FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Mecanismos de coordenação escaláveis para sistemas multi-veículo com restrições de comunicações e de energia

António Pedro Freire Martins Magalhães Rocha

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: João Tasso de Figueiredo Borges de Sousa

21 de Setembro de 2020

Resumo

Ao longo dos últimos anos temos visto um aumento significativo da utilização de veículos autónomos não tripulados com diversas aplicações. Um caso de uso são os veículos subaquáticos não tripulados, que servem como motivação para este trabalho, que tem aplicações em diversas áreas, como por exemplo, na oceanografia para mapear os fundos oceânicos, na arqueologia, na inspeção de instalações de petróleo/gás, entre outras.

Para maximizar a eficiência e melhorar a eficácia destas operações tem-se vindo a desenvolver diferentes formas de funcionamento em rede de vários veículos a cooperar entre si de modo a conseguir completar tais operações, uma vez que é sabido que os vários veículos a funcionar em cooperação entre si melhoram a performance de execução da operação em relação a uma operação realizada por um único veículo autónomo.

Neste documento vamos abordar estratégias de controlo e coordenação para uma rede de vários veículos autónomos que utilizam comunicações com alcances limitados para operação em zonas remotas. Vamos, depois, definir uma estratégia de controlo e coordenação descentralizada que permita efetuar a coordenação entre os múltiplos veículos autónomos a operar em rede e utilizar um sistema de comunicação baseado em mensagens partilhadas pelos veículos com horários e informação partilhada para coordenar o funcionamento individual de cada veículo da rede.

Pretende-se, posteriormente à realização deste documento, desenvolver um trabalho experimental onde se vai realizar a simulação dos mecanismos de controlo e coordenação projetados e desenvolvidos, e posterior análise dos resultados obtidos da simulação quando estes mesmos mecanismos de controlo e coordenação são aplicados a uma rede de veículos subaquáticos não tripulados para exploração dos oceanos.

Abstract

Over the past few years we have seen a significant increase in the use of automated unmanned vehicles with different applications. One use case is the unmanned underwater vehicles, that serve as motivation for this work, which have applications in several areas, such as in oceanography to map the ocean floor, in archaeology, in the inspection of oil/gas installations, among others.

In order to maximize efficiency and improve the effectiveness of these operations different forms of network operation have been developed for various vehicles to cooperate with each other in order to complete such operations, since it is known that the various vehicles operating in cooperation with each other improve the performance of the operation compared to an operation carried out by a single automated vehicle.

In this document we will address control and coordination strategies for a network of several automated vehicles with range-constrained communications capabilities operating in remote areas with no communication infra-structures. A decentralized control and coordination strategy will be introduced. It will allow coordination between the multiple automated vehicles operating in a network and use a communication system based on messages shared by the vehicles with timetables and shared information to coordinate the individual functioning of each vehicle in the network.

It is intended, after the completion of this document, to develop an experimental work where the simulation of the control and coordination mechanisms designed and developed will be carried out, and further analysis of the results obtained from that simulation when these control and coordination mechanisms are applied to a network of unmanned underwater vehicles for exploration of the oceans.

Agradecimentos

Chegou o momento que se pensa sempre que nunca vai chegar, até ao próprio dia. É o momento em que culmino o meu percurso de cinco anos na Faculdade de Engenharia da Universidade do Porto. E que cinco anos esses foram. Um percurso rico em vivências e aprendizagens, onde me formei não só academicamente mas também como pessoa, como homem. Aprendi a querer aprender, a ser curioso, a questionar. Aprendi a traçar objetivos, a lutar por eles e a conquistá-los.

Aprendi sozinho, com o meu esforço, mas não só. Aliás, aprendi maioritariamente em conjunto, com os outros, a observar, a ouvir. Desde professores, aos meus amigos e colegas. E por isso mesmo não posso deixar de dar uma palavra de apreço a quem mais me ensinou e auxiliou ao longo desta caminhada.

Primeiramente, quero deixar um agradecimento ao meu professor orientador João Borges Sousa, por ser a pessoa que de mais de perto acompanhou este meu último projeto, a dissertação. Apesar dos inconvenientes e dificuldades encontrados ao longo do ano letivo, e tendo noção daquilo que são as responsabilidades e ocupações do professor, nunca deixou de estar disponível para me ajudar e orientar, no sentido literal da palavra.

Há duas pessoas neste mundo, a quem mais que ninguém tenho de deixar o meu sincero obrigado, do fundo do coração, sendo elas, a minha mãe e o meu pai. Obrigado pelo acompanhamento, formação, educação, apoio... Não há quem me tenha dado mais na vida, pois eles deram-me tudo.

Gostaria ainda de agradecer à minha família que está sempre lá para mim quando necessito. São o meu porto de abrigo, e são também eles os meus melhores amigos. Um obrigado também à minha namorada, que hoje em dia é quem mais me atura, e também me ensina e ajuda a ultrapassar todos os obstáculos que a vida me coloca.

Quero também deixar uma palavra de apreço para com todos os docentes que me acompanharam, desde o início da minha formação, até aos que acompanharam o meu percurso pela FEUP e tornaram possível que eu esteja neste momento a completar o meu mestrado.

Por último, mas não menos importante, um agradecimento a todos os meus amigos que já trazia comigo antes de iniciar esta aventura pela faculdade, também eles me ensinaram e ainda ensinam e apoiam todos os dias. Aos meus companheiros de curso, amigos para a vida que ganhei, viva engenharia, viva a electrotecnia. Que o nosso espírito de companheirismo, a nossa entreajuda e amizade prevaleçam para sempre. Espero de hoje a muitos anos ainda ter jantares e reviver todos os momentos e memórias criadas ao longo do meu percurso, e criar outras novas.

Obrigado Faculdade de Engenharia da Universidade do Porto, foi um prazer.

António Pedro Freire Martins Magalhães Rocha

"I know that new situations can be intimidating.
You're looking around and it's all scary and different,
but you know... meeting them head-on, charging into them like a bull — that's how we grow as people."

Justin Roiland in Rick and Morty

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

LSTS	Laboratório de Sistemas e Tecnologia Subaquática
AUV	Veículo subaquático autónomo (Autonomous underwater vehicle)
TDMA	Acesso ao meio de divisão de tempo (Time division medium access)
ALN	Navegação alternativa por marcos (Alternative landmark navigation)
IDE	Ambiente de Desenvolvimento Integrado (Integrated Development Environment)
MATLAB	MATrix LABoratory

Terminologias e definições

Cooperativo (cooperar): 1. Agir com ou fornecer cooperação; ação de trabalhar ou laborar em grupo com outros para o mesmo objetivo; contribuir, colaborar ou assistir com.

[COOPERAR]. In: LEXICO, Dicionário Online de Português. Porto: 7Graus, 2018. Disponível em: [https://www.lexico.pt/cooperar/]. Acesso em: 20/09/2020.

Descentralizado (descentralização): 1. ato ou efeito de descentralizar; afastamento do centro; descentração 2. POLÍTICA processo político que visa a transferência de poderes e competências do poder central para o poder local 3. dispersão (de algo que estava concentrado).

descentralização in Dicionário infopédia da Língua Portuguesa [em linha]. Porto: Porto Editora, 2003-2020. [consult. 2020-09-20 23:14:43]. Disponível na Internet: https://www.infopedia.pt/dicionarios/lingua-portuguesa/descentralizaç~ao

Autónomo: 1. que se governa por leis próprias 2. independente; autossuficiente 3. (dispositivo, sistema) que funciona sem depender de ligação a outro dispositivo ou sistema.

autónomo in Dicionário infopédia da Língua Portuguesa [em linha]. Porto: Porto Editora, 2003-2020. [consult. 2020-09-20 23:17:14]. Disponível na Internet: https://www.infopedia.pt/dicionarios/lingua-portuguesa/autónomo

Automático: 1. Que é movido, que opera por meios puramente mecânicos. 2. [Figurado] Próprio de .autômato. 3. [Informática] Que permite .efetuar determinada tarefa sem recurso à intervenção humana. 4. Que não tem intervenção da consciência. = INCONSCIENTE 5. Diz-se dos movimentos do coração, do estômago, dos órgãos respiratórios, feitos sem influxo da vontade.

"automático", in Dicionário Priberam da Língua Portuguesa [em linha], 2008-2020, https://dicionario.priberam.org/autom%C3%A1tico [consultado em 20-09-2020].

Capítulo 1

Introdução

1.1 Enquadramento

Atualmente há certas tarefas que não são adequadas, ou mesmo possíveis, para ser executadas por humanos, como tal há uma necessidade de inovar com robôs e veículos autónomos que sejam capazes de realizar tais tarefas. Neste contexto há certas operações que são levadas a cabo por vários veículos autónomos a cooperar entre si, em rede, de modo a conseguir completar tais tarefas de forma eficaz e o mais eficiente possível.

Tais operações são levadas a cabo em ambientes sem estruturas de comunicação e com outros problemas como, por exemplo, o de pouca acessibilidade a energia e/ou combustível. Em particular, existe uma grande procura de soluções para operações de veículos autónomos em zonas remotas, como por exemplo nos oceanos [1]. Num artigo sobre "Veículos não tripulados para observação do oceano"[2], J. Borges de Sousa refere que a crescente evolução e utilização de veículos robóticos no estudo dos oceanos possa vir a revolucionar a observação dos oceanos e acrescenta ainda que isto passara por operações com diversos veículos a funcionar de forma coordenada, podendo até esses veículos serem submarinos, aéreos e espaciais e trabalhar em conjunto de forma contínua.

Decorre atualmente, portanto, uma intensa pesquisa na área de desenvolvimento de veículos autónomos de diversas características (aéreos, subaquáticos, etc.) adequadas ao ambiente em que vão operar e coordenação destes mesmos veículos a operar em rede. Em[3], P.F.J. Lermusiaux et al. realizaram experiências com *AUVs* em tempo real para prever e analisar os valores de parâmetros do ecossistema em que se inseriam. O artigo[4], de Trygve O. Fossum et al., mostra uma abordagem de mapeamento autónomo de biomassa de fitoplâncton usando uma rede de *AUVs*, na zona costeira do sul do mar da Noruega. Estes artigos comprovam as várias operações levadas a cabo por redes de veículos coordenados, e a importância que estes tem para obter informação, por exemplo, sobre a condição ecologia costeira

1.2 Motivação

O Laboratório de Sistemas e Tecnologia Subaquática é um laboratório de pesquisa que tem trabalhado na projeção, construção e operação de veículos subaquáticos, terrestres, aéreos e aéreos não tripulados e no desenvolvimento de ferramentas e tecnologias para a implantação de sistemas de veículos em rede.

Com as ferramentas de software desenvolvidas pelo *LSTS* é possível realizar o comando, controlo e facilitar as comunicações para operações com veículos, sistemas e operadores humanos em rede. Por exemplo, na campanha oceanográfica "Exploring Fronts with Multiple Robots"liderada por investigadores do *LSTS*[5], M. J. Costa et al. demonstraram uma abordagem à observação de oceanos com uma rede composta por diversos veículos de diferentes meios, submarinos, aquáticos de superfície e aéreos que são operados a partir de um navio oceanográfico, que identificavam e amostravam fenómenos dinâmicos do oceano. Com esta abordagem foi possível coordenar uma rede de veículos heterogéneos, com diferentes funcionalidades e característica, para trabalhar em conjunto e obter um objetivo em comum.

Noutro exemplo, num artigo[6], M. Thammawichai et al. apresentam uma proposta para melhorar a eficiência de missões em equipa de *AUVs*. Uma vez que normalmente estas missões consomem uma quantidade considerável de energia no processo de comunicação entre veículos, por comparação com a energia computacional despendida neste artigo é proposto um equilíbrio entre a energia computacional e a comunicação realizada entre veículos. Através da cooperação dos veículos, da utilização de esquemas de agrupamento e da incorporação de agregação de dados é possível otimizar o consumo de energia de computação e comunicação, o que aumenta a vida útil do sistema e permite realizar missões mais prolongadas.

É importante para o futuro pesquisar e perceber como devemos otimizar a construção de veículos autónomos para operações subaquáticas, ou outros ambientes remotos. É necessário também focar em desenvolver ferramentas e tecnologias que permitam controlar de forma eficiente estes veículos a operar em rede e que lhes confiram capacidade para comunicar entre eles em ambientes inóspitos, contornando assim as dificuldades devido, entre outros, à falta de estruturas de comunicação e difícil acesso a redes energéticas ou de combustível.

1.3 Objetivo do trabalho

O objetivo desta dissertação é estudar e definir estratégias de controlo descentralizadas que permitam efetuar a coordenação de múltiplos veículos autónomos que estão a operar em rede sem possuírem, no entanto, estruturas de comunicação entre si e que apresentam ainda problemas de acesso a energia e/ou combustível.

Para ilustrar os objectivos do trabalho consideramos o seguinte exemplo, que representa os resultados de um caso de estudo abordado no capítulo 7.

Exemplo 1.3.1 *Neste exemplo podemos visualizar um veículo (a verde), a operar numa rede, que está a coordenar a comunicação com os restantes elementos da rede. Este veículo funciona como*

um veículo mensageiro e é responsável por disseminar a informação, sobre a forma de mensagens, para os restantes componentes de forma a coordenar as suas operações. Como se pode ver na figura 1.1 o veículo está a deslocar-se entre duas bases (X) às quais estão alocados veículos (a vermelho) à espera de ordens para executar uma ação. À direita está exposta uma mensagem, que é uma estrutura que guarda a informação sobre as operações que os veículos vão executar. Esta mensagem está a ser transportada pelo veículo mensageiro.

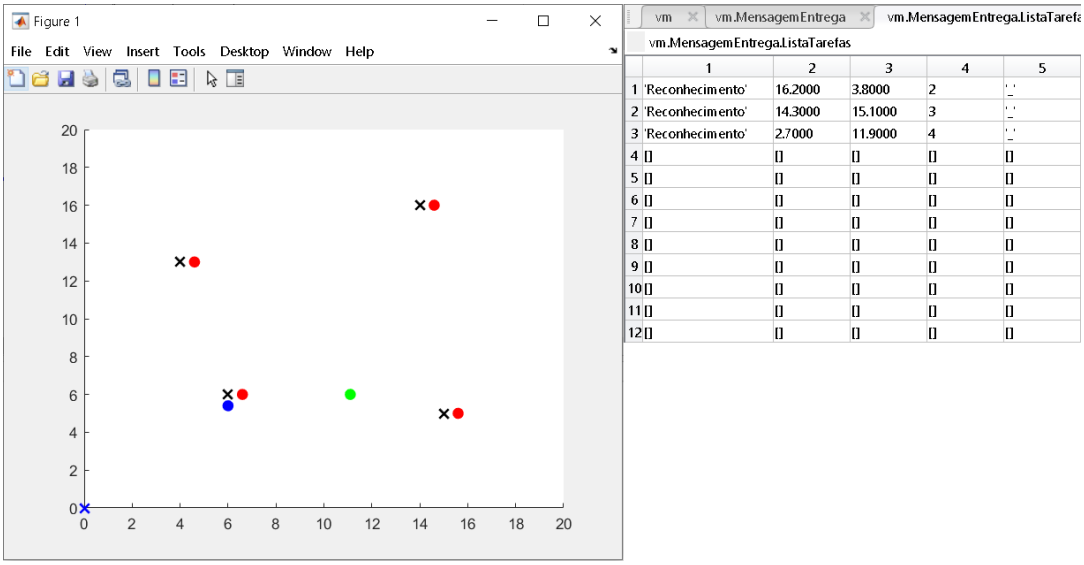


Figura 1.1: Veículo mensageiro a transportar tarefas para entregar em b2, b3 e b4.

Na seguinte figura 1.2 vemos a informação que foi disseminada pelas bases pelo veículo mensageiro quando este chegou ao alcance de estabelecer uma linha de comunicação com as mesmas. Esta informação enviada para as bases vai por sua vez gerar instruções para os veículos que vão depois despoletar outras ações.

Este exemplo é demonstrativo de uma rede com vários componentes a funcionar em conjunto. Demonstra também que é possível que estes veículos se coordenem de forma autónoma, sem intervenção exterior à rede, e a importância de definir mecanismos de comunicação para veículos que possam funcionar em locais remotos sem acesso a estruturas auxiliares para a comunicação.

b2_lt_espera

20x5 cell

	1	2	3	4	5
1	'Reconheci...	16.2000	3.8000	2	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

b3_lt_espera

20x5 cell

	1	2	3	4	5
1	'Reconheci...	14.3000	15.1000	3	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

b4_lt_espera

20x5 cell

	1	2	3	4	5
1	'Reconheci...	2.7000	11.9000	4	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

Figura 1.2: As bases b2, b3 e b4 com tarefas em lista de espera.

1.4 Abordagem

Na abordagem ao problema que vai ser considerada ao longo da dissertação considera-se um modelo de uma rede de vários veículos coordenados, da qual fazem parte os seguintes componentes: um operador, bases e veículos. Há ainda dois elementos na rede que não sendo considerados componentes físicos como os componentes anteriores são também importantes para o seu funcionamento: mensagens e tarefas.

- O operador corresponde a um utilizador externo à rede que introduz informação na rede
- As bases correspondem a locais físicos aos quais os veículos são alocados e que interagem e comunicam com eles.
- Os veículos são, como o próprio nome indica, veículos que se deslocam pela rede com diferentes funções, entre outras, transportar mensagens e executar tarefas. Estes veículos comunicam com as bases, sendo que apenas estabelecem uma linha de comunicação quando se encontram em proximidade com as mesmas.
- As mensagens são a forma de comunicação entre os componentes da rede e guardam informação para ser transportada e enviada a certos componentes.

- As tarefas referem-se a operações de diferentes tipos e com diferentes utilidades que alguns veículos da rede tem capacidade de executar.

1.5 Organização da tese

Este documento está estruturado em sete capítulos. No capítulo 2 vamos abordar os conceitos fundamentais que foram revistos para a realização desta dissertação. No capítulo 3 definimos o problema a ser tratado, onde abordamos as diferentes vertentes do problema e as suas premissas, definições e propriedades.

Em seguida, no capítulo 4, apresenta-se o estudo realizado acerca do estado de arte do problema e diferentes abordagens para resolver situações semelhantes. O capítulo 5 apresenta a abordagem a tomar para solucionar o problema, qual o modelo da rede e mecanismos de coordenação a desenvolver. O capítulo 6 descreve a implementação computacional realizada. O capítulo 7 expõe os resultados obtidos no ambiente de simulação, com uma posterior discussão dos mesmos. Por fim o capítulo 8 apresenta uma conclusão do trabalho realizado com as ilações a retirar e possíveis desenvolvimentos futuros.

Capítulo 2

Conceitos fundamentais

2.1 Organização

Neste capítulo vão ser expostos conceitos conhecidos, como são os diferentes autómatos cujo funcionamento vai ser de seguida explicado, com o intuito de melhor se compreender os mecanismos de coordenação, e como vai operar, a rede de veículos coordenados que se vai expor ao longo desta dissertação. O modelo dos autómatos abordados neste capítulo, baseado em *slides* [7] de uma aula de sistemas híbridos do professor João P. Hespanha, vai ser utilizado como base de funcionamento para diversos elementos desta rede, desde os veículos e as suas bases, até às mensagens trocadas na rede.

2.2 Autómato de estados finitos

Um autómato finito pode ser pensado como sendo uma máquina de estados finita. Um autómato finito é representado como um conjunto de estados, entradas e transições. O autómato tem um estado inicial onde se encontra quando é iniciado e à medida que o autómato vai lendo as entradas que lhe são introduzidas, se estas pertencerem ao conjunto de entrada, toma de acordo com estas entradas as transições correspondentes que o levam a alterar ou não o seu estado inicial.

O autómato finito é um sistema representado da seguinte forma: *Autómato finito* $= \{Q, \Sigma, \Phi\}$ em que:

- Q representa o conjunto finito de estados: $Q = \{Q\_1, Q\_2, Q\_3, ..., Q\_n\}$.
- Σ representa o conjunto finito de entradas: $\Sigma = \{a, b, c, d, ...\}$.
- Φ representa o conjunto das funções de transição: $\Phi = \{Q\ x\ \Sigma \rightarrow Q\}$.

Exemplo 2.2.1 *De acordo com o a representação gráfica da figura 2.1 este autómato apresenta um conjunto de entrada* q $\in Q = \{1, 2, 3\}$ *com o estado inicial* 1. *Com o seguinte conjunto de entradas* s $\in \Sigma = \{$a, b$\}$ *e respetivo conjunto de transições que levam a diferentes estados* Φ(q,s) $= \{1, 2, 3\}$.

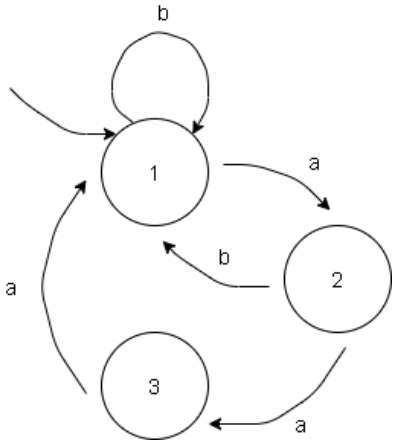


Figura 2.1: Representação gráfica de um autómato finito

Na seguinte tabela 2.1 podemos ver o fluxo dos estados e transições do autómato em função das entradas lidas.

Estado q ∈ Q	Entrada s ∈ Σ	Função de transição Φ(q,s)
1	a	2
1	b	1
2	a	3
2	b	1
3	a	1

Tabela 2.1: Estados, entradas lidas e respetivas transições de um autómato finito

2.3 Autómato híbrido

Um autómato híbrido é um sistema que modela a interação de processos contínuos com comportamentos discretos. É portanto possível pensar num autómato híbrido como uma máquina de estados finita, como o modelo acima apresentado do autómato finito, com um conjunto de variáveis contínuas.

O autómato híbrido é um sistema representado da seguinte forma: *Autómato híbrido* $= \{Q, \mathbb{R}, f, \Phi\}$ em que:

- Q representa o conjunto de estados discretos.
- $\mathbb{R}$ representa o conjunto estado-espaço contínuo.
- f representa o campo de vetores: $f = \{Q\ X\ \mathbb{R} \rightarrow \mathbb{R}\}$.
- Φ representa o conjunto de transições discretas: $\Phi = \{Q\ X\ \mathbb{R} \rightarrow Q\}$.

Exemplo 2.3.1 *Na seguinte figura apresenta-se uma representação gráfica de um exemplo específico de um autómato híbrido.*

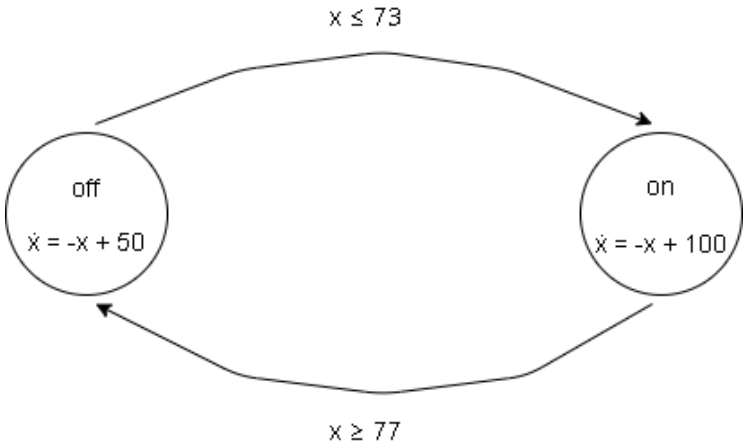


Figura 2.2: Representação gráfica de um autómato híbrido

O exemplo graficamente representado refere-se a um sistema de um termostato que regula a temperatura de uma sala. O termostato avalia continuamente a temperatura da sala, que aquece e arrefece naturalmente com o passar do tempo e diversas condições externas. Quando a temperatura baixa para além de um limite estabelecido o termostato liga um aquecedor para aumentar a temperatura da sala. Quando a temperatura da sala sobe para além de um limite previamente estabelecido o termostato desliga o aquecedor. Assim é possível regular a temperatura da sala interagindo um processo contínuo de avaliação da temperatura com comportamentos discretos como ligar e desligar o aquecedor.

De acordo com a figura 2.2 este autómato híbrido apresenta um conjunto de estados discretos $\mathrm{q} \in Q = \{\mathrm{off}, \mathrm{on}\}$.

O seguinte campo de vetores:

$$\mathrm{f(q,x)} = \begin{cases} -x+50 & q = off \\ -x+100 & q = on \end{cases} \tag{2.1}$$

E ainda, o seguinte conjunto de transições discretas entre estados:

$$\Phi\mathrm{(q,x)} = \begin{cases} on, & q = off, x \leq 73 \\ off, & q = off, x > 73 \\ off, & q = on, x \geq 77 \\ on, & q = on, x < 77 \end{cases} \tag{2.2}$$

2.4 Autómato aberto

Com um autómato aberto é possível enviar uma mensagem sob a forma de um evento para outro autómato de modo a coordenar esse mesmo autómato. Sendo assim é possível, por exemplo,

desconstruir um grande autómato em diversos autómatos cada um destinado a controlar uma tarefa específica sendo que os eventos de uns autómatos são utilizados em coordenação para controlar outros autómatos, como se pode ver pelo seguinte exemplo.

Exemplo 2.4.1 *Na figura 2.3 é possível observar uma representação gráfica de um autómato que controla um tanque de água. Este sistema do tanque de água é composto pelo próprio tanque, uma bomba de entrada de água, e uma torneira para saída de água. O* λ *representa o caudal de água que entra no tanque, enquanto que o* μ *representa o caudal de saída de água do tanque.*

Este sistema inicialmente tem água no tanque e está a sair um certo caudal de água μ *pela torneira. Quando o nível da água passa o limite mínimo de água no tanque o sistema manda ligar a bomba que fica ligada a encher o tanque com um certo caudal* λ *até que a água dentro do tanque chegue a um nível máximo e depois volta a desligar-se a bomba.*

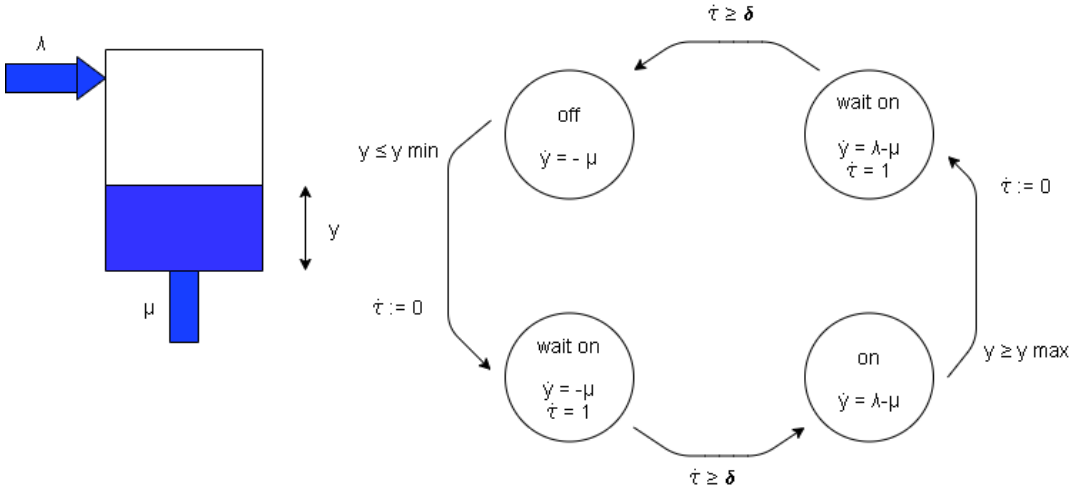


Figura 2.3: Autómato de controlo de um tanque de água

Com o conceito de autómato aberto é possível simplificar o autómato acima graficamente representado da seguinte forma, que se pode observar na seguinte figura 2.4.

Como é possível de se observar, nesta representação gráfica, temos dois autómatos abertos que dependem um do outro. O autómato da direita gera o evento change *que vai controlar quando a bomba de água abre ou fecha no autómato da esquerda, ao ativar as transições. O autómato da esquerda gera o evento* ask *que vai ativar o* timer *no autómato da direita, para controlar o tempo que demora a trocar o estado da bomba. Enquanto a bomba está desligada perguntamos se a água já desceu abaixo do nível mínimo. Quando este evento acontece (*ask*) o autómato da direita passa para o estado de* delay *e passado um certo tempo previamente definido volta ao estado* idle *gerando o evento* change*. Este evento (*change*) faz com que a bomba se volte a ligar e volta a gerar um evento* ask *para verificar se o nível da água já passou o nível máximo. Volta a ativar o* timer *no autómato da direita para, passado o tempo definido, desligar a bomba novamente.*

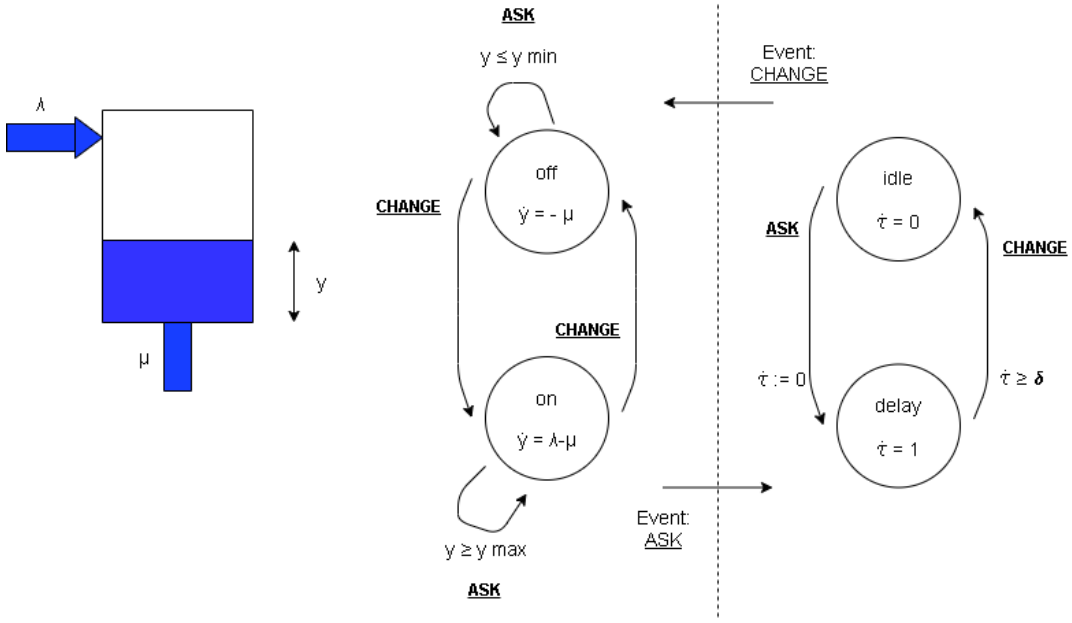


Figura 2.4: Autómatos abertos de controlo de um tanque de água

Capítulo 3

Definição do problema

3.1 Introdução

O objetivo principal desta dissertação passa por projetar uma rede de veículos autónomos a operar em zonas sem estruturas de comunicação. Neste capítulo, como o nome indica, vai ser definido o problema em questão a ser analisado. Inicialmente vai ser descrito o ambiente de trabalho e as diversas características desta rede coordenada de veículos, para depois definirmos algumas premissas que são uma base de funcionamento desta rede. Em seguido vai ser exposto o problema que se vai abordar ao longo desta dissertação e as diferentes propriedades que o sistema deve satisfazer de modo a permitir o funcionamento otimizado da rede de veículos coordenados a ser analisada.

3.2 Definições

Na seguinte figura 3.1 podemos ver uma representação da rede de veículos. Importa realçar que neste exemplo está representada uma rede a coordenar 5 veículos diferentes, com 4 nós fixos, no entanto é possível escalar esta rede com a adição de mais nós e/ou mais veículos.

Na figura 3.1 as bases (1,2 e 3) referem-se a nós fixos, com uma localização específica. Estas bases vão funcionar como locais aos quais vamos alocar os diferentes veículos da rede. Uma vez que estas bases tem capacidades de comunicação vão ser elas a passar a informação aos veículos para executar as tarefas definidas para a rede.

A base remota representa o local, distanciado da zona de operação da rede, onde se vai definir a lista de tarefas a serem executadas pela rede. Nesta base há uma pessoa, o operador, que é responsável por gerar as tarefas que se vão executar na rede. Existe ainda um veículo alocado a esta base remota (veículo da base remota) que é responsável por levar esta informação à base da rede mais próxima e voltar à base remota.

Uma tarefa refere-se a uma operação a executar pelos veículos da rede que são coordenados para otimizar a utilização de recursos da rede e concluir o objetivo final dessa mesma tarefa. Cada tarefa tem as suas especificações, isto é, dependendo do que se quer alcançar com a rede de

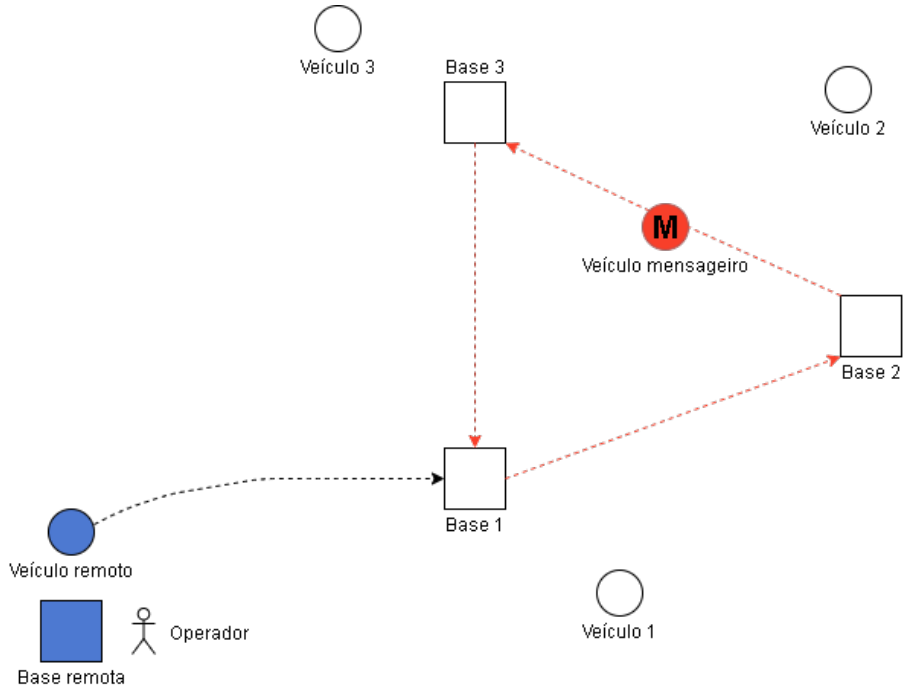


Figura 3.1: Diagrama exemplo da rede de veículos autónomos sem estruturas de comunicação

veículos envia-se uma tarefa que faça com que os veículos atuem e/ou recolham a informação que se pretende.

Para que a rede execute as diferentes tarefas, que diferem nas suas especificações, é então necessário passar as especificações corretas aos veículos para eles poderem executar as respetivas tarefas. Os exemplos de tarefas possíveis de executar na rede são:

- Tarefa de mapeamento. O objetivo desta tarefa é que o(s) veículo(s) da rede percorram uma certa área e recolham informação sobre o local que percorreram.

- Tarefa de reconhecimento. O objetivo desta tarefa é que o(s) veículo(s) da rede se desloquem até um ponto definido pelo operador e fiquem aí por um tempo predeterminado a retirar informação acerca do local.

- Tarefa de patrulha. O objetivo desta tarefa é que o(s) veículos(s) da rede circulem em torno de um ponto de interesse, durante um tempo a ser definido, para verificar se há algum intruso a aproximar-se de um certo local e comunicar à rede caso isso aconteça.

Para melhor percebermos uma tarefa vamos modelar, por exemplo, a tarefa de patrulha de acordo com as variáveis a passar aos veículos da rede.

Tarefa de patrulha: $Patrulha = \{x, y, tempo\}$ em que:

- X e Y: representam as coordenadas do ponto de interesse a ser patrulhado.

- Tempo: representa o tempo de execução da tarefa durante o qual o veículo deve deslocar-se em torno do ponto de interesse.

Para ser possível coordenar a rede e a execução das tarefas por parte dos veículos é necessário partilhar informação. Cada veículo deve saber as especificações da tarefa que a rede tem de executar. Uma vez que o intuito é operar em zonas sem estruturas de comunicação esta partilha de informação ocorre sobre a forma de trocas de mensagens, que acontecem localmente quando a mensagem é deixada por um veículo numa base para os veículos que estão alocados a essa mesma base terem acesso à informação.

Os veículos (1,2 e 3) bem como o veículo remoto e o veículo mensageiro representam o conjunto de veículos da rede com capacidade de comunicação e execução de tarefas. Este veículos têm funções diferentes:

- Veículos 1,2 e 3 – executar tarefas definidas pela rede;
- Veículo da base remota – como já foi referido, este veículo é responsável por introduzir na rede a lista de tarefas definida pela estação remota e recolher a lista de tarefas finalizadas;
- Veículo mensageiro – refere-se ao veículo que vai transportar informações e transmiti-las às diferentes bases da rede;

3.3 Premissas

Para definirmos concretamente a rede de veículos com que vamos trabalhar necessitamos de definir as premissas que caracterizam o funcionamento desta mesma rede para depois abordar o problema. Inicialmente definimos esta rede como um conjunto de diversos veículos e bases, e como tal vamos abordar as especificações de cada um destes elementos.

As bases representam um conjunto de nós fixos em que:

B1 As suas localizações são conhecidas e inalteráveis.

B2 Tem capacidade de comunicar com os veículos da rede. Podem enviar e receber informação.

B3 Esta capacidade de comunicação está limitada até um certo alcance à volta de cada base, que vai até uma distância máxima com um raio predefinido.

B4 Cada base tem um relógio interno que indica a data e hora. Consideramos que este relógio está sempre atualizado e sincronizado com os relógios do resto dos elementos da rede.

Existe ainda um conjunto de veículos na rede com as seguintes características:

V1 Cada veículo da rede está sempre alocado a uma base especificada para ele, com o veículo mensageiro como única exceção.

V2 Têm capacidade de comunicar com as bases da rede. Podem enviar e receber informação.

V3 Esta capacidade de comunicação está limitada até um certo alcance à volta de cada um dos veículos da rede, que vai até uma distância máxima com um raio predefinido.

V4 Têm capacidade de executar tarefas especificadas de acordo com as instruções que recebem das bases.

V5 Cada veículo apenas tem capacidade para executar uma tarefa de cada vez.

V6 Cada veículo tem um relógio interno que indica a data e hora. Consideramos que este relógio está sempre atualizado e sincronizado com os relógios do resto dos elementos da rede.

V7 Consideramos que os veículos têm uma fonte de combustível infinita.

Para além das características e premissas de cada um dos elementos integrantes da rede de veículos temos ainda especificações de funcionamento da própria rede como um todo, sendo essas as seguintes:

R1 A comunicação é feita sem qualquer estrutura de suporte, como tal, esta comunicação entre veículos e bases é feita apenas presencialmente quando estes se encontram.

R2 A comunicação tem um limite de alcance restrito, devido ao limite do alcance dos veículos.

R3 A área de trabalho da rede depende da localização e distanciamento de cada base e ainda do alcance de cada veículo que vai executar as tarefas.

R4 Não há erros de execução, ou seja, não existem estados de erro nem transições para ditos estados. Assumimos portanto o funcionamento ideal em toda a rede bem como de cada veículo e base que a constitui.

R5 Nenhum veículo colide com outros componentes da rede, outros veículos ou bases.

R6 As tarefas que tenham que ser executadas, ainda que apenas parcialmente, fora da área de trabalho da rede não são atribuídas à rede para ser executadas.

3.4 Problema geral

Coordenar uma rede de múltiplos veículos autónomos a funcionar em ambientes sem estruturas de comunicação e com problemas de acesso a combustível.

Este problema geral organiza-se nos seguintes subproblemas.

1. Disseminação da informação a passar pela rede.

   Como se vai garantir que a informação chega a todos os elementos da rede, que não há informação obsoleta, e que informação vai ser trocada.

2. Controlo da execução de tarefas pelos veículos.

   Como se vai decidir que veículo é o indicado para executar a tarefa. Que informações tem este veículo para tomar essa decisão.

3. Alocação de tarefas a veículos.

   Como se vai equilibrar a distribuição de veículos pela rede para balancear a carga de trabalho.

3.5 Propriedades que o sistema deve satisfazer

O sistema deve satisfazer as seguintes propriedades.

1. Nenhum veículo fica sem instruções ou,colocado de outra maneira, o estado de movimento do veículo está sempre bem definido.
2. Nenhum veículo bloqueia.
3. Mensagens enviadas pela base remota chegam às restantes bases da rede num período de tempo limitado.
4. Mensagens trocadas entre bases e veículo são entregues durante um intervalo de tempo máximo conhecido.
5. O sistema é escalável no que refere ao número de veículos e também de bases.

Capítulo 4

Trabalho relacionado

4.1 Introdução

Para compreender melhor o problema que vai ser abordado na dissertação e também para se obter uma proposta de solução viável e eficaz para esse mesmo problema iniciou-se um processo de pesquisa e recolha de informação quanto ao estado de arte do problema endereçado na dissertação. Como já foi brevemente referido o problema consiste em coordenar múltiplos veículos autónomos a operar em rede sem possuírem, no entanto, estruturas de comunicação viáveis entre si e que podem ainda apresentar problemas de acesso a combustível. Vai-se, portanto, analisar a abordagem de outras pessoas a este mesmo problema para perceber de que forma é possível coordenar estes veículos a operar em rede, que estratégias de controlo podem ser aplicadas na rede e como aplicar uma estrutura de comunicação eficiente entre os veículos.

Contudo, esta é uma área em que as experiências de campo, bem assim como os resultados que as possam suportar, escasseiam. Desta forma, a pesquisa de trabalho relacionado fica, por isso, algo limitada. Mais ainda, quando as limitações de tempo associadas ao desenvolvimento deste trabalho, colocaram restrições adicionais a esta pesquisa. Optou-se, por isso, pela discussão de alguns trabalhos considerados relevantes, uma vez que endereçam problemas e soluções em áreas afins.

4.2 Coordenação de veículos marítimos não tripulados em sistemas com limitações de comunicações

Em 2013, numa conferência em San Diego, foi apresentado, por Delande et al.[8], um problema relacionado com a coordenação de uma rede de *AUVs* e como contornar os problemas devido às dificuldades de comunicação em meios subaquáticos. Aqui começou por se abordar a tarefa a executar como um conjunto de sub tarefas para atribuir a cada *AUV* para realizar paralelamente várias sub tarefas e aumentar a eficiência do sistema de *AUVs* a operar em rede.

Para se coordenar o funcionamento de cada veículo é necessário partilhar a informação obtida por cada um dos veículos por toda a rede. Uma vez que cada *AUV* esteja ciente das tarefas a

executar, tanto dele próprio como do resto dos veículos da rede, é possível realocar as diferentes tarefas a cada um dos *AUVs* mais adequado à sua execução.

A rede de *AUVs* partilha regularmente mensagens através de um cronograma *TDMA* que consiste numa lista ordenada de intervalos de tempo em que apenas um *AUV* é responsável pela transmissão das mensagens. Um veículo anuncia a missão ao transmitir a informação para toda a rede e depois todos os *AUVs* partilham a informação sobre a sua localização e nível de energia para depois o primeiro veículo que anunciou a missão poder otimizar toda a rede e definir que veículos vão executar a tarefa em função das restrições de tempo, energia, entre outros.

Esta abordagem mostrou ser um método promissor para contornar os problemas devido à dificuldade de comunicação em meios subaquáticos.

Em 2017, Kemna et al.[9] abordaram um problema de coordenação de uma rede de *AUVs* num ambiente com restrições de comunicação. O objetivo era colocar os *AUVs* a mapear uma área e optaram por uma estrutura descentralizada para realizar a coordenação dos vários veículos, uma vez que esta aumenta a robustez do sistema e permite melhorar a performance de cada veículo individualmente.

Devido ao ambiente em que os *AUVs* vão operar e as dificuldades de comunicação conhecidas pretende-se que a quantidade de comunicação se mantenha no mínimo possível, até porque não podemos assumir um canal de comunicação muito fiável. Foi, portanto, definido que as comunicações debaixo de água iam ser poucas e simples, por exemplo para evitar colisões, e por vezes os veículos vinham à superfície para poder comunicar melhor grandes quantidades de informação importante.

A estrutura de coordenação foi descentralizada ao gerar o próprio modelo do ambiente para cada *AUV*, sendo que cada um usa esse próprio modelo para tomar as decisões quanto às tarefas que vai executar. Para garantir que não há sobreposição de ações e execução de tarefas por parte de cada um dos veículos foi utilizado o particionamento dinâmico de Voronoi que coordena efetivamente as ações entre os diferentes veículos.

Quando os *AUVs* vêm à superfície podem então partilhar informação obtida e combinar isso com novos cálculos de particionamento de Voronoi após partilha de informação para garantir a coordenação descentralizada dos vários veículos a operar em rede.

Numa abordagem diferente a um problema semelhante, neste caso para o mapeamento do fundo oceânico, em 2015 Matsuda et al.[10] mais tarde e mais desenvolvido em 2018[11] relataram os resultados de experiências no mar a mapear extensas áreas de superfície marítima com a utilização de vários *AUVs*.

Para realizarem o mapeamento do fundo oceânico foi proposto um método chamado *ALN*, este método consiste numa rede de vários *AUVs* em que uns *AUVs* pousam no fundo do mar e passam a ser um ponto de referência para os restantes *AUVs* da rede, sendo que este papel de *AUV* de referência pode ir alternando entre os diferentes veículos da rede.

Neste conceito de *ALN* partimos do pressuposto que os *AUVs* têm sensores que medem, entre outros, profundidade, distância e direção relativas entre *AUVs*. Os *AUVs* móveis podem então estimar a sua posição de modo altamente preciso de acordo com o posicionamento de referência

do *AUV* que se encontra estático no solo. Os *AUVs* móveis realizam a missão de observação do fundo oceânico e sempre que um *AUV* vai pousar no fundo do mar para passar a servir de referência faz uma estimativa precisa da sua posição para posteriormente calibrar o cálculo do posicionamento dos outros *AUVs* em relação a este novo *AUV* de referência.

Este método é característico por apenas os *AUVs* conseguirem executar a missão de forma autónoma, sem intervenções externas à rede. Isto acarreta as seguintes vantagens:

- O apoio da superfície não é necessário se tivermos as primeiras posições de pouso dos *AUVs* localizadas.
- O alcance da observação não é limitado, pois uns *AUVs* funcionam de referência para outros antes de atingir o limite do alcance.
- Os *AUVs* podem navegar durante períodos longos de tempo diminuindo os erros de estimativa devido à calibragem com o *AUV* de referência.
- Basta um *AUV* de referência para vários *AUVs* móveis poderem executar a tarefa de observação do fundo oceânico.

Também existem, no entanto, algumas desvantagens ao utilizar este método:

- Pode ocorrer um erro de posição no *AUV* que funciona de referência que influencia a rede toda.
- Pelo menos um *AUV* vai ser utilizado apenas como um marco e ficar estacionário no fundo do mar sem executar a função de observação do fundo oceânico.

Em 2017, Abad et al.[12], implementaram um algoritmo de controlo preditivo de modelo descentralizado para controlar uma rede de *AUVs*.

Nesta abordagem foi assumida um função objetivo em que a sua formação e otimização são o tema central do modelo descentralizado de controlo preditivo. Esta função objetivo é dividida em três funções sub objetivos que permitam a operação de uma rede de *AUVs* colaborativa:

- Objetivo *waypoint* (ponto final do trajeto);
- Objetivo de prevenção de colisões;
- Objetivo de comunicação;

Para os veículos em rede conseguirem manter a comunicação é necessário manter os veículos dentro de um determinado raio dos outros veículos da rede. Isto dependo do sinal dos modems acústicos que são usados para manter as linhas de comunicação. Portanto é necessário que nenhum dos *AUVs* da rede saia fora do raio máximo estipulado para manter uma linha de comunicação entre toda a rede de *AUVs*s.

Essa linha de comunicação permite, portanto que se passe informação desde uma ponta à outra da rede dependendo da necessidade e conveniência da informação que se tem de transmitir.

Capítulo 5

Abordagem

5.1 Acerca de modelos

A abordagem aos modelos a desenvolver neste trabalho será baseada em elementos da teoria dos autómatos híbridos, que foi apresentada no capítulo 2, uma vez que o problema endereçado inclui sistemas com dinâmicas discretas e contínuas.

O modelo básico dum autómato não captura a componente de interações dinâmicas entre vários autómatos, que se encontra presente neste problema. Para o efeito, e tendo em vista acomodar esta necessidade de modelização, estendeu-se o modelo do autómato para incluir variáveis e também ser especializado para o envio e recepção de mensagens, conforme indicado nas secções seguintes. Sendo assim recorreu-se a variáveis booleanas, que são utilizadas como sensores de proximidade, e ainda variáveis com uma estrutura que lhes permita guardar o conteúdo das mensagens trocadas pelos diferentes componentes da rede.

Este aspecto de desenvolvimento de modelo aproxima-se, por isso, dos elementos do modelo de Atores [13], que foi inicialmente desenvolvido nos anos 70. Este é um modelo matemático de computação concorrente que trata o ator como a primitiva universal da computação concorrente. Os atores trocam mensagens e respondem a mensagens de várias formas: tomando decisões locais, criando atores, enviando mensagens e determinando como responder à próxima mensagem. Os atores podem modificar o seu estado interno, mas só podem modificar outros atores indiretamente através do envio de mensagens. É este o espírito da abordagem proposta que, para o efeito, estende o modelo de autómato híbrido apresentado no capítulo 2.

5.2 Modelo de autómato híbrido estendido

Um autómato A é descrito por:
$A = \{Estados\ discretos, Variáveis\ internas, Sistema\ de\ transição\ de\ estados, Mensagens\}$.

Para facilitar a compreensão da descrição de um autómato A o *Sistema de transição de estados* vai ser sempre representado por uma tabela com uma estrutura semelhante à seguinte tabela

exemplo 5.1.

Estado inicial	Estado seguinte	Condições de transição	Ações
-	-	-	-
-	-	-	-

Tabela 5.1: Tabela exemplo de transição de estados dos diversos componentes da rede.

As tabelas de transição de estados tem todas uma semântica de execução associada que é a seguinte:

- A transição ocorre quando todas as condições de transição estão ativas, ou seja as condições têm valor 1 ou *true*.
- Após ocorrer uma transição cada uma das ações da lista de ações ocorre pela ordem apresentada na tabela, isto é, executa-se uma linha de código das ações de cada vez e não todas em simultâneo.

5.3 Componentes da rede

Para coordenar toda a rede de modo a que esta opere de forma sincronizada, garantindo que todos os componentes trabalham em prol de um mesmo objetivo final e com um funcionamento o mais otimizado possível importa iniciar a abordagem ao problema modelando cada um dos componentes integrantes da rede. Foi precisamente isso que foi feito inicialmente. Ao realizar a modelação de cada um dos diferentes componentes da rede obtemos um sistema que os representa. Pode-se então, posteriormente, organizar a coordenação da rede controlando as interações entre sistemas e ainda dentro de cada sistema.

Uma parte fulcral da coordenação passa pela disseminação de mensagens por toda a rede de modo a partilhar a informação por todos os elementos da rede, pelo que o meio de comunicação com todas as bases e veículos da rede vai ser precisamente a passagem de mensagens. Estas mensagens são todas devidamente identificadas e são de diversos tipos, dependendo da informação que se pretende passar. A estrutura de uma mensagem será sempre $Mensagem = \{ID, Tipo\ de\ tarefa, Argumentos\}$.

A rede de componentes é assim definida por um conjunto de autómatos, sendo que cada autómato representa cada um dos componentes da rede. Estes componentes trocam mensagens para efeitos de coordenação das sua actividades. A rede em consideração é um domínio plano em que existem *Sistemas*, dados por:
$Sistemas = \{Operador, Base\ remota, Veículo\ da\ base\ remota, Bases, Veículos, Veículo\ mensageiro\}$
em que:

Operador Representa um operador humano que envia mensagens para a base remota.

Base remota Representa a base que se encontra numa localização acessível ao operador e através da qual se comunica com os restantes componentes da rede.

Veículo da b.r. Representa um veículo que tem como função única estabelecer uma linha de comunicação entre a base remota e a base da rede mais próxima dessa mesma base remota, ou seja, b_1.

Bases É o conjunto de n bases $\{b_1, \ldots, b_n\}$ em que cada base representa um local onde é possível alocar veículos da rede para lhes atribuir tarefas e tem capacidade de guardar mensagens e comunicar.

Veículos É o conjunto de m veículos que comunicam com as bases e executam tarefas $\{v_1, \ldots, v_m\}$.

Veículo mensageiro É o conjunto de s veículos mensageiros $\{m_1, \ldots, m_s\}$ que comunicam com as diversas bases da rede, excetuando a base remota. No que se segue, considera-se que $s = 1$.

Por opção de projecto, a interacção de b_1 com todos os elementos da rede é diferente das interacções levadas a cabo pelas restantes bases. Tem, por isso, um modelo diferente das outras bases.

A dinâmica de movimento dos *Sistemas* que se movem é dada pela equação de movimento de uma partícula que se pode deslocar em qualquer direcção com uma velocidade fixa cujo módulo toma valores no conjunto $\{0, v_{max}\}$.

A dinâmica de movimento de todos os veículos é abstraída por duas manobras que se descrevem de seguida:

Goto(x,y) O veículo segue em linha recta da posição actual para o destino, cuja localização é dada pelo par (x, y), à velocidade v_{max}.

Hold(x,y) O veículo mantém-se imobilizado no local cujas coordenadas são dadas pelo (x, y)

Desta forma, e na perspectiva de controlo de movimento, estes veículos aceitam um comando *Goto(x,y)* e *Hold(x,y)*, que normalmente estão associados ao envio ou receção de uma mensagem.

5.4 Modelos dos componentes

Segue uma descrição dos modelos de todos os componentes desta rede. Estes modelos incorporam já a lógica de coordenação que tem por vista implementar as especificações definidas no capítulo 3.

5.4.1 Operador

O operador está associado à base remota, para a qual envia mensagens $mensagem\_opr(x)$, especificando listas de tarefas x a serem executadas pelo sistema. A lista de tarefas que o sistema pode executar está representada na seguinte tabela 5.2.

Tarefa	Argumentos
Mapeamento	{[x1, y1], [X2, Y2], área de trabalho}
Patrulha	{x, y, área de trabalho, tempo}
Reconhecimento	{x, y, área de trabalho}

Tabela 5.2: Tipos de mensagens e respetivos argumentos a ser passados.

Assume-se que o operador está permanentemente ligado à base remota. A certos momentos, definidos por um tempo de forma cíclica, é pedido o *input* ao operador para este poder enviar mensagens para a b_r.

5.4.2 Base remota

Primeiramente foi feita a modelação do sistema da *Base remota* cujo funcionamento se descreve de seguida. Na seguinte figura 5.1 está representado o diagrama de transição de estados associado.

Esta base vai interagir com o *operador* e ainda com um veículo ao qual se refere como *veículo da base remota*. Sendo assim, este operador humano, bem como o veículo, vão interferir no funcionamento desta base remota através da troca de mensagens.

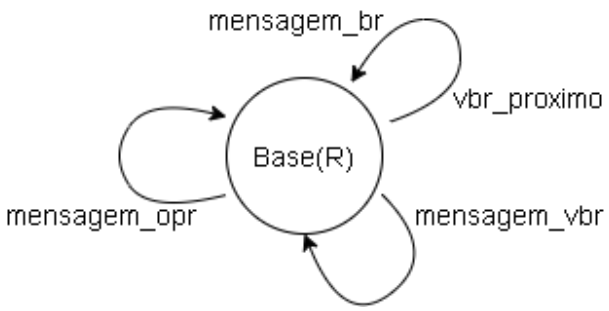


Figura 5.1: Lógica de funcionamento da base remota.

O sistema da *Base remota* tem apenas um estado discreto, que é também o estado inicial. Este sistema possui quatro variáveis internas. As transições são despoletadas pela chegada de mensagens ou pela alteração de valores das variáveis internas. Ao processo de transição de estado podem estar associadas também a geração de mensagens.

$Base\ remota = \{Estados\ discretos, Variáveis\ internas, Sistema\ de\ transição\ de\ estados, Mensagens\}$ em que:

1. $Estados\ discretos = \{Base(R)\}$ que inclui apenas o estado inicial.
2. $Variáveis\ internas = \{lt\_recebidas, lt\_enviadas, lt\_concluidas, vbr\_proximo\}$ em que:

- *lt_recebidas* guarda a lista de tarefas inseridas no sistema pelo operador.
- *lt_enviadas* guarda a lista de tarefas enviadas pela base remota para o veículo da base remota.
- *lt_concluidas* guarda a lista com todas as tarefas que já foram executadas pela rede.
- *vbr_proximo* representa uma variável booleana que está ativa se o veiculo da base remota estiver próximo da base remota, ao alcance da troca de mensagens entre ambos, e está inativa se isto não se verificar.

3. *Sistema de transição de estados* inclui três transições. A tabela seguinte 5.3 apresenta as transições de estados da *base remota*.

Estado inicial	Estado seguinte	Condições de transição	Ações
Base(R)	Base(R)	mensagem_opr;	lt_recebidas += mensagem_opr;
Base(R)	Base(R)	vbr_proximo; lt_recebidas $\neq \{\}$;	lt_enviadas = lt_recebidas; lt_recebidas = {}; mensagem_br{lt_enviadas};
Base(R)	Base(R)	mensagem_vbr;	lt_concluidas += mensagem_vbr; lt_enviadas -= mensagem_vbr;

Tabela 5.3: Transições do sistema base remota e as suas respetivas condições de transição e ações.

4. As *Mensagens* são eventos que ocorrem e referem-se ao conjunto de mensagens que podem ser trocadas pela base remota com outros sistemas. Podem ser enviadas ou recebidas pelo sistema da base remota. $Mensagens = \{mensagem\_br, mensagem\_opr, mensagem\_vbr\}$.

   As mensagens enviadas pela base remota designam-se como *mensagem_br*, que servem para passar uma lista de tarefas.

   As mensagens a ser recebidas pela base remota podem ser provenientes de um operador, contendo uma lista de tarefas, e são designadas como *mensagem_opr*, ou ser provenientes do veículo da base remota, e são designadas como *mensagem_vbr*.

5.4.3 Veículo da base remota

O sistema do *Veículo da base remota* foi modelado de acordo com a figura 5.2. De notar que este sistema interage com a *Base Remota* e ainda com b_1. Apresenta-se de seguida o diagrama de transição de estados.

O sistema do *Veículo da base remota* tem três estados discretos, sendo que um deles é o estado inicial. Possui também três variáveis internas. Possui também várias transições de estados que podem ser despoletadas e também despoletar elas alterações nos valores das variáveis internas ou geração de mensagens.

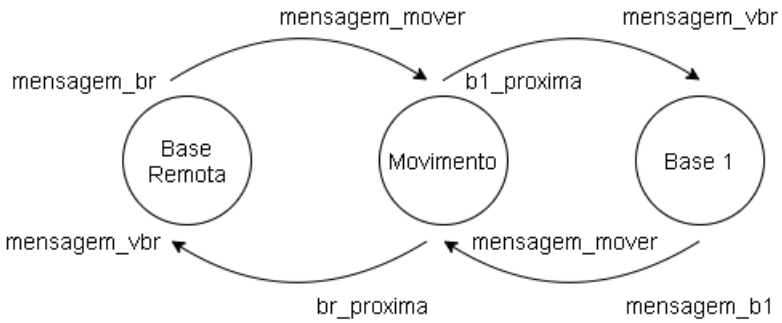


Figura 5.2: Lógica de funcionamento do veículo da base remota.

Veículo da base remota = {*Estados discretos*, *Estado inicial*, *Variáveis internas*, *Sistema de transição de estados*, *Mensag*
em que:

1. *Estados discretos* = {*Base remota*, *Movimento*, *Base 1*}.
2. *Estado inicial* = {*Base remota*} quando o veículo se encontra na base remota à espera de instruções.
3. *Variáveis internas* = {*lt_transporte*, *br_proxima*, *b1_proxima*} em que:
   - A variável *lt_transporte* guarda a lista de tarefas que vão ser transportadas pelo veículo da base remota entre a própria base remota e a base 1.
   - A variável *b1_proxima* representa uma variável booleana que está ativa se o veiculo da base remota tiver chegado à proximidade da base 1, ao alcance da troca de mensagens entre ambos, e está inativa se isto não se verificar.
   - A variável *br_proxima* representa uma variável booleana que está ativa se o veiculo da base remota tiver chegado à proximidade da base remota, ao alcance da troca de mensagens entre ambos, e está inativa se isto não se verificar.
4. *Sistema de transição de estados* inclui quatro transições e está representado na seguinte tabela 5.4.

Estado inicial	Estado seguinte	Condições de transição	Ações
Base remota	Movimento	mensagem_br;	lt_transporte = mensagem_br; mensagem_mover{b1};
Movimento	Base 1	b1_proxima;	mensagem_vbr{lt_transporte}; lt_transporte = {};
Base 1	Movimento	mensagem_b1;	lt_transporte = mensagem_b1; mensagem_mover{br};
Movimento	Base remota	br_proxima;	mensagem_vbr{lt_transporte}; lt_transporte = {};

Tabela 5.4: Transições do sistema do veículo da base remota e as suas respetivas condições de transição e ações.

5. $Mensagens = \{mensagem\_vbr, mensagem\_br, mensagem\_b1, mensagem\_mover\}$.

   As mensagens enviadas pelo sistema do veículo da base remota com uma lista de tarefas são designadas como *mensagem_vbr*. O veículo da base remota pode ainda enviar uma mensagem com coordenadas para se mover, que se designa como *mensagem_mover*.

   As mensagens a ser recebidas podem ser provenientes da base remota, e designam-se por *mensagem_br*, ou podem ser provenientes da base 1 e designam-se por *mensagem_b1*.

5.4.4 Bases

De seguida foi feita a modelação do sistema das bases da rede. Estas bases vão interagir com diferentes elementos da rede como os *Veículos* que executam as tarefas, o *Veículo mensageiro* e ainda o *Veículo da base remota*.

Para a modelação das bases, apesar destas serem todas semelhantes, temos algumas especificações diferentes na b_1 em relação às restantes, sendo que esta se refere à base mais próxima da base remota. Esta b_1 é a única que recebe a lista de tarefas vinda da base remota através do veículo da base remota que faz a ligação entre estas duas bases. A lista final de tarefas concluídas por todos os veículos vai também ser enviado para a base remota através da b_1.

É possível ver nas seguintes figuras 5.3 e 5.4 o diagrama de transição de estados respetivamente da b_1 e das restantes bases da rede as quais são referidas como *Base(i)* sendo que o *i* indica o índice de cada base.

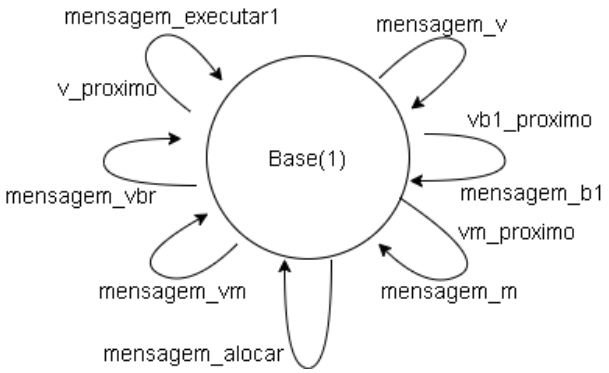


Figura 5.3: Lógica de funcionamento da b_1 da rede.

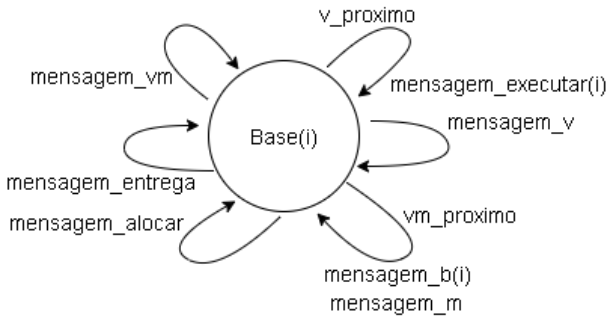


Figura 5.4: Lógica de funcionamento de uma *Base(i)* da rede.

O sistema de qualquer uma das bases da rede apresenta apenas um estado discreto sendo este portanto o estado inicial. Possui sete variáveis internas. O sistema de transição de estados é regulado pela receção ou envio de mensagens e ainda alterações nos valores de variáveis internas.

$Base = \{Estados\ discretos, Variáveis\ internas, Sistema\ de\ transição\ de\ estados, Mensagens\}$ em que:

1. $Estados\ discretos = \{Base(i)\}$ que inclui apenas o estado inicial.

2. $Variáveis\ internas = \{lt\_espera, t\_executar, lt\_finalizadas, lt\_entrega, v\_proximo, vm\_proximo, vb1\_proximo\}$ em que:

   - A variável *lt_espera* guarda a lista de tarefas na base para entregar aos veículos.
   - A variável *t_executar* guarda uma tarefa a ser passada a um veículo para ser executada.
   - A variável *lt_finalizadas* guarda a lista com todas as tarefas entregues e já executadas na base.
   - A variável *lt_entrega* guarda a lista de tarefas na base para passar pelo veículo mensageiro e entregar noutra base.

- A variável *v_proximo* representa uma variável booleana que está ativa se existir um veiculo próximo da base pronto a executar uma tarefa e está inativa se isto não se verificar.
- A variável *vm_proximo* representa uma variável booleana que está ativa se o veículo mensageiro estiver próximo da base e está inativa se isto não se verificar.
- A variável *vb1_proximo* representa uma variável booleana que está ativa se o veiculo da base remota estiver próximo da b_1 e está inativa se isto não se verificar. Esta variável apenas existe no sistema da b_1.

3. *Sistema de transição de estados* é diferente para b_1 ou para uma *Base(i)*, tendo respetivamente sete e seis transições. De seguida apresentam-se duas tabelas com as transições da b_1 5.5 e de uma *base(i)* 5.6 respetivamente.

Estado inicial	Estado seguinte	Condições de transição	Ações
Base(1)	Base(1)		mensagem_alocar;
Base(1)	Base(1)	mensagem_vbr;	lt_espera += mensagem_vbr;
Base(1)	Base(1)	mensagem_vm;	lt_finalizadas += mensagem_vm;

Base(1)	Base(1)	v_proximo;
		lt_espera \neq {};
			t_executar = lt_espera;
			lt_espera -= t_executar;
			mensagem_executar1{t_executar};

Base(1)	Base(1)	mensagem_v;	lt_finalizadas += mensagem_v;
Base(1)	Base(1)	vb1_proximo; lt_finalizadas \neq {};	mensagem_b1{lt_finalizadas}; lt_finalizadas = {};
Base(1)	Base(1)	vm_proximo;	mensagem_m{lt_entrega};

Tabela 5.5: Transições do sistema b_1 e as suas respetivas condições de transição e ações.

Estado inicial	Estado seguinte	Condições de transição	Ações
Base(i)	Base(i)		mensagem_alocar;
Base(i)	Base(i)	mensagem_entrega;	lt_espera += mensagem_entrega;
Base(i)	Base(i)	mensagem_vm;	lt_finalizadas += mensagem_vm;

Base(i)	Base(i)	v_proximo;
		lt_espera \neq {};
			t_executar = lt_espera;
			lt_espera -= t_executar;
			mensagem_executar(i){t_executar};

Base(i)	Base(i)	mensagem_v;	lt_finalizadas += mensagem_v;
Base(i)	Base(i)	vm_proximo;	mensagem_b(i){lt_finalizadas}; mensagem_m{lt_entrega}; lt_finalizadas = {};

Tabela 5.6: Transições do sistema de uma base e as suas respetivas condições de transição e ações.

4. $Mensagens = \{mensagem\_vbr, mensagem\_vm, mensagem\_v, mensagem\_executar(i), mensagem\_b(i), mensagem\_alocar, mensagem\_m\}$.

Deste conjunto uma base pode enviar uma *mensagem_executar(i)*, para um veículo, com uma tarefa a executar. Pode também enviar uma *mensagem_b(i)* que leva uma lista de tarefas finalizadas que tem como destino o veículo mensageiro, no entanto no caso da b_1 tem como destino o veículo da base remota. Pode ainda enviar uma *mensagem_alocar* para um veículo que esteja alocado nessa mesma base com o intuito de modificar a sua alocação, ou uma *mensagem_m* que tem como destino o veículo mensageiro e contém uma lista de tarefas a entregar na base seguinte.

As mensagens a ser recebidas podem ser provenientes do veículo da base remota, e designam-se por *mensagem_vbr*, sendo que estas apenas chegam à b_1, do veículo mensageiro e designam-se por *mensagem_vm*, ou de um veículo que executa tarefas, e designam-se por *mensagem_v*.

5.4.5 Veículos

O sistema de qualquer um dos *Veículos* da rede, que executam tarefas, foi modelado de acordo com a seguinte figura 5.5. Este sistema interage com o sistema da base à qual estão alocados.

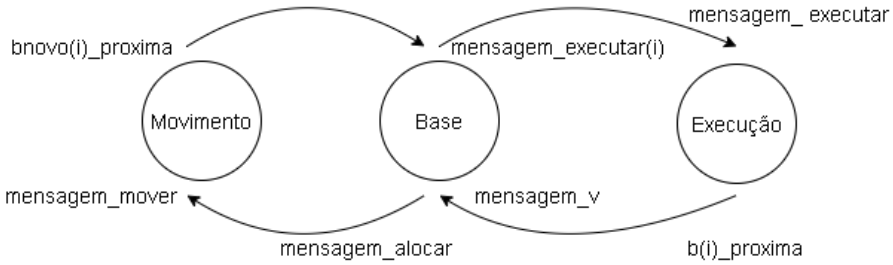


Figura 5.5: Lógica de funcionamento de um veículo da rede.

O sistema de um *Veículo* tem três estados discretos, um deles o estado inicial e ainda três variáveis internas. Possui também várias transições de estados que podem ser despoletadas e também despoletar elas alterações nos valores das variáveis internas ou geração de mensagens.

$Veículo = \{Estados\ discretos, Estado\ inicial, Variáveis\ internas, Sistema\ de\ transição\ de\ estados, Mensagens\}$ em que:

1. $Estados\ discretos = \{Movimento, Base, Execução\}$.
2. $Estado\ inicial = \{Base\}$ quando o veículo espera instruções na base.
3. $Variáveis\ internas = \{t\_execucao, b(i)\_proxima, bnovo(i)\_proxima\}$ em que:
   - A variável *t_execucao* guarda a tarefa que vai ser executada pelo veículo.
   - A variável *b(i)_proxima* representa uma variável booleana que está ativa se o veiculo se encontra na proximidade da base(i), ao alcance da troca de mensagens entre ambos, e está inativa se isto não se verificar.

- A variável *bnovo(i)_proxima* é em tudo semelhante à variável *b(i)_proxima* no entanto esta verifica a proximidade à nova base à qual o veículo vai ser alocado.

4. *Sistema de transição de estados* é composto por quatro transições e está representado na seguinte tabela 5.7.

Estado inicial	Estado seguinte	Condições de transição	Ações
Base	Execução	mensagem_executar(i);	t_execucao = mensagem_executar(i); mensagem_executar{t_execucao};
Execução	Base	b(i)_proxima;	mensagem_v{t_execucao}
Base	Movimento	mensagem_alocar;	novo(i) = mensagem_alocar{novo(i)}; mensagem_mover{bnovo(i)};
Movimento	Base	bnovo(i)_proxima;	(i) = novo(i);

Tabela 5.7: Transições do sistema de um veículo e as suas respetivas condições de transição e ações.

5. $Mensagens = \{mensagem\_executar(i), mensagem\_executar, mensagem\_v, mensagem\_alocar, mensagem\_mover\}$.

   Qualquer veículo da rede envia três tipos de mensagens distintas. São designadas como *mensagem_executar* as mensagens enviadas com os parâmetros de uma tarefa que deve ser executada, como *mensagem_mover* as mensagens enviadas com determinadas coordenadas quando o veículo necessita de se deslocar, ou como *mensagem_v* as mensagens que contém uma tarefa previamente executada.

   As mensagens a ser recebidas podem ser provenientes de uma base à qual o veículo está alocado sendo que se designam por *mensagem_executar(i)* ou *mensagem_alocar*.

5.4.6 Veículo mensageiro

O sistema do *Veículo mensageiro* que percorre a rede de base em base foi modelado de acordo com a seguinte figura 5.6. Este sistema interage com o sistema de todas as bases da rede, excetuando a *Base remota*.

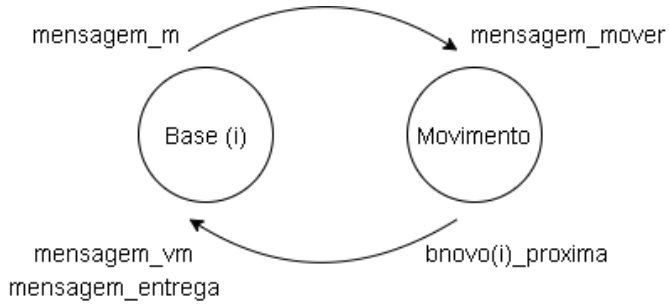


Figura 5.6: Lógica de funcionamento do veículo mensageiro da rede.

O sistema do *Veículo mensageiro* tem apenas dois estados discretos, sendo um deles o estado inicial, e ainda três variáveis internas. Possui também várias transições de estados que podem ser despoletadas e também despoletar elas alterações nos valores das variáveis internas ou geração de mensagens.

Veículo mensageiro $= \{$*Estados discretos*, *Estado inicial*, *Variáveis internas*, *Sistema de transição de estados*, *Mensagens*$\}$ em que:

1. *Estados discretos* $= \{$*Base(i)*, *Movimento*$\}$.
2. *Estado inicial* $= \{$*Base(i)*$\}$ quando se encontra numa base à espera de instruções.
3. *Variáveis internas* $= \{$*lt_m_entrega*, *lt_m_finalizadas*, *bnovo(i)_proxima*$\}$ em que:
   - A variável *lt_m_entrega* guarda a lista de tarefas que vão ser entregues na base seguinte, para ser executadas ou passadas para outra base.
   - A variável *lt_m_finalizadas* guarda a lista de tarefas finalizadas que vão ser entregues na base seguinte.
   - A variável *bnovo(i)_proxima* representa uma variável booleana que está ativa se o veículo se encontra na proximidade da base para a qual o veículo mensageiro se desloca, ao alcance da troca de mensagens entre ambos, e está inativa se isto não se verificar.
4. *Sistema de transição de estados* é composto por apenas duas transições e está representado na seguinte tabela 5.8.
5. *Mensagens* $= \{$*mensagem_m*, *mensagem_mover*, *mensagem_vm*, *mensagem_entrega*$\}$.

   O veículo mensageiro pode enviar uma mensagem que se designa como *mensagem_vm* que contém as tarefas finalizadas até ao momento, ou uma *mensagem_entrega* que contém as tarefas a ser passadas para a base seguinte. O veículo mensageiro pode ainda enviar uma *mensagem_mover* se necessitar de se deslocar.

   As mensagens a ser recebidas podem ser provenientes de uma qualquer base da rede sendo que se designam por *mensagem_m*.

Estado inicial	Estado seguinte	Condições de transição	Ações

Base(i)	Movimento	mensagem_m;	lt_m_entrega = mensagem_m;
			lt_m_finalizadas = mensagem_b(i);
			mensagem_mover{bnovo(i)};

Movimento	Base(i+1)	bnovo(i)_proxima;	mensagem_vm{lt_m_finalizadas}; mensagem_entrega{lt_m_entrega};

Tabela 5.8: Transições do sistema veículo mensageiro e as suas respetivas condições de transição e ações.

5.5 Propriedades do sistema

- É garantido que nenhum veículo fica sem instruções. b_1 processa as mensagens recebidas e reencaminha as mensagens que devem ser enviadas para as restantes bases, b_2, b_3 e b_4. Assim as bases fazem chegar as instruções a cada veículo da rede. No processo nenhum veículo fica sem instruções de movimento para executar. Se não tem tarefas para executar fica parado na base respectiva.

- Nenhum veículo bloqueia. Após ser dada uma instrução essa instrução não pode ser revertida e não há transição possível para um estado de erro.

- As mensagens enviadas pela base remota chegam às restantes bases da rede num período de tempo limitado. Este tempo está limitado ao tempo máximo que pode demorar o veículo mensageiro a recolher a mensagem na b_1 e entregar nas restantes bases. O veículo passa sempre obrigatoriamente e de forma cíclica em todas as bases da rede.

- Mensagens trocadas entre bases e veículos são entregues durante um intervalo de tempo máximo conhecido. Uma vez que os veículos só comunicam com as bases quando estão em proximidade o intervalo de tempo máximo para entregar as mensagens está relacionado com o tempo que o veículo ficará alocado a essa base.

- O sistema é escalável no que refere ao número de veículos e também de bases. Dada a modelação feita o número de bases da rede pode ser qualquer um uma vez que isto não altera o funcionamento da rede como um todo. A cada nova base pode ser alocado um novo veículo, ou ainda pode ser alocado mais do que um veículo por base, não interferindo com as interações entre a base e o veículo.

Capítulo 6

Implementação computacional

6.1 Linguagem de modelização

Para implementar o modelo da rede de veículos coordenados foi necessário optar por linguagem de modelização que se adequasse às propriedades do sistema e do tipo de implementação que se pretendia fazer.

Inicialmente foram ponderadas e analisadas várias hipóteses, e após uma breve discussão com o professor José Pinto foram indicadas duas opções viáveis. A escolha preferencial era uma versão de um modelo de atores que o professor se encontrava a desenvolver no entanto devido às restrições de tempo e uma vez que este ainda não tinha uma versão finalizada, a escolha recaiu sobre a linguagem Rebeca.

6.1.1 Rebeca

Rebeca é uma linguagem, cujo nome surge de *Reactive Object Language*. Os modelos Rebeca funcionam de forma semelhante ao modelo de ator [13] e são baseados em objetos, conhecidos como *rebecs*, que reagem a eventos e passam mensagens de forma assíncrona. Um *rebec* tem uma fila de mensagens e de acordo com os eventos que ocorrem, por exemplo a receção de uma mensagem, o Rebeca pega na mensagem do topo da fila e executa o servidor de mensagens ou método que aí esteja presente.

Seria de esperar, portanto, que a linguagem Rebeca fosse uma escolha acertada para implementar o sistema da rede de veículos. No entanto, devido à pandemia covid-19, havia uma limitação de recursos acessíveis, pelo que apenas havia um computador pessoal disponível para a implementação. Isto levou a uma dificuldade devido a incompatibilidades entre o computador disponível e a linguagem e o seu IDE. A instalação do IDE no sistema operativo Windows não estava a ser possível devido a incompatibilidades na versão do Java instalado, que foi várias vezes reinstalado, com métodos diferentes e versões diferentes. Entretanto a única solução possível foi instalar uma *virtual box* na qual foi instalado o sistema operativo *Ubuntu*, o *java* e depois de alguns problemas solucionados foi possível obter uma instalação final do *IDE Afra 3.0*.

Devido a todos estes contratempos foi gasto tempo em excesso, que juntamente com a falta de à vontade e experiência a trabalhar com a linguagem Rebeca e o seu IDE, foram motivos para rapidamente procurar outra solução para a linguagem e o ambiente de trabalho. Mais um dos motivos fortes para esta tomada de decisão foi o facto de esta linguagem ser utilizado por muito poucas pessoas, pelo que não tinha uma comunidade interativa e comunicativa, e tinha pouca informação disponível quer para aprender a funcionar com a linguagem quer para solucionar problemas que surgissem o que atrasou a implementação.

6.1.2 *MATLAB*

Dado o ponto da situação era necessário encontrar uma solução viável para iniciar a implementação o mais rapidamente possível, sendo assim, a linguagem que acabou mesmo por ser utilizada na implementação foi *MATLAB*. O primeiro motivo desta escolha recai no facto de esta ser uma linguagem com a qual já existia o à vontade e experiência para trabalhar.

O *MATLAB* é uma linguagem bastante flexível, com muitos recursos e ferramentas que podem ser utilizados para auxiliar a implementação que é pretendida. De realçar que esta linguagem só não foi selecionada desde o início uma vez que a proposta inicial era utilizar o modelo de actores, portanto, apesar de este ainda não estar concluído e pronto a ser utilizado, decidiu-se na altura avançar com a linguagem rebeca por ter mais semelhanças com o modelo de atores. O código *MATLAB* pode ser integrado e coordenado com outras linguagens e é possível de ser escalado para testar a implementação em conjuntos de dados maiores e obter resultados. Existem diversas *toolboxes*, bem como uma grande comunidade de utilizadores que é muito interativa para ajudar a resolver problemas que surjam. Adicionalmente com o *MATLAB* é bastante intuitivo fazer *debugging*. Para além destes motivos é sempre apelativo e importante ganhar mais experiência a trabalhar com uma ferramenta como o *MATLAB* tendo em conta a sua influência e a sua vasta utilização na indústria.

6.2 Implementação do modelo do sistema

Ao longo desta secção vai ser descrita a implementação realizada, no *MATLAB*, de acordo com o modelo do sistema desenvolvido já exposto no capítulo 5. Vão ser expostos alguns excertos do código desenvolvido de modo a simplificar a compreensão da descrição. Se alguma dúvida persistir, é possível consultar o código implementado na integra no link do *github* disponível nos anexos A.

6.2.1 Estruturas dos elementos da rede e ambiente de simulação

O princípio da implementação do modelo do sistema passou pela definição e inicialização dos diferentes elementos da rede. Assim, foram criadas estruturas para os diversos elementos da rede: base, veículo, veículo mensageiro e mensagem. Foi criada ainda uma estrutura para guardar as simulações dos veículos da rede.

Lista 6.1: Definição das estruturas do componentes da rede.

```
%%%%%%%%%%%%%%%     ESTRUTURAS     %%%%%%%%%%%%%%%

base = struct('X',{},'Y',{},'MensagemIn',{},'MensagemOut',{});
veiculo = struct('Base',{},'X',{},'Y',{},'Mensagem',{});
veiculo_m = struct('Base',{},'X',{},'Y',{},'MensagemEntrega',{},
    'MensagemFinalizadas',{});
vsim = struct('Sim',[]);
mensagem = struct('ID',{},'ListaTarefas',{});
```

Bases Foram criadas 5 variáveis, identificadas como b_r, b_1, b_2, b_3 e b_4, com a estrutura de uma base e os respetivos valores X e Y que correspondem às suas coordenadas no plano da rede. Estas variáveis correspondem às bases presentes na rede.

Veículos Foram criadas 5 variáveis, identificadas como *vbr*, v_1, v_2, v_3 e v_4, com a estrutura de um veículo, a base à qual estão alocados, e os respetivos valores X e Y que correspondem às suas coordenadas no plano da rede. Estas variáveis correspondem aos veículos presentes na rede. Para cada um destes veículos foi criada uma variável onde é guardada a simulação do veículo.

Veículo Mensageiro Foi criada uma variável, que corresponde ao veículo mensageiro presente na rede, identificado como *vm*, com a respetiva estrutura de um veículo mensageiro, que apenas difere de um veículo normal no tipo de mensagens que transporta.

Mensagens Foram criadas 9 variáveis, correspondentes às mensagens trocadas pela rede, com a estrutura de uma mensagem, denominados como: mensagem_opr; mensagem_br; mensagem_b1; mensagem_b2; mensagem_b3; mensagem_b4; mensagem_executar; mensagem_m; mensagem_vm.

Lista 6.2: Inicialização de componentes da rede.

```
%%%%%%%%%%%%%%%     BASES     %%%%%%%%%%%%%%%

br = base;
br(1).X = 0;
br(1).Y = 0;

%%%%%%%%%%%%%%%     VEICULOS     %%%%%%%%%%%%%%%

vbr = veiculo;
vbrsim = vsim;
vbr(1).Base = 'br';
```

```
12 vbr(1).X = 0.6;
13 x_vbr = vbr(1).X;
14 vbr(1).Y = 0;
15 y_vbr = vbr(1).Y;
16
17 %%%%%%%%%%%%%%%     MENSAGENS     %%%%%%%%%%%%%%%
18
19 mensagem_opr = mensagem;
```

De seguida, foi inicializado um ambiente de simulação onde podemos observar a interação entre os diferentes elementos da rede. Gerou-se uma figura e foram especificados limites para os eixos do X e Y. Esta figura mantém-se aberta até ser manualmente fechada pelo operador. Posteriormente é criado um *plot* que representa cada um dos elementos da rede, isto é: as bases b_r, b_1, b_2, b_3 e b_4; os veículos *vbr*, v_1, v_2, v_3, v_4; e o veículo mensageiro *vm*.

Lista 6.3: Ambiente de simulação e *plots* dos componentes da rede

```
1 %%%%%%%%%%%%%%%     AMBIENTE DE SIMULACAO     %%%%%%%%%%%%%%%
2
3 figure;
4 hold on;
5 xlim([0 20]);
6 ylim([0 20]);
7
8 %%%%%%%%%%%%%%%     PLOT VEICULOS E BASES     %%%%%%%%%%%%%%%
9
10 vbrsim.Sim = plot(vbr(1).X,vbr(1).Y,'bo','MarkerFaceColor','b','
       MarkerSize',7,'DisplayName','Veiculo BR');
11 vmsim.Sim = plot(vm(1).X,vm(1).Y,'go','MarkerFaceColor','g','
       MarkerSize',7,'DisplayName','Veiculo Msg');
12 b1sim = plot(b1(1).X,b1(1).Y,'kx','MarkerFaceColor','k','
       LineWidth',1.5,'MarkerSize',9,'DisplayName','Bases');
```

6.2.2 Variáveis auxiliares

Foi necessário criar um conjunto de variáveis auxiliares que serão utilizadas posteriormente. As variáveis auxiliares incluem várias listas de tarefas, como, por exemplo, a lista de tarefas à espera de ser executadas, ou já finalizadas, das várias bases da rede, ou também a lista de tarefas recebidas, enviadas ou concluídas da base remota. Cada tarefa corresponde a um *cell array* com 5 elementos e valores específicos guardados em cada um desses elementos. Assim sendo, estas listas de tarefas são também *cell array* com vários espaços, com 5 elementos de comprimento cada, para guardar outros *cell arrays*, que correspondem a cada tarefa. Cada uma dessas listas tem

um "id"respetivo que representa o índice atual de cada lista e existe também um "id"que funciona como um identificador para cada mensagem e é acrescentado a cada mensagem enviada.

Foram ainda criadas variáveis como a lista das tarefas possíveis de ser executadas pelos veículos da rede, a velocidade a que eles se deslocam, um timer e uma variável prio que funcionará como um semáforo.

Posteriormente foram ainda criadas várias variáveis booleanas que apenas tomam valores binários, sendo elas:

- init_mov e init_return, para cada veículo. Servem para inicializar o movimento que os veículos vão realizar e iniciar o retorno à base à qual estão alocados, respetivamente. Estas variáveis são todas inicializadas com o valor 1, assumindo que inicialmente os veículos estão parados junto da base à qual estão alocados.

- vbr_proximo e vb1_proximo, que se referem à localização do veículo da base remota e se encontram a 1 ou 0, se este veículo se encontrar próximo ou longe, respetivamente, da base remota e b_1.

- b_vproximo, para cada veículo. Estas variáveis tem valor 1 se existir um veículo perto da respetiva base, e são todas inicializadas a 1.

- m_b_proximo, respetivas a cada base. São variáveis semelhantes às b_vproximo mas indicam se o veículo mensageiro se encontra próximo ou não da respetiva base. A variável m_b1_proximo é inicializada com valor 1, as restantes são inicializadas a 0.

- mover_v, para cada veículo. As variáveis deste tipo tem valor 1 se determinado veículo se estiver a deslocar e 0 se tiver terminado a deslocação e se encontrar parado. São inicializadas a 0.

- v_executar, correspondentes a cada veículo. Estas terão o valor 1 quando um veículo deve executar uma tarefa e variam de valor ao longo da sua execução, alterando o estado de execução da tarefa.

- msg_v, para cada veículo. Estas variáveis são colocadas a 1 quando um veículo chegou a um destino e está ao alcance de enviar uma mensagem. O valor desta variável volta a 0 após o envio da mensagem.

- stop_v e hold_v, respetivas a cada veículo. Servem para controlar algumas deslocações que os veículos tem de executar.

Lista 6.4: Inicialização de algumas variáveis auxiliares

```
%%%%%%%%%%%%%%%     LISTAS DE TAREFAS     %%%%%%%%%%%%%%%

lt_finalizadas = cell(20,5);
b1_lt_espera = cell(20,5);
b1_lt_finalizadas = cell(20,5);
b1_lt_entrega = cell(20,5);

%%%%%%%%%%%%%%%     INDICES     %%%%%%%%%%%%%%%

id_lt_fin = 1;
id_lt_b1esp = 1;
id_lt_b1entr = 1;
id_lt_b1fin = 1;

%%%%%%%%%%%%%%%     OUTRAS     %%%%%%%%%%%%%%%

vbr_proximo = 1;
m_b1_proximo = 1;
b1_vproximo = 1;
mover_v1 = 0;
msg_vbr = 0;
```

6.2.3 Ciclo de execução da rede e escolha de tarefas

Para a execução da simulação do ambiente da rede ser contínua e ser possível observar os resultados ao longo do tempo foi criado um ciclo *while(1)* que é apenas interrompido caso a execução do código seja interrompida no *MATLAB*.

No início do ciclo *while* são escolhidas as tarefas a ser executadas pelos veículos da rede, uma vez que a condição da lista do operador estar vazia e do timer ter valor 0 são ambas verdadeiras. De cada vez que estas condições são verdadeiras são recolhidas as tarefas que o operador quer que sejam executadas na rede. Para recolher mais do que uma tarefa de cada vez, o código que vai ser descrito de seguida, foi colocado dentro de um ciclo "for"de 3 iterações, isto é, o código que está dentro do ciclo é executado por 3 vezes até sair do ciclo permitindo assim repetir a recolha de uma tarefa 3 vezes com o "input"do operador.

A cada iteração abre-se uma janela para o operador com a lista de tarefas possíveis de serem executadas na rede, isto é: Mapeamento, Patrulha e Reconhecimento. Depois do operador escolher qual a tarefa que quer que seja executada é recolhido um valor, "tarefa_id", que vai identificar que tipo de tarefa foi escolhida, para posteriormente obter as suas especificações. Mediante

esse mesmo valor da "tarefa_id"uma de 3 condições verificadas de seguida vai ser verdadeira: "tarefa_id"tem valor 1; "tarefa_id"tem valor 2; ou "tarefa_id"tem valor 3.

Reconhecimento Se "tarefa_id"tiver valor 3 estamos perante uma tarefa do tipo Reconhecimento. De seguida abre um janela onde o operador poderá definir as coordenadas X e Y onde quer que um veículo vá realizar um Reconhecimento. Dependendo das coordenadas definidas pelo operador esta tarefa recai numa determinada área de trabalho, sendo que elas são divididas em: X < 10 e Y < 10, área de trabalho 1 referente a b_1; X >= 10 e Y < 10, área de trabalho 2 referente a b_2; X >= 10 e Y >= 10, área de trabalho 3 referente a b_3; X < 10 e Y >= 10, área de trabalho 4 referente a b_4; Depois de calculada a área de trabalho a tarefa é guardada na lista de tarefas do operador com o respetivo nome, Reconhecimento, coordenadas, X e Y, e área de trabalho.

Patrulha Se "tarefa_id"tiver valor 2 estamos perante uma tarefa do tipo Patrulha. Depois de definir o valor de "tarefa_id"abre um janela onde o operador poderá definir especificações da tarefa, sendo elas, neste caso, as coordenadas X e Y onde quer que um veículo vá realizar a Patrulha, e o tempo durante o qual esta tarefa deve ser executada. Posteriormente é calculada a área de trabalho à qual a tarefa corresponde, de acordo com as coordenadas escolhidos pelo operador, de forma semelhante à tarefa de Reconhecimento. Depois de calculada a àrea de trabalho a tarefa é guardada na lista de tarefas do operador com o nome, Patrulha, as coordenadas, X e Y, a sua respetiva área de trabalho, e o tempo de execução da tarefa.

Lista 6.5: Janela do operador e excertos de definição do tipo de uma tarefa

```
1  %%%%%%%%%%%%%%    TAREFAS OPERADOR    %%%%%%%%%%%%%%
2
3  if (isempty(lt_opr{1,1}) && timer == 0)
4      %...
5      for i=1:1:3
6          %janela do operador (TAREFA)
7          [tarefa_id] = listdlg('ListString',lista,'
               PromptString','Selecionar o tipo de tarefa a
               executar.','SelectionMode','single','ListSize'
               ,[200,100]);
8
9          %tarefa do tipo RECONHECIMENTO
10         if tarefa_id == 3
11             %coordenadas
12             coords = (inputdlg(xy,titulo_r,tamanho)).';
13
14             %area de trabalho
15             if (str2double(coords(1)) < 10 && str2double(
                   coords(2)) < 10)
```

```
16                areatrab = 1;
17            %...
18            tarefa = {'Reconhecimento',str2double(coords(1))
                  ,str2double(coords(2)),areatrab,'_'};
19            lt_opr(id_lt_opr,:) = tarefa;
20            id_lt_opr = id_lt_opr + 1;
21            disp('Tarefa:');
22            disp(tarefa);
23
24        %tarefa do tipo PATRULHA
25        elseif tarefa_id == 2
26            %...
27            tarefa = {'Patrulha',str2double(coords(1)),
                  str2double(coords(2)),areatrab,str2double(
                  coords(3))};
28            %...
```

Mapeamento Se "tarefa_id"tiver valor 1 o operador escolheu uma tarefa do tipo Mapeamento. De seguida abre um janela onde o operador poderá definir as coordenadas X e Y para dois pontos entre os quais o operador quer que seja realizada a tarefa de Mapeamento. Dependendo das coordenadas definidas para cada um dos pontos esta tarefa pode, tal como as tarefas de Reconhecimento e Patrulha, recair em apenas uma área de trabalho. Isto acontece quando ambos os pontos se encontram no mesmo quadrante da área de trabalho. No entanto quando isto não acontece é necessário fazer uma divisão da tarefa, dependendo da quantidade de áreas de trabalho que vão ser percorridas por um veículo ao longo da execução, sendo que desta divisão resultará um número de tarefas igual a esse número de áreas de trabalho, até um máximo de 4 tarefas diferentes.

Algoritmo de divisão de uma tarefa de mapeamento

Para definir a respetiva área de trabalho e possivelmente dividir uma tarefa de mapeamento começa-se por verificar as seguintes condições correspondentes à coordenada X dos dois pontos em questão: ambos os pontos tem X < 10, pertencendo ao 1º e/ou 4º quadrante; ambos os pontos tem X >= 10 pertencendo ao 2º e/ou 3º quadrante; o 1º ponto tem X < 10 e o 2º ponto tem X >= 10; o 1º ponto tem X >= 10 e o 2º ponto tem X < 10; Para cada uma destas 4 condições verificadas para as coordenadas X de cada ponto voltam a ser consideradas 4 condições para as coordenadas Y de cada ponto.

1ª condição em X Ambos os pontos tem Y < 10, área de trabalho 1 referente a b_1; Ambos os pontos tem Y >= 10, área de trabalho 4 referente a b_4; O 1º ponto tem Y < 10 e o 2º ponto

tem Y >= 10, acrescentam-se 2 tarefas à lista do operador sendo que uma tem área de trabalho 1 e outra área de trabalho 4; O 1º ponto tem Y >= 10 e o 2º ponto tem Y < 10, acrescentam-se 2 tarefas à lista do operador sendo que uma tem área de trabalho 4 e outra área de trabalho 1;

2ª condição em X Ambos os pontos tem Y < 10, área de trabalho 2 referente a b_2; Ambos os pontos tem Y >= 10, área de trabalho 3 referente a b_3; O 1º ponto tem Y < 10 e o 2º ponto tem Y >= 10, acrescentam-se 2 tarefas à lista do operador sendo que uma tem área de trabalho 2 e outra área de trabalho 3; O 1º ponto tem Y >= 10 e o 2º ponto tem Y < 10, acrescentam-se 2 tarefas à lista do operador sendo que uma tem área de trabalho 3 e outra área de trabalho 2;

3ª condição em X] Ambos os pontos tem Y < 10, acrescentam-se 2 tarefas à lista do operador sendo que uma tem área de trabalho 1 e outra área de trabalho 2;

Ambos os pontos tem Y >= 10, acrescentam-se 2 tarefas à lista do operador sendo que uma tem área de trabalho 4 e outra área de trabalho 3;

O 1º ponto tem Y < 10 e o 2º ponto tem Y >= 10, acrescentam-se 4 tarefas à lista do operador sendo que cada uma tem um área de trabalho diferente (1, 2, 3 e 4). O 1º ponto tem Y >= 10 e o 2º ponto tem Y < 10, acrescentam-se 4 tarefas à lista do operador sendo que cada uma tem um área de trabalho diferente (1, 2, 3 e 4).

4ª condição em X Estrutura semelhante ao apresentado em *3ª condição em X* apenas com os pontos espelhados.

Sempre que é necessário dividir uma tarefa de Mapeamento em 2, ou até 4, tarefas com áreas de trabalho diferentes são utilizados pontos intermédios com as coordenadas do local onde ocorre a interseção com o limite da área de trabalho, sendo que estas interseções ocorrem sempre em X = 10 ou Y = 10, uma vez que são estes os eixos que separam as 4 áreas de trabalho.

Lista 6.6: Excertos da definição de uma tarefa de mapeamento

```
1 %tarefa do tipo MAPEAMENTO
2 elseif tarefa_id == 1
3     %divisao por quadrantes
4     if (str2double(coords(1)) < 10 && str2double(coords(3))
          < 10)
5         if (str2double(coords(2)) < 10 && str2double(coords
              (4)) < 10)
6             areatrab = 1;
7             tarefa = {'Mapeamento',[str2double(coords(1)),
                  str2double(coords(2))],[str2double(coords(3))
                  ,str2double(coords(4))],areatrab,'_'};
8             lt_opr(id_lt_opr,:) = tarefa;
```

```
9                id_lt_opr = id_lt_opr + 1;
10           %...
11       elseif (str2double(coords(1)) < 10 && str2double(coords
             (3)) >= 10)
12           %...
13            elseif (str2double(coords(2)) < 10 && str2double(
                 coords(4)) >= 10)
14                areatrab = 1;
15                tarefa = {'Mapeamento',[str2double(coords(1)),
                     str2double(coords(2))],[10,10],areatrab,'_'};
16               %...
17                areatrab = 2;
18                tarefa = {'Mapeamento',[str2double(coords(3)),
                     str2double(coords(2))],[10,10],areatrab,'_'};
19               %...
20                areatrab = 3;
21                tarefa = {'Mapeamento',[str2double(coords(3)),
                     str2double(coords(4))],[10,10],areatrab,'_'};
22               %...
23                areatrab = 4;
24                tarefa = {'Mapeamento',[str2double(coords(1)),
                     str2double(coords(4))],[10,10],areatrab,'_'};
25               %...
```

Em qualquer um destes casos, sempre que uma tarefa é introduzida na lista do operador o valor do id_lt_opr, índice da lista do operador, é acrescentado em 1 e é ainda apresentada a respetiva tarefa com um *display* na consola. Quando acontece uma divisão de tarefas no caso da tarefa de Mapeamento a tarefa apresentada na consola com um *display* é a tarefa original introduzida pelo operador, no entanto, na lista do operador aparecem as respetivas tarefas depois da divisão e o "id_lt_opr"é incrementado de acordo com a quantidade de tarefas guardadas.

6.2.4 Base remota e a interação com o veículo da base remota e b_1

Mensagem do operador para a base remota

Depois do operador introduzir as tarefas a serem executadas na rede e estas serem guardadas na lt_opr verifica-se o conteúdo da lista de tarefas do operador com a função *isempty*, que retorna 1 se a lista estiver vazia e 0 se a lista tiver alguma tarefa guardada. Caso a lista do operador tenha tarefas guardadas é gerada uma mensagem do operador, chamada mensagem_opr, com o id atual de mensagem e a lista de tarefas a ser passada na mensagem é uma cópia da lista do operador. Esta mensagem é depois passada para a base remota, sendo que é guardada na "MensagemIn"da base.

De seguida é corrido um ciclo "for", com um número de iterações correspondente ao tamanho da lista do operador, onde a cada iteração é copiado para a lista de tarefas recebidas da base remota (br_lt_recebidas), na posição do índice atual, o respetivo elemento da lista do operador, sendo que o valor do índice vai sendo também atualizado.

No fim do ciclo o valor do id das mensagens é atualizado, a lista do operador é apagada, o índice da lista volta a tomar o valor de 1 e a mensagem_opr é também apagada.

Mensagem da base remota para o vbr

Após a lista de tarefas recebidas na base remota deixar de estar vazia, se o veículo da base remota estiver próximo, que é verificado se a variável vbr_proximo tiver valor 1, e a variável da prioridade (prio) estiver a 0 é gerada uma mensagem, denominada mensagem_br, com o id atual e o conteúdo da br_lt_recebidas na sua lista de tarefas. Esta mensagem é passada ao veículo da base remota e a variável mover_vbr toma valor 1, para o *vbr* iniciar a sua deslocação. Esta variável prio funciona como um semáforo, se estiver a 0 deixa enviar a mensagem da b_r para o *vbr*, caso contrário impede que isso acontece para enviar outra mensagem primeiro, no sentido inverso, como será descrito posteriormente.

Corre-se depois um ciclo "for", com um número de iterações correspondente ao tamanho da lista do operador, para copiar o conteúdo da br_lt_recebidas para a lista de tarefas enviadas pela base remota (br_lt_enviadas). O valor do índice da br_lt_enviadas vai sendo também atualizado. No fim do ciclo o valor do id é atualizado, a br_lt_recebidas é apagada e o seu respetivo índice volta a tomar o valor de 1, e a mensagem_br é apagada.

Lista 6.7: Mensagem enviada pela base remota para o *vbr* e ações despoletadas

```
%%%%%%%%%%%%%%%    LT_RECEBIDAS =/= {}, MSG_BR    %%%%%%%%%%%%%%%

%existem tarefas na br_lt_recebidas e vbr_proximo, msg_br --->
    vbr
if(isempty(br_lt_recebidas{1,1}) == 0 && vbr_proximo && prio ==
    0)
     mensagem_br(1).ID = id;
     mensagem_br(1).ListaTarefas = br_lt_recebidas;
     vbr(1).Mensagem = mensagem_br;
     mover_vbr = 1;
     tam_br_lt_recebidas = sum(cellfun('size',br_lt_recebidas
         (:,1),1));

     for i=1:1:tam_br_lt_recebidas
          br_lt_enviadas(id_lt_brenv,:) = br_lt_recebidas(i,:);
          id_lt_brenv = id_lt_brenv + 1;
```

```
14      end
15
16      id = id + 1;
17      id_lt_brrec = 1;
18      br_lt_recebidas = cell(20,5);
19      mensagem_br = mensagem;
20  end
```

Mensagem do veículo da base remota para a b_1

Quando o veículo da base remota termina a sua deslocação até à base 1 (b_1) deve enviar uma mensagem a essa mesma base. Para tal acontecer é necessário que a variável vb1_proximo tenha valor 1 bem como a variável msg_vbr. Uma vez verificadas estas condições a mensagem transportada pelo *vbr* é enviada para a b_1, ficando esta guardada na MensagemIn da b_1, e sendo esta mesma mensagem posteriormente apagada.

Depois de obter o tamanho da lista de tarefas que acabou de ser guardada na MensagemIn da b_1 é utilizado um ciclo "for"com esse mesmo tamanho para percorrer toda a lista: se a área de trabalho de uma tarefa corresponder ao valor 1 essa tarefa é adicionada à lista de espera da b_1 (b1_lt_espera) e o valor do índice dessa lista é atualizado; caso contrário a tarefa é adicionada à lista de entrega da b_1 (b1_lt_entrega) e o valor do índice dessa lista é atualizado.

Caso seja a primeira vez que o *vbr* se desloca até à b_1 o valor da variável init_vm estará a 1, o que vai despoletar o início de uma deslocação contínua e cíclica do veículo mensageiro. Nesta situação é gerada uma mensagem, chamada mensagem_m, que contém o id atual e uma cópia da b1_lt_entrega, e esta mensagem é enviada para o veículo mensageiro e guardada na MensagemEntrega deste mesmo. A variável mover_vm toma valor 1, a variável init_vm toma valor 0 e nunca mais é alterada, o id é atualizado, a b1_lt_entrega é apagada e o seu id toma valor 1, e por fim a mensagem_m é apagada. Para terminar este evento o id é atualizado e a variavél msg_vbr volta a tomar valor igual a 0. Este excerto de código é posteriormente adaptado para as bases b_2, b_3 e b_4.

6.2.5 Bases da rede (b_1, b_2, b_3 e b_4)

Mensagem_executar de b_1 para v_1

Passando à interação das bases com os veículos que executam tarefas, tomando b_1 e v_1 como exemplo, são necessárias 3 condições para poder colocar o veículo a executar uma tarefa: a b1_lt_espera ter tarefas guardadas; a variável b1_vproximo ter valor 1; a variável msg_v1 ter valor 0, para garantir que caso um veículo esteja a retornar à base após uma tarefa e haja tarefas em lista de espera, este veículo tenha tempo de enviar uma mensagem à b_1 antes de ser enviado para executar outra tarefa.

Neste caso é gerada uma mensagem, denominada mensagem_executar, onde é guardado o id atual e o primeiro elemento da b1_lt_espera. Esta mensagem é enviada para o v_1, que a guarda em Mensagem, e a variável mover_v1 toma o valor 1. Seguidamente é corrido um ciclo "for"com o tamanho da b1_lt_espera, onde se copia para o índice atual o conteúdo do índice seguinte, de modo a apagar a tarefa que acabou de ser enviada na mensagem_executar. Por fim subtrai-se 1 ao valor do índice da lista de espera, atualiza-se o id, e apaga-se a mensagem_executar. Este excerto de código é posteriormente replicado para as restantes bases, b_2, b_3 e b_4, ajustando apenas as variáveis à respetiva base.

Mensagem de v_1 para b_1

Quando um veículo executa um tarefa e retorna à base à qual está alocado deve enviar uma mensagem, portanto, se a variável b1_vproximo tiver valor 1 e msg_v1 também tiver valor 1, indica que o veículo chegou à base. Sendo assim a tarefa existente na mensagem guardada pelo v_1 é copiada para a lista de tarefas finalizadas da b_1 (b1_lt_finalizadas). O índice desta lista é atualizado, a mensagem guardada pelo v_1 é apagada e a variável msg_v1 volta a ter valor 0. Tal como o excerto de código abordado anteriormente este também é replicado para as restantes bases, b_2, b_3 e b_4, ajustando apenas as variáveis à respetiva base.

Lista 6.8: Mensagem de v_1 para b_1 e atualização da lista de tarefas finalizadas

```
1 %%%%%%%%%%%%%%%     MSG_V1 -> B1     %%%%%%%%%%%%%%%
2
3 if (b1_vproximo && msg_v1)
4       b1_lt_finalizadas(id_lt_b1fin,:) = v1(1).Mensagem.
            ListaTarefas(1,:);
5       id_lt_b1fin = id_lt_b1fin + 1;
6       v1(1).Mensagem = mensagem;
7
8       msg_v1 = 0;
9 end
```

Mensagem do veículo mensageiro para a b_1

Quando as seguintes condições se verificam: as variáveis m_b1_proximo e msg_vm tem valor 1; significa que o veículo mensageiro está a passar pela b_1 após completar um ciclo por todas as bases, o que vai despoletar dois eventos. Primeiro é utilizado um ciclo "for"com o tamanho da lista de tarefas finalizadas por todas as bases, à exceção da b_1, recolhidas pelo veículo mensageiro (lt_finalizadas), de forma a copiar para b1_lt_finalizadas o conteúdo da lista lt_finalizadas. Depois é repetido o processo, tal como anteriormente quando init_vm tem valor 1, de forma a manter

o movimento cíclico do *vm* e entregar as tarefas presentes na b1_lt_entrega. Neste caso é também necessário apagar a lt_finalizadas e colocar o seu respetivo índice a 1, dar valor 0 à variável msg_vm e apagar também a MensagemFinalizadas do *vm*.

Mensagem de b_1 para a base remota

Tendo a b_1 a lista de tarefas finalizadas atualizada envia-se informação para o *vbr* de modo a que esta seja transportada para a base remota. Isto acontece quando a b1_lt_finalizadas tem tarefas guardadas e as variáveis vb1_proximo e m_b1_proximo tem valor 1, sendo que esta condição relativa ao veículo mensageiro tem o propósito de garantir que quando o *vbr* é enviado de volta para a base remota leva consigo a melhor atualização possível de tarefas finalizadas, recolhidas pelo *vm*, até ao momento.

Verificando estas condições é gerada uma mensagem (mensagem_b1) com o respetivo id de mensagem e uma cópia da b1_lt_finalizadas e enviada para o *vbr*, que a guarda em Mensagem. É dado o valor de 1 à variável mover_vbr, para iniciar o movimento do *vbr*, a variável prio toma valor 1 (para dar prioridade à mensagem enviada pelo *vbr* para a b_r ao invés da mensagem que pode ser enviada pela b_r ao *vbr* quando este lá chega), é atualizado o id, a b1_lt_finalizadas é apagada e o seu índice obtém valor 1 e a mensagem_b1 é apagada.

Mensagem do vbr para a base remota

Quando o *vbr* chega à base remota é guardada a informação das tarefas finalizadas nessa mesma base para permitir acesso à informação ao operador. Sabe-se que o *vbr* chegou à b_r quando as variáveis vbr_proximo e msg_vbr tem valor 1. É necessário correr 2 ciclos "for", sendo que o primeiro itera até ao tamanho da lista de tarefas finalizadas transportada pelo *vbr* e o segundo itera até ao tamanho da br_lt_enviadas. O primeiro ciclo tem o propósito de guardar as tarefas transportadas pelo *vbr* na br_lt_concluidas e ir atualizando o respetivo índice da lista. O segundo ciclo serve para, a cada iteração, comparar a tarefa da lista transportada pelo *vbr* da respetiva iteração em que se encontra com a br_lt_enviadas até encontrar essa mesma tarefa, e quando a encontrar apagar esta tarefa da br_lt_enviadas e retroceder uma posição a todos os restantes elementos da lista. De cada vez que uma tarefa é apagada da br_lt_enviadas o valor do seu respetivo índice é subtraído em 1. No fim o valor das variáveis msg_vbr e prio é colocado a 0.

Lista 6.9: Mensagem do *vbr* para a b_r e ações despoletadas

```
%%%%%%%%%%%%%%    MSG_VBR -> BR    %%%%%%%%%%%%%%

    if (vbr_proximo && msg_vbr)
        tam_lt_concluidas = sum(cellfun('size',vbr(1).Mensagem.
            ListaTarefas(:,1),1));

        for i=1:1:tam_lt_concluidas
            tam_lt_enviadas = sum(cellfun('size',br_lt_enviadas
                (:,1),1));

            for c=1:1:tam_lt_enviadas
                if (cellfun(@isequal,vbr(1).Mensagem.
                    ListaTarefas(i,:),br_lt_enviadas(c,:)))
                    br_lt_enviadas(c,:) = br_lt_enviadas(c+1,:);
                    id_lt_brenv = id_lt_brenv - 1;
                    a = 1;
                end
                if a == 1
                    br_lt_enviadas(c,:) = br_lt_enviadas(c+1,:);
                end
            end

            br_lt_concluidas(id_lt_brfin,:) = vbr(1).Mensagem.
                ListaTarefas(i,:);
            id_lt_brfin = id_lt_brfin + 1;
            a = 0;
        end

        prio = 0;
        msg_vbr = 0;
    end
```

Mensagem do veículo mensageiro para a b_2, e mensagens de b_2 para o vm

Como já foi referido as restantes bases, b_2, b_3 e b_4, são em grande parte semelhantes à b_1 no seu funcionamento com apenas algumas nuances onde diferem. Têm igual funcionamento no que toca à interação da base com o veículo que executa tarefas, apenas divergem na interação com o veículo mensageiro, uma vez que estas bases, à exceção da b_1, não interagem com o *vbr*, logo é o

vm que transporta até estas bases as tarefas que devem executar, e é também este que transporta as tarefas finalizadas por estas mesmas bases até à lista final de tarefas finalizadas que é guardada na b_1.

Assim sendo, verifica-se quando o veículo mensageiro está próximo de uma base, por exemplo b_2, através das seguintes condições: as variáveis m_b2_proximo e msg_vm tem valor 1. Nesta situação começa por acontecer algo semelhante ao momento em que o *vbr* chega à b_1, sendo que a b_2 recebe a informação guardada na MensagemEntrega do *vm* e guarda as tarefas recebidas na b2_lt_espera ou b2_lt_entrega. Depois, agora sem ser necessário o init_vm, é também gerada e enviada uma mensagem de entrega para o veículo mensageiro, a variável mover_vm toma o valor 1, tal como se apagam as respetivas mensagens e listas, os seus índices tornam a ter valor 1 e a variável msg_vm toma valor 0.

A diferença está na atualização da lt_finalizadas que vai ser entregue na b_1, pelo que é utilizado um ciclo "for"com o tamanho da b2_lt_finalizadas, de modo a copiar para a lt_finalizadas o conteúdo da b2_lt_finalizadas. Para passar esta informação é gerada uma mensagem (mensagem_b2), com o id atual e uma cópia da lt_finalizadas, que é enviada para o *vm* e guardada em MensagemFinalizadas. Para finalizar, a mensagem_b2 é apagada, o id é novamente atualizado, a b2_lt_finalizadas é apagada e o seu índice volta a 1. Após esta adaptação de código ser concluída, este mesmo excerto de código é utilizado para a b_2, b_3 e b_4.

Por fim o timer é incrementado a cada iteração do ciclo *while* até atingir o valor 4000, sendo que quando isto acontece o valor do timer é forçado a 0 novamente, para pedir novo "input"de tarefas ao operador no ciclo seguinte. Caso a função mover tenha valor 0 é realizada uma pausa de 0.01 segundos, para garantir a integridade temporal da simulação caso nenhum veículo se esteja a mover.

Seguidamente é necessário controlar as deslocações dos diferentes veículos da rede. Inicialmente existe uma variável mover que toma valor 1 caso qualquer uma das seguintes variáveis tenha valor 1, e 0 se todas elas estiverem a 0: mover_vbr; mover_vm; mover_v1; mover_v2; mover_v3; mover_v4. Caso a variável mover tenha valor 1 é verificado de seguida qual, ou quais, dos veículos se estão a mover.

6.2.6 Veículo da base remota

Se mover_vbr passar a ter valor 1:

- Começamos por fazer uma inicialização do movimento (init_mov_vbr a 1): se a base do *vbr* for "br"define-se o destino, x2_vbr e y2_vbr com as coordenadas correspondentes à b_1 e nb_vbr passa a ser "b1". De seguida calculam-se os vetores de movimento (xvec_vbr e yvec_vbr) e coloca-se init_mov_vbr e vbr_proximo a 0;

  se a base for "b1"define-se x2_vbr e y2_vbr com as coordenadas correspondentes à b_r e nb_vbr passa a ser "br"; Repete-se o restante processo.

- Atualiza-se a posição do veículo: x_vbr e y_vbr atualizam o seu valor ao somar ao valor anterior o respetivo vetor de deslocamento multiplicado pela velocidade. Estes valores são

arredondados a uma casa decimal com a função *round*. A posição do *plot* da simulação do *vbr* é atualizada com as coordenadas atuais do veículo.

Lista 6.10: Excerto da inicialização de movimento por parte do vbr

```
%%%%%%%%%%%%%%    MOVER    %%%%%%%%%%%%%%

mover = mover_vbr || mover_vm || mover_v1 || mover_v2 ||
    mover_v3 || mover_v4;

if mover
    %%%%%%%%%%%%%%    VBR    %%%%%%%%%%%%%%
    if mover_vbr
        %definir destino e trajetoria
        %br ---> b1
        if (vbr(1).Base == "br" && init_mov_vbr)
            x2_vbr = 6;
            y2_vbr = 5.4;
            nb_vbr = 'b1';
            xvec_vbr = (x2_vbr-x_vbr)/sqrt((x2_vbr-x_vbr)
                ^2+(y2_vbr-y_vbr)^2);
            yvec_vbr = (y2_vbr-y_vbr)/sqrt((x2_vbr-x_vbr)
                ^2+(y2_vbr-y_vbr)^2);
            init_mov_vbr = 0;
            vbr_proximo = 0;

        %b1 ---> br
        elseif (vbr(1).Base == "b1" && init_mov_vbr)
            %...
```

- Verificar se chegou ao destino (x_vbr e y_vbr igual a x2_vbr e y_vbr): a variável mover_vbr fica a 0, as variáveis init_mov_vbr e msg_vbr ficam a 1. A base é atualizada com o valor de nb_vbr, se este valor for "b1"vb1_proximo fica a 1, se o valor for "br"então vbr_proximo fica a 1.

6.2.7 Veículo mensageiro

Se mover_vm passar a ter valor 1:

- Começamos por fazer uma inicialização do movimento (init_mov_vm a 1) semelhante ao *vbr*, com apenas diferentes destinos e ajustes de variáveis: se a base do *vm* for "b1"ou "b2"ou

"b3"ou "b4"define-se o destino, x2_vbr e y2_vbr com as coordenadas correspondentes, respetivamente, a "b2", "b3", "b4"e "b1"e nb_vbr passa a ter esse mesmo valor. A variável m_b_proximo correspondente à base da qual o *vm* partiu fica a 0.

- Atualiza-se a posição do veículo, de forma semelhante ao *vbr*.
- Verificar se chegou ao destino (x_vbr e y_vbr igual a x2_vbr e y_vbr) semelhante ao *vbr*, com diferentes destinos. A variável m_b_proximo correspondente à base à qual o *vm* chegou fica a 1.

Lista 6.11: Excerto da finalização de uma deslocação do *vm*

```
%%%%%%%%%%%%%%    MENSAGEIRO    %%%%%%%%%%%%%%

if mover_vm
    %...
    %terminar viagem
    if (x_vmR == x2_vm && y_vmR == y2_vm)
        mover_vm = 0;
        init_mov_vm = 1;
        msg_vm = 1;
        vm(1).Base = nb_vm;

        if vm(1).Base == "b1"
            m_b1_proximo = 1;
        elseif vm(1).Base == "b2"
            m_b2_proximo = 1;
        elseif vm(1).Base == "b3"
            m_b3_proximo = 1;
        elseif vm(1).Base == "b4"
            m_b4_proximo = 1;
        end
    end
end
```

6.2.8 Veículos da rede (v_1, v_2, v_3 e v_4) e execução de tarefas

Uma vez que os veículos v_1, v_2, v_3 e v_4 tem todos uma implementação igual, apenas variando as variáveis para adequar ao veículo em questão, vai ser descrita a implementação do v_1 que foi posteriormente replicada para os restantes veículos mencionados.

A implementação do movimento de v_1 foi dividida de acordo com os três tipos diferentes de tarefas. Se mover_v1 passar a ter valor 1:

Reconhecimento Começamos por fazer uma inicialização do movimento (init_mov_v1 a 1): se b1_vproximo tem valor 1 (sair da base) definir o destino, x2_v1 e y2_v1 com as coordenadas presentes na tarefa guardada pelo veículo e colocar v1_executar a 1; caso contrário o destino corresponde às coordenadas da b_1. De seguida calculam-se os vetores de movimento (xvec_v1 e yvec_v1), coloca-se init_mov_v1 e b1_proximo a 0;

Atualiza-se a posição do veículo (stop_v1 a 0). Existe um ciclo for que incrementa 1 ao valor de hold_v1 a cada iteração, quando stop_v1 é colocado a 1. Quando este ciclo termina stop_v1 volta a 0 e incrementa-se 1 ao valor de hold_v1 para mais tarde sinalizar que o veículo retornou à base.

Verificar se chegou ao destino (x_v1 e y_v1 igual a x2_v1 e y2_v1): se v1_executar tem valor 1, v1_executar obtém valor 0 e init_mov_v1 (para realocar o destino) e stop_v1 (para parar o veículo) ficam a 1; se hold_v1 tem valor 301 o veículo regressou à base, como tal, as variáveis mover_v1 e hold_v1 ficam a 0 e as variáveis b1_vproximo, init_mov_v1 e msg_v1 ficam a 1.

Patrulha Começamos por fazer uma inicialização do movimento (init_mov_v1 a 1): se b1_vproximo (sair da base) ou v1_executar tem valor 1 definir como destino as coordenadas presentes na tarefa guardada no veículo desviadas em 0.5 para trás quer no X quer no Y; se v1_executar tiver valor 2 incrementa-se 1 ao valor do x2_v1 atual; se v1_executar tiver valor 3 incrementa-se 1 ao valor do y2_v1 atual; se v1_executar tiver valor 4 subtrai-se 1 ao valor do x2_v1 atual. De seguida calculam-se os vetores de movimento (xvec_v1 e yvec_v1), coloca-se init_mov_v1 e b1_proximo a 0;

Atualiza-se a posição do veículo (stop_v1 a 0). No fim da atualização da posição a variável hold_v1 é incrementada em 0.01.

Verificar se terminou a patrulha (se a variável hold_v1 tiver um valor inferior ao valor presente na tarefa guardada no veículo). Se x_v1 e y_v1 for igual a x2_v1 e y2_v1 realoca-se o destino atualizando o valor de v1_executar de forma cíclica. A variável init_mov_v1 é colocada a 1. Se hold_v1 for igual ao valor presente na tarefa guardada pelo veículo ou init_return_v1 está a 1 e define-se b_1 como destino, geram-se os vetores de deslocamento e init_return_v1 fica a 0, ou x_v1 e y_v1 é igual a x2_v1 e y2_v1, o veículo chegou à base as variáveis mover_v1 e hold_v1 ficam a 0 enquanto b1_vproximo, init_mov_v1, init_return_v1 e msg_v1 ficam a 1.

Mapeamento Começamos por fazer uma inicialização do movimento (init_mov_v1 a 1): se b1_vproximo (sair da base) as coordenadas dos 2 pontos presentes na tarefa guardada no veículo são guardadas em p1_v1 e p2_v1, define-se como destino as coordenadas X e Y do ponto p1_v1 e a variável v1_executar fica a 1; se v1_executar tiver valor 2 o y2_v1 passa a ser o Y do segundo ponto; se v1_executar tiver valor 3, verificamos qual o maior valor de X entre o p1_v1 e o p2_v1, se o X de p2_v1 for maior incrementa-se 1 ao valor de x2_v1, caso contrário subtrai-se 1; se v1_executar tiver valor 4 o y2_v1 fica com o valor do Y do p1_v1;

se v1_executar tiver valor 5 voltamos a repetir o processo de quando o v1_executar tem valor 3; se v1_executar tiver valor 6 o destino passa a ser as coordenadas de b_1; De seguida calculam-se os vetores de movimento (xvec_v1 e yvec_v1), coloca-se init_mov_v1 e b1_vproximo a 0;

Lista 6.12: Excerto da inicialização do movimento referente a uma tarefa de mapeamento

```
%%%%%%%%%%%%%%%     V1     %%%%%%%%%%%%%%%

if mover_v1
    %executar tarefa
    %...

    %MAPEAMENTO INIT-------------------------
    if (v1(1).Mensagem.ListaTarefas(1,1) == "Mapeamento" &&
        init_mov_v1)
        if b1_vproximo
        p1_v1 = cell2mat(v1(1).Mensagem.ListaTarefas(1,2));
        p2_v1 = cell2mat(v1(1).Mensagem.ListaTarefas(1,3));
        x2_v1 = p1_v1(1);
        y2_v1 = p1_v1(2);
        v1_executar = 1;
        elseif v1_executar == 2
        %...
        elseif v1_executar == 6
        x2_v1 = 6.6;
        y2_v1 = 6;
        end

        if b1_vproximo || v1_executar == 3 || v1_executar ==
            5 || v1_executar == 6
            xvec_v1 = (x2_v1-x_v1)/sqrt((x2_v1-x_v1)^2+(
                y2_v1-y_v1)^2);
        else
            xvec_v1 = 0;
        end
        %...
    end
end
```

Atualiza-se a posição do veículo.

Verificar se chegou ao destino (x_v1 e y_v1 igual a x2_v1 e y2_v1): se x_v1 e y_v1 igual a X e Y do ponto p2_v1 a variável v1_executar obtém valor 6 (retornar à base) e init_mov_v1 é colocada a 1; se x_v1 e y_v1 igual às coordenadas de b_1 variáveis as variáveis mover_v1 e v1_executar ficam a 0 e as variáveis b1_vproximo, init_mov_v1 e msg_v1 a 1 (chegou à base); se não nenhum dos outros casos incrementar v1_executar de forma cíclica para realocar o destino e colocar a variável init_mov_v1 a 1.

Lista 6.13: Excerto da movimentação de v_1 referente a uma tarefa de mapeamento

```
%MAPEAMENTO-------------------------------------------
if (cell2mat(v1(1).Mensagem.ListaTarefas(1,1)) == "
    Mapeamento")
    %atualizar posicao
    x_v1 = x_v1 + (xvec_v1*v);
    y_v1 = y_v1 + (yvec_v1*v);
    x_v1R = round(x_v1,1);
    y_v1R = round(y_v1,1);
    set(v1sim.Sim,'XData',x_v1R,'YData',y_v1R);
    v1(1).X = x_v1R;
    v1(1).Y = y_v1R;

    %terminar viagem
    if (x_v1R == x2_v1 && y_v1R == y2_v1)
        %acabou de mapear
        if (x_v1R == p2_v1(1) && y_v1R == p2_v1(2))
            %...
        %chegou a base
        elseif (x_v1R == 6.6 && y_v1R == 6)
            %...
        %realocar destino
        else
            if v1_executar == 5
                v1_executar = 2;
            else
                v1_executar = v1_executar + 1;
            end

            init_mov_v1 = 1;
        end
    end
end
```

Capítulo 7

Resultados

7.1 Introdução

Este capítulo tem por objectivo apresentar resultados de simulações que ilustram o funcionamento da implementação computacional realizada que foi descrita no capítulo 6.

Inicialmente é exposto o plano de experimentação, que explica a estratégia delineada para iniciar a implementação da rede, com os diferentes objetivos a ser cumpridos. Posteriormente é detalhado o mundo do ambiente de simulação, a sua configuração e que objetos vão interagir neste mundo. É, também, exposto o resultado final da implementação seguindo, etapa a etapa, o plano de experimentação já delineado. Cada uma destas etapas corresponde a um caso de estudo que vai ser descrito, explicando o que foi implementado com foco nos eventos que ocorrem e as respetivas mudanças de valor das variáveis. Por fim é feita uma análise dos resultados finais obtidos ao nível de toda a implementação rede, discutindo que objetivos foram cumpridos.

7.2 Plano de experimentação

O plano de experimentação elaborado tinha dois objetivos distintos pensados, que no entanto se conjugam entre si:

- Observar as variáveis ao longo do tempo de execução do ciclo da rede, para avaliar o comportamento dos elementos da rede, a passagem das diferentes mensagens, avaliar as listas de tarefas das diferentes bases, a passagem de tarefas entre outros eventos que ocorrem na rede;

- Implementar uma simulação, que proporciona-se a possibilidade de visualizar a interação entre os diferentes componentes da rede e ainda exemplificar o "input"de um operador.

Para garantir que a implementação seria eficiente e teria o resultado final esperado foram-se estabelecendo etapas, que funcionavam como *checkpoints*. O objetivo da criação destas etapas era

garantir que ao longo da implementação eram atingidos objetivos parciais, parando a implementação para testar o funcionamento até a essa etapa, para no fim de toda a implementação diminuir a probabilidade de incompatibilidades ou erros na simulação da rede.

Como tal, ainda antes de começar a implementação do modelo da rede, foram especificadas as seguintes etapas:

Mover veículos Criar estruturas para os diferentes componentes da rede, bases, veículos, veículo mensageiro, e mensagens. Colocar dois veículos a deslocar-se de um ponto A a um ponto B.

Tarefas e listas Escolher os diferentes tipos de tarefas que possam ser executadas pela rede. Gerar uma lista com tarefas, e armazená-las numa lista.

Mensagens Enviar mensagens entre o operador e a base remota, e também entre bases e veículos. Transportar uma lista de tarefas de uma base para outra.

Tarefas e a sua área Processar uma lista de tarefas que chega a uma base, dividindo-as para diferentes listas para mais tarde executar a tarefa ou entregar noutra base, dependendo se pertence ou não à sua área de trabalho.

Reconhecimento Executar uma tarefa de Reconhecimento. Guardar a tarefa finalizada na respetiva lista da sua base.

Veículo mensageiro Colocar o veículo mensageiro a enviar por mensagem para cada base as tarefas que devem ser executadas na sua respetiva área de trabalho e recolher as tarefas finalizadas.

Concluir tarefas Retornar o veículo da base remota à base remota para guardar as tarefas concluídas. Processar a lista de tarefas enviadas e concluídas.

Patrulha Executar uma tarefa de Patrulha.

Mapeamento Executar uma tarefa de Mapeamento. Permitir a divisão de tarefas por várias bases tendo em conta a área de trabalho.

7.3 Ambiente de simulação

O mundo da simulação foi criado para auxiliar a perceção de como estão a interagir todos os elementos da rede. Tendo uma apresentação visual dos eventos que estão a ocorrer na rede ao longo do tempo da simulação é mais fácil de compreender os resultados da implementação. Não obstante, a verificação dos resultados deve sempre ser acompanhada da visualização dos valores e estados das variáveis.

Este mundo começou por ser criado com uma figura do *MATLAB*, que é forçada a ficar aberta ao longo da simulação, a menos que esta seja manualmente fechada. Ao definir os limites dos valores possíveis dos eixos do X e do Y, sendo que ambos foram definidos com limite mínimo de 0 e limite máximo de 20, gera-se uma figura com 20 unidades de comprimento e também 20

unidades de largura. O ambiente onde se vai realizar a simulação tem portanto 400 unidades de área.

Após gerar o mundo com esta figura foram adicionados os elementos que correspondem à simulação dos componentes da rede. Foram, portanto, gerados *plots* para as diversas bases, veículos e um veículo mensageiro.

De seguida, apresenta-se uma figura 7.1 na qual é possível ver a janela onde decorre a simulação da rede em que os seus diferentes componentes interagem. É ainda apresentada uma tabela 7.1 na qual todos esses componentes da rede são identificados através do seu símbolo, a cor e as suas coordenadas iniciais.

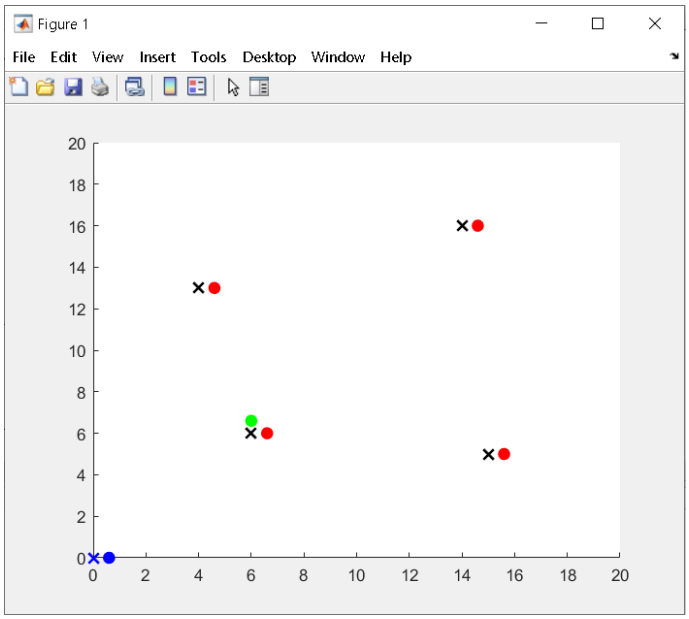


Figura 7.1: Janela com o mundo de simulação da rede.

Elemento	Símbolo	Cor	X inicial	Y inicial
Base Remota	X		0	0
Base 1 (B1)	X		6	6
Base 2 (B2)	X		15	5
Base 3 (B3)	X		14	16
Base 4 (B4)	X		4	13
Veículo da base remota (VBR)	O		0.6	0
Veículo mensageiro (VM)	O		6	6.6
Veículo 1 (V1)	O		6.6	6
Veículo 2 (V2)	O		15.6	5
Veículo 3 (V3)	O		14.6	16
Veículo 4 (V4)	O		4.6	13

Tabela 7.1: Componentes presentes no mundo da simulação.

Tal como já foi referido, para além de observar as interações entre componentes no ambiente de simulação, é também imprescindível monitorizar algumas variáveis para se garantir que os resultados da implementação são os pretendidos. Será então necessário acompanhar a evolução das seguintes variáveis ao longo da execução da simulação: as listas da base remota, ou seja, br_lt_recebidas, br_lt_enviadas e br_lt_concluidas; bem como as listas de espera (lt_espera), de entrega (lt_entrega) e finalizadas (lt_finalizadas) das restantes bases, b_1, b_2, b_3 e b_4;

Importa também monitorizar, dentro da estrutura de cada base, a atual "MensagemIn"em certos momentos, a atual "Mensagem"guardada na estrutura de cada veículo e por fim as atuais "MensagemEntrega"e "MensagemFinalizadas"guardadas na estrutura do veículo mensageiro.

Iniciando a simulação desta rede existem apenas duas formas de a terminar, sendo elas: utilizar o botão de *Pause* seguido do botão *Quit Debugging* do *MATLAB* (o segundo é necessário pois é possível utilizar o botão *Continue* depois de fazer pausa, que retoma a simulação no estado em que estava previamente à pausa); ou fechando manualmente a janela que contém o mundo da simulação caso existam veículos a deslocar-se (isto acontece pois quando se tenta atualizar a posição de um dos veículos em movimento a *figure* onde o *plot* está escrito não existe, dando portanto um erro que termina a simulação).

7.4 Casos de estudo

Ao longo desta secção onde se abordam os casos de estudo vão ser expostos os resultados da implementação, etapa a etapa, seguindo o plano de experimentação, demonstrando o estado do mundo da simulação quando ocorrem eventos importantes, e também analisando o valor de certas variáveis importantes para o funcionamento da rede. É de realçar que estes eventos e as respetivas ações que despoletam tem por base a modelação, e as respetivas transições e ações, exposta no capítulo 5 e são, ainda, o resultado da implementação que foi descrita no capítulo 6.

7.4.1 Mover veículos

O objetivo da primeira etapa, tal como o nome indica, era apenas começar por colocar um veículo a deslocar-se entre dois pontos quaisquer. Nesta etapa o mundo da simulação ainda não estava completo, pelo que na simulação apenas se vê um componente da rede, sendo ele um veículo.

Na seguinte imagem 7.2 é possível verificar que já se produz uma janela com o ambiente de simulação, sendo que as estruturas dos componentes da rede estão também já geradas (aqui apenas se utiliza a estrutura para um veículo). A imagem representa dois momentos distintos da simulação: o primeiro, à esquerda, é o momento em que o veículo vai iniciar o seu deslocamento num ponto predefinido; o segundo, à direita, é o momento em que o veículo já se encontra parado após alcançar o seu destino, que foi definido como o ponto (14,12).

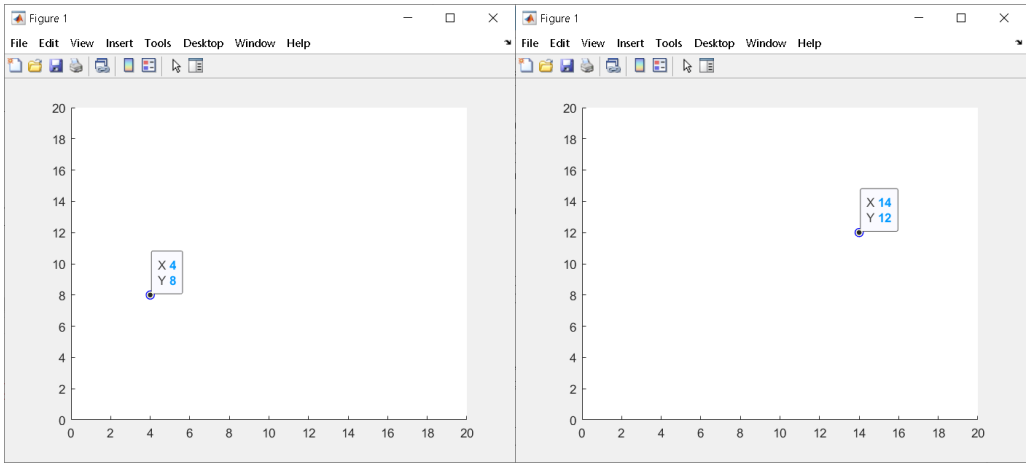


Figura 7.2: Deslocação de um veículo no mundo da simulação.

7.4.2 Tarefas e listas

Nesta etapa era pretendido gerar uma lista com diferentes tipos de tarefa, dando a possibilidade do operador escolher que tipo de tarefa pretende que a rede execute, e quais as especificações dessa mesma tarefa.

A figura 7.3 mostra a janela que se gera ao iniciar a simulação. Esta é uma janela na qual o operador vai escolher o tipo de tarefa a executar, de entre mapeamento, patrulha e reconhecimento. Após escolher um dos tipos de tarefa abre-se uma segunda janela, na qual é pedido ao operador que escolha as especificações dessa mesma tarefa que quer que a rede execute. Esta segunda janela varia de acordo com o tipo de tarefa elegido, sendo que as três janelas possíveis de abrir são apresentadas na figura 7.4.

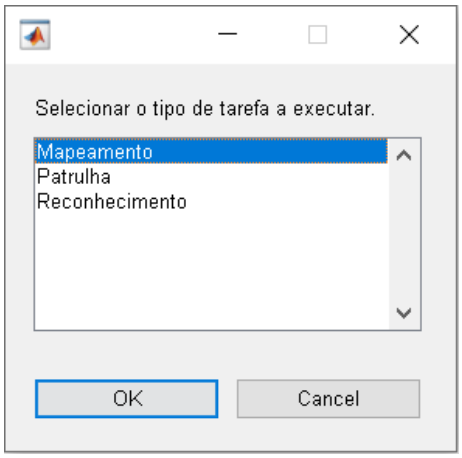


Figura 7.3: Janela de tarefas apresentada ao operador.

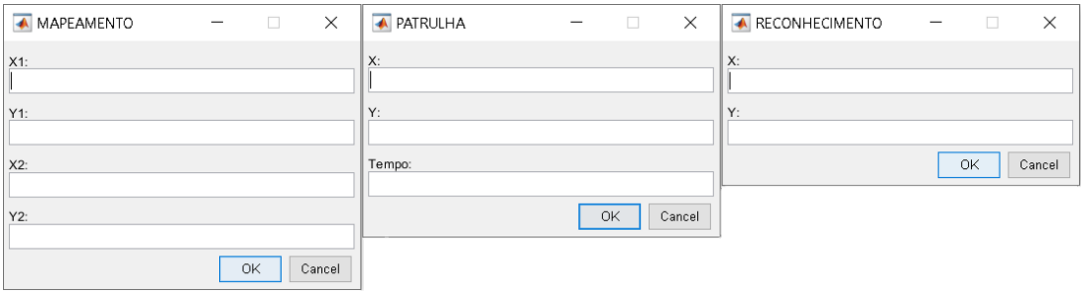


Figura 7.4: Janelas de especificação dos diferentes tipos de tarefas.

Uma vez que foi estabelecido que a rede de veículos poderia executar três tipos de tarefas diferentes de cada vez que a janela do operador (figura 7.3) é gerada é pedido o *input* para três tarefas com as suas respetivas especificações. Sempre que é dado o *input* para uma tarefa esta é impressa na consola do *MATLAB* através dum *display* e posteriormente é guardada numa lista (lt_opr) como é possível verificar nas figuras 7.5 e 7.6, respetivamente. Cada tarefa é guardada na forma de um *cell array* de cinco posições com: 1ª posição, nome do tipo de tarefa; 2ª posição, coordenada X, se for uma tarefa de reconhecimento ou patrulha, ponto inicial, se for uma tarefa de mapeamento; 3ª posição, coordenada Y, se for uma tarefa de reconhecimento ou patrulha, ponto final, se for uma tarefa de mapeamento; 4ª posição, área de trabalho da respetiva tarefa; 5ª posição, tempo de execução da tarefa (apenas utilizada em tarefas do tipo patrulha, caso contrário grava-se um caractere -"apenas para manter o tamanho do *cell array* constante entre todos os tipos de tarefa).

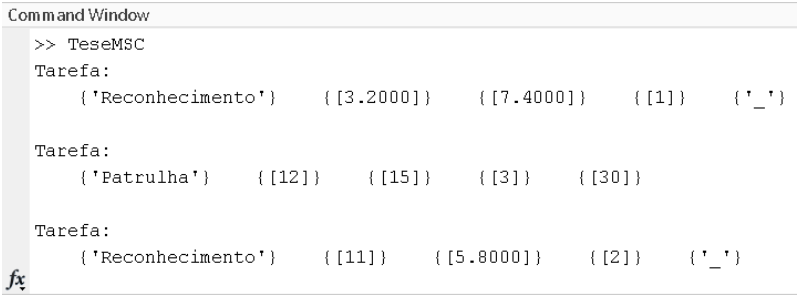


Figura 7.5: *Display* das tarefas na consola do *MATLAB*.

lt_opr

12x5 cell

	1	2	3	4	5
1	'Reconhecimento'	3.2000	7.4000	1	'_'
2	'Patrulha'	12	15	3	30
3	'Reconhecimento'	11	5.8000	2	'_'
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]
6	[]	[]	[]	[]	[]
7	[]	[]	[]	[]	[]
8	[]	[]	[]	[]	[]
9	[]	[]	[]	[]	[]
10	[]	[]	[]	[]	[]
11	[]	[]	[]	[]	[]
12	[]	[]	[]	[]	[]

Figura 7.6: Lista que guarda as tarefas escolhidas pelo operador.

7.4.3 Mensagens

O intuito desta terceira etapa era gerar mensagens que guardem listas de tarefas e utilizar estas mensagens para comunicar entre os diferentes elementos da rede coordenando as suas ações.

Na figura 7.7 é possível verificar a receção por parte da b_1 de uma mensagem enviada pelo operador (mensagem_opr) que é guardada na sua estrutura em MensagemIn (à esquerda). Á direita na mesma figura é ainda visível a lista (br_lt_recebidas) onde é guardada a lista de tarefas enviada através da mensagem_opr.

br | br.MensagemIn | br.MensagemIn.ListaTarefas

br.MensagemIn.ListaTarefas

	1	2	3	4	5
1	'Reconhecimento'	2	3	1	'_'
2	'Reconhecimento'	12	15	3	'_'
3	'Patrulha'	12.4000	17.1000	3	25
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]
6	[]	[]	[]	[]	[]
7	[]	[]	[]	[]	[]
8	[]	[]	[]	[]	[]
9	[]	[]	[]	[]	[]
10	[]	[]	[]	[]	[]
11	[]	[]	[]	[]	[]
12	[]	[]	[]	[]	[]

br_lt_recebidas

20x5 cell

	1	2	3	4	5
1	'Reconhecimento'	2	3	1	'_'
2	'Reconhecimento'	12	15	3	'_'
3	'Patrulha'	12.4000	17.1000	3	25
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]
6	[]	[]	[]	[]	[]
7	[]	[]	[]	[]	[]
8	[]	[]	[]	[]	[]
9	[]	[]	[]	[]	[]
10	[]	[]	[]	[]	[]
11	[]	[]	[]	[]	[]
12	[]	[]	[]	[]	[]

Figura 7.7: Mensagem_opr recebida pela b_1 e atualização da lista de tarefas recebidas da b_r.

Quando a br_lt_recebidas passa a ter uma lista de tarefas guardada, uma vez que o veículo da base remota (*vbr*) está parado junto à base remota (vbr_proximo), é enviada uma mensagem_br, com a lista de tarefas recebidas, da base remota para o *vbr*. Tal como está exposto na figura 7.8 esta

mensagem é recebida pelo *vbr*, que de seguida inicia a sua deslocação em direção a b_1. A lista de tarefas enviada nesta mensagem_br é retirada da br_lt_recebidas e acrescentada à br_lt_enviadas.

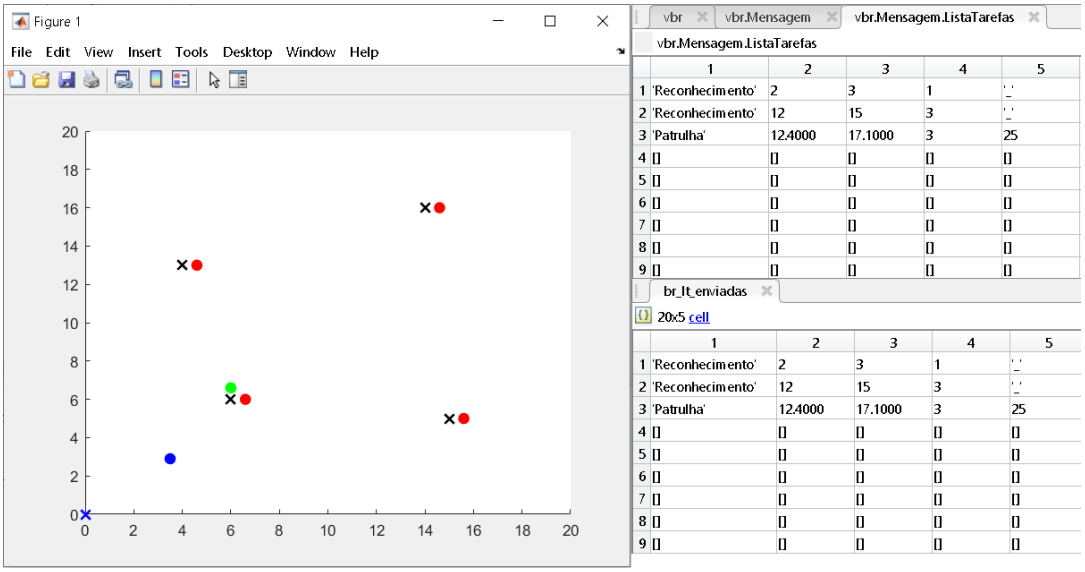


Figura 7.8: *Vbr* a transportar uma lista de tarefas e atualização da lista de tarefas enviadas da b_r.

Chegando próximo de b_1 (vb1_proximo) é enviada uma mensagem do *vbr* para a b_1, com a lista de tarefas enviada inicialmente pela base remota, que é guardada em MensagemIn, como está exposto na figura 7.9.

b1 | b1.MensagemIn | b1.MensagemIn.ListaTarefas

b1.MensagemIn.ListaTarefas

	1	2	3	4	5
1	'Reconhecimento'	2	3	1	'_'
2	'Reconhecimento'	12	15	3	'_'
3	'Patrulha'	12.4000	17.1000	3	25
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]
6	[]	[]	[]	[]	[]
7	[]	[]	[]	[]	[]
8	[]	[]	[]	[]	[]
9	[]	[]	[]	[]	[]
10	[]	[]	[]	[]	[]
11	[]	[]	[]	[]	[]
12	[]	[]	[]	[]	[]

Figura 7.9: Receção de uma mensagem do *vbr* pela b_1.

7.4.4 Tarefas e a sua área

Como é possível comprovar na figura 7.10 após a chegada de uma mensagem à b_1 é possível que a lista de tarefas contemple tarefas cuja área de trabalho se adequa à base em questão, bem como outras cuja área de trabalho não se adequa a essa base.

Sendo assim quando uma base recebe uma mensagem, b_1 recebe uma mensagem de *vbr* ou outra base recebe uma mensagem do *vm*, é corrido um algoritmo que analisa o valor da área de trabalho, que corresponde ao quarto elemento do *cell array* de uma tarefa, e mediante o seu valor guarda a tarefa na lista de espera ou de entrega da base, como se pode verificar na figura 7.11.

b1 | b1.MensagemIn | b1.MensagemIn.ListaTarefas

b1.MensagemIn.ListaTarefas

	1	2	3	4	5
1	'Reconhecimento'	3.5000	7	1	'_'
2	'Reconhecimento'	12.2000	8	2	'_'
3	'Patrulha'	2.7000	14	4	20
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]
6	[]	[]	[]	[]	[]
7	[]	[]	[]	[]	[]
8	[]	[]	[]	[]	[]
9	[]	[]	[]	[]	[]
10	[]	[]	[]	[]	[]
11	[]	[]	[]	[]	[]
12	[]	[]	[]	[]	[]

Figura 7.10: Exemplo de uma lista de tarefas presente na b_1.

b1_lt_espera

20x5 cell

	1	2	3	4	5
1	'Reconhecimento'	3.5000	7	1	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]
6	[]	[]	[]	[]	[]
7	[]	[]	[]	[]	[]
8	[]	[]	[]	[]	[]
9	[]	[]	[]	[]	[]
10	[]	[]	[]	[]	[]
11	[]	[]	[]	[]	[]
12	[]	[]	[]	[]	[]

b1_lt_entrega

20x5 cell

	1	2	3	4	5
1	'Reconhecimento'	12.2000	8	2	'_'
2	'Patrulha'	2.7000	14	4	20
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]
6	[]	[]	[]	[]	[]
7	[]	[]	[]	[]	[]
8	[]	[]	[]	[]	[]
9	[]	[]	[]	[]	[]
10	[]	[]	[]	[]	[]
11	[]	[]	[]	[]	[]
12	[]	[]	[]	[]	[]

Figura 7.11: Divisão de tarefas pela lista de espera e lista de entrega da b_1.

7.4.5 Reconhecimento

Nesta etapa o objetivo delineado foi colocar um veículo a executar uma tarefa que estivesse previamente na lista de espera de uma base.

Fez-se chegar à b_1 uma mensagem com uma tarefa de reconhecimento, atualizando a b1_lt_espera. Tendo b_1 um veículo parado e próximo da base (b1_vproximo) é enviada uma mensagem_executar para o v_1 com a primeira tarefa da b1_lt_espera, sendo esta tarefa posteriormente apagada desta mesma lista. Na figura 7.12 é possível ver um veículo a executar uma tarefa de reconhecimento, que está (à direita) guardada na mensagem que foi enviada para esse veículo. Uma tarefa de reconhecimento consiste no veículo deslocar-se até ao local identificado pelo operador e ficar aí durante um certo tempo predefinido a observar e reconhecer o local. Após passar esse tempo o veículo retorna à base à qual está alocado.

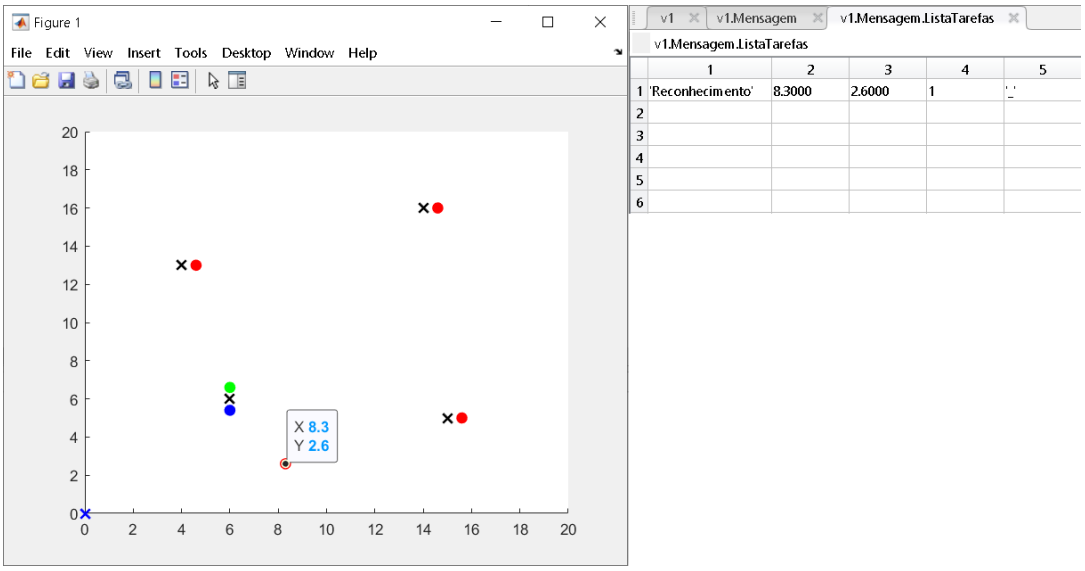


Figura 7.12: Um veículo a executar uma tarefa de reconhecimento.

Quando o v_1, após terminar a execução da tarefa, retorna à b_1 (b1_vproximo), envia uma mensagem levando a base a atualizar a sua lista de tarefas finalizadas com a tarefa que acabou de ser executada e que estava guardada nessa mesma mensagem, tal como está demonstrado na figura 7.13.

b1_lt_finalizadas

20x5 cell

	1	2	3	4	5
1	'Reconhecimento'	8.3000	2.6000	1	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

Figura 7.13: Lista de tarefas finalizadas da b_1 com uma tarefa de reconhecimento.

7.4.6 Veículo mensageiro

Esta etapa é fulcral para o funcionamento da rede uma vez que o veículo mensageiro é o elemento crucial para a coordenação e para estabelecer pontes de comunicação entre todos os componentes da rede. Quando b_1 tem uma lista de tarefas para entregar, em b1_lt_entrega, e o veículo mensageiro se encontra próximo (m_b1_proximo), é enviada uma mensagem_m que contém essa lista para o *vm*. Na figura 7.13 é possível ver o veículo mensageiro a deslocar-se depois de receber uma mensagem_m que está guardada na estrutura do *vm* em MensagemEntrega.

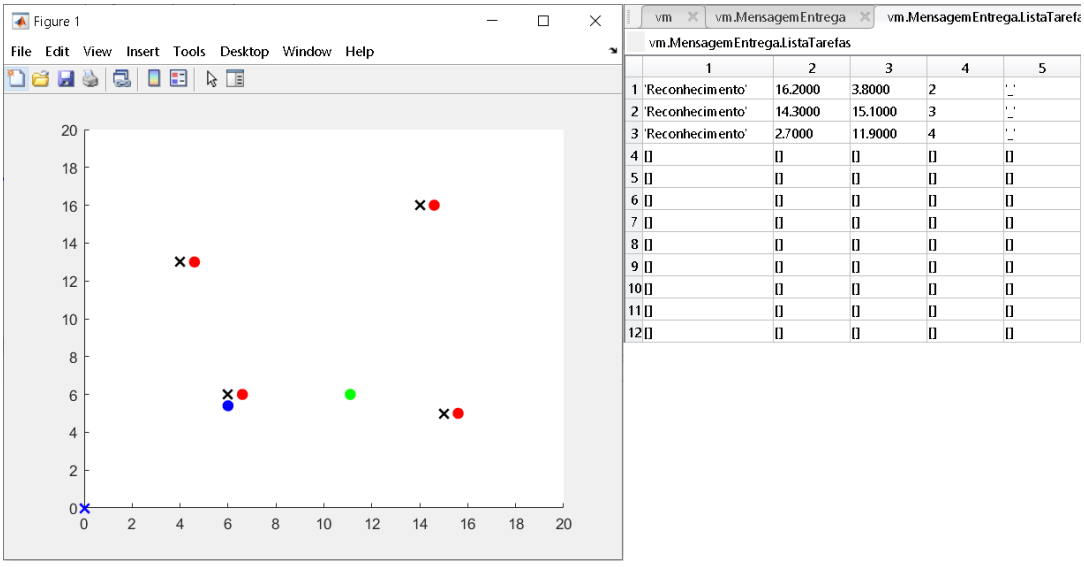


Figura 7.14: Veículo mensageiro a transportar tarefas para entregar em b_2, b_3 e b_4.

Chegando a uma base (b_2, b_3 ou b_4), sempre que o *vm* se encontre próximo (m_b_proximo), envia uma mensagem para essa base com a lista de tarefas para entrega. Essa base por sua vez vai processar essa lista e retornar uma mensagem_m com a lista de tarefas para entrega atualizada, que o *vm* guarda novamente em MensagemEntrega, e ainda uma mensagem_b com a lista de tarefas finalizadas se existir alguma (caso contrário envia uma lista vazia), que o *vm* guarda na sua estrutura em MensagemFinalizadas.

Na figura 7.15, apresentada de seguida, é possível ver a lista de espera das bases b_2, b_3 e b_4 após a passagem do veículo mensageiro por estas com a lista de entrega. Cada base contém uma tarefa que corresponde à sua área de trabalho em lista de espera.

b2_lt_espera

20x5 cell

	1	2	3	4	5
1	'Reconheci...	16.2000	3.8000	2	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

b3_lt_espera

20x5 cell

	1	2	3	4	5
1	'Reconheci...	14.3000	15.1000	3	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

b4_lt_espera

20x5 cell

	1	2	3	4	5
1	'Reconheci...	2.7000	11.9000	4	'_'
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

Figura 7.15: As bases b_2, b_3 e b_4 com tarefas em lista de espera.

Por sua vez a figura 7.16 apresenta um momento em que o veículo mensageiro já passou pelas bases b_2, b_3 e b_4 após terem sido executadas tarefas pelos veículos alocados a estas bases. Como se pode verificar à direita o *vm* já transporta consigo uma lista com as tarefas finalizadas na b_2, b_3 e b_4.

Por fim quando o veículo mensageiro retorna à b_1 (m_b1_proximo) atualiza a lista b1_lt_finalizadas com as tarefas finalizadas pelas restantes bases (lt_finalizadas) que corresponde à lista transportada pelo *vm* em MensagemFinalizadas, como se pode verificar na figura 7.17.

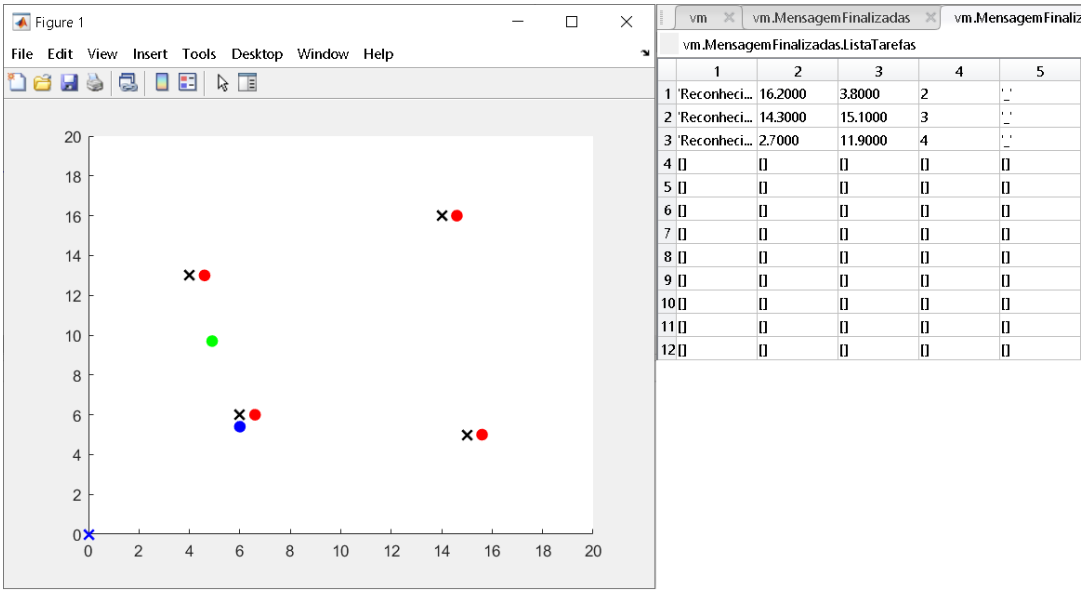


Figura 7.16: Veículo mensageiro a transportar uma lista de tarefas já finalizadas.

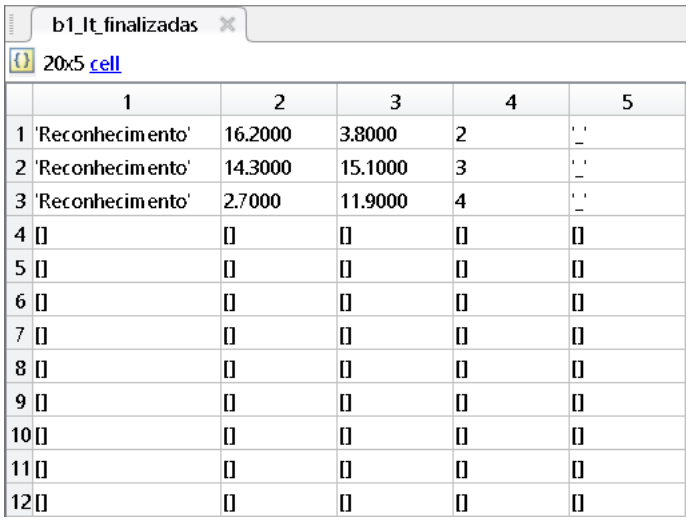


Figura 7.17: Lista de tarefas finalizadas da b_1 atualizada com tarefas finalizadas de outras bases.

7.4.7 Concluir tarefas

O objetivo a cumprir nesta etapa é fazer chegar ao operador, através da base remota, a informação das tarefas que foram concluídas. Quando o *vbr* se encontra junto da b_1 (vb1_proximo) e existe informação guardada na lista de tarefas finalizadas (b1_lt_finalizadas) é apenas necessário esperar que o *vm* também chegue próximo da b_1 (m_b1_proximo), para garantir que quando o *vbr* inicia a sua deslocação em direção à b_r leva consigo a lista de tarefas finalizadas mais atualizada possível. Quando estas condições são reunidas a b_1 envia uma mensagem_b1 com a b1_lt_finalizadas para o *vbr* transportar até à b_r. Na figura 7.18 é possível ver o estado da simulação precisamente quando

o *vbr* se encontra a transportar uma mensagem em direção à b_r, com (à direita) a lista de tarefas finalizadas guardada, e os atuais estados das listas de tarefas enviadas e concluídas da b_r.

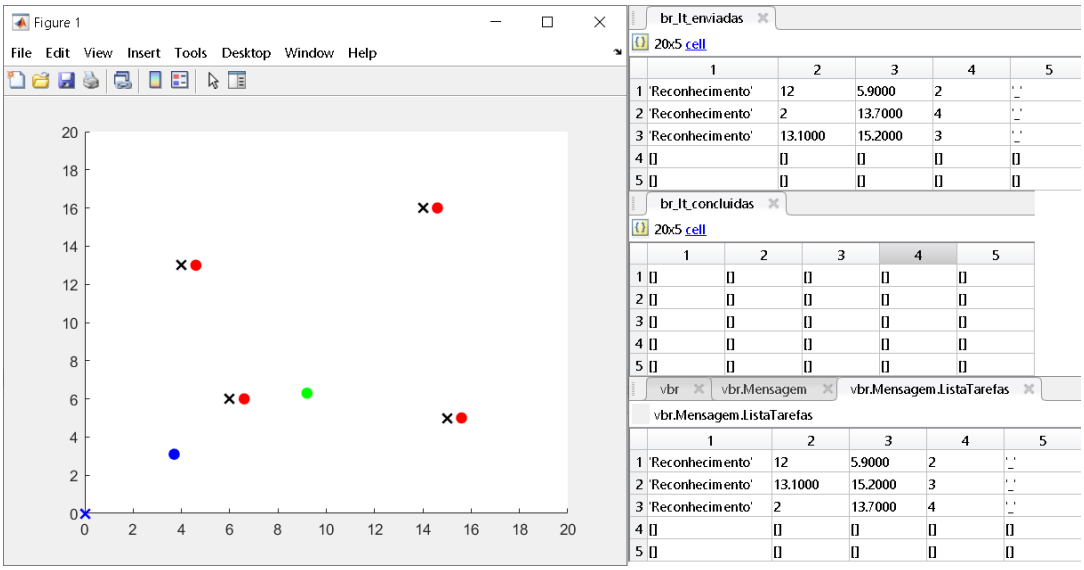


Figura 7.18: *Vbr* a deslocar-se em direção à b_r e as atuais listas de tarefas enviadas e concluídas.

Quando o *vbr* chega à base remota (vbr_proximo) envia a mensagem transportada com a lista de tarefas finalizadas. A base remota por sua vez atualiza as listas de tarefas enviadas, retirando as tarefas presentes na mensagem enviada, e concluídas, acrescentando as mesmas tarefas, como se verifica na figura 7.19.

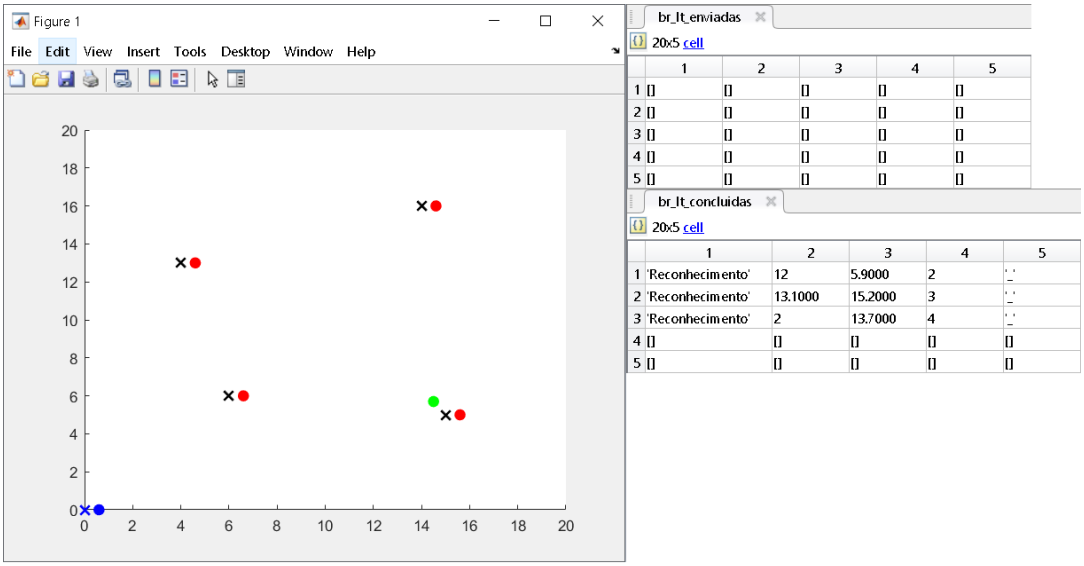
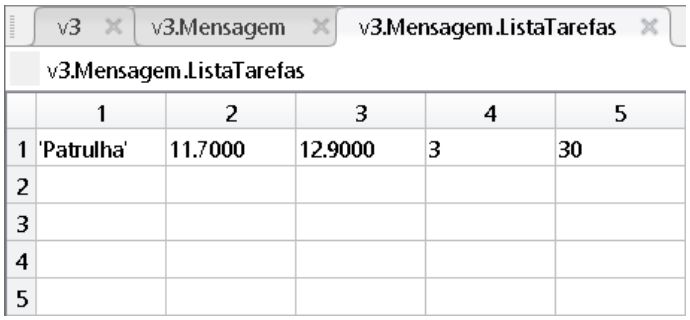


Figura 7.19: Chegada de *vbr* à b_r e atualização das listas de tarefas enviadas e concluídas.

7.4.8 Patrulha

Após a conclusão das etapas anteriores a rede estava já configurada dentro dos objetivos pretendidos, no entanto apenas executava um tipo de tarefa que era o de reconhecimento. O objetivo desta etapa era então implementar um novo tipo de tarefa, o de patrulha.

Sendo assim, de acordo com o exemplo deste caso de estudo, fez-se chegar à b_3 uma tarefa do tipo patrulha. Uma vez que esta tarefa está guardada em b3_lt_espera e há um veículo próximo desta base (b3_vproximo) envia-se uma mensagem_executar da base para o v_3, que este guarda na sua estrutura como se vê na figura 7.20. A lista de espera (b3_lt_espera) é atualizada nesse mesmo instante retirando a tarefa guardada na mensagem_executar.


v3 | v3.Mensagem | v3.Mensagem.ListaTarefas

v3.Mensagem.ListaTarefas

	1	2	3	4	5
1	'Patrulha'	11.7000	12.9000	3	30
2					
3					
4					
5					

Figura 7.20: Mensagem com uma tarefa de patrulha guardada no v_3.

A figura 7.21 representa dois momentos no tempo em que o v_3 se encontra a executar a tarefa de patrulha. A execução desta tarefa implica a deslocação num primeiro instante do veículo até a um ponto próximo do ponto de interesse identificado pelo operador, sendo que depois o veículo desloca-se em torno desse mesmo ponto de interesse durante o tempo que o operador definiu. Quando esse tempo chega ao fim o veículo retorna à base à qual está alocado.

Após terminar a execução da tarefa de patrulha o v_3 retorna à sua respetiva base (b3_vproximo) e envia uma mensagem que contém a tarefa que este acabou de executar. A base por sua vez atualiza a lista de tarefas finalizadas (b3_lt_finalizadas) com a tarefa que recebeu por mensagem, como se verifica na figura 7.22.

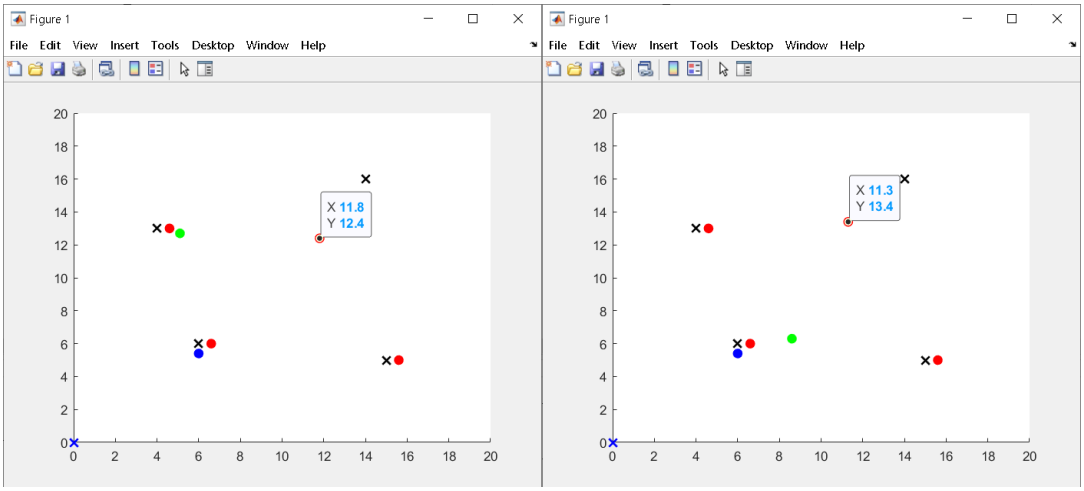


Figura 7.21: Dois instantes distintos do v_3 a realizar uma tarefa de patrulha.

b3_lt_finalizadas

20x5 cell

	1	2	3	4	5
1	'Patrulha'	11.7000	12.9000	3	30
2	[]	[]	[]	[]	[]
3	[]	[]	[]	[]	[]
4	[]	[]	[]	[]	[]
5	[]	[]	[]	[]	[]

Figura 7.22: Lista de tarefas finalizadas da b_3 atualizada após v_3 terminar a tarefa de patrulha.

7.4.9 Mapeamento

No seguimento da etapa anterior, esta etapa foi definida para implementar a possibilidade dos veículos da rede de executarem uma tarefa de mapeamento, sendo este o último dos três tipos de tarefa implementados e sendo, ainda, esta a última etapa do plano de experimentação.

A tarefa de mapeamento difere das tarefas de reconhecimento e patrulha principalmente pela forma como se avalia a sua área de trabalho. Nesta tarefa o operador define as coordenadas de dois pontos distintos, como se vê no exemplo da figura 7.23. A execução da tarefa por parte do veículo consiste em deslocar-se até ao primeiro ponto definido pelo utilizador e depois este veículo deve movimentar-se, de forma cíclica e com intervalos de uma unidade de distância, cobrindo a área desde o 1º ponto até ao 2º ponto definido pelo utilizador. Esta área é assumida como um retângulo, ou um quadrado. Adicionalmente e devido à implementação realizada a tarefa de mapeamento apenas funciona se as coordenadas dos pontos forem arredondados à unidade, não sendo possível utilizar unidades decimais.

Sendo assim, pegando no exemplo da figura 7.23, a área de mapeamento atravessa quatro áreas

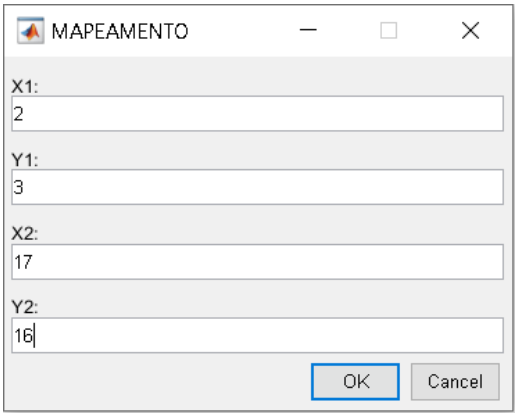


Figura 7.23: Janela de especificação de uma tarefa do tipo mapeamento.

de trabalho diferentes. Como tal esta tarefa após ser definida pelo operador corre um algoritmo que a divide em quatro tarefas diferentes devidamente adequadas a cada área de trabalho. A figura 7.24 representa o momento em que o *vbr* leva consigo uma lista de tarefas guardada após a divisão da tarefa de mapeamento inicialmente definida pelo operador. Como se verifica existem quatro tarefas diferentes nessa lista, sendo que cada uma delas deve ser entregue na sua respetiva base. A seguinte figura 7.25 representa o momento em que os quatro veículos, v_1, v_2, v_3 e v_4, estão a executar a tarefa de mapeamento em simultâneo. Isto sucede após as quatro tarefas terem sido entregues nas bases (b_lt_espera atualizada) e posteriormente entregues aos veículos que aí estavam alocados (b_vproximo) para as executar (com uma mensagem_executar), como podemos ver à direita na figura. As listas de espera foram também atualizadas retirando das mesmas as respetivas tarefas a executar pelos veículos.

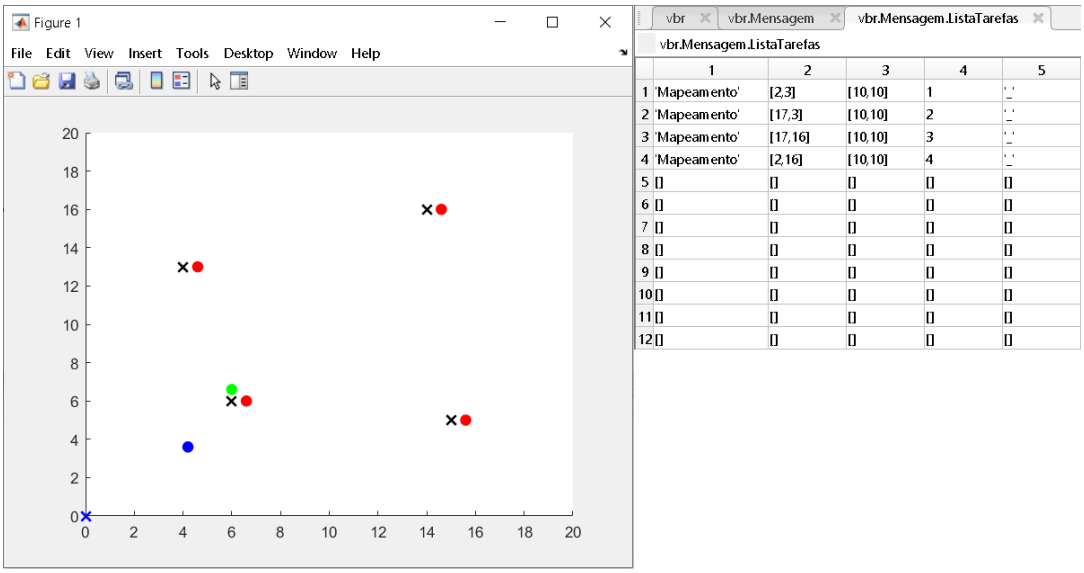


Figura 7.24: *Vbr* a transportar tarefas de mapeamento após divisão da tarefa original.

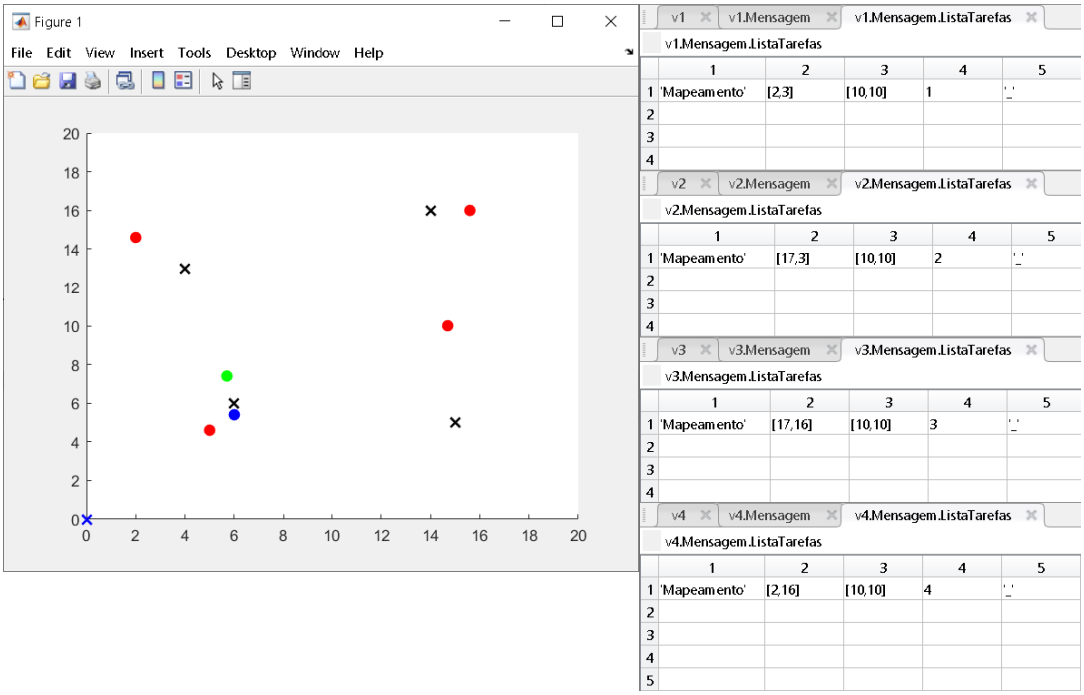


Figura 7.25: Momento em que v_1, v_2, v_3 e v_4 executam uma tarefa de mapeamento em simultâneo.

Como nas restantes tarefas, após retornarem às suas respetivas bases (b_vproximo) os veículos enviam uma mensagem com a tarefa que acabaram de executar para a base atualizar a sua lista de tarefas finalizadas. A figura 7.26 demonstra um momento após a execução das quatro partes da tarefa de mapeamento em que o *vm* já recolheu duas das quatro tarefas finalizadas, na b_2 e b_3, atualizando a b1_lt_finalizadas. Esta lista está a ser transportada pelo *vbr* para a b_r para atualizar a lista de tarefas concluídas. Existe ainda uma tarefa de mapeamento finalizada na b4_lt_finalizadas pois após a primeira recolha de tarefas finalizadas do *vm* o v_4 ainda não tinha retornado à b_4.

Por fim, depois do *vm* recolher a última parte da tarefa de mapeamento finalizada em b_4, e esta ser transportada pelo *vbr* até à b_r, é atualizada a lista de tarefas concluídas sendo que passa a ter guardadas as quatro tarefas de mapeamento correspondentes a cada área de trabalho, como se verifica na figura 7.27. Estas tarefas são retiradas da lista de tarefas enviadas.

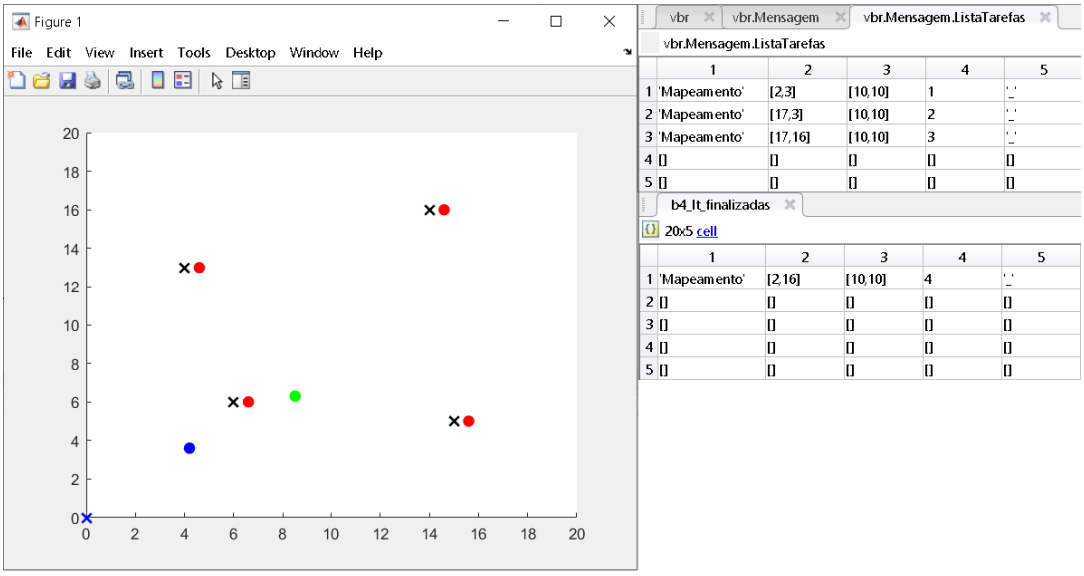


Figura 7.26: *Vbr* a transportar tarefas de mapeamento finalizadas. Uma parte da tarefa original ainda na b4_lt_finalizadas.

br_lt_concluidas | br_lt_enviadas

20x5 cell

	1	2	3	4	5
1	'Mapeame...	[2,3]	[10,10]	1	'_'
2	'Mapeame...	[17,3]	[10,10]	2	'_'
3	'Mapeame...	[17,16]	[10,10]	3	'_'
4	'Mapeame...	[2,16]	[10,10]	4	'_'
5	[]	[]	[]	[]	[]

Figura 7.27: Tarefas de mapeamento na br_lt_concluidas após terem sido executadas.

7.5 Discussão dos resultados

Numa primeira análise global, o sistema implementado cumpriu com os objetivos especificados. Tal como era pretendido a rede consegue estabelecer uma ponte de comunicação entre os seus diversos componentes, que se coordenam entre si de forma automatizada, apenas com o "input"do operador ao nível da base remota. Do ponto de vista do operador, que apenas teria acesso à informação presente na base remota, foi cumprido o objetivo de especificar as tarefas e receber no fim a informação devida das tarefas concluídas. Ao nível da rede todos os veículos e bases cumpriram o seu propósito e não tiveram qualquer problema ao nível das suas interações.

Um dos pontos negativos, onde os resultados não foram totalmente satisfatórios, foi ao nível das tarefas do tipo de mapeamento uma vez que, devido à implementação, o operador é limitado a especificar pontos com coordenadas arredondadas à unidade, o que diminui a precisão do tipo tarefa que se quer executar. Na tarefa do tipo de patrulha o tempo definido pelo operador não

corresponde com exatidão ao tempo em segundos que o veículo executa a tarefa devido em parte à implementação bem como aos atrasos a correr os ciclos no próprio *MATLAB*. É também possível especificar valores nas tarefas fora da área de trabalho de toda a rede, o que não dá os resultados esperados uma vez que se encontra fora do mundo da simulação.

Ainda assim, tendo em conta o propósito desta implementação, os objetivos cruciais para o funcionamento da rede e coordenação dos componentes, definidos para cada etapa do plano de experimentação, foram alcançados.

Capítulo 8

Conclusões

8.1 Trabalho desenvolvido

A partir do momento em que o tema desta dissertação me foi atribuído entrei em contacto com o meu orientador para elaborar um plano a cumprir para a elaboração da dissertação. De forma sucinta este plano consistia em realizar um estudo aprofundado do domínio do problema, passando de seguida para o desenvolvimento do projeto e finalmente concluir o trabalho com a escrita da dissertação.

Para aprofundar o domínio do problema foi inicialmente realizado um trabalho de pesquisa simples para melhor compreender o objetivo do trabalho a realizar nesta tese, de modo a permitir fazer uma primeira definição de especificações da rede de veículos a coordenar e controlar. Para concluir esta fase do trabalho foi feito depois um aprofundamento do estado de arte, com uma pesquisa sobre sistemas híbridos, mecanismos de cooperação e coordenação em redes com diversos veículos e sistemas sem estruturas de comunicação. Após esta pesquisa foi possível realizar ajustes em algumas especificações da rede e compreender melhor como abordar o problema e obter uma boa solução para o mesmo.

Analisando o desenvolvimento do projeto este foi, à priori, divido em três partes distintas: a abordagem ao problema e modelação da rede; a implementação computacional; e a análise de casos de estudo e os resultados obtidos.

A modelação da rede foi desenvolvida com base em máquinas de estado sendo que cada um dos diferentes componentes da rede se enquadrada numa destas máquinas de estado. Para complementar esta modelação foram criadas tabelas que descreviam as transições de estado de cada um dos sistemas dos componentes da rede em função das condições de transição e que ações isso despoletava. Com esta modelação deu para compreender qual a função que cada um dos componentes tinha de cumprir para coordenar a rede e contribuir assim para cumprir os objetivos de funcionamento da rede como um todo.

A implementação computacional acabou por sofrer uns percalços e alterações em relação ao plano previamente delineado, acabando por ser tomada a decisão de se fazer a implementação na

linguagem *MATLAB*. Apesar disso foi possível realizar a implementação de acordo com o planeado de modo com a elaboração de uma rede, composta pelos diversos componentes a interagir entre si de modo a coordenar o seu funcionamento. Após o termino e posterior análise do código implementado, comparando com a modelação que tinha sido realizada, é possível compreender que nem sempre todos os pormenores são implementados como tinha sido pensado na modelação do problema no entanto a modelação não deixa de ter uma grande importância para a implementação pois ajuda a compreender o problema, aprofundar o tema, fornece uma linha guia para como passar para código a solução do problema e evita que muitos erros sejam cometidos no código.

Para terminar o trabalho desenvolvido foram analisados casos de estudo da implementação realizada. Estes casos de estudo foram divididos em etapas e alguns deles foram sendo avaliados em paralelo com a implementação permitindo com isto fazer um *debugging* dos erros que existiam na implementação para por fim obter os seus resultados. Os resultados obtidos tiveram por base um ambiente de simulação desenvolvido e ainda a análise do valor de variáveis importantes para o funcionamento da rede. Estes resultados foram positivos uma vez que cumpriam com o geral dos objetivos delineados para o funcionamento da rede.

8.2 Trabalho futuro

O trabalho realizado teve em vista a implementação de uma rede que fosse escalável, o que abre sempre espaço para trabalho futuro a ser desenvolvido para complementar a implementação já existente. Para além disso nem todos os pontos da implementação foram de encontro ao que tinha sido pensado e modelado para o funcionamento da rede. Será, então, possível tornar a rede mais flexível se forem acrescentadas algumas funcionalidades que complementem o seu funcionamento.

Ficam, portanto, alguns tópicos sugeridos como trabalho futuro:

- Escalar a rede para abranger uma área de trabalho maior, aumentando o número de bases e veículos em funcionamento.
- Aumentar também o número de veículos mensageiros em circulação de forma intercalada, tornando assim mais eficiente a comunicação entre os diversos componentes da rede.
- Alocar mais do que um veículo a cada base, para permitir a execução de diversas tarefas em paralelo na mesma área de trabalho.
- Assumindo a execução do tópico anterior, seria interessante incluir um algoritmo que permitisse mudar a alocação de um veículo a uma certa base de modo a compensar a diferença de carga de trabalho entre as bases da rede.
- Incrementar o tipo de tarefas a executar pela rede, aumentando a sua funcionalidade e flexibilidade.

Finalmente, importa referir que o modelo e a abordagem utilizada pode acomodar com facilidade a consideração de aplicações em que, para além da troca de mensagens, o sistema permite

também a troca de combustível entre bases. Nesse caso, os veículos transportam também combustível para além de mensagens. O nível de combustível em cada base é mais uma variável interna.

8.3 Considerações finais

Todo o trabalho envolvido na realização da dissertação constitui uma nova experiência, sendo um projeto diferente de todos os restantes que havia realizado ao longo do meu percurso académico. A forma como o trabalho foi desenvolvido, as interações com o professor orientador, a definição de objetivos e prazos para os cumprir, entre outros aspetos, tornam a dissertação um projeto que dá uma experiência mais aproximada do mundo do trabalho.

Acrescendo a toda a experiência normal de realizar a dissertação, as adversidades relacionadas com o COVID19, bem assim como o meu contexto académico, que me levou a realizar mobilidade no 1º semestre e iniciar a dissertação com algum atraso, fez com que eu tivesse que aperfeiçoar as capacidades de tomada de decisão em momentos adversos e sentido de orientação para o planeamento de projetos em termos de horários e objetivos a cumprir. Como tal, toda a aprendizagem relacionada com o desenvolvimento da dissertação ajudará no meu futuro, quer a nível pessoal, quer profissional.

Anexo A

Anexos

Fica em anexo o link para o repositório do github onde o código implementado ao longo da dissertação foi guardado.

https://github.com/Antonio-Rocha/teseMSC

Referências

[1] James Bellingham e Kanna Rajan. Robotics in remote and hostile environments. *Science*, 318:1098–1102, 2007.

[2] J. Borges de Sousa. Veículos não tripulados para observação do oceano. *Gazeta Física*, Maio 2020.

[3] Pierre F.J. Lermusiaux, Tapovan Lolla, Patrick J. Haley Jr., Konuralp Yigit, Mattheus P. Ueckermann, Thomas Sondergaard, e Wayne G. Leslie. Science of autonomy: Time-optimal path planning and adaptive sampling for swarms of ocean vehicles. Em Tom Curtin, editor, *Springer Handbook of Ocean Engineering: Autonomous Ocean Vehicles, Subsystems and Control*, páginas 481–498. Springer, 2016.

[4] Trygve O. Fossum, Glaucia M. Fragoso, Emlyn Davies, Jenny E. Ullgren, Renato Mendes, Geir Johnsen, Ingrid Ellingsen, Jo Eidsvik, Martin Ludvigsen, e Kanna Rajan. Toward adaptive robotic sampling of phytoplankton in the coastal ocean. *Science Robotics*, 4:1–20, 2019.

[5] M. J. Costa, J. Pinto, P. S. Dias, J. Pereira, K. Lima, M. Ribeiro, J. B. Sousa, T. Lukaczyk, R. Mendes, M. P. Tomasino, C. Magalhães, I. Belkin, F. Lopez-Castejon, J. Gilabert, K. Skarpnes, M. Ludvigsen, K. Rajan, Z. Mirmalek, e A. Chekalyuk. Field report: Exploring fronts with multiple robots. Em *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, páginas 1–7, 2018.

[6] M. Thammawichai, S. P. Baliyarasimhuni, E. C. Kerrigan, e J. B. Sousa. Optimizing communication and computation for multi-uav information gathering applications. *IEEE Transactions on Aerospace and Electronic Systems*, 54(2):601–615, 2018.

[7] João P. Hespanha. Lecture on hybrid control and switched systems. Relatório técnico, Univesity of California at Santa Barbara.

[8] J. Delande, R. Balasubramanian, e C. Duarte. Task consensus among a team of heterogeneous auvs under intermittent communication. Em *2013 OCEANS - San Diego*, páginas 1–4, 2013.

[9] S. Kemna, J. G. Rogers, C. Nieto-Granda, S. Young, e G. S. Sukhatme. Multi-robot coordination through dynamic voronoi partitioning for informative adaptive sampling in communication-constrained environments. Em *2017 IEEE International Conference on Robotics and Automation (ICRA)*, páginas 2124–2130, 2017.

[10] T. Matsuda, T. Maki, Y. Sato, e T. Sakamaki. Performance verification of the alternating landmark navigation by multiple auvs through sea experiments. Em *OCEANS 2015 - Genova*, páginas 1–9, 2015.

[11] T. Matsuda, T. Maki, Y. Sato, e T. Sakamaki. Experimental evaluation of accuracy and efficiency of alternating landmark navigation by multiple auvs. *IEEE Journal of Oceanic Engineering*, 43(2):288–310, 2018.

[12] A. Abad, N. DiLeo, e K. Fregene. Decentralized model predictive control for uuv collaborative missions. Em *OCEANS 2017 - Anchorage*, páginas 1–6, 2017.

[13] Gul Agha. Actors: A model of concurrent computation in distributed systems. 10 2004.