MIGUEL NAKAJIMA MARQUES

SISTEMA DE CONTROLE DE NAVEGAÇÃO E COMUNICAÇÃO PARA UM ROBÔ MÓVEL AUTÔNOMO BASEADO NA TECNOLOGIA RASPBERRY PI

Trabalho de Conclusão de Curso apresentado
à Escola de Engenharia de São Carlos, da
Universidade de São Paulo

Curso de Engenharia Elétrica com ênfase em
Sistemas de Energia e Automação

ORIENTADOR: Eduardo do Valle Simões

São Carlos
2014

Marques, Miguel
M357s Sistema de controle de navegação e comunicação para um robô móvel autônomo baseado na tecnologia Raspberry Pi / Miguel Marques; orientador Eduardo Simões. São Carlos, 2014.

Monografia (Graduação em Engenharia Elétrica com ênfase em Sistemas de Energia e Automação) -- Escola de Engenharia de São Carlos da Universidade de São Paulo, 2014.

1. Raspberry Pi. 2. inteligência artificial. 3. robótica. I. Título.

FOLHA DE APROVAÇÃO

Nome: Miguel Nakajima Marques

Título: "Sistema de controle de navegação e comunicação para um robô móvel autônomo baseado na tecnologia Raspberry PI"

Trabalho de Conclusão de Curso defendido e aprovado em 25/06/2014,

com NOTA 8,0 (oito, zero), pela Comissão Julgadora:

Prof. Dr. Eduardo do Valle Simões - (Orientador - SSC/ICMC/USP)

Prof. Associado Adilson Gonzaga - (SEL/EESC/USP)

Prof. Associado Ivan Nunes da Silva - (SEL/EESC/USP)

Coordenador da CoC-Engenharia Elétrica - EESC/USP:
Prof. Associado Homero Schiabel

Sumário

Resumo

O objetivo deste trabalho é implementar um sistema de navegação, decisão e cumprimento de missões em robôs autônomos móveis de pequeno porte. A movimentação do robô será executada por um par de motores que acionam duas rodas. Os motores serão acionados por um *driver* de corrente controlado por um sinal PWM gerado pela placa Raspberry Pi, responsável pelo controle de locomoção, controlando a velocidade e direção do robô, também recebendo o sinal de pares de sensores/emissores de infravermelho localizados em várias partes do chassi do robô e também responsável pela navegação, utilizando a leitura dos sensores para encontrar a manobra adequada a fim de evitar colisões com obstáculos. Para viabilizar uma navegação adaptativa em ambientes dinâmicos e ruidosos, como salas fechadas com presença de pessoas, será avaliada a utilização de algoritmos evolutivos.

Palavras-chave: Raspberry Pi, Robótica Móvel, Controle Embarcado, Algoritmos Evolutivos.

Abstract

The goal of this work is to implement a system of navigation, decision making and mission accomplishment in small scale autonomous mobile robots. Its mobility will be executed by a pair of DC motors connected to wheels. The motors will be driven by a current driver controlled by a PWM signal generated by the Raspberry Pi board which will be responsible for the speed and direction of the robot. The board will also receive signals from infrared sensor/emitter pairs located in various parts of the robot chassis, using those signals to find the maneuver that is best suited for the situation, aiming to avoid obstacle collisions. To make the navigation viable in dynamic and noisy environments, such as rooms with the presence of people, a solution using evolutionary algorithms will be evaluated.

Keywords: Raspberry Pi, Mobile Robotics, Embedded Control, Evolutionary Algorithms.

1. Introdução

No cenário globalizado atual, no qual os baixos custos de transporte permitem a uma fábrica no interior da China colocar seu produto à venda ao lado de um equivalente americano em uma prateleira de supermercado no Brasil, a redução dos custos de produção torna-se um fator determinante para a sobrevivência mercadológica (KON, 1997). Sob esse pressuposto, a automação dos processos industriais é item importante na otimização do processo fabril, visando não somente à redução dos custos mas também à melhoria da qualidade do produto final. A implantação de robôs nas linhas de montagem é uma tendência mundial, na qual o diferencial decisivo entre dois processos concorrentes não está no acesso à tecnologia de automação (amplamente disponível em âmbito mundial a preços acessíveis) mas no sistema de controle e decisão melhor adaptado às necessidades do processo.

Sistemas inteligentes multirrobôs para realização de tarefas têm se mostrado uma solução ao mesmo tempo robusta e eficaz e, com a redução dos custos de *hardware*, têm sido aplicada em um número maior de casos de otimização de uma planta fabril. Esse método se mostra vantajoso devido principalmente à robustez atingida por meio da redundância dentro de um enxame de robôs idênticos, no qual um indivíduo tem as mesmas capacidades de qualquer outro, podendo a substituição ser feita de forma transparente e sem perda de capacidade do todo (ROMANO e DUTRA, 2002).

Uma das desvantagens dessa técnica é a necessidade de generalidade dos indivíduos, o que leva ao aumento da complexidade de cada robô. Essa característica porém, é mitigada com a popularização e massificação da produção de microcontroladores e microprocessadores de alto poder de processamento, o que gerou soluções como o PIC, ATMega e outros da mesma linha (GONÇALVES, 2007).

Com o aumento da complexidade dos processos industriais, a tecnologia de automação também se torna cada vez mais custosa do ponto de vista de processamento, o que requer processadores cada vez mais potentes e com maior possibilidade de interfaceamento com o ambiente e os usuários humanos. Além dessa limitação, há também a necessidade de processamento imposta pelos algoritmos de inteligência artificial, como RNA (Rede Neural Artificial) e AE (Algoritmos Evolutivos), utilizados na navegação, controle e tomada de decisão, necessária em ambientes multiobjetivo que se tornam cada vez mais abundantes no cenário

industrial (JUNG, 2005). Assim sendo, torna-se imperativo o uso de processadores com alto poder computacional, que não se enquadram na categoria de baixo custo (RABELO, 2002).

Nesse âmbito, os computadores de placa única (do inglês *single-board computer*) apresentam uma solução intermediária com capacidade computacional considerável e um custo acessível para aquisição em grande número. Esses computadores facilitam a integração de interfaces homem-máquina como teclado e monitor, além de conter nativamente padrões de conectividade amplamente difundidos no mundo computacional, como interface de rede por cabo e wireless e protocolos de comunicação padrão entre sistemas (List of single-board computers, 2014).

A placa Raspberry Pi foi idealizada objetivando auxiliar o ensino de computação e programação nas escolas, mas, por ser um hardware de baixo custo, com dimensões pequenas - 8,6 cm de comprimento por 5,4 cm de largura - baixo consumo de energia - aproximadamente 3,5 W - e possuir todos os componentes soldados na mesma placa, se tornou amplamente utilizado (About us | Raspberry Pi, 2012). Possui um processador ARM de 700 MHz, 512 MB de memória RAM, além de uma GPU, construído com a tecnologia *system on a chip (SoC)*, ou seja, traz CPU, RAM e GPU em um mesmo circuito integrado, o que maximiza o desempenho e minimiza o consumo. O armazenamento não volátil é feito em um cartão SD, e possui portas USB, Ethernet e HDMI, entre outras, o que facilita a integração com outros hardwares. Essa placa é um computador de baixo custo baseado em um processador RISC, que possui grande respaldo da comunidade envolvida com software livre, o que gera diversas opções de sistema operacional. Para esse trabalho será utilizado o SO de uso geral Raspian (Raspian About, 2014).

Como a Raspberry Pi possui o sistema Linux embarcado, fica transparente a programação de baixo nível durante a utilização de algoritmos complexos. Esse sistema também facilita a utilização de interfaces nativas e provenientes de terceiros, como o módulo Wi-Fi conectado a uma das portas USB e um teclado e monitor utilizados para a programação e manutenção dos robôs (RPi Hub, 2014).

Em comparação com soluções de menor poder computacional, como microcontroladores comerciais, as possibilidades de interfaceamento com *hardware* de baixo nível, como barramentos I^2C, são muito mais limitadas na Raspberry Pi, que tem, nativamente, somente uma porta serial, uma I^2C e uma PWM (Pulse-Width Modulation). Essas interfaces podem facilmente ser encontradas em *hardware* mais simples, assim como portas AD, que não estão disponíveis na placa utilizada.

A fim de contornar essa limitação, podem ser utilizados periféricos auxiliares, como chips AD dedicados, geradores de PWM (TOWNSEND, 2014) e multiplexadores de portas de comunicação. Isso aumenta o custo do projeto e a complexidade dos indivíduos, além de ser um limitante no caso da necessidade de miniaturização do *hardware*.

Uma opção de solução para a limitação de portas PWM menos dependente de *hardware* externo é a implementação de PWM por *software* (HENDERSON, 2014), que pode ser atingida utilizando threads da linguagem C. Essa técnica, porém, quando implementada na Raspberry Pi, apresenta a limitação de uma contagem de tempo imprecisa, já que essa placa não possui um RTC (Real-Time Clock), o que, para a solução em questão, resulta em um ciclo de trabalho impreciso,podendo levar a instabilidades no controle de velocidade de motores de corrente contínua em determinadas aplicações.

Do ponto de vista de controle, a técnica implementada deve ser capaz de guiar o robô nas mais diversas situações. Isso pode ser atingido utilizando sensores e atuadores de forma reativa, deliberativa ou híbrida (WOLF, SIMÕES, et al., 2009). A primeira opção interpreta instantaneamente os sinais gerados pelos sensores, usando o resultado para controlar os atuadores. A opção deliberativa utiliza padrões pré-estabelecidos, comparando-os com a situação apresentada pelos sensores e adotando o padrão de decisão que mais se aproxima do cenário apresentado. Enquanto o controle reativo apresenta como vantagem uma rapidez muito maior no processo decisório, o deliberativo se adequa muito melhor a ambientes controlados, nos quais os padrões de ação devem seguir linhas rígidas.

O presente trabalho baseia-se nos estudos com a placa Raspberry Pi para robótica, apresentados por Grana (2013) para a partir de suas conclusões traçar uma linha de estudo mais voltada para o sistema decisório e de cumprimento de missões para sistemas multirrobóticos.

Este projeto deve prover um sistema de controle e decisão para robôs móveis embarcados em computadores de placa única de baixo custo. Isso possibilita a utilização dessa solução em situações dinâmicas diversas, como chão de fábrica, reconhecimento e mapeamento de ambientes com e sem a presença de pessoas e auxílio em funções domésticas.

1.1 Objetivo

O objetivo deste trabalho é implementar um sistema de navegação, decisão e cumprimento de missões em robôs autônomos móveis de pequeno porte utilizando a placa Raspberry Pi para uso em transporte de peças de uma linha de montagem em um chão de fábrica.

A movimentação do robô será executada por um par de motores de corrente contínua que acionam duas rodas localizadas na lateral do chassi. Os motores serão acionados por um *driver* de corrente controlado por um sinal PWM gerado pela placa Raspberry Pi, responsável pelo controle de locomoção, controlando a velocidade e direção do robô.

Os pinos digitais 1 a 3 da placa ainda receberão o sinal de sensores de infravermelho localizados em várias partes do chassi do robô, utilizando-os para desviar de obstáculos e navegar pelo ambiente.

Para o acionamento dos motores será utilizado um driver comercial L298E que deverá adequar o sinal de PWM vindo da Raspberry Pi para a potência fornecida por quatro pilhas de íon de lítio associadas em série.

A fim de fazer a comunicação com outros dispositivos por meio de uma WLAN, será utilizado uma placa de rede USB com o padrão 802.11. Com isso será possível fazer o monitoramento da situação dos robôs, além de possibilitar que os robôs acessem dados em um servidor remoto para compartilhar informações como descrito pelo website *StackOverflow* (Opening a file in a remote location, 2014).

Para viabilizar uma navegação adaptativa em ambientes dinâmicos e ruidosos, como ambientes com presença de pessoas, será avaliada a utilização de algoritmos evolutivos.

Com todos os elementos apresentados nesse capítulo, tem-se o objetivo de viabilizar a utilização do enxame de robôs autônomos que serão capazes de cumprir missões de transporte da melhor maneira possível, sem colidir com obstáculos e se adaptando a diversas situações do ambiente que possam surgir no decorrer da operação.

1.2 Organização do documento

No capítulo 2, “Desenvolvimento de Hardware e Software embarcado para robôs móveis”, será feito o estudo de cada um dos módulos que compõe o projeto, detalhando as ferramentas utilizadas e seu funcionamento teórico.

A solução implementada e a maneira como essa implementação foi feita estão apresentados no capítulo 3, “Implementação do sistema de controle de navegação”.

A integração dos módulos e os resultados dos testes executados com cada módulo de forma independente e integrada à solução final está no capítulo 4, “Resultados”.

Finalmente, no capítulo 5, “Conclusões”, há uma breve dissertação sobre o resultado do trabalho, assim como as perspectivas futuras e indicações de linhas de trabalho a serem seguidas no futuro.

2. Módulos de Hardware e Software embarcado para robôs móveis

Neste capítulo serão apresentados os temas estudados para a realização do projeto. Será apresentado um breve histórico e as características da placa utilizada, os princípios de funcionamento dos sensores escolhidos para interfaceamento entre o ambiente virtual e o mundo real e conceitos sobre modulação por largura de pulso (PWM). Também serão apresentados o buffer e o driver utilizados para o acoplamento entre a placa de controle e os motores. São avaliados ainda a rede neural e o algoritmo evolutivo implementados na solução.

2.1 Raspberry Pi e Linux embarcado

O computador de placa única Raspberry Pi teve seu lançamento em 2012. Seus criadores tinham como objetivo inicial estimular o ensino de ciência da computação em escolas, buscando recuperar o interesse dos jovens pelo estudo de computação básica. (About us | Raspberry Pi, 2012) A fim de aumentar o escopo de uso e incentivar o estudo de funções de baixo nível, foi mantido o baixo custo desse hardware. Isso causou grande interesse da comunidade de desenvolvedores, o que fez a Raspberry Pi ser rapidamente popularizada, causando uma fila de espera nos primeiros meses de vendas.

Como informado pelo site oficial da fabricante (FAQ | Raspberry Pi, 2013), esse pequeno computador porta um system on a chip (SoC) Broadcom BCM2835, que inclui um processador ARM 11 de 700MHz, GPU dedicada, e 512MB de memória RAM. Mas o que realmente atraiu interesse da comunidade de Sistemas da Informação em geral foi a existência de uma saída HDMI, conexão Ethernet, duas portas USB, entre outras, o que implica em um grande número de periféricos possíveis.

A Figura 2.1 mostra uma placa Raspberry Pi versão B.

Figura 2.1. A placa Raspberry Pi versão B (raspberrypi.org)

Foi essa relação de custo/benefício que levou à popularização dessa placa. Rapidamente surgiram inúmeras aplicações e projetos, o que contribuiu para o surgimento de uma infinidade de material de referência para outros projetos e ampliou o suporte oferecido pela comunidade, principalmente a envolvida com *software* livre.

Apesar de não ser vinculado ao fabricante da placa, o sistema operacional que vem sendo mais utilizado é o Raspbian, uma variante não oficial do *Debian Wheezy armhf*, otimizada para o conjunto de instruções ARMv6 presente na Raspberry Pi (Rpi Distributions – eLinux.org, 2013). Essa distribuição oferece mais de 35.000 pacotes pré-compilados, abrindo um grande leque de aplicações possíveis para a Raspberry Pi, desde centrais de mídia até controle de *hardware* (Raspbian About, 2012). Não é o mais suscinto possível para a aplicação em um projeto específico, porém possui instalação facilitada e grande suporte da comunidade, levando a uma frequência de atualizações e correções de bugs maior e suporte a possíveis obstáculos na utilização desse SO.

Por ter essas características, a Raspberry Pi foi escolhida para a implementação do sistema de navegação robótico deste projeto, mais especificamente, por ser um processador de baixo custo e possuir capacidade de processamento e memória suficientes para executar algoritmos de navegação, planejamento e tomada de decisão mais complexos do que usualmente se obtém com microcontroladores como Arduino, PIC, MSP430, entre outros.

2.2. Sensor de distância por infravermelho

Como descrito por Lee e Chong (2011), é possível utilizar um par contendo um LED emissor e um receptor de luz infravermelha para a detecção básica de obstáculos em robôs móveis. Essa solução será adotada no presente trabalho para detecção de objetos no trajeto do robô.

O circuito simples da Figura 2.2 funciona para validar o sistema de detecção de objetos, onde a luz infravermelha gerada pelo LED emissor é refletida pelo obstáculo e atinge o fotodiodo receptor que entra na zona de condução, diminuindo sua impedância, o que faz com que a tensão na entrada inversora do amplificador operacional se aproxime de zero, ficando abaixo da tensão ajustada na porta não inversora através do trimpot, fazendo com que a saída do amplificador operacional vá de –Vcc (0V) para +Vcc(5V)

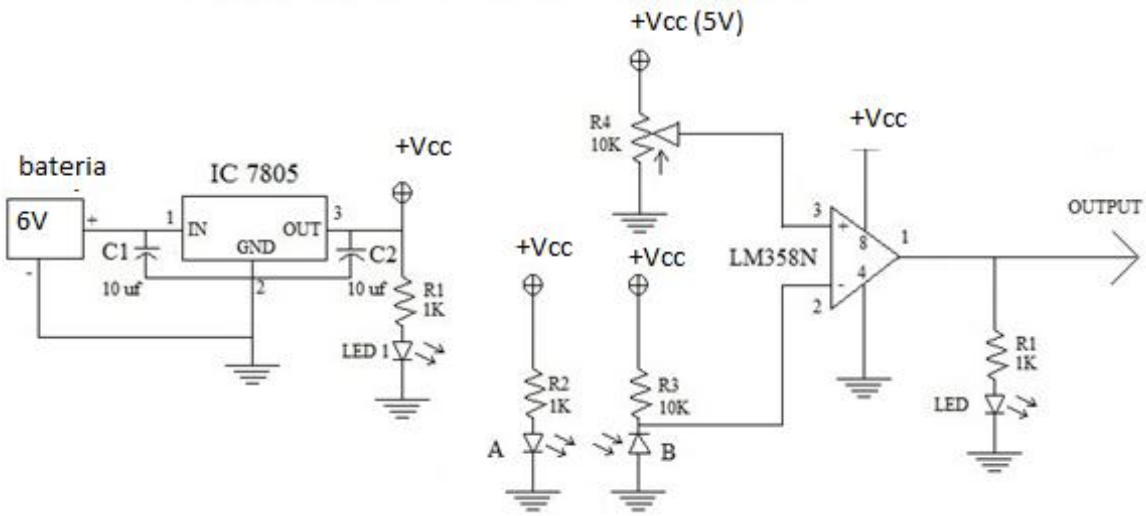

Figura 2.2. Circuito de validação dos sensores infravermelhos (roboticsbible.com).

Com isso, toda vez que um objeto se aproxima do sensor, um sinal é emitido na saída do circuito e será usado no processo de navegação do robô. O par receptor/emissor montado pode ser visto na Figura 2.3.

(a) **(b)**

Figura 2.3. Par emissor/receptor utilizado no circuito

Pode-se notar que na Figura 2.3.(b) o LED emissor está um pouco à frente do receptor. Essa configuração é importante para evitar que a iluminação do emissor interfira com a leitura de reflexão do receptor.

2.3. PWM

O controle das velocidades dos motores foi feito com um chaveamento em frequência constante, uma vez que se dispunha apenas de saídas digitais. Para isso foi utilizado o conceito de *Pulse-Width Modulation*. Assim, com uma onda quadrada com frequência constante e razão cíclica (*duty cycle*) ajustável é possível transferir uma quantidade de potência desejável através do valor médio de tensão do sinal (AHMED, 2000).

Segundo Ahmed (2000), é dado que a tensão média de saída V_0 é

$$V_0 = \frac{T_{ON} * Vi}{T} \quad (2.1)$$

Sendo V_0 a tensão média de saída, T_{ON} o período em segundos que o sinal fica em nível alto, T o período total do sinal e V_i a tensão do nível alto.

A potência de saída pode ser descrita por

$$P = V_0 * I_0 \quad (2.2)$$

Sendo P a potência de saída, V_0 a tensão de saída e I_0 a corrente de saída.

Portanto, a partir de (2.1) pode-se afirmar que:

$$V_0 = V_I * d \quad (2.3)$$

Onde:

$$d = \frac{T_{ON}}{T} \quad (2.4)$$

Sendo d chamado ciclo de trabalho ou, no inglês, *duty* cycle. Dessa relação e a partir da Lei de Ohm, tem-se:

$$I_0 = \frac{d*V_I}{R} \quad (2.5)$$

Assim, a potência de saída é dada por

$$P_O = \frac{(D*V_I)^2}{R} \quad (2.6)$$

Considerando uma carga totalmente resistiva, pode-se controlar a potência entregue de maneira proporcional ao quadrado da largura de pulso configurada no periférico.

É importante observar a frequência de trabalho para respeitar os limites do *hardware* de potência utilizado e em casos em que há carga indutiva, como em motores, é necessário pulsar em uma frequência que faça a corrente se manter estável em uma mesma largura de pulso, fazendo o motor girar suavemente, sem trancos ou ruído audível na frequência do pulso modulado em largura.

2.4. Buffer de acoplamento dos sensores e driver dos motores

O acoplamento dos comandos da Raspberry Pi com o módulo de potência do robô é feito utilizando opto acopladores, compostos de LEDs e foto transistores integrados em um encapsulamento, permitindo o acionamento desacoplado entre as partes (FAIRCHILD, 2010). Porém, para acionar o LED de maneira que o foto transistor entre em região de saturação, o que é desejável quando se está trabalhando de forma digital, é necessária uma corrente de 20mA (D217 Datasheet, 2013). Os pinos de *i/o* do microprocessador da Raspberry Pi não tem capacidade de fornecer essa magnitude de corrente (About us | Raspberry Pi, 2012), então foram utilizados buffers para os sinais transmitidos para a parte de potência.

A configuração de buffer utilizada nesse trabalho foi feita com um amplificador operacional, coma na Figura 2.4 naqual é obtida uma impedância de entrada muito baixa, além de fornecer a corrente necessária para o acionamento do LED, por exemplo o caso citado em Clayton e Winder (2003). A corrente de saída fica limitada apenas pela corrente máxima de saída do amplificador operacional, valor que é encontrado na folha de dados do componente. Além dessa situação em que a fonte não consegue fornecer a corrente necessária para a aplicação, uma impedância de entrada alta é utilizada também quando a tensão cai com uma carga alta, como o caso de um divisor resistivo, que sofre alteração na tensão quando uma corrente grande é exigida para a leitura. Dessa forma, a utilização de *buffers* é uma solução que exige um número reduzido de componentes para resolver problemas como os expostos acima. Uma alternativa de menor custo é obtida utilizando um transistor em região de saturação, porém por disponibilidade de componentes, foi utilizada a solução com amplificador operacional.

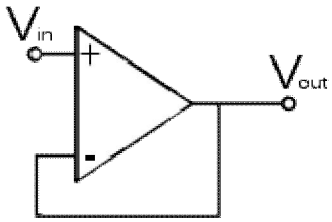

Figura 2.4. Configuração de amplificador operacional seguidor de tensão utilizado como buffer

2.5 Driver

Para que se possa controlar a velocidade e direção dos motores, é necessário que uma corrente possa fluir em ambas as direções dentro de sua bobina, gerando campos com intensidade e sentidos diferentes. A configuração utilizada para controlar a corrente nesse projeto é um driver em ponte, com configuração de ponte H, como é ser observado na Figura 2.5 (INOUE e OSUKA, 2004).

São apresentados aqui transistores de junção bipolar, porém também há a opção de utilizar MOSFETs ou mesmo chaves mecânicas como Relés. O que difere cada chave é o princípio de funcionamento e o tipo de acionamento, que possuem particularidades, principalmente quando são utilizados transistores de efeito de campo e é necessária uma referência entre Porta e Fonte para acioná-los, apresentando um circuito específico para as chaves de cima da ponte H.

Chavear uma corrente elevada com uma ponte H exige um cuidado especial para evitar que ocorra um curto-circuito em um dos braços da ponte, o que ocorreria se os dois transistores da esquerda, na Figura 2.6, fossem acionados simultaneamente. É utilizado então um circuito lógico em cada um dos braços da ponte para evitar o estado de curto-circuito (STMICROELECTRONICS, 2000). O circuito apresentado na folha de dados do driver utilizado no projeto está reproduzido na Figura 2.6

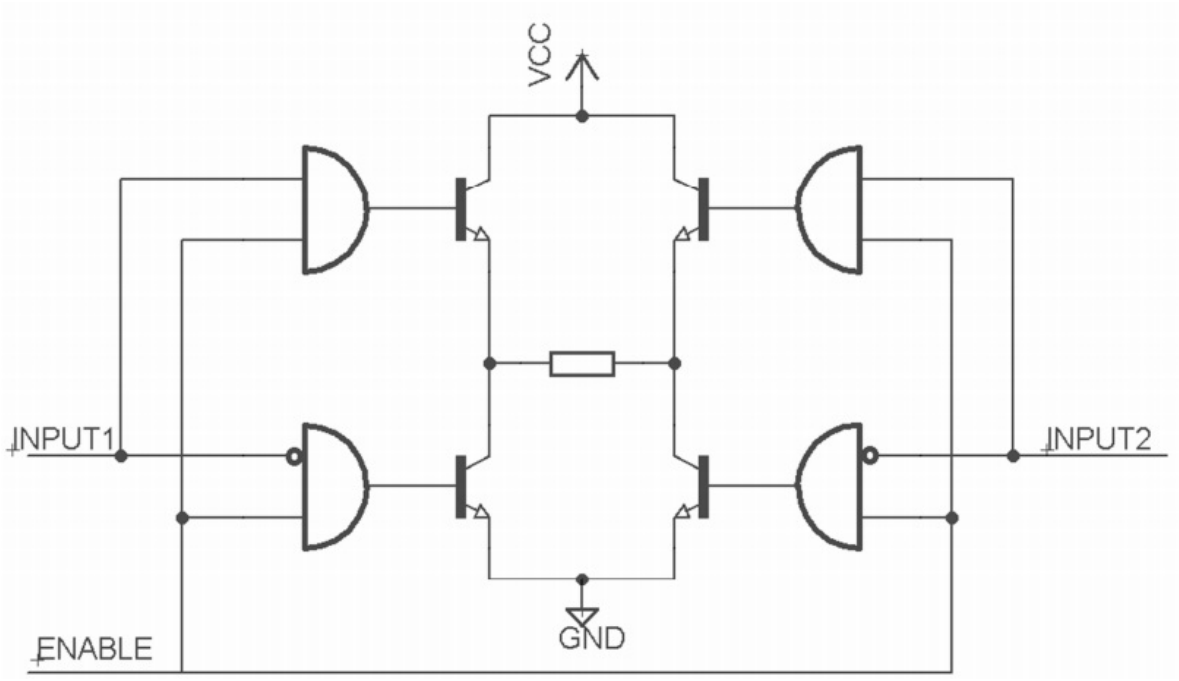

Figura 2.5. Ponte H com circuito lógico para prevenir ativação que leve ao curto circuito (INOUE e OSUKA, 2004)

Nesse caso, o controle de cada lado da ponte funciona de forma binária, não sendo permitido o curto-circuito, além de possibilitar liberdade para acionar os dois transistores inferiores ou superiores simultaneamente, o que pode ser desejado quando se deseja liberar o fluxo de corrente armazenado nas bobinas de um motor.

Outra técnica utilizada é a ligação de diodos de retorno em paralelo com os transistores e com polarização reversa, para que uma corrente induzida no motor possa fluir mesmo quando ambos os lados da ponte estiverem no estado ligado, ou seja, bloqueando a passagem de

corrente. A utilização desses diodos também evita que uma corrente recirculante no motor danifique um transistor em estado de corte (BARBI, 2006).

Na Figura 2.6, Ponte H com diodos de roda livre, pode-se ver a ponte com os diodos de retorno. Quando os transistores Q1 e Q4 estão acionados, a corrente flui de A para B, pois o potencial em A é maior; quando ambos estão cortados, ainda existe uma diferença de potencial nos terminais da carga; essa diferença de potencial faz com que a tensão em A seja menor do que zero e em B maior do que VCC, fazendo os diodos D2 e D3 conduzirem por um curto período de tempo suficiente para dissipar essa corrente residual, pois após a corrente ser dissipada a tensão de A volta a ser maior do que terra e B menor que VCC.

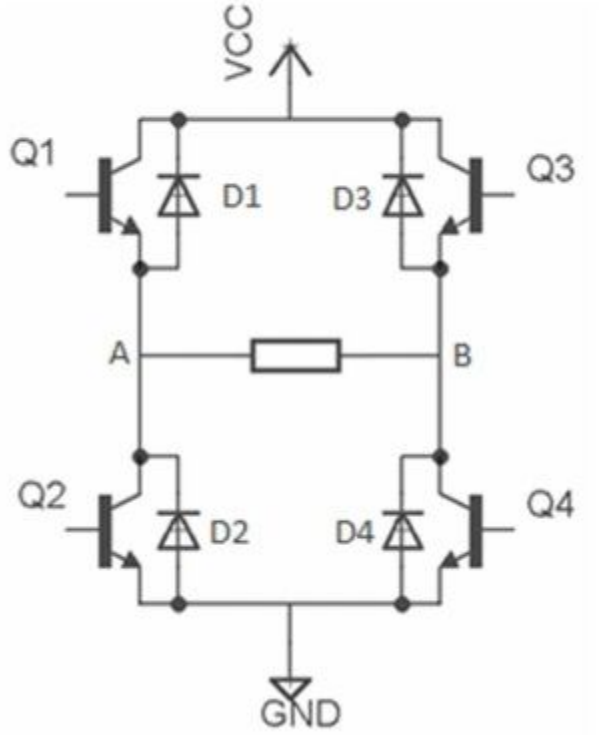

Figura 2.6. Ponte H com diodos de roda livre (INOUE e OSUKA, 2004)

2.6. Acesso remoto por meio do programa PuTTY

Uma das vantagens que a Raspberry Pi tem, por ter sido idealizada visando ao incentivo ao ensino de ciências da computação, é o suporte de programas e bibliotecas desenvolvidos especificamente para tornar esse *hardware* mais amigável aos usuários que não estão acostumados ao ambiente de programação embarcada ou o sistema Linux. Um desses é o

Samba, um programa para sistemas baseados em Unix, que simula um servidor Windows, o que permite o acesso remoto de dados por meio de uma rede padrão Microsoft. Um exemplo de como um computador Windows enxerga a Raspberry Pi depois de configurado o Samba é mostrado na figura 2.7. A instalação e configuração do Samba na Raspberry Pi podem ser encontradas em inúmeros tutoriais na Internet (Share your Raspberry Pi's file and folder across a network,s/d; Setting up a SAMBA Server on Raspberry Pi, s/d).

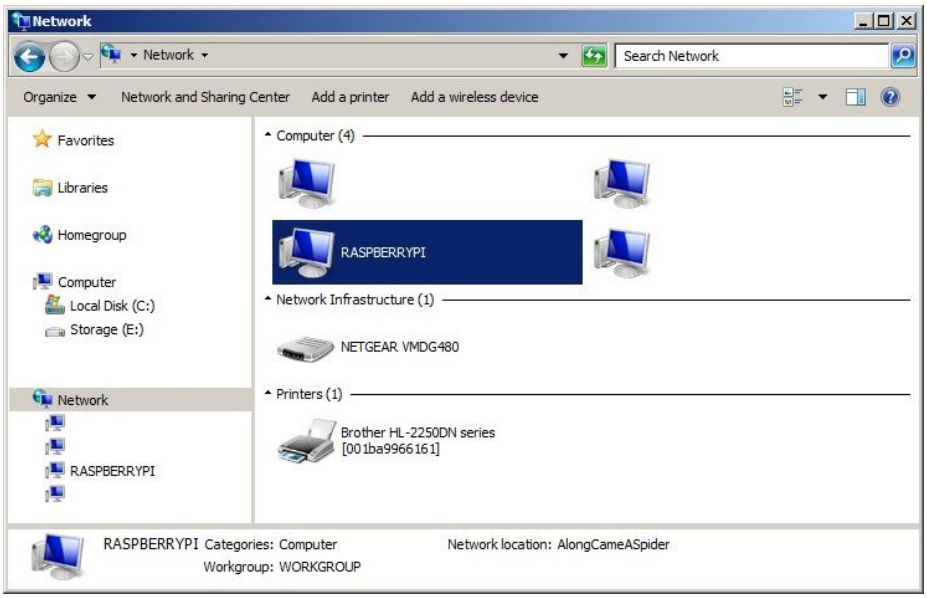

Figura 2.7. A Raspberry Pi como vista por um computador em uma rede Windows

Outra ferramenta utilizada para o acesso remoto ao robô é o programa PuTTY, um *software* livre para emulação de terminal que suporta protocolo SSH. Por meio dele é possível acessar o *prompt* de comando do sistema Linux da Raspberry Pi a partir de qualquer computador conectado à mesma rede. Apesar de não oferecer suporte para interfaces gráficas, não sendo, portanto, capaz de interpretar o X Window System, a interface gráfica do sistema Linux, o PuTTY será usado para enviar comandos para a Raspberry Pi. Alguns exemplos de tarefas possíveis de serem executadas por meio do PuTTY são: compilação de programa, edição de arquivos de configuração, interação com programas executados no robô.

Assim, com o uso em conjunto dessas duas ferramentas é possível acessar o cartão de memória da Raspberry Pi, editar o código fonte de um programa lá armazenado e, por meio de comandos SSH, compilá-lo e interagir com o programa em execução.

2.7 Algoritmos Evolutivos

Algoritmos Evolutivos são baseados em mecanismos inspirados na evolução biológica das espécies (HOLLAND e GOLDBERG, 1988). A motivação para a utilização dessas técnicas computacionais surgiu da observação da natureza, que resolveu problemas complexos utilizando tais recursos, ou seja, tenta-se construir de maneira computacional situações que possam se comparar com mecanismos que conhecidamente influenciam na evolução. Uma comparação entre as técnicas computacionais e a evolução ocorrida na natureza é mais explorada em Eiben e Smith (2003).

Aplica-se então técnicas evolutivas a problemas de otimização, mesmo quando o modelo do problema não possa ser definido completamente. Basta que se tenha uma resposta que atende as necessidades da requisição, como é o caso de sistemas robóticos em ambientes dinâmicos, nos quais o número de variáveis que influenciam no resultado final é muito grande, tornando o problema demasiadamente complexo para ser modelado por meios tradicionais.

As possíveis soluções são então tratadas como indivíduos, e seus conjuntos como populações, que interagem buscando a evolução através de operadores genéticos. A literatura clássica (GOLDBERG, 1989) traz o mecanismo padrão de evolução dividido nas etapas representadas no fluxograma da Figura 2.8.

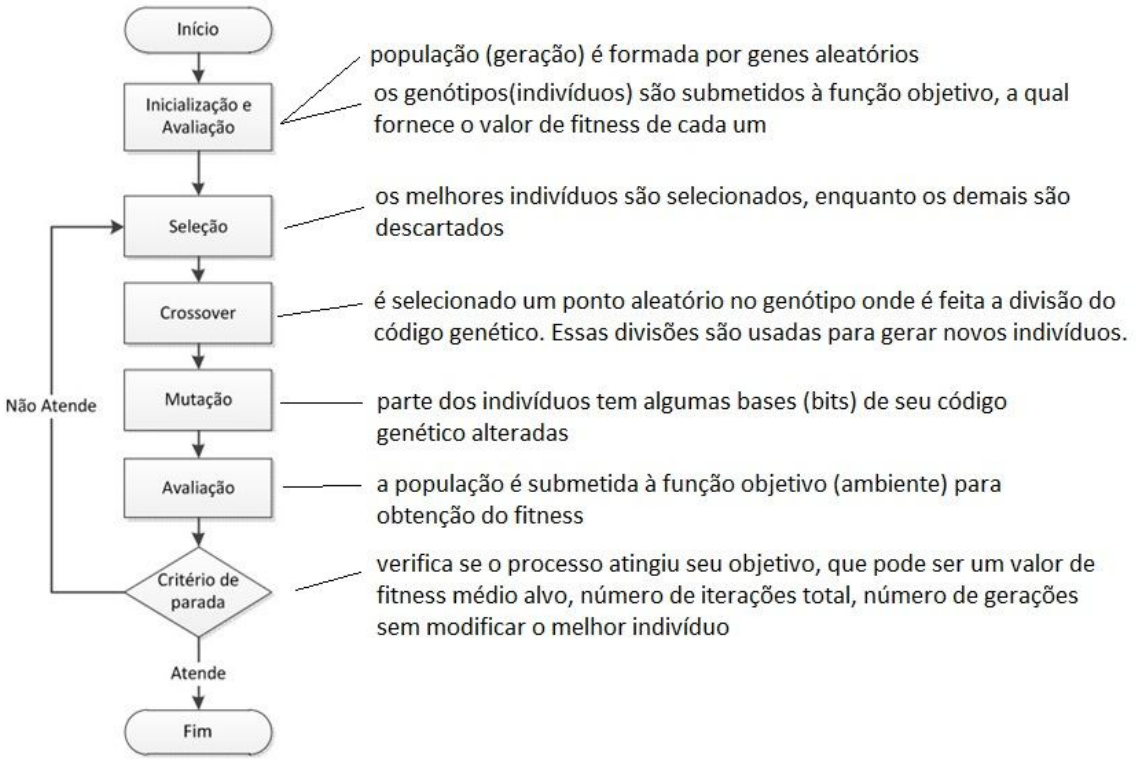

Figura 2.8. Fluxograma do algoritmo evolutivo

Porém, em robôs reais, a função *fitness* possui um ruído, pois a pontuação obtida na avaliação depende da leitura de sensores e pode sofrer interferência de agentes externos, como escorregamento e irregularidade do terreno. Com essas variações não é possível garantir que o indivíduo avaliado possuirá sempre o mesmo *fitness*, sendo necessário continuar a evolução para cada nova situação encontrada pelo robô.

2.8. Algoritmo A*

O A* (lê-se “A estrela”) é um algoritmo de busca de caminho que almeja a obtenção do melhor trajeto entre dois pontos. De modo generalizado, o algoritmo faz uma busca nos possíveis caminhos entre origem e destino, calculando através do método de busca primeiro-melhor (*best-first search)* o trajeto com o menor custo total. (ZHAN e NOON, 1998)

Para a obtenção do melhor caminho, o algoritmo analisa o custo para ir até cada posição adjacente válida, somando a esse cálculo o custo para chegar da origem até aquela posição. Com isso é possível obter a menor soma total de custos.

O algoritmo trabalha com listas onde são incluídas e retiradas coordenadas das posições e seus custos. As principais são: lista aberta, contendo as coordenadas onde é possível ir e que já tem o custo calculado por algum caminho; a lista fechada, contendo os itens que já foram

considerados para o caminho, indicando possíveis soluções; a lista de pais, contendo a informação de qual posição o custo para aquela posição foi calculado.

O custo para se chegar a uma certa posição é chamado de custo F, que é o resultado da soma dos custos G e H

O custo G de uma posição é o custo total para se mover do ponto inicial até aquela posição pelo caminho determinado até o momento.

O custo H é a estimativa de custo do ponto analisado até o destino final. Essa estimativa pode ser feita utilizando várias técnicas diferentes.

O pseudo código do algoritmo A* é representado a seguir:

Faça:

1. Inicialize a Lista Aberta com a posição inicial como única entrada;
2. Se Lista Aberta está vazia, interrompa. Se não, escolha o melhor elemento da Lista Aberta (menor custo);
3. Se a posição atual é o objetivo, retorne Lista Fechada;
4. (De outro modo) Remova posição atual da Lista Aberta;
5. Encontre as posições a partir da posição atual que não estão na Lista Fechada e crie todas as extensões da posição atual para cada posição descendente (calcule o custo F de cada descendente);
6. Adicione os caminhos estendidos a Lista Aberta e vá ao passo 2;

A representação em fluxograma da lógica do algoritmo A* pode ser visto na Figura 2.9.

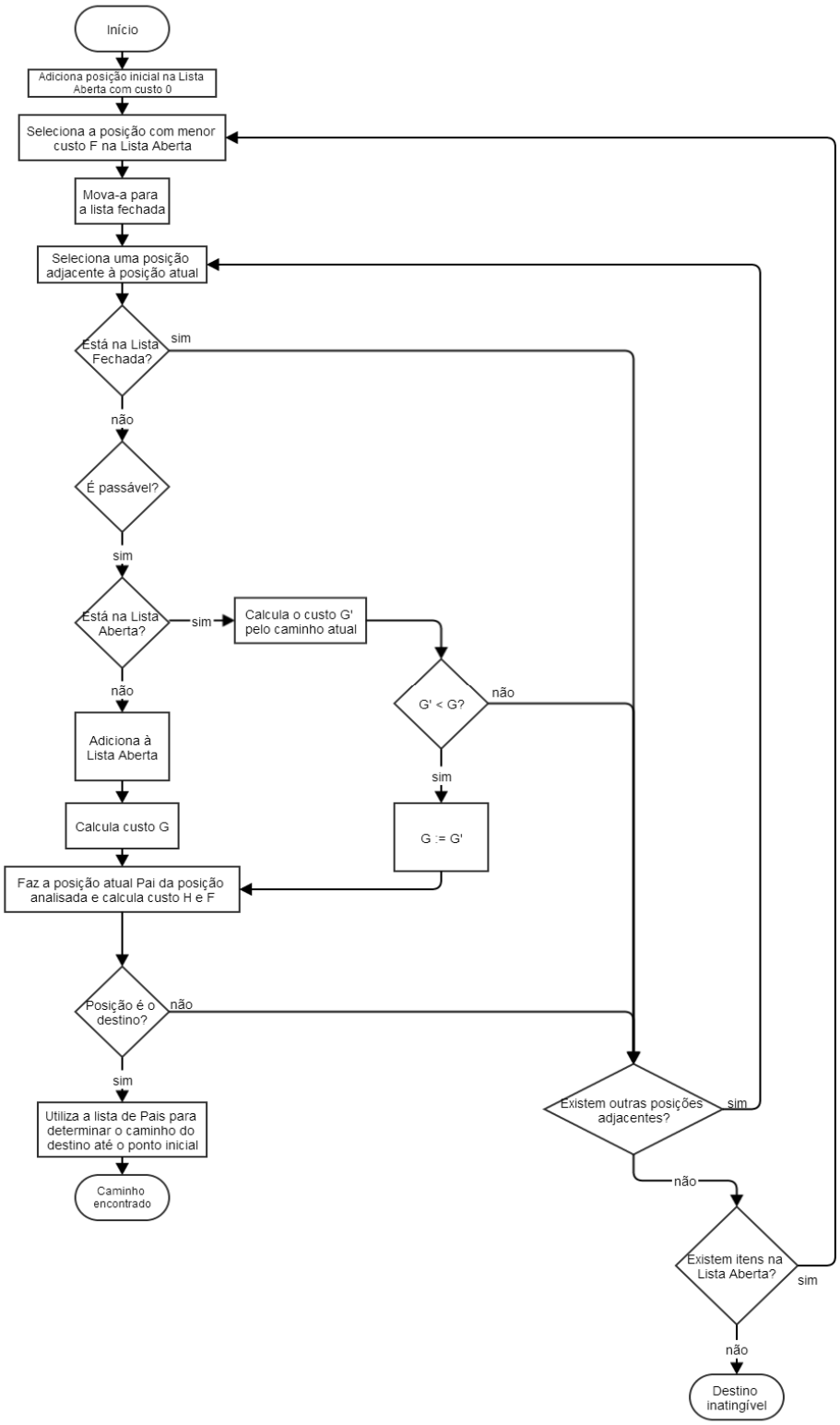

Figura 2.9. Fluxograma do algoritmo A* (ZHAN e NOON,1998)

3 Implementação do sistema de controle de navegação

O *hardware* utilizado nesse projeto possui alguns módulos que permitem o controle e sensoriamento do robô, além de comunicação sem fio. O bom funcionamento desses módulos é fundamental para que um algoritmo de navegação autônoma possa ser testado nesse *hardware*, evitando que problemas indiretos interfiram nos resultados dos testes.

3.1. Infravermelho

O circuito para teste do módulo de sensores por infravermelho foi alimentado com 3,3V para ter uma tensão compatível com a tensão de sinal da placa Raspberry Pi e sua saída foi conectada a uma entrada digital da placa.

Nos testes realizados com o par emissor/receptor de infravermelho, notou-se que a impedância do fotodiodo receptor varia conforme a intensidade luminosa do ambiente, o que faz a tensão na entrada inversora do amp-op variar em uma faixa considerável.

Consultando soluções comerciais utilizadas para acionamento automático de torneiras (Torneira para lavatório de mesa com sensor – Deca.com, 2014), notou-se que são utilizados placas de acrílico translúcido vermelho que diminuem o efeito da variação de luminosidade do ambiente sobre os sensores de infravermelho. Um exemplo pode ser visto na Figura 3.1.

Figura 3.1. Exemplo de sensor infravermelho comercial (Deca.com)

Para a solução implementada neste trabalho, o par emissor/receptor foi envolto em fita isolante para diminuir a interferência da luminosidade ambiente na leitura do sinal de infravermelho pelo receptor. Essa configuração pode ser vista na Figura 2.3. Foram implementados três pares como visto na Figura 3.2.

A sensibilidade do módulo infravermelho foi testada pelo circuito da Figura 2.2 montado no protoboard, aproximando um objeto do sensor e medindo com uma régua a distância em que o circuito indica a presença de obstáculo. O ensaio foi repetido para três condições de iluminação artificial e uma situação de iluminação natural. Os resultados são apresentados na Tabela 3.1

condição de luminosidade	**distância de detecção de obstáculo**	
luz artificial	**número de medidas**	**média das medidas [cm]**
1 lâmpada fluorescente de 25W	20	15,31
1 lâmpada fluorescente de 25W + lâmpada incandescente de 30W	20	7,03
1 lâmpada fluorescente de 25W + lâmpada incandescente de 30W + flash de iPhone ligado como lanterna	20	2,14
luz natural (13:00)	20	0,32

Tabela 3.1. Resultado do teste de sensibilidade do módulo infravermelho

3.2. Acionamento dos motores

Para efetuar o acionamento foi utilizado o circuito integrado L298E que possui duas pontes que podem ser controladas de maneira independente. Esse CI apresenta internamente a configuração de ponte H, muito utilizada para controle de motores DC e de passo. A corrente máxima fornecida por esse driver é de 2A por canal, o que é suficiente para acionar os motores contidos no chassi utilizado.

O CI utilizado recebe como entrada dois pulsos por ponte H, um sinal para cada lado da ponte, permitindo assim que os dois transistores inferiores da ponte fiquem habilitados, possibilitando uma recirculação de corrente no caso de uma inversão. Essa funcionalidade não foi explorada nesse projeto e foram adicionados diodos de roda livre, para que uma corrente reversa não cause danos ao driver. Foi utilizado apenas um sinal de controle de direção binário por motor para isso, o sinal de um lado da ponte vem diretamente do opto-acoplador e outro sinal é obtido com inversão lógica utilizando um transistor de junção bipolar em coletor aberto. Dessa maneira, o nível lógico baixo faz o motor girar em um sentido e o nível lógico alto no sentido inverso.

A velocidade é controlada por meio da entrada do CI L298 chamada *Enable*, que permite desabilitar a saída de uma ponte H, de maneira independente. Assim, aplicando um sinal PWM nessa porta, pode-se alterar a potência entregue a cada motor independentemente.

3.3. WiringPi

O acionamento de pinos do processador quando é utilizado um sistema operacional de uso geral não é direto como em microcontroladores. Para facilitar essa interface entre *software* e *hardware* foi utilizada uma biblioteca de uso gratuito criada por Henderson (2013), chamada de WiringPi. O uso dessa biblioteca tornou a programação mais natural e clara, uma vez que a configuração dos registradores com função especial do microprocessador fica abstrato.

3.4. Sistema de navegação

Em Grana (2013) foi desenvolvido um sistema de navegação inteligente, com funcionalidades como adaptação em relação à mudanças na disposição de obstáculos em relação ao robô, baseando-se em padrões pré-configurados e a melhora das soluções utilizando um algoritmo evolutivo. Esse método mostrou-se eficiente para ambientes desconhecidos e onde possa haver mudanças extremas na disposição dos elementos do ambiente.

Em contrapartida, o presente trabalho parte do pressuposto de que o ambiente é parcialmente conhecido (como um chão de fábrica que pode ser previamente mapeado) e que há elementos dinâmicos transitando por ele. Assim sendo, será utilizado um algoritmo A* para navegar o robô dentro do ambiente.

O algoritmo evolutivo utilizado por Grana (2013) para configurar a rede neural que controla o sistema reativo, será adaptado para melhorar o sistema de navegação modificando os parâmetros do algoritmo A* e do sistema de decisão de missões.

Os itens implementados serão detalhados a seguir.

3.4.1. Algoritmo A*

Do ponto de vista do algoritmo A*, o custo para se mover na vertical ou horizontal foi definido como 10 e o custo para movimentação diagonal como 15.

Para estimar o custo H será utilizado a soma dos custos para se atingir a coordenada y e posteriormente a coordenada x do destino, assim trabalhando somente com números inteiros.

O A* será acionado somente no início da movimentação do robô e quando houver atualização do mapa virtual.

3.4.2. Sensores

Foram montados no chassi do robô três sensores: um com direcionamento frontal e outros dois com angulação de 45 graus para detecção de obstáculos nas laterais, como visto na Figura 3.2.

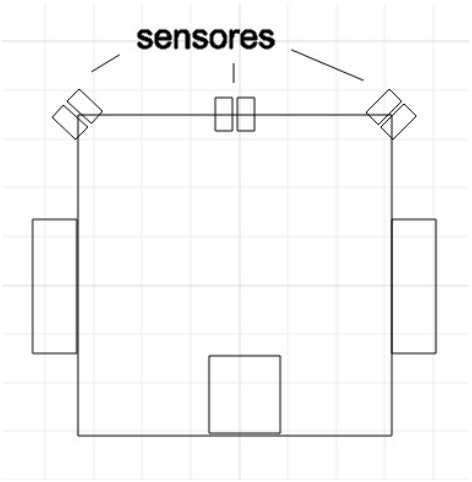

Figura 3.2. Localização dos sensores de infravermelho

3.4.3. Sistema de mapeamento

Foi implementado um sistema embarcado que, através da conexão sem fio da placa Raspberry Pi, acessa um arquivo em um servidor da rede local onde está mapeada a planta do ambiente onde o robô irá atuar. A partir desse mapa gravado em sua memória local, os sensores atualizam os dados com possíveis obstáculos novos e removidos e, gravando esse arquivo no banco remoto, possibilita que outros agentes do sistema possam considerar o obstáculo em suas rotinas de formação de rota e decisão de aceitação de missões.

Um exemplo do arquivo gerado e do ambiente em que o robô foi colocado pode ser visto na Figura 3.3.

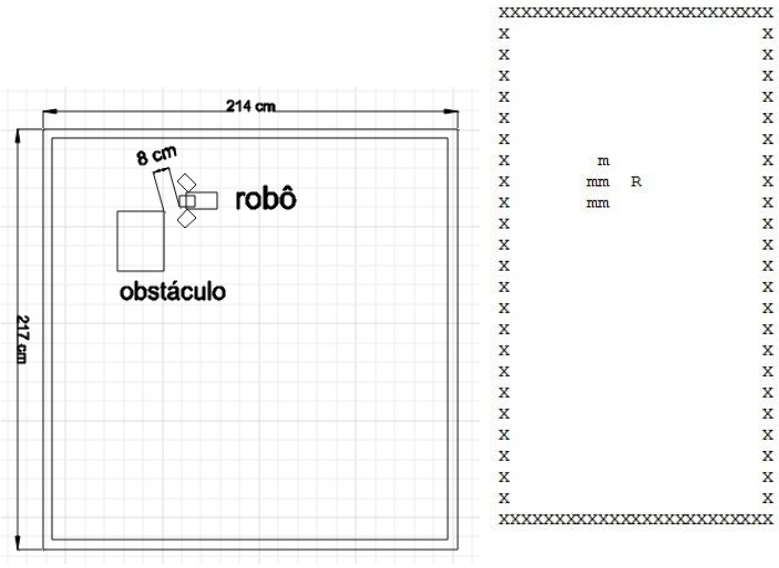

Figura 3.3. Comparação entre o ambiente real e mapa virtual.

Para essa validação, o robô teve a função de movimentação automática suspensa e o controle da movimentação foi feito manualmente, entrando com os passos do robô através de comandos por um terminal remoto, (cf. Windows Networking on the Raspberry Pi | Arcsofware Consultancy Blog).

Como pode ser notado, há uma defasagem entre o ambiente real e o virtual, causada principalmente por erro na discretização da posição do robô, escorregamento e falha na distância percorrida a cada passo.

3.4.4. Algoritmo supervisor

O algoritmo supervisor é responsável por, a cada passo no mapa, verificar se há necessidade de refazer o roteamento obtido pela função A*. Esse algoritmo também é responsável por modificar a pontuação do sistema atual de parâmetros do indivíduo. Essa pontuação será utilizada como *fitness* do algoritmo evolutivo.

A cada missão aceita e cumprida, o algoritmo incrementa a pontuação de um valor de 10 e a cada missão deixada para outro indivíduo, onde o robô avaliado é o elemento mais próximo, esse valor é decrementado de 1.

As missões consistem em dois destinos que devem ser alcançados em ordem, simulando o transporte de pacotes de uma linha de montagem de um ponto ao outro. O destino “D” deve ser alcançado primeiro, seguido do destino “d”. Um exemplo do mapa virtual gerado pela colocação de uma missão pode ser visto na Figura 3.4.

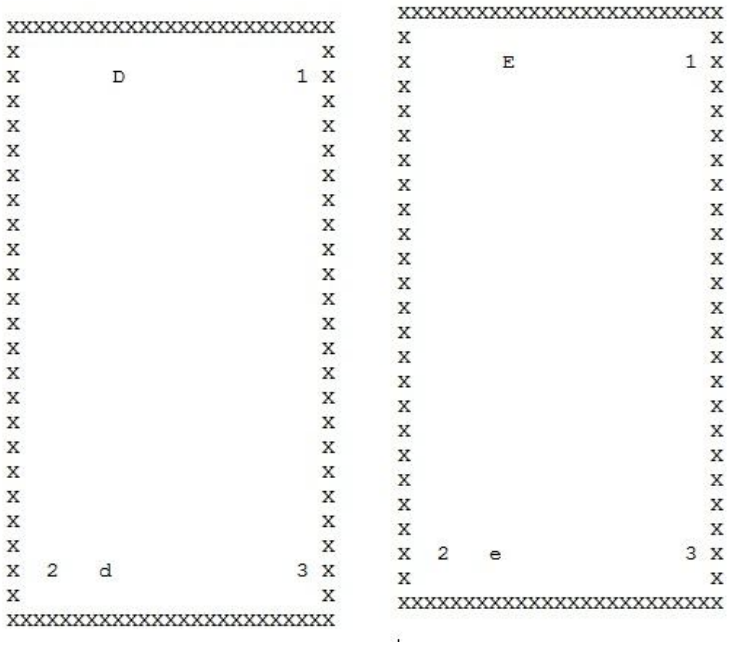

Figura 3.4. Mapa virtual gerado pela adição de uma missão e posteriormente pela aceitação pelo indivíduo 1.

No mapa da Figura 3.4 cada número representa a localização de um indivíduo. O algoritmo supervisor, então, aguarda que um indivíduo assuma a missão, o que é indicado acrescendo o valor do indivíduo aos indicadores de destino. Caso a missão seja aceita pelo indivíduo que o algoritmo supervisor está avaliando, ele acresce a pontuação daquele robô.

Para ser possível cada indivíduo se localizar no início da operação, as posições iniciais são fixadas e devem ser respeitadas a cada nova execução. As posições iniciais de cada indivíduo podem ser vistas na Figura 4.2.

As missões são aceitas pelos indivíduos baseando-se, primariamente, na distância entre eles e o primeiro destino da missão, assim são aceitas apenas as missões nas quais o indivíduo é o elemento mais próximo entre todas as posições de robôs conhecidas. Foi, porém, inserido um fator chamado "fator de rebeldia", onde há uma chance de um indivíduo aceitar uma missão

mesmo que não seja o mais próximo do primeiro destino. Esse fator varia de 0 a 50% e é considerado antes do descarte da missão pelo algoritmo de aceitação de missões.

3.4.5. Algoritmo evolutivo

Ao final do período de avaliação da população, o algoritmo evolutivo é acionado e são feitas as fases de seleção, *crossover* e mutação. Cada indivíduo é composto por dois genes, representantes da velocidade da movimentação, por meio de um valor de variação do ciclo de trabalho do PWM e do fator de rebeldia. Cada gene é formado por 10 bits, o que gera um cromossomo de 20 bits, resultando em uma variação possível de 2^{20} indivíduos.

Foi utilizado um algoritmo evolutivo, com as etapas de seleção, *crossover* e mutação. Para efetuar uma busca refinada do resultado foi implementada uma mutação variável dependente da velocidade com a qual o algoritmo progride, ou seja, quantas gerações se passam até que o melhor indivíduo mude. A partir das conclusões sobre taxa de mutação variável de Grana (2013), foi aplicado um algoritmo no qual o valor da mutação é aumentado quando o tempo que o melhor indivíduo permanece o mesmo faz com que o valor da mutação atinja zero. Assim, caso o melhor indivíduo permaneça o mesmo por um número de gerações suficiente, a rotina de refino da solução faria com que a taxa de mutação atingisse o valor zero, mas com a ação do algoritmo supervisor, essa taxa é novamente mantida em um valor mínimo.

A mutação variável é útil para encontrar soluções em todo o espaço de busca e refinar o resultado quando este se aproxima de um máximo local. Quando utilizada uma taxa fixa de mutação, o *fitness* pode não atingir o máximo se ela for muito alta, ou pode-se ficar preso em um máximo local e não percorrer todas as soluções se a taxa for muito baixa.

Para esse trabalho foi utilizada mutação de ponto único com *bit flip* e taxa de mutação variável entre 5% e 20%. Significando que passarão pelo processo de mutação entre 5 a 20 indivíduos a cada cem e que de cada indivíduo será invertido um bit entre os 20 bits do gene.

Com o intuito de viabilizar a implantação da solução proposta em um ambiente de chão de fábrica, foi idealizada a aplicação do algoritmo evolutivo em etapas em que é fixado o parâmetro de mutação. Assim existem três etapas: a inicial, em que é feito um período de testes para obtenção de um *fitness* mínimo; a etapa chamada de “semana”, na qual o parâmetro de mutação é reduzido; e finalmente a etapa de fim de semana, em que a mutação é aumentada gradualmente até o valor máximo. As duas últimas etapas se alternam para simular o ritmo de trabalho de uma linha de produção. Com isso foi possível, durante os dias de funcionamento normal da fábrica, manter o *fitness* médio dentro de uma faixa de valores próximos ao melhor

indivíduo, enquanto em dias sem expediente é possível buscar soluções melhores através do aumento da variabilidade genética, em detrimento do fitness médio da população.

A representação em fluxograma do algoritmo evolutivo contínuo pode ser visto na Figura 3.5.

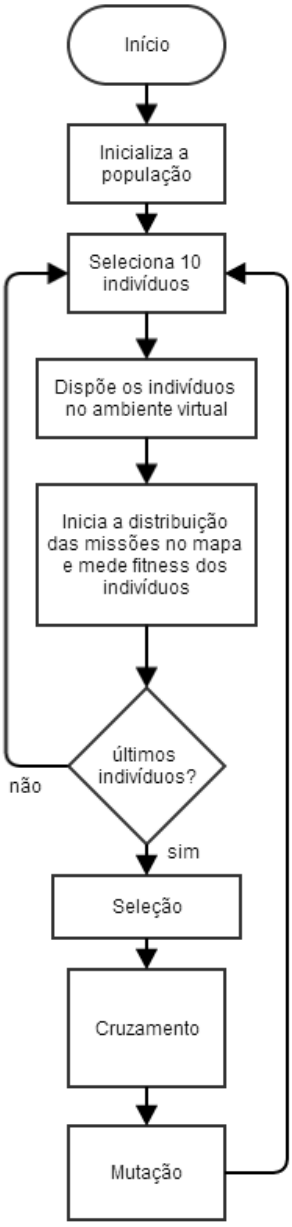

Figura 3.5. Algoritmo com evolução contínua

3.4.6 Sistema integrado de comunicação

O mapa do ambiente é armazenado no servidor e pode ser previamente construído contendo o *layout* básico do local, indicando obstáculos não removíveis como pilastras e máquinas fixas. Nesse mapa também são consideradas as posições de cada robô, informadas no momento da atualização do arquivo.

O robô tem acesso ao servidor através da rede *wi-fi*, onde ele lê o arquivo de mapa compartilhado a cada 30 segundos e faz uma cópia em sua memória local. Essa cópia é utilizada pelo sistema de navegação para traçar as rotas através do algoritmo A* e pelo sistema de decisão para aceitar missões. Caso os sensores do robô detectem um obstáculo que não estava indicado no mapa, é feita a inclusão no arquivo local e a atualização no arquivo principal no servidor. Os obstáculos detectados pelos robôs são indicados como objetos móveis, assim caso outro indivíduo passe adjacente ao espaço do objeto e não o detecte, é feita a remoção do obstáculo do mapa.

As missões podem ser inseridas no mapa do servidor através de programas que editam e atualizam o mapa. Esses programas podem ser construídos através de várias linguagens de programação como Python, Java, Visual Basic, entre outras, sendo utilizadas rotinas básicas de edição de texto para a inserção das missões.

O sistema de comunicação entre os elementos pode ser visto na Figura 3.7.

3.4.7 Integração entre módulos

A integração entre os módulos de *hardware* e *software* é mostrada na figura 3.6.

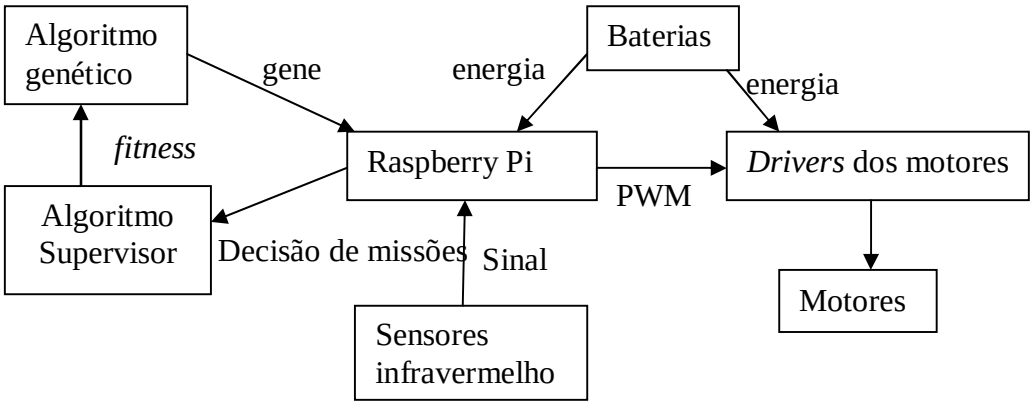

Figura 3.6. Integração dos módulos

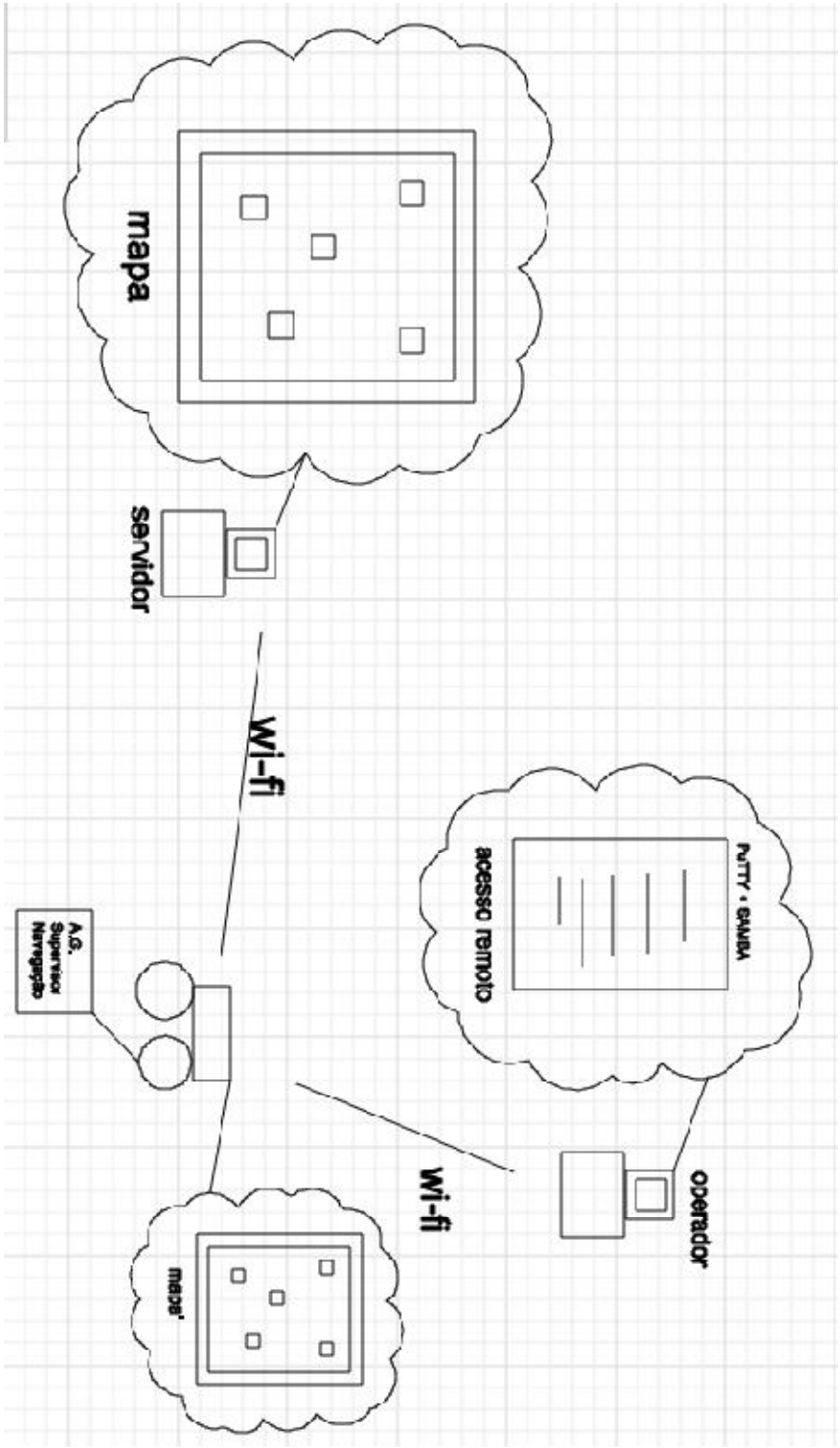

Figura 3.7. Esquemático do sistema integrado de comunicação

4 Resultados

4.1 Experimentos

Um *protoboard* unificando o condicionamento dos sinais dos sensores infravermelho, o controle do robô e o *driver*, foi construída e está reproduzida na Figura 4.1. Os resultados dos módulos estão expostos a seguir.

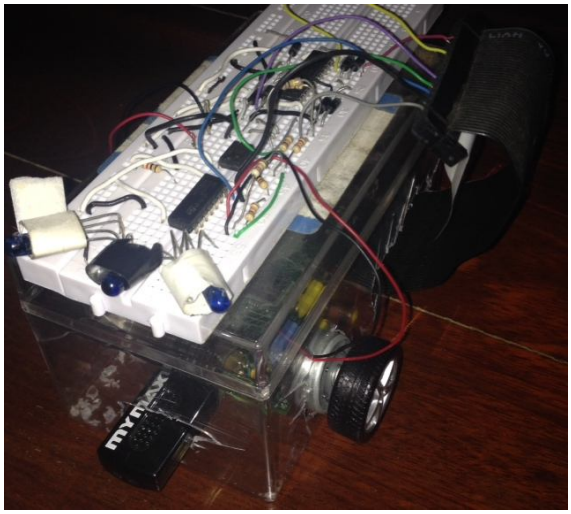

Figura 4.1. O protoboard montado sobre o robô

4.1.1.Infravermelho

Os primeiros testes foram feitos com o conjunto de *software* e *hardware* para avaliar a qualidade dos sensores de infravermelho.

Configurando o valor de tensão da entrada não inversora dos amp-ops para 2V e posicionando manualmente um obstáculo à frente do módulo de sensores em diversas condições de luminosidade, obteve-se um resultado que variou de 1 a 15 centímetros de sensibilidade à presença do obstáculo, o que foi considerado satisfatório para operação em ambiente interno.

Quando testado em ambiente sujeito à luz direta do sol, o circuito apresentou resultados mistos, não sendo capaz de identificar a presença do obstáculo em certas intensidades luminosas.

A Raspberry Pi, por não possuir uma entrada de conversor AD, depende de uma lógica binária para a detecção de obstáculos e não é capaz de medir o sinal emitido quando a luminosidade ambiente é muito intensa. Caso a placa contasse com esse recurso, seria possível, por meio de um par sensor/emissor voltado para o alto, medir o sinal gerado pela luz ambiente

quando não há obstáculo e compará-lo com o valor lido nos sensores do chassi, mitigando assim o efeito da variação da iluminação ambiente.

4.1.2. Controle de velocidade e direção dos motores

Foi necessário gerar dois pulsos PWM e dois sinais digitais para controlar as velocidades e direções dos dois motores, respectivamente. Os pulsos digitais foram gerados utilizando as portas de entrada e saída de uso geral, que são disponibilizadas na Raspberry Pi. O acionamento em software dessas portas foi feito utilizando a biblioteca Wiring Pi, já apresentada.

O *driver* utilizado necessita de quatro sinais digitais para determinar o sentido da rotação do motor, porém foi possível controlá-lo somente com dois pinos através de dois transistores bipolares configurados como inversores.

A geração de PWM por software foi demonstrado, em projetos anteriores, como sendo custosa do ponto de vista de processamento, mas para a aplicação pretendida foi considerada suficiente. Uma opção utilizando os periféricos já presentes no ARM da placa é apresentada em Grana (2013).

4.1.3. Simulação do algoritmo evolutivo

Para a validação do algoritmo evolutivo, foi executada uma simulação na qual o ambiente virtual é alimentado com missões com periodicidade variável e onde a movimentação dos indivíduos ocorre de forma mais rápida que no mundo real, a fim de acelerar o processo de evolução.

O ambiente virtual utilizado para a simulação tem dimensões 200x200 posições.

O processo de inicialização da população foi feito de forma aleatória, sendo gerados 100 indivíduos, testados em grupos de 10. A disposição dos indivíduos no ambiente virtual é mostrada na figura 4.2.

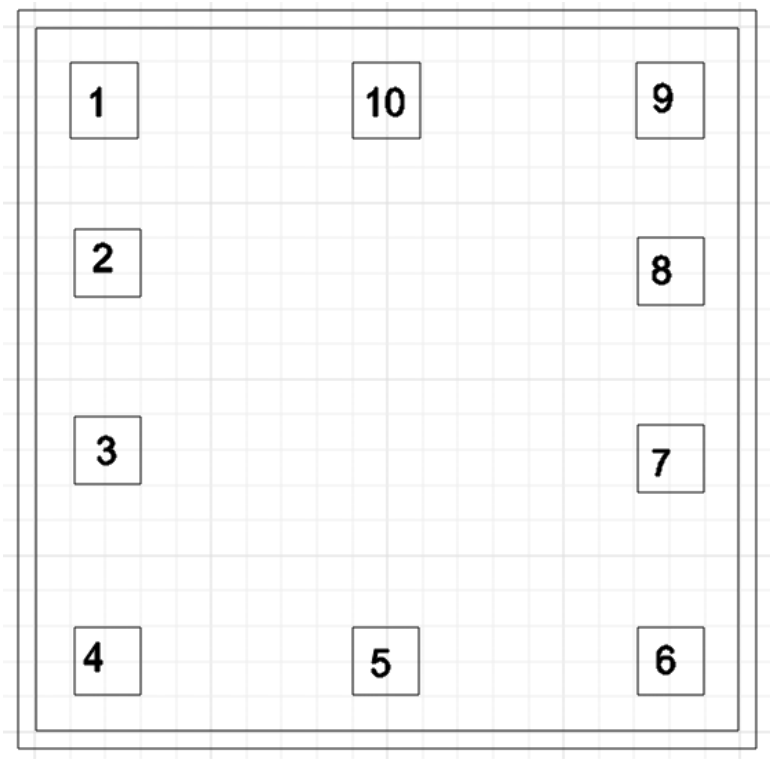

Figura 4.2. Disposição dos indivíduos no ambiente virtual para simulação

As missões são colocadas no ambiente de forma aleatória, sendo escolhido um espaço para o primeiro destino e outro para o segundo destino de cada objetivo. Cada período de avaliação dura 10.000 passos, sendo que os indivíduos com maior velocidade conseguem se deslocar por um maior número de posições por passo. O resultado da simulação é apresentado na Figura 4.3.

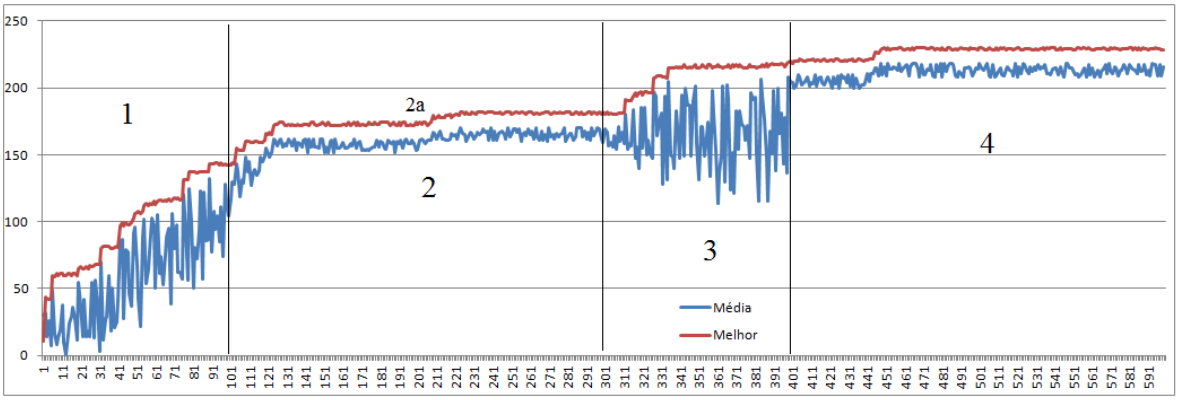

Figura 4.3. Resultado da simulação do algoritmo genético em ambiente virtual

Nas 100 primeiras gerações, indicadas pela divisão 1 na Figura 4.3, a taxa de mutação é iniciada em 10% e varia até 50% dependendo da velocidade de mudança do melhor indivíduo. Esse período foi descrito no capítulo 3, item 3.4.4 como etapa inicial.

Entre a geração 101 e a 300 tem-se a divisão 2 da Figura 4.3, em que foi simulado o período de funcionamento normal da suposta fábrica, na qual os genótipos são mantidos o mais próximo possível dos indivíduos com os melhores *fitness*, a fim de evitar discrepâncias visíveis na operação dos elementos do sistema. Para esse período, chamado de "semana", a taxa de mutação foi mantida em 5%. É possível notar na Figura 4.3, no detalhe 2a, que mesmo com essa taxa de mutação baixa, foram encontradas soluções com *fitness* melhores dentro desse período, podendo ser considerados um refinamento do máximo local.

A partir da geração 301 até a 400 é compreendido o período nomeado de "fim de semana", indicado pela divisão 3 da Figura 4.3, onde supostamente não há expediente na fábrica e poderão ser executadas avaliações com maior taxa de mutação, que pode atingir até 50%, o que causa grande variação no *fitness* médio da população. Esse período serve para buscar possíveis soluções distantes das áreas já analisadas.

Essa simulação do sistema robótico serve a dois propósitos nesse projeto: validar o modelo de algoritmo supervisor e algoritmo genético e demonstrar uma maneira de simular, em ambiente virtual, as características do chão de fábrica onde será implementado o sistema multirrobótico, para assim encontrar soluções a serem aplicadas no ambiente real. A simulação em paralelo é interessante em fábricas onde não há período sem expediente para executar a fase evolutiva chamada de "fim de semana", que pode, então, ser executada no ambiente simulado.

Não foi possível executar o ensaio do algoritmo em ambiente real devido à imprecisão dos sensores e ao escorregamento do sistema de rodas, o que gerou imprecisão cumulativa na localização do robô dentro do ambiente.

4.2 Análise dos resultados

A utilização do gerador de PWM por software se mostrou suficiente para a aplicação desejada, já que o consumo de processamento pelo *thread* ainda permitia o funcionamento das outras rotinas do robô. Caso se desejasse uma operação dos algoritmos envolvidos na navegação e no sistema decisório com custo computacional mais elevado, uma solução utilizando as portas nativas do processador deve ser implementada, como a desenvolvida por Grana (2013).

A utilização de rodas em vez de esteiras se mostrou problemática em ambientes com alto grau de escorregamento, já que o sistema de navegação compartilhada depende de uma

localização mais precisa de cada indivíduo. Mesmo com a velocidade do motor reduzida, há escorregamento que depende da rugosidade da superfície do piso.

A simulação do ambiente de avaliação se mostrou eficiente para atingir soluções com *fitness* suficiente e pode ser usada como ferramenta de evolução acelerada em paralelo à evolução em ambiente real. Esse método computacional pode obter novas soluções através do aumento da taxa de mutação, já que no ambiente virtual não há o comprometimento com o *fitness* médio da população, o qual precisa, no caso real, manter-se dentro de uma faixa viável para a continuidade do processo produtivo.

As soluções encontradas mais rapidamente em ambiente virtual podem ser transferidas para os robôs reais para serem submetidas ao ruído e às interações ambientais não consideradas na simulação. Esse procedimento acelera a validação em campo, onde é desejável que cada geração leve várias horas para ser avaliada, a fim de se obter maior confiabilidade nos resultados.

5 Conclusões

Este trabalho demonstrou uma forma de implementar um sistema embarcado de navegação e decisão para o cumprimento de missões em robôs de baixo custo utilizando a placa Raspberry Pi.

O algoritmo evolutivo embarcado se mostrou capaz de adaptar os parâmetros dos indivíduos a um ambiente dinâmico onde a disponibilidade de missões pode variar, além da presença de obstáculos dinâmicos. O sistema de mapa compartilhado por um servidor foi bem sucedido em auxiliar a navegação em um ambiente parcialmente conhecido, como um chão de fábrica ou uma residência, se mostrando robusto e de fácil adequação às várias situações onde se propõe aplica-lo. Já o sistema de rodas é suscetível ao escorregamento, o que atrapalhou a navegação por causar imprecisão na localização do robô dentro do mapa compartilhado.

O hardware nativo da Raspberry Pi se mostrou adequado para o acoplamento de alguns módulos básicos para a solução do problema proposto, porém para soluções mais complexas é recomendável a adição de hardware auxiliar, como a conexão com uma placa Arduino, conversores AD e módulos para navegação por ultrassom.

A interface da Raspberry Pi é adequada para o desenvolvimento de software, sendo a principal facilidade o acesso remoto através da rede *wi-fi* permitindo que o desenvolvimento ocorra em ambiente distinto do local do robô.

Em relação ao trabalho desenvolvido por Grana (2013), este projeto se focou em desenvolver um sistema de decisão voltado à uma aplicação específica, já o trabalho anterior buscou uma solução para a movimentação e desvio de obstáculos do robô. Este trabalho utilizou a mesma plataforma de *hardware* de controle (caracterizado por uma placa Raspberry Pi conectada a sensores e *drivers*) para desenvolver uma solução sistêmica de alto nível para o transporte de peças em uma linha de montagem. O módulo de detecção de obstáculos por ultrassom foi trocado pelo módulo infravermelho, sem perdas na determinação da presença de objetos. O sistema de mapeamento colaborativo também não havia sido implementado no trabalho de 2013, devido ao foco diferente dos trabalhos.

5.1 Trabalhos futuros

O sistema de locomoção por rodas se mostrou inadequado para operação em conjunto com o sistema de mapa compartilhado, assim sugere-se de adoção de um sistema mais preciso e com

menos escorregamento. Nesse âmbito, pode ser estudada também a possibilidade de utilizar os dispositivos Bluetooth com função de localização disponíveis no mercado atualmente, que utilizam a medição da potência recebida para calcular a distância que o receptor se encontra do emissor. O uso dessa tecnologia com técnicas de triangulação deve render bons resultados na mitigação de erros de localização. Esses equipamentos podem ser vistos em (A Strong range of new Bluetooth tracking devices, 2014).

Outra opção para melhoria da localização é a utilização de emissores acústicos em pontos fixos do ambiente, que podem ser simples caixas de som, receptores nos elementos móveis, que podem ser microfones, e um link de rádio. Ao emitir um pulso acústico em uma das caixas de som, o sistema envia um sinal pelo link de rádio o qual faz com que seja iniciado um contador interno aos robôs. Ao receber o pulso sonoro, o robô utiliza o valor do contador para calcular a distância que ele se encontra da fonte emissora. Com um sistema de triangulação pode ser possível obter uma localização mais precisa de cada robô no ambiente.

O sistema de infravermelho se mostrou suficiente para ambientes internos, mas o acoplamento de uma peça de acrílico na interface entre sensor e ambiente, como adotado em soluções comerciais, pode ser capaz de melhorar a precisão do sistema, assim como torná-lo menos suscetível a variações na intensidade da iluminação ambiente. Uma dessas soluções são os módulos de infravermelho utilizados em controle remoto de televisores e outros aparelhos, nos quais o sinal é considerado válido somente se for transmitido com uma frequência de pulso específica. Assim pode ser possível mitigar o efeito da iluminação ambiente, além de possibilitar o uso do pulso em períodos curtos, o que diminuiria a interferência do emissor de um indivíduo no receptor de outro.

Outras duas linhas de pesquisa que podem ser desenvolvidas a partir do presente trabalho são: a inclusão dos custos do algoritmo A* no algoritmo evolutivo e a variação desses custos a medida que uma rota é mais utilizada que as demais, criando assim caminhos preferenciais e o aprofundamento do conceito de "rebeldia", tentando emular melhor o comportamento de indivíduos na natureza.

6. Referências Bibliográficas

A Strong range of new Bluetooth tracking devices. **Dallas News,** 2014. Disponível em <http://www.dallasnews.com/business/technology/headlines/20140313-a-strong-range-of-new-bluetooth-tracking-devices.ece>. Acesso em: 4 de jun. 2014.

ABOUT us | **Raspberry Pi. Raspberry Pi**, 2012. Disponível em: <HTTP://www.raspberrypi.org/about>. Acesso em: 20 fev. 2014.

COVER, Thomas; HART, Peter. Nearest neighbor pattern classification.**Information Theory, IEEE Transactions on**, v. 13, n. 1, p. 21-27, 1967.

FAQS | Raspberry Pi. **Raspberry Pi,** 2014. Disponível em: <http://www.raspberrypi.org/faqs>. Acesso em: 20 fev. 2014.

GONÇALVES, Paulo C. **Protótipo de um robô móvel de baixo custo para uso educacional**. 2007. Tese de Doutorado. Dissertação (Mestrado em Ciência da Computação), Universidade Estadual de Maringá, Maringá, PR.

GRANA, A. L. S. **Sistema de controle de navegação embarcado para um robô móvel autônomo baseado no computador de baixo custo Raspberry Pi.** Monografia (Graduação em 2013), Universidade de São Paulo, São Carlos, SP.

HENDERSON, G. Gordon. **Raspberry Pi | wiringPi | Gordon Projects**,2014.Disponível em: <https://projects.drogon.net/raspberry-pi/wiringpi/software-pwm-library/>. Acesso em 20 jan. 2014.

JUNG, Cláudio Rosito et al. Computação embarcada: Projeto e implementação de veículos autônomos inteligentes. **Anais do CSBC**, v. 5, 2005.

KON, Anita. Tecnologia e trabalho no cenário da globalização. **Desafios da globalização**, v. 3, 1997.

LEE, Geunho; CHONG, Nak Young.Low-cost dual rotating infrared sensor for mobile robot swarm applications. **Industrial Informatics, IEEE Transactions on**, v. 7, n. 2, p. 277-286, 2011.

LIST of single-board computers. **Wikipedia**, 2013. Disponível em: <http://en.wikipedia.org/wiki/List_of_single-board_computers>. Acesso em: 15 jan. 2014.

Making of air sensor module. **Robotics Bible,** 2013. Disponível em: <http://www.roboticsbible.com/making-of-ir-sensor-module.html>. Acesso em: 10 de dez. 2013.

Opening a file in a remote location. **Stack Overflow,** 2014. Disponível em: <http://stackoverflow.com/questions/3308245/fopen-file-from-windows-network-location>. Acesso em 12 fev. de 2014.

RABELO, Ricardo J. Sistemas Multiagentes. **Universidade Federal de Santa Catarina – Departamento de Automação e Sistemas**, 2002.

RASPBIAN About. **Raspbian**, 2012. Disponível em <http://www.raspbian.org/RaspbianAbout>. Acesso em: 25 jan. 2014.

ROMANO, Vitor Ferreira; DUTRA, Max Suell. Introdução à Robótica Industrial. **Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos**, v. 1, p. 1-19, 2002.

RPI Distributions – eLinux.org. **eLinux.org**, 5 de maio de 2014. Disponível em: <http://www.elinux.org/RPi_Distributions>. Acesso em 10 maio 2014.

RPi Hub – eLinux.org. **eLinux.org**, 9 de maio de 2014. Disponível em: <http://elinux.org/RPi_Hub>. Acesso em: 10 maio 2014.

Setting up a SAMBA Server on Raspberry Pi. **The Urban Penguin,** 2013. Disponível em <http://theurbanpenguin.com/wp/?p=2415>. Acesso em: 3 jun. 2014.

Share your Raspberry Pi's file and folder across a network. **Raspberry Pi Web Server,** 2014. Disponível em: <http://raspberrywebserver.com/serveradmin/share-your-raspberry-pis-files-and-folders-across-a-network.html>. Acesso em 3 jun. 2014.

SIMÕES, E. V. **Development of an Embedded Evolutionary Controller to Enable Collision-Free Navigation of a Population of Autonomous Mobile Robots**. Tese (Doutorado), University of Kent, Canterbury. 2000.

SIQUEIRA, T. F. et al. Desenvolvimento de Sistemas Embarcados para Aplicações Críticas IV Escola Regional de Redes de Computadores, Passo Fundo, setembro 2006. SCHMIDT R. **Certificação de Software para Aplicações Críticas utilizando a norma IEC**, v. 61508, 2009.

Torneira para lavatório de mesa com sensor. **Deca.com**, 2014. Disponível em <http://www.deca.com.br/produtos/torneira-acion-c-sens-decalux-bivolt-super-luxo-2/>. Acesso em: 20 fev. 2014.

TOWNSEND, K. Adafruit 16 Channel Servo Driver with Raspberry Pi. **Adafruit Learning System**, 13 de Abril de 2014. Disponível em <https://learn.adafruit.com/adafruit-16-channel-servo-driver-with-raspberry-pi/overview>. Acesso em: 14 abril 2014.

Windows Networking on the Raspberry Pi. **Arcsoftware Consultancy Blog**, 2013. Disponível em <http://blogs.arcsoftwareconsultancy.com/pi/2013/03/07/windows-networking/>. Acesso em: 20 fev. 2014.

WOLF, D. F. Et al. Robótica Móvel Inteligente: Da Simulação às Aplicações no Mundo Real. In: **XXVIII Jornadas de Atualização em Informática**. [S.l.]: [s.n.], 2009.

ZHAN, F. Benjamin; NOON, Charles E. Shortest path algorithms: an evaluation using real road networks. **Transportation Science**, v. 32, n. 1, p. 65-73, 1998.