – E valor retornado pela transição for verdadeiro executa aplicação
- 2 – E valor retornado pela transição for falso executa aplicação

Esse registo irá ser efectuado numa lista ligada como o representado de seguida.

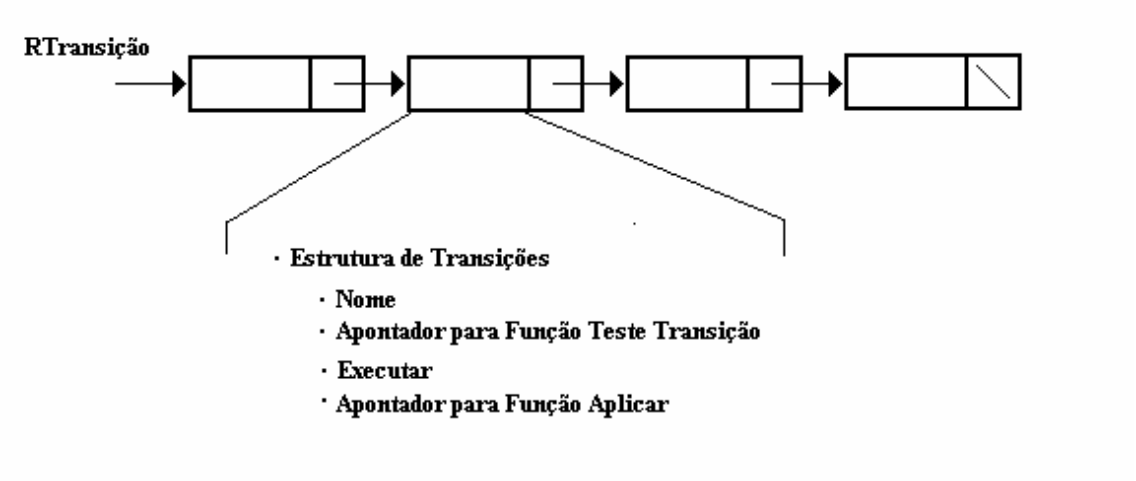


Fig 3. Lista de Transições

5. Gestor de Missões

O gestor de missões terá como principal objectivo a execução e controlo das missões a executar por cada veículo autónomo. Deste modo serão criadas diversas estruturas e listas, sendo algumas das entradas destas copiadas a partir do registo de acções e registo das transições, para o correcto teste e execução de todas as manobras do veículo autónomo, lidas segundo algumas normas a partir do ficheiro de missões.

Cada veículo autónomo para atingir o objectivo da missão poderá ter uma única sequência de manobras, como poderá ter várias sequências de manobras ou acções em paralelo, sem qualquer relação a funcionarem ao mesmo tempo (paralelismo).

Deste modo esta aplicação (Fig 4 em anexo), poderá ser utilizada em diversos veículos, assim como em veículos industrias, que executem diversas tarefas ao mesmo

tempo ou um simples autómato, com um único controlo, uma única sequência de manobras.

5.1. Paralelismo

Ao existir vários conjuntos de sequências de manobras a correr em paralelo, podemos ter o controlo de diversos aspectos de um veículo autónomo ao mesmo tempo, podendo existir dois tipos de paralelismo, "paralelismo de alto nível" e "paralelismo de baixo nível".

No caso de paralelismo de alto nível poderemos ter como exemplo um robô industrial, que ao mesmo tempo que tem que percorrer um percurso sem parar, tem que apanhar com um braço robótico todo o material que encontrar nas plataformas ao seu lado direito e colocar nas plataformas do seu lado esquerdo, contendo assim um conjunto sequencial de manobras que fará o controlo da trajectória do mesmo, e um outro conjunto sequencial de manobras que sempre que encontrar ao seu alcance algum material irá transporta-lo, assim sendo temos dois conjuntos sequenciais de manobras sem relação aparente e a funcionarem ambas ao mesmo tempo em paralelo, para atingir um único objectivo, a missão.

No caso de paralelismo de baixo nível poderemos ter como exemplo um robô que aspira, em que este durante a sua locomoção, se detectar lixo terá que ligar o sistema de aspiração, assim sendo, depois de detectado o lixo este autómato continuara a sua locomoção (trajecto normal) e em paralelo terá o sistema de apanhar lixo ligado, tendo assim uma manobra que conterá duas acções em paralelo.

5.1.1. Paralelismo de Alto Nível

Para a realização do paralelismo de alto nível, foi criada uma lista ligada a qual demos o nome de missão, sendo essa a nossa primeira lista do gestor de missões. Essa lista conterá um registo de todas as sequências de manobras que poderão estar a funcionar em paralelo. No caso do veículo autónomo ter uma tarefa simples (uma única sequência de manobras), existirá simplesmente uma entrada nessa lista, no caso de existirem duas ou mais sequências a funcionarem em paralelo, essa lista conterá duas ou mais entradas.

Cada sequência de manobras a ser inserida no gestor de missões será constituída por:

- Nome – Nome do ficheiro onde foi lida a sequência de manobras
- Apontador para a estrutura manobra
- Estado – Variável de estado
  - 1 – Sequência de manobras criada com sucesso
  - 2 – Sequência de manobras em execução automática
  - 3 – Sequência de manobras em pausa
  - 4 – Sequência de manobras parada
- Apontador para a estrutura Manobras em que parou

A variável de estado foi criada para saber o estado de cada sequência de manobras, e para que posteriormente se possa construir um gestor de todas as sequências de manobras que estão a ser executadas, ou até mesmo criar um gestor que em caso de necessidade mande parar uma sequência de manobras (das que estão a correr em paralelo), até que outra termine a sua tarefa.

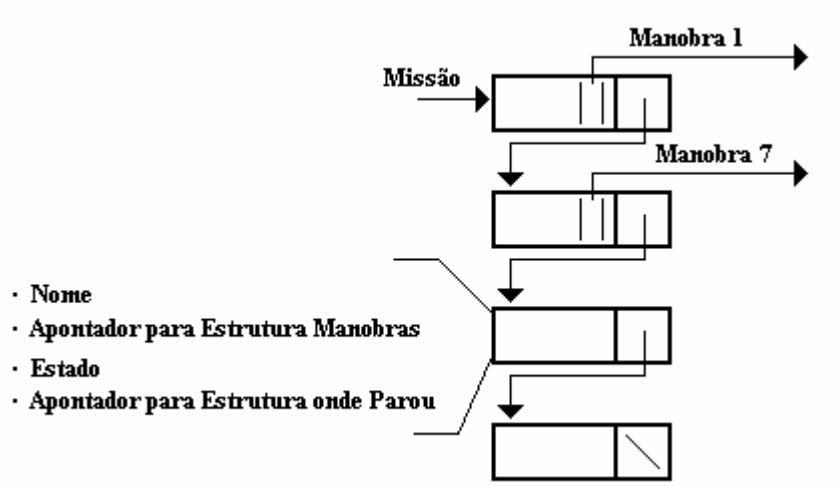


Fig 4.1. Estrutura de Missão

5.1.2. Paralelismo de Baixo Nível

Para a realização do paralelismo de baixo nível o registo de cada manobra vai conter um diverso conjunto de acções, sendo essas guardadas numa lista na estrutura de manobra (lista de acções). Nessas acções serão encontrados todos os apontadores e funções para a implementação das manobras. Se fosse uma manobra simples, em que não existisse paralelismo, simplesmente

encontraríamos uma acção na lista de acções, se existir paralelismo em cada lista existiriam mais entradas, de acordo com o número de acções a funcionar em paralelo.

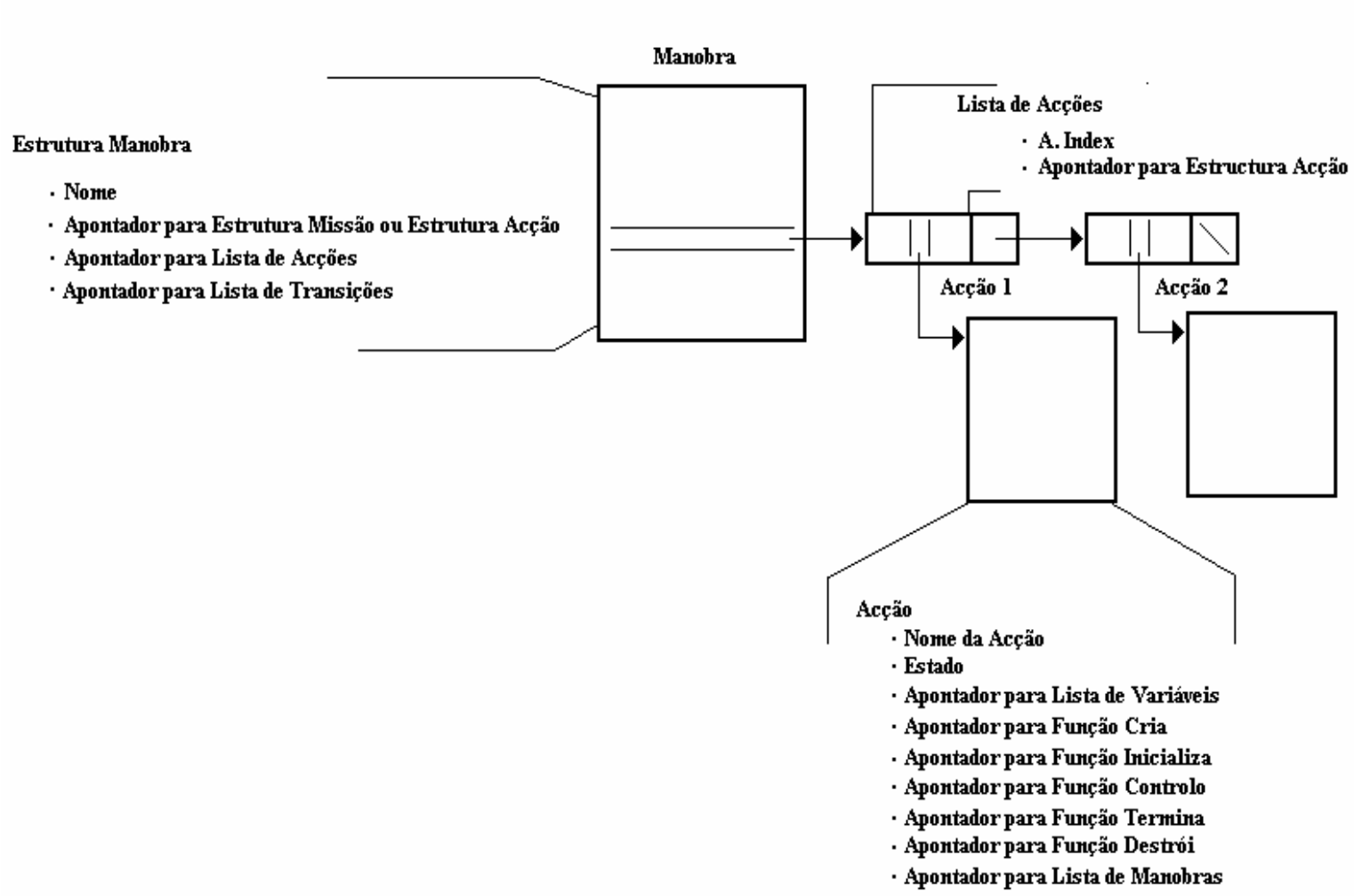


Fig 4.2. Estrutura Manobra e Acção

Na figura seguinte (Fig 4.3.) poderemos encontrar um exemplo de uma manobra que contém duas acções em paralelo, visto contermos duas entradas na lista de acções. Uma dessas acções (a 1ª) contém um conjunto de manobras (manobra 9 e 10), isto é, é uma acção que vai fazer executar uma sequência de manobras sequenciais, dependendo das transições de cada manobra.

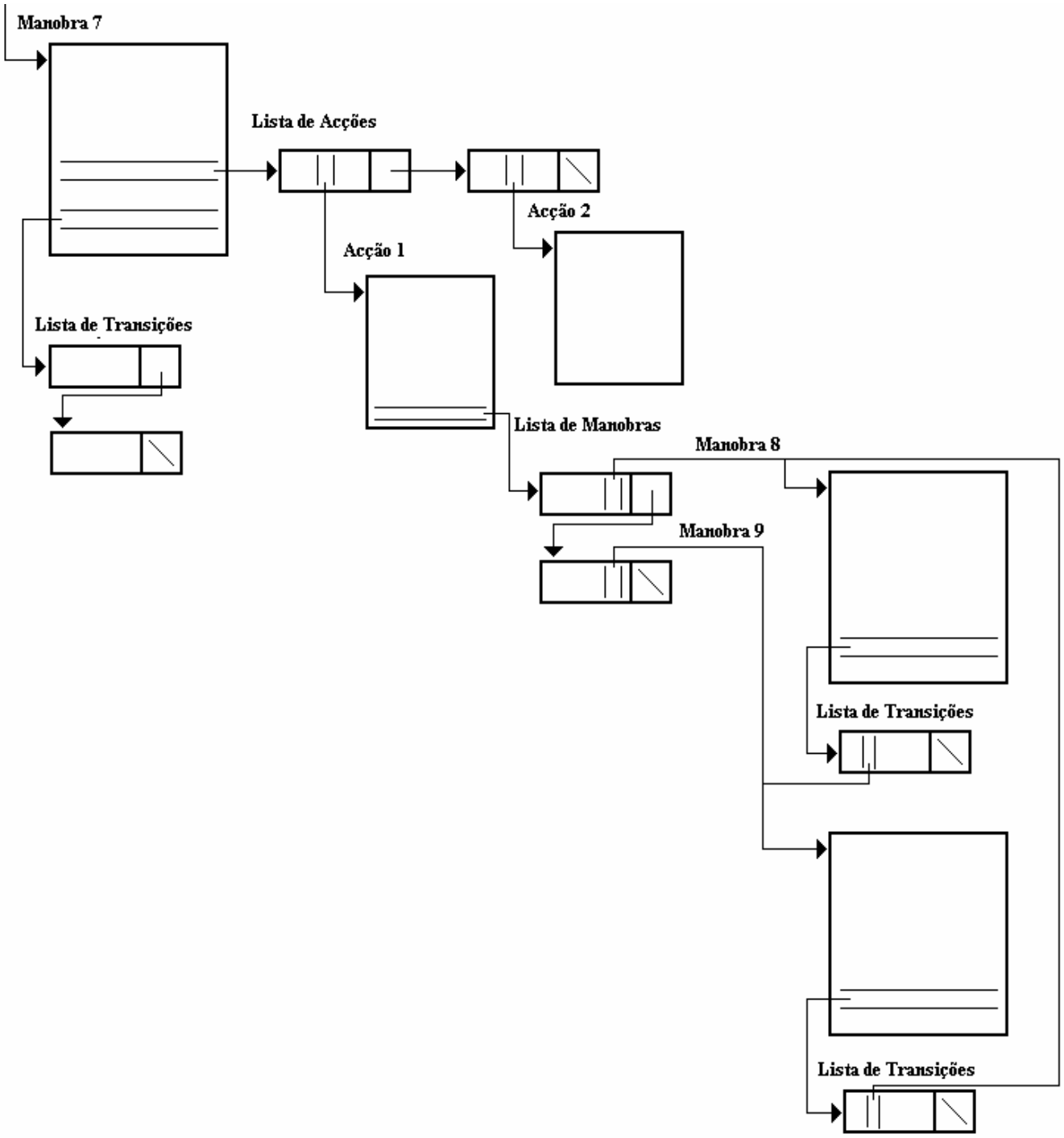


Fig 4.3. Exemplo: Paralelismo Baixo Nível

5.2. Estrutura Manobras

A estrutura manobra conterá todos os dados necessários para a execução de uma manobra e suas transições, para que se possa executar a próxima manobra, quando necessário.

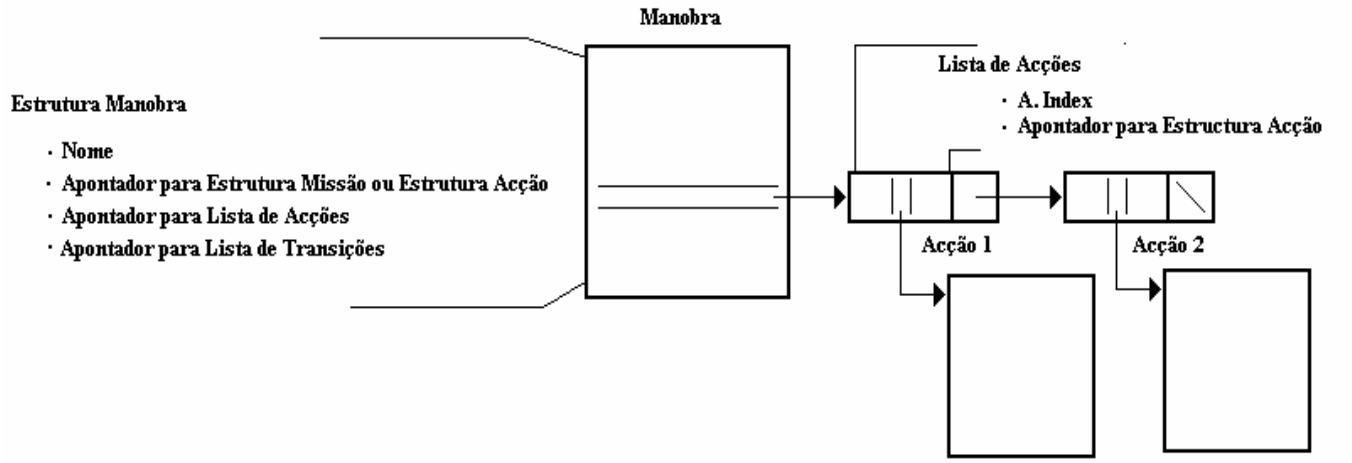


Fig 4.4. Estrutura Manobra

Esta estrutura contem um campo chamado nome, no qual será gravado o nome da manobra, lido a partir do ficheiro. Contém também um campo que é um apontador par uma estrutura missão ou acção, campo esse que servirá para se alterar o valor da variável estado na estrutura missão ou acção.

Na estrutura manobra podemos encontrar duas listas essenciais, a lista das acções e a lista das transições. A lista das acções e como o visto anteriormente servirá para aplicar o paralelismo de baixo nível, correspondendo o número de acções dessa lista ao número de acções em paralelo. Cada acção a ser inserida nessa lista conterá um campo chamado A. Index (um campo de numeração automática, feito pelo gestor de missões, meramente informativo), e outro campo que é um apontador para a estrutura de acção.

Na lista de transições, cada entrada contém uma cópia de uma transição registada no registo de transições, e poderá conter um apontador para uma manobra:

- Estrutura de transições
  - Nome
  - Apontador para a função teste transição
  - Executar
  - Apontador para a função aplicar
- Apontador para a próxima manobra

O apontador para a próxima manobra, servirá para sabermos qual a próxima manobra a executar caso a condição a verificar seja verdadeira. Teremos ainda, e como explicado anteriormente aplicar uma só função. Se o campo do apontador para uma manobra for nulo (“NULL”) só se executará a aplicação. No caso especifico de se querer que quando se carregue num botão ele pare a missão, simplesmente chamamos uma função que irá mandar parar de executar a sequência de manobras pretendida, alterando o seu estado na estrutura de missão (variável Estado).

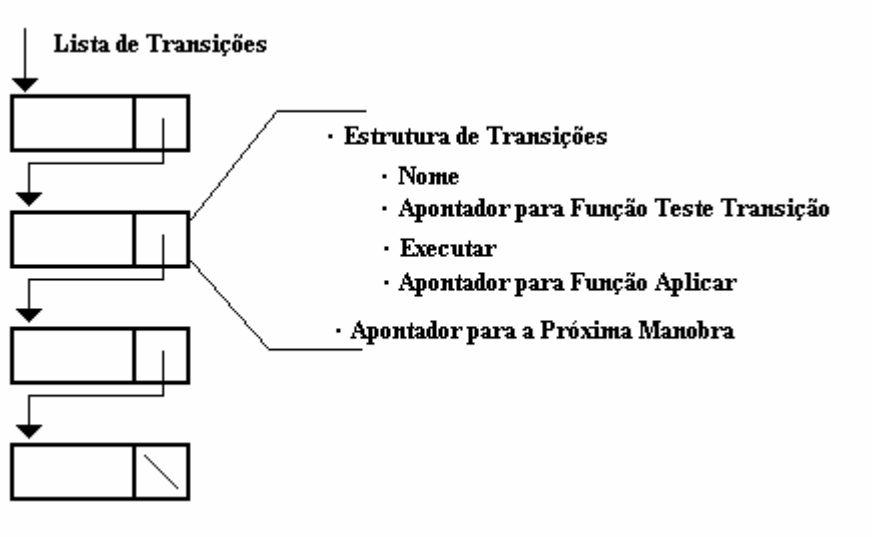


Fig 4.5. Listas de Transições

5.3. Estrutura Acção

A estrutura acção é essencialmente a estrutura de execução e controlo de uma acção/manobra. Nessa estrutura podemos encontrar como primeiro campo o nome, no qual será guardado o nome da acção (do registo acções), de onde foram copiados os apontadores para as funções de controlo e execução. A variável estado servirá para saber o estado das sub-manobras (manobras pertencentes a sua lista de manobras), deste modo nesta variável poder-se-á saber qual a sub-manobra que se encontra em execução. Todas as estruturas acção poderão ter uma lista de variáveis, sendo esta uma cópia das variáveis registadas no registo das funções das acções. Se aquando da leitura de uma acção de um ficheiro for encontrado um valor para uma

variável ele será guardado no campo valor, se não for encontrado nenhum valor, será guardado o valor por defeito (a partir do registo de acções).

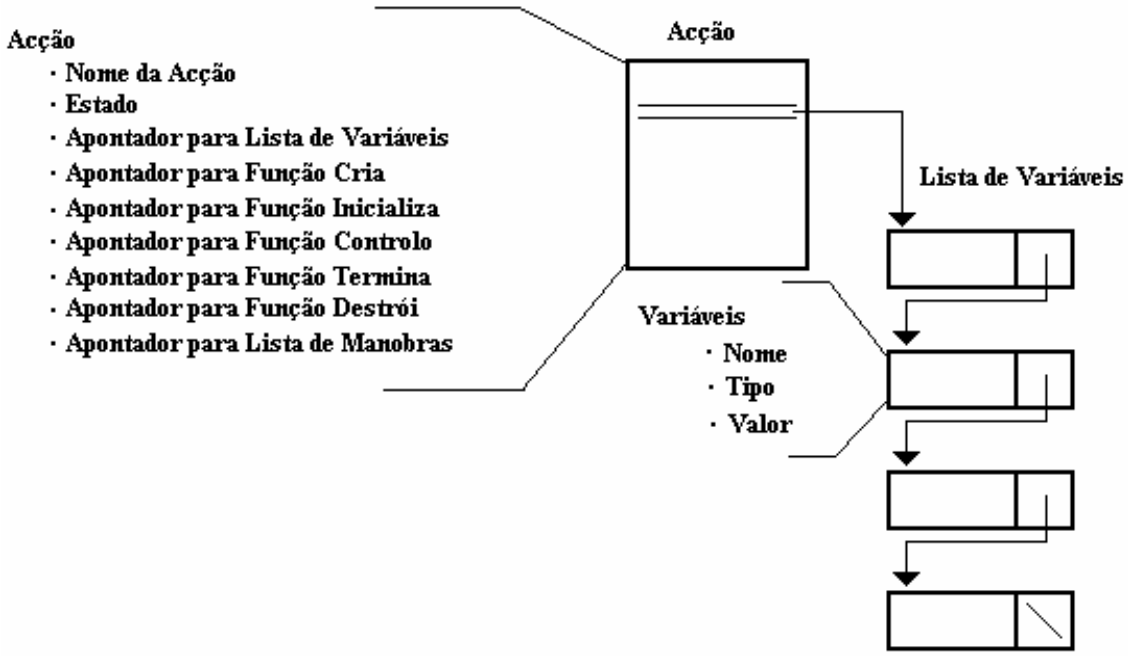


Fig 4.6. Estrutura Acção

Outro aspecto importante na estrutura de acções, é a lista de manobras, deste modo podemos criar sub-manobras dentro das acções das manobras. Na lista de manobras podemos encontrar apontadores para as diversas estruturas de manobras (sub-manobras) que poderão ser executadas pela acção de uma manobra. Cada sub-manobra terá uma numeração sequencial, sendo essa registada no campo M. Index.

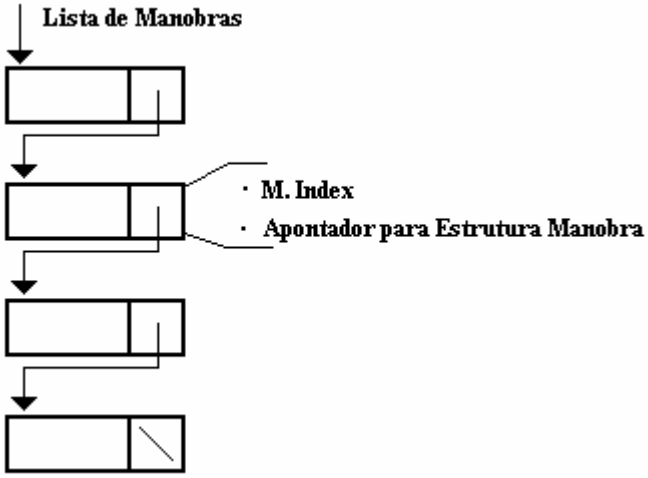


Fig 4.7. Listas de Manobras

6. Ficheiro de Missões

Os ficheiros de missões conterão descritos todos os conjuntos de manobras, acções e transições para que uma missão possa ser carregada por software com as estruturas anteriormente descritas e executada. Um ficheiro de missões pode ainda ter uma chamada para outro ficheiro de missões, sendo que o ultimo contém mais manobras. Todos estes ficheiros de missões serão estruturados e escritos segundo algumas normas, normas essas que passaremos a descrever de seguida.

6.1. Tipos de Variáveis e Funções

Ao longo do ficheiro podemos encontrar diversos tipos de variáveis e funções.

O tipo de variáveis identificará de que género são as variáveis, a quando da sua declaração ou a serem declaradas. As variáveis serão muitas das vezes passadas para as funções como argumentos, ou variáveis de retorno. As variáveis de retorno serão muito pouco utilizadas visto que muitas das variáveis serão globais a uma manobra, sendo utilizadas pelas diferentes funções da manobra.

6.1.1. Int

O tipo de dados int (inteiro) serve para armazenar valores numéricos inteiros, podendo ocupar 2 ou 4 bytes.

6.1.2. Char

Este tipo é utilizado para se guardar valores definidos dentro da tabela ASCII (-127 a 127).

6.1.3. Float

O tipo de dados float serve para armazenar números de ponto flutuante, ou seja com casas decimais, em que cada número ocupa 4 bytes.

6.1.4. Double

O tipo de dados double serve para armazenar números de ponto flutuante de dupla precisão, tem o dobro do tamanho do float e portanto o dobro da capacidade, em que cada número ocupa 16 bytes.

6.1.5. Struct

Em C podem ser usadas estruturas (chamados de registos em outras linguagens de programação). As estruturas são grupos de variáveis agrupadas. Uma estrutura cria um novo tipo de variáveis.

6.1.6. Long

Este modificador aumenta a capacidade de um tipo de variável.

6.1.7. Short

Normalmente utilizada para representar inteiros mais curtos.

6.1.8. Unsigned

A palavra reservada unsigned indica que não haverá representação de sinal do valor (todos os valores tornam-se positivos), o normal é signed, em que todas as variáveis têm uma gama positiva e negativa.

Tipo	Num de bits	Intervalo	
		Inicio	Fim
Char	8	-128	127
unsigned char	8	0	255
signed char	8	-128	127

Int	16	-32.768	32.767
unsigned int	16	0	65.535
signed int	16	-32.768	32.767
short int	16	-32.768	32.767
unsigned short int	16	0	65.535
signed short int	16	-32.768	32.767
Long int	32	-2.147.483.648	2.147.483.647
signed long int	32	-2.147.483.648	2.147.483.647
unsigned long int	32	0	4.294.967.295
Float	32	3,4E-38	3.4E+38
Double	64	1,7E-308	1,7E+308
Long double	80	3,4E-4932	3,4E+4932

Deste modo no ficheiro para a descrição de variáveis utilizaremos a seguinte estrutura, em que no campo N indica qual a variável, TYPE conterá o tipo de variável (um dos anteriores), e no campo VAL será escrito qual o conteúdo que a variável deverá conter:

<VAR>
{N=”x” TYPE=”int” VAL=”-3”}
{N=”y” TYPE=”unsigned int” VAL=”2”}
</VAR>

6.2. Funções Manobra

Todas as manobras conterão acções e transições, para criar uma nova manobra, começaremos por lhe dar um nome (MMAN). Depois de termos indicado toas as acções e transições de uma manobra, concluiremos essa manobra com o comando </MMAN>.

<MMAN=”nome_m”>
</MMAN>

6.3. Funções Acção

As acções já estão registadas no registo de acções, deste modo aqui só necessitaremos de indicar o nome da acção (MAC), para posteriormente a procurarmos nesse registo. No caso de uma acção simplesmente implementar sub-manobras, poderá ser necessário indicar qual a sub-manobra que será iniciada primeiro, deste modo existirá um campo (INIT) no qual indicaremos o número da sub-manobra (a contagem das sub-manobras começará em 0). No caso de não ser indicada nenhuma sub-manobra utilizar-se-á a sub-manobra zero (0).

<MAC=”nome_acção” INIT=”0”>

</MAC>

6.4. Funções de Transição

Do mesmo modo que as acções se encontram registadas, as funções de transição também estão registadas no registo de transições, deste modo aqui só necessitaremos de indicar o nome da transição (TTRANS), para posteriormente a procurarmos nesse registo. Será ainda necessário indicar qual manobra seguinte a executar (JPM), no caso de a transição ser verdadeira. Posteriormente podemos indicar de quanto em quanto tempo queremos que a transição seja verificada (TIME), assim deste modo se as restantes funções de controlo da manobra demorarem muito tempo, a transição será executada num tempo mais curto que este.

<TTRANS=”nome_t” JPM=”manobra” TIME=”em segundos”>

6.5. Funções Ler Ficheiros

Como o dito inicialmente, um ficheiro missões pode ter uma chamada para outro ficheiro de missões, em que este pode simplesmente conter uma acção, uma manobra, ou um conjunto de transições. Deste modo terá que existir alguma atenção onde chamaremos esse ficheiro.

<LFILE=”nome_ficheiro”>

Normalmente os ficheiros conterão manobras, assim sendo antes da instrução de ler o ficheiro teremos que iniciar uma nova acção. Se o ficheiro contivesse uma acção teria que ser colocada a sua chamada depois da abertura de uma nova

manobra, se por outro lado o ficheiro contivesse um conjunto de transições teríamos que ter o cuidado de o ler antes de fechar uma manobra.

7. Leitura de um Ficheiro Missões

Depois de estruturado o modo de escrita do ficheiro de missões, há que estruturar o modo e a sequência de leitura desse ficheiro. Durante a leitura de uma missão irá ser criado um ficheiro ".log" que vai conter os erros, não só de leitura do ficheiro, como também os erros a quando da criação de todas as estruturas.

7.1. Ficheiro ".log"

Neste ficheiro ira conter todos os erros de leitura do ficheiro, como também os erros a quando da criação de todas as estruturas. Todos os erros a serem escritos serão iniciados por caracteres predefinidos, sendo que tudo que estiver escrito depois até ser encontra uma nova linha será referente a esse erro. Teremos então os seguintes caracteres de descrição de erros:

Erro	Descrição
<LF>	Caracteres inválidos no ficheiro de missões (leitura)
<CM>	Criação na memória (espaço…)
<R>	Falta de registos das funções (cópia dos registos das funções)
<AV>	Impossível alterar variáveis (Manobras criadas)
<TJPM>	Manobra inválida para transitar (transições)

8. Exemplo de um Ficheiro Missões

Para a leitura de um ficheiro, e referente a fig 4., podemos ter o seguinte exemplo:

"Missão.mjrf"

<MMAN="**Manobra 1**">
<MAC="**Acção_1**" INIT="2">
<MMAN="**Manobra 2**">

```
                    <MAC=”Recta”>
                        <VAR>
                            {N=”x” TYPE=”float” VAL=”0”}
                            {N=”y” TYPE=”float” VAL=”0”}
                            {N=”final” TYPE=”int” VAL=”10”}
                        </VAR>
                    </MAC>
                    <TTRANS=”Stop” JPM=”Manobra 3”>
                </MMAN>
                <MMAN=”Manobra 3”>
                    <MAC=”Curva”>
                        <VAR>
                            {N=”r” TYPE=”float” VAL=”0”}
                            {N=”l” TYPE=”int” VAL=”0”}
                            {N=”t” TYPE=”int” VAL=”10”}
                        </VAR>
                    </MAC>
                    <TTRANS=”VirarDireita” JPM=”Manobra 2”>
                    <TTRANS=”VirarEsquerda” JPM=”Manobra 4”>
                </MMAN>
                <MMAN=”Manobra 4”>
                    <MAC=”Acção_1”>
                        <LFILE=”Obras.mjrf”>
                    </MAC>
                    <TTRANS=”VirarDireita” JPM=”Manobra 2”>
                    <TTRANS=”Stop” JPM=”Manobra 3”>
                </MMAN>
            </MAC>
        </MMAN>
        <MMAN=”ParaleloDuasAcções”>
            <MAC=”Acção_1” INIT=”1”>
                <MMAN=”Manobra 8”>
                    <MAC=”CaptarSinal”>
                        <VAR>
                            {N=”ang” TYPE=”float” VAL=”30”}
                            {N=”L” TYPE=”int” VAL=”40”}
                        </VAR>
                    </MAC>
                    <TTRANS=”Forte” JPM=”Manobra 9”>
                </MMAN>
                <MMAN=”Manobra 9”>
                    <MAC=”Transmitir”>
                        <VAR>
                            {N=”canal” TYPE=”int” VAL=”5”}
                        </VAR>
                    </MAC>
                    <TTRANS=”End” JPM=”Manobra 8”>
                </MMAN>
            </MAC>
            <MAC=”RodarAntena”>          ←——— Acção 2
```

```
            </MAC>
            <TTRANS=”DelocarSinal”>
            <TTRANS=”PararRodar”>
        </MMAN>

“Obras.mjrf”
        <MMAN=”Manobra 5”>
            <MAC=”Linha”>
                <VAR>
                    {N=”f” TYPE=”float” VAL=”20”}
                    {N=”x” TYPE=”float” VAL=”40”}
                </VAR>
            </MAC>
            <TTRANS=”Meco” JPM=”Manobra 6”>
        </MMAN>
        <MMAN=”Manobra 6”>
            <MAC=”Túnel”>
                <VAR>
                    {N=”c” TYPE=”int” VAL=”40”}
                </VAR>
            </MAC>
            <TTRANS=”Stop” JPM=”Manobra 5”>
        </MMAN>
```

9. Referências

- http://www.cis.upenn.edu/group/mars/site/multimedia.htm#pictures
- http://www.swarms.org/
- http://www.cis.upenn.edu/mars
- http://www.grasp.upenn.edu/

10. Anexos

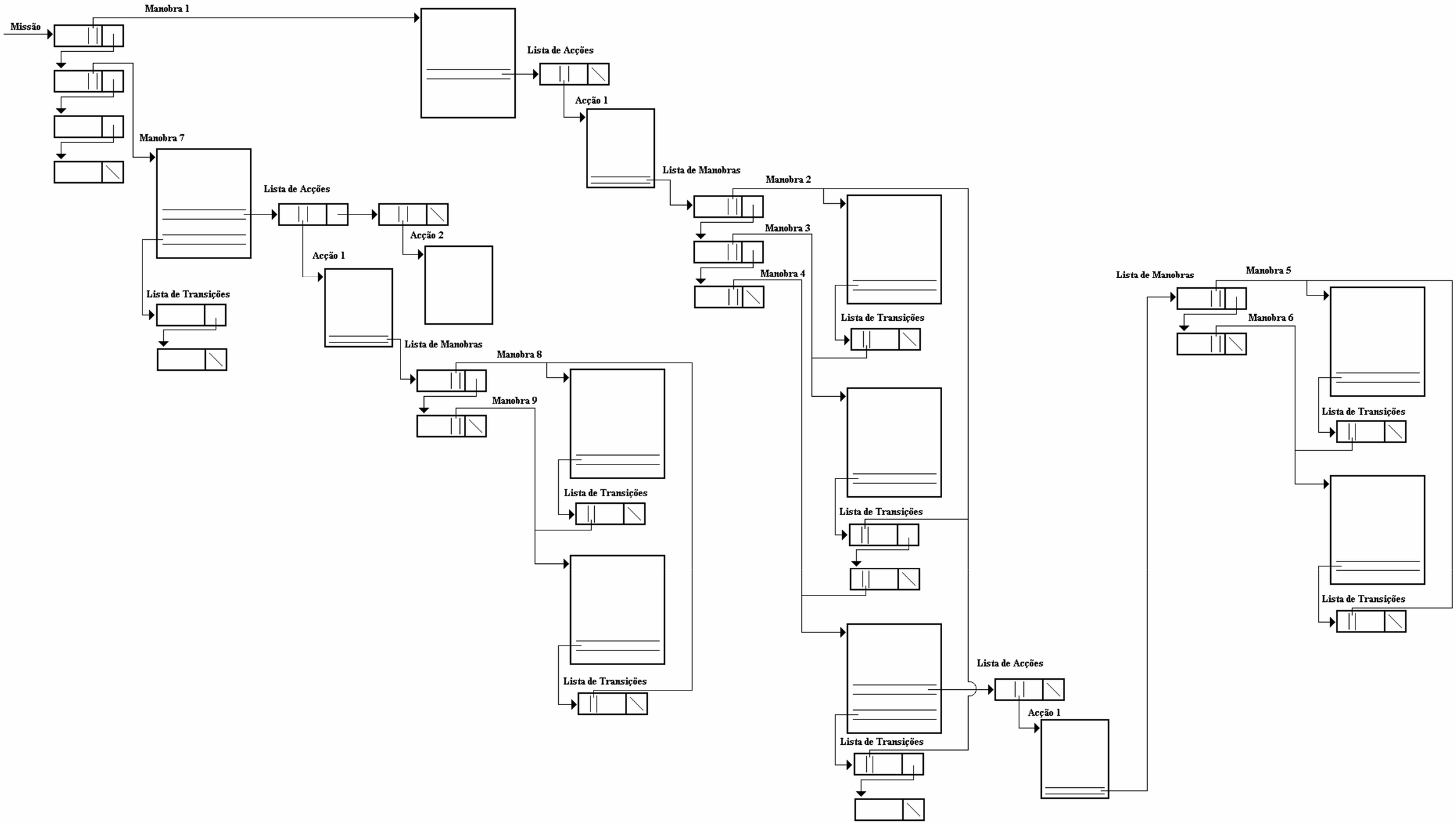


Fig 4. Gestor de Missões

11. Vocabulário

- **Planeador –** Parte do sistema que terá como principal função o planeamento da missão a realizar pelo veículo autónomo.
- **Missão –** A missão será tudo o que o veículo autómato irá realizar ou precisará de saber para completar a sua tarefa (os seus objectivos).
- **Ficheiro de Missões –** Ficheiro com uma organização previamente estipulada, contendo todos os dados referentes a missão que se pretende executar, sequência de funções que se podem chamar, valores das condições inicias no caso de serem necessárias e condições de teste (transições).
- **Sequência de Manobras –** Conjunto de manobras que serão executadas sequencialmente, isto é, somente irá executar uma manobra de cada vez, transitando para a seguinte quando existir uma transição verdadeira.
- **Manobra –** Uma manobra é constituída por acções, que irão fazer com que o veículo autónomo execute uma função, contendo ainda transições para poder executar outras manobras.
- **Acções –** As acções serão constituídas por todas as funções para o controlo do veículo autónomo, e concretização da missão.
- **Transições –** São conjuntos de funções de teste de alguns parâmetros do veículo autónomo, para transitar de manobra para manobra ou funções de execução de parâmetros específicos (exemplo: se um veículo detecta que há pouca luminosidade executa função que acenderá as luzes do mesmo).
- **Sub-manobras –** São manobras que serão encontradas dentro das acções de uma manobra.