MARCO VINÍCIUS MUNIZ FERREIRA

Aplicação da iPNRD com feromônio de formiga para controle autônomo e distribuído de robôs móveis.

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA MECÂNICA
2020

MARCO VINÍCIUS MUNIZ FERREIRA

Aplicação da iPNRD com feromônio de formiga para controle autônomo e distribuído de robôs móveis.

Tese apresentada ao Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia, como parte dos requisitos para obtenção do título de **DOUTOR EM ENGENHARIA MECÂNICA.**

Área de concentração: Mecânica dos Sólidos e Vibrações

Orientador: Prof. Dr. Marcus Antônio Viana Duarte
Coorientador: Prof. Dr. José Jean-Paul Zanlucchi de Souza Tavares

UBERLÂNDIA - MG
2020

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

F383 Ferreira, Marco Vinicius Muniz, 1988-
2020 Aplicação da iPNRD com feromônio de formiga para controle autônomo e distribuído de robôs móveis. [recurso eletrônico] / Marco Vinicius Muniz Ferreira. - 2020.

Orientador: Marcus Antônio Viana Duarte.
Coorientador: José Jean-Paul Zanlucchi de Souza Tavares.
Tese (Doutorado) - Universidade Federal de Uberlândia, Pós-graduação em Engenharia Mecânica.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.te.2020.24
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia mecânica. I. Duarte, Marcus Antônio Viana,1959-, (Orient.). II. Tavares, José Jean-Paul Zanlucchi de Souza,72 -, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Engenharia Mecânica. IV. Título.

CDU: 621

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:
Gizele Cristine Nunes do Couto - CRB6/2091
Nelson Marcos Ferreira - CRB6/3074

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Engenharia Mecânica
Av. João Naves de Ávila, nº 2121, Bloco 1M, Sala 212 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4282 - www.posgrad.mecanica.ufu.br - secposmec@mecanica.ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Engenharia Mecânica				
Defesa de:	Tese de Doutorado, 288, COPEM				
Data:	17/02/2020	Hora de início:	14:00	Hora de encerramento:	17:14
Matrícula do Discente:	11523EMC016				
Nome do Discente:	Marco Vinícius Muniz Ferreira				
Título do Trabalho:	Aplicação de iPNRD com Feromônio de Formiga para Controle Autônomo e Distribuído de Robôs Móveis				
Área de concentração:	Mecânica dos Sólidos e Vibrações				
Linha de pesquisa:	Projetos de Sistemas Mecânicos				
Projeto de Pesquisa de vinculação:					

Reuniu-se no Sala 1M206 - Bloco 1M, Campus Santa Mônica, da Universidade Federal de Uberlândia, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Engenharia Mecânica, assim composta: Professores Doutores: José Jean-Paul Zanlucchi de Souza Tavares (co-orientador) - FEMEC/UFU; Rogério Sales Gonçalves - FEMEC/UFU; Gina Maira Barbosa de Oliveira - FACOM-UFU; João Carlos dos Santos Basílio - UFRJ; Dennis Brandão - USP e Marcus Antônio Viana Duarte - FEMEC/UFU orientador(a) do(a) candidato(a).

Iniciando os trabalhos o(a) presidente da mesa, Dr(a). Marcus Antônio Viana Duarte, apresentou a Comissão Examinadora e o candidato(a), agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor(a) presidente concedeu a palavra, pela ordem sucessivamente, aos(às) examinadores(as), que passaram a arguir o(a) candidato(a). Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o(a) candidato(a):

Aprovado(a).

Esta defesa faz parte dos requisitos necessários à obtenção do título de Doutor.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Marcus Antonio Viana Duarte**, **Professor(a) do Magistério Superior**, em 17/02/2020, às 17:23, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **José Jean Paul Zanlucchi de Souza Tavares**, **Professor(a) do Magistério Superior**, em 17/02/2020, às 17:24, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rogerio Sales Gonçalves**, **Professor(a) do Magistério Superior**, em 17/02/2020, às 17:25, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Gina Maira Barbosa de Oliveira**, **Professor(a) do Magistério Superior**, em 17/02/2020, às 17:26, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **JOAO CARLOS S BASILIO**, **Usuário Externo**, em 17/02/2020, às 17:26, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Dennis Brandão**, **Usuário Externo**, em 17/02/2020, às 17:29, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **1869934** e o código CRC **9A145B45**.

Referência: Processo nº 23117.010166/2020-84 SEI nº 1869934

MARCO VINÍCIUS MUNIZ FERREIRA

APLICAÇÃO DA IPNRD COM FEROMÔNIO DE FORMIGA PARA CONTROLE AUTÔNOMO E DISTRIBUÍDO DE ROBÔS MÓVEIS.

Tese **APROVADA** pelo Programa de Pós-graduação em Engenharia Mecânica da Universidade Federal de Uberlândia.

Área de concentração: Mecânica dos Sólidos e Vibrações

Banca Examinadora:

_____

Prof. Dr. Marcus Antônio Viana Duarte – UFU – Orientador

_____

Prof. Dr. José Jean Paul Zanlucchi de Souza Tavares – UFU – Coorientador

_____

Prof. Dr. Rogério Sales Gonçalves– UFU

_____

Prof. Dra. Gina Maira Barbosa de Oliveira – UFU

_____

Prof. Dr. Dennis Brandão – USP

_____

Prof. Dr. João Carlos dos Santos Basílio– UFRJ

Uberlândia, 17 de fevereiro de 2020

AGRADECIMENTOS

Aos meus pais, às minhas irmãs e à minha avó pelo carinho, amor, compreensão e incentivo para a conclusão deste projeto.

Ao meu orientador José Jean-Paul por acreditar na relevância do tema, pelo apoio técnico, científico e emocional, pelos ensinamentos e pelos conselhos que em muito auxiliaram o meu desenvolvimento acadêmico e pessoal.

À Universidade Federal de Uberlândia, à Faculdade de Engenharia Mecânica e ao Programa de Pós-graduação em Engenharia Mecânica por acreditarem neste trabalho, pelo espaço físico, materiais e equipamentos disponibilizados para a concretização do projeto.

À CAPES pelo apoio financeiro oferecido durante a realização deste trabalho.

Aos colegas do *Manufacturing Automated Planning Lab* (MAPL), especialmente ao João Paulo pelo auxílio na implementação do ambiente computacional em V-REP, ao Carlos Eduardo pelo suporte na implementação das bibliotecas PNRD e iPNRD, ao Gustavo Umezaki pelas ideias durante o projeto do robô móvel e aos demais pelo companheirismo, pela amizade e pelos momentos de descontração.

Aos colegas de trabalho Fábio e Rogério que contribuíram com inúmeras ideias para projetar a bancada experimental e sempre estimularam a conclusão deste projeto.

E a tantos outros que de alguma forma contribuíram para a conclusão deste trabalho.

FERREIRA, M.V.M. **Aplicação da iPNRD com feromônio de formiga para controle autônomo e distribuído de robôs móveis.** 2020. 130 f. Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia.

Resumo

A robótica de enxame possui vários desafios em relação ao controle do comportamento social quando objetivos específicos precisam ser alcançados. Abordagens de inspiração biológica, tal como algoritmos de colônia de formigas, estão no topo das soluções que não utilizam inteligência artificial. Nestes algoritmos as informações locais espalhadas no ambiente são usadas em vez de um banco de dados local centralizado. Esta tese propõe uma solução para o controle autônomo e distribuído de robôs terrestres através da emulação do feromônio de formiga. Para tal, a abordagem iPNRD, que utiliza a base de dados dos elementos da RFID baseada na estrutura de dados forma de uma rede de Petri é implementada. Nesta abordagem o vetor de disparos é salvo nas *tags* do ambiente, enquanto o vetor de marcação e a matriz de incidência estão salvos no leitor do robô. Durante a aquisição de dados da etiqueta o robô atualiza seu vetor de marcação, alterando seu comportamento. A comunicação robô-ambiente é realizada através de *pseudoboxes,* que representam informações de controle externas aos componentes das redes de Petri. Para validação experimental um robô baseado em formigas cortadeiras (*Attabot*) é projetado, e para o ambiente de simulação a plataforma V-REP foi utilizada. Teste operacionais indicam que o método proposto é capaz de coordenar os robôs terrestres para executar uma tarefa específica sem que haja qualquer tipo de controle centralizado. Para isso, é necessário que o ambiente seja modelado e que possua inteligência.

Palavras-chave: Feromônio de formiga. Robôs móveis. Arquitetura de controle. iPNRD.

FERREIRA, M.V.M. iPNRD with ant pheromone applied to mobile robots' autonomous and distributed control**.** 2020. 130 f. Ph.D. Thesis, Universidade Federal de Uberlândia, Uberlândia.

Abstract

Swarm robotics has several challenges in controlling social behavior when specific goals need to be achieved. Biologically inspired approaches, such as ant colony algorithms, are at the top of solutions that do not use artificial intelligence. In these algorithms local information scattered around the environment is used instead of a centralized local database. This thesis proposes a solution for the autonomous and distributed control of terrestrial robots through ant pheromone emulation. To this end, the iPNRD approach, which uses the data structure-based RFID element database as a Petri net is implemented. In this approach the trigger vector is saved in the environment tags, while the marking vector and the incidence matrix are saved in the robot reader. During tag data acquisition the robot updates its marking vector, changing its behavior. Robot-environment communication is performed through pseudoboxes, which represent control information external to the components of Petri nets. For experimental validation, a leaf cutting ant-based robot (Attabot) is designed, and for the simulation environment the V-REP platform was used. Operational tests indicate that the proposed method can coordinate ground robots to perform a specific task without any centralized control. This requires that the environment be modeled and have intelligence.

Keywords: Pheromone of ants. Mobile robots. Control architecture. iPNRD.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

A	Matriz de incidência de uma rede de Petri
A^t	Função transporta da matriz de incidência A
E	Matriz (associada à função I) que identifica o conjunto de lugares de entrada para determinada transição de uma rede de Petri
I	Uma função de entrada que identifica o conjunto de lugares de entrada para determinada transição de uma rede de Petri
i	operador matricial associado aos lugares de uma rede de Petri
j	operador matricial associado às transições de uma rede de Petri
k	instante associado a determinada marcação em uma rede de Petri
M	Marcação ou estado da rede de Petri
M_k	Marcação ou estado da rede de Petri no instante k
M_{k+1}	Marcação ou estado da rede de Petri no instante $k+1$
m	quantidade de lugares associados a uma rede de Petri
n	quantidade de transições associadas a uma rede de Petri
O	Uma função de saída que identifica o conjunto de lugares de saída para determinada transição de uma rede de Petri
P	Um conjunto finito com m lugares relacionados a uma rede de Petri
P_i	i-ésimo lugar em uma rede de Petri
R	Uma tupla de cinco elementos que descrevem uma rede de Petri
S	Matriz (associada à função O) que identifica o conjunto de lugares de saída para determinada transição de uma rede de Petri
T	Um conjunto finito com n transições relacionadas a uma rede de Petri
T_j	j-ésima transição em uma rede de Petri
u_k	vetor de disparo associado ao instante k

LISTA DE ABREVIATURAS

ABS	Acrilonitrila Butadieno Estireno
CAD	*Computer Aided Design*
DC	*Direct Current*
DIY	*Do it yourself*
DXF	*Drawing Exchange Format*
FDM	*Fused Deposition Modeling*
FIFO	*First in First out*
HF	*High Frequency*
IoT	*Internet of Things*
iPNRD	*inverted Petri nets inside RFID distributed database*
LED	*Light-Emitting Diode*
LF	*Low Frequency*
MDF	*Medium Density Fiberboard*
NFC	*Near Field Communication*
PN	*Petri Nets*
PNRD	*Petri nets inside RFID distributed database*
PVC	Policloreto de vinila
RFID	*Radio Frequency Identification*
STL	*Stereolithography*
UHF	*Ultra High Frequency*
V-REP	*Virtual Robot Experimentation Platform*

SUMÁRIO

CAPÍTULO I

INTRODUÇÃO

Segundo Jaulin (2019), a robótica móvel pode ser definida como um sistema mecânico capaz de mover-se em um ambiente de forma autônoma. Para esse propósito, os robôs móveis precisam ser equipados com sensores, que coletam o conhecimento de seu entorno; atuadores, que permitem a movimentação; e inteligência, que permite calcular os comandos dos atuadores baseado nas informações coletadas dos sensores.

Atualmente existem iniciativas de veículos autônomos das marcas Tesla Motors (Fig. 1.1a), Apollo, Apple, Waymo, Google, Ford, entre outras. Além disso, existem iniciativas de enxame de drones (Fig. 1.1b) no ramo militar e no ramo de entretenimento. Sendo assim, percebe-se a tendência de se criar um sistema colaborativo de veículos autônomos sejam terrestres, aéreos, aquáticos etc.

(a) (b)

Figura 1.1 – (a) carro autônomo da Tesla Motors e enxame de drones (b).

Os robôs bioinspirados têm o potencial de igualar ou exceder a versatilidade e multifuncionalidade de organismos naturais (Coyle *et al.*, 2018). A biomimética permite o desenvolvimento ou aperfeiçoamento de soluções em engenharia, estimulando novas ideias a partir da compreensão de padrões encontrados na natureza. Por isso, o projeto de robôs bioinspirados requer forte integração de sensores, atuadores e controle.

Yogeswaran e Ponnambalam (2010) classificam os trabalhos referentes a sistemas bioinspirados em quatro aspectos: a inspiração (insetos, mamíferos etc.), o tipo de comunicação (explícita ou implícita), o controle (centralizado ou distribuído), e a tarefa (mapeamento, navegação etc.).

A comunicação implícita é o método de comunicação através do ambiente, conhecida como estigmergia. O feromônio é um tipo de comunicação implícita e funciona como uma mensagem química entre indivíduos (KARLSON; BUTENANDT, 1997).

No controle distribuído, a organização do sistema composto por agentes robóticos é autônoma e o comportamento dos robôs é o ponto interesse a ser analisado. Sendo assim, os ambientes inteligentes são a junção de uma comunicação implícita e um controle distribuído, em que o próprio ambiente é capaz de alterar o comportamento de um robô móvel.

Existem trabalhos com diferentes formas de emular o feromônio: calor (RUSSEL, 1997), álcool (FUJISAWA *et al.*, 2008), tinta fosforescente (MAYET *et al.,* 2010) (RANJBAR-SAHRAEI *et al.,* 2013), diodo emissor de luz (LED) (GARNIER *et al.,* 2007), virtual (LIMA; OLIVEIRA, 2017), identificação por rádio frequência (KIM; KURABAYASHI, 2011) (TANG *et al.*, 2017), entre outras.

De acordo com Glover e Bhatt (2006), RFID (*Radio Frequency Identification*) é a denominação de sistemas eletrônicos capazes de realizar identificações através de comunicação por RF (rádio frequência) utilizando dois elementos básicos desses sistemas, o leitor (*reader*) e as etiquetas eletrônicas de identificação (*tags*).

Por outro lado, a rede de Petri (PETRI, 1962) é uma técnica clássica de modelagem gráfica com fundamentação matemática utilizada na modelagem de eventos discretos e sistemas baseados em conhecimento. Fonseca (2018) afirma que, apesar de bem fundamentada e possuir recursos interessantes para a investigação da utilização de robôs móveis, as redes de Petri em aplicações de robótica móvel ainda são um desafio.

A PNRD (*Petri nets inside RFID distributed database*) (TAVARES; SARAIVA, 2010), ou rede de Petri inserida em base de dados RFID, é uma abordagem que utiliza a base de dados dos elementos da RFID baseada na estrutura de dados formal de uma rede de Petri, ou seja, os componentes da rede de Petri são distribuídos nos leitores e *tags.*

A PNRD auxilia sistemas logísticos e de manufatura, sendo que o foco é a identificação e monitoramento de agentes passivos, tais como peças e itens comerciais. Uma modificação na PNRD, a iPNRD ou PNRD invertida, foi apresentada por Fonseca (2018) a fim de utilizar a mesma abordagem para agentes ativos.

Existe então, a necessidade de um estudo de como a iPNRD pode ser aplicada a enxame de robôs (agentes ativos) com base em feromônio de formigas.

1.1. Objetivos

O principal objetivo da tese consiste em estruturar uma arquitetura de controle distribuída para enxame de robôs terrestres baseada na emulação do feromônio de formiga utilizando iPNRD.

Destacam-se os seguintes objetivos específicos:

1. Projetar e construir um robô móvel de código aberto aplicado ao ambiente de análise;
2. Modelar o comportamento dos agentes em iPNRD;
3. Desenvolver simulação computacional do ambiente de testes;
4. Definir cenários para testes e ensaios funcionais;
5. Realizar testes e ajustes necessários;
6. Avaliar o comportamento do sistema quanto ao controle autônomo e distribuído, limitações e propostas de melhorias.

1.2. Justificativa

Pesquisas científicas relacionadas a feromônio, rede de Petri, RFID e enxame de robôs são bastante escassas. Tang *et al.* (2017) propuseram um método de busca estigmérgico para enxame de robôs em que os agentes se movem pelo ambiente enquanto leem e escrevem etiquetas RFID. Com isso, um mapa de feromônio é gerado para guiar os robôs sem utilização de sistema de localização. Porém, avaliações como esta são realizadas somente em ambiente simulado. Esse é um campo de pesquisa emergente e há necessidade de pesquisas de campo que demandem maiores investigações.

Com o aumento do desenvolvimento de sistemas robóticos aliado à redução no preço de microcontroladores, surge uma carência por sistemas robóticos mais autônomos para enxame de robôs. Grieco *et al*. (2014) descrevem aplicações robóticas auxiliadas por internet das coisas, ou *Internet of Things* (IoT), como uma nova área de pesquisa e descreve suas implicações tecnológicas, levanta o estado da arte e apresenta os problemas em aberto.

As pesquisas em robótica móvel podem assumir um caráter social também. Isso se deve ao fato de que a robótica está cada vez mais inserida no processo de aprendizagem de alunos nas etapas de educação básica. Os alunos aprendem robótica como complemento ao ensino de ciências, matemática e até de português. (MEC, 2016)

Nesse contexto, a popularização da ciência é uma necessidade evidente para difusão do conhecimento científico para um público não especializado. Para isso, faz-se necessário

o desenvolvimento de projetos do tipo *open source* e que possam contribuir com o movimento DIY (*do it yourself)*.

1.3. Estrutura da Tese

Os capítulos que descrevem o desenvolvimento, os resultados e as conclusões estão organizados neste documento da forma descrita a seguir.

No capítulo II, introduzem-se conceitos fundamentais para o entendimento do projeto. Além da robótica móvel na seção 2.1, abordam-se a tecnologia RFID na seção 2.2, as redes de Petri na seção 2.3 e as tecnologias PNRD e iPNRD na seção 2.4.

No capítulo III apresentam-se a metodologia e o desenvolvimento do projeto. Incialmente, na seção 3.1, define-se a metodologia adotada. Na sequência, a seção 3.2 aborda a análise do problema, seguida pela proposta da solução na seção 3.3, pela definição dos cenários avaliados na seção 3.4 e pela modelagem dos agentes em rede de Petri na seção 3.5.

O capítulo IV detalha a implementação experimental na seção 4.1 e a implementação em V-REP na seção 4.2. Alguns pontos importantes são apresentados na seção de testes, a seção 4.3.

O capítulo V mostra os resultados encontrados para o experimento real (seção 5.1), os resultados da simulação computacional (seção 5.2) e as discussões acerca dos resultados encontrados, na seção 5.3.

O capítulo VI pondera as principais conclusões, limitações da solução proposta, sugestões de trabalhos futuros e contribuições associadas a esta tese. Por fim, listam-se as referências bibliográficas utilizadas seguido dos apêndices.

CAPÍTULO II

FUNDAMENTAÇÃO TEÓRICA E REVISÃO DA LITERATURA

Este capítulo expõe uma sucinta revisão bibliográfica dos elementos essenciais para o desenvolvimento deste projeto. A seção 2.1 apresenta as principais definições relacionadas à robótica móvel, abordando temas como enxame de robôs, robôs bio-insipirados e feromônio de formiga. Além disso, os principais simuladores de robótica móvel são apresentados. A tecnologia RFID é discutida na seção 2.2 enquanto a seção 2.3 discute os conceitos e formulações das redes de Petri lugar/transição, temporizadas e coloridas, *pseudoboxes* e aplicações de Rede de Petri em semáforos. Na seção 2.4 a PNRD e a iPNRD, abordagens utilizando RFID e Rede de Petri, são apresentadas, além da biblioteca PNRD para Arduino. Os trabalhos relacionados a essa tese, que compõem o estado da arte, são discutidos ao longo das seções supracitadas.

2.1. Robótica Móvel

Os mecanismos de locomoção mais comuns entre os robôs móveis são as rodas, visto que são mecanismos de simples implementação (SIEGWART; NOURBAKHSH, 2004). A estabilidade de um robô com rodas é garantida quando há três rodas em contato com o solo, porém robôs com apenas duas rodas podem ser estáveis. Para mais de quatro rodas é necessário um sistema de suspensão a fim de garantir que todas as rodas estejam em contato com o solo em terrenos desnivelados.

Os quatro tipos de rodas comumente utilizados são: a roda padrão, a roda castor, a roda sueca e a roda esférica, sendo esta última de difícil construção. A Fig. 2.1 apresenta oito tipos de configurações de robôs móveis utilizando rodas (que podem ser motorizadas ou não). Na configuração (a) temos os robôs com duas rodas diferenciais com centro de massa sob o eixo. Este é o tipo de configuração dos robôs pêndulo invertido que requerem um grande custo computacional para o controle de sua estabilidade. A configuração (b) apresenta os robôs com duas rodas independentes na frente/traseira e uma roda

omnidirecional não motorizada na traseira/frente. Os robôs seguidores de linha de competições de robótica, na maioria das vezes, utilizam essa configuração. A configuração (c) difere-se da anterior uma vez que as rodas diferenciais estão no centro do robô e há um terceiro apoio não motorizado na frente ou traseira do robô. Observa-se que o centro de massa do robô deve estar entre as rodas centrais e o ponto de apoio para garantir que o robô não tombe ou perca o ponto de apoio. A configuração (d) é a última utilizando três pontos de contato com o solo e, por utilizar rodas suecas, torna o robô omnidirecional, isto é, o robô pode partir para qualquer direção ou girar em torno do centro a partir da força resultante (somatória das forças de cada roda) no centro de massa. As rodas estão dispostas em 120º uma das outras. As configurações com quatro pontos de contato com o solo podem ser de diversas formas: duas rodas diferenciais centrais e dois pontos de apoio (e); duas rodas de tração (diferenciais) na frente/traseira e duas rodas omnidirecionais não motorizadas na traseira/frente (f); e quatro rodas omnidirecionais (g). Por fim, a última configuração (h) apresenta um robô com duas rodas de tração (diferenciais) no centro e uma roda omnidirecional em cada canto do robô. Essa configuração garante a estabilidade do robô e é uma alternativa para a configuração (e) quando há a necessidade da utilização de sensores em locais que as rodas de apoio atrapalhariam.

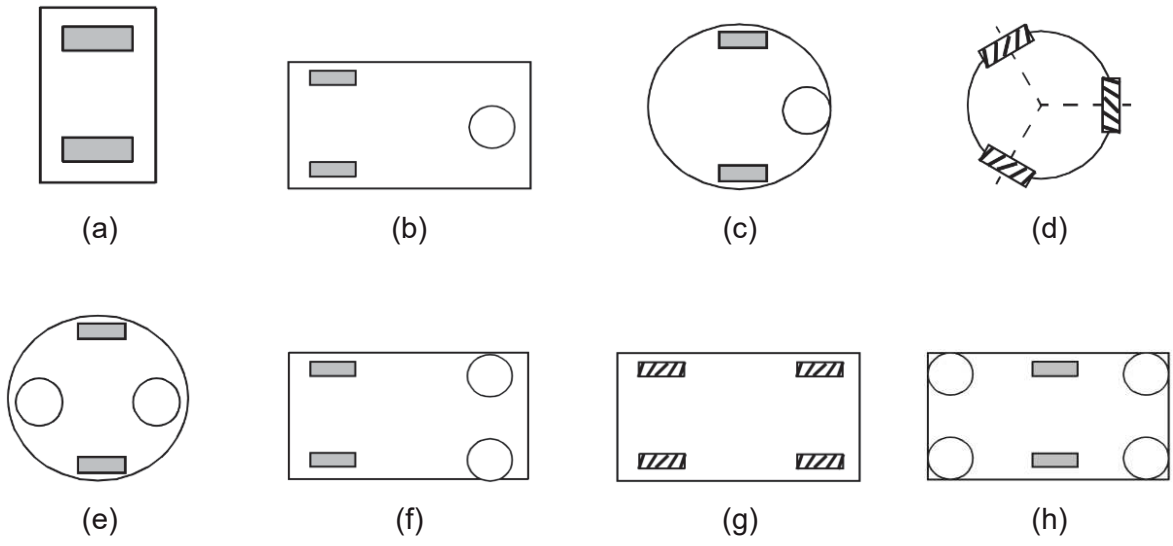


Figura 2.1 – Configurações de alguns robôs móveis equipados com rodas. Adaptado de Siegwart e Nourbakhsh (2004).

2.1.1. Enxame de robôs

Şahin (2005) define que a robótica de enxame é o estudo de como muitos agentes podem ser projetados de forma que um comportamento coletivo desejado surge de interações locais entre os agentes e entre os agentes e o ambiente.

De acordo com Brambilla *et al.* (2013), um conjunto de robôs é considerado como um enxame de robôs quando os mesmos são autônomos; estão situados em um ambiente e podem agir a fim de modificá-lo; suas capacidades de sensoriamento e comunicação são locais; não possuem acesso a um controle centralizado e/ou a um conhecimento global; e cooperam para executar uma tarefa.

A taxonomia dos robôs multi-agentes foi apresentada por Dudek *et al.* (1996) em que os multi-agentes podem ser classificados de acordo com a quantidade de robôs, a distância de comunicação, a topologia da comunicação, a largura de banda da comunicação, a capacidade de reconfiguração dos robôs, a capacidade de processamento e a composição dos agentes.

A quantidade de agentes autônomos no sistema pode ser definida por *size-alone* quando há apenas um robô, *size-pair* quando há dois robôs, *size-lim* quando a quantidade de robôs é pequena se comparada ao tamanho da atividade ou do ambiente, e *size-inf* quando o número de robôs é muito maior do que 1, isto é, existe efetivamente um infinito número de robôs. É estranho imaginar as definições *size-alone* e *size-pair* como um enxame de robôs, porém para Dudek *et al.* (1996) essa definição é válida visto que um único robô é o menor coletivo que pode ser formado e que dois robôs podem executar uma ação que um único robô não seria capaz de realizar. Já para Şahin (2005), a aplicação de controle para um pequeno número de robôs e que não busca a escalabilidade não é considerada como robótica de enxame.

Quanto à composição dos agentes, Dudek *et al.* (1996) afirma que os robôs podem ser idênticos (*cmp-ident*) quando os robôs são homogêneos em forma e função, isto é, *hardware* e *software.* Os robôs do tipo *cmp-hom* são considerados homogêneos e possuem as mesmas características físicas. Já os robôs do tipo *cmp-het* são considerados heterogêneos, pois, não são fisicamente uniformes. Essa definição também se aplica para diferença de comportamento dos robôs.

2.1.2. Robôs bio-inspirados

O comportamento das formigas, peixes, pássaros, abelhas, entre outros, são a principal inspiração para os enxames de robôs. Isto ocorre, pois, estes animais são

exemplos de como indivíduos simples executam tarefas complexas e atingem o objetivo quando se reúnem em grupos.

Os insetos voadores são capazes de realizar movimentos aerodinâmicos sofisticados apenas com asas e minúsculos sistemas nervosos. Tais movimentos vão desde evitar que uma mão o acerte a pousar em flores balançando ao vento (MA *et al.*, 2013).

Segundo Tangorra *et al.* (2011), os peixes ósseos nadam com um nível de agilidade impossível de ser atingido em sistemas desenvolvidos pelo homem. Tal fato ocorre devido à capacidade dos peixes de controlarem as forças hidrodinâmicas através da modulação ativa da cinemática e do desempenho mecânico das nadadeiras.

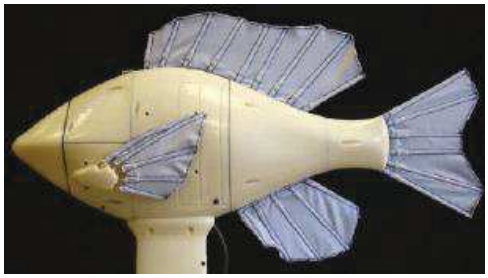
O morcego possui mais de 40 juntas ativas e passivas em suas asas gerando uma grande flexibilidade e uma cinemática complexa (RAMEZANI; CHUNG; HUTCHINSON, 2017). Por esses motivos trazem uma perspectiva única para as áreas de robótica e biologia.

A Fig. 2.2 apresenta cinco tipos de robôs bio-inspirados. O primeiro robô (Fig. 2.2a) é inspirado em seres com seis patas, como por exemplo, uma barata. Os robôs inspirados em insetos podem ser do tipo voadores tais como abelhas (Fig. 2.2b), mariposas, besouros e entre outros ou do tipo rastejante tal qual uma aranha (Fig. 2.2c). Robôs subaquáticos são na maioria das vezes inspirados em peixes (Fig. 2.2d). Por fim, o robô da Fig. 2.2e é inspirado em morcegos.

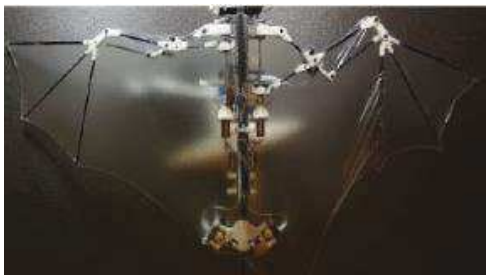
(a)

(b)

(c)



(d)



(e)

Figura 2.2 – Robôs bio-inspirados. Adaptado de Gravish e Lauder (2018).

O comportamento de grupos de animais sociais é robusto, escalável e flexível (BRAMBILLA *et al*., 2013). **Robustez** é a capacidade de lidar com a perda de indivíduos, que em animais socais, é representada pela redundância e pela ausência de um líder. **Escalabilidade** é a capacidade de execução com diferentes tamanhos de grupos, isto é, a introdução ou remoção de indivíduos não resulta em uma mudança drástica no desempenho de um enxame. Em animais sociais, a escalabilidade é representada pelo sensoriamento e comunicação locais. Por fim, a **flexibilidade** é a capacidade de lidar com um amplo espectro de diferentes ambientes e tarefas, que em animais sociais, é representada pela redundância, simplicidade de comportamento e mecanismos tais como a alocação de tarefas.

2.1.3. Feromônio de formigas

Brutschy *et al.* (2012) indagam em seu trabalho se “Formigas podem inspirar robôs?”. E essa pergunta surge visto que as interações entre os robôs de um enxame são baseadas em regras comportamentais simples que exploram apenas informações locais. Tais robôs não possuem nem conhecimento global, nem um controlador central. Sendo assim, as decisões em um enxame devem ser definidas através de interações locais.

O feromônio é uma substância química ou um conjunto de substâncias químicas produzidas por organismos vivos que transmitem uma mensagem para outros membros da mesma espécie (KARLSON e BUTENANDT, 1959). Para Fujisawa *et al.* (2008) o método de comunicação através do feromônio é químico, local e indireto. Sendo assim o feromônio de trilha pode ser utilizado para a aplicação em enxame de robôs, uma vez que essa substância é volátil e evapora com o tempo.

Existem diferentes maneiras de representar o feromônio, cada um com suas peculiaridades. Russel (1997) desenvolveu uma trilha de aquecimento com base em uma lâmpada de 70W e um sensor piroelétrico. Um problema é que a geração elétrica de calor não é possível, mesmo em robôs móveis maiores, devido a restrições na energia da bateria. Fujisawa *et al*. (2008) aplicou o álcool e o sensor de álcool para imitar o feromônio da formiga. Embora esta seja uma imitação muito realista das trilhas baseadas em feromônio de formigas, os sensores químicos utilizados nessa configuração, a combinação de robótica e substâncias demonstraram ser pouco confiáveis e não muito práticas. Mayet *et al*. (2010) apresentou uma abordagem que emite luz ultravioleta em tinta fosforescente, porém não há garantias de que a colisão entre robôs será evitada. Kim e Kurabayashi (2011) apresentaram um feromônio digital através da interação entre agentes e *tags* RFID. O

cálculo do feromônio é centralizado dentro do robô móvel. O trabalho é voltado para o problema de filtragem de um campo potencial para suavizar a superfície de modo que as informações do gradiente potencial possam ser amostradas e utilizadas pelos robôs móveis. Portanto, assim, o feromônio de formigas em robôs móveis de enxame requer evitamento de colisões, restrições de energia da bateria, implementação prática e independência entre o fenômeno do ambiente e os robôs móveis. Tang *et al.* (2017) também utilizam RFID para coordenação de um enxame de robôs em um ambiente, porém apenas em ambiente simulado. Lima e Oliveira (2017) desenvolveram um autômato celular para o controle de enxame de robôs em tarefas de forrageamento. Garnier *et al.* (2007) utilizaram LEDs como emulação do feromônio por permitirem um rastreamento fácil e confiável em condições de alteração do brilho de fundo. Ranjbar-Sahraei *et al.* (2013) utilizaram o robô comercial *e-puck* para testes de feromônio utilizando tinta fosforescente.

A Tab. 2.1 apresenta um comparativo das abordagens de feromônio utilizadas nos principais trabalhos relacionas à tese. Na tabela é possível identificar o tipo de feromônio, e se o trabalho possui resultados de simulação e/ou resultados experimentais. Foram utilizados três robôs móveis comerciais (Alice, ARGOS-01 e e-puck) e um robô móvel desenvolvido pelo próprio autor. Nos casos de simulação, em um dos trabalhos não é possível identificar se foi utilizado um simulador comercial ou se o simulador foi desenvolvido pelo próprio autor.

Tabela 2.1 – Comparativo das abordagens de feromônio utilizadas em trabalhos relacionados à tese.

Autores	**Tipo de feromônio**	**Simulação**	**Experimental**
Russel (1997)	Calor	-	Não comercial
Garnier *et al.* (2007)	LED	Webots	Alice
Fujisawa et al. (2008)	Álcool	-	ARGOS-01
Mayet et al. (2010)	Tinta fosforescente	NetLogo	e-puck
Kim e Kurabayashi (2011)	RFID	-	e-puck
Ranjbar-Sahraei *et al.* (2013)	Tinta fosforescente	-	e-puck
Lima e Oliveira (2017)	Virtual	Implementado em C	-
Tang *et al.* (2017)	RFID	Não mencionado	-

2.1.4. Simulação em robótica móvel (ambiente virtual)

Os simuladores mais utilizados no campo da robótica são: *ARGoS* (PINCIROLI *et al.*, 2011), *Webots* (MICHEL, 2004), *Gazebo* (KOENIG; HOWARD, 2004), *RFCSIM* (FABREGAS *et al.*, 2014) e *V-REP* (COPPELIA ROBOTICS GMBH, 2015). Tais simuladores são importantes, pois, testam teorias, ideias e arquiteturas antes da implementação em ambientes reais. Para Peralta *et al.* (2016), robôs de alta performance ainda são caros e, consequentemente, escassos em ambientes de pesquisa. Sendo assim, é evidente a importância do uso de simuladores para esse tipo de pesquisa.

Existem vários simuladores comerciais e *open source* no mercado, sendo que alguns deles possuem licença gratuita para propósitos educacionais. A maioria dos simuladores incluem os robôs comerciais e os mais conhecidos (PERALTA *et al.*, 2016).

O *V-REP* (*Virtual Robot Experimentation Platform*) é o resultado de um esforço em conciliar todos os requisitos em uma estrutura de simulação versátil e escalável (ROHMER; SINGH; FREESE, 2013). A plataforma possui diversos exemplos e modelos de robôs, além de sensores e atuadores estarem disponíveis para a criação de ambientes virtuais. Também é possível inserir novos objetos com definição de propriedades dinâmicas, isto é, modelos existentes podem ser modificados e modelos de robôs personalizados podem ser inseridos, facilitando a criação de cenários de simulação customizados.

Não existe funcionalidade principal ou central no V-REP. Isto significa que o V-REP possui diversas funcionalidades relativamente independentes, que podem ser habilitadas ou não no modelo simulado. Por exemplo, em um cenário de simulação em que um robô industrial pega caixas e as move para outro local, o V-REP calcula o modelo dinâmico para agarrar e segurar as caixas e executa uma simulação cinemática para as outras partes do ciclo quando os efeitos dinâmicos são desprezíveis. Com essa abordagem é possível calcular o movimento do robô industrial de forma rápida e precisa, o que não seria possível caso tivesse sido simulado utilizando apenas as bibliotecas complexas de modelos dinâmicos. Essa simulação híbrida é justificada nesse tipo de situação se o robô é rígido e fixo e não é influenciado pelo ambiente (ROHMER; SINGH; FREESE, 2013).

2.2. Identificação por Rádio Frequência

RFID é a tecnologia que utiliza comunicação via ondas de rádio para troca de dados entre um leitor e uma etiqueta eletrônica, tradicionalmente fixada em um objeto de interesse, principalmente para identificação e rastreamento. (COSKUNM, OK; OZDENIZCI, 2011)

Segundo Fennani, Daham e Dahmane (2011), um sistema RFID pode ser dividido em três componentes principais (Fig. 2.3):

A etiqueta (*tag*): um conjunto que pode possuir antena, memória e *chip* colados ou injetados em objetos. Cada etiqueta possui seu próprio código identificador que será associado na identificação do objeto.

O leitor (*reader*): é o equipamento que coleta dados da etiqueta, filtram e transferem para uma unidade de processamento.

O servidor (*server/host*): é a parte de processamento do sistema RFID.

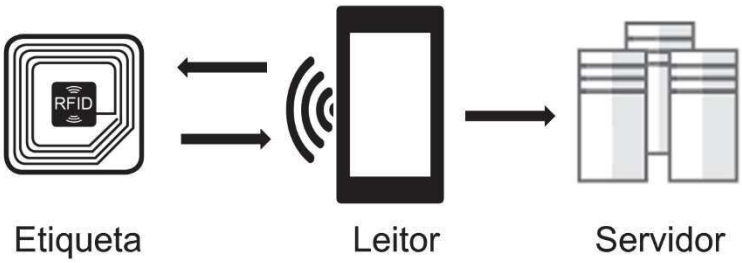


Figura 2.3 – Componentes de um sistema RFID.

As etiquetas eletrônicas são classificadas de acordo com suas características construtivas e as respectivas funcionalidades (AZAMBUJA, 2011). Uma das categorias diz respeito à existência ou não de circuitos integrados nas etiquetas ("*chip tags*" vs "*chipless tags*"). As etiquetas comumente utilizadas em sistemas de gestão de livros para bibliotecas são do tipo sem *microchip* enquanto as etiquetas utilizadas no cadastro e rastreamento de animais bovinos possuem um *microchip*.

Outra categoria caracteriza a forma como se dá a comunicação entre as etiquetas e os equipamentos leitores e divide as etiquetas em: passivas, semi-passivas e ativas. O primeiro tipo de etiquetas, classes 1 e 2, não possui fonte de energia nem dispositivo de transmissão ativa, ou seja, só respondem aos sinais enviados pelas antenas de leitores. As etiquetas semi-passivas (classe 3) possuem fonte de energia, porém não possuem transmissão ativa. Já as etiquetas ativas (classe 4) possuem fonte de energia e transmissão ativa, ou seja, podem tomar a iniciativa de iniciar a transmissão de dados.

Por fim, a última categoria refere-se ao tipo de acesso à memória, uma vez que as etiquetas podem ser de somente-leitura ou leitura-e-escrita.

A alocação de frequência do sistema RFID se dá em três faixas: LF (*low frequency*), HF (*high frequency*) e UHF (*ultra high frequency*). As frequências de 125 kHz e 134,2 kHz fazem parte da faixa LF e não são afetadas por metais ou água, o que torna a aplicação bastante ampla. A frequência de 13,56 MHz está na faixa HF e é utilizada no mundo todo,

com isso o custo deste tipo de sistema é menor do que para LF. Normalmente essa faixa de frequência é utilizada em cartões de crédito, passaportes e cartões de controle de acesso. A faixa de UHF utiliza as frequências de 433 MHz, 2,45 GHz e a banda 860-956 MHz. A faixa UHF é muito utilizada em cadeia de varejista e na indústria, devido ao seu baixo custo de produção e a possibilidade de identificação de várias etiquetas simultaneamente. (FENNANI; HAMAM; DAHMANE, 2011)

Outra forma de comunicação utilizando radio frequência é a comunicação por campo próximo, ou *Near Field Communication* (NFC). À primeira vista, NFC não se trata de um sistema RFID, mas uma interface *wireless* para comunicação entre dispositivos, similar ao infravermelho e ao *bluetooth*. Entretanto, apresenta diversas características comuns aos sistemas RFID. A transmissão de dados entre duas interfaces NFC utiliza a mesma faixa de comunicação de um sistema RFID HF, ou seja, 13,56 MHz e a distância máxima de comunicação é de 20 cm (FINKENZELLER, 2010).

Para os desenvolvedores que utilizam plataformas de prototipagem rápida como Arduino existe um módulo NFC RFID, o PN532 (ELECHOUSE, 2013). A Fig. 2.4 apresenta as possibilidades de comunicação utilizando este módulo leitor: comunicação entre leitores utilizando a tecnologia NFC e comunicação entre leitor e etiqueta utilizando a tecnologia RFID.

Figura 2.4 – Comunicação RFID leitor – etiqueta e leitor NFC – leitor NFC.

2.3. Redes de Petri

As redes de Petri (*Petri nets* - PN) são uma ferramenta de modelagem gráfica e matemática aplicada a vários sistemas (MURATA, 1989). São utilizadas para descrever e estudar sistemas de processamento da informação que podem ser caracterizados como concorrentes, assíncronos, distribuídos, paralelos, não determinísticos, e/ ou estocásticos.

Do ponto de vista de ferramenta gráfica, as PN podem ser utilizadas como uma comunicação visual similar a fluxogramas, diagramas de blocos e redes. Além disso, as fichas (ou *tokens*) simulam as atividades dinâmicas e concorrentes dos sistemas. Do ponto de vista de ferramenta matemática, as PN são capazes de definir equações de estado,

equações algébricas e outros modelos matemáticos que regem o comportamento do sistema.

O desenvolvimento de sistemas concorrentes é particularmente desafiador (JENSEN; KIRSTENSEN, 2009). O principal motivo é o fato de que tais sistemas possuem concorrência e não determinismo, isto é, a execução desses sistemas pode ocorrer de maneiras diferentes. Uma forma de abordar a dificuldade de desenvolvimento de sistemas concorrentes é através da construção de um modelo.

Segundo Desel e Reisig (1996) as redes são constituídas por um conjunto de lugares, um conjunto de transições e um conjunto de arcos direcionados. Em representação gráfica, os lugares são desenhados com círculos, transições com quadrados (retângulos) e arcos com setas.

A rede de Petri da Fig. 2.5 é composta por três lugares e quatro transições e modela o controle de uma máquina de vendas. Na marcação inicial (p_1) a máquina espera pela inserção de uma moeda (representado pelo *token*). Quando inserida, a moeda pode ser rejeitada ou aceita. Se a moeda for rejeitada, o sistema volta ao estado inicial. Caso contrário, o sistema libera um item antes de voltar ao estado inicial. A Tab. 2.2 apresenta a descrição de cada lugar e transição da rede de Petri.

Tabela 2.2 – Descrição dos lugares e transições para o controle de uma máquina de vendas.

Lugar/Transição		**Descrição**
lugares	P_1	Pronto para inserção de moeda
	P_2	Com moeda
	P_3	Pronto para entregar
transições	T_1	Inserir moeda
	T_2	Rejeitar moeda
	T_3	Aceitar moeda
	T_4	Entregar item

A rede de Petri lugar/transição é definida, formalmente, como uma quíntupla $R = (P, T, I, O, M)$, em que:

$P = \{P_1, P_2, \dots P_m\}$ é um conjunto finito com m lugares, $m > 0$,

$T = \{T_1, T_2, \dots T_n\}$ é um conjunto finito com n transições, $n > 0$,

O conjunto $P \cap T = \emptyset$,

$I: T \rightarrow P$ é a função de entrada e identifica o conjunto de lugares de entrada para uma transição,

$O: T \rightarrow P$ é a função de saída e identifica o conjunto de lugares de saída para uma transição,

$M: P \rightarrow N$, é a marcação ou estado da rede, onde N é um conjunto de inteiros incluindo o zero.

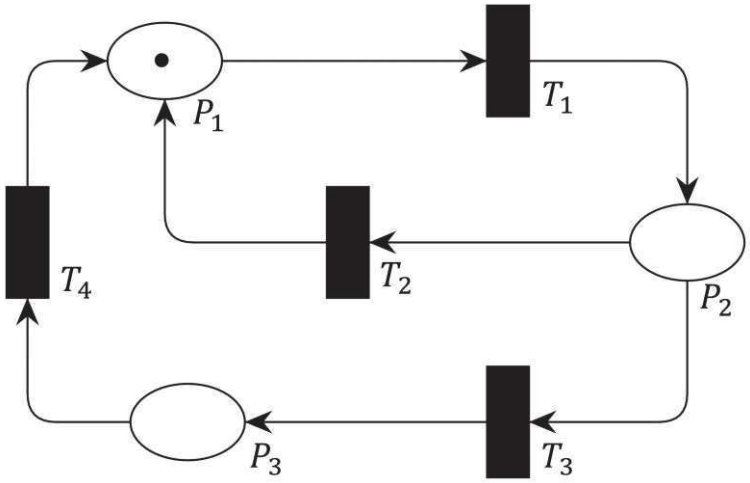


Figura 2.5 – Exemplo de uma rede de Petri lugar/transição. Modificado de Desel e Reisig (1996).

Através de notação matricial, as propriedades de uma rede de Petri podem ser algebricamente analisadas. A marcação armazena o estado atual da rede de Petri, enquanto o próximo estado é dado pela Eq. 2.1.

$$M_{k+1} = M_k + A^t . u_k \tag{2.1}$$

em que:

M_k é um vetor coluna (m x 1), denominado vetor de estados ou vetor de marcação, cujo elemento da m-ésima linha corresponde à quantidade de marcações no m-ésimo lugar, em determinado instante de tempo k;

M_{k+1} é o vetor de marcação resultante do disparo k;

A^t é a função transposta da matriz de incidência, que representa o conjunto de arcos da rede de Petri. A matriz de incidência possui m linhas e n colunas. O elemento a_{ij} terá valor −1 quando o lugar correspondente à linha i for uma entrada da transição correspondente à coluna j, e terá valor 1 quando for uma saída. Caso não exista arco ligando o lugar e a transição correspondentes, o elemento terá valor 0.

u_k é um vetor coluna (*n* x 1), ou vetor de disparo, cujo elemento da enésima coluna corresponde à quantidade de vezes que a transição n foi realizada no disparo.

Para a rede de Petri da máquina de vendas da Fig. 2.5 temos a matriz de incidência transposta (Eq. 2.2) e o vetor de marcações (Eq. 2.3) inicial. Na matriz da Eq. 2.2, as colunas representam os lugares e as linhas as transições. A posição A_{23} indica que a transição T_3 ao ser disparada, consome o *token* do lugar P_2 e coloca no lugar P_3 (posição A_{33}).

$$A^t = \begin{bmatrix} -1 & 1 & 0 & 1 \\ 1 & -1 & -1 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{2.2}$$

$$M_k = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \tag{2.3}$$

2.3.1. Pseudoboxes

Conforme apresentado por Silva e del Foyo (2012), *pseudoboxes* são uma condição observável que não é controlada pelo sistema modelado e pode representar informações de controle externas aos componentes hierárquicos. O exemplo de uma rede de Petri com *pseudoboxes* é apresentado na Fig. 2.6. Os lugares p_1 e p_2 indicam a representação formal de uma sub-rede (possivelmente o comportamento de um objeto) e o mundo exterior (DEL FOYO; SILVA, 2004). Nessa figura as *pseudoboxes* são representadas por lugares de coloração cinza. e são constituídos por lugares de coloração cinza.

As pseudoboxes não são representadas na Eq. 2.1 por se tratar de comunicação entre sistemas.

Ainda em relação a Fig 2.6, o arco entre o lugar p_2 e a transição a_5 é chamado de arco inibidor e indica que a transição a_5 só será habilitada se o lugar p_2 possuir um número de fichas menor do que o peso do arco (que é 1 para a rede de Petri apresentada). Ou seja, a transição só é habilitada quando não há fichas no lugar p_2. Os arcos inibidores são usados para modelar prioridades de disparo em caso de conflito. Já o arco entre o lugar p_1 e a transição a_2 é chamado de arco habilitador e indica que após o disparo da transição a marcação mantém inalterada no lugar p_1.

2.3.2. Rede de Petri de alto nível

As redes de Petri temporizadas são úteis para avaliar desempenho (sistemas computacionais, sistemas de manufatura, entre outros). Cronometrar o tempo pode estar associado com a duração de uma operação ou com um tempo esperado antes de algum evento ocorrer como em uma falha, por exemplo (DAVID; ALLA, 2010). O tempo nas redes de Petri temporizadas podem ser associadas aos lugares (*P-Timed Petri Nets*) ou às transições (*T-Timed Petri Nets*).

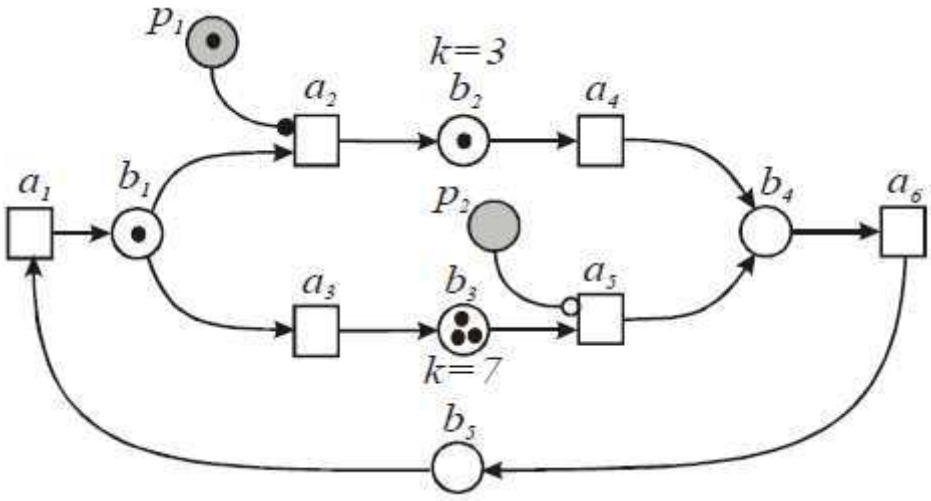


Figura 2.6 – Exemplo de uma rede de Petri com *pseudoboxes* e elementos hierárquicos. Retirado de del Foyo e Silva (2004).

Sistemas do mundo real frequentemente contém muitas partes similares, mas não idênticas. Ao utilizar redes de Petri lugar/transição, essas partes são representadas por sub-redes com estruturas quase idênticas. O desenvolvimento de redes de Petri de alto nível constitui uma melhoria significativa deste problema. As redes de Petri coloridas pertencem a classe de redes de Petri de alto nível. (JENSEN, 1997)

Para Jensen (1997), as redes de Petri coloridas foram desenvolvidas para ser um modelo teórico promissor com uma linguagem completa para o projeto, especificação, simulação, validação e implementação de grandes sistemas de *software* (e outros sistemas em que seres humanos e / ou computadores comunicam através de regras mais ou menos formais).

O CPN Tools é uma ferramenta para edição, simulação, verificação e análise de desempenho de modelos em redes de Petri colorida (JENSEN; KRISTENSEN, 2009).

2.3.3. Semáforo

Alguns tipos de redes de Petri possuem aplicação em problemas de tráfego de automóveis em ambientes urbanos. As redes de Petri coloridas são capazes de fornecer uma representação suficientemente precisa e válida de sistemas de tráfego utilizando dados de sensores posicionados na entrada do tráfego, como apresentado por Dotoli e Fanti (2006). Já as redes de Petri temporizadas podem auxiliar na prevenção do congestionamento do tráfego urbano quando ocorrem acidentes (QI, ZHOU; LUAN, 2016), ou evitar o transtorno ocorrido quando há passagem de veículos de emergência por semáforos (HUANG, WENG; ZHOU, 2015).

A Fig. 2.7 apresenta a modelagem apresentada por Huang, Weng e Zhou (2015). Sempre que um veículo de emergência aproxima do semáforo, a transição t_7 é disparada e um *token* é gerado no lugar p_9. Assim, após o disparo de t_5, t_1 é disparado e então o semáforo do lado direito fica verde, enquanto o semáforo do lado esquerdo fica vermelho.

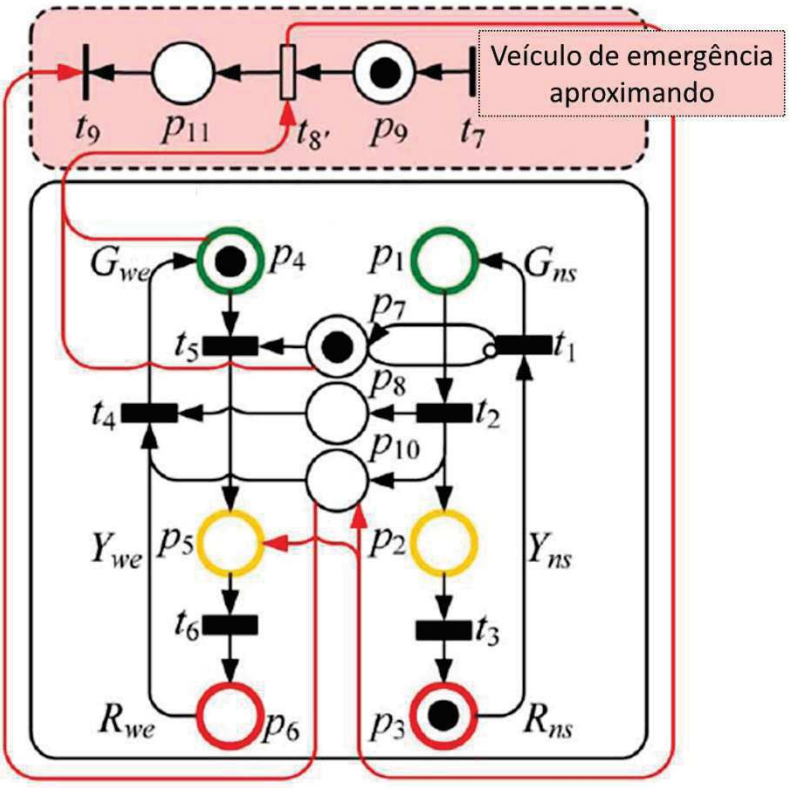


Figura 2.7 – Exemplo de uma rede de Petri modelando um semáforo com veículos de emergência. Adaptado de Huang, Weng e Zhou (2015)

2.4. PNRD e iPNRD

A rede de Petri inserida em base de dados RFID (PNRD) definida por Tavares e Saraiva (2010) é uma estrutura de dados formal baseada em duas abordagens principais: a

utilização das etiquetas ou *tags* RFID como um banco de dados do processo ao qual o item tagueado está inserido, bem como o armazenamento do estado vigente do item no processo; e a atribuição da condição de disparo da rede de Petri vinculado à leitura da tag pelo leitor RFID adicionado ou não a outras pré-condições. Esse disparo automaticamente atualiza o estado do item identificado, alterando a marcação do seu estado em acordo com a Equação 4.1.

Já a PNRD invertida (iPNRD), proposta apresentada por Fonseca e Tavares (2017), consiste em gravar o vetor de disparos da rede de Petri nas *tags*, enquanto a matriz de incidência e os vetores de estado são salvos no robô móvel. Assim, quando o robô que tenha um leitor RFID acoplado passa por uma *tag*, uma transição é disparada e ele atualiza seu estado (marcação) na rede de Petri modificando assim seu comportamento.

A Fig. 2.8 apresenta os termos da equação de atualização vetor de marcações (Eq. 2.1) e como eles se associam aos componentes do sistema RFID, tanto na PNRD quanto na iPNRD.

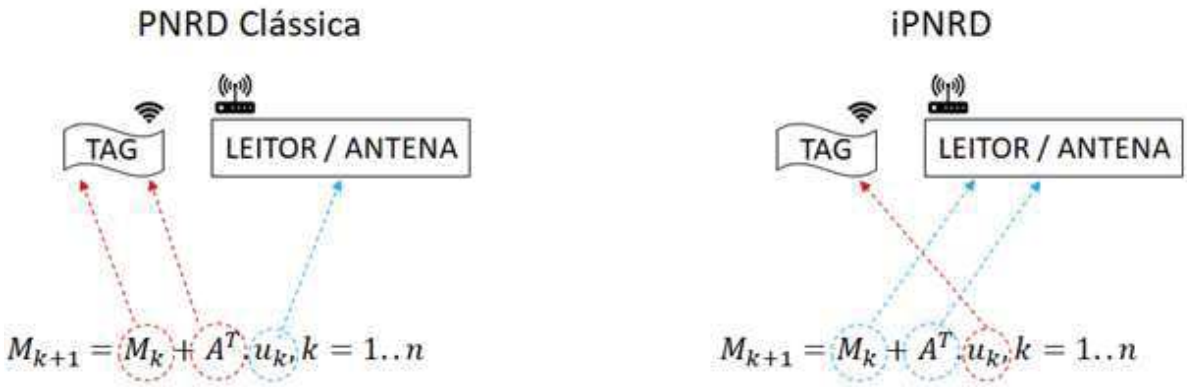


Figura 2.8 - Associação dos termos da equação de atualização do vetor de marcações com os elementos RFID para as abordagens PNRD clássica e iPNRD. Adaptado de Fonseca (2018)

Fonseca (2018) desenvolveu em seu trabalho uma aplicação da iPNRD para o problema de busca e salvamento. A travessia Petrópolis – Teresópolis que acontece no Parque Nacional da Serra dos Orgãos, no Rio de Janeiro, foi simplificada como apresenta a Fig. 2.9. Neste caso o caminhante (agente ativo) possui um leitor RFID e os pontos estratégicos (portarias de Petrópolis e Teresópolis, Castelos do Açu e Pedra do Sino) possuem *tags* RFID (transições da PN). A PN associada à localização do agente possui cinco lugares e oito transições. A Tab. 2.3 possui a descrição dos lugares e transições para a travessia. Como o agente encontra-se entre os Castelos do Açu e a Pedra do Sino a marcação da rede de Petri está em p_3.Caso o agente não se perca durante todo o percurso,

ele passará por quatro etiquetas RFID representadas por: t_1, t_2, t_3 e t_4 no caso da travessia Perópolis-Teresópolis ou t_5, t_6, t_7 e t_8 quando está no sentido contrário.

As *tags* armazenam o vetor de disparos u_k referente à PN dos usuários. Os usuários armazenam a matriz de incidência A^t da PN referente ao seu próprio comportamento e a marcação correspondente M_k, ou seja, em qual região do trajeto o usuário se encontra. O agente de busca terrestre verifica (por meio do leitor) em qual parte do trajeto o usuário perdido se encontra.

Tabela 2.3 – Descrição dos lugares e transições para a travessia Petrópolis-Teresópolis.

Lugar/Transição		**Descrição**
lugares	p_1	Em Petrópolis
	p_2	Entre Petrópolis e Castelos do Açu
	p_3	Entre Castelos do Açu e Pedra do Sino
	p_4	Entre Pedra do Sino e Teresópolis
	p_5	Em Teresópolis
transições	t_1	Sair de Petrópolis rumo a Teresópolis
	t_2	Sair do Castelos do Açu rumo a Teresópolis
	t_3	Sair da Pedra do Sino rumo a Teresópolis
	t_4	Chegar em Teresópolis
	t_5	Sair de Teresópolis rumo a Petrópolis
	t_6	Sair da Pedra do Sino rumo a Petrópolis
	t_7	Sair do Castelos do Açu rumo a Petrópolis
	t_8	Chegar em Petrópolis

O problema específico de busca e salvamento também pode ser beneficiado pela iPNRD através do armazenamento do histórico de usuários que passaram por um local associado a uma *tag* (FONSECA, 2018). Esta informação é bastante relevante em casos de desaparecimento, uma vez que é possível direcionar as buscas para a região mais provável de se encontrar a vítima. Como isso o tempo de procura é aumentado e, consequentemente, as chances de salvamento.

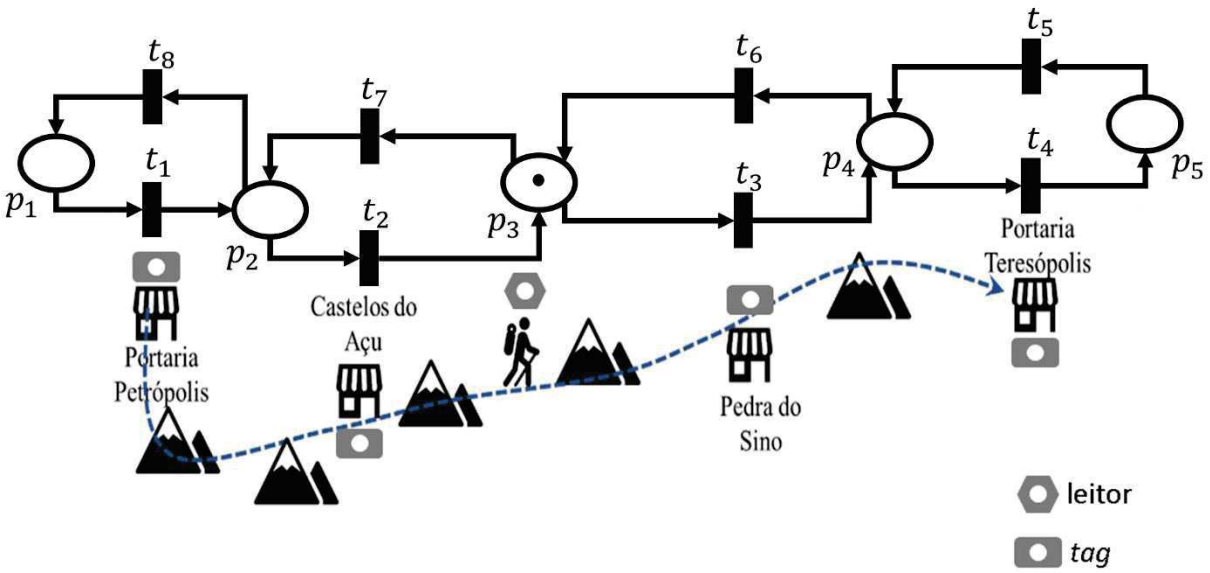


Figura 2.9 – Exemplo de aplicação da abordagem iPNRD. Adaptado de (FONSECA, 2018)

Silva, Tavares e Ferreira (2018) implementaram uma biblioteca na plataforma de prototipagem eletrônica Arduino capaz de atuar nos diferentes tipos de aplicações da PNRD. A biblioteca permite configurar quais informações serão salvas na etiqueta, tornando possível a implementação tanto da PNRD quanto da iPNRD. Como é possível separar cada conjunto de dados que é armazenado na etiqueta, é possível criar outras variações da PNRD e da iPNRD como apresentado na Tab. 2.4. A biblioteca foi implementada para o leitor NFC RFID PN532. O projeto tem abordagem iPNRD com a variante original.

Tabela 2.4 – Arranjo de dados entre leitor e etiqueta RFID. Retirado de Silva, Tavares e Ferreira (2018).

Abordagem	Variante	Leitor	Etiqueta
PNRD	Original	u_k	M_k e A^T
	Variante 1	A^T	M_k e u_k
	Variante 2	M_k	A^T e u_k
iPNRD	Original	M_k e A^T	u_k
	Variante 1	M_k e u_k	A^T
	Variante 2	A^T e u_k	M_k

CAPÍTULO III

METODOLOGIA E DESENVOLVIMENTO

Este trabalho aborda a utilização da iPNRD em enxame de robôs para controle autônomo e distribuído através da emulação do feromônio de formiga. O intuito é gerar uma interação robô/ambiente de forma que o comportamento dos robôs seja orientado com base na troca de dados entre leitores e *tags* RFID. Este capítulo descreve a metodologia e o desenvolvimento do projeto. A priori, a metodologia utilizada é descrita na seção 3.1, seguida da análise do problema referente à tomada de decisão para robôs móveis (seção 3.2). A proposta da solução é detalhada na seção 3.3, com o modelo do feromônio utilizado. A seção 3.4 apresenta os cenários avaliados na tese (experimental e simulação). Por fim, a seção 3.5 contém a modelagem em Rede de Petri dos agentes ativos e passivos do sistema.

3.1. Metodologia

A metodologia considerada para o desenvolvimento do trabalho foi o da pesquisa exploratória, com enfoque no aprofundamento do comportamento do sistema, levantamento das hipóteses, e definição dos métodos e procedimentos.

O método de desenvolvimento adotado segue os seguintes pontos:

- Análise do problema de coleta de alimento por formigas (capítulo 3);
- Especificação da iPNRD associada ao ambiente inteligente e ao robô móvel para solução do problema (capítulo 3);
- Modelagem das Redes de Petri em CPN *Tools* (capítulo 3)*;*
- Definição das arquiteturas de controle para o método proposto (capítulos 3 e 4);
- Projeto e construção do robô móvel (apêndice A);
- Preparação dos ambientes de desenvolvimento:
    - Programação dos dispositivos embarcados (capítulo 4);
    - Programação do ambiente de simulação do V-REP (capítulo 4);

- Testes funcionais (capítulo 4);
- Análises e conclusões (capítulo 5).

O enfoque do trabalho está no comportamento estratégico dos robôs e do ambiente inteligente, portanto os modelos cinemáticos e dinâmicos não são abordados. O sensoriamento e atuação em níveis de controle reativo foram aplicados apenas para evitar a colisão entre os robôs. As funcionalidades como navegação, visão, mapeamento, localização e controle de trajetória não são tratadas.

3.2. Análise do problema

O problema proposto consta com a combinação dos dois trabalhos apresentados na Fig. 3.1. O primeiro trabalho apresenta a análise do comportamento de robôs quando há dois caminhos distintos para a busca de uma única fonte de alimento (Fig. 3.1a). Já o trabalho da Fig. 3.1b apresenta, em azul, uma bifurcação para duas regiões com alimento. Em ambos os casos há a necessidade de uma tomada de decisão, seja a escolha do caminho (primeiro caso) ou a escolha da fonte de alimento (segundo caso).

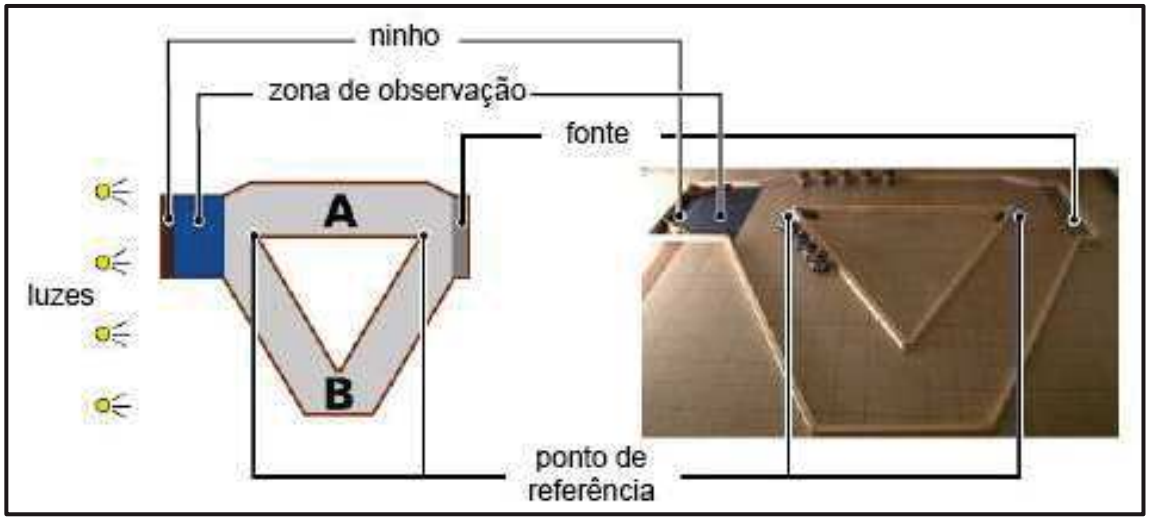
No caso da Fig. 3.1a há um grande desperdício de energia quando os robôs optam por utilizar o caminho B para buscar alimento da fonte, além de que o tempo para esgotar a fonte é muito maior do que quando os robôs optam pelo caminho A. Já no caso da Fig. 3.1b o que deve ser levado em consideração é a quantidade de alimento disponível na fonte mais próxima ao ninho, pois, desta maneira há um menor gasto energético para esgotar esta fonte.

Sendo assim, o problema abordado na tese pode ser expresso na forma de uma questão: Como criar controle autônomo e distribuído para enxame de robôs?

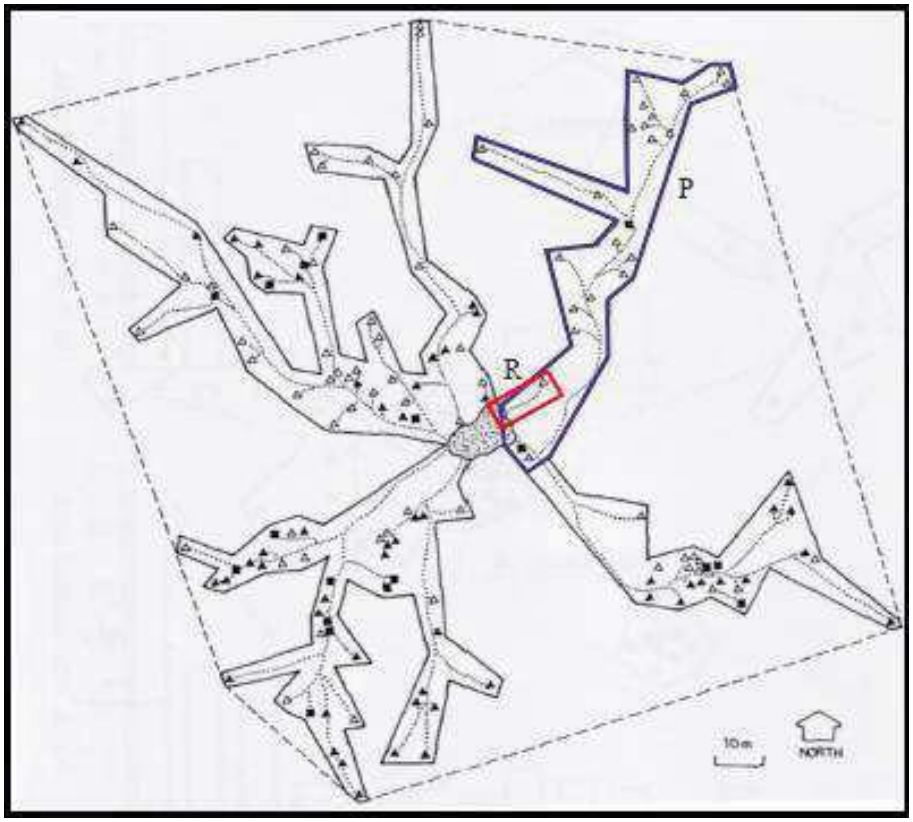
3.3. Solução proposta

A solução proposta pode ser simplificada na análise do comportamento dos robôs em um ambiente com um ninho e duas trilhas, como mostra a Fig. 3.2a. Os robôs movimentam-se em sentido anti-horário e ao sair do ninho devem decidir se seguem a trilha A ou a trilha B (caminho mais longo). Em cada caminho há uma certa quantidade de alimento disponível que influencia na tomada de decisão do caminho a ser escolhido. Os pontos de entrada e saída do ninho são distintos para que não haja colisão entre os robôs que chegam e saem. Com isso é possível determinar dois pontos de troca de informações (Fig. 3.2b) entre o

ambiente e o robô: *Tag Feromônio 1* e *Tag Feromônio 2.* No primeiro ponto o robô informa a trilha que percorreu, enquanto o segundo ponto indica ao robô a trilha que deverá seguir.

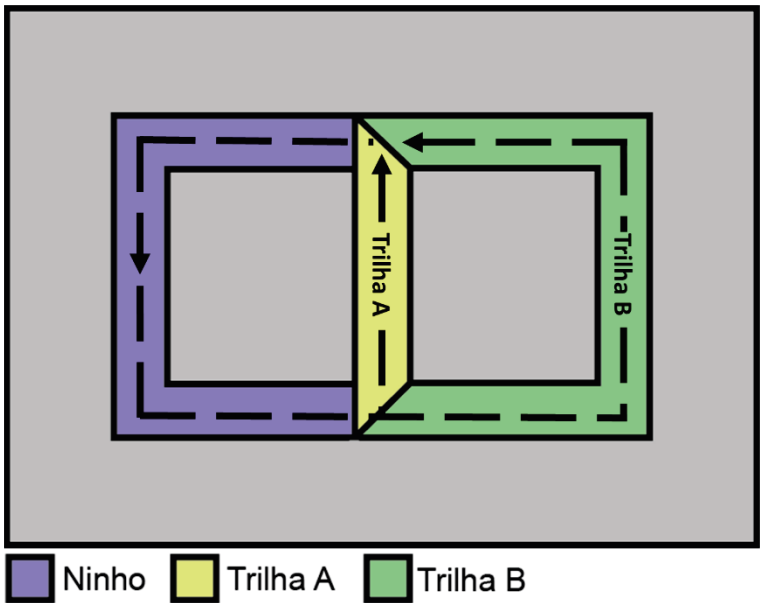


(a)

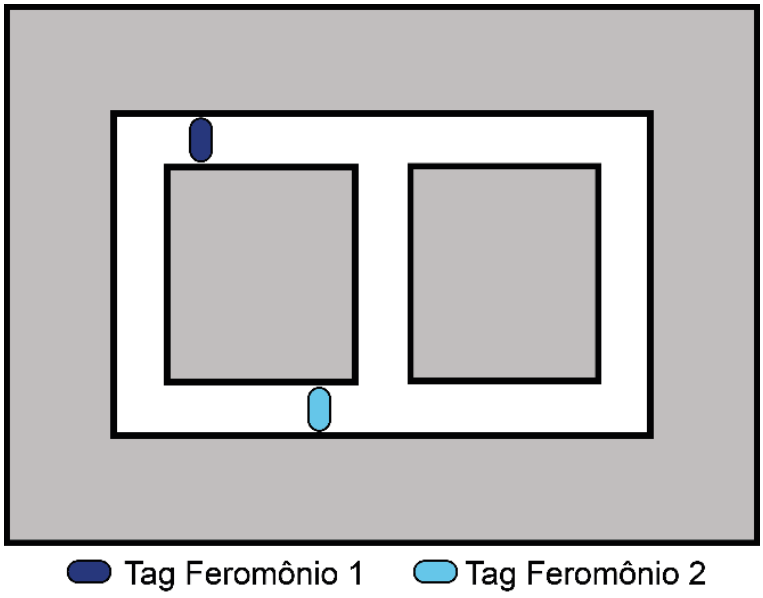


(b)

Figura 3.1 – (a) Sistema baseado em exame de formigas com dois caminhos distintos entre o ninho e a fonte, adaptado de Brutschy *et al.* (2012). (b) Regiões de forrageamento de formigas cortadeiras, retirado de Teixeira *et al.* (2014).



(a)



(b)

Figura 3.2 – Ambiente simulando um ninho e duas trilhas para busca de alimentos.

A equação que define a concentração de feromônio em cada trilha é apresentada na Eq. 3.1. A curva segue uma exponencial decrescente baseada na concentração de feromônio após um robô atualizar a *Tag Feromônio 1*, o tempo decorrido desde a última atualização da *Tag Feromônio 1* para aquela trilha, e a concentração de feromônio inicial daquela trilha. A curva possui esses parâmetros, pois, toda vez que um robô retorna por aquele caminho é necessário atualizar os parâmetros de decaimento.

Além da equação de decaimento do feromônio foi considerado que a quantidade de sinal “químico” referente a trilha possui um limite máximo, representando assim uma saturação do feromônio. A volatilidade do feromônio é representada pelos parâmetros da exponencial decrescente: tempo e concentração de feromônio inicial.

$$phe = cphe.10^{-\frac{t}{cphei^2}} \tag{3.1}$$

Sendo

phe: feromônio no instante calculado

$cphe$: concentração de feromônio atualizado por um robô

$cphei$: concentração de feromônio inicial

t: tempo decorrente desde a última atualização

A equação da concentração de feromônio é individual para cada trilha. Com isso, a indicação da trilha pela *Tag Feromônio 2* depende de uma comparação como mostra o pseudocódigo do Quad. 3.1. Para a trilha ser indicada, é necessário que a concentração de feromônio daquela trilha seja superior a 52 % da soma das concentrações de feromônio. Este valor foi escolhido para garantir que em regime transiente, os robôs possam transitar pelas duas trilhas, até que uma delas se torne a trilha indicada. Isto significa que inicialmente nenhuma das trilhas seja indicada (‘trilha = 0’ no pseudocódigo) e a escolha de caminho a seguir seja de forma randômica e por determinação do robô.

Quadro 3.1 – Pseudocódigo para atualização da trilha indicada pela Tag Feromônio 2.

```
Início
  se ferA/(ferA + ferB) > 0,52 então
    trilha = A
  senão se ferB/(ferA + ferB) > 0,52 então
    trilha = B
  senão
     trilha = 0
  fim-se
Fim
```

Como o ninho possui internamente apenas um caminho, os robôs se comportam como uma fila do tipo FIFO (*First in, first out*), isto é, o robô retorna, atualiza a *Tag Feromônio 1*, e segue até a *Tag Feromônio 2,* onde recebe a informação da trilha a seguir e então, sai do

ninho. Quando há robôs percorrendo as duas trilhas, há um problema de concorrência quando retornam ao ninho (Fig. 3.3). Esta concorrência existe, pois, há apenas um caminho de retorno ao ninho e o robô da *trilha A* choca com o robô da *trilha B*. Para resolver este problema é necessário que algum robô tenha prioridade de movimentação quando há robô na outra trilha. Este problema de concorrência é resolvido inserindo um semáforo para o retorno do ninho. A Fig. 3.4 apresenta a disposição dos semáforos no sistema.

O semáforo possui duas cores: vermelho (movimentação proibida) e verde (movimentação liberada). O semáforo só torna verde na trilha se houver robô para transitar. Caso contrário, ele se mantém vermelho. Quando há robô nas duas trilhas a prioridade está na trilha de caminho mais curto (*trilha A*). As informações trocadas entre o semáforo e o robô são através de etiquetas RFID, sendo *Tag A* a etiqueta da trilha A e *Tag B* a etiqueta da trilha B.

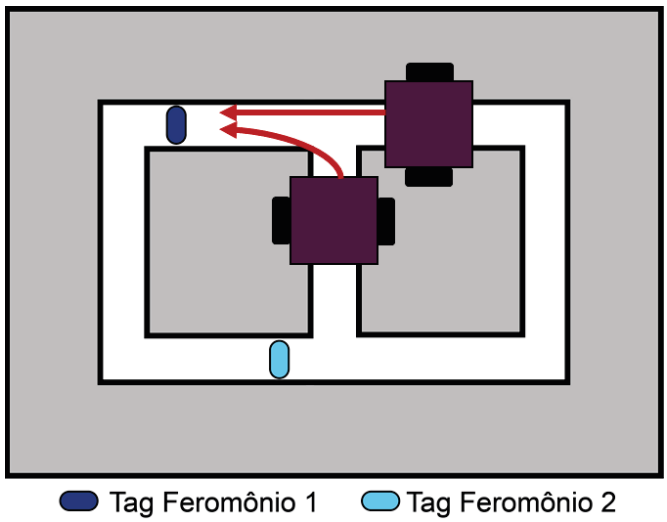


Figura 3.3 – Concorrência entre robôs que retornam ao ninho.

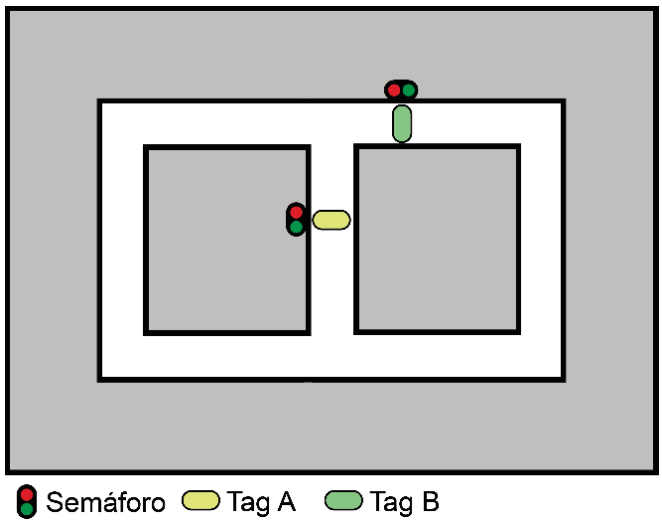


Figura 3.4 – Semáforo para resolver o problema de concorrência no retorno dos robôs.

A comunicação robô-ambiente pode ser feita através de duas tecnologias: NFC (leitor-leitor) ou através das etiquetas (leitor-etiqueta-leitor). No primeiro caso, para os leitores utilizados no desenvolvimento da tese há uma limitação na implementação da leitura, uma vez que os leitores precisam ficar em uma posição específica para que a comunicação ocorra sem erros. No segundo caso, a etiqueta funciona como um intermediário entre os agentes do sistema. Sendo assim, cada leitor lê e escreve nas etiquetas e as informações são transmitidas entre os agentes através dos dados salvos na etiqueta.

O módulo PN532 é um leitor RFID com tecnologia NFC capaz de realizar tanto a comunicação leitor-leitor quanto a comunicação leitor-etiqueta. É um módulo versátil, porém há conflito de leitura quando dois módulos (robô e ambiente) tentam acessar uma etiqueta ao mesmo tempo. Com isso é necessário que apenas um leitor acesse uma etiqueta por vez. Foi determinado que os robôs habilitem os leitores apenas no momento da leitura da etiqueta e os leitores do ambiente (feromônio e semáforo) são habilitados a cada cinco segundos. Esta estratégia minimiza significativamente a quantidade de conflitos de leitura, porém não elimina este problema.

3.4. Modelagem dos Agentes em Redes de Petri

Para compreender as relações existentes entre robô e ambiente é necessário entender quais agentes fazem parte do projeto. A Fig. 3.5 apresenta a comunicação do robô com os dois agentes do ambiente através das etiquetas. Cada agente do ambiente (feromônio e semáforo) comunicam com suas respectivas etiquetas. Na imagem é possível identificar que os agentes do ambiente são independentes e não há relação entre eles.

Os três agentes são modelados em redes de Petri uma vez que a abordagem iPNRD baseia-se neste tipo de modelagem. São definidas quatro redes, duas para o robô e uma para cada agente restante. A troca de informações entre as redes são realizadas através de *pseudoboxes*.

A Fig. 3.6 ilustra a comunicação entre o agente ativo e o ambiente (semáforo ou feromônio) indicando os elementos da equação de estados da rede de Petri. O vetor de disparos (u_k) é salvo na etiqueta enquanto o vetor de estados atual (M_k) e a matriz de incidência (A^t) estão salvos nos agentes. A *pseudobox* é a forma de transmissão de informações entre os agentes e é representada por um lugar com marcação nas redes de Petri dos agentes. As redes de Petri modeladas foram baseadas no trabalho de Ferreira, Tavares e Silva (2018).

O projeto do robô móvel de baixo custo também é um dos objetivos da tese, pois, o custo dos robôs móveis comerciais tornaria este projeto inviável. O *Attabot* (robô

desenvolvido) possui apenas os sensores e atuadores necessários para atender às necessidades de locomoção e interação com o ambiente proposto. O projeto mecânico e eletrônico do robô está detalhado no Apêndice A. O robô móvel possui uma autonomia de aproximadamente 60 minutos.

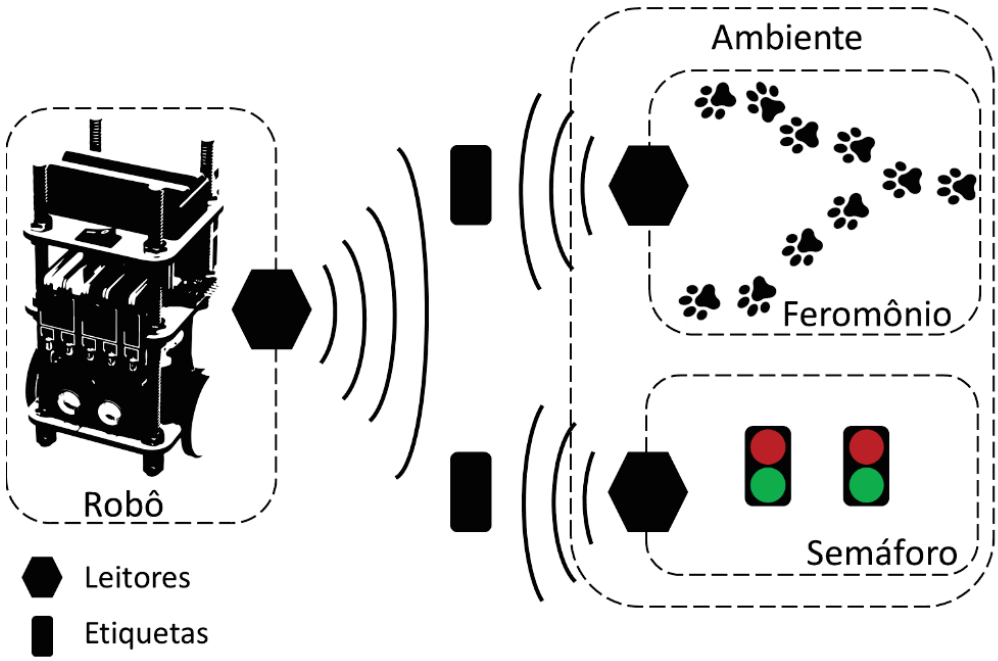


Figura 3.5 – Agentes do sistema.

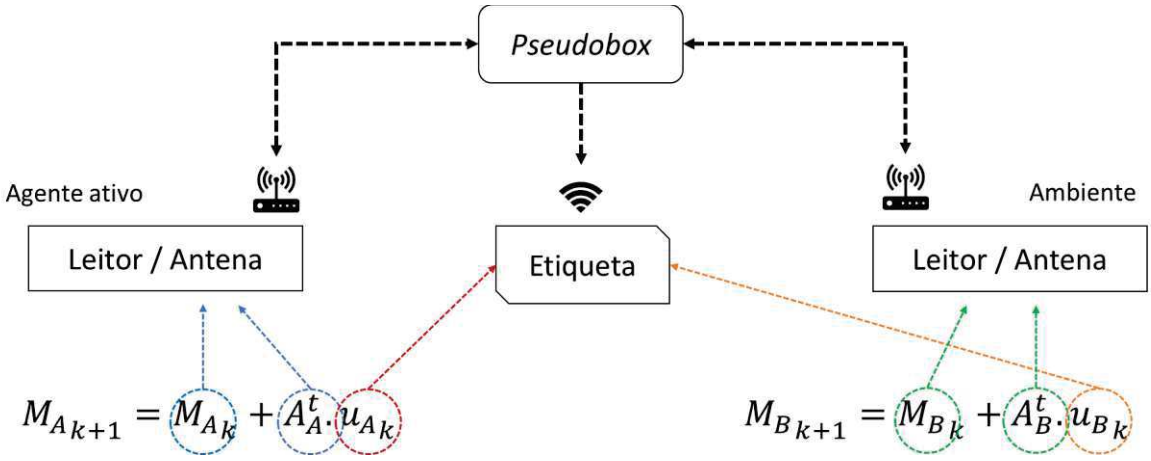


Figura 3.6 – Comunicação entre os agentes através das *pseudoboxes*.

3.5.1. Robô móvel e feromônio

O robô móvel possui dois comportamentos independentes no sistema (feromônio e semáforo), portanto são modeladas duas redes de Petri, sendo uma para cada

comportamento. Para o comportamento do robô em relação ao feromônio, tem-se a rede de Petri da Fig. 3.7. Na marcação inicial (P_1), o robô encontra-se no ninho entre as etiquetas Feromônio 1 e Feromônio 2. Quando o robô encontra a etiqueta Feromônio 2, a transição T_1 é disparada e o robô passa para o lugar P_2 (sobre a etiqueta). Neste momento o robô recebe a informação da etiqueta representada por uma das *pseudoboxes* ($PS1$, $PS2$ ou $PS3$) e dispara a transição que estiver habilitada. O comportamento na forma matemática é disposto nas Eqs. 3.2-3.4.

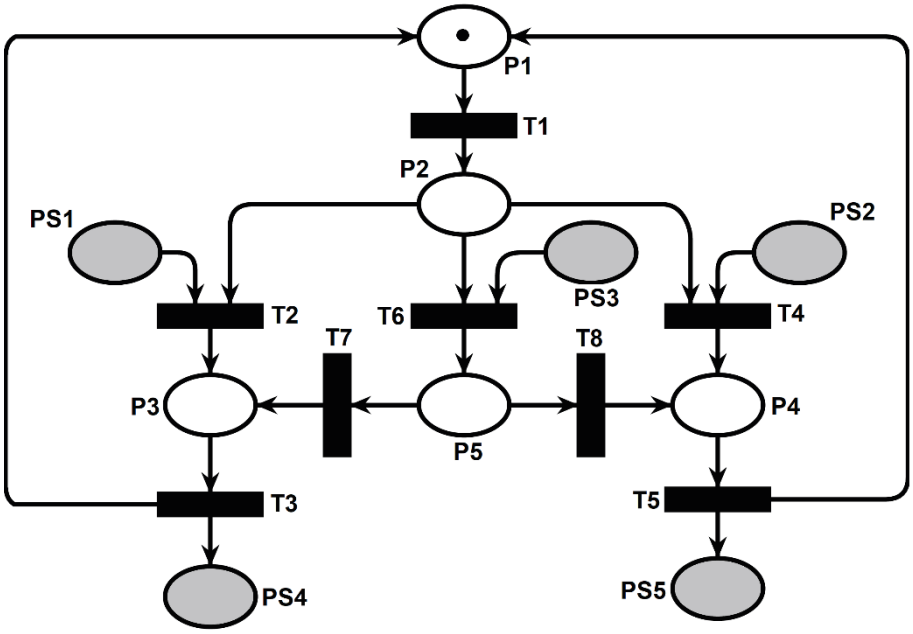


Figura 3.7 – Modelo comportamental do robô móvel – feromônio em rede de Petri.

$$P = \{P_1, P_2, \dots, P_5\} \tag{3.2}$$

$$T = \{T_1, T_2, \dots, T_8\} \tag{3.3}$$

$$M_0 = [1 \quad 0 \quad 0 \quad 0 \quad 0]^t \tag{3.4}$$

Quando a etiqueta do feromônio não indica o caminho a ser seguido ($PS3$), a transição T_6 é disparada. Por se tratar de uma rede de Petri de alto nível, o disparo da transição é realizado através de um código que altera a cor do *token*, conforme Quad. 3.2.

Já as transições T_7 e T_8 são disparadas de acordo com o *token* em P_5. No caso vermelho o robô segue pela trilha A (T_7 disparado), no caso azul o robô segue pela trilha B. As transições T_3 e T_5 são disparadas quando o robô retorna ao ninho e encontra a etiqueta Feromônio 1. Neste momento o robô informa para o feromônio, através das *pseudoboxes*

$PS4$ e $PS5$ a trilha percorrida. A Tab. 3.1 apresenta a descrição dos lugares e transições do robô móvel – feromônio.

Quadro 3.2 – Pseudocódigo para disparo da transição T_6.

```
Se aleatório (0..1) < 0,5 então
        Token é vermelho
Senão
        Token é azul
Fim-se
```

Tabela 3.1 – Descrição dos lugares e transições utilizadas na modelagem do robô móvel – feromônio.

Lugar/Transição		Descrição
lugares	P_1	Robô movimentando no ninho entre as etiquetas Feromônio 1 e Feromônio 2
	P_2	Robô sobre a etiqueta Feromônio 2
	P_3	Robo movimentando pela trilha A
	P_4	Robô movimentando pela trilha B
	P_5	Robô decidindo a trilha que irá seguir
transições	T_1	Leitura da etiqueta Feromônio 2
	T_2	Seguir trilha A
	T_3	Leitura da etiqueta Feromônio 1 por A
	T_4	Seguir trilha B
	T_5	Leitura da etiqueta Feromônio 1 por B
	T_6	Escolha da trilha que irá seguir
	T_7	Escolha da trilha A
	T_8	Escolha da trilha B

As *pseudoboxes* são descritas na Tab. 3.2. As três primeiras são as informações do ambiente para o robô enquanto as duas últimas são informações do robô para o ambiente.

Tabela 3.2 – Descrição das *pseudoboxes* de comunicação entre o robô móvel e o ambiente feromônio.

Pseudobox	**Descrição**
PS1	Trilha A indicada pela etiqueta Feromônio 2
PS2	Trilha B indicada pela etiqueta Feromônio 2
PS3	Etiqueta Feromônio 2 sem indicação de trilha
PS4	Retorno pela trilha A indicado
PS5	Retorno pela trilha A indicado

Como o feromônio e o semáforo são atualizados a cada cinco segundos, há mais de uma possibilidade de modelagem dos agentes. Pode-se utilizar redes de Petri temporizadas para determinar o momento de atualização da rede, bem como pode-se utilizar um *clock* para coordenar os disparos da rede como utilizado em Ferreira, Tavares e Silva (2018). Porém, percebe-se que a atualização periódica é decorrente do problema de conflito de leitura do dispositivo eletrônico utilizado e não é inerente ao agente (feromônio ou semáforo). Com isso, optou-se por uma modelagem comportamental em que os tempos de disparo não são considerados, uma vez que caso haja a troca do dispositivo de leitura não há a necessidade de alteração na modelagem dos agentes. Sendo assim, a cada cinco segundos as transições podem ser disparadas.

O ambiente feromônio possui a rede de Petri modelada conforme Fig. 3.8. As marcações iniciais são de feromônio A desatualizado (P_6) e feromônio B desatualizado (P_8). O feromônio da trilha A pode ser incrementado (quando o robô indica o retorno através da *pseudobox* $PS4$) ou decrementado quando não há marcação na *pseudobox*. O arco inibidor em T_9 indica desabilita o decremento quando há informação de retorno na etiqueta Feromônio 1. A mesma situação ocorre para a trilha B com a pseudobox $PS5$ e a transição T_{10}. O comportamento na forma matemática é disposto nas Eqs. 3.5-3.7.

$$P = \{P_6, P_7, \dots, P_{10}\} \tag{3.5}$$

$$T = \{T_9, T_{10}, \dots, T_{16}\} \tag{3.6}$$

$$M_0 = [1 \quad 0 \quad 1 \quad 0 \quad 0]^t \tag{3.7}$$

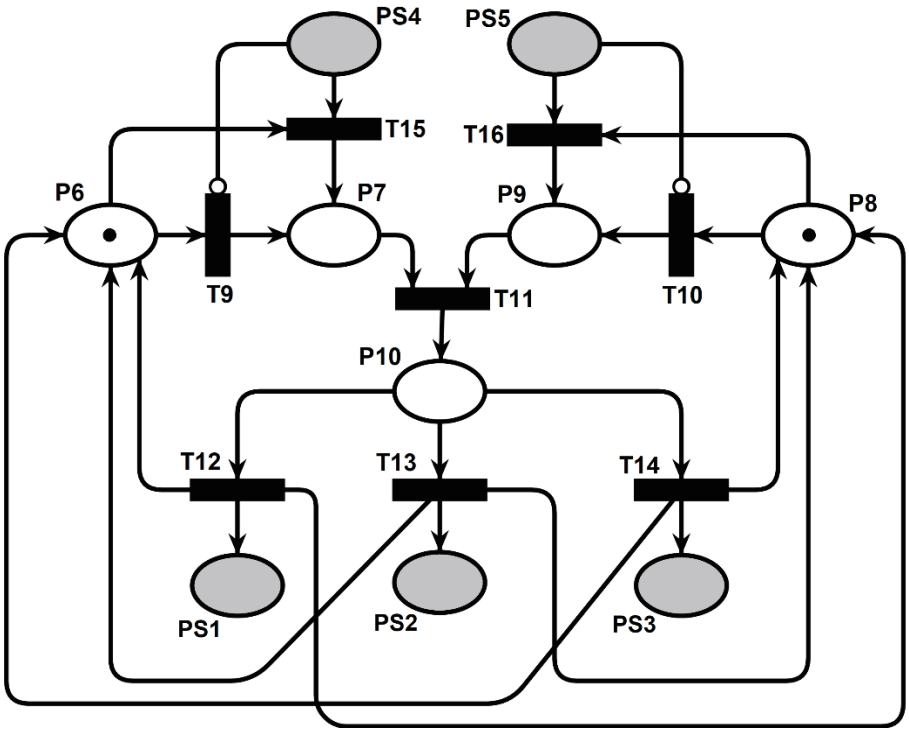


Figura 3.8 – Modelo comportamental do feromônio em rede de Petri.

A transição T_{11} altera a cor do token de acordo com o Quad. 3.3

Quadro 3.3 – Pseudocódigo para disparo da transição T_{11}.

```
Se ferA/(ferA + ferB) > 0,52 então
        Token é vermelho
Senão se ferB/(ferA + ferB) > 0,52 então
        Token é azul
Senão
        Token é verde
Fim-se
```

As transições T_{12}, T_{13} e T_{14} são disparadas de acordo com a cor do *token* em P_{10}. Se o *token* for vermelho, então a transição T_{12} é disparada. Se for azul, a transição T_{13} é disparada. No caso de *token* verde, a transição T_{14} é disparada.

A Tab. 3.3 apresenta a descrição dos lugares e transições utilizadas na modelagem do feromônio.

Tabela 3.3 – Descrição dos lugares e transições utilizadas na modelagem do feromônio.

Lugar/Transição		Descrição
lugares	P_6	Quantidade de feromônio A desatualizada
	P_7	Quantidade de feromônio A atualizada
	P_8	Quantidade de feromônio B desatualizada
	P_9	Quantidade de feromônio B atualizada
	P_{10}	Feromônio atualizado
transições	T_9	Decrementar quantidade de feromônio A
	T_{10}	Decrementar quantidade de feromônio B
	T_{11}	Atualizar feromônio
	T_{12}	Feromônio A maior do que feromônio B
	T_{13}	Feromônio B maior do que feromônio A
	T_{14}	Feromônios iguais
	T_{15}	Incrementar quantidade de feromônio A
	T_{16}	Incrementar quantidade de feromônio B

3.5.2. Robô móvel e semáforo

O comportamento do robô móvel em relação ao semáforo (Fig. 3.9) pode ser definido pode três lugares (robô fora do semáforo, robô no semáforo A e robô no semáforo B). O lugar P_{11} possui concorrência entre as transições T_{17} e T_{19}, porém não é necessário utilizar de uma rede de Petri de alto nível para este modelo, pois, as transições são disparadas por etiquetas distintas.

Quando o robô segue a trilha A, a transição T_{17} será disparada quando o robô encontra a etiqueta semáforo A. Já a transição T_{18} será disparada quando o robô encontra a etiqueta semáforo B.

A sequência de comunicação em cada etiqueta é a seguinte:

1º o robô indica a presença no semáforo ($PS6$ ou $PS9$)

2º o robô recebe o sinal verde do semáforo ($PS7$ ou $PS10$)

3º o robô indica que está passando pelo semáforo ($PS8$ ou $PS11$)

O comportamento na forma matemática é disposto nas Eqs. 3.8-3.10.

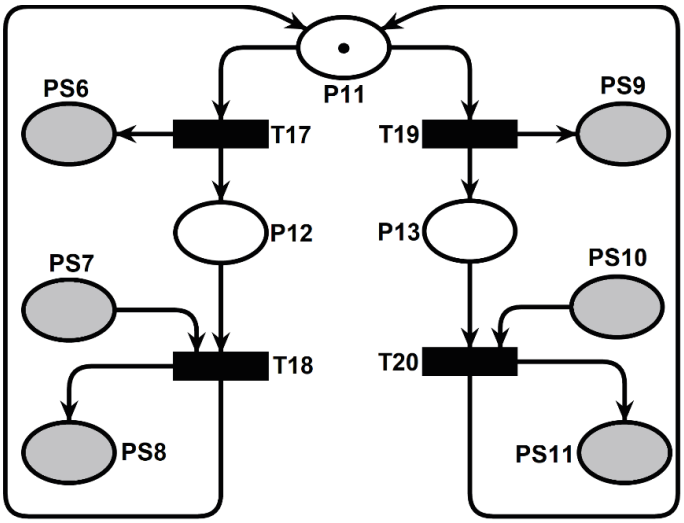


Figura 3.9 – Modelo comportamental do robô móvel – semáforo em rede de Petri.

$$P = \{P_{11}, P_{12}, P_{13}\} \tag{3.8}$$

$$T = \{T_{17}, T_{18}, \dots, T_{20}\} \tag{3.9}$$

$$M_0 = [1 \quad 0 \quad 1]^t \tag{3.10}$$

A Tab. 3.4 apresenta a descrição dos lugares e transições utilizadas na modelagem do robô móvel – semáforo, enquanto a Tab. 3.5 apresenta as *pseudoboxes* utilizadas para a comunicação entre o robô e o ambiente semáforo.

A modelagem do semáforo basea-se em dois estados possíveis: verde e vermelho. Além isso há dois estados de utilização: livre e ativo. Semáforo livre (P_{22}) indica que está vermelho para as duas trilhas, e o robô que chegar primeiro em uma das trilhas terá direito de passagem. Já semáfor ativo (P_{23}) indica que está verde para uma das trilhas, ou o semáforo ainda não foi reiniciado. A Fig. 3.10 apresenta a modelagem do semáforo em rede de Petri enquanto a Tab. 6 apresenta a descrição dos lugares e transições da modelagem. Inicialmente há três marcações: Semáforo SA vermelho (P_{14}), Semáforo SB vermelho (P_{18}) e Semáforo livre (P_{22}). A prioridade do caminho mais curto está relacionada pelo arco inibidor entre P_{15} e T_{25}, ou seja, a trilha A possui prioridade de passagem em relação a trilha B. O comportamento na forma matemática é disposto nas Eqs. 3.11-3.13.

Tabela 3.4 – Descrição dos lugares e transições utilizadas na modelagem do robô móvel – semáforo.

Lugar/Transição		Descrição
lugares	P_{11}	Robô fora do semáforo
	P_{12}	Robô no semáforo na trilha A
	P_{13}	Robô no semáforo na trilha B
transições	T_{17}	Leitura da tag SA chegada
	T_{18}	Leitura da tag SA saída
	T_{19}	Leitura da tag SB chegada
	T_{20}	Leitura da trilha SB saída

Tabela 3.5 – Descrição das *pseudoboxes* de comunicação entre o robô móvel e o ambiente semáforo.

Pseudobox	**Descrição**
PS6	Presença no semáforo SA indicada
PS7	Liberação do semáforo SA recebida
PS8	Saída do semáforo SA indicada
PS9	Presença no semáforo SB indicada
PS10	Liberação do semáforo SB recebida
PS11	Saída do semáforo SB indicada

$$P = \{P_{14}, P_{15}, \dots, P_{24}\} \tag{3.11}$$

$$T = \{T_{21}, T_{22}, \dots, T_{29}\} \tag{3.12}$$

$$M_0 = [1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0]^t \tag{3.13}$$

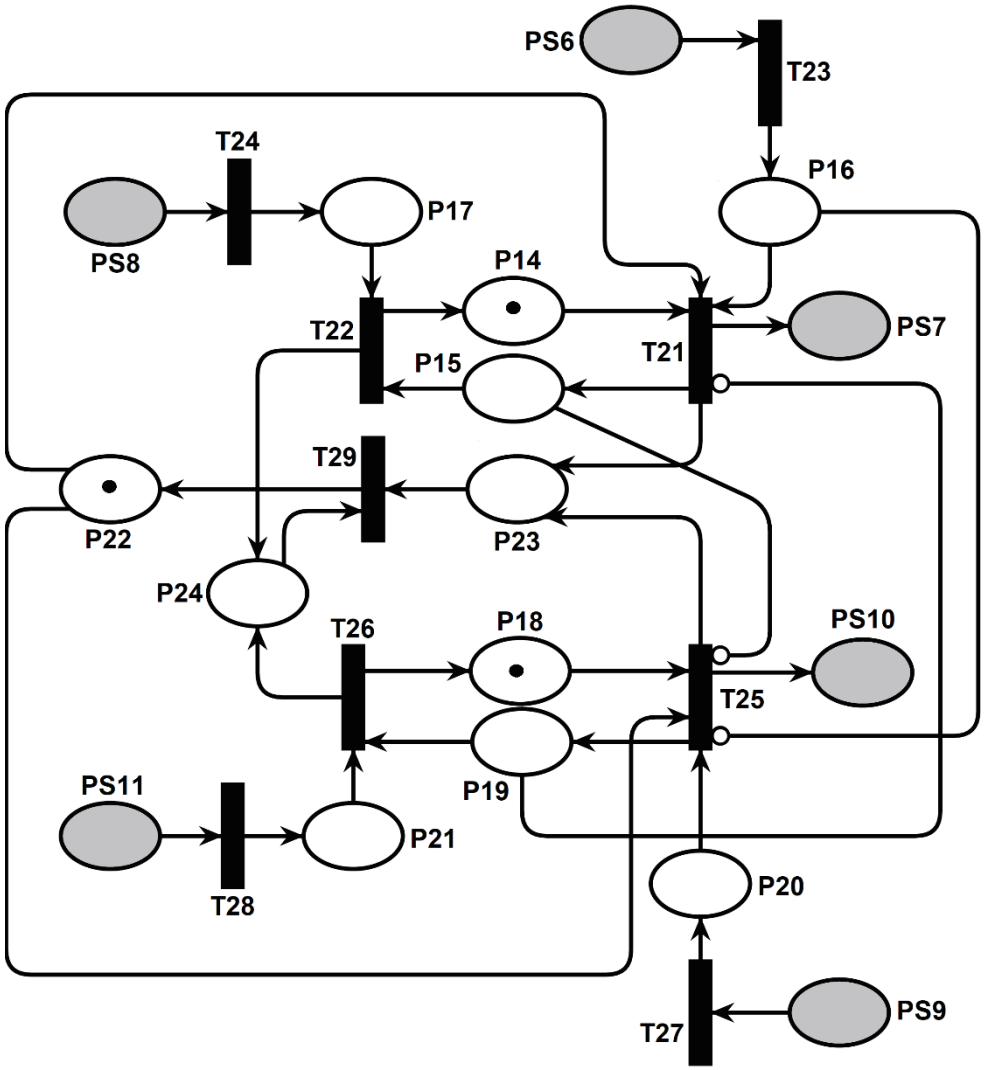


Figura 3.10 – Modelo comportamental do semáforo em rede de Petri.

Tabela 3.6 – Descrição dos lugares e transições utilizadas na modelagem do semáforo.

Lugar/Transição		Descrição
lugares	P_{14}	Semáforo SA vermelho
	P_{15}	Semáforo SA verde
	P_{16}	Robô no semáforo SA
	P_{17}	Robô saindo do semáforo SA
	P_{18}	Semáforo SB vermelho
	P_{19}	Semáforo SB verde
	P_{20}	Robô no semáforo SB
	P_{21}	Robô saindo do semáforo SB
	P_{22}	Semáforo livre
	P_{23}	Semáforo ocupado
	P_{24}	Pronto para liberar o semáforo
transições	T_{21}	Mudar semáforo SA para verde
	T_{22}	Mudar semáforo SA para vermelho
	T_{23}	Indicar robô entrando em SA
	T_{24}	Indicar robô saindo de SA
	T_{25}	Mudar semáforo SB para verde
	T_{26}	Mudar semáforo SB para vermelho
	T_{27}	Indicar robô entrando em SB
	T_{28}	Indicar robô saindo de SB
	T_{29}	Liberar semáforo

3.5.3. Visão geral das redes de Petri.

As relações entre as redes de Petri através das *pseudoboxes* utilizadas em cada etiqueta estão apresentadas nas Fig. 3.11 (robô – feromônio) e Fig. 3.12 (robô – semáforo). As *pseudoboxes* iguais são representadas pela mesma cor, enquanto os lugares de mesma tonalidade representam as etiquetas.

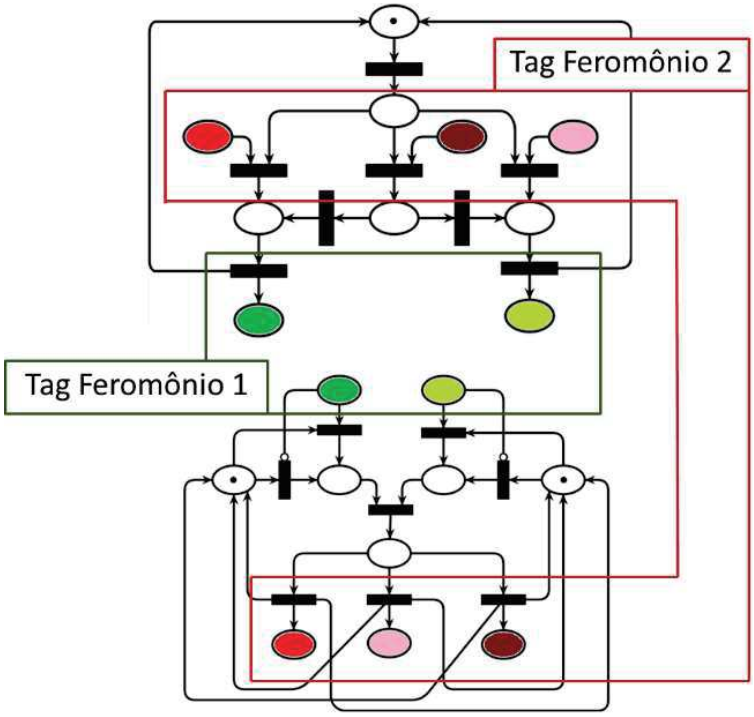


Figura 3.11 – Modelo comportamental do semáforo em rede de Petri.

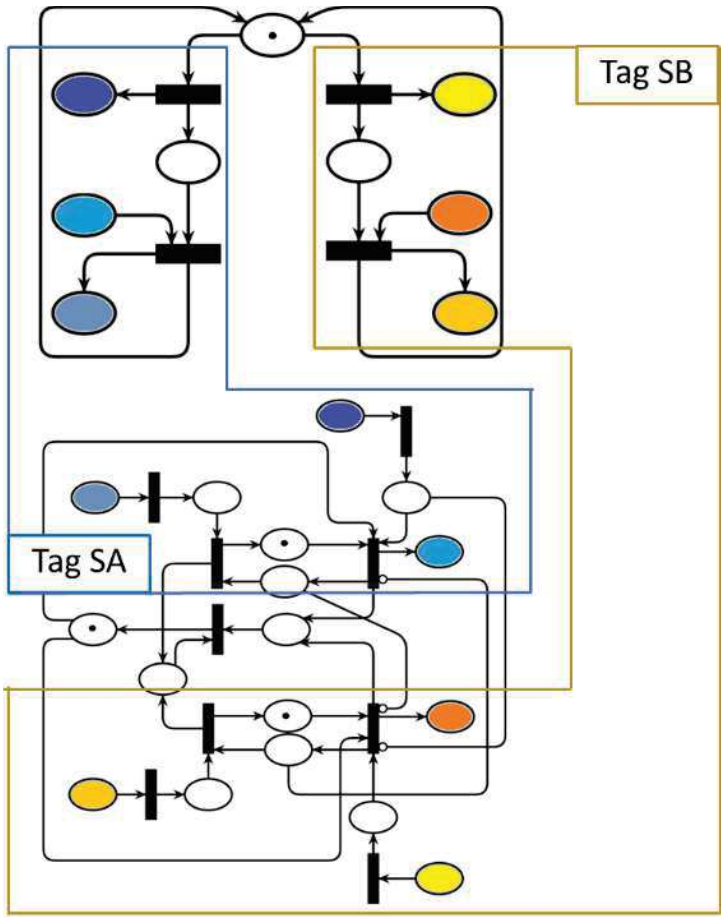


Figura 3.12 – Modelo comportamental do semáforo em rede de Petri.

CAPÍTULO IV

IMPLEMENTAÇÃO E TESTES

A solução proposta foi implementada de duas formas: em uma bancada física e em um ambiente de simulação. Para enxames de robôs, uma bancada física torna-se impraticável quando o tamanho do enxame aumenta, devido aos custos envolvidos nos experimentos. Sendo assim, torna-se necessário um ambiente de simulação para explorar situações mais complexas. Todavia um ambiente de simulação deve ser validado operacionalmente com dados de um sistema físico.

O intuito das duas implementações é de comparar o comportamento na bancada física com o comportamento no ambiente de simulação. Caso o robô e o ambiente interajam da mesma forma, pode-se utilizar o ambiente de simulação para situações mais complexas e inferir os comportamentos do robô e do ambiente.

Os critérios de avaliação são inclinação da curva de subida da quantidade de feromônio, decaimento da quantidade de feromônio ao longo do tempo e indicação da trilha, além do tempo necessário para esgotar a quantidade de alimento no sistema. Estes critérios estão detalhados no item 4.3.

Os códigos da implementação experimental encontram-se no Apêndice B, enquanto os códigos da simulação computacional encontram-se no Apêndice C.

4.1. Implementação experimental

4.1.1. Robô móvel

Ao realizar os testes do robô móvel no ambiente, foi observado um conflito de leitura dos módulos PN532 do leitor e do ambiente, como apresentado no capítulo 3. Com isso foi acrescentado dois componentes no robô: mais um leitor ultrassônico HC-SR04 e um módulo relé. A Fig. 4.1 mostra a representação esquemática dos componentes eletrônicos aplicados

ao robô móvel. O sensor ultrassônico adicional, inserido na lateral direita do robô, é responsável por verificar a distância entre o robô e a parede. Os pontos de leitura das etiquetas foram modificados de modo que o sensor ultrassônico identifique este ponto. Sendo assim, o robô só habilita o módulo (através do módulo relé) quando realmente é necessário fazer a leitura. Quando o leitor do robô e qualquer um dos leitores do ambiente fazem a leitura da etiqueta ao mesmo tempo, ocorre uma falha, hora no robô, hora no ambiente.

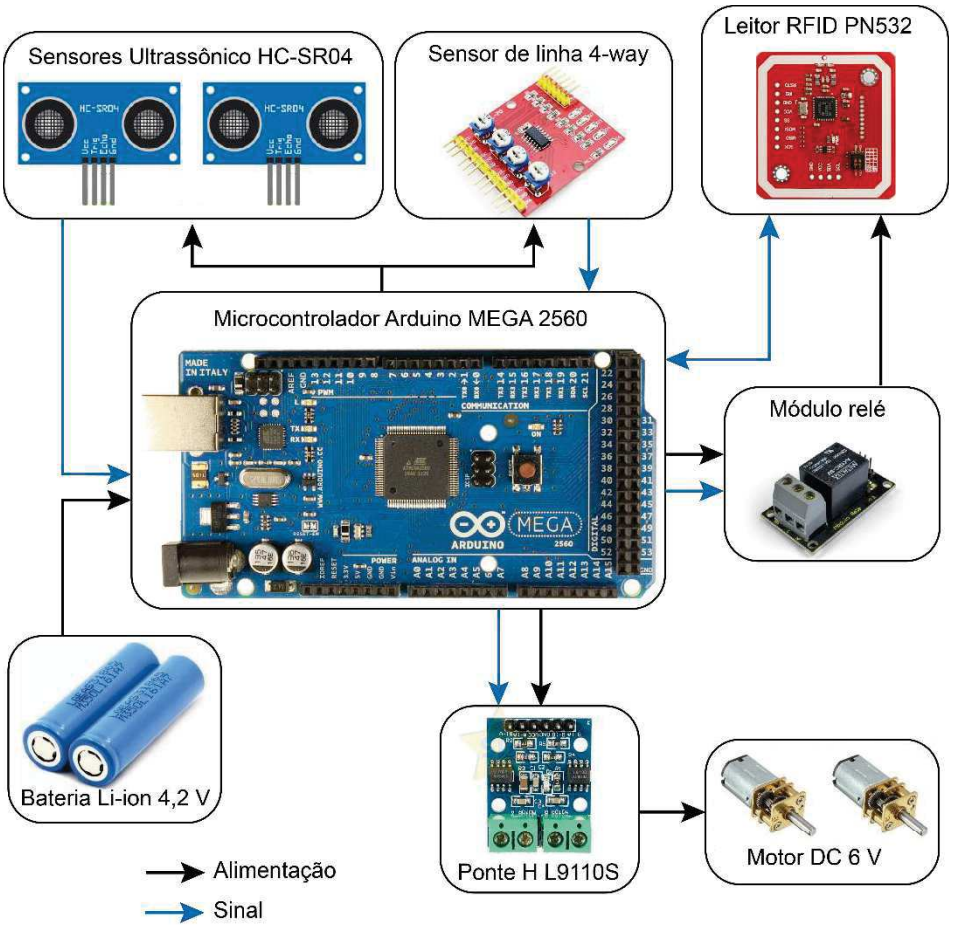


Figura 4.1 – Representação esquemática dos componentes eletrônicos aplicados ao robô móvel.

A Fig. 4.2 mostra o robô móvel em três posições diferentes. No robô da esquerda é possível verificar o sensor ultrassônico e o sensor de faixa responsáveis pelo controle reativo do robô. No robô do centro, a face da direita está visível e percebe-se nela o ultrassônico inserido a posteriori para indicar o ponto de leitura da etiqueta. Já no robô da direita, a face traseira está visível com o microcontrolador Arduino Mega 2560 e o módulo relé que habilita o leitor RFID.

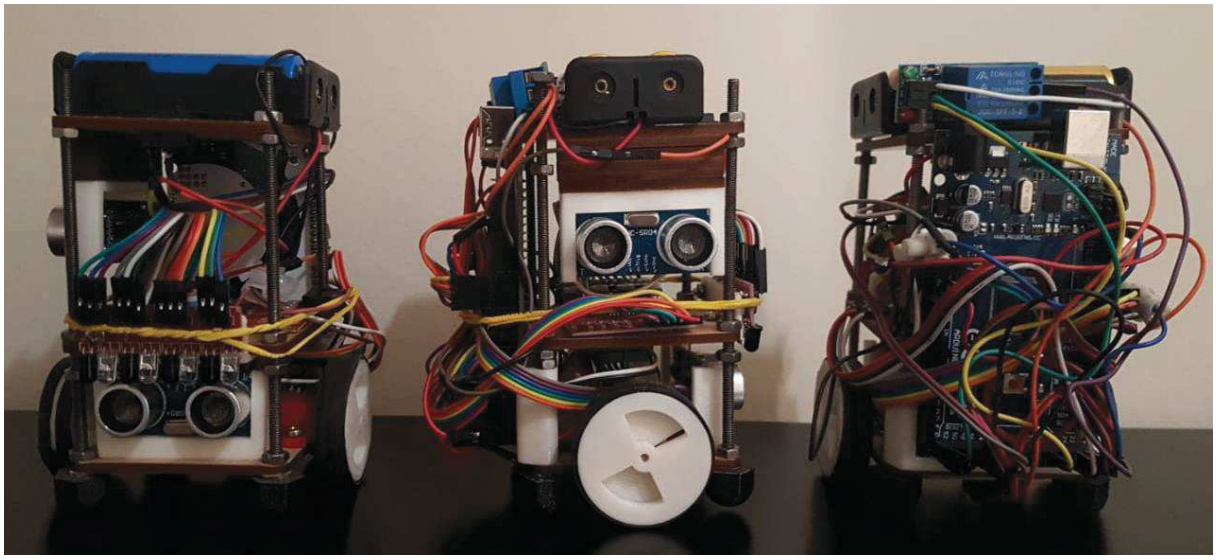


Figura 4.2 – Fotografia detalhando o robô móvel em três posições diferentes.

4.1.2. Bancada experimental

A bancada experimental (Fig. 4.3) possui na parte superior o caminho por onde os robôs transitam e na parte inferior toda eletrônica responsável pelo ambiente (feromônio e semáforo). Os paralelepípedos na área interna da bancada indicam para os robôs os pontos de leitura das etiquetas.

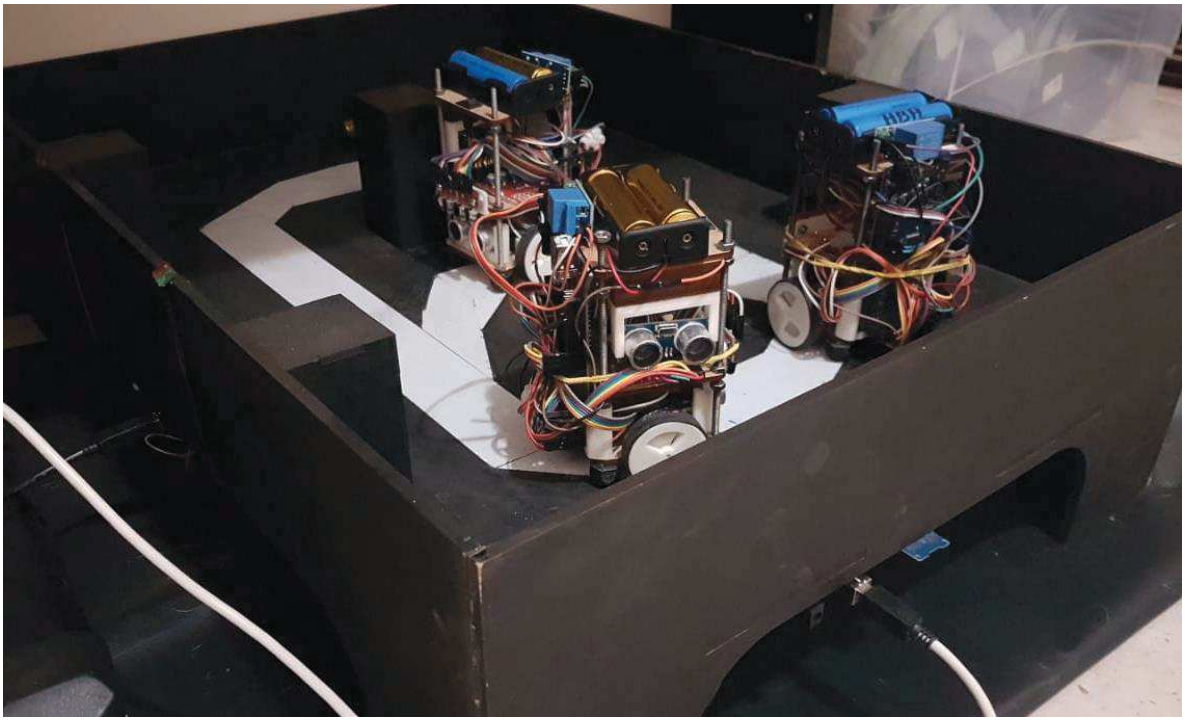


Figura 4.3 – Fotografia da bancada experimental com área interna de 50 cm x 70 cm.

4.1.3. Feromônio

O feromônio de formiga é emulado em um microcontrolador Arduino Mega 2560 (Fig. 4.4). Para implementação do feromônio é necessário utilizar dois leitores PN532 para leitura e escrita das etiquetas (*Tag Feromônio 1* e *Tag Feromônio 2*). Os leitores são habilitados através de um módulo relé de 2 canais que é acionado a cada cinco segundos, conforme explicado no capítulo 3. O modo de execução do feromônio é dividido em três etapas:

1) Leitura da etiqueta: o módulo relé habilita a leitura da etiqueta *Tag Feromônio 1* e se há informações de entrada para o controlador, o *token* é gerado em uma das *pseudoboxes* de entrada da Rede de Petri do feromônio.
2) Cálculo do feromônio: a quantidade de feromônio em cada trilha é atualizada (conforme Eq. 3.1) e então é realizada a comparação para determinação da trilha a ser indicada.
3) Atualização da etiqueta: a trilha indicada é salva na *Tag Feromônio 2*.

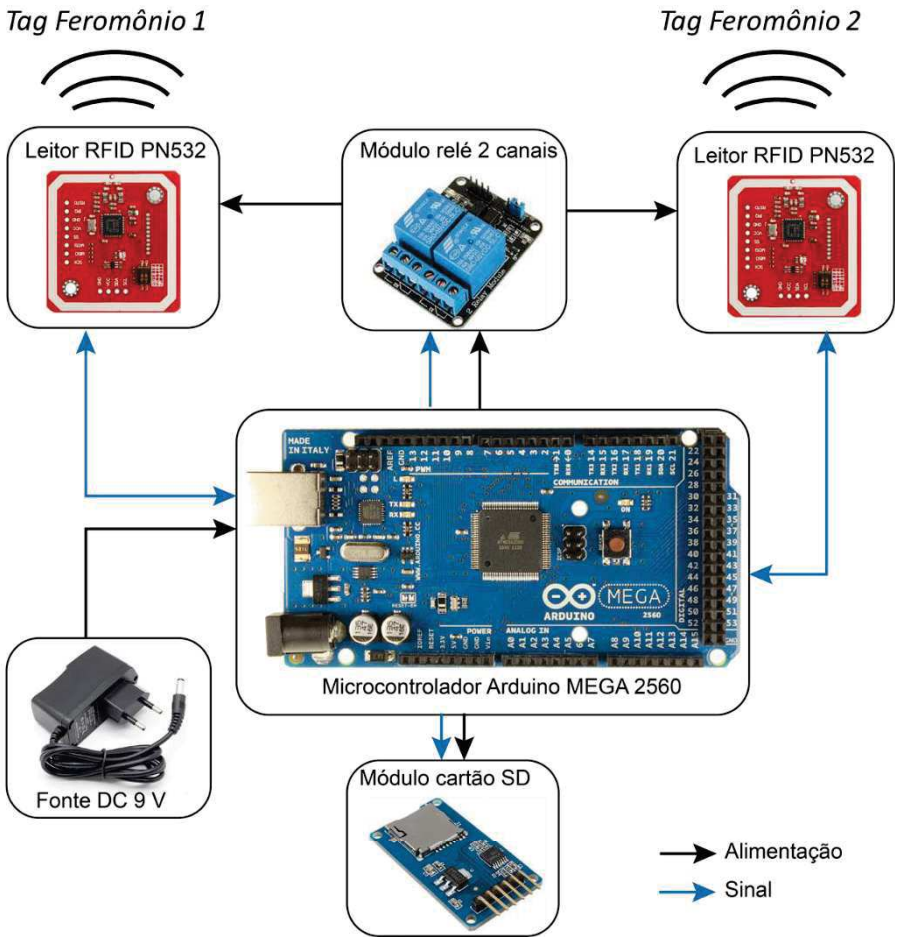


Figura 4.4 – Representação esquemática dos componentes eletrônicos aplicados ao feromônio.

As análises dos resultados são baseadas na quantidade de feromônio em cada trilha, na trilha indicada em cada momento e na quantidade de alimento total de cada trilha ao longo do tempo. Para isso fez-se necessário utilizar um módulo de cartão SD para salvar os dados coletados em arquivo .txt.

4.1.4. Semáforo

O semáforo também implementado em um microcontrolador Arduino Mega 2560 utiliza dois leitores PN532 e dois conjuntos de LEDs verde e vermelho para indicação visual do semáforo. Assim como no feromônio, o módulo relé habilita a leitura do semáforo a cada cinco segundos. A Fig. 4.5 mostra a representação esquemática dos componentes eletrônicos aplicados ao semáforo.

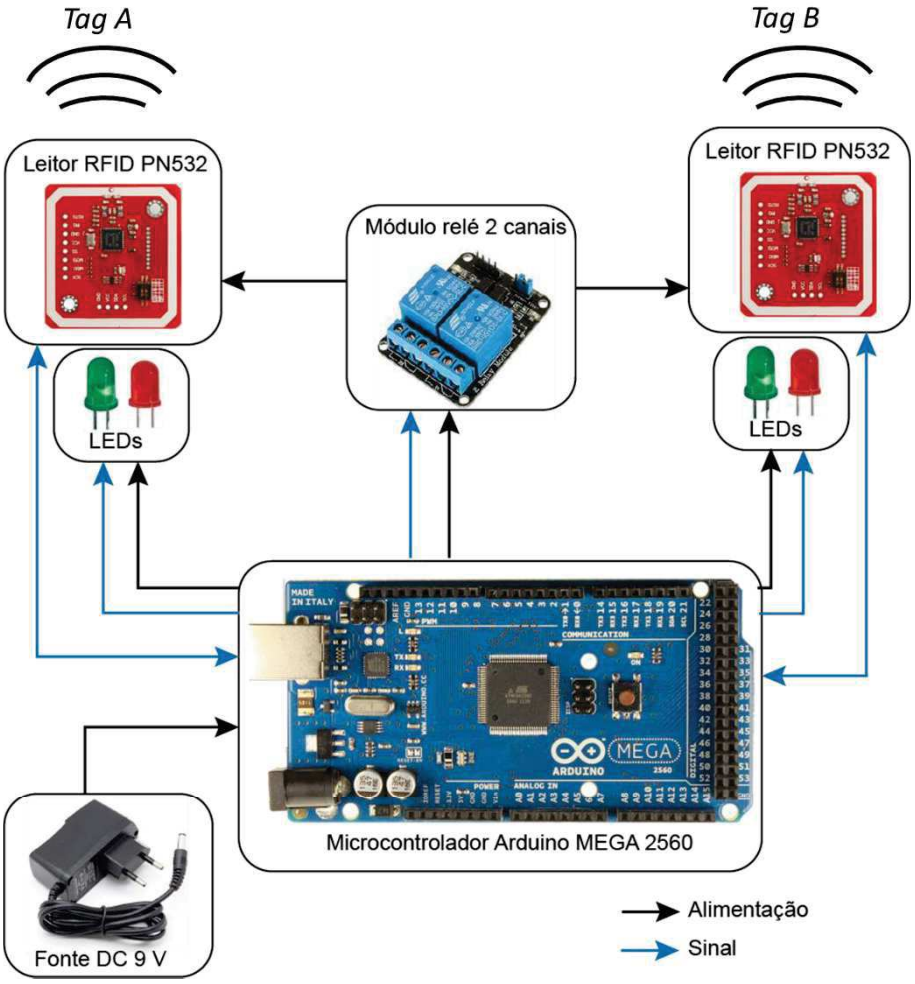


Figura 4.5 – Representação esquemática dos componentes eletrônicos aplicados ao semáforo.

O modo de execução do semáforo também é dividido em três etapas:

1) Leitura das etiquetas: as etiquetas *Tag A* e *Tag B* são lidas e os possíveis *tokens* são gerados nas *pseudoboxes* de entrada da rede de Petri do semáforo.
2) Atualização da Rede de Petri: todas as transições habilitadas são disparadas e as marcações passam para os devidos lugares.
3) Escrita na etiqueta: se há marcação em algum lugar que altera a etiqueta, a gravação é realizada neste momento.

4.1.5. Mensagens trocadas entre os agentes

Os três agentes se relacionam por mensagens trocadas através das etiquetas RFID. As mensagens foram definidas com dois bits, totalizando quatro possíveis informações [0 0], [0 1], [1 0] e [1 1].

Considerando a situação em que o robô saí do ninho, ao encontrar a etiqueta *Tag Feromônio 2* o robô receberá a informação de qual trilha deve seguir. Se a mensagem salva na etiqueta for [1 0] ($PS1$), o robô seguirá a trilha A; se a mensagem for [0 1] ($PS2$), o robô seguirá a trilha B; porém se a mensagem for [0 0] ($PS3$), o robô que determinará o caminho que deve seguir de forma randômica. A Fig. 4.6 apresenta o diagrama de sequência da etiqueta. Inicialmente o feromônio atualiza a etiqueta com a mensagem correspondente e todos robôs móveis que passam por essa etiqueta, ao realizarem a leitura, recebem a indicação da trilha.

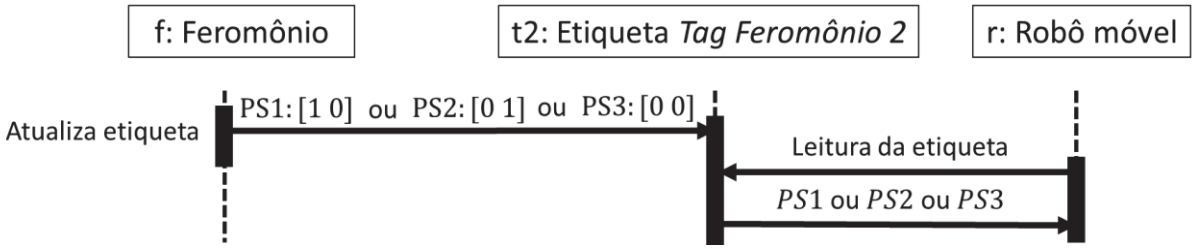


Figura 4.6 – Diagrama de sequência da etiqueta *Tag Feromônio 2*.

Em seguida, se o robô seguir a trilha A, encontrará a etiqueta *Tag A*. Neste momento a mensagem salva na etiqueta é [0 0]. Então ao fazer a leitura, o robô indica sua presença no semáforo com a mensagem [0 1] ($PS4$). O semáforo ao receber esta mensagem atualiza o semáforo para verde e salva a mensagem [1 0] ($PS5$) na etiqueta, caso seja possível o disparo da rede de Petri. O robô então, faz a leitura da etiqueta com esta mensagem e salva a indicação de saída do semáforo [1 1] ($PS6$). Quando o semáforo recebe esta informação,

altera a mensagem da etiqueta para [0 0] para que outro robô possa utilizar o semáforo. A Fig. 4.7 apresenta o diagrama de sequência desta etiqueta.

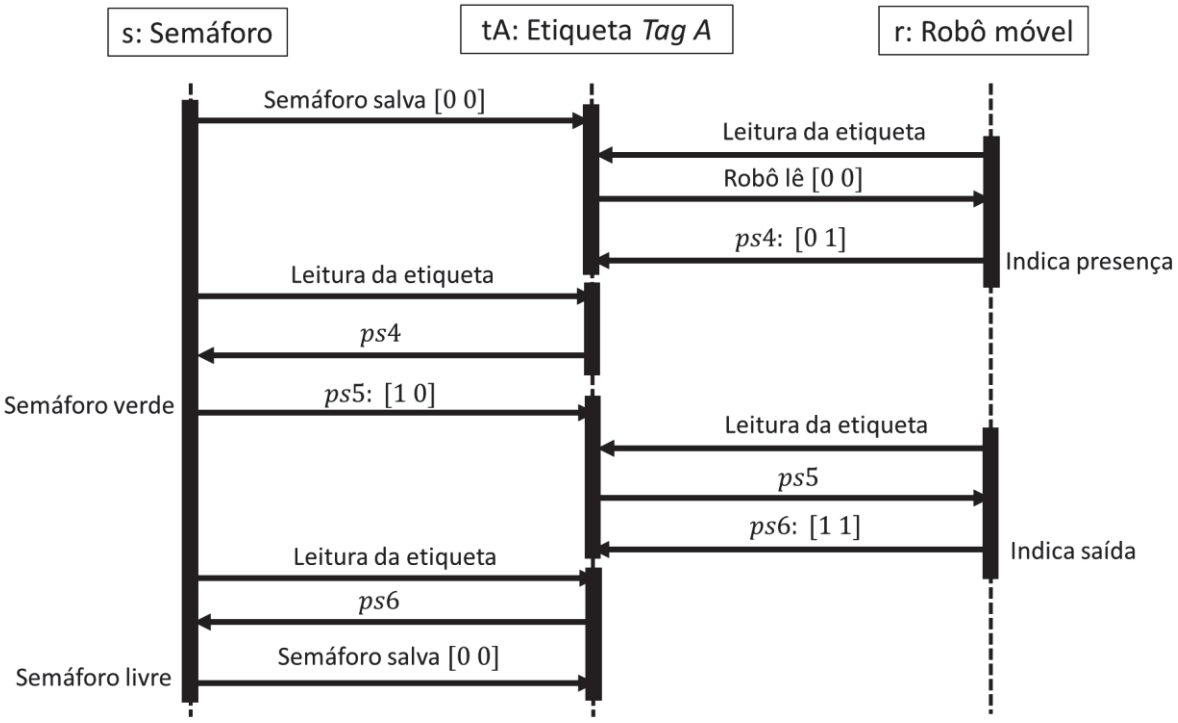


Figura 4.7 – Diagrama de sequência da etiqueta *Tag A*.

Caso o robô siga a trilha B, a mesma sequência acontece, porém com a etiqueta *Tag B.* As únicas mudanças entre os dois diagramas são as nomenclaturas utilizadas. Para o primeiro caso as *pseudoboxes* são: *PS4, PS5* e *PS6,* enquanto para o segundo caso são: *PS8, PS9* e *PS10.* A Fig. 4.8 apresenta o diagrama de sequência para a etiqueta *Tag B.*

Por fim o robô retorna ao ninho, e ao encontrar a etiqueta *Tag Feromônio 1* salva o caminho percorrido: [1 0] (*PS7)* quando seguiu a trilha A, ou [0 1] *(PS11*) quando seguiu a trilha B. O robô só conhece o caminho que está percorrendo quando encontra as etiquetas do semáforo. Com isso, se um robô receber um caminho da etiqueta *Tag Feromônio 2* e, por algum motivo, seguir outro caminho ele saberá o caminho que percorreu e salvará a etiqueta *Tag Feromônio 1* de forma correta. O diagrama de sequência desta etiqueta está apresentado na Fig. 4.9.

4.2. Simulação computacional

A construção do cenário de simulação em V-REP iniciou-se com a criação do modelo do robô móvel e do ambiente de interação semelhante ao construído para testes

experimentais. A Fig. 4.10 apresenta o ambiente de simulação. Na imagem é possível verificar quatro pontos sobre faixa branca. Tais pontos são as etiquetas do feromônio e do semáforo. Elas estão posicionadas da mesma forma que a bancada experimental, ou seja, a etiqueta mais próxima do robô na imagem é a *Tag Feromônio 2.*

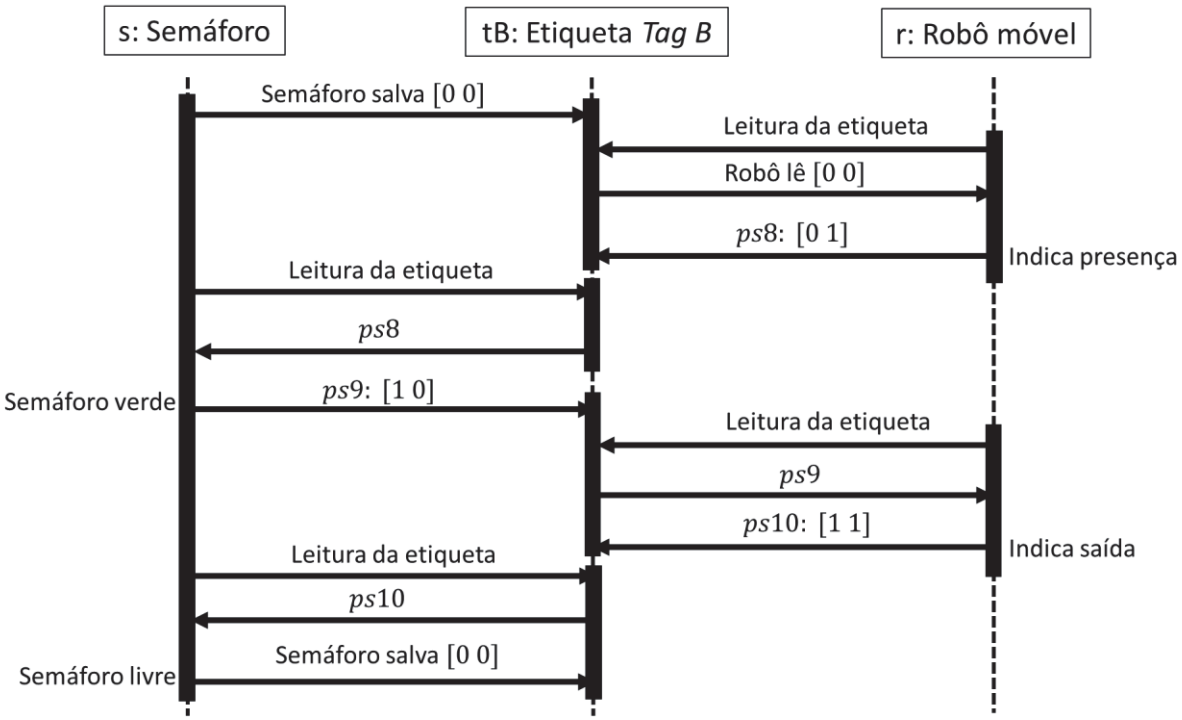


Figura 4.8 – Diagrama de sequência da etiqueta *Tag SB*.

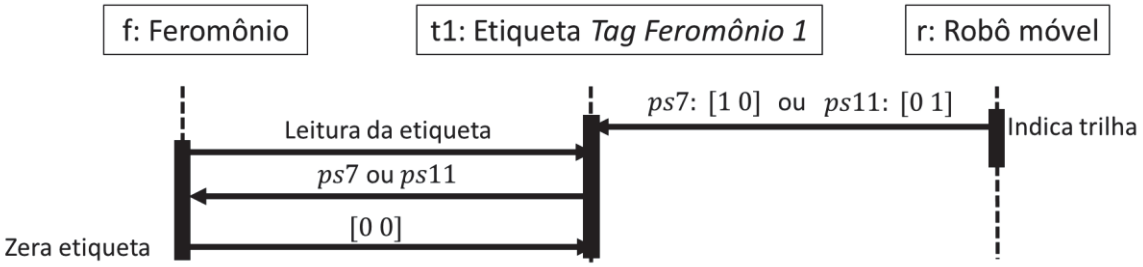


Figura 4.9 – Diagrama de sequência da etiqueta *Tag Feromônio 1*.

Como o ambiente de simulação tem como foco analisar o comportamento dos robôs diante das informações contidas no feromônio, apenas um robô foi utilizado durante as simulações. Sendo assim, como não há problema de colisão de robôs, o semáforo não foi implementado no ambiente de simulação. As duas etiquetas referentes ao semáforo foram utilizadas para indicar ao robô qual caminho foi percorrido.

A árvore hierárquica na lateral esquerda da Fig. 4.10 apresenta as etiquetas do feromônio e do semáforo além do robô móvel (que foi inserido de arquivo CAD – *Computer*

Aided Design). A estrutura de leitores e etiquetas RFID foram implementados através de cuboides primitivos. As funções *sim.setStringSignal()* e *sim.getStringSignal()* foram utilizadas para transmitir as informações entre os *scripts* dos mesmos agentes. Por exemplo, as etiquetas do feromônio *Tag FA* e *Tag FB* trocam informações para atualizar a quantidade de feromônio em cada trilha e o leitor do robô troca informação com o *script* de controle do robô (*chassis_dyn*).

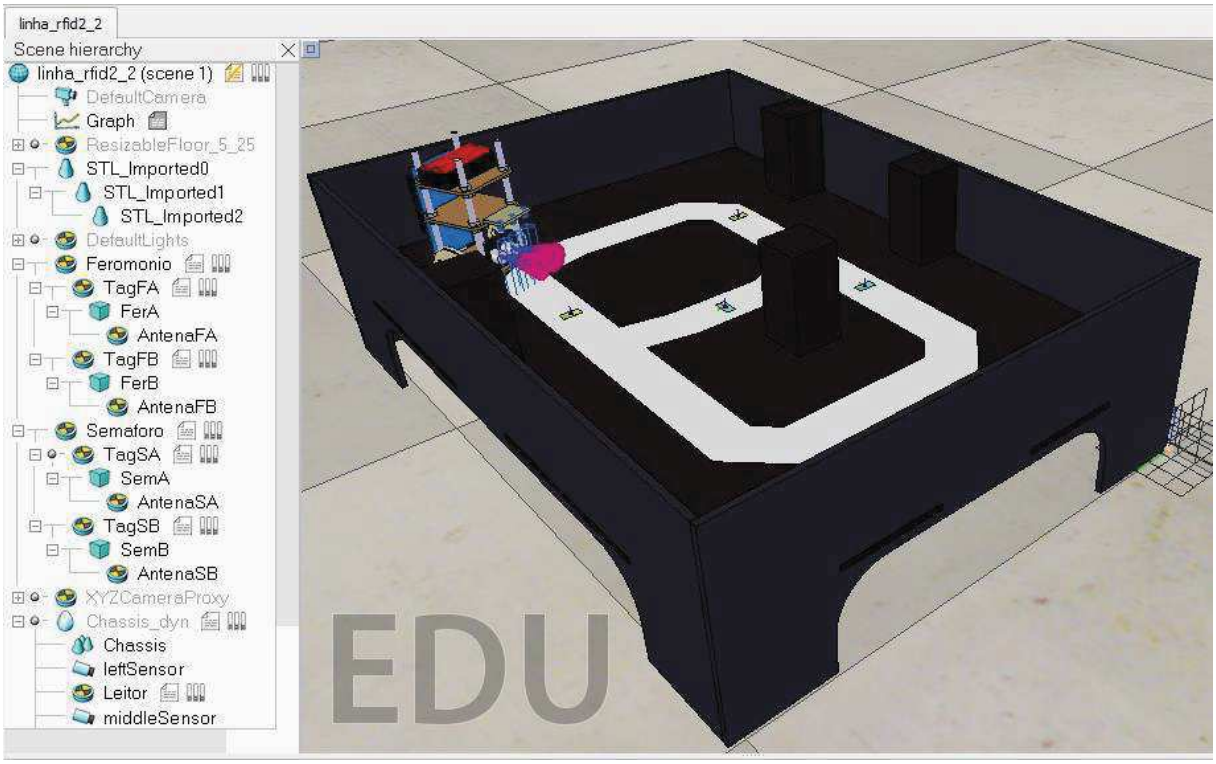


Figura 4.10 – Ambiente de simulação composto por um robô móvel e da bancada experimental.

O sensor de faixa do robô móvel utiliza quatro canais (emissor e receptor), enquanto o robô implementado no simulador possui apenas três canais, que é a quantidade suficiente para fazer com que o robô siga a faixa. O robô móvel só possui quatro canais porque o módulo sensor de linha é de quatro canais.

4.3 Cenários avaliados

Para avaliação da relação robô/ambiente e como o feromônio influencia na escolha das trilhas seguidas pelos agentes móveis foram avaliados quatro cenários em que foram alteradas as quantidades de alimento disponíveis nas trilhas e o período em que o alimento foi disponibilizado na trilha. Com a combinação destas características é possível observar a

forma com que o feromônio deve influenciar na coleta de alimento das trilhas ao longo do tempo, e se o mesmo pode ser utilizado para tomada de decisões em tarefas mais complexas. A Tab. 4.1 apresenta os quatro cenários avaliados com as principais diferenças.

Para todos os cenários, considera-se que o robô é capaz de coletar uma quantidade randômica de alimento, dentro de um limite mínimo e máximo de coleta de alimento.

O primeiro cenário avaliado é o cenário em que a quantidade de alimento disponível nas duas trilhas é ilimitada, ou seja, o robô pode explorar a trilha infinitamente e nunca retornará ao ninho sem alimento. Neste cenário a quantidade de alimento nas duas trilhas estão disponíveis desde o início do experimento.

O cenário 2 diferencia-se do cenário 1 na quantidade de alimento disponível nas trilhas. Para este cenário, a quantidade de alimento é finita, sendo assim, os robôs esgotam uma trilha e depois passam a explorar a outra trilha.

No terceiro e quarto cenário a quantidade de alimento está disponível desde o início apenas em uma das trilhas (trilha B para o cenário 3 e trilha A para o cenário A). Com isso, os robôs esgotam a trilha com alimento disponível apenas após identificarem a trilha detentora do alimento.

As trilhas de alimento utilizadas pelas formigas são divididas em três principais situações:

- trilha em plena atividade: quando a maioria das formigas utilizam a trilha para a busca de alimento;
- trilha abandonada: quando as formigas deixam de utilizar uma trilha devido a escassez de alimento e
- trilha recém retomada: quando as formigas volta a utilizar uma trilha, o que acontece quando o alimento passa estar em abundância (efeito de sazonalidade).

Tabela 4.1 – Possíveis cenários alterando a quantidade de alimento na trilha, a quantidade coletada e a disponibilidade do alimento ao longo do tempo.

		Cenário 1	Cenário 2	Cenário 3	Cenário 4
Quantidade de alimento disponível na trilha	**A**	infinita	finita	finita	finita
	B	infinita	finita	finita	finita
Alimento disponível desde o início		em A e B	em A e B	apenas em B	apenas em A

A trilha B é cerca de três vezes maior do que a trilha A, ou seja, os robôs levam mais tempo para percorrer a trilha B e demoram mais para atualizar o feromônio. Com isso curva de subida da quantidade de feromônio tende a seguir uma reta com coeficiente angular menor. Por este motivo, a inclinação da curva de subida da quantidade de feromônio é um dos critérios de avaliação da interação robô/ambiente.

Outro critério de avaliação é o decaimento da quantidade de feromônio depois que uma trilha foi esgotada. Este decaimento indica o nível de persistência do feromônio no ambiente, isto é, a forma como o feromônio evapora. Se o decaimento for muito lento, o tempo para alterar a trilha indicada pode ser muito alto. Porém, se o decaimento for muito rápido, o sistema ficará oscilando entre as trilhas, deixando o sistema instável. Por isso foram analisadas duas situações. Na primeira situação, não há penalização quando um robô retorna para o ninho sem alimento (depois que a trilha já foi esgotada) e o decaimento da quantidade de feromônio mantem-se constante o tempo todo. Na segunda situação, sempre que o robô retorna sem alimento de uma trilha indicada, há uma penalização na quantidade de feromônio da trilha, dando um caráter mais dinâmico ao sistema.

A indicação da trilha também é um critério de avaliação, pois, é possível verificar se o feromônio está atualizando de forma correta. Outro critério de avaliação é o tempo necessário para esgotar a quantidade de alimento no sistema. Este critério está relacionado diretamente ao critério de persistência no ambiente. Nos casos em que ocorre a penalização, espera-se que os robôs rapidamente percebam que uma trilha foi esgotada e procurem alimento na outra trilha.

4.4. Testes

Experimento físico

Durante os testes foram verificados alguns problemas físicos na bancada e no robô móvel. Alguns desses problemas já foram expostos durante o texto, porém esta seção apresentará com mais detalhes cada situação e como foram resolvidas.

Conflito de leitura e escrita dos leitores. Conforme informado no item 3.3, **i**nicialmente o sistema foi projetado para utilizar a tecnologia NFC para comunicação entre os robôs e o ambiente. Ao realizar os testes de bancada com um robô ocorreram problemas de comunicação entre os agentes. O leitor PN532 para funcionar com comunicação NFC necessita que os leitores estejam alinhados e a uma distância específica. Como o robô não possui odometria era quase impossível garantir que o robô estivesse exatamente sobre um

dos leitores do ambiente. Por este motivo optou-se por utilizar as etiquetas como intermediação entre ambiente e robô.

Sendo assim, o robô foi projetado para que o leitor PN532 estivesse habilitado o tempo todo buscando a leitura das etiquetas. Quando o robô encontrava uma etiqueta, ele parava, realizava a leitura e esperava o novo comando (avançar ou salvar novamente na etiqueta). Porém, quando o leitor do robô e o leitor do ambiente, do feromônio ou do semáforo, estão sobre a etiqueta e habilitados ao mesmo tempo, ocorre um conflito de acesso à etiqueta, travando hora o código do robô, hora o código do ambiente. Com isso o experimento funcionava de forma correta durante poucas leituras (menos de um minuto), e era necessário reiniciar o sistema.

Mesmo quando um leitor (energizado) não está realizando a leitura ou escrita da etiqueta e outro leitor tenta acessar a mesma etiqueta há o conflito. Com isso, tanto o leitor do robô quanto os leitores do ambiente passaram a ser habilitados através de um módulo relé que é acionado apenas no exato momento de leitura e escrita da etiqueta. As Fig. 4.1, 4.4 e 4.5 apresentaram estes módulos na representação esquemática dos componentes de cada agente.

O tempo entre a habilitação de cada leitor do ambiente é de cinco segundo. O leitor é energizado, ocorre a leitura da etiqueta, se necessário, ocorre a gravação e então o módulo relé é desabilitado. Quando há apenas leitura, o módulo fica energizado por menos tempo do que quando há leitura e escrita.

Já para o robô móvel, o leitor é habilitado apenas quando o robô está sobre a etiqueta. Para identificar o ponto de leitura e escrito, foi acrescentado um sensor ultrassônico HC-SR04 na parte lateral direita do robô. Tal sensor, é responsável por verificar a distância entre o robô e a parede. Quatro identificadores foram inseridos na bancada para indicar a posição de cada etiqueta. Com isso o leitor do robô fica desabilitado durante grande parte do experimento, sendo habilitado apenas sobre as etiquetas, reduzindo consumo e aumentando a autonomia.

Não há sincronismo entre os leitores do ambiente e o leitor do robô. Porém, com os módulos relés utilizados a quantidade de conflitos de leitura caiu drasticamente.

Controle de velocidade e posição das rodas. Durante a etapa de projeto do robô móvel foi definido que não seriam utilizados motores com *encoder* para que o custo fosse o menor possível. Como o robô é um seguidor de faixa, o controle de velocidade das rodas realmente não se faz necessário. Porém no ponto de saída do ninho, há uma divisão em dois caminhos. Para seguir a trilha A é necessário que o robô vire à esquerda na saída do ninho, porém em alguns casos o robô recebia a informação de que deveria virar e não conseguia entrar na trilha. Existem dois motivos que explicam este fato:

1. A faixa foi delimitada por uma fita adesiva branca de PVC (policloreto de vinila) e em alguns momentos gerava uma resistência à movimentação dos robôs móveis.
2. A carga da bateria do robô caia ao longo do experimento, reduzindo a velocidade do motor e consequentemente, tornando o algoritmo de giro do robô cada vez menos eficiente.

Para minimizar este problema o trajeto foi alterado na saída do ninho com dois caminhos mais bem definidos do que quando havia o formato em ‘8’. Além disso, sempre que o robô estava visualmente mais lento, era retirado do experimento para troca das baterias.

Simulação

O primeiro desafio da implementação em ambiente de simulação está relacionado à importação do modelo 3D do robô para o V-REP. O primeiro modelo 3D do *Attabot* possuía muitos vértices, faces e arestas, tornando a simulação do comportamento dinâmico do robô sobrecarregada. Foi necessário retirar os detalhes do modelo 3D, tais como, as roscas das barras rosqueadas, os furos do Arduino, a fiação dos sensores, o modelo 3D do sensor de faixa, entre outros. Além disso o *software MeshLab* foi utilizado para eliminar sobreposições de vértices e faces do modelo final.

Outro ponto que deve ser mencionado é a dificuldade de utilizar variáveis globais no V-REP. Criou-se variáveis que eram utilizadas tanto no *script* do leitor do robô quanto no *script* do chassi do robô, porém ao executar a simulação os dois algoritmos não se comunicavam. Para resolver este problema foram utilizadas as funções *sim.setStringSignal* e *sim.getStringSignal*. A primeira função é utilizada no algoritmo do leitor do robô enquanto a segunda função é utilizada no algoritmo do comportamento dinâmico do robô.

CAPÍTULO V

RESULTADOS E DISCUSSÕES

Este capítulo apresenta os resultados coletados tanto para os experimentos na bancada quanto para a simulação no V-REP. Foram realizados experimentos em quatro cenários, definidos no capítulo IV (Tab. 4.1). Os gráficos são analisados de acordo com o comportamento do robô e do ambiente inteligente (feromônio).

Inicialmente são expostos os resultados referentes ao experimento na bancada física. Para estes experimentos foram utilizados três robôs móveis homogêneos, sendo que a todo tempo havia pelo menos dois robôs móveis na bancada. O percurso da trilha A é pequeno e a utilização de três robôs o tempo todo gerava uma fila de robôs entre a saída do ninho e o semáforo da trilha. Por isso, em alguns momentos um dos robôs era retirado da bancada. Sempre que o robô diminuía a velocidade de movimentação ele era retirado da bancada para troca das baterias.

Tanto para o experimento em bancada quanto para o simulador foram coletados os dados do feromônio em cada trilha e a quantidade de alimento na fonte ao longo do tempo. Com esses dados é possível realizar uma validação computacional do feromônio através do *software* $MATLAB^{®}$.

Como no primeiro cenário a quantidade de alimento das trilhas é infinito, os resultados apresentados são curvas de *quantidade de feromônio x tempo* e *trilha indicada x tempo*. Para os cenários seguintes a curva *quantidade de alimento na fonte x tempo* também é apresentada.

Por fim, os comportamentos dos agentes tanto para o experimento quanto para o ambiente simulado são discutidos.

5.1. Resultados experimentais

Os ensaios experimentais possuem duração entre meia hora e quatro horas e meia, dependendo do cenário avaliado.

5.1.1. Cenário 1

Este cenário compreende o caso em que a quantidade de comida disponível nas duas trilhas é infinita ao longo do tempo. Sendo assim, espera-se que a partir do momento que uma trilha seja indicada como a detentora do alimento, todos os robôs passem a seguir este caminho. As Fig. 5.1, Fig. 5.2 e Fig. 5.3 exibem os resultados de três experimentos em que nos dois primeiros a trilha A foi a indicada e no último, a trilha indicada é a trilha B. Observa-se que no início não há indicação de trilha pois a quantidade de feromônio inicial era igual para as duas trilhas. Sendo assim, a tomada de decisão é realizada pelo próprio robô móvel de forma randômica. Quando a quantidade de feromônio em uma trilha é maior do que de 52 % do total de feromônio (A e B) a trilha é indicada e o robô passa a seguir apenas este caminho.

Nas Fig. 5.1 e 5.2 (caso em que a trilha A foi indicada) nota-se que em alguns momentos o robô recebeu a indicação da trilha A, porém seguiu o caminho da trilha B devido aos motivos explicados no final do capítulo anterior (faixa de PVC e bateria fraca).

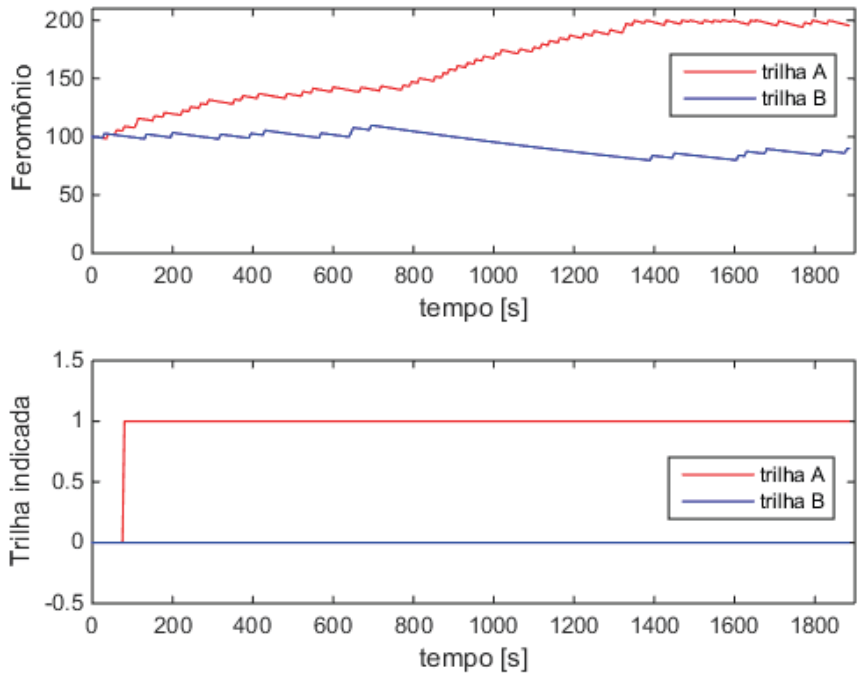


Figura 5.1 – Quantidade de feromônio e a trilha indicada para o primeiro experimento no cenário 1.

Este cenário permite verificar que o tempo para o feromônio de uma trilha atingir a saturação é maior para a trilha B. Este fenômeno deve-se ao fato do caminho da trilha B ser mais longo que o caminho da trilha A, com isso o robô demora mais tempo para atualizar a quantidade de feromônio.

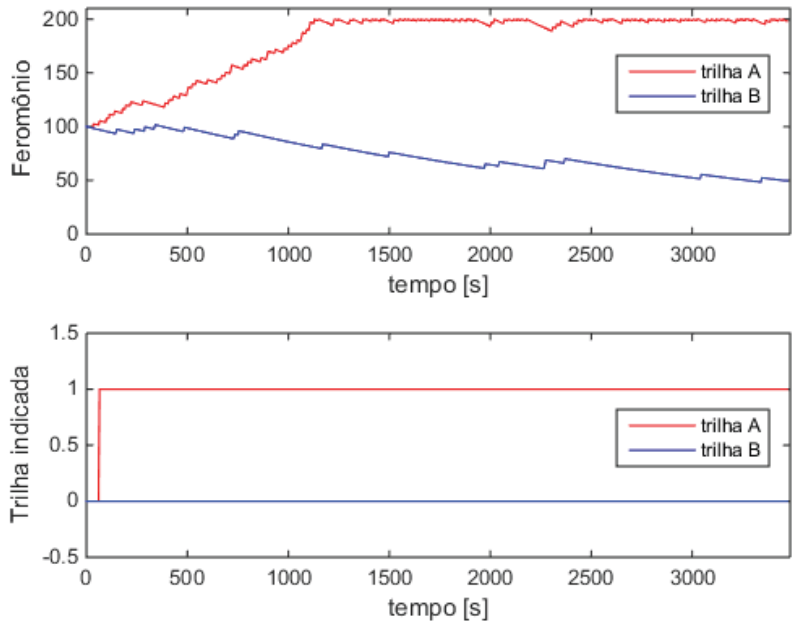


Figura 5.2 – Quantidade de feromônio e a trilha indicada para o segundo experimento no cenário 1.

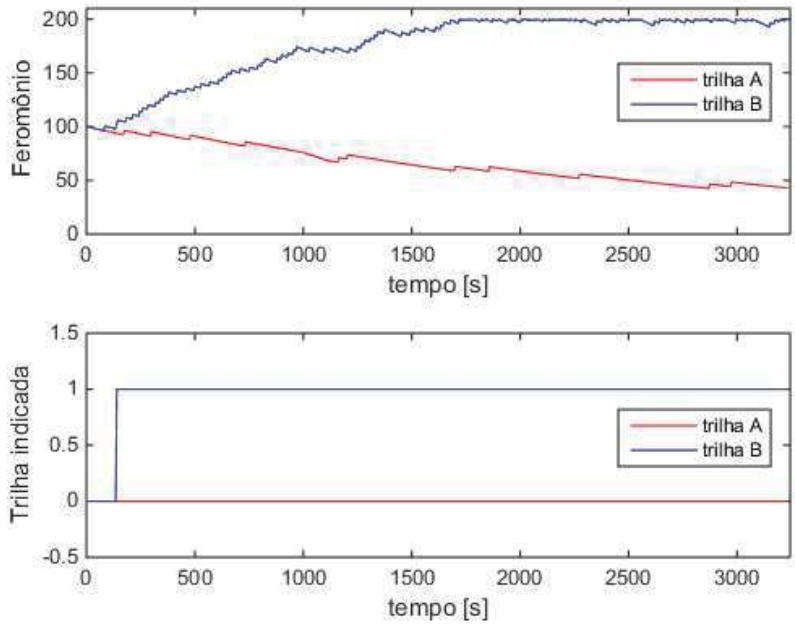


Figura 5.3 – Quantidade de feromônio e a trilha indicada para o terceiro experimento no cenário 1.

Quando a trilha A é indicada, o tempo para saturação da quantidade de feromônio é por volta de 1300 segundos. Quando a trilha B é indicada, este tempo de saturação é cerca de 20 % maior.

5.1.2. Cenário 2

O cenário 2 possui quantidade de alimento finita nas duas trilhas. Com isso, é possível verificar como os robôs se comportam quando uma fonte de alimento é esgotada. A Fig. 5.4 apresenta os resultados do primeiro experimento para este cenário.

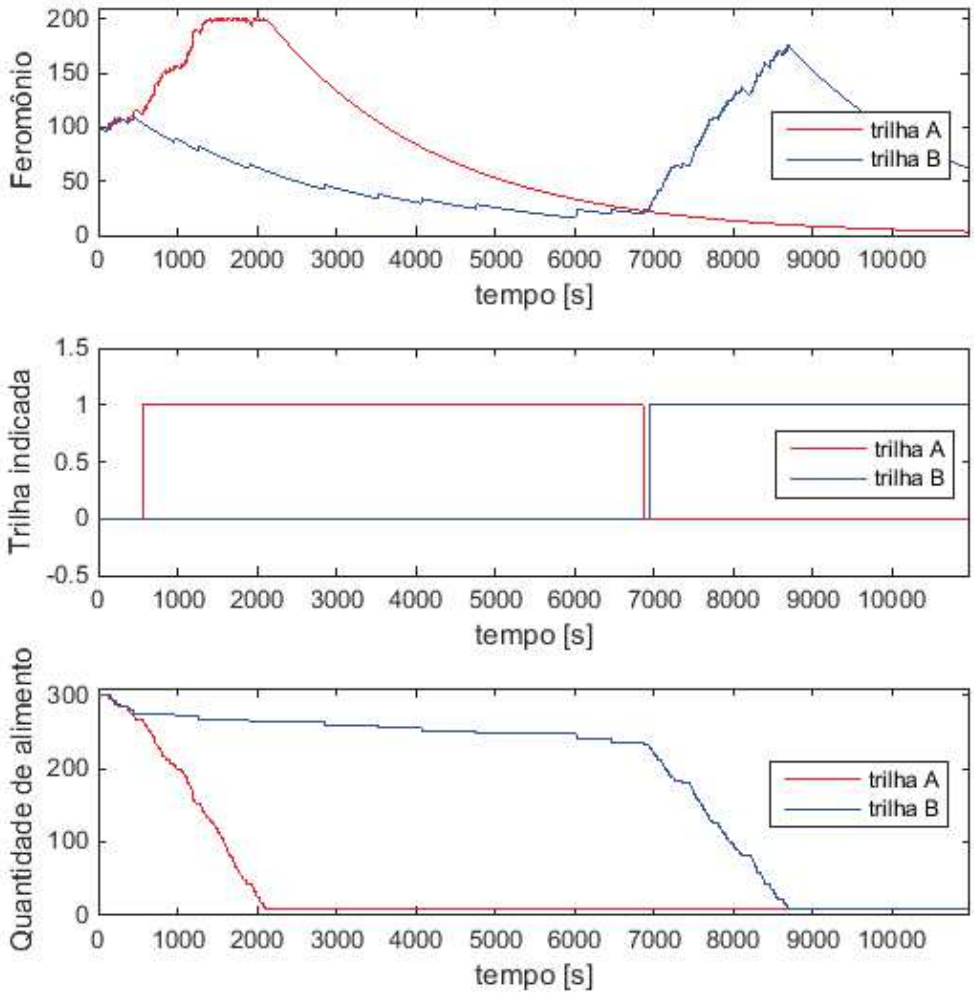


Figura 5.4 – Quantidade de feromônio, trilha indicada e quantidade de alimento para o primeiro experimento no cenário 2.

A quantidade de alimento na trilha A acabou com pouco menos de 4000 segundos, porém a trilha B só foi indicada após 8000 segundos. Isso ocorre, pois, depois que o alimento acaba na trilha A, os robôs continuam por este caminho e retornam sem alimento. Assim, a concentração de feromônio na trilha A decai conforme a exponencial decrescente e apenas após igualar a concentração de feromônio da trilha B que o sistema passa a não ter indicação de caminho a ser seguido.

A fim de reduzir esse longo período de busca de uma trilha sem alimento, o feromônio foi alterado de modo que, quando um robô retorna de um local sem alimento, ocorre uma penalização na concentração de feromônio da trilha. A Fig. 5.5 apresenta os resultados para este tipo de cenário. Percebe-se que, quando esgota o alimento na trilha A a concentração de feromônio da trilha decai significativamente.

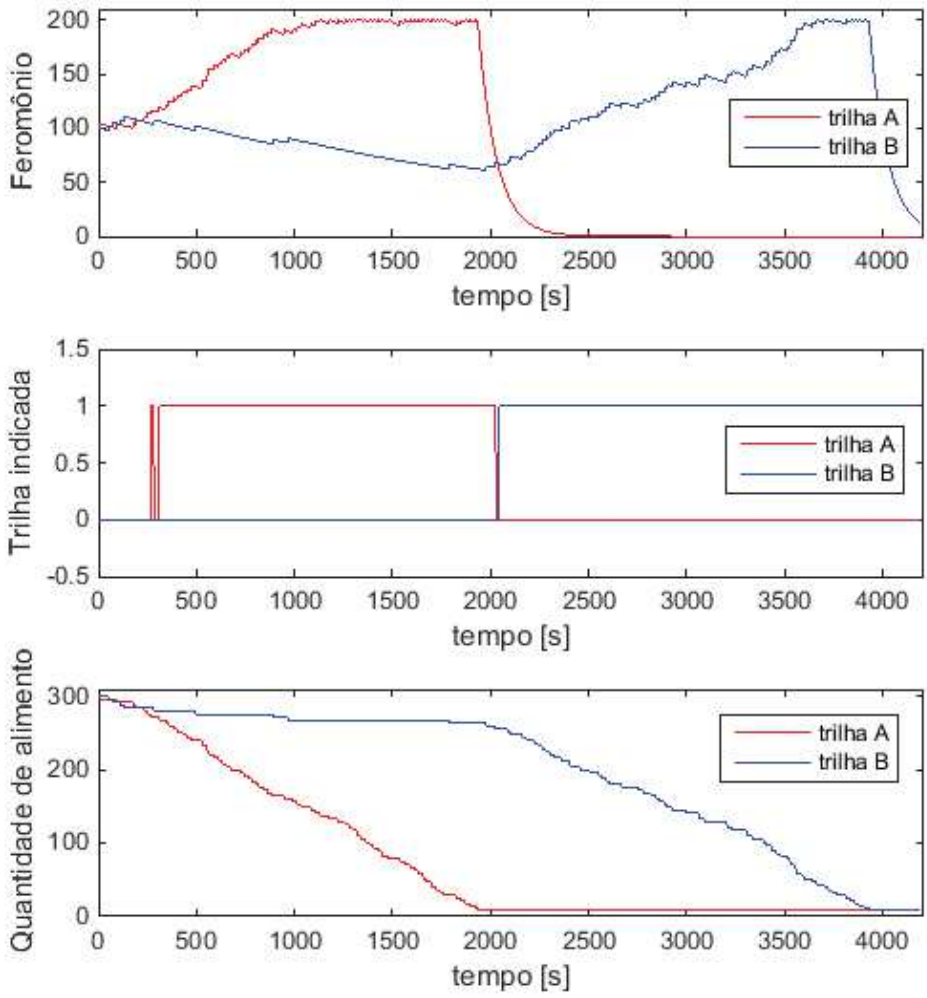


Figura 5.5 – Quantidade de feromônio, trilha indicada e quantidade de alimento para o segundo experimento no cenário 2.

O tempo total para esgotar as duas trilhas caiu de 11000 segundos para 8000 segundos. A curva de queda da quantidade de feromônio da trilha A decai rapidamente e após três robôs retornarem da trilha sem alimento a trilha B passou a ser indicada. Com isso a interação robô/feromônio se torna mais dinâmica e eficiente. Diante deste comportamento favorável, os cenários 3 e 4 foram avaliados com a mesma penalização.

5.1.3. Cenário 3

A Fig. 5.6 apresenta os resultados para o experimento em que o alimento está disponível, inicialmente, apenas em B. Após 1500 segundos o alimento na trilha A está disponível.

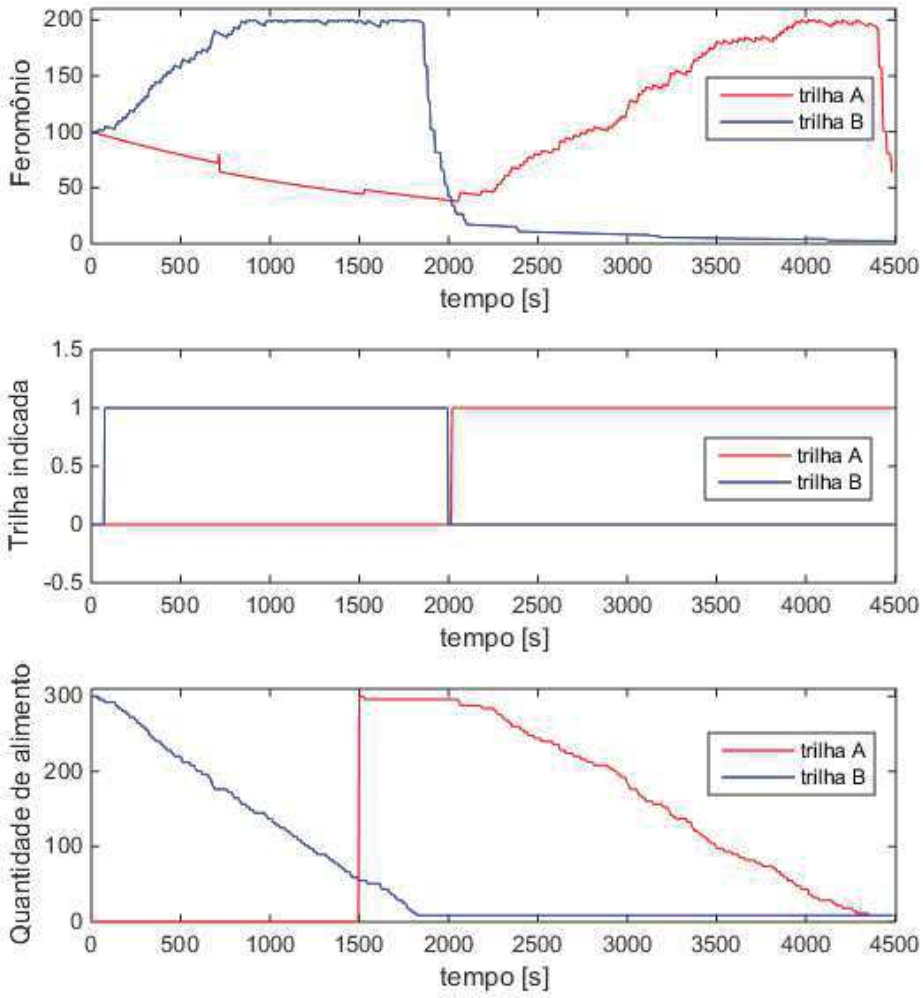


Figura 5.6 – Quantidade de feromônio, trilha indicada e quantidade de alimento para o experimento no cenário 3.

5.1.4. Cenário 4

No último cenário, o alimento está disponível, inicialmente, apenas na trilha A. A Fig. 5.7 apresenta os resultados para este cenário. Nota-se que no início, quando um robô retornou da trilha B, houve uma penalização, pois, não havia alimento naquela trilha, sendo assim, a indicação da trilha A se torna mais rápida e os robôs passam a seguir apenas este caminho.

Observa-se que o tempo para esgotar as duas trilhas tanto no cenário 3 quanto no cenário 4 são próximas (por volta de 4300 segundos)

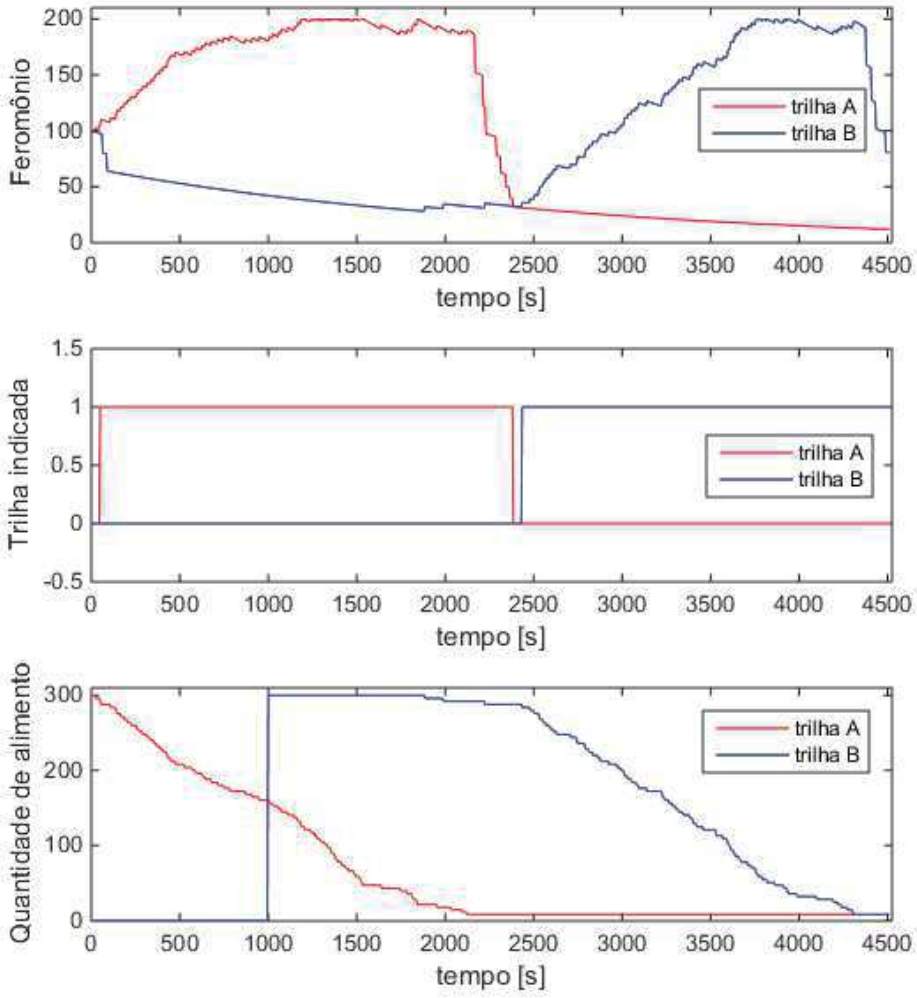


Figura 5.7 – Quantidade de feromônio, trilha indicada e quantidade de alimento para o experimento no cenário 4.

5.2. Resultados das simulações

Os resultados do ambiente de simulação no V-REP servem para validar o comportamento mensurado nos experimentos reais. Caso o feromônio se comporte da mesma forma na simulação e no real é possível inferir resultados reais para enxame de robôs avaliados no simulador.

5.2.1. Cenário 1

Os resultados do primeiro cenário em ambiente simulado são apresentados nas Fig. 5.8 – 5.10, sendo as duas primeiras para o caso em que a trilha A foi a trilha indicada. Assim como no experimento real, o tempo de subida da quantidade de feromônio na trilha B é maior do que na trilha A, pelo fato do caminho ser mais longo.

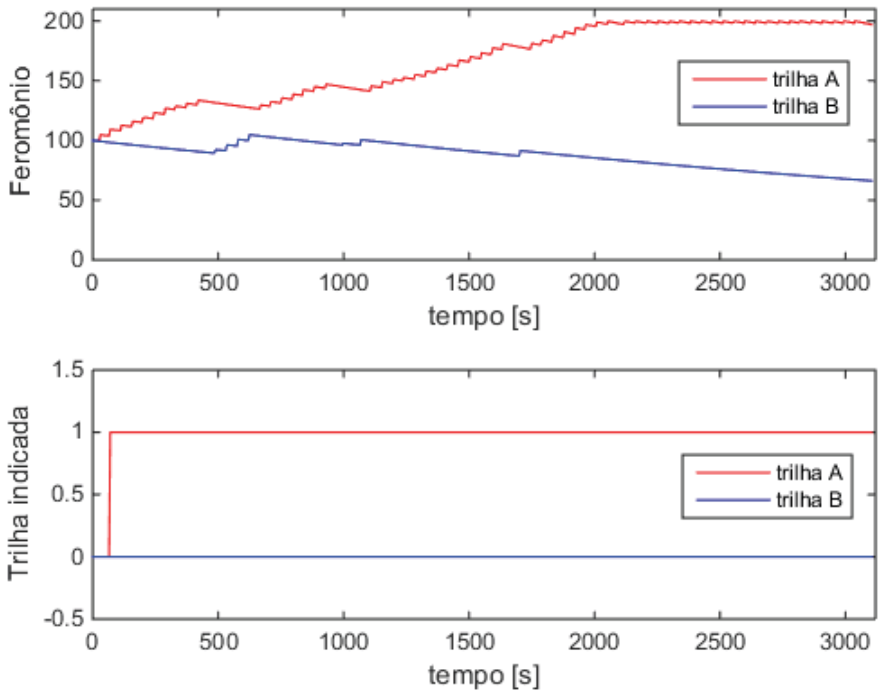


Figura 5.8 – Quantidade de feromônio e a trilha indicada para a primeira simulação no cenário 1.

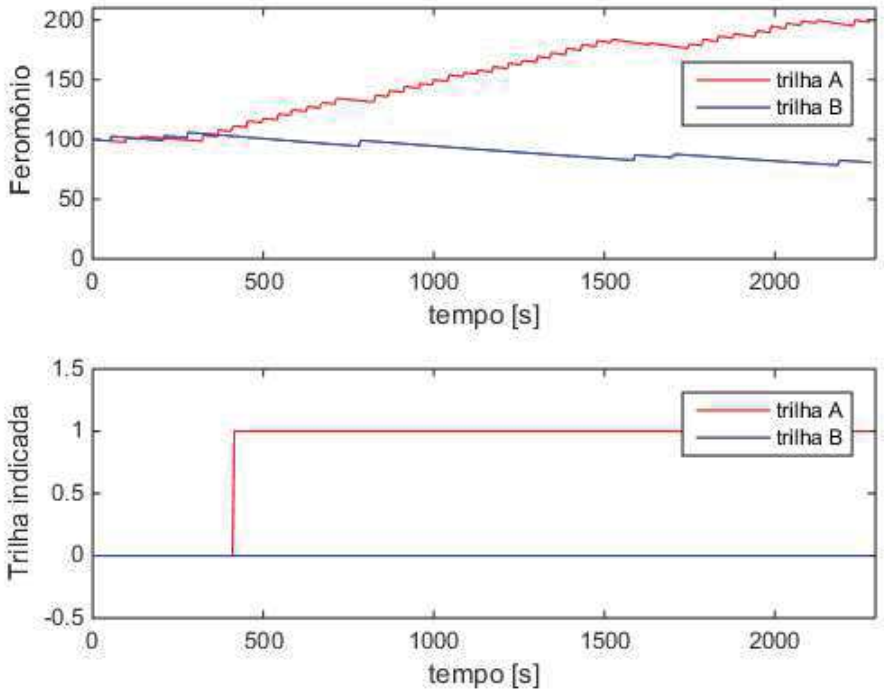


Figura 5.9 – Quantidade de feromônio e a trilha indicada para a segunda simulação no cenário 1.

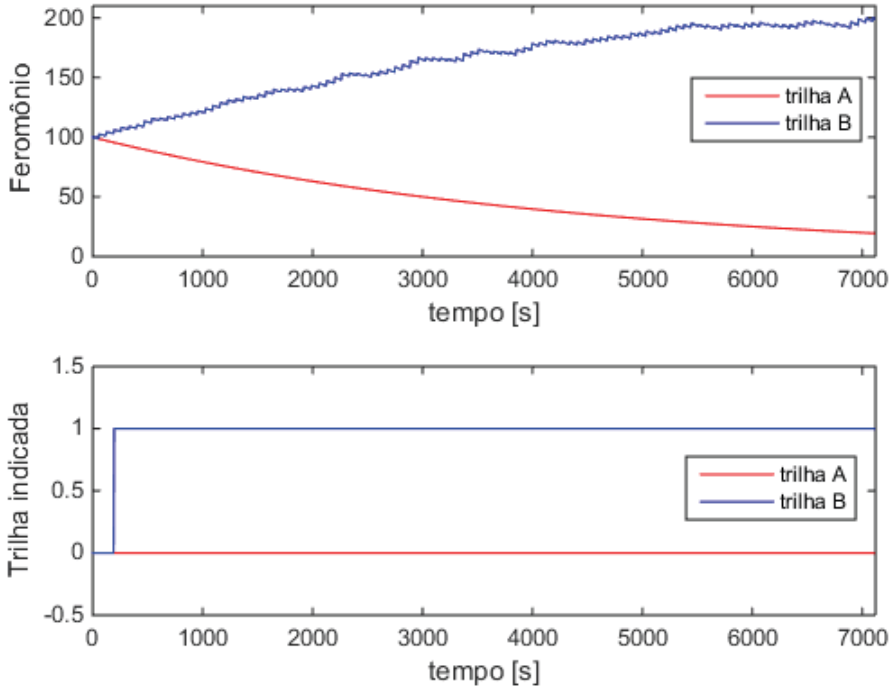


Figura 5.10 – Quantidade de feromônio e a trilha indicada para a terceira simulação no cenário 1.

5.2.2. Cenário 2

O cenário 2 foi simulado da mesma forma com que foram realizados os experimentos. No primeiro caso, o decaimento da quantidade de feromônio quando o robô retorna de uma trilha sem alimentos mantém-se o mesmo.

No segundo caso há uma penalização na quantidade de feromônio da trilha sempre que o robô retorna sem alimento. A Fig. 5.11 ilustra o primeiro caso, enquanto a Fig. 5.12 o segundo caso. Nota-se que o tempo total para esgotar as duas trilhas caiu de 12 000 segundos para 8 000 segundos, deixando clara a necessidade de se utilizar uma abordagem híbrida na emulação do feromônio.

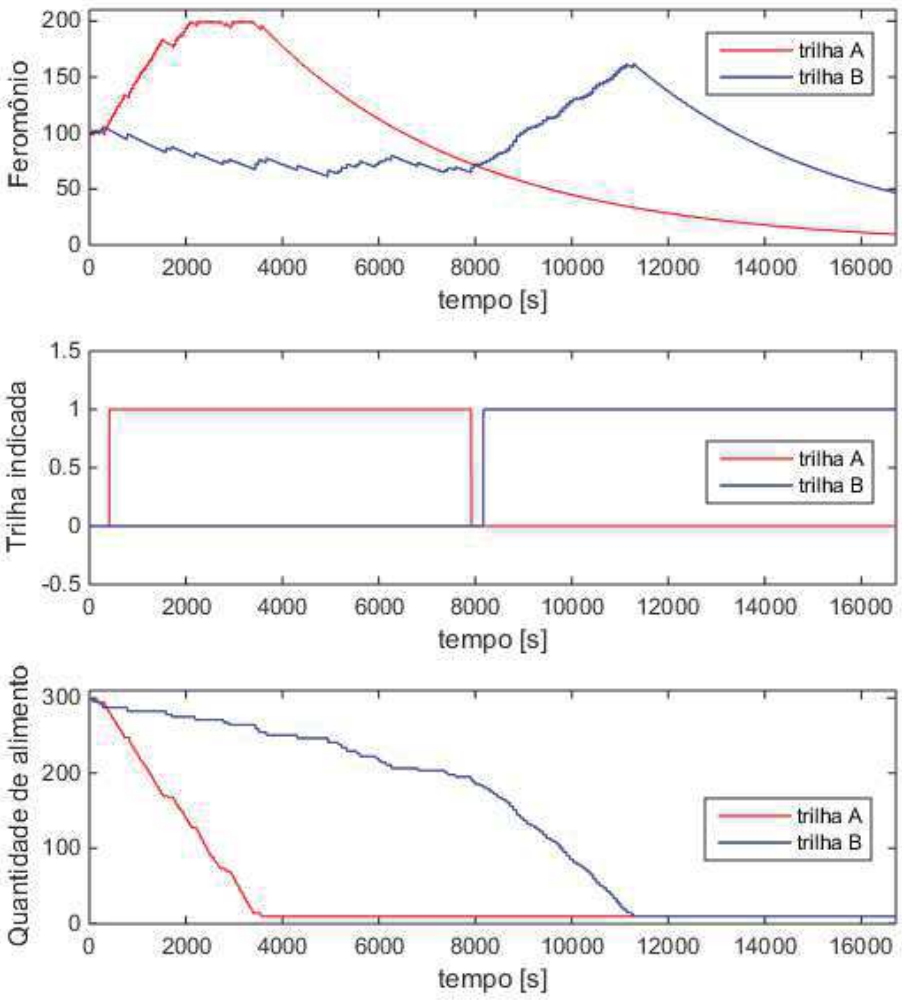


Figura 5.11 – Quantidade de feromônio, trilha indicada e quantidade de alimento para a primeira simulação no cenário 2.

5.2.3. Cenário 3

A Fig. 5.13 apresenta o resultado para o cenário 3. Percebe-se no começo do experimento que quando o robô retorna pela trilha A sem alimento, a quantidade de feromônio desta trilha cai significativamente fazendo com que a trilha B seja indicada rapidamente. Tão logo a quantidade de alimento na trilha B acaba, os robôs que retornam por este caminho penalizam a quantidade de feromônio em B fazendo com que a trilha A seja indicada.

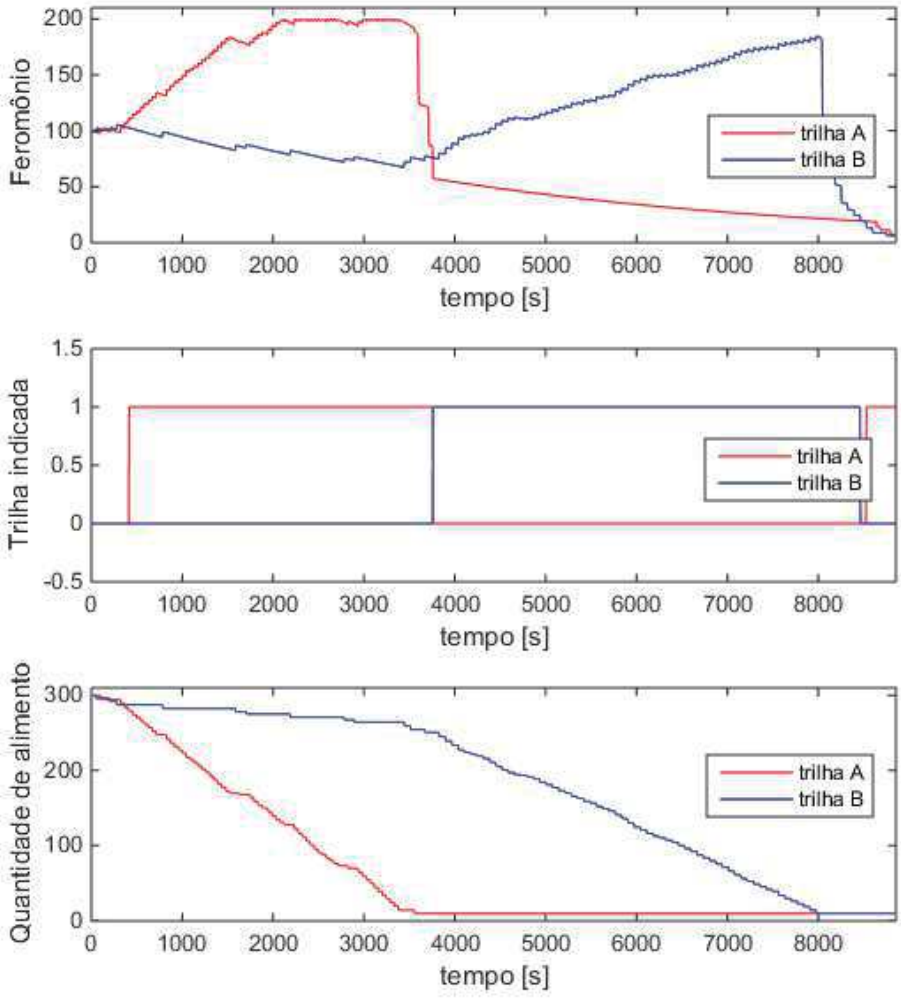


Figura 5.12 – Quantidade de feromônio, trilha indicada e quantidade de alimento para a segunda simulação no cenário 2.

5.2.4. Cenário 4

O último experimento simulado foi do cenário 4 e está apresentado na Fig. 5.14. Neste cenário, a quantidade de alimento na trilha B surge apenas depois de um tempo de simulação, fazendo com que a trilha A seja indicada primeiro.

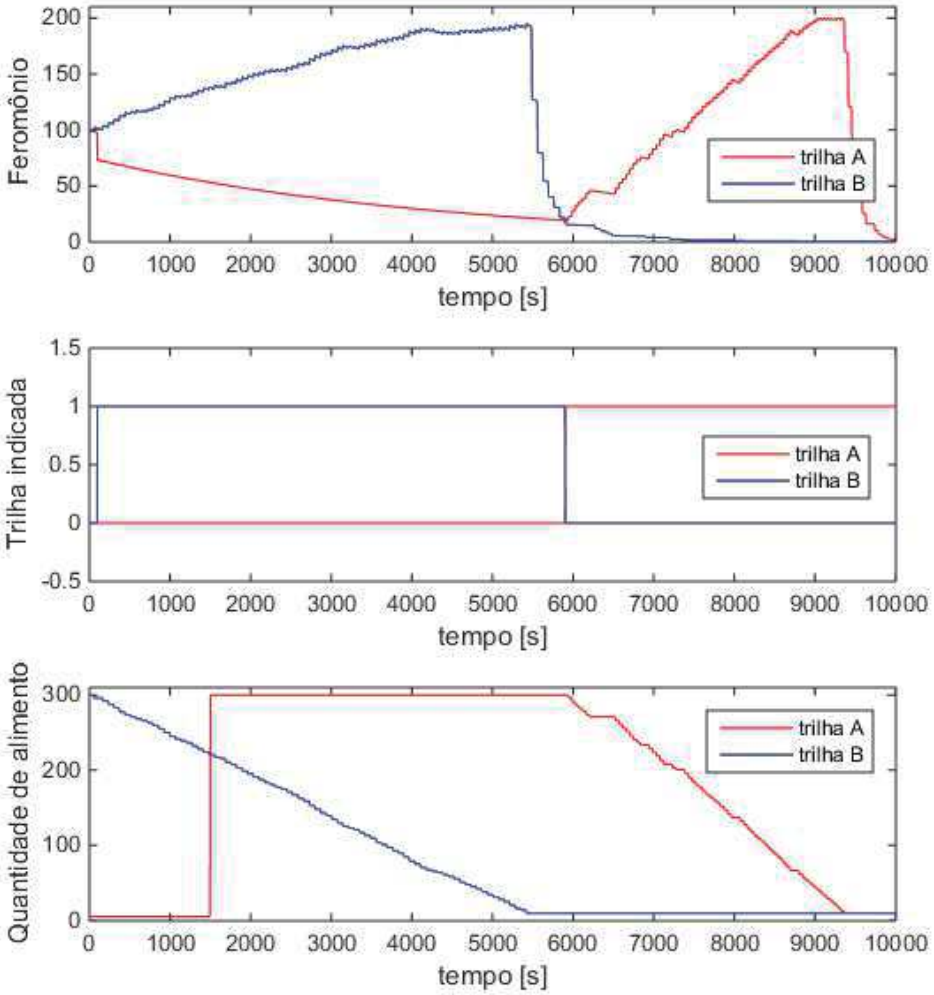


Figura 5.13 – Quantidade de feromônio, trilha indicada e quantidade de alimento para a simulação no cenário 3.

5.3. Discussões

Brambilla *et al.* (2013) definiram o comportamento dos animais sociais como robusto, escalável e flexível. Os robôs móveis possuem a capacidade de lidar com a perda de indivíduos, pois, quando um robô é retirado do experimento, os outros robôs não são

afetados uma vez que a comunicação entre os robôs ocorre através das etiquetas do feromônio e do semáforo. Um robô inserido no ninho em um momento qualquer também não afeta o comportamento dos outros robôs. Logo podemos dizer que os robôs são robustos.

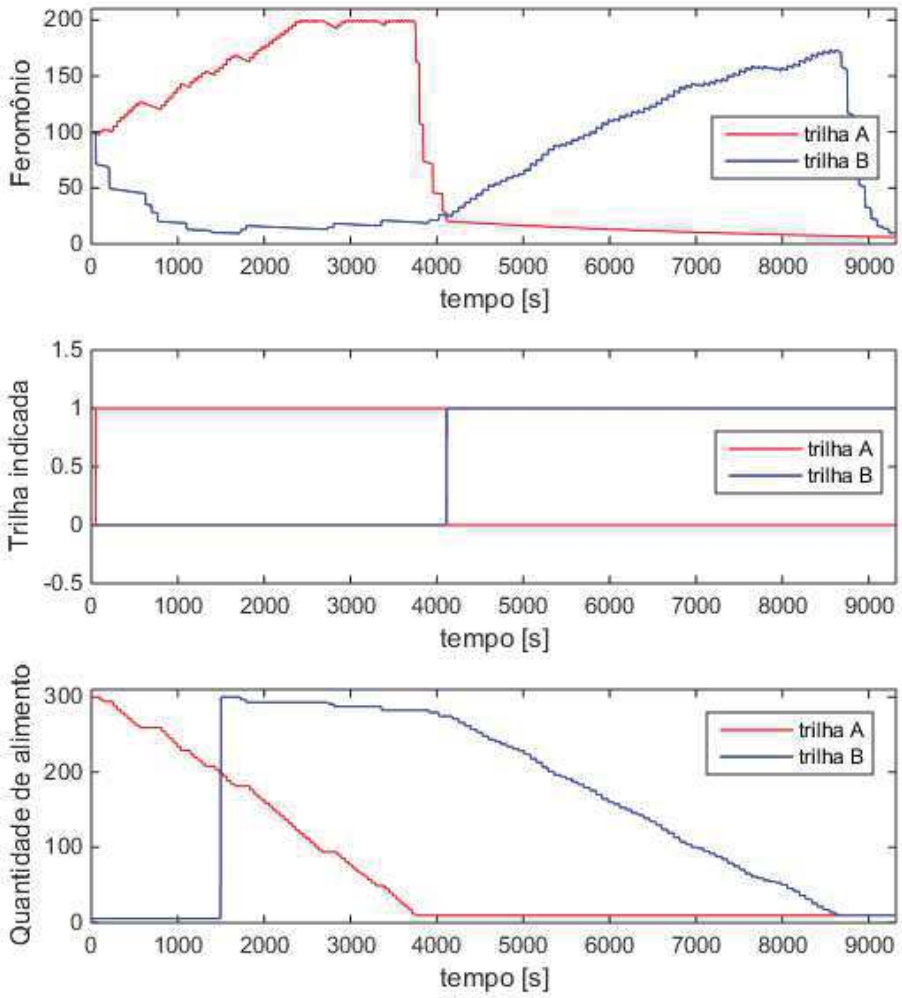


Figura 5.14 – Quantidade de feromônio, trilha indicada e quantidade de alimento para a simulação no cenário 4.

O comportamento dos robôs (trilha seguida e atualização da etiqueta do feromônio) dentro do ambiente experimental e simulado não depende da quantidade de robôs. Nos testes experimentais foram utilizados três robôs, enquanto os ensaios no V-REP utilizaram apenas um robô. Para os dois casos, o comportamento dos robôs foi semelhante, assim como a forma com que o feromônio evoluiu ao longo do tempo. Esta característica indica que os robôs móveis são escaláveis.

Flexibilidade está relacionada a capacidade dos robôs lidarem com um amplo espectro de diferentes ambientes e/ou tarefas. Para os diferentes cenários, tanto nos testes experimentais quanto na simulação, os robôs foram capazes de realizar a tarefa de busca de alimento (sempre que disponível na trilha). Com isso, os robôs móveis também são considerados flexíveis. Sendo assim, pode-se afirmas que os robôs móveis modelados na tese simulam animais sociais.

De acordo com Kim e Kurabayashi (2011), o feromônio de formiga é volátil, pois, possui duas características específicas: difusão e evaporação. A volatilidade é verificada quando uma trilha é desfeita. No feromônio emulado, esta situação ocorre quando uma das trilhas é indicada como a detentora do alimento.

A difusão do feromônio emulado é caracterizada pela informação salva na *Tag Feromônio 2* que indica o caminho a ser seguido pelo robô. Isto é, o sinal 'químico' é difundido para todos os agentes móveis que passam pelo local.

Já a evaporação é emulada pela exponencial decrescente. A persistência no ambiente é uma das características importantes do feromônio de formiga, pois, ela indica durante quanto tempo o feromônio está presente na trilha em questão. Vale ressaltar que, foram analisadas duas situações de persistência no ambiente (caso sem penalização e caso com penalização). No caso em que não há penalização da quantidade de feromônio da trilha quando os robôs que retornam sem alimento, a quantidade de feromônio persiste durante um grande período, tornando o sistema menos dinâmico. Já no caso em que os robôs que retornam sem alimento há uma penalização na quantidade de feromônio, faz com que a trilha indicada seja alterada de forma mais rápida (quando há alimento na outra trilha).

A abordagem da equação do feromônio por uma exponencial decrescente mais a penalização quando há falta de alimento é a solução mais viável para os casos avaliados na tese.

A utilização da tecnologia RFID como solução do problema de tráfegos urbanos é discutida no ambiente acadêmico: Sharma *et al.* (2013), Ali e Hussein (2017), Choosri *et al.* (2015), entre outros. A heterogeneidade dos veículos, a concorrência de serviços, o tempo de resposta e o volume de dados, por exemplo, são pontos a serem considerados.

O semáforo utilizando tecnologia RFID e modelagem em rede de Petri consegue trabalhar alguns dos pontos mencionados. Apesar de os robôs móveis serem *cmp-ident* (Dudek *et al.*, 1996), o posicionamento dos leitores e das etiquetas espalhadas pelo ambiente tornam o sistema aplicável a qualquer conjunto de veículos. A concorrência entre os veículos é resolvida durante modelagem em rede de Petri, podendo ser utilizada para veículos de emergência, tais como, carros de polícia, bombeiro e ambulâncias. A utilização de arco inibidores é suficiente para resolver problemas de concorrência.

Já o volume de dados para o semáforo com iPNRD é muito menor quando comparado ao semáforo com RFID desenvolvido por Choosri *et al.* (2015), pois, o segundo necessita de um sistema de banco de dados para funcionar. O semáforo da tese, por utilizar informações locais e consumir estas informações nos disparos da rede de Petri, não necessita de um grande volume de dados.

O problema de tempo de resposta é comum para qualquer tipo de sistema utilizando tecnologia RFID, visto que, existe um tempo de leitura e escrita dos dispositivos de comunicação. Sendo assim, só é resolvido com a utilização de dispositivos com maiores taxas de leitura, além de leituras múltiplas.

Ao observar o comportamento do robô móvel durante os testes simulados, percebe-se por algumas curvas (Fig. 5.14, por exemplo) que o robô apesar de receber a indicação da trilha A, seguiu a trilha B. Isto ocorre, pois, o leitor e o comportamento dinâmico do robô executam em *scripts* distintos. Sendo assim, é necessário que o *script* do leitor do robô envie uma mensagem através da função *sim.setStringSignal* para o *script* do comportamento dinâmico do robô. Este recebe a mensagem através da função *sim.getStringSignal*. Em alguns casos, apesar de o *script* do leitor do robô indicar a trilha de forma correta, esta mensagem não chegou ao *script* do comportamento dinâmico. Assim, o robô mantinha o sentido de direção de antes da leitura da etiqueta, ou seja, andava em linha reta.

CAPÍTULO VI

CONCLUSÕES E PROPOSTAS DE TRABALHOS FUTUROS

Esta tese apresentou a concepção de um ambiente inteligente que integra tecnologia iPNRD entre leitores para controle autônomo e distribuído e desenvolvimento de um método/metodologia de desenvolvimento de projeto de robôs para enxame de robôs. O objetivo principal, de emular o feromônio de formiga para realizar o controle comportamental de robôs móveis baseado em iPNRD foi alcançado, pois, projetou-se e construiu-se um robô móvel; modelou-se o feromônio, o robô móvel e o semáforo através da iPNRD; implementou-se em dispositivos embarcados baseados no Arduino Mega 2560; desenvolveu-se computacionalmente um simulador para o ambiente em V-REP; definiram-se cenários para ensaios; realizaram-se os testes e discutiram-se os resultados.

A aplicação da tecnologia RFID para emulação do feromônio de formiga é pouco abordada na literatura. Alguns trabalhos utilizam outras metodologias para emular o feromônio, tais como, álcool, tinta fosforescente, calor, entre outros, porém cada abordagem possui desvantagens. Não há indícios de trabalhos em que o comportamento real de um ambiente inteligente com robôs móveis e o comportamento em um ambiente simulado são confrontados.

Com a iPNRD foi possível emular o feromônio de formiga tanto em ambiente de simulação quanto em experimento real. A utilização da modelagem do feromônio em rede de Petri possibilitou o controle do agente do ambiente (feromônio) que alterou o comportamento dos agentes móveis, tornando o sistema autônomo.

A interação robô-ambiente descentraliza a lógica de controle do sistema dando mais autonomia e robustez ao ambiente, representado pelo feromônio. Com isso a tomada de decisão não é mais função do robô e sim do agente do ambiente. Esta abordagem, presente na tese, é uma das formas de tratar as tomadas de decisões em ambientes inteligentes. Com o avanço das tecnologias que utilizam internet das coisas fica cada vez mais evidente a necessidade de descentralização das lógicas de controle em ambientes dinâmicos.

A interação robô-robô no âmbito do comportamento diante da tarefa de coleta de alimento em trilhas passa, obrigatoriamente, pelo ambiente, isto é, a interação é do tipo robô-ambiente-robô. Tal interação cria um sistema em que todos os agentes (internos e externos) são participativos e integrados, quebrando o paradigma do controle centralizado e hierárquico, e desenvolvendo um novo paradigma descentralizado, distribuído e integrativo.

A única forma de interação robô-robô está no controle reativo para evitar a colisão entre os agentes móveis. Vale ressaltar que a abordagem proposta não utiliza odometria, nem orientação global.

Outra contribuição da tese está ligada ao próprio robô móvel. O *Attabot* é um projeto *open source* de baixo custo (aproximadamente R$ 120,00) que pode ser utilizado no desenvolvimento de projetos de robótica móvel, bem como em projetos de programação básica devido à sua característica *do it yourself*. O único dispositivo específico utilizado no robô é o leitor PN532 que pode ser retirado sem alterar suas características comportamentais básicas (movimentação e prevenção de colisão).

As funcionalidades básicas de robôs móveis, como navegação, localização e mapeamento não foram desenvolvidas, o que pode aumentar o custo computacional dos robôs. Estas funcionalidades melhorariam a movimentação do robô nos experimentos reais, ao custo da troca do *hardware* e implementação do sistema de visão.

O leitor PN532 possui problemas de conflito de leitura e escrita, e mesmo com a utilização dos módulos relé para habilitar o leitor em momentos distintos, ocorreram alguns conflitos que ocasionaram no travamento do algoritmo do semáforo. Toda vez que o sistema parava era necessário reinicializar o controlador. Em poucos momentos, o robô móvel ou o feromônio travaram. Para aplicar o semáforo em ambiente real é necessário alterar o tipo de leitor (trocar por um leitor UHF), pois, será possível realizar mais de uma leitura ao mesmo tempo, reduzindo a probabilidade de travamento do algoritmo.

As modelagens em redes de Petri dos agentes tornam o sistema mais robusto, porém algumas exceções não foram consideradas durante a modelagem. Uma situação exemplo está na modelagem do semáforo. Imagine que um robô chegue em uma trilha e realize o procedimento de passagem por aquele semáforo. Quando o robô recebe a informação de que o semáforo está verde, ele deve informar ao semáforo que sairá ($PS6$ ou $PS10$). Caso ocorra uma falha de escrita na etiqueta e robô mesmo assim saia do semáforo, a rede de Petri não será disparada e o semáforo permanecerá verde indefinidamente. Este problema ocorreu durante a realização dos testes e foi resolvido apenas no algoritmo do semáforo. Caso o semáforo não receba a informação de saída do robô em durante dois ciclos de leitura das etiquetas, é considerado que o robô já saiu e que a gravação da etiqueta teve algum problema. Com isso, a transição é disparada e o semáforo volta ao seu

funcionamento normal. Portanto, é necessária uma análise de todas as redes de Petri em busca de possíveis exceções.

6.1. Propostas de Trabalhos Futuros

Existem diversos trabalhos que podem ser desenvolvidos a partir do desenvolvimento da tese. Dentre eles, destacam-se:

- Implementação da fonte de alimento em etiquetas de RF;
- O aperfeiçoamento dos modelos comportamentais dos agentes em redes de Petri;
- A redução das simplificações adotadas e aperfeiçoamento dos modelos de simulação em V-REP;
- O aprimoramento de *hardware* e *software* com o objetivo de atender às funcionalidades básicas de robôs móveis, como navegação, comunicação, localização e mapeamento;
- A avaliação de cenários de maior complexidade: ambiente mais amplo com maior número de trilhas e maior número de robôs;
- Avaliação de cenários em que a quantidade de alimento é cíclica, emulando as quatro estações do ano.
- A investigação das restrições de tempo real da arquitetura de controle proposta;
- A investigação da solução para *hardwares* RFID de melhor qualidade;
- A expansão da solução baseada na iPNRD para outros tipos de aplicações da robótica móvel.
- A investigação da aplicação das outras variantes da iPNRD.

REFERÊNCIAS BIBLIOGRÁFICAS

ALI, A.; HUSSEIN, H.A. Traffic Lights System Based on RFID for Autonomous Driving Vehicle. In: ANNUAL CONFERENCE ON NEW TRENDS IN INFORMATION & COMMUNICATIONS TECHNOLOGY APPLICATIONS (NTICT). Baghdad, março 7-9. 2017. doi: 10.1109/NTICT.2017.7976149

AZAMBUJA, M. C. **Modelos e técnicas para simulação de sistemas UHF de identificação por rádio frequência (RFID).** 2011, 137p. Tese de Doutorado. Pontifícia Universidade Católica do Rio Grande do Sul.

BRAMBILLA, M.; FERRANTE, E.; BIRATTARI, M.; DORIGO, M. Swarm robotics: a review from the swarm engineering perspective. **Swarm Intelligence.** v. 7-1. p. 1-41. 2013. doi:10.1007/s11721-012-0075-2

BRUTSCHY, A.; SCHEIDLER, A.; FERRANTE, E.; DORIGO, M.; BIRATTARI, M. Can ants inspire robots? Self-organized decision making in robotic swarms. In: IEEE INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. Outubro 7-12, Portugal. 2012.

CHOOSRI, N.; PARK, Y.; GRUDPAN, S.; CHUARJEDTON, P.; ONGVISESPHAIBOON, A. IoT-RFID Testbed for Supporting Traffic Light Control. **International Journal of Information and Electronics Engineering.** v. 5-2. 2015. doi: 10.7763/IJIEE.2015.V5.511

COPPELIA ROBOTICS. Virtual Robot Experimentation Platform User Manual. In: COPPELIA ROBOTICS, GMBH. Manual do usuário. 2018.

COSKUN, V.; OK, K. and OZDENIZCI, B. **Near Field Communication (NFC): From Theory to Practice**. Chichester: John Wiley & Sons, 2011. 632 p.

COYLE, S.; MAJIDI, C.; LEDUC, P.; HSIA, K.J. Bio-inspired soft robotics: Material selection, actuation, and design. **Extreme Mechanics Letters.** v. 22. 2018. doi: 10.1016/j.eml.2018.05.003

DAVID, R.; ALLA, H. **Discrete, Continuous, and Hybrid Petri Nets.** Springer-Verlag Berlin Heidelberg. 2ed. 2010. 550p.

DEL FOYO, P. M. G.; SILVA, J. R. Towards a unified view of Petri nets and object oriented modeling. **ABCM Symposium Series in Mechatronics.** Rio de Janeiro, v. 1, p. 518-524, set. 2004.

DESEL, J.; REISIG, W. Place/transition Petri Nets. In: Reisig W., Rozenberg G. (eds) **Lectures on Petri Nets I**: Basic Models. ACPN 1996. **Lecture Notes in Computer Science**, v. 1491. Berlin, Heidelberg: Springer, 1998, p. 122-173.

DOTOLI, M.; FANTI, M.P. An urban traffic network model via coloured timed Petri nets. **Control Engineering Practice.** v. 14. 2006. doi: 10.1016/j.conengprac.2006.02.005

DUDEK, G.; JENKINS, M. R. M.; MILIOS, E.; WILKES, D. A Taxonomy for Multi-Agent Robotics. **Autonomous Robots.** p. 375-397. 1996.

ELECHOUSE. **PN532 NFC RFID Module User Guide**. Elechouse product info, nov. 2013. Guia do usuário.

FABREGAS, E.; FARIAS, G.;·DORMIDO-CANTO, S.;·DORMINDO, S. RFCSIM Simulador Interactivo de Robótica Móvil para Control de Formación con Evitación de Obstáculos. In: CONGRESO LATINOAMERICANO DE CONTROL AUTOMÁTICO, Cancún, p. 1392–1397. 2014.

FERREIRA, M.V.M.; FONSECA, J.P.S.; TAVARES, J.J.P.S.Z. Attabot: Open Platform Inspired on Brazilian Ants for Swarm Robots. In: 2018 LATIN AMERICAN ROBOTIC SYMPOSIUM, 2018 BRAZILIAN SYMPOSIUM ON ROBOTICS (SBR) AND 2018 WORKSHOP ON ROBOTICS IN EDUCATION (WRE). Novembro 6-10. João Pessoa. 2018. doi: 10.1109/LARS/SBR/WRE.2018.00049

FERREIRA, M.V.M.; TAVARES, J.J.P.Z.S.; SILVA, J.R. The Pheromone of Ant Emulated by Petri Net Inserted Inversely in RFID Database for Swarm Robots. In: PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON PETRI NETS AND SOFTWARE ENGINEERING (PNSE'18). Junho 24-29, Bratislava. 2018.

FENNANI, B.; HAMAM, H. and DAHMANE, A.O. RFID overview. In: ICM 2011 PROCEEDING. 2011. doi:10.1109/icm.2011.6177411

FINKENZELLER, K. **RFID Handbook: Fundamentals and Applications in Contactless Smart Cards, Radio Frequency Identification and Near-Field Communication**. 3.ed. Chichester: John Wiley & Sons, 2010. 478 p.

FONSECA, J.P.S.; TAVARES, J.J.P.S.Z. Petri Net with RFID Distributed Database for Autonomous Search and Rescue in Trails and Crossings. In: PROCEEDINGS OF THE INTERNATIONAL WORKSHOP ON PETRI NETS AND SOFTWARE ENGINEERING. Zaragoza, Spain, June 25–30. 2017.

FONSECA, J.P.S. **Redes de Petri de Alto Nível e PNRD Invertida Associadas ao Controle de Robôs Móveis: Uma Abordagem para Operações de Busca e Salvamento em Trilhas e Travessias.** 146 f. 2018. Tese de Doutorado, Universidade Federal de Uberlândia, Uberlândia.

FUJISAWA, R.; DOBATA, S.; KUBOTA, D.; IMAMURA, H.; MATSUNO, F. Dependency by concentration of pheromone trail for multiple robots. In: PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON ANT COLONY OPTIMIZATION AND SWARM INTELLIGENCE (ANTS 2008). Berlin, Germany: Springer, p. 283-290. 2008.

GARNIER, S.; TÂCHE, F.; COMBE, M.; GRIMAL, A.; THERAULAZ, G. Alice in Pheromone Land: An Experimental Setup for the Study of Ant-like Robots. In: PROCEEDINGS OF THE 2007 IEEE SWARM INTELLIGENCE SYMPOSIUM. Abril 1-7, Honolulu. 2007.

GLOVER, B.; BHATT, H. **RFID Essential**. (Theory in Practice (O'Reilly)). O'Reilly Media, 2006, 278p.

GRAVISH, N.; LAUDER, G.V. Robotics-inspired biology. **Journal of Experimental Biology.** v. 221-7**.** 2018. doi: 10.1242/jeb.138438

GRIECO, L. A.; RIZZO, A.; COLUCCI, S.; SICARI, S.; PIRO, G.; DI PAOLA, D.; BOGGIA, G. IoT-aided robotics applications: Technological implications, target domains and open issues. **Computer Communications**. Amsterdam, v. 54, p. 32-47, 2014.

HUANG, Y.S.; WENG, Y.S; ZHOU, M. Design of Traffic Safety Control Systems for Emergency Vehicle Preemption Using Timed Petri Nets. In: IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, v. 16-4, p. 2113-2120. 2015. doi: 10.1109/TITS.2015.2395419

JAULIN, L. **Mobile Robotics**. 2ed. STE Press – Elsevier. 2019. 314p.

JENSEN, K. **Coloured Petri Nets**. Basic Concepts, Analysis Methods and Practical Use. Volume 1. 2ed. Springer-Verlag Berlin Heidelberg. 1996. 236p.

JENSEN, K.; KRISTENSEN, L. M. **Coloured Petri Nets: Modelling and Validation of Concurrent Systems**. Berlin, Heidelberg: Springer-Verlag. 374 p. 2009.

KARLSON, P; BUTENANDT, A. Pheromones (Ectohormones) in insects. **Annual Review of Entomology.** v. 4, p. 39-58. 1959. https://doi.org/10.1146/annurev.en.04.010159.000351

KIM. P; KURABAYASHI, D. Forming an Artificial Pheromone Potential Field Using Mobile Robot and RFID Tags. In: IEEE/SICE INTERNATIONAL SYMPOSIUM ON SYSTEM INTEGRATION (SII). Kyoto, IEEE. 2011. doi: 10.1109/SII.2011.6147520

KOENIG, N.; HOWARD, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, Sendai, p. 2149-2154. 2004.

LIMA, D.A.; OLIVEIRA, G.M.B. A cellular automata ant memory model of foraging in a swarm of robots. **Applied Mathematical Modelling.** v. 47, p. 551-572. 2017. doi: https://doi.org/10.1016/j.apm.2017.03.021

MA, K.Y., CHIRARATTANANON, P., FULLER, S.B.; WOOD, R.J. Controlled flight of a biologically inspired, insect-scale robot. **Science.** v. 340, p. 603-607. 2013. doi: 10.1126/science.1231806

MAYET, R.; ROBERZ, J.; SCHMICKL, T.; CRAILSHEIM, K. Antbots: a feasible visual emulation of pheromone trails for swarm robots. In: PROCEEDINGS OF THE 7TH INTERNATIONAL CONFERENCE ON SWARM INTELLIGENCE. Berlin, Heidelberg: Springer-Verlag, p. 84-94. 2010.

MEC (Ministério da Educação). Aulas de robótica transformam reforço em apoio à criatividade. 2016. Disponível em: <http://portal.mec.gov.br/ultimas-noticias/222-537011943/42541-aulas-de-robotica-transformam-reforco-em-apoio-a-criatividade>. Acesso em: 10 dez. 2019.

MICHEL, O. Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. **International Journal of Advanced Robotics Systems.** Los Angeles, v. 1-1, p. 39-42. 2004.

MURATA, T. Petri Nets: Properties, analysis and applications. In: PROCEEDINGS OF THE IEEE**.** New York, v. 77-4, p. 541-580. 1989.

PERALTA, E.; FABREGAS, E.; FARIAS, G.; VARGAS, H.; DORMIDO, S. Development of a Khepera IV Library for the V-REP Simulator. **IFAC-PapersOnLine.** Amsterdam, v. 49-6, p. 81-86. 2016.

PETRI, C. A. **Kommunikation mit Automaten**. 1962. Phd dissertation. Darmstadt Technical Universiy, Darmstadt, Alemanha.

PINCIROLI, C.; TRIANNI, V.; O'GRADY, R.; PINI, G.; BRUTSCHY, A.; BRAMBILLA, M.; MATHEWS, N.; FERRANTE, E.; DICARO, G.; DUCATELLE, F.; STIRLING, T.; GUTIERREZ, A.; GAMBARDELLA, L.; DORIGO, M. ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotic. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS. San Francisco, p. 5027-5034. 2011.

QI, L.; ZHOU, M.; LUAN, W. A Two-level Traffic Light Control Strategy for Preventing Incident-Based Urban Traffic Congestion. In: IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, v. 19-1, p. 13-24. 2016. doi: 10.1109/TITS.2016.2625324

RAMEZANI, A., CHUNG, S.-J.; HUTCHINSON, S. A biomimetic robotic platform to study flight specializations of bats. **Science Robotics.** v. 2. 2017. doi: 10.1126/scirobotics.aal2505

RANJBAR-SAHRAEI, B.; ALERS, S.; TUYLS, K.; WEISS, G. StiCo in Action (Demonstration). In: 12TH INTERNATIONAL CONFERENCE ON AUTONOMOUS AGENTS AND MULTIAGENT SYSTEMS (AAMAS). Minnesota, maio 6-10. 2013.

ROHMER, E., SINGH, S.P.N.; FREESE, M. V-REP: a versatile and scalable robot simulation framework. In: PROC. OF THE INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS (IROS). Tokyo, IEEE. 2013. doi: 10.1109/IROS.2013.6696520

RUSSEL, R. A. Heat trails as short-lived navigational makers for mobile robots. In: PROCEEDINGS OF IEEE INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION (ICRA 1997). Piscataway, NJ: IEEE Press, p. 3534-3539. 1997.

SAHIN, E. Swarm Robotics: From Sources of Inspiration to Domains of Application. **Lecture Notes in Computer Science**. v. 3342, p.10-20. 2005. doi: 10.1007/978-3-540-30552-1_2

SHARMA, S.; PITHORA, A.; GUPTA, G.; GOEL, M.; SINHA, M. Traffic Light Priority Control For Emergency Vehicle Using RFID. **International Journal of Innovations in Engineering and Technology (IJIET).** 2013.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to Autonomous Mobile Robots.** Cambridge: The MIT Press, 2004. 335 p.

SILVA J.R., DEL FOYO P.M.G. “Timed Petri Nets”. Pawel Pawlewski (Ed.), **Petri Nets: Manufacturing and Computer Science**, InTech, p. 359-378. 2012.

SILVA, C.E.A.; TAVARES, J.J.P.Z.S.; FERREIRA, M.V.M. Arduino Library Developed for Petri Net Inserted into RFID Database and Variants. **Application and Theory of Petri Nets and Concurrency**, p.396-405. 2018. doi: 10.1007/978-3-319-91268-4_22

SIMMONS, R. G. Structured Control for Autonomous Robots**. IEEE Transactions on Robotics and Automation**. v. 10-1. 1994.

TANG Q., YU F., ZHANG Y., DING L., EBERHARD P. A Stigmergy Based Search Method for Swarm Robots. In: ADVANCES IN SWARM INTELLIGENCE. ICSI 2017. LECTURE NOTES IN COMPUTER SCIENCE, v. 10386. Springer, Cham. 2017.

TANGORRA, J., PHELAN, C., ESPOSITO, C.; LAUDER, G. Use of biorobotic models of highly deformable fins for studying the mechanics and control of fin forces in fishes**. Integrative and Comparative Biology.** v. 51-1, p. 176-189. 2011. doi: 10.1093/icb/icr036

TAVARES, J. J. P. Z. S.; SARAIVA, T. A. Elementary Petri net inside RFID distributed database (PNRD). **International Journal of Production Research**. London, v. 48, n. 9, p. 2563-2582, 2010. doi: 10.1080/00207540903564934

TEIXEIRA, G.T.; JAFELICE, R.S.M.; ALMEIDA, C.G.; VASCONCELOS, H.L. Autômato celular no estudo da reocupação de formigas cortadeiras em regiões de forrageamento. **Biomatemática**. v. 24. p. 77-90. 2014.

YOGESWARAN, M.; PONNAMBALAM, S.G. An Extensive Review of Research in Swarm Robotics. In: WORLD CONGRESS ON NATURE AND BIOLOGICALLY INSPIRED COMPUTING, NABIC 2009. p. 140-145. 2010. doi: 10.1109/NABIC.2009.5393617

APÊNDICE A

PROJETO DO *ATTABOT*

Esta seção disserta todo o projeto mecânico e eletrônico para a construção e montagem do robô móvel utilizado na tese (*Attabot*). O robô foi projetado uma vez que robôs autônomos comerciais possuem um alto valor no mercado, como por exemplo, um robô autônomo *e-puck* (https://www.cyberbotics.com/e-puck) custa em torno de $ 850. Tal valor de mercado deve-se à quantidade de sensores e atuadores agregados ao robô que não eram obrigatórios para este projeto, além da necessidade de acrescentar um leitor RFID. Sendo assim, projetar um robô próprio tornou-se viável.

O projeto mecânico e eletrônico bem como o modelo implementado no *software V-REP* foi publicado em Ferreira, Fonseca e Tavares (2018). Mais informações além das dispostas neste apêndice podem ser encontradas na referência.

A Fig. A.1 exibe o fluxograma do desenvolvimento do projeto iniciando pela etapa de definição dos requisitos de projeto. Com os requisitos bem definidos, pode-se então definir os componentes (estruturais e eletrônicos) utilizados no robô. As etapas de projeto eletrônico e projeto mecânico são realizadas em paralelo e se complementam, uma vez que qualquer alteração pode alterar ambas as etapas. As duas etapas concluem na etapa de montagem do robô e o projeto de desenvolvimento é finalizado com testes no robô. Caso algum problema seja identificado nesta etapa os requisitos de projeto devem ser revisados, visto que a definição dos mesmos não tenha sido feita de forma adequada.

A.1. Requisitos de projeto e definição dos componentes

A etapa de requisitos de projeto é de suma importância para o desenvolvimento de um projeto, pois, é ela quem definirá o tipo de robô além de suas funcionalidades. A Tab. A1 define os requisitos que o robô deve conter. Os requisitos obrigatórios são de que o robô deve ser autônomo e ser capaz de ler e escrever em etiquetas RFID, já que toda a ideia da tese se baseia nestes dois princípios.

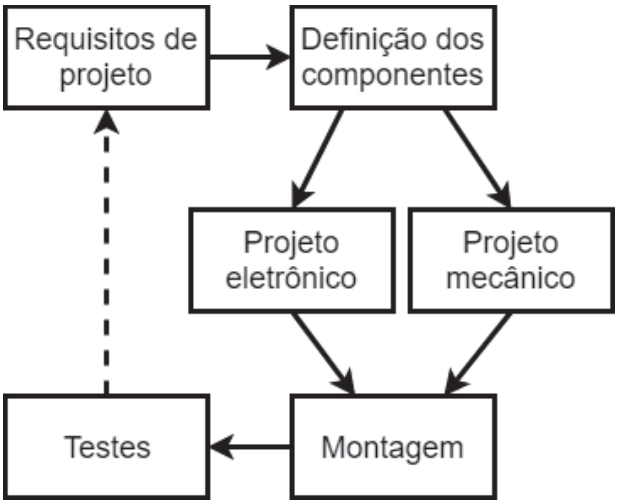


Figura A.1 – Fluxograma para desenvolvimento do robô móvel.

Tabela A.1 – Requisitos do *Attabot*.

	Requisitos	Prioridade
RQ1	O robô deve ser autônomo	Altíssima
RQ2	O robô deve ler e escrever em etiquetas de rádio frequência anexadas no chão do ambiente	Altíssima
RQ3	O robô deve ser de baixo custo	Alta
RQ4	O robô deve seguir uma faixa	Alta
RQ5	O robô não pode colidir com outro robô	Alta
RQ6	O robô deve movimentar-se para frente e para trás além de fazer curvas	Alta
RQ7	O robô deve ter dimensões reduzidas	Baixa
RQ8	O robô deve funcionar por pelo menos meia hora	Baixa

Tais requisitos levam à definição dos componentes:

a) Quantidade de rodas e motores: optou-se por dois motores DC 6 V com caixa de redução e duas rodas para reduzir os custos do projeto, além de um *driver* de motor com ponte H (L298N). Para não tornar o robô um pêndulo invertido as rodas localizam-se no eixo central do robô e há apoios na parte frontal e traseira do *Attabot*.

b) Leitor RFID: é o elemento responsável por ler e gravar nas etiquetas RFID espalhadas pelo ambiente e o leitor PN532 foi utilizado neste projeto.

c) Sensores de distância e linha: estes sensores são responsáveis pelo controle reativo do robô e optou-se por um sensor de distância ultrassônico (HC-SR04) e um sensor de linha de 4 canais.

d) Controlador: a fim de facilitar a replicação dos robôs o microcontrolador é da família Arduino, sendo que a quantidade de entradas e saídas necessárias para alimentar todos os sensores e atuadores definiu a utilização do Arduino Mega 2560.
e) Alimentação: optou-se por utilizar duas baterias de 4,2 V de Li-íon com aproximadamente 8000 mAh de carga.

Os componentes foram escolhidos para atender os requisitos, isto é, o leitor RFID é responsável pelo RQ2, o sensor de linha pelo RQ4, o sensor ultrassônico pelo RQ5, a ponte H do *driver* do motor pelo RQ6, a bateria de Li-íon pelo RQ8. Os RQ3 e RQ7 são responsáveis pela escolha de apenas dois motores e duas rodas.

A.2. Projeto mecânico e projeto eletrônico

Como o projeto do robô tem o caráter de seguir o movimento DIY para que mais pessoas possam replicar ou até mesmo propor melhorias no modelo do robô optou-se por utilizar diferentes princípios de construção mecânica. Algumas peças são de MDF (*Medium-Density Fiberboard*) e podem ser cortadas em máquina de corte a laser, algumas peças são produzidas através do processo FDM (*Fused Deposition Modeling*) em impressora 3D com plástico ABS (Acrilonitrila butadieno estireno), e o restante das peças podem ser adquiridas em casas ferragistas.

Como um dos requisitos do projeto é de que o robô tenha dimensões reduzidas, o robô foi projetado em andares, sendo que cada andar aloca uma certa quantidade de componentes. A Fig. A.2 apresenta os modelos 2D dos andares com os furos de 4 mm de diâmetro para fixação dos parafusos e barra rosqueada. O chassi é quadrado e possui 66 mm de largura. Para evitar cantos vivos, as quinas foram arredondadas com um raio de 5 mm.

Os suportes dos componentes eletrônicos e as rodas são impressas e a Fig. A.3 expõe os cinco modelos 3D: suporte para o motor DC (Fig. A.3.1); roda de 40 mm de diâmetro (Fig. A.3.2); suporte para módulo seguidor de linha (Fig. A.3.3); suporte para sensor ultrassônico (Fig. A.3.4); e suporte para Arduino Mega 2560 (Fig. A.3.5). O suporte do microcontrolador é fixado pelas barras rosqueadas na parte traseira do robô. Os suportes restantes possuem furos para fixação com parafusos de 3 mm de diâmetro.

Figura A.2 – Desenhos 2D para o corte do chassi em MDF.

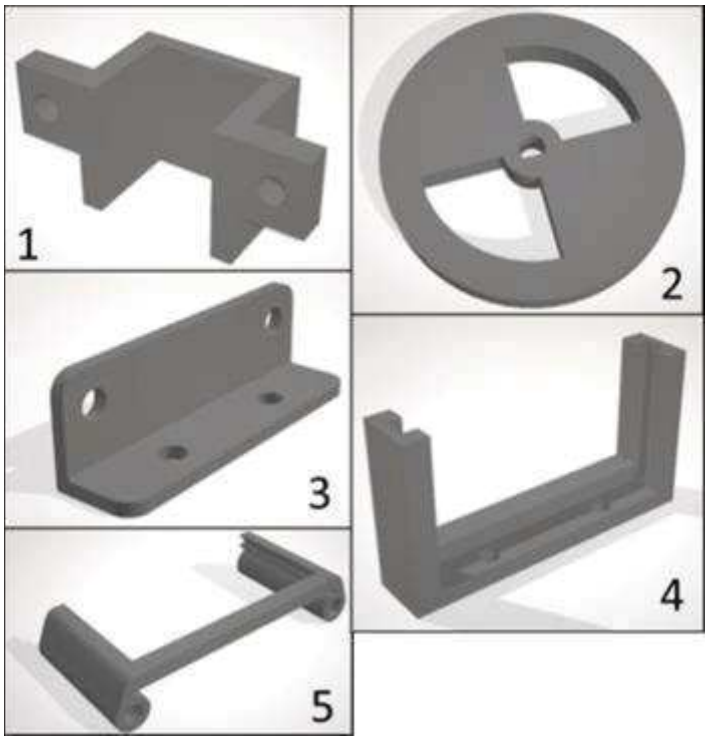


Figura A.3 – Modelos 3D para impressão.

Os arquivos 2D (extensão .DXF) e 3D (extensão .STL) estão disponíveis para *download* em *https://github.com/jpsfonseca/Attabot*.

A Fig. A.4 aponta a forma como os componentes eletrônicos estão associados ao microcontrolador. As setas de cor preta indicam alimentação, ou seja, a bateria é responsável por alimentar o microcontrolador que por sua vez, alimenta os sensores e os motores através do *driver* ponte H. Isso significa que apesar do motor DC ter uma tensão nominal de 6 V, a máxima tensão de saída do microcontrolador é de 5 V, o que afeta apenas na velocidade do robô. As setas de cor azul indicam o sinal trocado entre os dispositivos. Os sensores ultrassônicos e seguidor de linha apenas enviam sinal ao microcontrolador. Visto

que o leitor RFID PN532 é capaz de acessar uma *tag* para leitura e escrita ele funciona hora como sensor, hora como atuador. Por fim, a ponte H recebe o sinal de atuação do microcontrolador e atua sobre os motores.

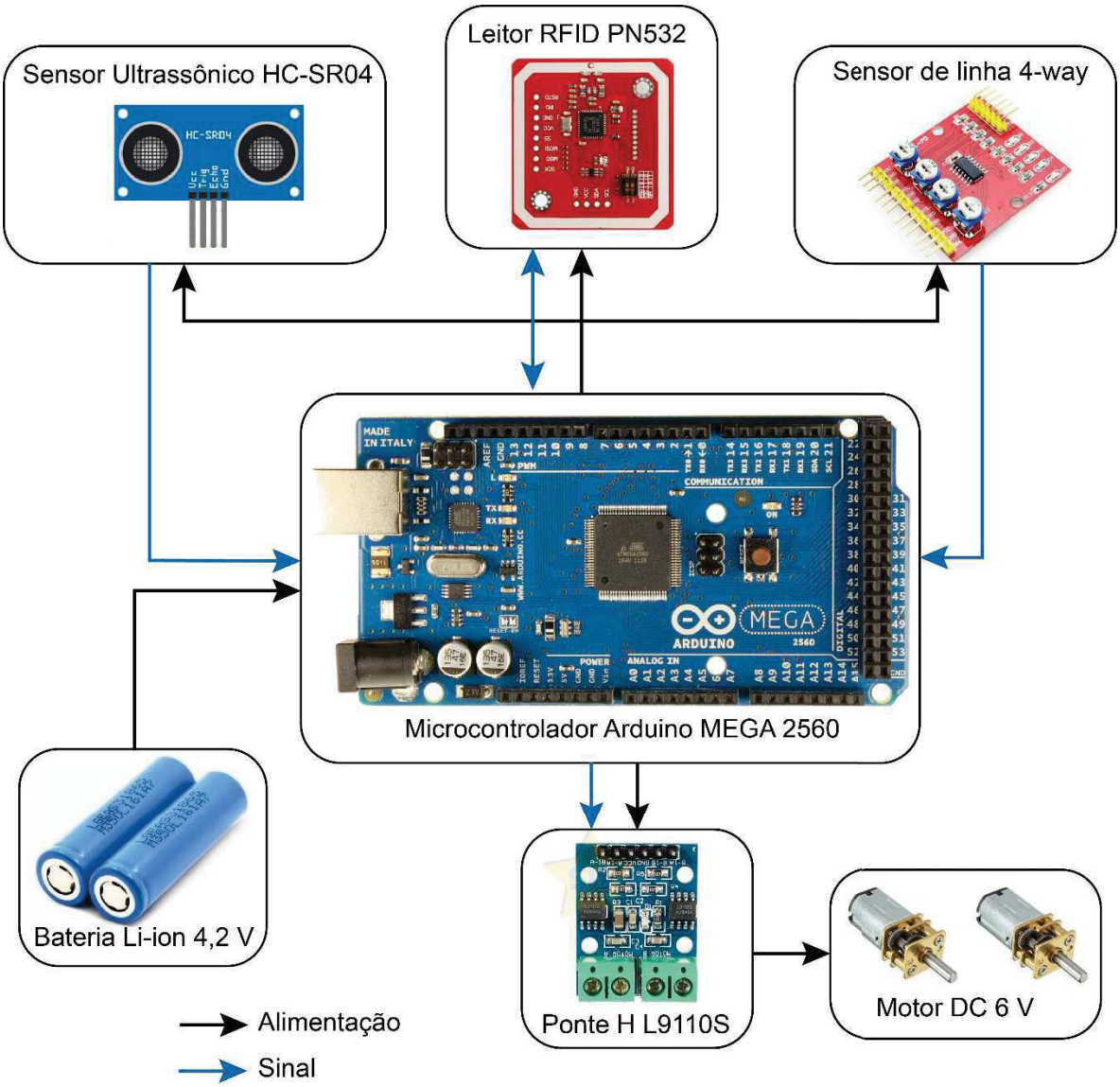


Figura A.4 – Representação esquemática dos componentes eletrônicos aplicados ao robô móvel.

A.3. Comportamento reativo e comportamento deliberativo

Para Simmons (1994), o comportamento reativo é capaz de detectar e responder às mudanças do ambiente quanto ao “aqui e agora”. Reativo não necessariamente implica e reflexivo (conexões diretas entre sensores e atuadores), ou seja, qualquer cálculo computacional é permitido desde que o sistema seja rápido o suficiente para responder à

situação requerida. Já o comportamento deliberativo concerne em como as tarefas interagem tal qual em um planejamento de trajetória.

O comportamento reativo do *Attabot* é realizado por um sensor ultrassônico alocado na frente do robô com o intuito de evitar a colisão com obstáculos. Isto é, o robô para sempre que percebem uma parede ou outro robô. O módulo seguidor de linha faz com que o robô siga um determinado caminho. No comportamento deliberativo o módulo RFID PN532 é responsável pelas tomadas de decisões do *Attabot* através das informações salvas nas *tags* espalhadas pelo ambiente.

A.4. Montagem

Para a montagem do robô são apresentadas a Tab. A.2 e Fig. A.5. A tabela lista os componentes eletrônicos e mecânicos do robô, enquanto a figura esclarece a posição de cada componentes (de *A* à *E* para os componentes eletrônicos e de *a* à *h* para os mecânicos).

Tabela A.2 – Especificação dos componentes do *Attabot*.

	Identificador	Descrição
Componentes eletrônicos	A	Arduino Mega 2560
	B	Bateria Li-ion 18650 4.2V 8860 mAh
	C	Sensor de linha infravermelho de 4 sondas
	D	RFID *reader/writer* PN532
	E	Sensor ultrassônico HC-SR04
	F	Motor DC 6V 40 rpm
	E	Botão *on/off*
Componentes mecânicos	a	Barra roscada M4
	b	Porca sextavada M4
	c	MDF 3 mm cru
	d	Suporte impresso para o sensor ultrassônico
	e	Roda impressa
	f	Suporte impresso para o microcontrolador
	g	Suporte impresso para os motores
	h	Suporte impresso do sensor de linha

Como o robô foi projetado em andares, a montagem deve ser iniciada pelo primeiro andar do chassi. Um vídeo com a montagem do *Attabot* está disponível em https://youtu.be/uXzjaMAsQyQ. O módulo RFID PN532 encontra-se na parte inferior do chassi (oculto na figura). O *driver* dos motores, também oculto na figura, está localizado na parte inferior do chassi do segundo andar.

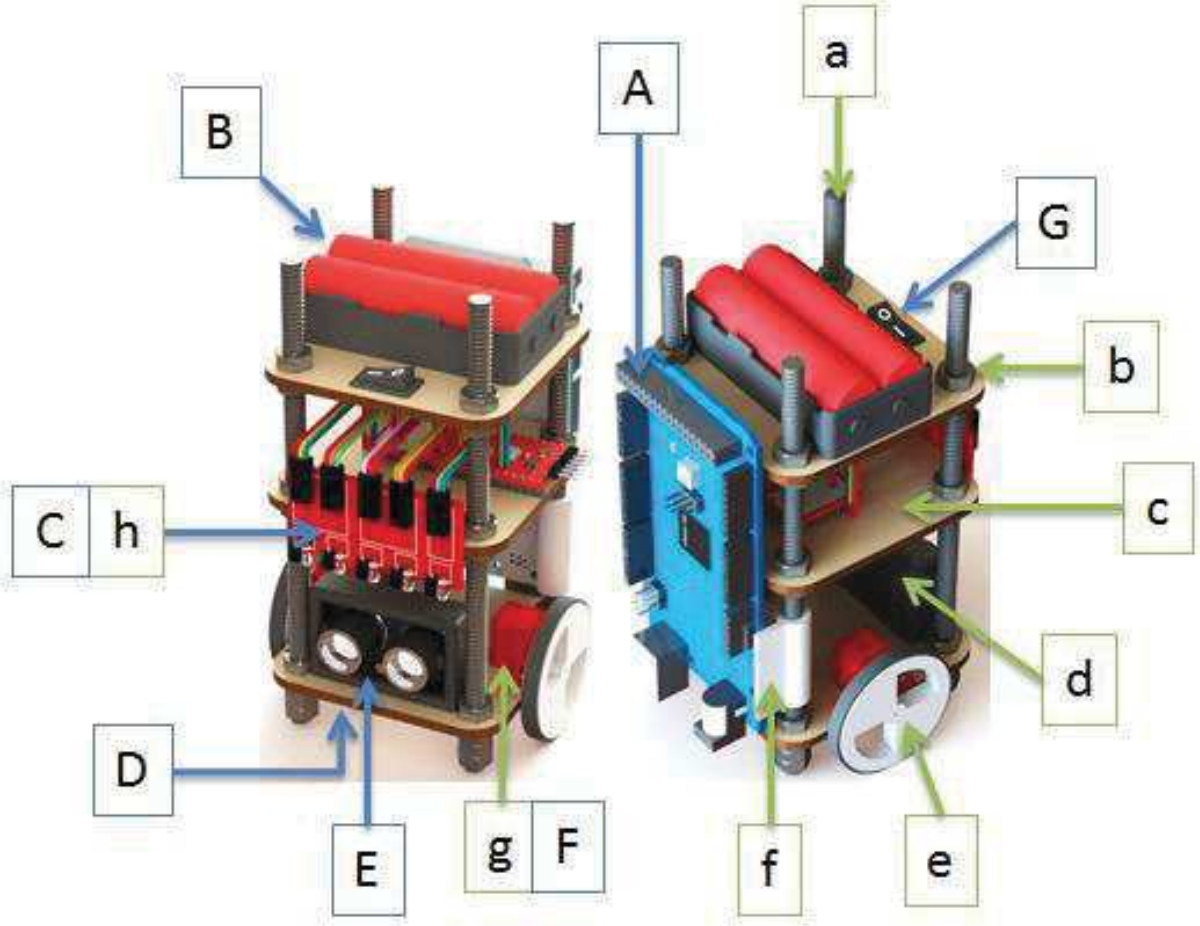


Figura A.5 – Renderização do robô móvel com os principais elementos para montagem.

APÊNDICE B

IMPLEMENTAÇÃO EXPERIMENTAL

B.1. Robô móvel

O Quad. B.1 apresenta o código em Arduino do robô móvel. O disparo das transições da iPNRD utiliza a biblioteca para Arduino desenvolvida por Silva, Ferreira e Tavares (2018). As funções *leituraTag()*, *leitura()* e *salva()* são responsáveis pela leitura e escrita das etiquetas, além da execução da Rede de Petri.

Quadro B.1 – Código em Arduino para o robô móvel.

```
// definições do leitor
#include <Pn532NfcReader.h>
#include <PN532_HSU.h>
uint16_t data[] = {0, 0};
uint16_t data1[] = {0, 0};
uint16_t datae[] = {0, 0};
int pin = 30; // porta do rele
//Rotines related with the configuration mof the RFID reader PN532
PN532_HSU pn532hsu(Serial1);
NfcAdapter nfc1 = NfcAdapter(pn532hsu);
//Creation of the reader and PNRD objects
Pn532NfcReader* reader1 = new Pn532NfcReader(&nfc1);
Pnrd pnrd1 = Pnrd(reader1,2,1, false, false);
uint32_t tagID = 0xFF;

// Definições dos sensore e atuadores do Robo
// Motores
// Motor A -> esquerda
// Motor B -> direita
int motorB1A = 4;
int motorB1B = 5;
int motorA1A = 6;
int motorA1B = 7;

// Leitor Seguidor de linha
// 0 - branco - led on
// 1 - preto - led off
int pin_out1 = 22;
int pin_out2 = 23;
int pin_out3 = 24;
```

```
int pin_out4 = 25;

boolean out1, out2, out3, out4;

void D90();
void E90();
void D();
void E();
void M();
void S();
void B();
void C();

void movimenta();
void leituraTag();

boolean erro = false;

boolean executando = false;
int timem = 50;
int tempoE = 8000;

int t1 = 0;
int t2 = 0;
boolean leu = false;

int cont = 0;

int pwmval = 170;
int caso =-1;

int trilha = 0;


// funções de leitura e escrita na tag
void leitura();
void salva();

#include <HCSR04.h>
UltraSonicDistanceSensor sensorD(44, 45);
UltraSonicDistanceSensor sensorF(46, 47);
int distD = 0;
int distF = 0;
boolean parou = false;

int tb = 400;

void setup() {
  // put your setup code here, to run once:
  pinMode(motorB1A, OUTPUT);
  pinMode(motorB1B, OUTPUT);
  pinMode(motorA1A, OUTPUT);
  pinMode(motorA1B, OUTPUT);
  digitalWrite(motorB1A, LOW);
  digitalWrite(motorB1B, LOW);
  digitalWrite(motorA1A, LOW);
  digitalWrite(motorA1B, LOW);
  pinMode(pin_out1, INPUT);
  pinMode(pin_out2, INPUT);
  pinMode(pin_out3, INPUT);
```

```
  pinMode(pin_out4, INPUT);
  Serial.begin(9600);
  pinMode(pin, OUTPUT);
  digitalWrite(pin,LOW);

}

void loop() {

  distD = sensorD.measureDistanceCm();
  distF = sensorF.measureDistanceCm();

  //Serial.println(distF);

  parou = false;
  if(distF > 0 && distF <=4){
    B();
    delay(600);
    S();
    delay(tempoE);
    parou = true;
    }

    if(parou == false){

  // leitura do seguidor de linha
  out1 = digitalRead(pin_out1);
  out2 = digitalRead(pin_out2);
  out3 = digitalRead(pin_out3);
  out4 = digitalRead(pin_out4);
  // analise do seguidor de linha
  if((out1 == HIGH) && (out2 == HIGH) && (out3 == HIGH) && (out4 == HIGH))
{ // 1111 tudo preto
    caso = 0;
    //Serial.println(caso);
  }else if(((out1 == HIGH) && (out2 == HIGH) && (out3 == LOW) && (out4 ==
HIGH)) || ((out1 == HIGH) && (out2 == LOW) && (out3 == HIGH) && (out4 ==
HIGH)) || ((out1 == HIGH) && (out2 == LOW) && (out3 == LOW) && (out4 ==
HIGH)) ){ // 1101 ou 1011 ou 1001
     caso = 1;
     //Serial.println(caso);
  }else if ((out1 == HIGH) && (out2 == LOW) && (out3 == LOW) && (out4 ==
LOW) ){ // 1000
    caso = 2;
    //Serial.println(caso);
  }else if ((out1 == HIGH) && (out2 == HIGH) && (out3 == LOW) && (out4 ==
LOW)) { // 1100
    caso = 3;
    //Serial.println(caso);
  } else if((out1 == LOW) && (out2 == LOW) && (out3 == LOW) && (out4 ==
HIGH)) { // 0001
    caso = 4;
    //Serial.println(caso);
  } else if((out1 == LOW) && (out2 == LOW) && (out3 == HIGH) && (out4 ==
HIGH)) { // 0011
    caso = 5;
    //Serial.println(caso);
  } else if((out1 == LOW) && (out2 == LOW) && (out3 == LOW) && (out4 ==
LOW)) { // 0000 tudo branco
    caso = 6;
```

```
    //Serial.println(caso);
  } else {
    caso = 7;
    //Serial.println(caso);
  }

  if(leu){
    cont = cont + 1;
    if(cont > 50){
      leu = false;
      cont = 0;
      }
    }

  if(distD > 0 && distD <= 4 && leu == false){
      leituraTag();
    }

  movimenta();
    }
}

void E90() {
  // motor B gira e motor A gira invertido
  analogWrite(motorB1A, pwmval);
  analogWrite(motorB1B, 0);
  analogWrite(motorA1A, 0);
  analogWrite(motorA1B, pwmval);
  delay(timem);
  S();
}

void D90() {
  // motor A gira e motor B gira invertido
  analogWrite(motorB1A, 0);
  analogWrite(motorB1B, pwmval);
  analogWrite(motorA1A, pwmval);
  analogWrite(motorA1B, 0);
  delay(timem);
  S();
}

void E() {
  // motor B gira e motor A para
  analogWrite(motorB1A, pwmval);
  analogWrite(motorB1B, 0);
  analogWrite(motorA1A, 0);
  analogWrite(motorA1B, 0);
  delay(timem);
  S();
}

void D() {
  // motor A gira e motor B para
  analogWrite(motorB1A, 0);
  analogWrite(motorB1B, 0);
  analogWrite(motorA1A, pwmval);
  analogWrite(motorA1B, 0);
  delay(timem);
  S();
```

```
}

void M() {
  // motor A e B giram
  analogWrite(motorB1A, pwmval);
  analogWrite(motorB1B, 0);
  analogWrite(motorA1A, pwmval);
  analogWrite(motorA1B, 0);
  delay(timem);
}

void S() {
  // motor A e B param
  analogWrite(motorB1A, 0);
  analogWrite(motorB1B, 0);
  analogWrite(motorA1A, 0);
  analogWrite(motorA1B, 0);
}

void B(){
  analogWrite(motorB1A, 0);
  analogWrite(motorB1B, pwmval);
  analogWrite(motorA1A, 0);
  analogWrite(motorA1B, pwmval);
  delay(timem);
  }

  void C(){
    M();
    }

void movimenta(){

  if(executando){

    }else{
    switch (caso) {
    case 0:
      B();
      B();
      E90();
      S();
      break;
    case 1:
      B();
      B();
      E90();
      S();
      break;
    case 2:
      D();
      break;
    case 3:
      D90();
      break;
    case 4:
      E();
      break;
    case 5:
      E90();
```

```
      break;
    case 6:
      M();
      break;
    case 7:
      B();
      B();
      E90();
      S();
      break;
    default:
      B();
      B();
      E90();
      S();
    }
  }
}

  void leitura(){
    delay(100);
      pnrd1.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
      ReadError readError = pnrd1.getData();
      if(readError == ReadError::NO_ERROR){
          pnrd1.getTokenVector(data);
          tagID = pnrd1.getTagId();
      }
    }

    void salva(){
      delay(100);
      Serial.println("Salvando");
      pnrd1.setTokenVector(data);
        if(pnrd1.saveData() == WriteError::NO_ERROR){
          //Serial.println("Tag configurated successfully.");
//          Serial.println("Verificando");
//          leitura();
          }else{
            Serial.println("Erro");
            erro = true;
        }
    }

    void leituraTag(){
      S();
      digitalWrite(pin, LOW);
      delay(300);
      reader1->initialize();

          data[0] = 0;
          data[1] = 0;
          tagID = 0xFF;
          leitura();
          data1[0] = data[0];
          data1[1] = data[1];

          Serial.print("Data: ");
          Serial.print(data1[0]);
          Serial.print(" ");
          Serial.println(data1[1]);
```

```
if(tagID == 0xE5){ // trilha B
  trilha = 2;
  Serial.println("tag N2");
  if(data1[0] == 0 && data1[1] == 0){
    data[0] = 0;
    data[1] = 1;
    salva();
    digitalWrite(pin, HIGH);
    B();
    delay(0.7*tb);
    S();
    digitalWrite(pin, HIGH);
    delay(tempoE);
    }else if(data1[0] == 1 && data1[1] == 0){
      data[0] = 1;
      data[1] = 1;
      salva();
      digitalWrite(pin, HIGH);
      M();
      leu = true;
      }else{
        digitalWrite(pin, HIGH);
        B();
        delay(tb);
        S();
        delay(2000);
        }
  }else if(tagID == 0x55){ // trilha A
    trilha = 1;
    Serial.println("tag N1");
    if(data[0] == 0 && data[1] == 0){
      data[0] = 0;
      data[1] = 1;
      salva();
      digitalWrite(pin, HIGH);
      B();
      delay(0.7*tb);
      S();
      delay(tempoE);
      }else if(data[0] == 1 && data[1] == 0){
        data[0] = 1;
        data[1] = 1;
        salva();
        digitalWrite(pin, HIGH);
        M();
        leu = true;
      }else{
        digitalWrite(pin, HIGH);
        B();
        delay(tb);
        S();
        delay(2000);
        }
  }else if(tagID == 0xC5){ // entrada do feromônio
    if(trilha == 1){
      data[0] = 1;
      data[1] = 0;
      }else if(trilha == 2){
        data[0] = 0;
```

```
                    data[1] = 1;
                    }
                trilha = 0;
                salva();
                digitalWrite(pin, HIGH);
                delay(5000);
                M();
                leu = true;
                }else if(tagID == 0x54){ // saindo do feromonio
                  digitalWrite(pin, HIGH);
                  delay(5000);
                  if(data[0] == 1 && data[1] == 0){ // vira pra trilha A
                    E90();
                    E90();
                    M();
                    M();
                    E90();
                    M();
                    E90();
                    M();
                    leu = true;
                    }else if(data[0] == 0 && data[1] == 1){ // vira pra
trilha B
                      M();
                      delay(300);
                      leu = true;
                      }else{
                        long randN = random(20);
                        if(randN < 10){
                          E90();
                          M();
                          M();
                          E90();
                          M();
                          E90();
                          M();
                          leu = true;
                          }else{
                            M();
                            delay(300);
                            leu = true;
                            }
                        }
                  }else{
                    digitalWrite(pin, HIGH);
                    B();
                    delay(tb);
                    S();
                    delay(2000);
                  }
          digitalWrite(pin, HIGH);
        }
```

B.2. Feromônio

O Quad. B.2 apresenta o código em Arduino do feromônio. Como este agente salva os parâmetros do feromônio para avaliação é necessária a inclusão das bibliotecas *SD.h* e *SPI.h* para utilização do cartão SD. As funções *zeraTags()*, *lephe1(), atualizaphe()* e *salvaphe2()* são responsáveis pela leitura e escrita das etiquetas, além da execução da Rede de Petri.

Quadro B.2 – Código em Arduino para o feromônio.

```
// código do feromônio
#include <Pn532NfcReader.h>
#include <PN532_HSU.h>

//Rotines related with the configuration of the RFID reader PN532
PN532_HSU pn532hsu(Serial1);
NfcAdapter nfc1 = NfcAdapter(pn532hsu);
//Creation of the reader and PNRD objects
Pn532NfcReader* reader1 = new Pn532NfcReader(&nfc1);
Pnrd pnrd1 = Pnrd(reader1,2,1, false, false);

//Rotines related with the configuration of the RFID reader PN532
PN532_HSU pn532hsu2(Serial2);
NfcAdapter nfc2 = NfcAdapter(pn532hsu2);
//Creation of the reader and PNRD objects
Pn532NfcReader* reader2 = new Pn532NfcReader(&nfc2);
Pnrd pnrd2 = Pnrd(reader2,2,1, false, false);

uint32_t tagId = 0xFF;

uint16_t phe1[] = {0, 0};
uint16_t phe2[] = {0, 0};
uint16_t phe2a[] = {0, 0};
int pin1 = 30;
int pin2 = 32;

int cont = 0;

unsigned int t1 = 0;
unsigned int t2 = 0;
unsigned int tmax = 5000;

void zeraTags();
void lephe1();
void atualizaphe();
void salvaphe2();

int contpheA = 0;
int contpheB = 0;
float cpheAi = 100;
float cpheA = 100;
float cpheBi = 100;
float cpheB = 100;
int cphemax = 200;
float pheA = 100;
```

```
float pheB = 100;
float sourceA = 0;
float sourceB = 0;
float sourceAt = 300;
float sourceBt = 0;
float sourceAc = 0;
float sourceBc = 0;

int i = 0;
boolean comida = false;


// cartão SD
#include <SD.h>
#include <SPI.h>
File myFile;
int pinoSS = 53; // Pin 53 para Mega / Pin 10 para UNO

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);

  delay(100);
  zeraTags();

  pinMode(pinoSS, OUTPUT); // Declara pinoSS como saída
  if (SD.begin()) { // Inicializa o SD Card
  Serial.println("SD Card pronto para uso."); // Imprime na tela
  }else {
  Serial.println("Falha na inicialização do SD Card.");
  return;
  }

  myFile = SD.open("dado.txt", FILE_WRITE); // Cria / Abre arquivo .txt
   if (myFile) { // Se o Arquivo abrir imprime:
  Serial.println("Escrevendo no Arquivo .txt"); // Imprime na tela
  }

}

void loop() {
  // put your main code here, to run repeatedly:
  i = i+1;
//  Serial.print("i: ");
  Serial.println(i);
  if(i > 200 && comida == false){
//    Serial.println("ALTEROU");
    comida = true;
    sourceBt = 300;
    }
  t1 = millis();

  lephe1();
  delay(100);
  atualizaphe();
  delay(100);
  salvaphe2();
  // pheA pheB phe2[0] phe2[1] sourceAt sourceBt
```

```
  myFile.print(pheA);
  myFile.flush();
  myFile.print(" ");
  myFile.flush();
  myFile.print(pheB);
  myFile.flush();
  myFile.print(" ");
  myFile.flush();
  myFile.print(phe2[0]);
  myFile.flush();
  myFile.print(" ");
  myFile.flush();
  myFile.print(phe2[1]);
  myFile.flush();
  myFile.print(" ");
  myFile.flush();
  myFile.print(sourceAt);
  myFile.flush();
  myFile.print(" ");
  myFile.flush();
  myFile.println(sourceBt);
  myFile.flush();


  t2  = millis();
  delay(tmax - (t2-t1));
}

void lephe1(){
  // inicializa leitor
  digitalWrite(pin1, LOW);
  //Serial.println("lephe1");
  delay(200);
  reader1->initialize();
  delay(400);

  // le tag
  pnrd1.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
  ReadError readError = pnrd1.getData();
  if(readError == ReadError::NO_ERROR){
      pnrd1.getTokenVector(phe1);
  }else{
    phe1[0] = 0;
    phe1[1] = 0;
    Serial.println("Não leu");
    }
  Serial.print("phe1: ");
  Serial.print(phe1[0]);
  Serial.print(" ");
  Serial.println(phe1[1]);

  if(phe1[0] > 0 && phe1[1] == 0){
    //sourceA = phe1[0];
    if(sourceAt > 10){
      sourceA = 4*(sourceAt - sourceAc/100)/sourceAt;
      sourceAc = sourceAc + sourceA;
      sourceAt = sourceAt - sourceA;
      }else{
        pheA = 0.80*cpheA;
        cpheA = pheA;
```

```
        sourceA = 0;
        sourceAc = sourceAc + sourceA;
        sourceAt = sourceAt - sourceA;
        }

    contpheA = 0;
    cpheA = pheA + sourceA;
    if (cpheA > cphemax){
        cpheA = cphemax;
    }
    phe1[0] = 0;
    phe1[1] = 0;
    pnrd1.setTokenVector(phe1);
    if(pnrd1.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag Phe1 salva.");
    }else{
      Serial.println("Tag Phe1 nao foi salva.");
      }
    }else if(phe1[0] == 0 && phe1[1] > 0){
      //sourceB = phe1[1];
      if(sourceBt > 10){
        sourceB = 4*(sourceBt - sourceBc/100)/sourceBt;
        sourceBc = sourceBc + sourceB;
        sourceBt = sourceBt - sourceB;
      }else{
        pheB = 0.8*cpheB;
        cpheB = pheB;
        sourceB = 0;
        sourceBc = sourceBc + sourceB;
        sourceBt = sourceBt - sourceB;
        }
      contpheB = 0;
      cpheB = pheB + sourceB;
      if (cpheB > cphemax){
        cpheB = cphemax;
      }
      phe1[0] = 0;
      phe1[1] = 0;
      pnrd1.setTokenVector(phe1);
      if(pnrd1.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag Phe1 salva.");
    }else{
      Serial.println("Tag Phe1 nao foi salva.");
      }
    }
      digitalWrite(pin1, HIGH);
}

 void  atualizaphe(){
//    Serial.println("atualizaphe");
//    Serial.print("PheA: ");
//    Serial.print(pheA);
//    Serial.print("   PheB: ");
//    Serial.println(pheB);
//    Serial.print("cpheA: ");
//    Serial.print(cpheA);
//    Serial.print("   cpheB: ");
//    Serial.println(cpheB);
//    Serial.print("cpheAi: ");
//    Serial.print(cpheAi);
```

```
//    Serial.print("   cpheBi: ");
//    Serial.println(cpheBi);

    Serial.print("SourceA: ");
    Serial.print(sourceAt);
    Serial.print("   sourceB: ");
    Serial.println(sourceBt);

    pheA = cpheA*pow(10, - contpheA/(cpheAi*cpheAi));
    pheB = cpheB*pow(10, - contpheB/(cpheBi*cpheBi));

    Serial.print("ContA: ");
    Serial.print(contpheA);
    Serial.print("   ContB: ");
    Serial.print(contpheB);
    Serial.print("   CpheA: ");
    Serial.print(cpheA);
    Serial.print("   CpheB: ");
    Serial.println(cpheB);
    contpheA = contpheA + 10;
    contpheB = contpheB + 10;

    Serial.print("PheA: ");
    Serial.print(pheA);
    Serial.print("   PheB: ");
    Serial.println(pheB);

    if(pheA/(pheA + pheB) > 0.52){
     Serial.println("Feromonio A");
     phe2[0] = 1;
     phe2[1] = 0;
     }else if(pheB/(pheA+pheB) > 0.52){
       Serial.println("Feromonio B");
       phe2[0] = 0;
       phe2[1] = 1;
       }else{
         //Serial.println("Feromonios iguais");
         phe2[0] = 0;
         phe2[1] = 0;
       }
}

void salvaphe2(){
  //Serial.println("salvaphe2");

  if(phe2a[0] == phe2[0] && phe2a[1] == phe2[1] && cont < 20){
    //Serial.println("Nao precisa atualizar");
    cont = cont + 1;
    }else{
      cont = 0;
      phe2a[0] = phe2[0];
      phe2a[1] = phe2[1];
      digitalWrite(pin2, LOW);
      delay(200);
      reader2->initialize();
      delay(200);
      //Setting of the classic iPNRD approach
      pnrd2.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
      pnrd2.setTokenVector(phe2);
        if(pnrd2.saveData() == WriteError::NO_ERROR){
```

```
          Serial.println("Tag2 salva.");
        }else{
          Serial.println("Tag2 nao foi salva.");
          }
          digitalWrite(pin2, HIGH);
    }
}

void zeraTags(){
  Serial.println("zera tags");
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  delay(100);
  reader1->initialize();
  delay(200);
  //Setting of the classic iPNRD approach
  pnrd1.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
  Serial.println("Reader1 iniciado");
   pnrd1.setTokenVector(phe1);
    if(pnrd1.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag1 salva.");
    }else{
      Serial.println("Tag1 nao foi salva.");
      }

  reader2->initialize();
  delay(200);
  //Setting of the classic iPNRD approach
  pnrd2.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
  Serial.println("Reader2 iniciado");
    pnrd2.setTokenVector(phe2);
    if(pnrd2.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag2 salva.");
    }else{
      Serial.println("Tag2 nao foi salva.");
      }
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, HIGH);
}
```

B.3. Semáforo

O Quad. B.3 apresenta o código em Arduino do semáforo. As funções *zeraTags()* e *atualizaPN()* são responsáveis pela leitura e escrita das etiquetas, além da execução da Rede de Petri.

Quadro B.3 – Código em Arduino para o semáforo.

```
#include <Pn532NfcReader.h>
#include <PN532_HSU.h>

uint16_t data1[] = {0, 0};
uint16_t data2[] = {0, 0};
uint16_t datae1[] = {0, 0};
int pin1 = 30;
int pin2 = 32;


// definição dos leds do semáforo
  int ledsema1Red = 48;
  int ledsema1Green = 49;
  int ledsema2Red = 52;
  int ledsema2Green = 51;

  // places da PN
  boolean sema1Red = true;         // p1
  boolean sema1Green = false;      // p2
  boolean sema2Red = true;         // p3
  boolean sema2Green = false;      // p4
  boolean inA = false;             // p5
  boolean inB = false;             // p6
  boolean outA = false;            // p7
  boolean outB = false;            // p8
  boolean leaving = false;         // p9
  boolean TLfree = true;           // p10
  boolean TLbusy = false;          // p11

  // trasitions da PN
  boolean foundN1 = false;
  boolean foundN2 = false;
  boolean N1G2R = false;
  boolean N1R2G = false;
  boolean N2G2R = false;
  boolean N2R2G = false;
  boolean leavingN1 = false;
  boolean leavingN2 = false;
  boolean Busy2Free = false;

  unsigned int tant = 0;
  unsigned int t = 0;
  unsigned int t1 = 0;
  unsigned int t2 = 0;
  unsigned int tmax = 5000;
  unsigned int td = 400;

  boolean erro = false;
```

```
  boolean altera1 = false;
  boolean altera2 = false;
  boolean altera3 = false;


//Rotines related with the configuration of the RFID reader PN532
PN532_HSU pn532hsu(Serial1);
NfcAdapter nfc1 = NfcAdapter(pn532hsu);
//Creation of the reader and PNRD objects
Pn532NfcReader* reader1 = new Pn532NfcReader(&nfc1);
Pnrd pnrd1 = Pnrd(reader1,2,1, false, false);


//Rotines related with the configuration of the RFID reader PN532
PN532_HSU pn532hsu2(Serial2);
NfcAdapter nfc2 = NfcAdapter(pn532hsu2);
//Creation of the reader and PNRD objects
Pn532NfcReader* reader2 = new Pn532NfcReader(&nfc2);
Pnrd pnrd2 = Pnrd(reader2,2,1, false, false);

uint32_t tagId = 0xFF;

int contg1 = 0;
int contg2 = 0;

void zeraTags();

void setup() {
  //Initialization of the communication with the reader and with the
computer Serial bus
  Serial.begin(9600);
  //Serial.println("Iniciou");
  pinMode(pin1, OUTPUT);
  pinMode(pin2, OUTPUT);

  pinMode(ledsema1Red, OUTPUT);
  pinMode(ledsema1Green, OUTPUT);
  pinMode(ledsema2Red, OUTPUT);
  pinMode(ledsema2Green, OUTPUT);
  digitalWrite(ledsema1Red, sema1Red);
  digitalWrite(ledsema1Green, sema1Green);
  digitalWrite(ledsema2Red, sema2Red);
  digitalWrite(ledsema2Green, sema2Green);

  delay(100);
  zeraTags();
}

void loop() {

  if(contg1 > 0){
    contg1 = contg1 + 1;
    }

    if(contg2 > 0){
      contg2 = contg2 + 1;
      }

  t1 = millis();
```

```
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  delay(200);

  if(erro){
    erro = false;
    }else{

  reader1->initialize();
  delay(td);
  pnrd1.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
  ReadError readError = pnrd1.getData();
  if(readError == ReadError::NO_ERROR){
      pnrd1.getTokenVector(data1);
  }else{
    Serial.println("Não leu");
    }
  Serial.print("data1: ");
  Serial.print(data1[0]);
  Serial.print(" ");
  Serial.println(data1[1]);

  reader2->initialize();
  delay(td);
  pnrd2.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
  ReadError readError2 = pnrd2.getData();
  if(readError2 == ReadError::NO_ERROR){
      pnrd2.getTokenVector(data2);
  }else{
    Serial.println("Não leu");

    }
  Serial.print("data2: ");
  Serial.print(data2[0]);
  Serial.print(" ");
  Serial.println(data2[1]);

  if(data1[0] == 0 && data1[1] == 1 && inA == false){
    foundN1 = true;
    }else if(data1[0] == 1 && data1[1] == 1 && outA == false){
      leavingN1 = true;
      }

  if(data2[0] == 0 && data2[1] == 1 && inB == false){
    foundN2 = true;
    }else if(data2[0] == 1 && data2[1] == 1 && outB == false){
      leavingN2 = true;
      }

  atualizaPN();

    Serial.print("Places:");
    Serial.print(sema1Red);
    Serial.print(" ");
    Serial.print(sema1Green);
    Serial.print(" ");
    Serial.print(sema2Red);
    Serial.print(" ");
    Serial.print(sema2Green);
    Serial.print(" ");
```

```
    Serial.print(inA);
    Serial.print(" ");
    Serial.print(inB);
    Serial.print(" ");
    Serial.print(outA);
    Serial.print(" ");
    Serial.print(outB);
    Serial.print(" ");
    Serial.print(leaving);
    Serial.print(" ");
    Serial.print(TLfree);
    Serial.print(" ");
    Serial.println(TLbusy);
  }

  // Salvando
  if(altera1){
    reader1->initialize();
    delay(td);
    Serial.print(data1[0]);
    Serial.print(" ");
    Serial.println(data1[1]);
    pnrd1.setTokenVector(data1);
    if(pnrd1.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag1 salva.");

      Serial.println("Verificando");
      pnrd1.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
      ReadError readError = pnrd1.getData();
      //In case of a successful reading
      if(readError == ReadError::NO_ERROR){
          pnrd1.getTokenVector(datae1);
      }
      Serial.print("data1 erro: ");
      Serial.print(datae1[0]);
      Serial.print(" ");
      Serial.println(datae1[1]);
      if((data1[0] == datae1[0])&&(data1[1] == datae1[1])){
        Serial.println("Blz");
        altera1 = false;
      }else{
        Serial.println("Sem erro sua bunda");
        erro = true;
        }
    }else{
      erro = true;
      Serial.println("Tag1 nao foi salva.");
      }
    }

  if(altera2){
    reader2->initialize();
    delay(td);
    pnrd2.setTokenVector(data2);
    //Serial.println("salvar tag2");
    if(pnrd2.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag2 salva.");
      altera2 = false;
    }else{
      erro = true;
```

```
      Serial.println("Tag2 nao foi salva.");
      }
    }



  digitalWrite(ledsema1Red, sema1Red);
  digitalWrite(ledsema1Green, sema1Green);
  digitalWrite(ledsema2Red, sema2Red);
  digitalWrite(ledsema2Green, sema2Green);

  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, HIGH);

  t2  = millis();
  //Serial.println(t2-t1);
  delay(tmax - (t2-t1));

}

void atualizaPN(){

// lugares
  if(inA && sema1Red && TLfree && !sema2Green){
    N1R2G = true;
    inA = false;
    sema1Red = false;
    TLfree = false;
    data1[0] = 1;
    data1[1] = 0;
    altera1 = true;
  }

  if(sema1Green && outA || contg1 == 3){
    N1G2R = true;
    sema1Green = false;
    outA = false;
    data1[0] = 0;
    data1[1] = 0;
    altera1 = true;
    contg1 = 0;
  }

  if(sema2Red && inB && TLfree && !sema1Green && !inA){
    N2R2G = true;
    sema2Red = false;
    inB = false;
    TLfree = false;
    data2[0] = 1;
    data2[1] = 0;
    altera2 = true;
  }

  if(sema2Green && outB || contg2 == 3){
    N2G2R = true;
    sema2Green = false;
    outB = false;
    data2[0] = 0;
    data2[1] = 0;
    altera2 = true;
```

```
    contg2 = 0;
  }

  if(leaving && TLbusy){
    Busy2Free = true;
    leaving = false;
    TLbusy = false;
    }

// transições
  if(foundN1){
    foundN1 = false;
    inA = true;
    }

  if(foundN2){
    foundN2 = false;
    inB = true;
    }

  if(N1R2G){
    N1R2G = false;
    sema1Green = true;
    TLbusy = true;
    contg1 = 1;
  }

  if(N1G2R){
    N1G2R = false;
    sema1Red = true;
    leaving = true;
  }

  if(N2R2G){
    N2R2G = false;
    sema2Green = true;
    TLbusy = true;
    contg2 = 1;
  }

  if(N2G2R){
    N2G2R = false;
    sema2Red = true;
    leaving = true;
    }

  if(Busy2Free){
    Busy2Free = false;
    TLfree = true;
    }

  if(leavingN1){
    leavingN1 = false;
    outA = true;
    }

  if(leavingN2){
    leavingN2 = false;
    outB = true;
    }
```

```
}


void zeraTags(){
  Serial.println("zera tags");
  digitalWrite(pin1, LOW);
  digitalWrite(pin2, LOW);
  delay(100);
  reader1->initialize();
  delay(td);
  //Setting of the classic iPNRD approach
  pnrd1.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
  Serial.println("Reader1 iniciado");
   pnrd1.setTokenVector(data1);
    if(pnrd1.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag1 salva.");
    }else{
      Serial.println("Tag1 nao foi salva.");
      }



  reader2->initialize();
  delay(td);
  //Setting of the classic iPNRD approach
  pnrd2.setAsTagInformation(PetriNetInformation::TOKEN_VECTOR);
  Serial.println("Reader2 iniciado");
    pnrd2.setTokenVector(data2);
    if(pnrd2.saveData() == WriteError::NO_ERROR){
      Serial.println("Tag2 salva.");
    }else{
      Serial.println("Tag2 nao foi salva.");
      }
  digitalWrite(pin1, HIGH);
  digitalWrite(pin2, HIGH);
  }
```

APÊNDICE C

SIMULAÇÃO COMPUTACIONAL

C.1. Robô móvel

O Quad. C.1 apresenta o código em V-REP para o leitor do robô móvel. A função *sim.setStringSignal* é responsável por enviar ao código do chassi do robô a leitura da etiqueta do feromônio, indicando o caminho a ser seguido.

Quadro C.1 – Código em V-REP para o leitor do robô móvel.

```
function round(x, n)
    return math.floor(x*math.pow(10,n)+0.5)/math.pow(10,n)
end
if (sim_call_type==sim.syscb_init) then
    i=0
    trilha = 0
    data_anterior=" "
    messageToSend=" "
end

if (sim_call_type==sim.syscb_cleanup) then

end

if (sim_call_type==sim.syscb_sensing) then
    if (sim.boolAnd32(i,1)==1) then

sim.sendData(sim.handle_all,0,"Robot",messageToSend,sim.getObjectHandle("Leitor"),0.03,
3.1415*150/180,3.1415*140/180)
```

```
        --sim.addStatusbarMessage(sim.getScriptName(sim.handle_self).." just sent
'"..messageToSend.."'")
    end
    i=i+1
    t=round(sim.getSimulationTime(),1)
    data=sim.receiveData(0,"ponto",sim.getObjectHandle("Leitor"))
    while (data) do
        if data=="trilhaA" and data_anterior ~= data then
            -- TAG FB
            sim.addStatusbarMessage("' received '"..data.."'")
            local fb = 1
            sim.setStringSignal("myTrilha",fb)
            data_anterior=data
        elseif data=="trilhaB" and data_anterior ~= data then
            -- TAG FB
            sim.addStatusbarMessage("' received '"..data.."'")
            local fb = 2
            sim.setStringSignal("myTrilha",fb)
            data_anterior=data
        elseif data=="trilha0" and data_anterior ~= data then
            -- TAG FB
            sim.addStatusbarMessage("' received '"..data.."'")
            n = sim.getRandom()
            if (n > 0.5) then
                local fb = 1
                sim.setStringSignal("myTrilha",fb)
                data_anterior=data
            else
                local fb = 2
                sim.setStringSignal("myTrilha",fb)
                data_anterior=data
            end
        elseif data=="FerA" and data_anterior ~= data then
            --sim.addStatusbarMessage("' received '"..data.."'")
            -- TAG FA
            local fa = 1
```

```
            sim.setStringSignal("myFero",fa)
            data_anterior=data
        elseif data=="tA" and data_anterior ~= data then
            -- TAG SA
            sim.addStatusbarMessage("' received '"..data.."'")
            messageToSend="tA"
            data_anterior=data
        elseif data=="tB" and data_anterior ~= data then
            -- TAG SB
            sim.addStatusbarMessage("' received '"..data.."'")
            messageToSend="tB"
            data_anterior=data
        else
            local fb = 0
            sim.setStringSignal("myTrilha",fb)
            local fa = 0
            sim.setStringSignal("myFero",fa)
            --messageToSend = " "
        end
        data=sim.receiveData(0,"ponto",sim.getObjectHandle("Leitor"))
    end
end
```

O Quad. C.2 apresenta o código em V-REP para o chassi do robô móvel. A função *sim.getStringSignal* é responsável por receber o caminho a ser seguido. O controle de velocidade das rodas do robô é realizado através da função *sim.setJointTargetVelocity.*

Quadro C.2 – Código em V-REP para o robô móvel.

```
function sysCall_init()
    -- Refer also to sim.getCollisionhandle, sim.getDistanceHandle, sim.getIkGroupHandle,
etc.
    leftmotor = sim.getObjectHandle("LeftMotor") -- Handle of the left motor
    rigthmotor = sim.getObjectHandle("RigthMotor") -- Handle of the rigth motor
    lefts = sim.getObjectHandle("leftSensor")
    rigths = sim.getObjectHandle("rigthSensor")
    middles = sim.getObjectHandle("middleSensor")
    -- We take care of setting the desired wheel rotation speed:

minMaxSpeed={20*math.pi/180,30*math.pi/180,40*math.pi/180,50*math.pi/180,60*math.pi/1
80,70*math.pi/180,80*math.pi/180,90*math.pi/180,100*math.pi/180} -- Min and max speeds
for each motor
    speed = minMaxSpeed[5]
    sim.setJointTargetVelocity(leftmotor,speed)
    sim.setJointTargetVelocity(rigthmotor,speed)
    entrou = 0
    entrouN = 0
    linha = true
    a = 0
end

function sysCall_actuation()

    if(sim.getSimulationTime() > 0.2) then

        local fb = sim.getStringSignal("myTrilha")
        if (fb == "1" and entrou == entrouN) then
            speedL = minMaxSpeed[3]
            speedR = minMaxSpeed[7]
            sim.setJointTargetVelocity(leftmotor,speedL)
```

```
        sim.setJointTargetVelocity(rigthmotor,speedR)
        fb = " "
        entrou = entrou + 1
        linha = false
    elseif (fb == "2" and entrou == entrouN) then
        speedL = minMaxSpeed[5]
        speedR = minMaxSpeed[5]
        sim.setJointTargetVelocity(leftmotor,speedL)
        sim.setJointTargetVelocity(rigthmotor,speedR)
        fb = " "
        entrou = entrou + 1
        linha = false
    end

    local fa = sim.getStringSignal("myFero")
    if (fa == "1" and entrou == entrouN + 1) then

        entrouN = entrou
        linha = true
        fa = "0"
    end

    if(linha == true) then
        resultL,dataL = sim.readVisionSensor(lefts)
        resultR,dataR = sim.readVisionSensor(rigths)
        resultM,dataM = sim.readVisionSensor(middles)
        if(dataR[11] > 0.75 and dataM[11] > 0.75 and dataL[11] > 0.75) then
            speedL = minMaxSpeed[5]
            speedR = minMaxSpeed[5]
            sim.setJointTargetVelocity(leftmotor,speed)
            sim.setJointTargetVelocity(rigthmotor,speed)
        elseif(dataR[11] > 0.75 and dataM[11] > 0.75 and dataL[11] > 0.55) then
            speedL = minMaxSpeed[7]
            speedR = minMaxSpeed[3]
            sim.setJointTargetVelocity(leftmotor,speedL)
            sim.setJointTargetVelocity(rigthmotor,speedR)
```

```
            elseif(dataR[11] > 0.75 and dataM[11] > 0.75 and dataL[11] < 0.5) then
                speedL = minMaxSpeed[8]
                speedR = minMaxSpeed[2]
                sim.setJointTargetVelocity(leftmotor,speedL)
                sim.setJointTargetVelocity(rigthmotor,speedR)
            elseif(dataR[11] > 0.75 and dataM[11] < 0.5 and dataL[11] < 0.5) then
                speedL = minMaxSpeed[9]
                speedR = minMaxSpeed[1]
                sim.setJointTargetVelocity(leftmotor,speedL)
                sim.setJointTargetVelocity(rigthmotor,speedR)
            elseif(dataR[11] > 0.55 and dataM[11] > 0.75 and dataL[11] > 0.75) then
                speedL = minMaxSpeed[3]
                speedR = minMaxSpeed[7]
                sim.setJointTargetVelocity(leftmotor,speedL)
                sim.setJointTargetVelocity(rigthmotor,speedR)
            elseif(dataR[11] < 0.5 and dataM[11] > 0.75 and dataL[11] > 0.75) then
                speedL = minMaxSpeed[2]
                speedR = minMaxSpeed[8]
                sim.setJointTargetVelocity(leftmotor,speedL)
                sim.setJointTargetVelocity(rigthmotor,speedR)
            elseif(dataR[11] < 0.5 and dataM[11] < 0.5 and dataL[11] > 0.75) then
                speedL = minMaxSpeed[1]
                speedR = minMaxSpeed[9]
                sim.setJointTargetVelocity(leftmotor,speedL)
                sim.setJointTargetVelocity(rigthmotor,speedR)
            end
        else
            a = a + 1
            if (a == 50) then
                linha = true
                a = 0
            end
        end -- linha = true
    end -- simulacao
end -- function
function sysCall_sensing()
```

```
end

function sysCall_cleanup()
end
```

C.2. Feromônio

O Quad. C.3 apresenta o código em V-REP para o leitor *Tag Feromônio 1*. Este é o algoritmo que realiza todos os cálculos de atualização da quantidade de feromônio em cada uma das trilhas.

Quadro C.3 – Código em V-REP para o leitor *Tag Feromônio 1*.

```
if (sim_call_type==sim.syscb_init) then
    pheA = 100
    pheB = 100
    cpheAi = 100
    cpheBi = 100
    cpheA = 100
    cpheB = 100
    cphemax = 200
    contpheA = 0
    contpheB = 0
    pheAA = 0
    pheBB = 0
    sourceAt = 5
    sourceBt = 300
    sourceA = 0.5
    sourceB = 0.5
    i=0
    k = 0
    j = 0
    transition=0;
    tag1History={"","","","","","","","","",""}
    flag = false
end
```

```
if (sim_call_type==sim.syscb_cleanup) then

end

if (sim_call_type == sim.syscb_actuation) then
    if(sim.getSimulationTime() > 0.2) then
        j = j + 1
        if(j > 0 and flag == false) then
            sourceAt = 300
            flag = true
            sim.addStatusbarMessage("' Atualiza fonte A "')
        end
        pheA = cpheA*math.pow(10, - contpheA/(cpheAi*cpheAi))
        contpheA = contpheA + 0.05
        --sim.addStatusbarMessage("' pheA: "'..pheA..""')
        pheB = cpheB*math.pow(10, - contpheB/(cpheBi*cpheBi))
        contpheB = contpheB + 0.05

    if (pheA/(pheA + pheB) > 0.52) then
        local tri = 1
        pheAA = 1
        pheBB = 0
        sim.setStringSignal("Trilha",tri)
    elseif (pheB/(pheA + pheB) > 0.52) then
        local tri = 2
        pheAA = 0
        pheBB = 1
        sim.setStringSignal("Trilha",tri)
    else
        local tri = 0
        pheAA = 0
        pheBB = 0
        sim.setStringSignal("Trilha",tri)
    end

        k = k + 1
```

```
        if(k == 100) then
            k = 0

            name = "teste.txt"
            file = io.open(name, "a")
            file:write(pheA)
            file:write(" ")
            file:write(pheB)
            file:write(" ")
            file:write(pheAA)
            file:write(" ")
            file:write(pheBB)
            file:write(" ")
            file:write(sourceAt)
            file:write(" ")
            file:write(sourceBt)
            file:write("\n")
            file:close()
            --sim.addStatusbarMessage("salvou")

        end

    end
end

if (sim_call_type==sim.syscb_sensing) then
    if (sim.boolAnd32(i,1)==1) then
        messageToSend="FerA"

sim.sendData(sim.handle_all,0,"ponto",messageToSend,sim.getObjectHandle("AntenaFA"),0
.02,3.1415*120/180,3.1415*140/180)
        --sim.addStatusbarMessage(sim.getScriptName(sim.handle_self).." just sent
'"..messageToSend.."'")
    end
    i=i+1
    data=sim.receiveData(0,"Robot",sim.getObjectHandle("AntenaFA"))
```

```
while (data) do
    if data == "tA" then
        --sim.addStatusbarMessage("' received '"..data.."'")
        if(sourceAt < 10) then
            sourceA = 0
            --pheA = 0.96*pheA
        else
            sourceA = 0.5
        end
        sourceAt = sourceAt - sourceA
        cpheA = pheA + sourceA

        if (cpheA > cphemax) then
            cpheA = cphemax
        end
        contpheA = 0
        sim.addStatusbarMessage("' pheA: '"..pheA.."'")
        data_anterior = data
    elseif data == "tB" then
        --sim.addStatusbarMessage("' received '"..data.."'")

        if(sourceBt < 10) then
            sourceB = 0
            --pheB = 0.96*pheB
        else
            sourceB = 0.5
        end
        sourceBt = sourceBt - sourceB
        cpheB = pheB + sourceB

        if (cpheB > cphemax) then
            cpheB = cphemax
        end
        contpheB = 0
        sim.addStatusbarMessage("' pheB: '"..pheB.."'")
        data_anterior = data
```

```
        else


        end
    data=sim.receiveData(0,"ponto",sim.getObjectHandle("Leitor"))
    end
    data_anterior = " "
end
```

O Quad. C.4 apresenta o código em V-REP para o leitor *Tag Feromônio 2*. Este é o algoritmo que indica ao robô qual o caminho a ser seguido.

Quadro C.4 – Código em V-REP para o leitor *Tag Feromônio 2*.

```
if (sim_call_type==sim.syscb_init) then
    i=0
    k = 0
    messageToSend = " "
    flag = false
end

if (sim_call_type==sim.syscb_cleanup) then
end

if (sim_call_type==sim.syscb_sensing) then
    if (sim.boolAnd32(i,1)==1) then


        local tri = sim.getStringSignal("Trilha")
        if (tri == "1") then
            messageToSend=  "trilhaA"
        elseif (tri == "2") then
            messageToSend = "trilhaB"
        elseif (tri == "0" and flag == false) then
            messageToSend = "trilha0"
        end
        if (flag == true) then
            k = k + 1
```

```
            if(k == 100) then
                flag = false
            end
        end
        -- sim.addStatusbarMessage("' i '"..i.."'")

sim.sendData(sim.handle_all,0,"ponto",messageToSend,sim.getObjectHandle("AntenaFB"),0
.025,3.1415*120/180,3.1415*140/180)
    end
    i=i+1
end
```

O Quad. C.5 apresenta o código em V-REP para o leitor *Tag A*. Este é o algoritmo que indica ao robô que ele seguiu a trilha A.

Quadro C.5 – Código em V-REP para o leitor *Tag A*.

```
if (sim_call_type==sim.syscb_init) then
    i=0
end

if (sim_call_type==sim.syscb_cleanup) then
end

if (sim_call_type==sim.syscb_sensing) then
    if (sim.boolAnd32(i,1)==1) then
        messageToSend = "tA"

sim.sendData(sim.handle_all,0,"ponto",messageToSend,sim.getObjectHandle("AntenaSA"),
0.025,3.1415,2*3.1415)
    end
    i=i+1
end
```

O Quad. C.6 apresenta o código em V-REP para o leitor *Tag B*. Este é o algoritmo que indica ao robô que ele seguiu a trilha B.

Quadro C.6 – Código em V-REP para o leitor *Tag B*.

```
if (sim_call_type==sim.syscb_init) then
    i=0
end

if (sim_call_type==sim.syscb_cleanup) then
end

if (sim_call_type==sim.syscb_sensing) then
    if (sim.boolAnd32(i,1)==1) then
        messageToSend="tB"

sim.sendData(sim.handle_all,0,"ponto",messageToSend,sim.getObjectHandle("AntenaSB"),
0.025,3.1415*120/180,3.1415*140/180)
    end
    i=i+1
end
```