UNIVERSIDADE ESTADUAL PAULISTA JÚLIO DE MESQUITA FILHO
INSTITUTO DE BIOCIÊNCIAS, LETRAS E CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Takao Matsumura

Desenvolvimento de uma plataforma aberta de robô móvel para propósitos gerais

São José do Rio Preto
2014

Takao Matsumura

Desenvolvimento de uma plataforma aberta de robô móvel para propósitos gerais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas, Campus de São José do Rio Preto da Universidade Estadual Paulista Júlio de Mesquita Filho, como requisito para a obtenção do título de Mestre em Ciência da Computação.

Linha de Pesquisa: Arquitetura de Computadores e Sistemas Distribuídos

Orientador: Norian Marranghello
Coorientador: Humberto Ferasoli Filho

São José do Rio Preto
2014

Matsumura, Takao.
Desenvolvimento de uma plataforma aberta de robô móvel para propósitos gerais / Takao Matsumura. -- São José do Rio Preto, 2014
131 f. : il.; tabs.

Orientador: Norian Marranghello
Coorientador: Humberto Ferasoli Filho
Dissertação (mestrado) – Universidade Estadual Paulista "Júlio de Mesquita Filho", Instituto de Biociências, Letras e Ciências Exatas

1. Computação. 2. Robótica. 3. Robôs móveis. 4. Robôs – Sistemas de Controle. I. Marranghello, Norian. II. Ferasoli Filho, Humberto. III. Universidade Estadual Paulista "Júlio de Mesquita Filho". Instituto de Biociências, Letras e Ciências Exatas. IV. Título.
CDU – 681.51

Ficha catalográfica elaborada pela Biblioteca do IBILCE
UNESP – Câmpus de São José do Rio Preto

Takao Matsumura

Desenvolvimento de uma plataforma aberta de robô móvel para propósitos gerais

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Biociências, Letras e Ciências Exatas, Campus de São José do Rio Preto da Universidade Estadual Paulista Júlio de Mesquita Filho, como requisito para a obtenção do título de Mestre em Ciência da Computação.

Linha de Pesquisa: Arquitetura de Computadores e Sistemas Distribuídos

Orientador: Norian Marranghello
Coorientador: Humberto Ferasoli Filho

Comissão Examinadora

Prof. Dr. Humberto Ferasoli Filho, Presidente
Universidade Estadual Paulista Júlio de Mesquita Filho – Bauru - SP

Prof. Dr. Teodiano Freire Bastos Filho
Universidade Federal do Espírito Santo – Vitória - ES

Prof. Dr. João Eduardo Machado Perea Martins
Universidade Estadual Paulista Júlio de Mesquita Filho – Bauru -SP

São José do Rio Preto

15 de julho de 2014

"The machine does not isolate man
from the great problems of nature but
plunges him more deeply into them."

Antoine de Saint-Exupéry

AGRADECIMENTOS

Aos meus familiares, especialmente meu pai Tatsuo e minha mãe Yoshiko que, além de fazerem parte da formação do meu caráter, priorizaram minha educação, sempre estiveram ao meu lado quando precisei e me apoiaram em todas as fases da minha vida.

Ao professor Norian, por acreditar na minha capacidade, pela receptividade e oportunidade de cursar o mestrado e ao aceitar orientar o meu mestrado. Agradeço-o pela sua orientação, paciência e suas observações que contribuíram de forma decisiva no decorrer desses últimos anos.

Ao professor Humberto, pela sua motivação e incentivos que me apoiaram desde a graduação e que me conduziram a almejar o mestrado. Agradeço-o por aceitar orientar meu trabalho de conclusão da graduação, e desta vez, coorientar meu mestrado. Agradeço-o pelos finais de semana e horários diferenciados dedicados à minha orientação (e cobrança), seus inúmeros auxílios e pelo seu tempo cedido para contribuir diretamente com meu trabalho no laboratório e na produção dos meus textos. Agradeço em especial à sua dedicação com o laboratório, ferramentas, materiais e ao seu *know-how* sem os quais não teria sido possível imaginar ou prosseguir com os meus trabalhos práticos.

Aos professores Renê Pegoraro e João Perea, pelas suas colaborações e auxílios no laboratório. Agradeço-os pelos seus empréstimos de componentes, ferramentas e dispositivos.

Aos meus colegas e amigos de laboratório, Silas Alves, Rogério Barbosa e Diego Voltan, pela convivência descontraída, reflexões, contribuições diretas e indiretas com meu trabalho e por me aguentarem ao longo desses anos. Agradeço ao Silas também pelo seu apoio, colaboração no laboratório (até tarde da noite e finais de semanas) e por ceder seu tempo e atenção nas revisões dos meus textos.

À Unesp, ao Departamento de Computação de Bauru, ao Departamento de Ciência da Computação e Estatítica de São José do Rio Preto e ao Programa de Pós Graduação

em Ciência da Computação, que por meio de suas estruturas, possibilitaram o desenvolvimento deste trabalho. Ao corpo docente, que contribuiu através de conteúdos disciplinares, e aos professores que contribuíram com este trabalho, de forma direta ou indireta.

Aos funcionários do Lepec e das oficinas da Faculdade de Arquitetura, Artes e Comunicação, que me deram suporte na utilização dos laboratórios. Aos funcionários da oficina mecânica da Faculdade de Engenharia de Bauru pelo auxílio no projeto e pela confecção de peças sob medida que foram utilizados neste trabalho.

Aos amigos presentes e aos que passaram pela minha vida. Agradeço a todos aqueles que me apoiaram, torceram por mim e àqueles que sempre estiveram presente quando precisei.

Por fim, agradeço à CAPES pelo suporte financeiro ao longo do desenvolvimento deste trabalho.

RESUMO

A pesquisa em robótica móvel tem sido impulsionada pelo avanço tecnológico. Existem frentes de pesquisas que abordam diferentes aspectos e desafios da robótica móvel, dentre as quais estão tópicos como a locomoção, navegação e arquiteturas de controle. Esse crescimento em pesquisas também acarreta uma maior necessidade por bases móveis, em outras palavras, por plataformas de robôs móveis de propósitos gerais, destinadas à pesquisa e, também, para fins educacionais. Esta situação é aplicável ao Laboratório de Automação e Computação Evolutiva (LACE) do IBILCE, Campus da Unesp de São José do Rio Preto. Este trabalho apresenta o desenvolvimento de uma plataforma de robô móvel de baixo custo e de arquitetura de *hardware*, *software* e controle aberta, para propósitos gerais. A plataforma almeja a facilitação e flexibilização do processo de desenvolvimento de aplicações robóticas e estudos por meio de uma interface de comunicação simplificada e pela abstração da heterogeneidade dos dispositivos periféricos de hardware. Esta plataforma é capaz de oferecer maior liberdade em relação às linguagens de programação, paradigmas de controle e ambientes de desenvolvimento de seu *software* de controle, se estendendo aos dispositivos tecnológicos de controle.

Palavras-Chave: Arquitetura Aberta. Plataforma de Robô Móvel. Robô Móvel. Interface de Comunicação.

ABSTRACT

The research on mobile robots has been driven by technological advance. There are research fronts on different aspects and challenges of mobile robots, among which are topics such as locomotion, navigation and control architectures. This growth in research also leads a greater need for mobile bases, in other words, for mobile robot platforms for general purposes, destined for research and educational purposes. This situation is applicable to the Laboratory of Evolutionary Automation and Computation of IBILCE, Unesp Campus of São José do Rio Preto. This work presents the development of a low cost mobile robot platform with an open hardware, software and control architecture, intended for general purposes. The platform aims at facilitating and flexibilizing the development process of robotic applications and studies through a simplified communication interface and abstracting the heterogeneity of the peripheral hardware devices. This platform is able to offer greater freedom in relation to programming languages, control paradigms, development environments of its control software, extending to the technological control devices.

Keywords: Open Architecture. Mobile Robot Plataform. Mobile Robot. Communication Interface.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface* (Interface de Programação de Aplicativos)
CAN	*Controller Area Network* (Controlador de Rede de Área)
CORBA	*Common Object Request Broker Architecture* (Intermediário Comum para Requisições de Objetos)
CMOS	Complementary Metal-Oxide-Semiconductor (Semicondutor Metal-Óxido Complementar)
DARPA	*Defense Advanced Research Projects Agency* (Agência de Projetos de Pesquisa Avançada de Defesa)
DOF	*Degree of Freedom* (Grau de Liberdade)
FFI	*Foreign Function Interface* (Interface de Funções Estrangeiras)
GPS	*Global Positioning System* (Sistema de Posicionamento Global)
IEEE	*Institute of Electrical and Electronics Engineers* (Instituto de Engenheiros Eletricistas e Eletrônicos)
I^2C	*Inter-Integrated Circuit* (Circuito Inter-Integrado)
LACE	Laboratório de Automação e Computação Evolutiva
LISDI	Laboratório de Integração de Sistemas e Dispositivos Inteligentes
MARIE	*Mobile and Autonomous Robotics Integration Environment* (Ambiente de Integração de Robôs Móveis e Autônomos)
MIRO	*Middleware for Mobile Robots* (Middleware para Robôs Móveis)
NASA	*National Aeronautics and Space Administration* (Administração Nacional do Espaço e da Aeronáutica)
PC	*Personal Computer* (Computador Pessoal)
PIC	*Programmable Interface Controller* (Controlador de Interface Programável)
PYRO	*Python Robotics* (Python Robótico)
PWM	*Pulse Width Modulation* (Modulação por Largura de Pulso)

RDE	*Robot Development Environment* (Ambiente de Desenvolvimento de Robô)
RIA	*Robotic Industries Association* (Associação das Indústrias Robóticas)
RSCA	*Robot Software Communication Architecture* (Arquitetura de Comunicação de Software Robótico)
ROS	*Robot Operating System* (Sistema Operacional para Robôs)
RF	Radio Frequency (Rádio Frequência)
RT	*Robot Technology* (Tecnologia Robótica)
RTCAN	Real Time CAN (CAN de Tempo Real)
RT-CORBA	Real-Time CORBA (CORBA de Tempo Real)
SO	Sistema Operacional
SONAR	Sound Navigation and Ranging (Navegação e Determinação da Distância pelo Som)
SPI	*Serial Peripheral Interface* (Interface de Periférico Serial)
TCP	*Transmission Control Protocol* (Protocolo de Controle de Transmissão)
TTL	*Transistor-Transistor Logic* (Lógica Transistor-Transistor)
UPnP	*Universal Plug-and-Play* (Plug-and Play Universal)
USB	(Barramento Serial Universal)
URB	*Universal Robot Bus* (Barramento Robótico Universal)

LISTA DE SÍMBOLOS

Letras Latinas

C	Centro geométrico do robô
CX_lY_l	Sistema de coordenadas local do robô
$C_rX_rY_r$	Sistema de coordenadas de referência
$\mathcal{F}_g$	O mesmo que OX_gY_g
$\mathcal{F}_l$	O mesmo que CX_lY_l
OX_gY_g	Sistema de coordenadas global ou inercial
P	Ponto de interseção entre o eixo de simetria e o eixo das rodas
q	Vetor de coordenadas generalizadas
$\dot{q}$	Vetor das velocidades em coordenadas generalizadas
r	Raio das rodas
R	Distância entre as rodas e o eixo de simetria
$S(q)$	Matriz jacobiana ou modelo cinemático
v	Velocidade linear
$v_{\max}$	Velocidade linear máxima
x	Posição no eixo das abscissas
$\dot{x}_c$	Velocidade do robô móvel no eixo das abcissas do plano $\mathcal{F}_g$
y	Posição no eixo das ordenadas
$\dot{y}_c$	Velocidade do robô móvel no eixo das ordenadas do plano $\mathcal{F}_g$

Letras Gregas

θ	Ângulo do robô em relação ao eixo das abscissas
ω	Velocidade angular
φ	Ângulo da roda do robô
$\dot{\varphi}_d$	Velocidade angular da roda direita de um robô de tração diferencial
$\dot{\varphi}_e$	Velocidade angular da roda esquerda de um robô de tração diferencial
$\dot{\theta} = \omega$	

SUMÁRIO

1 INTRODUÇÃO

A pesquisa em robótica móvel tem sido impulsionada pelos avanços tecnológicos. Existem frentes de pesquisas que abordam diferentes aspectos e desafios da robótica móvel, dentre os quais estão tópicos como a locomoção, navegação e arquiteturas de controle. Esse crescimento em pesquisas também acarreta uma maior necessidade por bases móveis, em outras palavras, por plataformas de robôs móveis que possam ser destinadas à pesquisa e também para fins educacionais.

Embora existam soluções comerciais para essas plataformas de robôs móveis, ainda existem sérias limitações em relação à flexibilidade tanto em termos de *hardware* utilizado, como sensores e atuadores, como também em relação ao *software*, que costumam ser proprietários e fechados. Essas limitações se estendem em relação às necessidades de alterações de *hardware* que são comuns, como nos módulos de sensoriamento e de atuadores. Isso acaba influenciando de forma decisiva sobre a empregabilidade de tais soluções tanto no âmbito geral, como no científico e em iniciativas didáticas. Por outro lado, uma plataforma de robô móvel que apresente atributos de manobrabilidade, associada à facilidade de alterações físicas (de sensores e atuadores), além de uma arquitetura de controle, de *software* e de *hardware*, aberta, apresenta-se como uma ferramenta não apenas interessante como também essencial no desenvolvimento de aplicações relacionadas à robótica móvel, assim como no desenvolvimento de pesquisas científicas e no ensino de robótica e disciplinas relacionadas.

O Laboratório de Automação e Computação Evolutiva - LACE, do Departamento de Ciência da Computação e Estatística (DCCE), da Unesp de São José do Rio Preto, tem conduzido pesquisas em tópicos como navegação de robôs móveis, empregando diferentes tecnologias para o sistema de controle de robôs móveis, como FPGA, microcontroladores, dispositivos computacionais embarcáveis, dentre outras. Desta forma, tal interesse se resume a uma plataforma de robô móvel para uso geral, incluindo atividades didáticas, teóricas ou experimentais, assim como atividades científicas envolvendo a robótica móvel e tópicos relacionados. É desejável, portanto, que essa plataforma de robô móvel possua uma

arquitetura que ofereça facilidades para o desenvolvimento de aplicações envolvendo tópicos como navegação e controle.

Este trabalho tem como objetivo apresentar a pesquisa e o desenvolvimento de uma plataforma de robô móvel de baixo custo, de arquitetura de *hardware* e *software* aberta e que ofereça suporte a diferentes cenários e tecnologias de sistemas de controle através de uma interface padronizada de comunicação. A plataforma de robô móvel apresentada implementa conceitos simples e bem estabelecidos como arquitetura cliente-servidor, sistema fortemente acoplado e interface de comunicação serial. Devido a essa simplicidade conceitual envolvida, esta plataforma pode facilmente ser empregada tanto em aplicações de caráter geral, de iniciativas didáticas ou científicas. Além disso, a plataforma é capaz de ser modificada e expandida de acordo com as necessidades das aplicações devido à sua flexibilidade ainda que dentro de suas limitações. Assim, esta plataforma de baixo custo se apresenta como uma alternativa viável para a facilitação do processo de desenvolvimento de *software* de controle, uma vez que motiva o reaproveitamento de módulos de modo independente ao esquema ou à tecnologia do sistema de controle adotado.

Por fim, dois experimentos foram elaborados e realizados com o intuito de validar a plataforma apresentada, mostrando a arquitetura proposta empregada em duas iniciativas que abordam diferentes aspectos envolvendo a robótica móvel. Os *software* de controle da plataforma destas aplicações foram implementados utilizando a linguagem de programação C++, tanto para ambiente Windows quanto para Arduino.

1.1 Trabalhos correlatos

Existem outras iniciativas na literatura que propuseram ambientes e arquiteturas voltadas para o desenvolvimento de robôs móveis, que objetivaram a facilitação e a simplificação do processo de desenvolvimento de robôs móveis. É o caso de trabalhos como por exemplo, o OROCOS (BRUYNINCKX, 2001), o Player/Stage (GERKEY et. al., 2003) e o Pyro (*Phyton Robotics*) (BLANK et. al., 2003), que propõem ferramentas e ambientes de

programação robóticos, com bibliotecas de código aberto concebidas em diversas linguagens de programação, adquirindo assim, neutralidade em relação à plataforma. Já Bonarini et. al. (2013) apresentam o R2P (*Rapid Robot Prototyping*) com uma abordagem de arquitetura modular e aberta tanto de *hardware* como de *software*, objetivando prototipagem rápida e baixo custo. Existem outras abordagens, como é o caso do URB (*Universal Robot Bus*) (SARANLI et. al., 2011), na qual é apresentado um *framework* composto por uma arquitetura modular de barramento, um protocolo capaz de prover desempenhos em tempo real, e um conjunto de APIs (*Application Programming Interfaces*) associados às bibliotecas de interface, que somados, são capazes de prover modularidade, flexibilidade e desempenho em aplicações envolvendo robótica móvel e instrumentação. Uma das características que se mostra comum em todos esses trabalhos citados é a abstração obtida por *software* para compensar a heterogeinedade dos dispositivos de *hardware*.

1.2 Motivação

O crescente interesse por pesquisas na robótica móvel tem sido impulsionado pelos recentes avanços tecnológicos. Na literatura, existem diversas propostas que objetivaram a facilitação do processo de desenvolvimento de robôs móveis como, por exemplo, projetos citados em 1.1. Ainda que existam plataformas comerciais e proprietárias destinadas a tal finalidade, questões como flexibilidade e custo acabam dificultando a aplicabilidade de tais plataformas em iniciativas didáticas ou científicas.

O LACE é o Laboratório de Automação e Computação Evolutiva do Instituto de Biociências, Letras e Ciências Exatas (IBILCE da Unesp de São José do Rio Preto), responsável por pesquisas na área de robótica e visão computacional. Uma situação que se passa atualmente no LACE é o interesse por uma plataforma de robô móvel, de baixo custo e flexível para ser utilizada em atividades relacionadas à robótica móvel. Por ser destinada ao uso geral, esta plataforma poderia ser empregada também em pesquisas e atividades que auxiliem o aprendizado dos alunos da graduação. Para a pesquisa, esta plataforma de robô móvel poderia ser empregada para desenvolver e avaliar técnicas de navegação, controles de

tracionamento, arquiteturas de controle, visão computacional, colaboração e interação humano-robô, dentre outros tópicos. Já para o uso didático, esta plataforma poderia ser utilizada em atividades práticas voltadas aos alunos da graduação e relacionadas à robótica como eletrônica, circuitos digitais, assim como em algoritmos e inteligência artificial. Logo, o desenvolvimento de uma plataforma capaz de atender necessidades gerais e, que ainda possa explorar iniciativas de estudos, além de focar no baixo custo e simplicidade, pode contribuir de forma significativa tanto no processo de aprendizado como em atividades voltadas à pesquisa e outras que envolvam a robótica móvel.

1.3 Objetivos gerais

O objetivo geral do presente trabalho é pesquisar, desenvolver e avaliar uma plataforma de robô móvel de arquitetura de *hardware*, controle e *software*, aberta, simples, flexível e de baixo custo, destinada a propósitos gerais.

1.4 Objetivos específicos

Os objetivos específicos deste trabalho são:

- Investigar a natureza de um sistema robótico, os diversos aspectos como são tratadas as questões da tecnologia robótica e como têm sido as abordagens das diversas soluções e propostas apresentadas na literatura;
- Desenvolver uma plataforma flexível que facilite e flexibilize o processo de desenvolvimento de aplicações e *software* de controle de robôs móveis por meio de uma interface de comunicação padronizada e da abstração das camadas de aquisição de dados e de controle de atuadores;

- Implementar módulos essenciais de *hardware* destinados à locomoção e navegação para robôs móveis e seus respectivos módulos de controle.

1.5 Requisitos

Para que este trabalho possa cumprir todos os objetivos almejados, buscou-se satisfazer alguns requisitos funcionais, que serão brevemente descritos nesta seção.

- Flexibilidade de uso: como já mencionado, em relação à finalidade de uso, é pretendido disponibilizar uma plataforma de robô móvel para ser empregada em iniciativas gerais que envolvam robótica móvel. Para tal, este deve apresentar características que o torne flexível, mas ao mesmo tempo simples, possibilitando e motivando sua utilização também em outras atividades e iniciativas relacionadas. Em relação à arquitetura de *hardware*, o desenvolvimento e a construção da plataforma de robô móvel, considerou, por exemplo, a questão da adaptabilidade, de modo que diferentes esquemas de controle possam ser implementados e investigados, utilizando a mesma estrutura básica de *hardware* embarcada no robô. Foi concebido um número inicial de módulos de *hardware* padrão de modo que a plataforma ofereça suporte à iniciativas essenciais envolvendo à robótica móvel e, ainda, possibilitar alterações e acréscimos de módulos de *hardware*.

- Ambiente de aplicação e locomoção: a plataforma de robô móvel concebida será destinada a aplicações terrestres, em superfícies planas com poucas irregularidades. Logo, em termos estruturais mecânicos, seu sistema de locomoção deverá ser adequado a tal ambiente, e para o seu desenvolvimento. Sendo assim, foi escolhido um mecanismo de locomoção, levando em consideração questões como manobrabilidade, graus de liberdade, estabilidade e alguns aspectos que cada forma de locomoção terrestre acarreta, como, por

exemplo, aspectos relacionados à eficiência, consumo de energia, dimensões e custos da plataforma de robô móvel.

- Interface de comunicação e abstração: a plataforma de robô móvel deve oferecer acesso facilitado aos seus módulos de *hardware* através de uma interface de comunicação padronizada e simplificada, somada à abstração dos detalhes técnicos relacionados aos dispositivos de *hardware* implementados. A intenção no desenvolvimento da plataforma robótica é a redução da necessidade de conhecimentos avançados de *hardware* para o programador através de um certo grau de abstração, e ainda buscar independência da plataforma de controle escolhida. Uma vez que o *software* de controle se adeque ao protocolo de comunicação disponibilizado, o controle poderá ser embarcado, empregando tecnologias embarcáveis como FPGA, microcontroladores, Arduino, Raspberry PI ou, ainda, ser remoto, empregando tecnologias de transmissão de dados por RF, como por exemplo, Zigbee, Wi-Fi ou *Bluetooth*.

1.6 Escopo do trabalho

A proposta deste trabalho abrange o desenvolvimento, concepção e disponibilização de uma plataforma de *hardware* de robô móvel para ser destinada a propósitos gerais. A Figura 1.1 ilustra a plataforma proposta neste trabalho e sua abrangência.

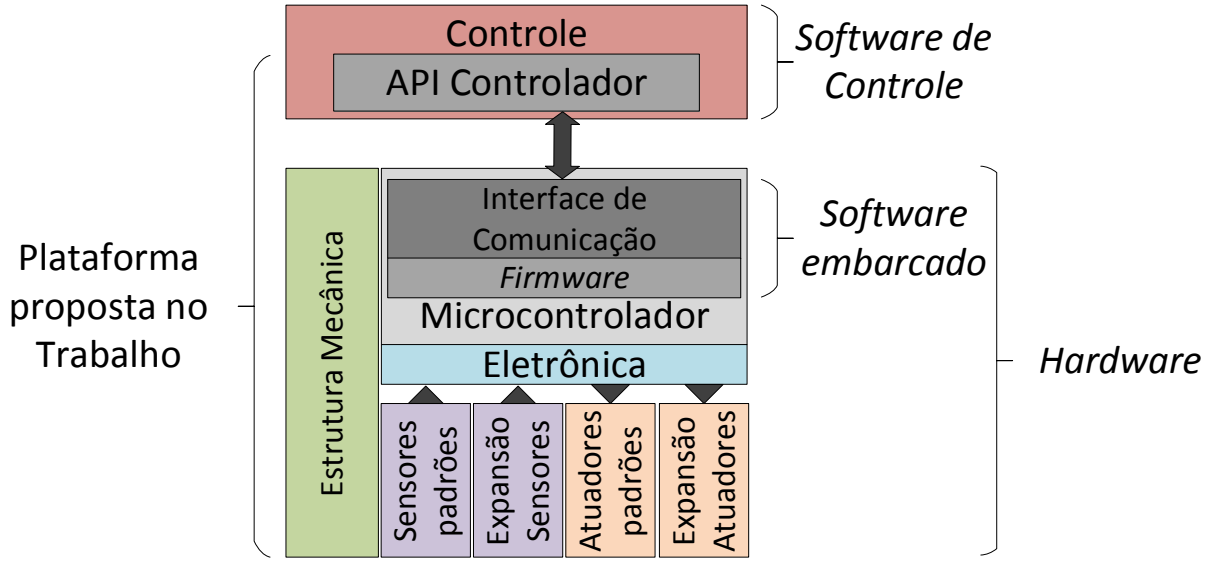


Figura 1.1 – Abrangência do trabalho proposto.

Fonte: Elaborado pelo autor.

A plataforma abrange a estruturação mecânica, eletrônica, escolha do microprocessador, programação do *software* embarcado (*firmware* e interface de comunicação), e API (*Application Programming Interface*) do controlador.

Ainda que seja desenvolvido um *software* de controle básico, este será de caráter unicamente para fins de testes de implementação e avaliação da plataforma. Quanto aos paradigmas de controle robótico, não serão abordados com profundidade as diversas concepções possíveis, como controle direto, indireto e sem estado, controle baseado em comportamento, máquina de estados finitos, arquiteturas de subsunção, e lógica fuzzy, já que essas discussões se encontram fora do escopo do que está sendo proposto neste trabalho. Ainda que o presente trabalho implemente algum método de navegação e de acionamento do sistema de tração, tais tópicos não serão aprofundados, devido às suas extensões. Desta forma, questões relacionadas às técnicas e algoritmos de navegação, por exemplo, ficarão abertas à discussão, podendo desta forma, serem objetos de estudo em trabalhos futuros. Tal raciocínio se estende também em termos do sistema de controle de tracionamento e locomoção, visando dar uma liberdade maior ao desenvolvedor do *software* de controle da plataforma robótica proposta.

1.7 Método

Para o desenvolvimento deste trabalho, o mesmo foi dividido nas etapas a seguir:

1. Levantamento de requisitos do trabalho;
2. Síntese da revisão bibliográfica;
3. Desenvolvimento e implementação da plataforma proposta;
4. Testes de validação;
5. Considerações finais.

A primeira etapa consta o levantamento dos requisitos do trabalho para tratar das especificações e funcionalidades que foram consideradas ao longo do desenvolvimento deste trabalho. O escopo do trabalho delimita a proposta dos aspectos e tópicos que não foram abordados neste trabalho.

A segunda etapa abrange o estudo da fundamentação teórica, das tecnologias envolvidas e do estado da arte da robótica móvel. A revisão bibliográfica abordou alguns dos aspectos fundamentais a respeito dos robôs móveis autônomos, uma vez que trata-se do tema principal e foco deste trabalho. Neste sentido, foi estudada a definição de robôs móveis autônomos, questões sobre ambiente de aplicação, locomoção, navegação, arquiteturas de controle e de *software* que dão suporte ao desenvolvimento de aplicações na robótica móvel, finalizando com um breve levantamento sobre comunicação de dados na robótica. Através dos estudos realizados envolvendo a parte conceitual da robótica móvel, foram embasadas e apresentadas as partes que foram escolhidas para compor a plataforma.

A terceira etapa trata do desenvolvimento e concepção da plataforma de robô móvel. Em termos de *hardware* foram definidos os tipos de dispositivos essenciais de sensoriamento e atuação que foram implementados no sistema robótico, abrangendo o sistema de comunicação de dados e *software* embarcado. Foram apresentadas as escolhas dos aspectos

mecânicos, interface física de comunicação e seu protocolo, e por fim, a API da plataforma que facilitará o processo de desenvolvimento de aplicações. Nesta etapa foram brevemente analisadas algumas possibilidades de linguagem de programação, ambientes de desenvolvimento e tecnologias adotadas. A importância desta etapa se deve à avaliação das diversas possibilidades existentes, do sistema operacional à linguagem de programação e sistemas de comunicação, incluindo os ambientes de desenvolvimento e ferramentas. Inclui ainda uma breve descrição das tecnologias adotadas para a construção da parte estrutural mecânica, dos dispositivos periféricos (sensores e atuadores), concepção do Controlador, Interface de Comunicação e sua API de suporte. Foi decidido expor no Apêndice C o detalhamento do protocolo de comunicação da plataforma de robô móvel concebida.

A quarta etapa é destinada aos testes e validações da plataforma apresentada, ou seja, para verificar se os objetivos almejados foram alcançados com a elaboração de dois experimentos. Cada experimento buscou verificar diferentes propriedades da plataforma. O primeiro experimento avaliou a capacidade da plataforma de oferecer suporte à uma aplicação genérica envolvendo navegação, assim como a facilidade de uso da API de plataforma. Já o segundo experimento, sua flexibilidade em relação às tecnologias de controle e facilidade de alterações de *hardware* por meio de uma aplicação científica hipotética.

Finalmente, a quinta e última etapa, consiste na apresentação das conclusões da presente dissertação e discussões sobre algumas possibilidades de trabalhos futuros.

1.8 Organização do trabalho

O texto desta Dissertação de Mestrado foi organizado da seguinte forma: o capítulo 1 trata da apresentação do trabalho; o capítulo 2 discute seu arcabouço teórico; o capítulo 3 apresenta a plataforma de robô móvel proposta e o robô móvel Kihon; o capítulo 4 apresenta os experimentos de validação; o capítulo 5 encerra a dissertação. O conteúdo de cada um dos capítulos mencionados acima é descrito a seguir, compondo o texto da dissertação.

O capítulo 1 apresenta o tema do trabalho proposto, trabalhos correlatos, seus objetivos gerais e específicos, seguido pelos requisitos, além de apresentar o método e a organização deste trabalho.

O capítulo 2 apresenta uma síntese da revisão bibliográfica na qual analisa sucintamente alguns aspectos da robótica móvel. Nela, é apresentada uma definição de robôs móveis autônomos; a composição de um robô móvel, tratando brevemente sobre ambiente de aplicação, locomoção e navegação. Ainda apresenta o levantamento realizado a respeito das arquiteturas de controle e de *software* no contexto da robótica móvel e encerrando com os sistemas de comunicação na robótica.

O capítulo 3 introduz e discute a plataforma proposta neste trabalho, implementada por meio da concepção do robô móvel Kihon. Apresenta a arquitetura da plataforma escolhida, o sistema de comunicação de dados, e a API que dará suporte ao desenvolvimento de aplicações. Nesta etapa foram relevados e definidos os aspectos tecnológicos empregados, evidenciando desta forma como a plataforma dará suporte aos objetivos almejados neste trabalho e como cada módulo foi desenvolvido.

O capitulo 4 apresenta os experimentos práticos realizados com a plataforma de robô móvel concebida com o intuito de averiguar se esta é capaz de satifazer os objetivos almejados, com a discussão dos objetivos, desenvolvimento e resultados dos experimentos.

Por fim, o capítulo 5 finaliza esta dissertação por meio das considerações finais acerca do trabalho realizado, e apresenta algumas das possibilidades de trabalhos futuros.

2 SÍNTESE DA REVISÃO BIBLIOGRÁFICA

A robótica, ou seja, o estudo do robô, é um campo relativamente novo da tecnologia moderna (SPONG et. al., 2005). E de fato, suas raízes incluem fundamentos de diversas áreas tradicionais do conhecimento, dentre as quais é possível citar a engenharia elétrica, engenharia mecânica, mecatrônica, ciência da computação e matemática. Ainda é possível mencionar áreas como a da psicologia cognitiva e neurociência, por estudar a forma como organismos biológicos solucionam problemas similares (DUDEK; JENKIN, 2010), assim como outros trabalhos na robótica inspirados na biologia, como os apresentados na obra de Habib (2007). Isso faz da robótica um campo intrinsecamente multidisciplinar e extenso. Assim, existem diferentes formas de abordar as questões que envolvem a robótica e, portanto, esta pode vista por diferentes perspectivas. E ainda, a construção de um sistema autônomo móvel sempre conduz a uma interação de sensores e efetores com o software responsável pelo controle (KUBITZ et. al., 1998). Observando as plataformas robóticas comerciais disponíveis e as encontradas na literatura revela uma grande variedade de arquiteturas de *software* disponíveis na robótica. De fato, apesar dos avanços tecnológicos recentes e das diversas propostas na literatura, houve pouco avanço no estabelecimento de uma padronização dos sistemas de controle utilizados, assim como não há sistema operacional robótico estabelecido (SIEGWART; NOURBAKHSH, 2004), uma vez que robótica móvel é uma área recente em comparação às areas tradicionais do conhecimento. Este capítulo de revisão bibliográfica apresenta um estudo a respeito da robótica móvel sob a perspectiva da computação.

2.1 Robôs móveis autônomos

Assim como em muitas outras áreas da ciência e tecnologia, não há uma padronização e nem mesmo um consenso no que diz respeito à própria definição para robô. A definição oficial de robô dada pela Robotic Industries Association (RIA) e mencionada em Spong et. al.

(2005) pode ser entendida como “um manipulador multifuncional reprogramável desenvolvido para mover materiais, peças, ferramentas, ou dispositivos especializados por meio de movimentos programados para executar uma variedade de tarefas”. Entretanto, esta definição se volta basicamente a robôs industriais. Já Bekey (2005), ainda que vagamente, apresenta uma breve definição de robô como “uma máquina que pensa e age”. No entanto, essa definição abrangente pode caracterizar sistemas automatizados que não são considerados robôs. Uma outra definição dada por Matarić (2007) descreve robô como "um sistema autônomo que existe no mundo físico e que pode sentir seu ambiente e atuar sobre ele de modo a alcançar alguns objetivos". A partir de tal definição, é possível deduzir que um robô possui, além de um corpo físico, formas de sensoriamento e atuadores que o permitem, por meio de um sistema de controle autônomo, coordenar seus recursos de forma a cumprir determinadas tarefas.

No entanto, tal definição ainda engloba dispositivos robóticos, como os considerados manipuladores, assim como os robôs móveis. No manipulador (braço robótico), o robô efetua operações de manipulação dentro do seu limite de alcance (volume de trabalho), uma vez que sua base permanece estacionária, ou seja, fixa. A Figura 2.1 (a) ilustra um exemplo de robô manipulador. Por outro lado, na robótica móvel, o robô é capaz de se locomover através de um determinado ambiente, o que expande o campo de operação. Levando isso em consideração, a robótica móvel enfatiza questões relacionadas ao entendimento de ambientes substancialmente maiores se comparado a áreas como a robótica de manipulação, visão computacional ou inteligência artificial, pois estas últimas apresentam ambientes vistos a partir de uma posição vantajosa (DUDEK; JENKIN, 2010). A Figura 2.1 (b) ilustra um exemplo de robô móvel.

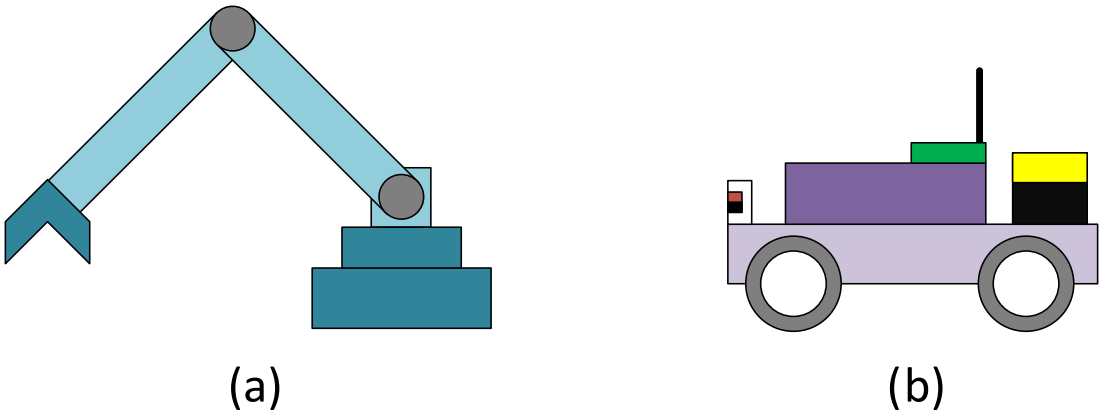


Figura 2.1 - Exemplos de robôs: (a) Robô manipulador; e (b) Robô móvel.

Fonte: Elaborado pelo autor.

Para que o robô móvel consiga se locomover de forma satisfatória através de um determinado ambiente, é necessário que o mesmo possua mecanismos que o torne capaz de se mover e coletar informações a respeito do ambiente no qual está inserido. Além disso, deve reconhecer eventuais objetos e obstáculos para, então, tornar possível sua navegação no ambiente de forma harmoniosa e livre de colisões. Já para executar uma determinada tarefa, o robô necessita de um sistema de controle para coordenar de forma sincronizada a utilização de seus módulos sensores e atuadores. Desta forma, para sua concepção, deverá ser levada em consideração a estruturação de seu corpo, de forma que seja capaz além de se deslocar, como interagir adequadamente com objetos, outros robôs ou seres vivos.

No contexto da robótica, na qual a autonomia pode ser tratada em termos de regime operacional, uma definição interessante pode ser obtida segundo Barber e Martin (1999) que a descreve como "uma capacidade ativa do robô em perseguir seus objetivos sem a intervenção de nenhum outro agente nos processos de tomada de decisão que serão usados para determinar como tais objetivos deverão ser perseguidos". Uma outra definição de autonomia pode ser entendida como a habilidade de um agente ou indivíduo de tomar suas próprias decisões e agir de acordo elas (MATARIĆ, 2007). Além disso, não é incomum que robôs sejam comparados a sistemas biológicos vivos, uma vez que compartilham os mesmos desafios no desenvolvimento de suas tarefas (BEKEY, 2005). Afinal, conforme Medeiros (1998), “um robô móvel autônomo deve ser extremamente autossuficiente para operar em

ambientes complexos, desafiadores e parcialmente conhecidos, utilizando limitados recursos físicos e computacionais". E de acordo com Dudek e Jenkin (2010), a autonomia na robótica pode ser total ou parcial, onde, na autonomia total, o sistema é projetado de forma a operar sem a necessidade de ser controlado todo o tempo por humanos e, na autonomia parcial, esse controle humano é requerido constantemente. Ainda, conforme os autores, existem poucos robôs móveis que podem ser considerados totalmente autônomos operacionalmente.

2.2 Composição de um robô móvel

A partir da definição de robô móvel autônomo dada por Matarić (2007) é possível ter uma noção dos principais componentes que o compõe. É esperado que o robô possua um corpo físico para que possa efetuar alguma tarefa no mundo físico. Sensores são fundamentais para que o robô seja capaz de sentir e perceber o ambiente. Da mesma forma, atuadores são essenciais para que o robô consiga atuar sobre o ambiente. Por último, para que o robô seja autônomo, um sistema de controle não supervisionado se faz necessário.

Um robô deve existir e atuar no mundo físico (BROOKS, 1991), pois apenas o mundo real apresenta a riqueza de desafios a serem superados pelo projetista do robô, ao contrário dos simuladores, que são limitados pela complexidade do modelo do mundo em sua implementação por meio de *software*. Como consequência, os robôs devem obedecer às mesmas leis físicas as quais todos se submetem, da mesma forma como os seres humanos, animais, objetos e outros materiais existentes. Isso inclui os mesmos cuidados em relação às colisões, quedas, objetos e obstáculos. Em suma, possuir um corpo adequado que o permita interagir adequadamente com o ambiente e tudo o que se encontra nele também se torna um aspecto fundamental e que poderá definir o sucesso ou o fracasso de uma base mecânica robótica. Portanto, é fundamental conhecer e definir o ambiente em que o robô será inserido. Além disso, o processo de desenvolvimento de um robô deve considerar tanto limitações tecnológicas como financeiras.

De modo a objetivar um sistema robótico móvel com elevado nível de autonomia operacional e para torná-lo capaz de operar em ambientes desestruturados, é necessário satisfazer um nível de independência de forma que o robô tenha conhecimento a respeito do que existe ao seu redor, adquirindo e manipulando um modelo detalhado do ambiente de aplicação. De forma análoga aos animais, que possuem seus próprios mecanismos sensoriais muito bem adaptados e adequados às suas necessidades, os robôs necessitam igualmente de mecanimos que possibilitem sentir não somente o ambiente, mas também coletar informações a respeito de si mesmo. Para tal, existe a necessidade de uma variedade de sensores para ser capaz de interagir com o mundo real, e de mecanismos para extrair informações significativas a partir dos dados coletados (ELFES, 1986). Conforme Matarić (2007), na robótica, os sensores são dispositivos físicos que permitem aos robôs coletar informações a respeito de si próprios (proprioceptivos) e, também, do ambiente que os cercam (exteroceptivos). E ainda, de acordo com a autora, os termos sentir e perceber são sinônimos no contexto da robótica. A partir da ponderação de aspectos eletrônicos, processamento de sinal e computacional, a autora ainda categoriza os sensores de acordo com três níveis de processamento, sendo estes: nível baixo (sensores de contato e odometria); nível médio (sonares e voz) e nível alto (câmeras). Os sensores eletrônicos podem ser classificados em dois diferentes tipos, de acordo com o tipo de sinal fornecido em sua saída: digital e analógico. Sensores digitais podem fornecer uma representação binária de um dado fenômeno, por exemplo, um botão de contato está pressionado ou não, ou transmitir a medida por meio de um protocolo de comunicação. Sensores analógicos necessitam de conversão para a forma digital, ainda que possam ser conectados às entradas de circuitos de conversão analógico para digital, ou seja, conversores AD. Sensores que necessitam processamento de sinal exigem uma etapa de análise e modificação sobre o sinal coletado para obter informações e, assim, torná-los adequados para uma dada aplicação. Já os sensores relacionados à computação necessitam de complexos processamentos envolvendo visão computacional, ou seja, como os sistemas artificiais (máquinas) obtêm informações a partir de imagens ou dados multidimensionais.

Devido às limitações intrínsecas apresentadas por qualquer tipo de sensor, existe a necessidade de utilizar múltiplos sensores e combinar suas informações obtidas para, então,

estimar um modelo de representação coerente do mundo real. Desta forma, a escolha dos mecanismos adequados, assim como o devido posicionamento físico dos mesmos e a combinação de múltiplos sensores, possibilitam ao robô "sentir" as informações necessárias para que ele possa executar suas tarefas de forma satisfatória.

Os atuadores na robótica são dispositivos físicos responsáveis pela atuação do robô dentro do seu ambiente de aplicação. Na robótica móvel, os atuadores também estão intimamente ligados ao sistema de locomoção (SICILIANO; KHATIB, 2008). Os efetuadores permitem que os robôs atuem fisicamente sobre seus ambientes, seja para se locomoverem ou para manipularem objetos. Os efetuadores são compostos por mecanismos básicos chamados atuadores (MATARIĆ, 2007). Na biologia animal, é possível relacionar os efetuadores robóticos às pernas, asas, bocas e outras partes que permitam a um dado animal atuar sobre o ambiente. Nos animais, os atuadores são como os músculos e tendões que compõem braços e pernas. Na robótica, atuadores podem ser atuadores elétricos, cilindros pneumáticos ou hidráulicos, entre outros dispositivos, embora na robótica móvel sejam utilizados basicamente atuadores elétricos (motores) uma vez que possibilitam a utilização de baterias para a sua alimentação.

O sucesso de um sistema robótico é influenciado não apenas pelo custo da plataforma, mas também pela qualidade e disponibilidade de seus módulos e componentes no mercado. Logo, é desejável que o *hardware* do robô móvel, assim como seus componentes elétricos e mecânicos, sejam reaproveitáveis, em outras palavras, flexíveis para satisfazer diferentes requisitos e também permitir rápidos ajustes a novas aplicações (MERTEN; GROSS, 2008). A reutilização pode tornar o desenvolvimento muito mais rápido, além de reduzir substancialmente o índice de erros. No entanto, ainda de acordo com os autores, "a reutilização do *hardware* em sistemas robóticos, assim como a possibilidade de adicionar componentes, ainda continua muito restrita nas soluções propostas na literatura". De fato, como os componentes de *hardware* empregados na robótica costumam ser altamente especializados e adaptados às aplicações finais, isso acaba dificultando na hora de adicionar novos módulos de *hardware*. A solução de adotar arquiteturas de *hardware* modulares, em outras palavras, em subsistemas, ainda que seja uma abordagem válida, não se mostra

flexível suficiente, além de possuir um pequeno número de funcionalidades (MERTEN; GROSS, 2008). Por outro lado, as dimensões e espaços bem definidos na base mecânica do robô podem facilitar a adição e adaptações exigidas na inserção de mais sensores e atuadores e, assim, ampliar os recursos de um robô, possibilitando o reaproveitamento de sua base robótica.

A utilidade de um agente robótico autônomo é sua habilidade de executar uma tarefa que lhe foi atribuída, sem a necessidade de supervisão constante (NEVES; OLIVEIRA, 1997). O grau de autonomia na robótica está diretamente relacionado à quantidade de sensores de aquisição de dados, processamento e tomada de decisão (SANTIAGO; MEDEIROS, 2011). Na robótica móvel, a autonomia proporciona tanto o planejamento como as tomadas de decisões do robô móvel e está diretamente associada ao sistema de computação. E, geralmente, o sistema de computação fica atrelado a um sistema de comunicação, conferindo assim, a capacidade de se configurar uma arquitetura distribuída (DUDEK; JENKINS, 2010) ou, ainda, de embarcar sistemas computacionais.

Uma outra forma de descrever a composição de um robô móvel é por meio da modularização. Para facilitar o entendimento e o gerenciamento de sistemas complexos, é comum utilizar a técnica de segregação, dividindo um sistema complexo em subsistemas menores e mais simples. Aplicando este conceito na robótica, é possível efetuar sua decomposição em duas categorias de módulos: físicos e lógicos. Módulos físicos representam a parte do robô móvel responsável pela sua manifestação no mundo real, abrangendo estruturas mecânicas, sensores, atuadores, e sistemas embarcados. Já os módulos lógicos compreendem recursos essenciais de *software* para processar os dados provenientes do módulo físico (NIRANJAN et. al., 2012), como por exemplo as arquiteturas de controle e sistemas de navegação. De fato, robôs autônomos são sistemas complexos que requerem a integração e interação eficiente entre numerosos componentes heterogêneos, tanto de *hardware* como *software*.

2.3 Ambiente de aplicação

Um robô móvel, ao contrário dos agentes de *software*, que habitam em um ambiente simulado (como discutido em 2.2) deve ser capaz de executar suas tarefas no mundo real (KRAMER; SCHEUTZ, 2007), e especificamente, em seu ambiente de aplicação no qual foi destinado a operar. Portanto, o processo de elaboração do corpo de um robô móvel deve ser feito de modo a buscar a forma que melhor se adapta aos requisitos da aplicação. Para tal, é fundamental conhecer o ambiente de aplicação. Existem diferentes ambientes nas quais um robô móvel pode ser projetado para atuar, dentre os quais é possível citar: ambiente aquático (seja na superfície ou submerso), aéreo ou terrestre. E de modo que o robô móvel seja capaz de se locomover em um determinado ambiente, tanto sua composição como a disposição de seus mecanismos devem permitir que o robô móvel coexista e, ainda, oferecer-lhe a habilidade de se deslocar e navegar satisfatoriamente dentro desse ambiente. Além disso, o robô deve ser capaz de executar sua tarefa adequadamente.

Desenvolver um robô móvel é um processo criativo e ao mesmo tempo desafiador para o projetista, uma vez que o força a pular etapas que o sistema natural de evolução levou milhões de anos para atingir resultados ótimos. Desta forma, o processo de desenvolvimento do corpo robótico deve levar em consideração aspectos estruturais e mecânicos necessários para que o robô móvel consiga alcançar seus objetivos quando inserido em seu ambiente de aplicação. É comum ainda procurar inspirações em áreas como biologia, em busca de soluções particulares (HABIB, 2007). Trata-se de uma tarefa multidisciplinar e complexa, já que abrange conhecimento de diferentes áreas do conhecimento e, muitas vezes, exige soluções criativas por parte dos desenvolvedores.

Além disso, o desenvolvimento de um robô móvel também depende de outros fatores, como a tecnologia disponível e as limitações econômicas. Desta forma, no processo de desenvolvimento deve-se ter em mente o custo financeiro e a tecnologia disponível no momento. Além disso, a escolha de seus componentes deve ser feita sem perder de vista o propósito do robô. Isso inclui o dimensionamento adequado de seu corpo e de seus componentes, incluindo considerações relacionadas às suas especificações, como consumo

energético, dimensões, formas, materiais e pesos, e ainda, sem perder de vista o custo. Um outro aspecto importante é a manutenibilidade, ou seja, a facilidade que o sistema oferece à execução de manutenção.

Centrando nos requisitos levantados em 1.5, este trabalho se limitou a abordar os aspectos relevantes para a estruturação física e mecânica de um robô móvel destinado ao ambiente terrestre, cujas superfícies são planas e com poucas irregularidades.

2.4 Locomoção

Na robótica móvel, a locomoção refere-se ao modo como o robô se move de um lugar a outro (MATARIĆ, 2007). Como mencionado em 2.3, os robôs móveis podem se locomover sobre o solo, na água e no ar com o uso de mecanismos adequados. Existe uma variedade de formas possíveis de se fazer isso e muitas delas se baseiam em sistemas biológicos encontrados na natureza, sejam aéreos, aquáticos ou terrestres. Robôs móveis aquáticos são os que permanecem sobre a água, enquanto que os capazes de locomover sob a superfície são categorizados como sub-aquáticos. Robôs móveis aéreos e sub-aquáticos são mais difíceis de se controlar, uma vez que podem se mover em mais dimensões (3D), assumindo posicionamentos mais complexos, uma vez que, quanto maior o grau de liberdade (do termo em inglês: *Degree Of Freedom*, DOF), mais difícil se torna o controle.

No caso da locomoção em ambiente terrestre é possível citar mecanismos como andar, pular, deslizar e rolar, que são bastante comuns na natureza. No entanto, há uma exceção cuja invenção é atribuída ao ser humano, e que alcança elevado nível de eficiência para locomoção em superfícies planas: a roda (SIEGWART; NOURBAKHSH, 2004). As rodas se apresentam como uma alternativa em relação às outras formas de locomoção, como as encontradas na natureza. O sistema de locomoção baseado em rodas explora a fricção do ponto de contato de suas rodas com o chão para deslocar. Tal sistema de locomoção apresenta alguns aspectos interessantes, como maior eficiência em relação aos mecanismos de locomoção baseados em pernas, baixa complexidade na implementação e despreocupação com o balanço do robô.

Três rodas em contato permanente com o chão é o suficiente para estabilizar o equilíbrio. O uso de quatro ou mais rodas necessitam de sistemas de suspensão para manter o contato de todas as rodas com a superfície. Com a utilização de rodas, a preocupação se volta basicamente a problemas de tracionamento, estabilidade, manobrabilidade e controle.

Robôs móveis com rodas são mais eficientes em termos de consumo de energia em superfícies planas e lisas, quando comparados a robôs móveis que utilizam pernas e esteiras (KALE; SHRIRAMWAR, 2009). Além disso, requerem componentes relativamente mais simples, o que os tornam mais fáceis e baratos de se construir. E a tarefa de controlar a tração das rodas é menos complexa que a de controlar e coordenar os atuadores de pernas com múltiplas articulações. Entretanto, em terrenos irregulares, sistemas robóticos que utilizam pernas se apresentam mecanicamente superiores em termos de locomoção quando comparados a sistemas baseados em rodas e esteiras (HABIB, 2007). Siegwart e Nourbakhsh (2004) afirmam que, geralmente, a locomoção por pernas requer maior grau de liberdade, e portanto, apresenta maior complexidade mecânica do que a locomoção por rodas. É possível observar que a escolha da abordagem adotada, assim como o devido direcionamento à aplicação, são aspectos importantes na concepção do sistema de locomoção de um robô móvel. Questões relacionadas à locomoção robótica baseada em pernas não serão aprofundadas por se tratar de um tópico relativamente extenso e complexo que foge do escopo da proposta e, ainda, questões relevantes relacionadas a robôs móveis baseados em pernas podem ser encontradas em Habib (2007).

Diferentemente da robótica de manipulação, a robótica móvel traz consigo um grande desafio que é a estimação do posicionamento, já que de acordo com (SIEGWART; NOURBAKHSH, 2004), “não existe uma forma direta ou instantânea de determinar o posicionamento de um robô móvel”. Logo, segundo os autores, para estimar a movimentação do robô, “deve-se recorrer à compreensão e modelagem cinemática do modelo de locomoção adotado”. A robótica móvel deve necessariamente recorrer à cinemática para compreender sua locomoção. A cinemática é o estudo do comportamento de sistemas mecânicos, sendo, portanto, essencial no projeto mecânico do robo móvel para torná-lo apto a cumprir suas tarefas e também para desenvolver o *software* de controle responsável por sua locomoção.

Ainda, conforme Siegwart e Nourbakhsh (2004), no caso dos robôs móveis baseados em rodas, este processo começa com a descrição da contribuição que cada roda oferece à movimentação. Um robô que possui múltiplas rodas acarreta múltiplas formas de manipular essas rodas. Muitos robôs móveis empregam um mecanismo de locomoção chamado tração diferencial (*differential drive*), no qual duas rodas ativas são alinhadas a um mesmo eixo, empregando uma ou duas rodas passivas para apoio. A tração diferencial consiste na habilidade de controlar o acionamento das rodas de modo individual e independente, de modo que sua resultante movimenta e manobra o robô. O termo diferencial significa que a rapidez com que o robô executa uma manobra é determinada pela diferença entre a velocidade das duas rodas ativas. Uma análise detalhada a respeito das modelagens cinemáticas e suas restrições pode ser encontrada em Siegwart e Nourbakhsh (2004) e também em Dudek e Jenkin (2010).

Ainda em relação à locomoção, existe a questão da estabilidade, uma vez que a maioria dos robôs não pode balançar, inclinar e nem tombar (MATARIĆ, 2007). A estabilidade está associada ao centro de massa dos corpos e pode ser estática ou dinâmica. Na estabilidade estática, o robô pode facilmente permanecer parado sem tombar, no entanto, exige que o corpo do robô possua formas de prover pontos de contatos suficientes para isso. Já na estabilidade dinâmica, o robô deve ativamente se balancear ou mover-se para manter-se estável.

Existem algumas métricas utilizadas para avaliar objetivamente a capacidade de locomoção de sistemas robóticos baseados em rodas. No trabalho de Thueer (2009), algumas métricas são apresentadas no intuito de comparar os diferentes arranjos de rodas em terrenos variados. Estas métricas, que estão sintetizadas na Figura 2.2, não serão usadas no decorrer deste trabalho por fugir de seu escopo.

Figura 2.2 – Métricas de avaliação apresentadas por Thueer (2009).
Fonte: Adaptado de Thueer (2009).

2.5 Navegação

Na locomoção autônoma existe uma variedade de atividades para solucionar problemas, como o planejamento do caminho de curto e longo prazo, desvio de obstáculos, detecção de emergências, dentre outras (ELFES, 1986). E essas diferentes atividades são resolvidas por módulos que executam serviços específicos. A navegação na robótica móvel pode ser sumarizada em 3 questões: "onde estou?", "para onde vou?" e "como chego lá?" (LEONARD; DURRANT-WHYTE, 1991).

A primeira questão "onde estou?" se refere a uma técnica de localização que retorna a posição do robô dentro do ambiente. A localização é um dos problemas fundamentais na robótica móvel e no planejamento de caminho (GANGANATH; LEUNG, 2012), uma vez que o robô móvel não é capaz de se mover com precisão (BETKE; GURVITS, 1995), de forma que a sua posição e orientação real nunca é a mesma da posição e orientação desejada.

As técnicas de localização podem ser categorizadas em dois grupos: relativas e absolutas (BORENSTEIN et al., 1996). Técnicas de navegação relativas fornecem dados de

posicionamento em relação à localização inicial do robô. Por outro lado, técnicas de navegação absolutas fornecem dados de posicionamento em relação ao ambiente como um todo. Técnicas combinadas entre esses dois grupos podem, de forma complementar, aumentar a precisão da estimativa de localização do robô.

Borenstein et. al. (1996) categoriza as técnicas relativas de localização entre:

Odometria. Trata-se de uma das técnicas mais comuns para se estimar o quanto um robô móvel se deslocou (DUDEK; JENKIN, 2010). A odometria é feita por meio de encoders ou seja, codificadores ópticos, que medem a quantidade de rotação efetuada pelas rodas ou pelos motores, para então, o sistema estimar a distância percorrida. Trata-se de uma técnica simples do ponto de vista matemático. No entanto, devido à sua imprecisão, sua taxa de erro é cumulativa ao longo do percurso, não sendo adequado para longas distâncias.

Navegação inercial. Este método utiliza acelerômetros e giroscópios para medir a taxa de rotação e aceleração para, então, estimar a variação do deslocamento de um robô móvel. Assim como na odometria, não é indicado para uso por longos percursos já que os erros se acumulam ao longo do deslocamento em função de sua imprecisão, pois necessita integrar os dados medidos para obter a localização.

Já as técnicas absolutas podem ser:

Marcadores ativos (*active beacons*). Este método estima a posição global do robô, por exemplo por triangulação utilizando ultrasom, luz estruturada ou RF a partir da recepção de sinais de pelo menos três emissores. Como métodos recentes de localização global, é possível utilizar tecnologias de comunicação de dados sem fios para uso em ambientes internos, como o *Bluetooth*, proposto por Raghavan et. al. (2010), o *Wi-Fi* ou o Zig-Bee, proposto por Rodrigues et. al. (2012). Tais métodos utilizam o modelo de propagação de sinal de RF em ambientes fechados para estimativar a localização do robô móvel.

Bússolas magnéticas. Teoricamente são capazes de localizar um robô móvel em qualquer parte do globo terrestre, em virtude do campo eletromagnético natural do planeta. No entanto, apresenta problemas de precisão em ambientes internos, uma vez que o campo

eletromagnético terrestre pode sofrer distorções nas proximidades de objetos metálicos e cabos de redes elétricas.

Casamento de modelos. Neste método, a estimativa da localização é obtida com a comparação das informações adquiridas por sensores embarcados no robô com um mapa ou modelo de representação do ambiente, seja geométrico ou topológico. Na representação geométrica, o ambiente é representado por um sistema de coordenadas globais enquanto que na representação topológica, o ambiente é representado por meio de uma rede de nós e vértices.

Pontos de referência (*landmarks*). Este método utiliza características, sejam naturais (LEONARD; DURRANT-WHYTE, 1991) ou artificiais (MEYER-DELIUS, 2011) do ambiente para servir como pontos de referência ao robô móvel. O ambiente deve ser conhecido antecipadamente (BOREINSTEIN et. al., 1996), seja no uso de pontos naturais ou artificiais.

Sistema de Posicionamento Global (GPS). Utiliza a rede de satélites artificais como marcadores ativos. O GPS é amplamente utilizado para rastrear objetos em ambientes externos, uma vez apresenta boa estimativa de localização em locais abertos. Dispositivos GPS que apresentam alguns cm de imprecisão possuem elevado custo atualmente. Já os dispositivos GPS de custo acessível no mercado possuem imprecisão da ordem de alguns metros (PESSIN et. al., 2011). E segundo Rodrigues et. al. (2012), o GPS não é apropriado para ambientes internos.

Para melhorar a precisão dos sistemas de localização, é comum a utilização de técnicas probabilísticas, como o filtro de Kalman (RODRIGUES et. al., 2012), cadeias de Markov, e o método de Monte Carlo (MEYER-DELIUS et. al., 2011), assim como há propostas que utilizam redes neurais artificiais (PESSIN et. al., 2011).

A segunda questão "para onde vou?" está relacionada diretamente com a aplicação do robô móvel. Usualmente, assume-se que as técnicas de navegação sejam implementadas uma vez que se conhece a aplicação do robô e seu ambiente (ALVES et. al., 2012). Já a terceira questão "como chego lá?", diz respeito ao planejamento do caminho, ou seja, traçar a rota que o robô móvel deverá seguir para cumprir sua tarefa. Considerando que a proposta deste

trabalho é o desenvolvimento de uma plataforma de robô móvel que ofereça suporte às técnicas de navegação, ou seja, independe da aplicação, uma discussão a respeito dessas duas últimas questões fogem do escopo deste trabalho.

2.6 Arquiteturas de controle

De acordo com Matarić (2007), o termo *arquitetura* na robótica pode ser empregado da mesma maneira que na computação, uma vez que refere-se à concepção de um dispositivo complexo a partir de um conjunto de blocos bem compreendidos. De fato, a arquitetura tem sua utilidade na robótica com a organização formal de blocos e ferramentas disponíveis, de modo a facilitar o desenvolvimento e o entendimento de um sistema robótico.

A arquitetura de controle é uma parte essencial da robótica móvel, por prover um método formal na organização do sistema de controle (MATARIC, 1992), incluindo importantes aspectos da reatividade e arquiteturas de subordinação ou subsunção (do termo em inglês, *subsumption*) (BROOKS, 1986), e também por determinar o fluxo de informações dentro de um sistema robótico móvel (MURPHY, 2000). Portanto, a arquitetura de controle define os componentes arquiteturais do robô móvel e como tais componentes se comunicam e se relacionam.

O controle de um robô pode ser implementado diretamente em *hardware* ou em *software*. No entanto, quanto maior a complexidade do controle, há uma maior tendência que este seja implementado em *software* (MATARIC, 2007), devido à sua flexibilidade e facilidade de programação. O *hardware* tende a não oferecer flexibilidade ou adaptabilidade em vista do seu elevado grau de especialização. Assim, o controle robótico implica na utilização de programas (*software*), sendo executados em tempo real em sistemas computacionais embarcados ou remotos. No caso remoto, a comunicação pode ser estabelecida com a adoção de tecnologias de conectividade sem fio (*wireless*) por RF. Entretanto, a comunicação sem fio por RF não garante conectividade confiável e em todas as situações por diversas razões, logo, não se apresenta como uma alternativa confiável por completo. Por esta razão, é mais

confiável embarcar o sistema computacional responsável pelo controle no robô móvel para garantir tal conectividade.

Na robótica, assim como na computação, a tarefa de solucionar um determinado problema é realizada por meio de programas. Uma sequência finita de instruções e procedimentos, executados passo a passo, é chamado de algoritmo. Um algoritmo pode ser executado por uma pessoa, autômato ou por um sistema computacional. A robótica também recorre ao uso de algoritmos para executar tarefas específicas, como de locomoção, navegação, manipulação, dentre outras. Como a programação robótica faz uso da programação computacional para executar suas tarefas, ela pode ser implementada nas mesmas linguagens de programação utilizadas na computação. As linguagens de programação, como ferramentas de geração de algoritmos e programas, podem se apresentar nos diversos padrões e usos. Portanto, não existe uma linguagem de programação ótima ou ideal para a robótica, uma vez que existem linguagens destinadas a usos genéricos e outras, desenvolvidas para uma aplicação específica. Devido à diversidade de arquiteturas de controle, Mataric (2007) afirma que "o que realmente importa não é a linguagem utilizada para programar o robô, mas a arquitetura de controle usada para implementar o controle".

A classificação das arquiteturas de controle na robótica móvel em relação à organização da inteligência diz respeito à relação entre três primitivas: sentir, planejar e agir (MURPHY, 2000). A partir das primitivas da primeira visão, Murphy (2000) apresenta 3 paradigmas:

- Controle Hierárquico;
- Controle Comportamental;
- Controle Híbrido.

A Figura 2.3 ilustra os três paradigmas apresentados por Murphy (2000).

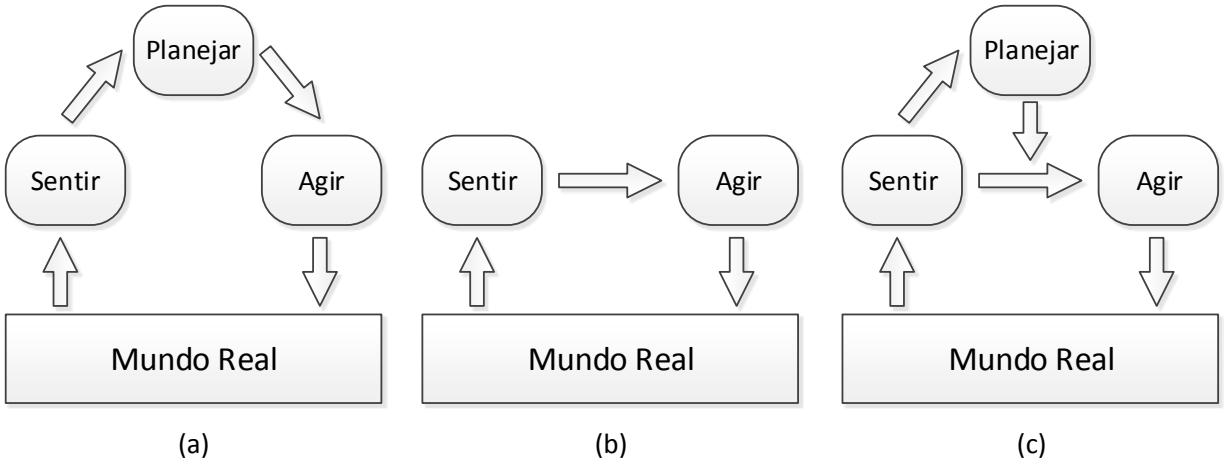


Figura 2.3 - Paradigmas das arquiteturas de controle na robótica móvel: (a) Hierárquico; (b) Comportamental; e (c) Híbrido.

Fonte: Adaptado de Murphy (2000).

No paradigma de controle hierárquico, ilustrado na Figura 2.3 (a), a informação segue progressivamente o fluxo "sentir-planejar-agir" de forma cíclica e deliberada sobre o ambiente, na qual os sensores alimentam um sistema de planejamento que, por sua vez, coordena os atuadores do robô (FAYEK, et. al., 1993). Com a presença da primitiva “planejar”, este paradigma dedica uma etapa para projetar suas ações futuras, e portanto, sua reatividade passa a depender do tempo que se leva para se fazer esse planejamento. Essa demanda de tempo pode se apresentar de forma negativa na reatividade do robô às alterações do ambiente. No paradigma de controle comportamental, ilustrado na Figura 2.3 (b), o fluxo da informação se dá em "sentir-agir", caracterizado pela ausência do módulo responsável pelo planejamento, uma vez que o comportamento se estabelece através da forma reativa. Logo, o mapeamento entre os módulos atuadores e os sensores é estabelecida diretamente, dando a este paradigma capacidade de reagir em tempo real às alterações ambientais. O paradigma comportamental é similar aos reflexos que não envolvem raciocínio (Mataric, 2007). Já no paradigma híbrido, ilustrado na Figura 2.3 (c), a informação segue o fluxo na ordem "planejar, sentir-agir", na qual as etapas de sensoriamento e atuação ocorrem de forma simultânea, enquanto o planejamento ocorre à parte, por meio de uma relação entre as primitivas "sentir-planejar". Logo, este paradigma é capaz de apresentar características de reatividade do paradigma comportamental ao mesmo tempo que é capaz projetar suas ações futuras.

De acordo com Mataric (2007), basicamente, três aspectos são relevantes na escolha de qual arquitetura de controle utilizar: escala de tempo, modularidade e representação. Em termos de escala de tempo, é relacionado o tempo de resposta do robô às alterações ambientais com a velocidade de reação do robô. A modularidade diz respeito ao modo como o sistema de controle é dividido em componentes funcionais e ainda, como tais componentes interagem entre si. Por último, a representação se refere à forma como as informações são armazenadas e codificadas dentro de um sistema robótico. Em discussões relacionadas ao desenvolvimento da arquitetura de controle, devem ser ponderadas questões como: diagrama de controle, informação centralizada ou descentralizada, reatividade, programabilidade, dentre outros (CHATILA; CAMARGO, 1990). No entanto, devido à extensão deste tópico, este estudo não aprofundará os aspectos relacionados ao desenvolvimento de arquiteturas de controle, uma vez que este encontra-se fora do escopo da proposta. Apesar da proposta deste trabalho não conter necessariamente o desenvolvimento ou a implementação de um método de controle específico, a importância de um estudo sobre arquiteturas de controle neste trabalho se dá na medida em que conhecer as diversas abordagens na literatura é fundamental para o desenvolvimento de uma plataforma capaz de suportar os diversos paradigmas de controle.

As arquiteturas de controle definem o fluxo de informação dentro de um sistema robótico através de um método formal de organização dos elementos de controle e estabelecem como os elementos se relacionam. No entanto, a arquitetura de controle não considera as técnicas que podem ser empregadas em sua implementação. Isso significa que a arquitetura de controle estabelece "o que" deve ser feito em relação ao controle, porém não determina "como" o controle será formado. Sendo assim, fica a cargo da arquitetura de *software*, determinar a composição e a organização da arquitetura de modo a suportar uma determinada arquitetura de controle.

De acordo com Santiago e Medeiros (2011), a arquitetura de *software* na robótica móvel é responsável pelo gerenciamento de algoritmos de tomada de decisão enquanto a arquitetura de *hardware* gerencia os processos de captura de sensores, suas configurações e a entrega das informações dos processos de decisão aos atuadores de movimento. Logo, a

arquitetura de *software* determina como os dados são distribuídos e processados, influenciando diretamente sobre a facilidade de desenvolvimento e reutilização de código. A arquitetura de *software* sempre atua de forma determinante no projeto e na complexidade do *hardware*, de modo que uma arquitetura de *software* bem estruturada pode resultar em uma ótima performance e simplicidade do *hardware* (NIRANJAN et. al.; 2012).

No contexto da robótica móvel, existem diferentes abordagens no que diz respeito à composição da arquitetura de *software* na literatura. Geralmente, podem ser classificadas como bibliotecas, sistemas operacionais, *middlewares* e baseadas em protocolos de redes de computadores (ALVES et. al., 2012).

Bibliotecas de *software* são constituídas por uma coleção de códigos fonte reutilizáveis e escritas em uma linguagem de programação. Num caso simples de biblioteca, esta é compilada junto ao aplicativo do usuário e integrada ao aplicativo. Isso acaba limitando o desenvolvimento de *software*, uma vez que exige uma codificação em uma única linguagem de programação. Por outro lado, bibliotecas podem se apresentar na forma de código externo, oferecendo assim vantagens como a modularidade e não-replicação de código. Entretanto, ainda se prendem a uma única linguagem de programação. Ainda que tal dependência possa ser superada com a utilização de técnicas de FFI (acrônimo de *Foreign Function Interface*, ou Interface de Funções Estrangeiras) ou ligação de linguagens. O suporte do SO (Sistema Operacional) é fundamental, já que cada SO gera binários (executáveis) diferentes. Existem bibliotecas, conhecidas com *cross-plataform*, que preocupam em gerar código que possa ser compilado em diversos SOs. O projeto CLARAty (NESNAS et. al., 2003) da NASA é um exemplo de arquitetura *cross-plataform* aplicada em robôs móveis.

Na abordagem de sistema operacional, diferentemente das bibliotecas, as funcionalidades do robô são vistas como módulos do sistema operacional e funcionam apenas internamente ao SO. Suas funcionalidades podem ser invocadas em diversas situações por diferentes aplicativos, e o acesso aos módulos depende da interface de programação suportada pelas linguagens. O ROS (*Robot Operating System*) (GUIGLEY et. al., 2009), é um exemplo de sistema operacional para robô de código aberto.

Já em arquiteturas baseadas em *middleware*, os diferentes componentes de *software* são vistos como aplicativos independentes, que podem ser executados na mesma máquina ou através da rede. Nela, um componente é responsável por intermediar a comunicação entre os módulos, o *middleware*. O MIRO (*Middleware for Robots*) (UTZ et. al., 2002) e o RT-*Middleware* (*Robot Technology Middleware*) (ANDO et. al., 2005) são exemplos de *middlewares* robóticos baseados no CORBA, um *middleware* de uso mais genérico e destinado a sistemas distribuídos (OMG, 1995). O CORBA ainda possui especificações como o *minimum* CORBA e o RT-CORBA (*Real Time* CORBA) v.1.1 (SCHMIDT et. al., 1997) capazes de prover mecanismos de comunicação em tempo real a componentes distribuídos e heterogêneos, característica muito interessante no contexto da robótica móvel. No entanto, o principal aspecto a considerar sobre o CORBA é seu caráter genérico, acarretando *software* extensos e complexos, que o prejudica na questão de facilidade de uso (NAMOSHE et. al., 2008). Ainda existem outros *middlewares*, como RSCA (*Robot Software Communication Architecture*) (YOO et. al., 2006), destinado à utilização em sistemas robóticos distribuídos.

Arquiteturas baseadas em redes fundamentam-se na popularização da Internet e seus protocolos para propor suas infraestruturas. Conforme Mohamed et al. (2008) relata em seu trabalho, em antigas gerações de robôs, estes eram projetados para serem capazes de executar tarefas específicas e, para tal, eram desenvolvidos como uma única unidade, enquanto nas gerações modernas, além dos robôs serem autônomos, são também ubíquos, ou seja, são capazes de se conectarem às redes e usufruirem dessas conexões. E pelo fato de serem constituídos por um conjunto integrado de sofisticados dispositivos de *hardware* e módulos de *software*, os robôs modernos são considerados sistemas distribuídos complexos, segundo Elkady e Sobh (2012) e Mohamed et al. (2008). Sistemas robóticos distribuídos utilizam subsistemas, que são módulos embarcados e independentes, que compartilham recursos e operam como um único sistema, visando ao mesmo objetivo (NIRANJAN et. al., 2012). Apesar de não ter sido desenvolvido especificamente para a robótica, padrões como a arquitetura UPnP (*Universal Plug-and-Play*) oferecem conectividade *peer-to-peer* (ponto-a-ponto) em ambientes computacionais distribuídos e dinâmicos. Na robótica, o UPnP pode ser utilizado na criação de módulos reusáveis e de fácil manutenção (MOK; WU, 2006), em vista

de sua capacidade de monitoramento e descoberta dinâmica de serviços disponíveis ao longo da rede.

Arquiteturas abertas objetivam prover uma interface comum para os componentes do robô, tais como módulos de sensoriamento e atuação, de modo que seja possível diferenciar e identificar características, como funções e especificações de cada componente. É possível relacionar isso ao *plug-and-play* ("conectar e usar") dos computadores (BEKEY, 2005).

2.7 Sistemas de comunicação na robótica

A comunicação de dados entre os diversos módulos que compõem um sistema robótico é de fundamental importância, uma vez que é por meio desta comunicação que dispositivos de diferentes funções irão se comunicar. Entretanto, a robótica usualmente requer a integração de diferentes módulos, fabricados por diferentes indústrias, portanto, possuem diferentes padrões de alimentação e adotam diferentes conectores físicos. Essa questão de incompatibilidade se estende à comunicação de dados nos quais os dipositivos adotam diferentes conexões de dados e empregam ainda, diferentes protocolos de comunicação. Em suma, além das incompatibilidades de alimentação, o problema da conectividade é tanto física quanto lógica. Essa heterogeneidade é justificável uma vez que muitos dos componentes adotados na robótica não são produtos ou tecnologias que foram desenvolvidos visando a sua aplicação na robótica. Uma forma de contornar essa situação é através de uma padronização, obtida por meio de uma arquitetura que emprega padrões de comunicação já estabelecidos e também por meio de módulos de *software* responsáveis pela abstração dessa heterogeinedade dos dispositivos.

Muitas vezes, sistemas robóticos necessitam executar tarefas de processamento e comunicação em tempo real, como por exemplo, mapeamento e nagevação. Logo, para garantir a reatividade e a capacidade de resposta às alterações em um ambiente em tempo real, o tráfego de informação entre os módulos de sensoriamento, planejamento e atuação deve ser mantido o menor possível, de forma a evitar atrasos na comunicação (FAYEK et. al.,

1993). Isso sugere que a troca de informação deva assumir elevado grau de abstração. A comunicação entre os processos pode ser feita por meio de protocolos de comunicação síncrona ou assíncrona. A maioria dos sistemas utilizam controladores embarcados que suportam a forma assíncrona e utilizam um barramento serial comum para a comunicação. A forma serial de comunicação de dados substituiu o tradicional uso da comunicação por meio do barramento paralelo, uma vez que esta forma necessita de mais recursos de *hardware* e também é mais suscetível a efeitos capacitivos.

Nesse contexto, um dos trabalhos analisados foi o URB (Saranli et. al., 2011), na qual é apresentada uma arquitetura modular de barramento de comunicação em tempo real destinada à robótica e outros domínios similarmente estruturados. A topologia física das conexões do URB é mostrada na Figura 2.4.

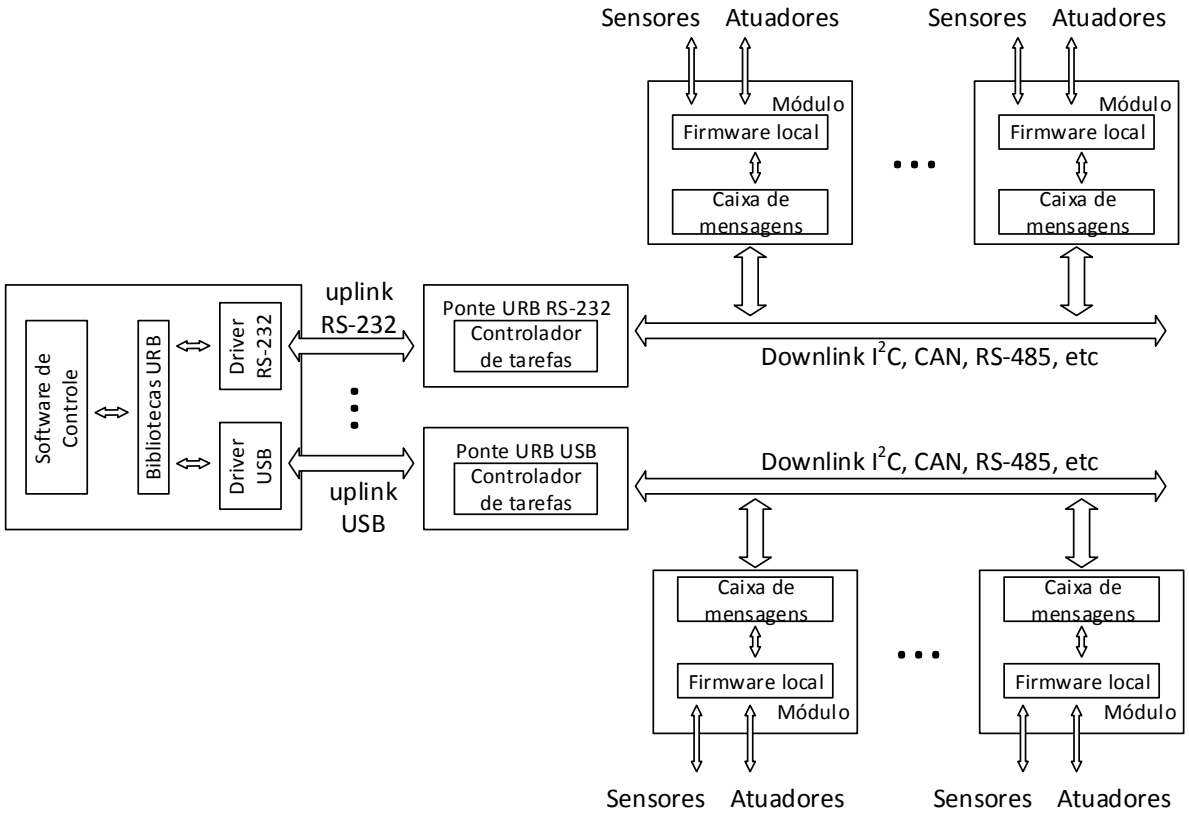


Figura 2.4 - Topologia física de conexões do URB.

Fonte: Adaptado de Saranli et. al. (2011).

Nesta arquitetura, a comunicação pode ser dividida em duas categorias: *uplink* e *downlink*. O *uplink* é responsável pela comunicação entre o *Software* de Controle e o Controlador de Tarefas, na qual pode ser estabelecida por meio de padrões como o RS-232 e o USB. Já o *downlink* remete ao barramento de conexão estabelecido entre o Controlador de Tarefas e os módulos de *hardware* (sensores e atuadores), conexão esta que pode ser estabelecida por meio de padrões como I^2C, CAN ou RS-485.

Um outro trabalho interessante é o R2P (*Rapid Robot Prototyping*) (BONARINI et. al., 2013) na qual é empregado o RTCAN. O RTCAN (*Real-Time* CAN) (MIGLIAVACCA et. al., 2013) foi desenvolvido a partir dos protocolos do barramento CAN e direcionado a aplicações robóticas. O RTCAN é capaz de explorar tanto o paradigma de transmissão *time-triggered* (disparado por tempo) como o *event-triggered* (disparado por evento), oferecendo, desta forma, suporte a três tipos distintos de mensagem: mensagens periódicas em tempo real, mensagens disparadas por eventos e mensagens sem prioridade. A arquitetura do R2P é mostrada na Figura 2.5.

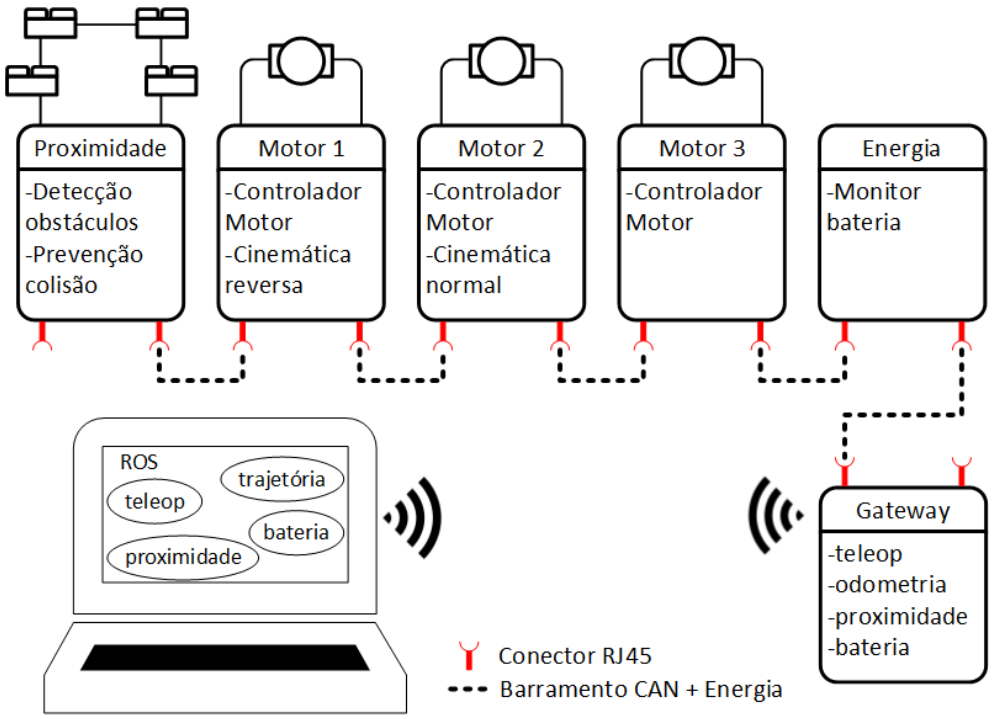


Figura 2.5 – Arquitetura do framework R2P.

Fonte: Adaptado de Bonarini et. al. (2013).

É possível observar que tanto o URB como R2P adotam uma abordagem na qual suas funcionalidades são implementadas modularmente tanto em termos de *hardware* quanto em termos de *software*. Desta forma, componentes frequentemente utilizados como controle dos motores, sensores de proximidade, etc, podem ser implementados em módulos padronizados, com seus respectivos *firmwares* locais. Esses módulos são conectados a um barramento comum de periféricos, no qual existe um módulo responsável pelo seu controle e interface com o *software* de controle.

2.8 Conclusões do capítulo

Este capítulo apresentou uma síntese a respeito da definição de robôs móveis autônomos, a composição característica dos mesmos, abordando tópicos como locomoção e técnicas de localização comumente utilizadas na robótica móvel, e ainda levantou aspectos relevantes acerca das arquiteturas de controle e *software*. Além disso, por meio de um breve estudo, observou como a comunicação é um aspecto fundamental na arquitetura robótica. Observando os tópicos abordados, é possível notar a extensão e a contribuição de cada um deles para o campo da robótica móvel.

A questão da reutilização dos módulos de *software* e mecanismos de controle de outras plataformas na robótica pode facilitar o processo de desenvolvimento, no entanto, normalmente acaba inviabilizado pela especificidade, uma vez que os módulos de *software* normalmente são destinados a um determinado *hardware* (KUBITZ et. al., 1998). Além disso, existe o problema da abstração dos recursos de *hardware*, que se torna mais difícil com o aumento da complexidade tanto do sistema de controle como da interface de *hardware*.

De acordo Coste-Manière e Simmons (2000), “os sistemas robóticos frequentemente necessitam satisfazer requisitos em tempo real e essa necessidade costuma ser complexa demais para ser desenvolvida e operada utilizando técnicas convencionais de programação”. Além disso, diferentes aplicações possuem diferentes necessidades, que deverão ser

satisfeitas por meio de diferentes arquiteturas. Isso explica a razão pela qual existem diferentes abordagens para a composição de arquiteturas de *software* na literatura.

Considerando o objetivo deste trabalho, que é o desenvolvimento de uma plataforma de robô móvel de arquitetura aberta, foi dada maior atenção a questões relacionadas a tópicos como composição, locomoção, arquiteturas de controle e comunicação. Tais tópicos são abordados com maior detalhamento no capítulo seguinte, onde são apresentadas as técnicas e os métodos escolhidos para o desenvolvinmento deste trabalho.

3 PLATAFORMA DE ROBÔ MÓVEL

Considerando que o objetivo da plataforma proposta é sua utilização em aplicações de propósito geral, é apropriada a adoção de uma arquitetura que ofereça facilidade de entendimento e utilização aos usuários ao mesmo tempo que permita um grau de flexibilidade de *hardware*, para ser modificada de acordo com as necessidades das aplicações. Desta forma, buscou-se uma arquitetura visando sua simplicidade conceitual e técnica, de modo que permita fácil compreensão e implementação, características que motive a utilização da plataforma em diversas iniciativas. Logo, a disposição da arquitetura da plataforma do robô móvel foi dividido em três camadas, sendo estas:

- Camada Superior. Camada na qual é implementado o *software* de controle, responsável pelo processamento dos dados e leituras dos sensores, tomada de decisões e pelo controle de acionamento dos atuadores;

- Camada Intermediária. Camada responsável por estabelecer a comunicação entre a Camada Superior e a Camada Inferior, através de uma interface comum de *hardware*, um protocolo de comunicação padronizado e APIs e bibliotecas de interface que abstraem os detalhes específicos de todos os dispositivos periféricos localizados na Camada Inferior;

- Camada Inferior. Corresponde aos dispositivos periféricos de *hardware* implementados e funcionais, abrangendo tanto sensores como atuadores.

A disposição da arquitetura descrita e suas camadas é ilustrada na Figura 3.1.

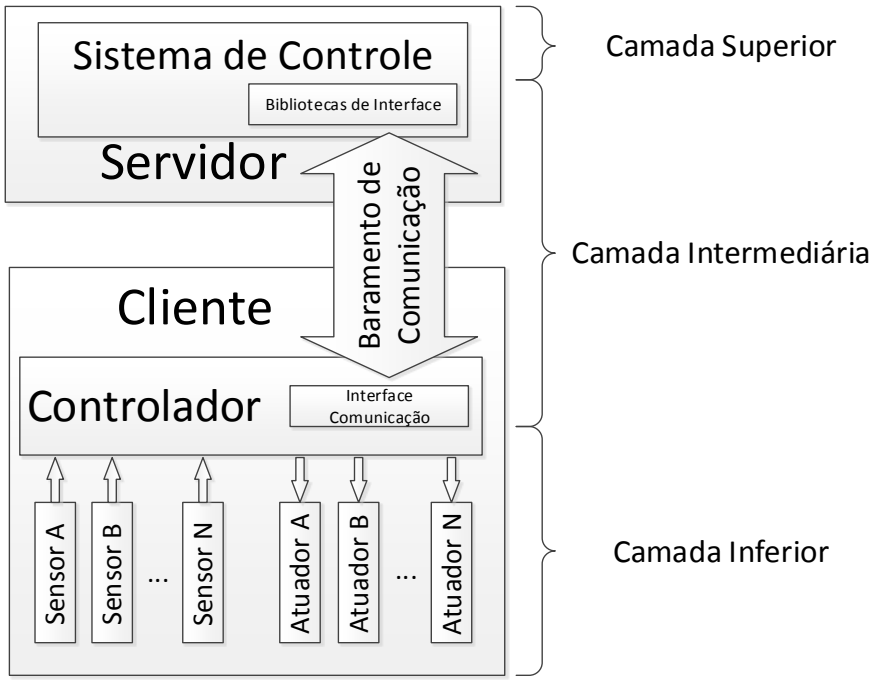


Figura 3.1 – Disposição da arquitetura e suas camadas.

Fonte: Elaborado pelo autor.

Esta plataforma propõe uma arquitetura de *software* do tipo cliente-servidor, onde o Cliente representa a base do robô móvel e o Servidor corresponde ao sistema disponível para o controle. O Servidor pode tanto ser embarcado, com a conexão sendo estabelecida diretamente no barramento de comunicação, ou remota, sendo a comunicação estabelecida com a utilização de tecnologias de conexão sem fio, como RF, Wi-Fi ou Bluetooth.

Nesta arquitetura, o Servidor é composto pelo Sistema de *Software* de suporte para a Arquitetura de Controle, APIs e bibliotecas responsáveis pela interface e abstração da comunicação estabelecida com o Cliente. A Camada Superior compreende tecnologias computacionais como computadores pessoais, computadores portáteis ou dispositivos embarcáveis de processamento, como microcontroladores, FPGA, Arduino e Raspberry Pi.

Já o Cliente corresponde à base do robô móvel, onde se encontra o Controlador e os módulos de *hardware* periféricos. Para o Controlador, foi adotado o conceito de sistema fortemente acoplado, atribuindo a um único agente a tarefa de gerenciar todos os dispositivos de sensoriamento e atuação. Isso confere à plataforma menor complexidade em

comparação à abordagem conceitual que um sistema fracamente acoplado acarreta. Um outro fator que levou a essa escolha é a necessidade de um número reduzido de dispositivos dedicados de gerenciamento dos dispositivos periféricos, e portanto, menor custo de implementação e manutenção, além da mitigação de erros e falhas relacionadas ao desenvolvimento, implementação e alterações dos módulos localizados na Camada Inferior. A adoção de uma abordagem fracamente acoplada, apesar de apresentar maior flexibilidade, por sua vez, necessita de um barramento dedicado à conexão dos dispositivos periféricos ao módulo de gerenciamento, além de um protocolo de comunicação comum para estas conexões, como mostrado em 2.8.

Com esta disposição de arquitetura mostrada na Figura 3.1, é possível implementar e experimentar diferentes paradigmas de controle, uma vez que a arquitetura adotada é capaz de oferecer recursos essenciais para suportar qualquer um dos três paradigmas: hierárquico, reativo ou híbrido deliberativo/reativo (Murphy, 2002).

Cada uma das três camadas são melhor descritas nas seções seguintes.

3.1 Camada Superior

A Camada Superior refere-se aos módulos responsáveis pelas tarefas como processamento do dados obtidos pelos sensores, tomada de decisões e acionamento dos dispositivos atuadores. Esta camada compreende tecnologias computacionais utilizadas para a realização dessas tarefas, que podem ir desde dispositivos embarcáveis de processamento a complexos sistemas computacionais. De um modo geral, o processo de tomada de decisões é realizado por meio de computadores. Isso se deve à popularização dos computadores, alavancada pelo contínuo avanço tecnológico e redução dos custos de produção. Além disso, é possível mencionar outros fatores implícitos, como miniaturização, aumento da capacidade de processamento, comunicação e memória, assim como a redução do consumo e otimização dos recursos energéticos. Com essa popularização dos computadores, surgem novas linguagens de programação, provendo diferentes formas de abstrações de estruturas de

dados, assim como paradigmas de programação. A robótica faz uso da programação em computadores, escritos em linguagens de programação, para desenvolver suas arquiteturas de controle. Logo, a escolha de uma arquitetura de *software* impacta sobre a implementação, manutenção e reaproveitamento de código, além de determinar o modelo de processamento e a plataforma tecnológica computacional.

Ao deixar em aberto as questões como arquitetura de controle e arquitetura de *software*, esta plataforma oferece neutralidade em relação à implementação de qualquer paradigma de controle, implicando também em uma neutralidade em relação à linguagem de programação utilizada. Essa abordagem permite ao usuário, maior liberdade na escolha da linguagem de programação que melhor atenda às necessidades da aplicação ou simplesmente em razão da familiaridade com determinada linguagem ou ambiente de desenvolvimento. E essa liberdade, por sua vez, se estende também à tecnologia de controle empregada.

O desenvolvimento e implementação da arquitetura de controle junto a uma arquitetura de *software* para este trabalho foi conduzido unicamente com o intuito de averiguar se a plataforma proposta é capaz de suportar cenários de aplicações empregando diferentes abordagens em aplicações distintas. Duas situações de aplicação foram elaboradas: a primeira relacionada a uma aplicação de um experimento envolvendo navegação e outra para uma aplicação científica hipotética, situações que são discutidas com mais detalhes no capítulo 4.

3.2 Camada Intermediária

O sistema de comunicação de dados é uma parte essencial para robôs móveis, uma vez que é responsável pela interface de comunicação entre seus módulos. Esta seção apresenta o sistema de comunicação de dados adotado para efetuar a comunicação entre a Plataforma do Robô Móvel e o Sistema de Controle.

A comunicação de dados adotada foi a serial, compatível e com suporte ao padrão RS-232, embora existam outros padrões de comunicação serial, como o padrão USB. De acordo com Saranli et. al. (2011), o RS-232 é uma interface cuja conectividade pode ser direcionada em aplicações que exijam pouca ou média largura de banda. Além disso, este padrão pode ser encontrado em dispositivos computacionais embarcáveis, como FPGAs. Já o USART (acrônimo do termo inglês: "*Universal Synchronous Asynchronous Receiver Transmitter*" de Transmissor Receptor Universal Síncrono Assíncrono) é um padrão de comunicação de dados serial ponto a ponto, sendo comumente empregado em conjunto com o padrão RS-232, RS-485, dentre outros. O USART pode operar em modo *full-duplex* ou *half-duplex*. O *full duplex* é uma comunicação assíncrona, na qual dois canais físicos são utilizados para o envio de dados, um canal para cada sentido, podendo deste modo, ocorrer transmissão e recepção simultânea de dados. Neste modo, a taxa de comunicação de ambos os dispositivos conectados devem ser comuns. Já o modo half-duplex caracteriza uma comunicação síncrona estabelecida entre dispositivos na configuração mestre-escravo, na qual um canal físico é empregado para a transmissão bidirecional dos dados, e um outro canal é dedicado à sincronização estabelecida pelo dispositivo mestre. Esta comunicação limita a transmissão e recepção para um único sentido de cada vez e suporta taxas de comunicação inferiores ao modo assíncrono.

De forma a oferecer maior flexibilidade em relação à sua conectividade, uma linha de comunicação direta dá ao usuário do Kihon, acesso ao terminais EUSART (acrônimo do termo inglês "*Enhanced Universal Synchronous Asynchronous Receiver Transmitter"*, ou "Transmissor Receptor Universal Síncrono Assíncrono Aprimorado") do microcontrolador utilizado, dando maior liberdade de acoplar diferentes esquemas físicos de plataformas de controle ou dispositivos de comunicação sem fio (rádio, Wi-Fi ou *bluetooth*).

Como mencionado anteriormente, o padrão do tipo serial apresenta aspectos favoráveis em relação ao tipo paralelo e até mesmo, em relação a padrões mais modernos, como o próprio USB. O principal aspecto considerado para tal escolha foi sua simplicidade, uma vez que o porto serial apresenta-se como uma alternativa de fácil entendimento, implementação e uso. Além disso, padrões mais modernos como o USB, por possuir diversos

modos de operação, conectividade e recursos como plug-and-play, apresentam extensas documentações e acabam exigindo conhecimentos mais avançados. Por outro lado, a simplicidade oferecida pelo USART em conjunto com o padrão RS-232 é interessante e conveniente, permitindo fácil compreensão, implementação e utilização aos usuários. Com isso, é possível concentrar a maior parte dos esforços nas questões relacionadas à robótica ou ao tópico de interesse, e sem desestimular o usuário.

A proposta deste trabalho se limita à concepção do firmware do Controlador, desta interface de comunicação e protocolo e, por fim, de uma API, cuja função é abstrair e especificar como os componentes do *software* de controle devem interagir com os dispositivos localizados na Camada Inferior.

3.3 Camada Inferior

A Camada Inferior refere-se aos módulos periféricos de *hardware*. O conjunto desses módulos foi escolhido para compor o Kihon, seguindo critérios como baixo custo e facilidade de aquisição no mercado atual. A escolha dos módulos tanto de sensoriamento como atuação foi feita objetivando a concepção do Kihon, de modo que este apresente funcionalidades essenciais para suportar iniciativas diversas envolvendo a robótica móvel e tópicos relacionados.

Os atuadores escolhidos foram:

- Dois motores CC com caixa de redução, para a tração diferencial;
- Um servomotor para alterar a direção do sonar acoplado em seu eixo.

Os sensores escolhidos foram:

- Dois sensores de odometria (*encoders*), acoplados aos eixos dos motores CC – antes da redução;

- Um sonar acoplado ao servomotor;
- Três sensores de proximidades posicionados na parte frontal do Kihon;
- Dois sensores de solo.

A Figura 3.2 mostra o diagrama em blocos dos dispositivos de *hardware* implementados no robô Kihon.

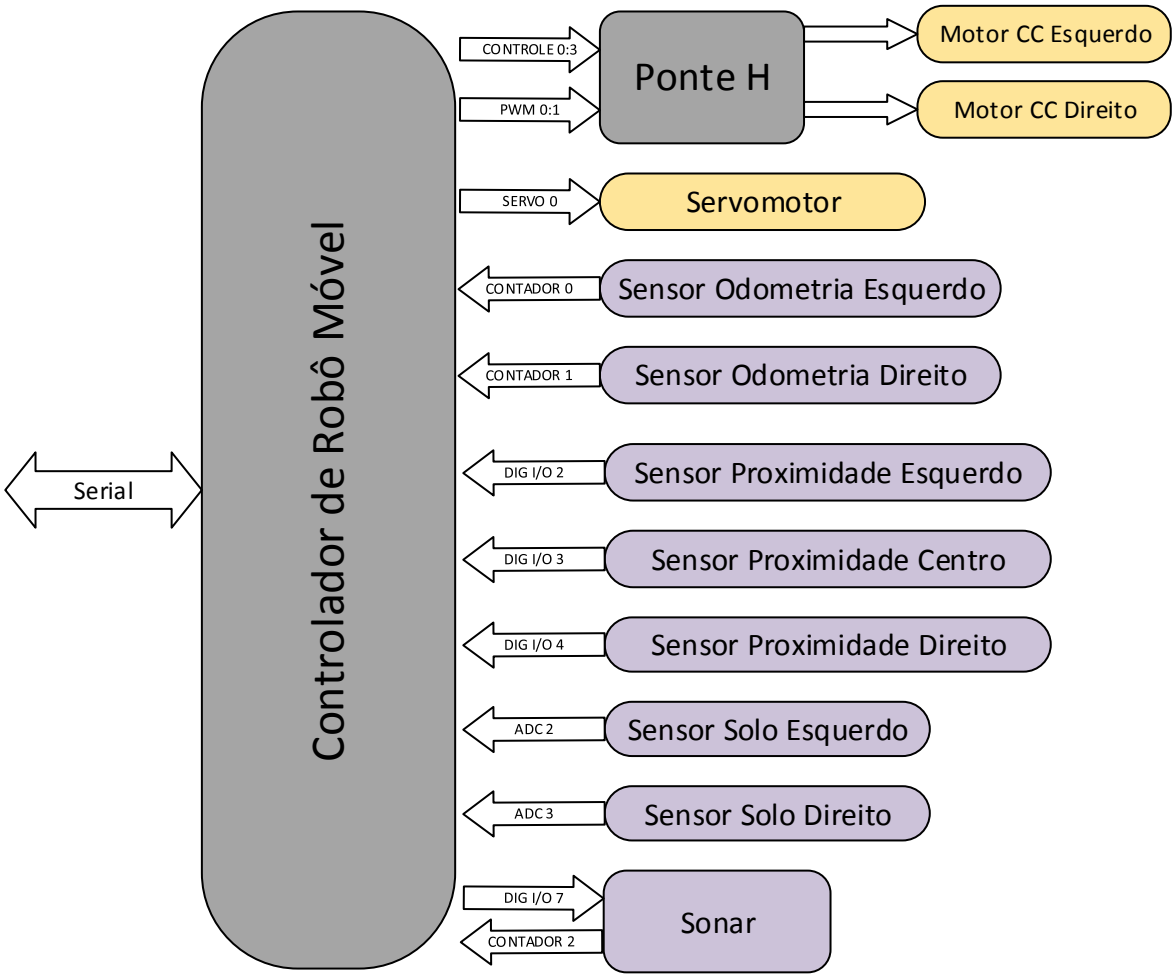


Figura 3.2 – Diagrama em blocos dos dispositivos implementados no Kihon.

Fonte: Elaborado pelo autor.

Por questões de praticidade, facilidade de implementação e custo, um único microcontrolador foi adotado para coordenar e gerenciar todos os recursos de *hardware* localizados na Camada Inferior. Portanto, para esta proposta, optou-se por adotar um sistema

centralizado e fortemente acoplado, ainda que, num segundo momento, seja possível implementar um sistema fracamente acoplado, baseado em uma rede de microcontroladores com funções específicas.

3.4 O robô móvel Kihon

Nesta seção é apresentado o robô móvel Kihon, concebido com o propósito de servir como uma plataforma básica de robô móvel. O nome é formado a partir dos kanjis (caracteres chineses utilizados na língua japonesa) 基 (ki), que pode significar *fundamental* ou *fundação*, e 本 (hon), que pode significar *principal* ou *origem*. Assim, Kihon (基本) é um termo do idioma japonês que no contexto educacional pode ser traduzido como *princípios básicos* ou *fundamentos*. Portanto, no contexto deste trabalho, o nome Kihon se deu em virtude dos recursos essenciais oferecidos pela plataforma proposta, de modo a permitir que esta possa servir como uma ferramenta de aprendizado de conceitos, os quais poderão ser explorados e praticados por meio do robô móvel Kihon.

O Kihon corresponde à base do robô móvel e se enquadra no bloco de Estrutura Mecânica apresentada da Figura 1.1 e à Camada Inferior da arquitetura apresentada na Figura 3.1. Sua construção foi motivada para averiguar se a plataforma aberta do robô móvel apresentada neste capítulo pode ser concebida de modo a oferecer os requisitos almejados. Para sua concepção, foram considerados o propósito de utilização da plataforma, características como flexibilidade e, por fim, a neutralidade em relação às tecnologias de controle.

3.4.1 Aspectos mecânicos e cinemáticos

O tamanho do corpo físico do robô determina o que poderá ser embarcado ou acoplado sobre o mesmo. Considerando o propósito de uso do robô móvel, uma situação prevista é sua

utilização em aplicações empregando plataformas de controle embarcados, como por exemplo, *netbook*, Raspberry Pi, Arduino ou FPGA. Para dar suporte a tais situações, iniciou-se um estudo para o dimensionamento do robô, ou seja, seu porte físico. Para permitir o acoplamento de tecnologias de sistemas de controle variados, foi estimado seu porte físico, de modo a possibilitar o acoplamento do sistema de controle que, dentre os previstos, requer maior espaço físico. E dentre as possibilidades previstas para o sistema de controle, o maior, em termos de dimensões, foi determinado pelo *netbook* adotado, o Eee PC 900 da Asus, disponível no LISDI, e cujas dimensões aproximadas são de 3×27×20 cm (A×L×P).

Para a locomoção do Kihon, dentre outras alternativas possíveis, adotou-se a locomoção baseada em rodas. Mecanismos de locomoção baseados em rodas são os mais comumente encontrados na robótica móvel por uma série de fatores, dentre as quais é possível citar baixo custo, seja na implementação como manutenção, estabilidade e, finalmente, por sua simplicidade mecânica e de controle. As rodas apresentam elevado grau de eficiência em aplicações terrestres, especificamente em superfícies planas. E para este trabalho, tal escolha ainda se deu por se adequar ao ambiente de aplicação no qual o Kihon será destinado e, ainda, pela facilidade de implementação e uso.

Além disso, decidiu-se pela adoção do modelo de tração diferencial por ser comumente empregado na robótica móvel, uma vez que seu modelo cinemático é menos complexo em comparação às outras disposições possíveis. É possível citar algumas características dessa disposição, como simplicidade, relativo baixo custo e facilidade de construção. E, ainda, sua compreensão é mais intuitiva, o que facilita o desenvolvimento de seu controle de locomoção.

O cálculo das velocidades linear v e angular ω do centro geométrico (C) de robôs com sistema de tração diferencial e aplicável ao Kihon é mostrado na equação (3.1). O detalhamento a equação é apresentado no Apêndice A.

$$\begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_e \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{R}{r} \\ \frac{1}{r} & -\frac{R}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad , \tag{3.1}$$

onde r de ambas as rodas ativas empregadas no Kihon corresponde a 4,445cm e a distância R do Kihon corresponde a 11,5cm.

Sabendo-se que o raio da roda empregada é de 4,445cm, o motor CC com caixa de redução empregado apresenta nominalmente 80rpm, e que a roda é acoplada por meio de uma ponta de eixo à saída da caixa de redução do motor CC, é possível determinar a velocidade linear máxima $v_{\max}$ do robô móvel como sendo 37,144cm/s (cálculos no Apêndice A).

Por último, o peso é um item que deve ser ponderado, uma vez que componentes de *hardware* pesados acarretam em um maior esforço que deverá ser destinado ao sistema de locomoção. Assim, um motor com maior torque e mais pesado deverá ser empregado. Isso afeta também o sistema de alimentação de energia do sistema, que necessitará de baterias de maior capacidade de fornecimento de carga, mas que são maiores e mais pesadas. A questão do peso envolve a escolha adequada dos componentes, devendo, além de minimizar os custos econômicos, satisfazer os requisitos funcionais do robô. Outros detalhes do modelo cinemático do robô móvel Kihon pode ser encontrado no Apêndice A.

3.4.2 Desenvolvimento do hardware

O robô móvel Kihon foi construído basicamente a partir de materiais previamente disponíveis no LISDI. Os materiais utilizados incluem baterias, placa de circuito impresso padrão, componentes eletrônicos, motores CC, servomotor, barras de alumínio, microcontrolador, parafusos, dentre outros itens. O custo total de material utilizado para a construção do Kihon foi de aproximadamente R$500,00. No entanto, foram desconsiderados os custos do seu desenvolvimento e da sua montagem. Este valor serve para elucidar o baixo custo relativo do robô, pois o valor real depende de variáveis de mercado que fogem do escopo deste trabalho.

A princípio, para fins de validação, o Kihon embarca uma quantidade mínima de dispositivos que o permita apoiar iniciativas gerais relacionadas à robótica móvel. E ainda, o Kihon apresenta uma API própria que torna sua programação facilitada. A API escrita para o Kihon contém rotinas essenciais para efetuar a leitura de suas entradas e também, controlar dispositivos atuadores. A Figura 3.3 mostra o robô móvel Kihon.

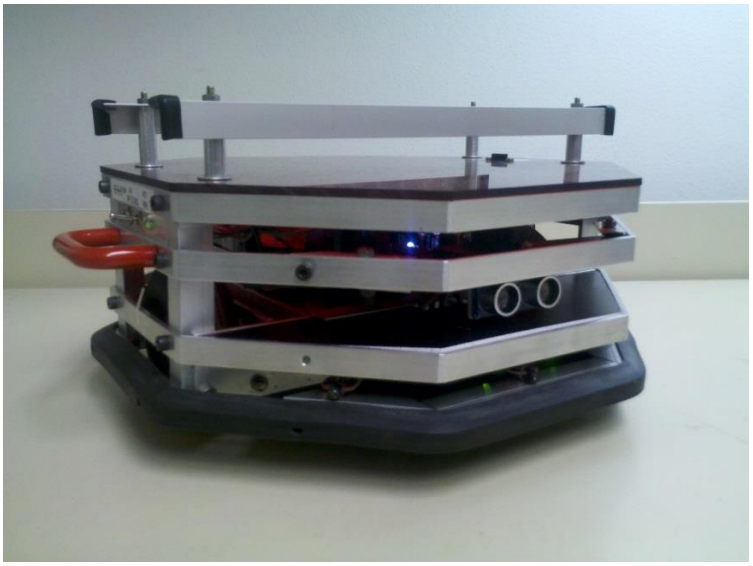


Figura 3.3 - Robô Móvel Kihon.

Fonte: Elaborado pelo autor.

De modo a alcançar os objetivos almejados em 1.3 e 1.4, foi implementado um conjunto inicial padrão de módulos sensores e atuadores no Kihon. Esses módulos têm a finalidade de oferecer funcionalidades essenciais ao Kihon, como possibilitar sua locomoção com a adoção de um sistema de locomoção, e suporte a técnicas relativas de localização e navegação. A seguir, serão discutidos de forma objetiva os módulos escolhidos para compor o conjunto inicial padrão de módulos, justificando brevemente as razões que levaram à escolha de implementá-los no Kihon.

Os mecanismos atuadores escolhidos foram:

- Dois Motores CC com caixa de redução para o sistema de tracionamento: como atuadores na robótica móvel, motores CC podem facilmente ser empregados como atuadores das rodas. Uma vez delimitado o ambiente terrestre de aplicação para

superfícies planas e com a adoção do sistema de tracionamento diferencial, dois motores são suficientes para habilitar a locomoção do robô móvel. Motores CC com caixa de redução são baratos, abundantes no mercado e possuem diversos tamanhos e relações de redução. Além disso, são fáceis de implementar e usar. Para compor o sistema de tracionamento do Kihon, decidiu-se empregar dois motores CC com caixa de redução;

- Um Servomotor para o direcionamento do Sonar: um servomotor foi empregado para direcionar um Sonar de modo a varrer a região frontal do Kihon. O eixo do servo usualmente é limitado a uma amplitude de rotação de cerca de 180 graus. Além disso, seu eixo pode assumir posições específicas dentro desse intervalo de amplitude de rotação. Desta forma, com a fixação de um Sonar sobre o eixo do servomotor, é possível detectar a distância de eventuais objetos e obstáculos da região frontal do Kihon.

Os mecanismos de sensoriamento escolhidos foram:

- Dois *encoders* (codificadores ópticos) para a odometria: pela simplicidade conceitual e facilidade de implementação e manuseio, optou-se por implementar a técnica da odometria como método de estimativa de localização relativa do Kihon. A odometria é uma técnica de localização relativa comumente empregada na robótica móvel. Nos robôs móveis baseados em rodas, usualmente, a implementação da odometria acopla *encoders* diretamente aos eixos de rotação dos motores de tracionamento ou no eixo das rodas do robô. Através da combinação de informações de rotação das rodas com o modelo cinemático do robô, é possível estimar a movimentação do robô móvel;

- Três módulos ópticos digitais de sensores de proximidade: para a detecção de objetos e obstáculos próximos ao Kihon, decidiu-se acoplar três módulos sensores de proximidade na parte frontal do Kihon. Esses módulos utilizados, através do princípio de reflexão da luz, são capazes de detectar a presença de objetos apresentando determinado nível de sinal lógico digital em suas saídas e nível lógico contrário na ausência de objetos detectados;

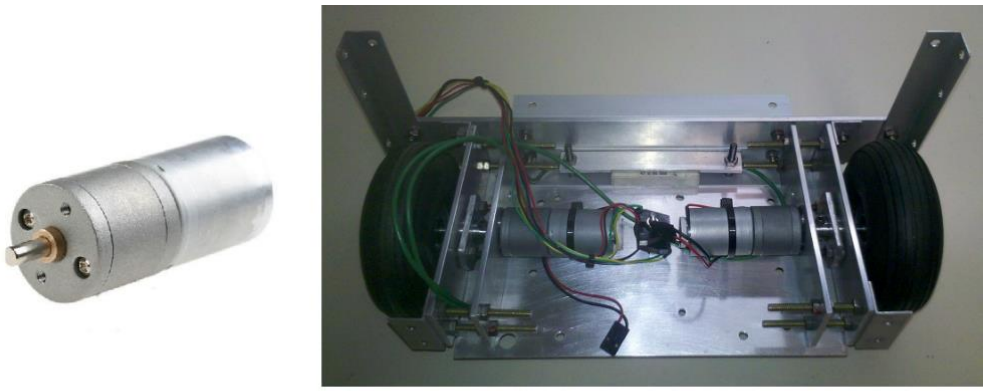
- Sensores para seguir linhas com o emprego de dois sensores ópticos reflexivos: na robótica móvel, é comum haver experimentos de navegação envolvendo a tarefa de seguir linhas ou efetuar leituras do solo como parte da aplicação. Além disso, é comum o emprego de dispositivos de sensoriamento que fornecem níveis de sinais analógicos em suas saídas. De modo a oferecer uma funcionalidade ao Kihon aliada ao fato de oferecer uma forma de sensoriamento analógico simples e útil à plataforma, decidiu-se pela implementação de dois sensores óticos reflexivos em sua base;

- Sonar: sonares são comumente empregados na robótica móvel para auxiliar no processo de navegação dos robôs móveis e detecção de obstáculos. Como uma implementação de um dispositivo sensor de distância, decidiu-se empregar um Sonar em conjunto com o servomotor.

A descrição detalhada de cada um dos mecanismos citados acima e implementados no robô móvel Kihon é dada a seguir.

3.4.2.1 Motores CC

Dois motores CC de 12V com caixa de redução são utilizados para compor o sistema de locomoção do Kihon. Com a caixa de redução, o motor empregado é capaz de oferecer em sua saída 80rpm a 3kgfcm. A Figura 3.4 (a) mostra o motor utilizado e a Figura 3.4 (b) os dois motores CC engrenados instalados na estrutura mecânica do Kihon.



(a) (b)

Figura 3.4 – (a) Motor CC engrenado utilizado; e (b) Motor acoplado à estrutura mecânica do Kihon.

Fonte: Elaborado pelo autor.

O acionamento de ambos os motores CC é feito através do L298, um circuito integrado que desempenha a função de duas pontes H completa. Os canais de Controle 0-3 do Controlador são conectados às entradas do L298, enquanto que as saídas PWM0 e PWM1 são conectadas aos pinos habilitadores (*enable*) do L298. Por meio desta conexão, é possível alterar tanto o sentido de rotação de cada motor CC como as tensões médias aplicadas aos mesmos.

A Figura 3.5 ilustra as combinações possíveis de controle e acionamento do motor. O canal PWM 0 efetua o controle de acionamento (*enable*) do motor enquanto os canais de Controle 0 e 1, conectados às entradas Input 0 e Input 1 do L298 definem o sentido de rotação do motor.

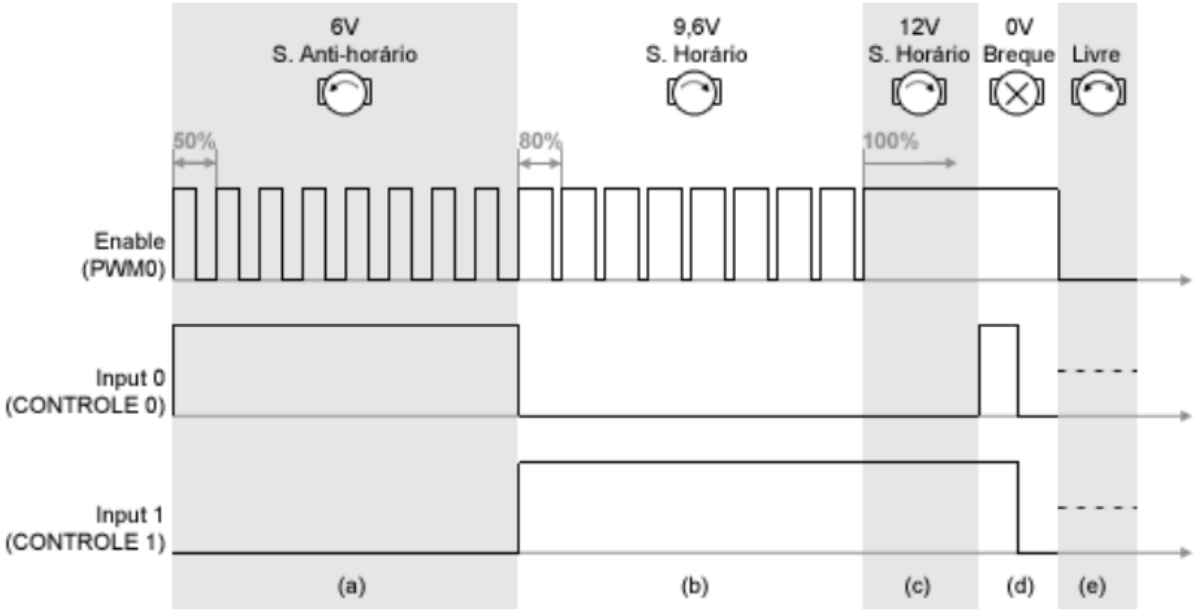


Figura 3.5 – Formas de ondas dos canais de controle PWM e seus efeitos no motor CC.

Fonte: Elaborado pelo autor.

A Figura 3.5 (a) ilustra o comportamento do motor quando o Enable está a 50%, Input 0 setado (nível alto) e Input 1 resetado (nível baixo). Nessas condições, o motor opera em sentido anti-horário com uma tensão média de 6V, equivalente a 50% do ciclo de trabalho PWM da tensão de alimentação de 12V. Já a Figura 3,5 (b) mostra o comportamento do motor quando o Enable está a 80%, Input 0 resetado (nível baixo) e Input 1 setado (nível alto). Essas condições resultam em uma tensão média de 9,6V sobre o motor, que agora opera em sentido horário. A Figura 3.5 (c) ilustra uma continuidade do estado mostrado em (b), no entanto, opera a 100% do ciclo de trabalho, alimentando o motor com tensão média de 12V, permanecendo a rotação no sentido horário.

A Figura 3.5 (d) ilustra a configuração necessária para que o motor opere em estado de breque. Para que isso ocorra, o ciclo de trabalho do PWM deve ser 100% e o Controle 0 deve ser igual ao Controle 1. Um motor elétrico opera em estado de breque quando seus terminais estão em curto-circuito. Tal comportamento ocorre devido à capacidade do motor em produzir corrente quando seu eixo é rotacionado por forças externas (força eletromotriz - fem). No entanto, quando um motor cujos terminais estão curto-circuitados tem seu eixo

rotacionado, uma corrente é gerada pelas bobinas do motor, realimentando-a no sentido contrário ao sentido rotacionado, forçando o motor a girar no sentido contrário. Isso faz com que o motor ofereça uma resistência às variações do estado de seu eixo, na chamada força contra eletromotriz (fcem). Assim, o motor parado tende a permanecer parado quando seus terminais estão em curto-circuito, bloqueando a rotação das rodas e atuando com freio.

Finalmente, a Figura 3.5 (e) mostra o comportamento do motor quando o PWM está desabilitado, ou seja, seu ciclo de trabalho é 0%. Nestas condições, a ponte-H é desabilitada, logo, independentemente dos estados dos Controle 0 e Controle 1, não há nenhuma conexão dos terminais do motor, permanecendo este desligado e, seu eixo, livre.

3.4.2.2 Servomotor

Um servomotor foi utilizado para controlar a direção do sonar. Por meio desse direcionamento, possibilita-se ao Kihon uma cobertura frontal e lado a lado de aproximadamente 180 graus da sua parte dianteira.

Servomotor é um motor de corrente contínua que possui internamente um circuito responsável pelo controle de seu eixo. Os servomotores operam por meio de pulsos periódicos de 5V, usualmente, no intervalo de 500µs a 2500µs e repetidos a cada 20ms. Esses pulsos correspondem linearmente à posição do eixo do servo motor, que costumam ser limitados no intervalo de 0 a 180 graus. A Figura 3.6 ilustra a relação entre forma de onda e o posicionamento do eixo de um servomotor.

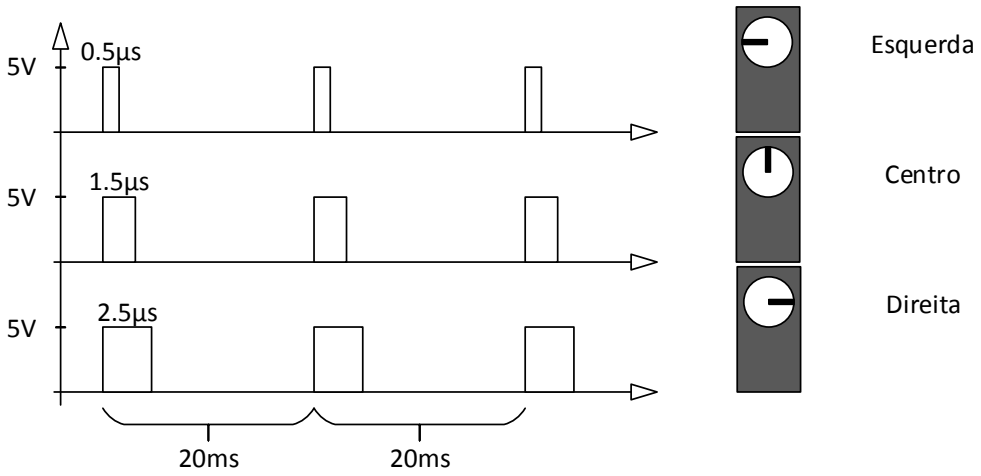


Figura 3.6 – Relação entre forma de onda e posicionamento do eixo de um servo motor.

Fonte: Elaborado pelo autor.

No entanto, existe uma enorme variedade de servomotores no mercado, com diferentes especificações, que incluem diferentes intervalos dos pulsos. O envio de pulsos fora do intervalo suportado pelo servomotor pode danificar as engrenagens do componente, comprometendo-o permanentemente. O servomotor HS-81, da Hitec, utilizado no Kihon é capaz de operar entre 0,9 a 2,1ms, no entanto opera no intervalo de 1,1ms a 1,9ms. Essa limitação foi estipulada pelo *software* de controle, uma vez que o Controlador permite o intervalo usual. No robô Kihon, o servomotor utilizado foi conectado ao canal Servo 0 do Controlador. O servomotor utilizado no Kihon é mostrado na Figura 3.7 (a), e a Figura 3.7 (b) mostra o servomotor já acoplado à estrutura mecânica do Kihon.

Figura 3.7 – (a) Servomotor utilizado no Kihon; e (b) Servomotor acoplado ao Kihon.

Fonte: Elaborado pelo autor.

3.4.2.3 Sensores de odometria

Como uma técnica comum de localização relativa, a odometria é responsável por estimar a distância percorrida pelas rodas, a partir de um ponto inicial. No Kihon foram implementados dois sensores de odometria, um para cada motor CC do sistema de locomoção. O sistema de odometria consiste no conjunto motor CC, *encoder* e emissor-receptor de luz infravermelha. Para o *encoder*, foi elaborado um pequeno disco circular, dividido radialmente em duas partes iguais de modo que uma metade de sua superfície seja preta e outra metade, branca. Cada disco foi acoplado a um motor CC engrenado, em seu eixo antes da caixa de redução. Para determinar o quanto esse eixo rotacionou, um componente emissor-receptor de luz infravermelha foi posicionado perpendicularmente à superfície do disco do *encoder*, de modo a detectar cada transição da luz refletida de preto-branco e branco-preto. Conhecendo a relação da caixa de redução do motor CC, para cada giro do eixo do respectivo motor, o sensor infravermelha é capaz de detectar essas transições, tornando possível determinar matematicamente o quanto a respectiva roda girou. A Figura 3.8 ilustra o conjunto *encoder* e sensor de odometria.

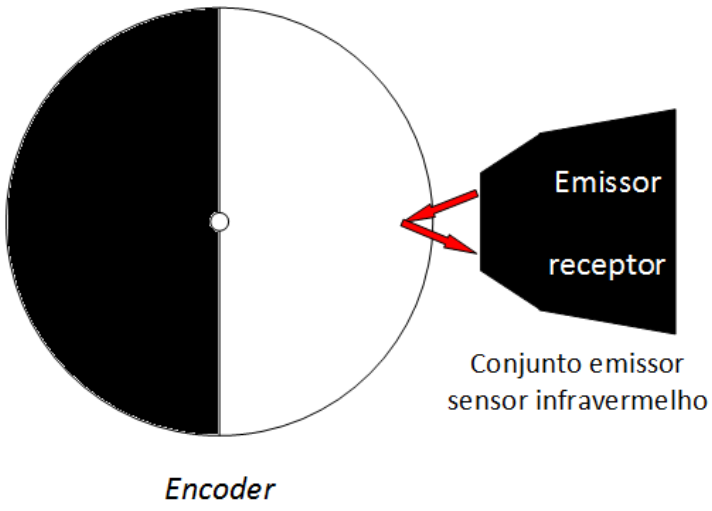


Figura 3.8 – *Encoder* e Sensor de odometria de um pulso por revolução.

Fonte: Elaborado pelo autor.

Quando o emissor-receptor infravermelho incide perpendicularmente sobre a superfície preta, maior parte da luz emitida é absorvida pela superfície escura e, com isso, praticamente não há reflexão do feixe de luz para o receptor. O receptor nada mais é que um foto-transistor de luz infravermelha. Com pouca luz refletida, não há excitação do foto-transistor, cortando-o. Por outro lado, quando o emissor-receptor infravermelho incide sobre a superfície branca, maior parte da luz emitida é refletida pela superfície clara. Nessa situação, o receptor recebe a maior parte do feixe de luz refletida na superfície, excitando o foto-transistor. O foto-transistor, quando em estado de corte, apresenta em sua saída, tensão próxima à alimentação – neste caso, 5V. Quando excitado, o foto-transistor apresenta tensão próxima de 0V.

O componente utilizado para efetuar a leitura do *encoder* foi o OPB705, composto por um LED e um foto-transistor, ambos operando no espectro de luz infravermelha. Pelo fato desses componentes utilizarem esta faixa específica de frequência luminosa, eles são imunes à maior parte dos ruídos luminosos do ambiente. No entanto, o foto-transistor empregado, responsável pela recepção do feixe, não apresenta níveis de sinais bem definidos em sua saída, apresentando faixas de tensões próximas a 1V quando excitado e 4V em corte. De modo a garantir níveis lógicos bem definidos em sua saída, empregou-se o ULN2004, um circuito integrado com 7 canais de *drivers* NPN Darlington. A Figura 3.9 (a) ilustra o componente OPB705, e a Figura 3.9 (b), mostra os OPB705 acoplados ao o sistema de odometria do Kihon.

(a) (b)

Figura 3.9 – (a) Sensor de odometria utilizado; e (b) Sensores acoplados ao sistema de odometria.

Fonte: Elaborado pelo autor.

A partir dessa disposição de motor, disco de odometria e emissor-receptor, quando o eixo do motor é rotacionado, ocorre uma série de alternâncias entre absorção e reflexão na superfície do disco de odometria. Essa alternância gera na saída do circuito ondas quadradas cujos períodos são inversamente proporcionais à velocidade de rotação do eixo do motor CC. Em outras palavras, quanto mais rápida a velocidade de rotação do eixo do motor, menores são os períodos das ondas quadradas e, portanto, estas apresentam maiores frequências. A frequência obtida pode ser correspondida a uma estimativa de velocidade de deslocamento do robô enquanto que a quantidade de ciclos de ondas quadradas podem ser relacionadas à uma estimativa da distância percorrida pelo robô. A Figura 3.10 mostra a vista superior do sistema de odometria do robô Kihon e sua relação com o sistema de locomoção do robô móvel Kihon.

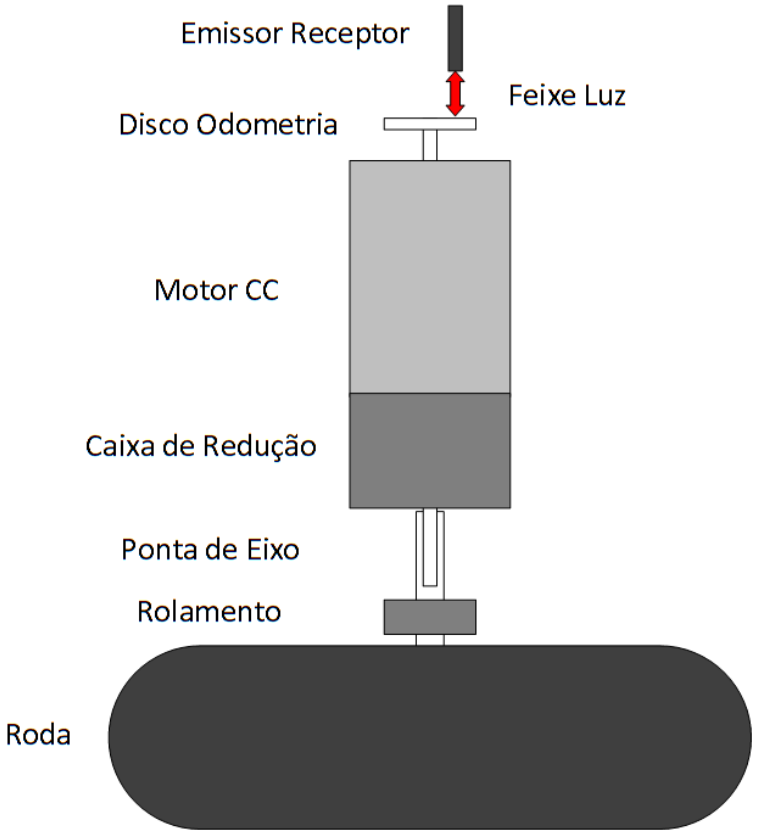


Figura 3.10 – Vista superior da odometria junto ao sistema de locomoção do Kihon.

Fonte: Elaborado pelo autor.

Como o *encoder* (disco de odometria) foi acoplado ao motor CC antes da redução, no eixo que gira em alta rotação, quando alimentado com 12V DC a 7600 rpm sem carga, foi decidido capturar apenas a transição de subida da onda quadrada, ao invés de ambas as rampas, de subida e descida. Isso significa que em sua velocidade máxima é esperado que o odômetro capture cerca de 7600 pulsos por minuto.

O motor CC utilizado, quando alimentado com 12V, sem carga, executa 7600rpm antes da redução e 80rpm após a redução, assim, é possível estabelecer a relação da caixa de redução, de 127:1. Sabendo-se que o eixo de saída do motor CC (após a redução) é conectada por meio de uma ponta doe eixo à roda, e que a roda possui diâmetro de 8,89 cm, e que são capturados 127 pulsos por rotação completa da roda, é possível calcular a resolução do contador de 1,5099 mm/pulso, conforme a Equação 3.2.

$$Resolução\ Odômetro = \frac{\pi * 8{,}89cm}{127 * 127} \tag{3.2}$$

Obviamente, tal cálculo desconsidera o fator de deformação da roda, uma vez que a roda empregada utiliza pneu inflado à base de borracha. Esta parte de borracha da roda é suscetível a deformações conforme a pressão que lhe é exercida, e que, neste caso, ocorre em função da massa do robô Kihon. Além disso, a massa do robô varia com a embarcação de dispositivos ou Sistemas de Controle.

As saídas dos dois sensores de odometria estão conectadas ao Contador 0 e Contador 1 do Controlador. Com isso é possível que o *software* de controle consiga estimar tanto a distância percorrida como a velocidade de cada roda do robô Kihon.

3.4.2.4 Sensores de proximidade

Para detecção de colisão, foram posicionados três módulos digitais de sensores de proximidade na parte frontal do robô Kihon. O sensor de proximidade adotado foi do tipo

infravermelho, modelo HWBZCGQ da Robox. A Figura 3.11 (a) ilustra o módulo, a Figura 3.11 (b) a disposição dos três módulos na estrutura mecânica, e a Figura 3.11 (c) os sensores já acoplados na parte frontal do Kihon.

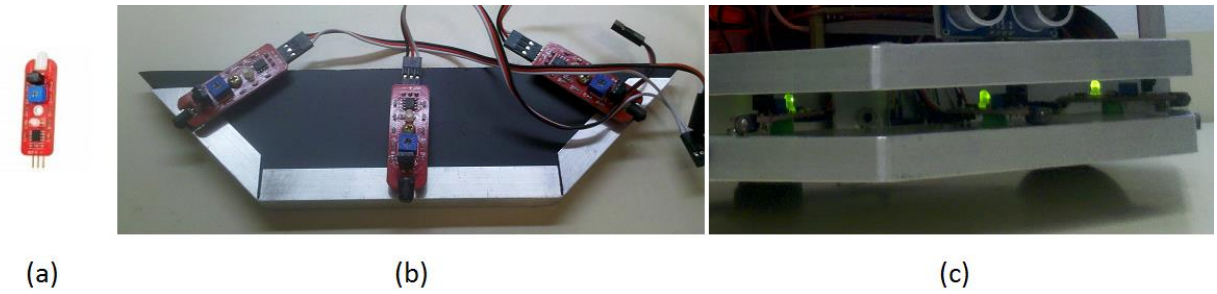


Figura 3.11 – (a) Sensor de proximidade empregado; (b) Disposição dos três sensores; e (c) Sensores acoplados na parte frontal do Kihon .

Fonte: Elaborado pelo autor.

O princípio de funcionamento desse módulo sensor de proximidade é similar ao conjunto emissor-receptor utilizado na odometria, no entanto, este é montado sobre uma pequena placa de circuito impresso; além disso, possui ajuste de sensibilidade e seus terminais se resumem ao Vcc, Out e Gnd, o que facilita sua utilização. Quando o foto-sensor obtiver retorno da luz emitida, e exceder o limiar definido pelo ajuste de sensibilidade, o nível lógico em sua saída será 0, caso contrário, apresentará nível lógico 1. Em outras palavras, quando o sensor detecta objeto, o mesmo retornará nível lógico 0, caso contrário, retornará nível lógico 1. Nominalmente, este sensor é capaz de detectar objetos a até 20cm. No entanto, a reflexão, como fenômeno óptico, depende do tipo e cor da superfície dos objetos onde se incide o feixe, e ainda, do ângulo de incidência do feixe. No caso do sensor de proximidade e, quanto ao seu posicionamento em relação aos objetos, quanto mais perpendicular à superfície, maior é o seu alcance de detecção e, de modo similar, quanto mais reflexiva a superfície dos obstáculos, maior é a distância de detecção. Na prática, este sensor se mostrou pouco eficaz, sendo capaz de detectar objetos a distâncias muito inferiores do esperado, como 5 ou 7 cm, para o tipo de superfície existente no ambiente onde fora realizados os experimentos.

As saídas dos sensores frontais de proximidade são conectados às portas digitais do Controlador. O sensor esquerdo é conectado à I/O Digital 2; o sensor central é conectado à I/O Digital 3; e o sensor direito é conectado à I/O Digital 4.

3.4.2.5 Sensores para seguir linhas

Para acrescentar ao robô móvel a habilidade de seguir linhas, foram utilizados dois OPB705 da Optek Technnology, o mesmo empregado para compor a odometria. A Figura 3.12 ilustra os dois sensores para seguir linhas, acoplados à estrutura mecânica do Kihon.

Figura 3.12 – Sensores para seguir linhas do Kihon.
Fonte: Elaborado pelo autor.

No entanto, para a leitura do solo, as saídas destes sensores foram acopladas às entradas do ADC do Controlador. As conexões das saídas dos foto-transistores às entradas analógicas do Controlador permitem que o robô Kihon possa navegar em superfícies que possuam coloração e tons diferentes (preto e branco), uma vez que superfícies com coloração dentro do intervalo entre branco e preto pode resultar em valores proporcionais de 0 a 5V na saída do foto-transistor. A estrutura mecânica do Kihon permite que os sensores possam ser posicionados de modo a otimizar a leitura do solo, permitindo que os sensores possam ser aproximados ou afastados em relação ao solo e também permite ajustes na distância entre os sensores. Os sensores para seguir linhas esquerdo e direito do Kihon foram conectados, respectivamente, aos canais ADC1 e ADC2 do Controlador.

3.4.2.6 Sonar

O Sonar - acrônimo de *Sound Navigation and Ranging* (navegação e determinação de distância pelo som), é utilizado na robótica móvel para auxiliar na sua localização relativa e no mapeamento de regiões próximas ao robô. Como descrito em 3.5.1, um sonar acoplado ao eixo de um servomotor permite ao robô Kihon medir distâncias de eventuais objetos ou obstáculos, que surjam na região frontal do robô. Para o sonar, foi utilizado o HC-SR04, um sonar monoestático, direcional, ativo e ultrassônico, capaz de mensurar distâncias no intervalo de 2cm a 4 m e precisão de até 3mm. A Figura 3.13 (a) ilustra o módulo empregado, e a Figura 3.13 (b), mostra o Sonar já acoplado ao servo motor e embarcado no Kihon.

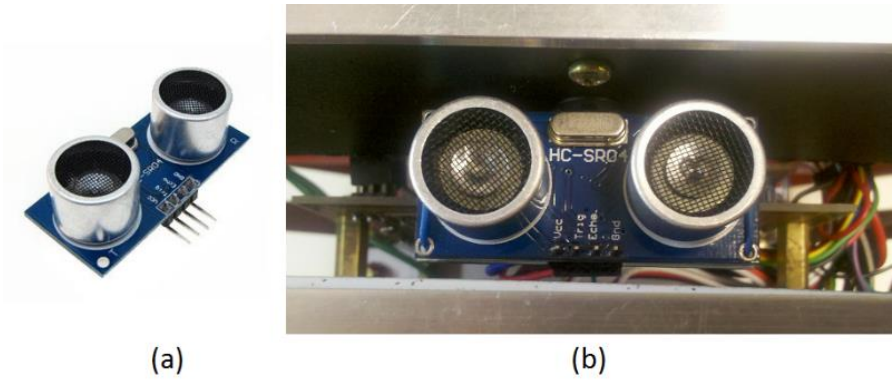


Figura 3.13 – (a) Sensor Sonar utilizado no Kihon; e (b) Sonar acoplado ao Kihon.
Fonte: Elaborado pelo autor.

O processo de detecção e/ou localização de objetos a partir do Sonar é chamada de *ecolocalização*. A ecolocalização baseia-se na propagação física do som, na qual, após emitido, sua reflexão e retorno ao emissor como eco é utilizada para estimar a localização de objetos ou obstáculos.

Quando o *Trigger* do sonar é acionado, seu transmissor emite um pulso direcionado de som ultrassônico e, então, é iniciada a contagem de tempo. Este som se propaga pelo ambiente, e ao atingir um obstáculo, ocorre uma reflexão do som, que retorna ao receptor do

sonar. O sonar registra o intervalo de tempo decorrido entre a transmissão e a recepção do som propagado, possibilitando a medição da distância entre o sonar e o objeto mais próximo detectado. Por fim, através do terminal de *Echo*, o sonar informa a distância mensurada. A Equação 3.3 mostra o cálculo para determinar a distância D_{sonar} detectada pelo sonar.

$$D_{sonar} = \frac{1}{48MHz} * 4 * 8 * TMR1 * \frac{1}{2} * \frac{340m}{s} \quad , \tag{3.3}$$

onde TMR1 corresponde ao timer encarregado de contar o tempo decorrido entre o envio da onda e seu retorno.

De modo a tornar a resposta à requisição de medição do sonar mais rápida e não deixar a cargo do microcontrolador efetuar cálculos que consumirão ciclos de clock extras, foi decidido não atribuir ao Controlador a tarefa de efetuar esse cálculo. Portanto, o valor do TMR1 será o valor retornado pelo Controlador cada vez que o sonar é acionado para efetuar a medição. Isso quer dizer que existe a necessidade do *Software* de Controle efetuar a conversão da distância para metros ou centímetros.

Os terminais de *Trigger* e *Echo* do Sonar foram conectados aos terminais I/O Digital 7 e Contador 3 do Controlador, respectivamente. Mais detalhes a respeito do funcionamento do sonar podem ser encontrados no datasheet do componente utilizado. O Controlador concebido dispõe de contadores de 2 bytes, sendo assim, sua contagem em decimal vai até 65535. Isso quer dizer que, para o Sonar, é possível medir distâncias de objetos a até 7,4 metros, sem que o contador "estoure". E, de modo a não limitar a contagem para o alcance máximo de 4m do sonar utilizado, foi decidido não resetar o contador caso o sonar não receba o eco após contagem atingir 35295, valor este equivalente às interrupções necessárias para aguardar o eco de objetos localizados a até 4 metros de distância. Assim, é possível acoplar sonares com alcance de até 7,4 metros para esta configuração do contador do Controlador.

O gerenciamento dos módulos periféricos apresentados acima é realizado por meio de um único agente, o Controlador de Robô Móvel. Para este Controlador de Robô Móvel, buscou-se um microncontrolador que possuísse uma quantidade suficiente de recursos de

E/S (entrada e saída) e que, somados à sua capacidade de processamento, programabilidade e periféricos de comunicação, permitisse sua utilização de forma descomplicada para os usuários, sejam alunos ou pesquisadores. Tendo isso em mente, dentre outras possibilidades de escolha, optou-se pela utilização do PIC18F4550, um modelo de microcontrolador da família PIC produzido pela Microchip®. Diversos fatores contribuíram para sua escolha, como enquadramento ao escopo do projeto, baixo custo, baixo consumo, disponibilidade no mercado, disposição imediata no LISDI, dentre outros. Além disso, este microcrontrolador possui funcionalidades que permitem programação otimizada em compiladores C, por meio do compilador MPLAB da Microchip®.

3.4.2.7 Fonte de energia

De nada adiantaria satisfazer os requisitos da composição de um robô móvel, se o mesmo não tiver uma fonte de energia para alimentá-lo. Possuir uma fonte de força motriz é essencial para que todos os componentes funcionais do robô móvel possam operar satisfatoriamente. Para tal, uma grande parcela dos robôs móveis fazem uso de baterias para o fornecimento de energia elétrica a seus sistemas, que incluem circuitos, microcontroladores, motores, sensores, dentre outros dispositivos e componentes.

Baterias armazenam energia química e a convertem em energia elétrica sob demanda. Existem baterias comerciais de tipo recarregável e não recarregável. As baterias recarregáveis podem ser compostas por diferentes composições químicas e, por conta disso, apresentam diferentes paradigmas de descarga e recarga. No entanto, todas elas estão sujeitas a atingir um ponto em que sua recarga não é mais efetivamente possível, sendo assim, necessária a sua substituição. Em vista de seu custo relativamente baixo, e também por razões de praticidade, é comum a utilização de baterias recarregáveis na robótica móvel. Além do baixo custo e da facilidade de acesso no mercado, existem diversos tipos e tamanhos de baterias recarregáveis, desde as menores, empregadas, por exemplo, na telefonia móvel e computadores portáteis, até as de maior porte, comuns na indústria automobilística.

Para a concepção do robô móvel Kihon, foi decidido combinar serialmente duas baterias comerciais de 6V/4,5Ah, seladas de chumbo ácido, frente às outras alternativas, como por exemplo, adotar uma única bateria de 12V/7Ah. Os 12V obtidos através da disposição serial das baterias empregadas possibilitam alimentar motores de corrente contínua de até 12V, além de gerar tensão suficiente para alimentar reguladores de tensão para níveis TTL e CMOS.

Alguns fatores foram considerados na adoção da disposição escolhida, como o custo relativamente baixo em relação às outras opções e boa relação entre capacidade de carga, dimensões, peso e tamanho. Um último aspecto que influenciou na sua escolha foi devido à sua imediata disponibilidade para uso no GISDI. Além disso, em termos de espaço e disposição física, apesar de empregar 2 unidades, foi possível distribuí-las de forma mais conveniente em virtude de suas dimensões individuais. A Figura 3.14 ilustra a disposição física das baterias utilizadas.

Figura 3.14 - Disposição das baterias empregadas.

Fonte: Elaborado pelo autor.

Considerando a questão da autonomia da configuração adotada, visto que os motores empregados consomem individualmente, aproximadamente 50mA quando em operação (sem carga e desconsiderando os picos de corrente de partida), somados aos circuitos eletrônicos e ao sistema de controle embarcado, foi estimada uma autonomia energética da plataforma de cerca de pelo menos 5 horas de utilização contínua da mesma. Isto é, supondo

que consumo aproximado médio seja inferior a 800mA de toda a parte eletro-mecânica e computacional embarcada, já que somente o robô móvel Kihon apresenta consumo em torno de 220mA quando ligado e, pelo menos, 450mA quando em movimento e sem carga.

De modo a oferecer facilidade de uso ao robô móvel Kihon, foi dimensionado um sistema de recarga das baterias empregadas. O sistema de recarga se resume a um simples circuito de recarga e a uma fonte de tensão contínua externa. O circuito de recarga foi elaborado de modo a ser alimentado por uma fonte externa de corrente contínua de 16V, sendo composto basicamente por um limitador de corrente resistivo, de modo que a corrente de recarga não ultrapasse 10% da capacidade de carga das baterias (carga rápida), ou seja, 450mA. Logo, o tempo máximo de recarga é de cerca de 10 horas para carga completa das baterias empregadas. A Figura 3.15 ilustra o conector disponibilizado para a recarga das baterias do Kihon, além da chave liga/desliga e do LED indicador de estado operacional do Kihon.

Figura 3.15 – Conector de Recarga, chave L/D e LED de sinalização na lateral do Kihon.
Fonte: Elaborado pelo autor.

O LED bicolor posicionado ao lado da chave liga/desliga do robô móvel Kihon foi empregado para sinalizar visualmente as 4 situações possíveis de uso do robô móvel Kihon. Na primeira situação, com o robô com sua chave na posição desligado e com o carregador desconectado, este LED permanecerá apagado. Na segunda situação, com sua chave na posição ligado e o robô móvel Kihon operacional, o LED assumirá a cor verde. Na terceira situação, com a fonte de recarga plugada na tomada e conectada ao robô móvel Kihon, e com a chave na posição desligada, o LED acenderá a cor vermelha, indicando que o recarregador está plugado. Por último, com a fonte de recarga na tomada e conectada ao robô móvel Kihon,

e este com a chave na posição ligada, o LED acenderá simultaneamente as cores vermelha e verde, já que pela implementação do LED indicador e do circuito de recarga, o LED acenderá vermelho sempre que o recarregador estiver plugado ao robô móvel, independentemente da posição da chave liga/desliga.

3.4.3 Concepção da Camada Intermediária

Para averiguar a arquitetura da plataforma proposta neste trabalho, foi necessária a implementação da Camada Intermediária, apresentada em 3.2. Para tal, contou-se com o desenvolvimento do *firmware* do controlador, protocolo de comunicação e também da concepção de uma API para o *software* de controle, este último que pode ser remoto ou embarcado no Kihon.

O *firmware* como Controlador de Robôs Móveis compõe o *software* embarcado da plataforma, atuando como cliente na arquitetura proposta, e foi desenvolvido no microcontrolador PIC18F4550 da Microchip®. O ambiente de desenvolvimento utilizado para a concepção do *firmware* foi o MPLAB C Compiler for PIC18 MCUs da Microchip®, um componente integrável à IDE do MPLAB também da Microchip®. O funcionamento geral do *firmware* implementado é descrito por meio do fluxograma apresentado na Figura 3.16.

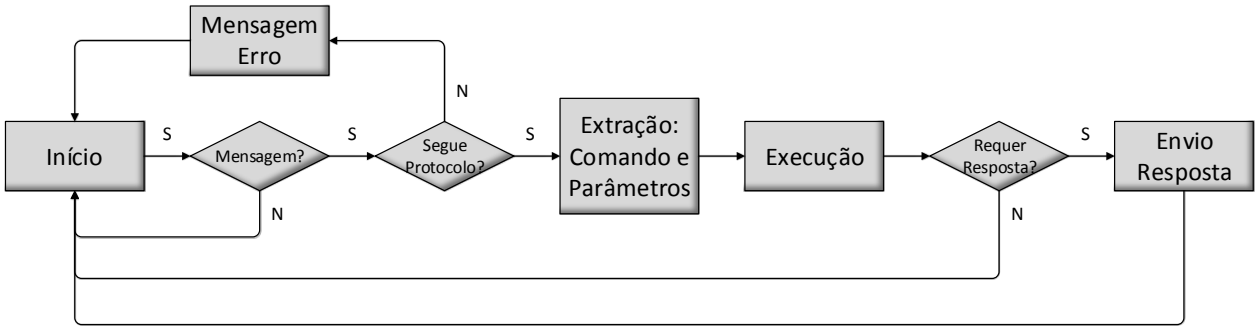


Figura 3.16 – Fluxograma do *Firmware* do Controlador.

Fonte: Elaborado pelo autor.

3.4.3.1 Interface de comunicação e canais de alimentação de baixa potência

A comunicação da plataforma de robô móvel Kihon ao sistema de controle é estabelecida através de sua interface de comunicação serial. Neste trabalho, foi adotado o padrão de comunicação RS-232 em conjunto com o padrão DB9 de conector físico. O padrão USART implementado em modo full-duplex dedica um canal para envio e outro para recepção de dados, possibilitando envio e recepção simultâneo. Adicionalmente, os canais de comunicação TX e RX do microcontrolador, ou seja, seus respectivos pinos 25 e 26 do PIC18F4550, foram dispostos diretamente a um conector genérico na tampa de acrílico do robô móvel. O nível de tensão de operação desses terminais TX e RX é o TTL, ou seja, de 5V CC. A Figura 3.17 ilustra o conector DB9 localizado na lateral do Kihon e o conector genérico na tampa no Kihon.

Figura 3.17 – Interfaces de comunicação no Kihon.

Fonte: Elaborado pelo autor.

Ainda neste conector genérico, junto aos canais TX e RX, foram dispostas linhas de 5V, 3.3V e GND. As tensões de 5V e 3.3V são comuns na eletrônica digital, uma vez que a padronização de componentes TTL (acrônimo de *Transistor-Transistor Logic*, ou seja, Lógica Transistor-Transistor) adota a tensão de 5V, e tecnologias CMOS (acrônimo de Complementary Metal-Oxide Semiconductor, ou seja, Semicondutor Metal-Óxido Complementar) adota tensão de 3.3V. Por meio dos reguladores LM7805 e LP2905, para a disponibilização de 5V e 3.3V, respectivamente, o fornecimento de ambas as tensões oferece

proteção contra quaisquer problemas de dimensionamento de carga, uma vez que ambos os reguladores possuem internamente circuitos de proteção contra curto-circuitos e limitadores de corrente. Desta forma, a plataforma é capaz de alimentar sistemas embarcados de baixa potência de forma segura, além de oferecer uma interface de comunicação serial por meio de um conector físico genérico.

Esta interface genérica foi implementada visando o fornecimento de alimentação de tecnologias de baixa potência como microcontroladores, Raspberry Pi e Arduino, atuando como plataforma de controle embarcada. O emprego de outras tecnologias como netbook embarcados não necessita das linhas de 5V e 3.3V, uma vez que o equipamento dispõe de bateria e autonomia energética própria.

A alternativa de fornecer uma linha de alimentação de 12V foi considerada, no entanto, a disponibilização dessa linha de 12V implicaria em uma linha obtida diretamente a partir das baterias empregadas. Para o fornecimento seguro de uma linha de tensão de 12V, seria necessário o uso de reguladores, como o LM7812, mas estes reguladores de 12V, de acordo com os fabricantes, necessitam que sua alimentação seja de pelo menos 19V, tensão esta que não pode ser obtida por meio da associação serial das baterias empregadas, de 6V cada uma. Logo, nesta implementação, não é possível obter a tensão de 12V de forma segura para o usuário do robô móvel Kihon. O fornecimento de uma linha de 12V direta pode ser disponibilizada ao usuário, desde que este tenha plena ciência de que está lidando com uma linha desprotegida contra eventuais problemas de dimencionamento de carga e/ou curto-circuito – uma prática não tão incomum em usuários em fase de iniciação à eletrônica e circuitos digitais. Nestas condições, qualquer situação imprópria e sem o uso adequado de componentes de proteção poderia resultar em danos permanentes aos componentes do circuito e fiação e, na pior hipótese, perigos de incêndio e explosão das baterias utilizadas. Um simples componente de proteção que poderia ser empregado na linha de 12V é o fusível, cuja corrente poderá ser dimensionada de acordo com o consumo de carga do dispositivo a ser alimentado.

3.4.3.2 Protocolo de comunicação

O protocolo de comunicação adotado é enxuto, ainda que implemente recursos de verificação de erros de comunicação, como conexão, transmissão e recepção das mensagens. Neste protocolo, todas as mensagens enviadas pelo Sistema de Controle (e não pelo Controlador) devem ser iniciadas e finalizadas por caracteres específicos. O caracter de início foi definido como sendo o "<", ou seja, o sinal de menor. Já o caractere de fim foi definido como sendo o ">", ou seja, o sinal de maior. Desta forma, a comunicação é sempre iniciada pelo *software* de Controle, que envia um vetor de bytes iniciados e finalizados, respectivamente, pelos caracteres "<" e ">". Já as mensagens enviadas pelo Controlador não possuem essa estruturação, ou seja, qualquer dado de resposta transmitido pelo Controlador é feita sem esses caracteres de início e fim. O Controlador nunca inicia uma transmissão de dados, mas responde às requisições e comandos recebidos pelo Sistema de Controle quando necessário.

O Controlador, ao receber uma dada mensagem, interpreta o primeiro byte após o caracter de início "<". De acordo com o comando especificado por este primeiro byte, o Controlador tratará os demais bytes como parâmetros, enviando uma resposta ao Sistema de Controle quando necessário. O formato geral das requisições enviadas pelo *software* de controle é mostrado na Figura 3.18.

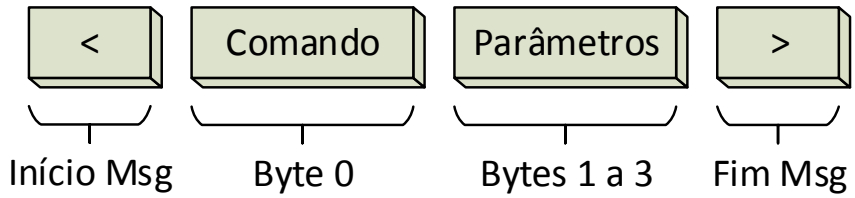


Figura 3.18 – Formato geral das requisições.

Fonte: Elaborado pelo autor.

Um resumo das requisições é apresentada na Tabela 3.1. O detalhamento de cada requisição é descrito no item F do Apêndice C.

Tabela 3.1 - Requisições enviadas para o Controlador por meio da comunicação serial.

Requisição	Byte 0	Byte 1	Byte 2	Byte 3
Leitura das entradas ADC	0 ASCII	-----	-----	-----
Leitura dos Contadores	1 ASCII	-----	-----	-----
Medição do Sonar	2 ASCII	-----	-----	-----
Configuração da frequencia PWM	3 ASCII	Período	Prescaler	-----
Configuração dos ciclos PWM	4 ASCII	DcycleA	DcycleB	DcycleC
Configuração de controle PWM	5 ASCII	Estado	-----	-----
Configuração do tempo do Servo 0	6 ASCII	Tempo A	Tempo B	-----
Configuração do tempo do Servo 1	7 ASCII	Tempo A	Tempo B	-----
Configuração do tempo do Servo 2	8 ASCII	Tempo A	Tempo B	-----
Configuração do tempo do Servo 3	9 ASCII	Tempo A	Tempo B	-----
Configuração I/O Digitais	10 ASCII	Sentido	-----	-----
Leitura I/O Digitais	11 ASCII	-----	-----	-----
Escrita I/O Digitais	12 ASCII	Estado	-----	-----

A API do Controlador da Plataforma de Robô Móvel foi elaborada com o intuito de intermediar e facilitar o processo de comunicação entre o *software* de Controle e o Controlador. Esta API tem a função de abstrair o envio de comandos, interpretar os dados de resposta enviados pelo Controlador, e ainda é encarregado de efetuar eventuais cálculos e conversões necessários para a configuração dos dispositivos do PIC, de modo a aliviar a carga de processamento do microcontrolador utilizado. Uma vez o usuário utilize esta API, ele não precisa conhecer a fundo os detalhes técnicos dos módulos de dispositivos de *hardware* para fazer uso da plataforma. A API foi escrita em linguagem C++, uma vez que esta linguagem oferece uma série de facilidades de comunicação com portos de E/S (entrada e saída) e também por se tratar de uma linguagem popular ensinada na maior parte das universidades, além de possuir uma base vasta de conhecimento na internet. Isto a torna uma forte linguagem para fins educativos, embora ainda haja muita discussão sobre qual é a melhor linguagem para ensinar programação aos alunos.

Para o desenvolvimento da API, foram utilizados o complilador MinGW 5.1.6 , a IDE (*Integrated Development Environment*) Eclipse Kepler, e a API nativa do Windows. Apesar de

existir a possibilidade de utilizar um ambiente Linux, o ambiente Windows foi escolhido devido à sua popularidade.

O diagrama da classe Controlador é ilustrada na Figura 3.19.

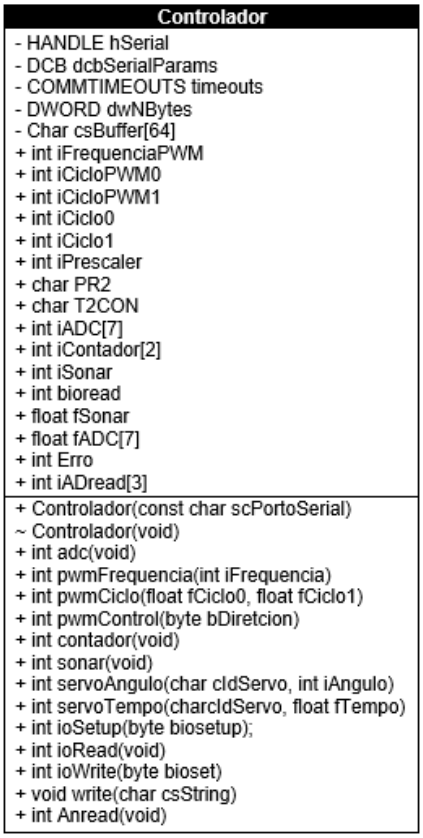


Figura 3.19 – Diagrama da Classe Controlador.

Fonte: Elaborado pelo autor.

3.5 Considerações sobre a Plataforma de Robô Móvel

Existem diversos aspectos da Plataforma de Robô Móvel que podem ser melhorados futuramente. Nesta seção, serão expostos e discutidos alguns destes aspectos.

3.5.1 Solução baseada em um único microcontrolador

O Controlador concebido neste projeto tem como base um único microcontrolador, o PIC18F4550, conferindo-lhe uma arquitetura embarcada fortemente acoplada. Isso ainda confere ao projeto uma reduzida quantidade de componentes, resultando em um baixo custo de construção e manutenção de *hardware*, além de reduzir as chances de problemas de conectividade e falhas. No entanto, o desempenho obtido pode não ser o melhor possível, já que o Controlador possui suas próprias limitações. O procedimento de interrupção dos contadores consomem, no melhor caso, 56 ciclos de instrução, o que, teoricamente, permitem a contagem de pulsos a uma frequência máxima de 214kHz. A frequência mínima alcançada pelos módulos de PWM é de 2,9kHz, um valor significativamente maior do que o requerido por muitas aplicações. Além disso, a aquisição de dados analógicos (ADC) não ocorre em canais dedicados, dependendo do chaveamento (multiplexação) de canais do conversor, e sua execução demanda uma parte considerável do processamento do PIC nesta arquitetura.

Com a utilização de componentes e dispositivos dedicados para separar cada uma dessas funcionalidades (contador de pulsos, módulo de PWM e ADC), ou seja, por meio da exploração da modularização e da granularização dos dispositivos, existe a possibilidade de se obter ganhos no desempenho do Controlador, uma vez que o mesmo deixará de compartilhar a CPU para processar as diferentes rotinas de cada módulo. Tal arquitetura é mostrada na Figura 3.20.

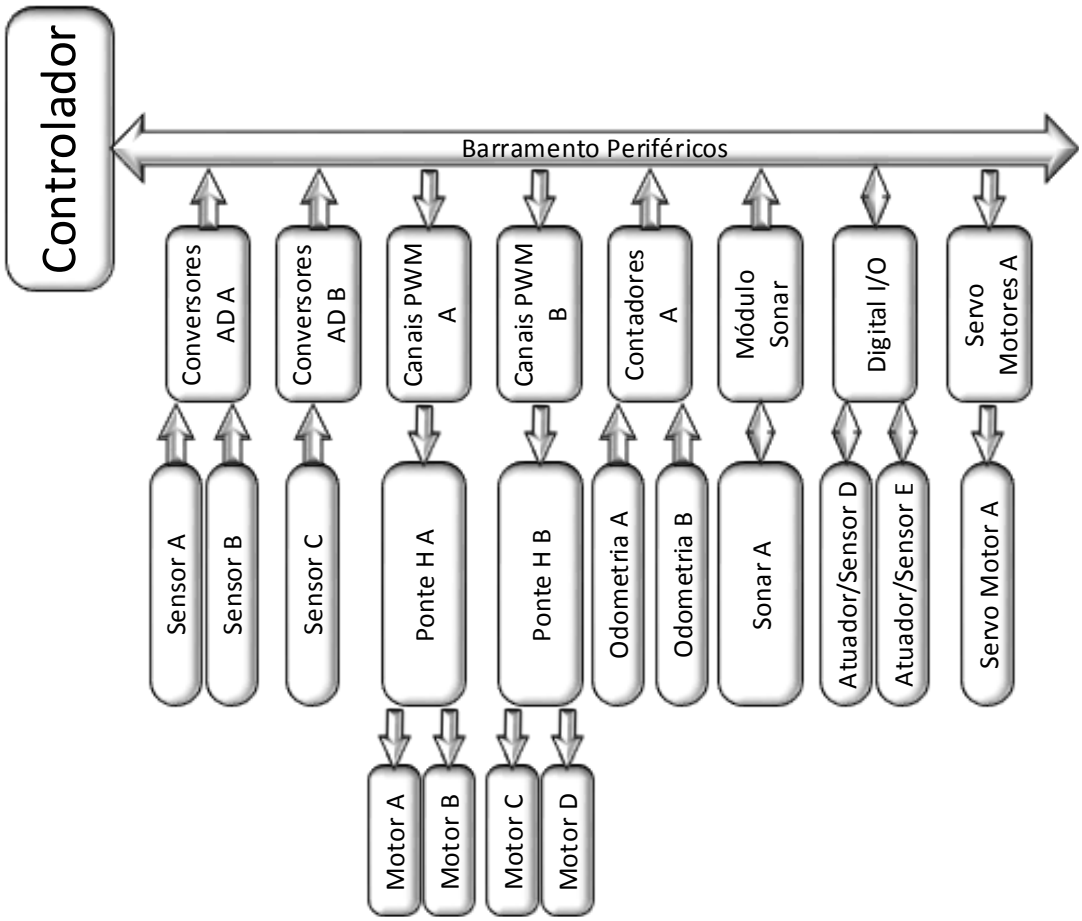


Figura 3.20 – Arquitetura de um Sistema Fracamente Acoplado.

Fonte: Elaborado pelo autor.

No entanto, por necessitar de microcontroladores dedicados para cada módulo de dispositivo periférico, tal abordagem necessita de uma maior quantidade de microcontroladores. Isso acarreta em um custo relativamente maior quando comparado à uma arquitetura fortemente acoplada. E ainda, um outro fator é o barramento de comunicação entre o Controlador e os dispositivos periféricos, que precisará ser implementado, seja criando um ou adotando padrões de barramentos já existentes, como por exemplo o I^2C, CAN ou SPI. Nesta disposição, os módulos conectados ao barramento de periféricos são capazes de enviar e receber dados do controlador, mas não simultaneamente. Como é possível observar, essa abordagem é também visivelmente mais complexa em comparação à adotada neste trabalho.

3.5.2 Escolha da comunicação serial

O porto serial é um padrão de comunicação que, em virtude de sua concepção, apresenta uma série de restrições. O primeiro aspecto é seu caráter de legado, junto ao porto paralelo. Ambos os padrões, serial e paralelo, são limitados à conectividade ponto a ponto e, nos dias atuais, suas taxas de transferência são relativamente muito inferiores, quando comparados como, por exemplo, ao USB (*Universal Serial Bus*). O porto serial apresenta taxa de transferência de até 115,2 Kb/s e o porto paralelo, chega a 2 Mb/s. Já o USB, ainda na versão 1.0 é capaz de atingir 1,5 Mb/s, o USB 2.0 pode chegar a 470Mb/s e finalmente, o USB 3.0 pode alcançar 4,8Gb/s.

O USB, por exemplo, oferece uma série de vantagens em comparação ao porto serial e paralelo, como a capacidade de conectar vários dispositivos periféricos e reconhecer o dispositivo automaticamente pelo computador. No entanto, de modo a oferecer tais vantagens, o USB acaba trazendo uma série de preocupações inexistentes em comparação aos portos legatórios, como maior complexidade e uma extensa documentação, devido aos seus diferentes modos de operação e configuração, além de necessitar de componentes dedicados para a comunicação. O PIC18F4550 utilizado possui registradores, instruções e *hardware* interno dedicado para a comunicação USB 2.0. Entretanto, o USB necessita que o usuário conheça uma série de conceitos, que podem desmotivar sua utilização.

Existem outras alternativas de comunicação além do USB 2.0, como o padrão I^2C e o SPI, que são suportados pelo PIC18F4550 utilizado. O I^2C, acrônimo de *Inter-Integrated Circuit*, permite a conexão de múltiplos dispositivos escravos e opera em *half duplex* no modo mestre-escravo com apenas duas linhas bidirecionais de 3,3V ou 5V – ainda que outros valores de tensão sejam possíveis. Além disso, o I^2C é empregado majoritariamente em dispositivos embarcados e pode operar com várias taxas de comunicação, incluindo 100kbit/s no chamado *Standard Mode* e pode chegar aos 3,4Mbit/s no chamado *High Speed Mode*. Já o SPI, acrônimo de *Serial Peripheral Interface*, é um barramento serial síncrono e *full duplex* de comunicação que permite ao microcontrolador se conectar a diversos dispositivos também em modo mestre-escravo, apesar de necessitar de pelo menos 4 canais de

comunicação. O SPI não define uma taxa de comunicação máxima, no entanto, sendo *full duplex*, é capaz de atingir velocidades muito superiores em relação ao I^2C – que é *half duplex*. No entanto, ambas as alternativas, de forma similar ao caso do USB, possuem vasta documentação e suas utilizações são significativamente mais complexas em relação ao porto serial, o que as tornam até certo ponto desestimulantes aos usuários.

Portanto, apesar da sua limitada taxa de comunicação, optou-se pela utilização do porto serial em virtude de sua simplicidade, facilidade de implementação tanto em termos de *hardware* como *software*.

Em trabalhos futuros, é possível implementar o Controlador utilizando protocolos como o I^2C, SPI e USB, já que o PIC utilizado possui módulos que dão suporte a tais padrões de comunicação.

3.6 Conclusões do capítulo

Este capítulo apresentou a plataforma do robô móvel e detalhou os módulos que compõem a plataforma. Como já mencionado, por limitação de tempo e orçamento, sua implementação visou oferecer suporte as iniciativas essenciais relacionadas à robótica móvel.

O capítulo 4 é destinado aos testes e verificações se a plataforma concebida foi capaz de alcançar os objetivos almejados e que motivaram o seu desenvolvimento.

4 TESTES E VALIDAÇÃO

Neste capítulo são discutidos os testes de validação da plataforma proposta do robô móvel Kihon e a análise dos resultados obtidos. Os experimentos foram elaborados com o intuito de averiguar os quesitos da plataforma que motivaram seu desenvolvimento. Inicialmente, avaliou-se a capacidade da plataforma em oferecer suporte às tarefas estabelecidas pelas aplicações, além de verificar a questão da sua facilidade de uso, tanto em termos de *hardware* como de *software*. Do ponto de vista de *hardware*, os aspectos observados foram sua adaptabilidade e facilidade de manutenção, importantes principalmente em vista do direcionamento dado a este trabalho. Já do ponto de vista de *software*, os aspectos observados foram se a plataforma oferece liberdade, por ser de código aberto, e independe da linguagem de programação do Sistema de Controle, uma vez que a interface de comunicação permite tal característica. Por último, se a arquitetura de comunicação apresenta flexibilidade quando submetido a diferentes tecnologias de Sistema de Controle. A partir dos experimentos, foi possível verificar se a plataforma satisfaz todos os requisitos e objetivos almejados em 1.3 e 1.4.

4.1 Experimento 1

O primeiro experimento teve como objetivo averiguar a capacidade da plataforma de oferecer suporte a uma aplicação hipotética e, neste caso, buscando uma situação próxima a um estudo básico sobre navegação. A questão da usabilidade da API concebida também foi verificada. Para tal, foi elaborado um experimento no qual verifica-se a capacidade da plataforma de navegar através de um ambiente, respondendo aos estímulos do ambiente em tempo real, quando controlado de forma remota. Um dos aspectos que também pode ser observado neste experimento é se a disposição dos sensores e dos atuadores escolhidos e implementados no robô móvel o torna capaz de se locomover e navegar apropriadamente através de seu ambiente de aplicação. Neste sentido, como tarefa escolhida para o

experimento, foi determinado que o robô deveria navegar pelo ambiente seguindo uma parede e evitando colisões. A averiguação se deu tendo como base o comportamento do robô móvel Kihon ao interagir com o ambiente. Este experimento foi realizado empregando um *notebook* atuando como uma base computacional remota e sua comunicação sem fio foi tecnologicamente estabelecida por meio de módulos Xbee.

O diagrama de conexão cliente-servidor estabelecido neste experimento, entre o robô móvel Kihon ao sistema de controle é ilustrada na Figura 4.1.

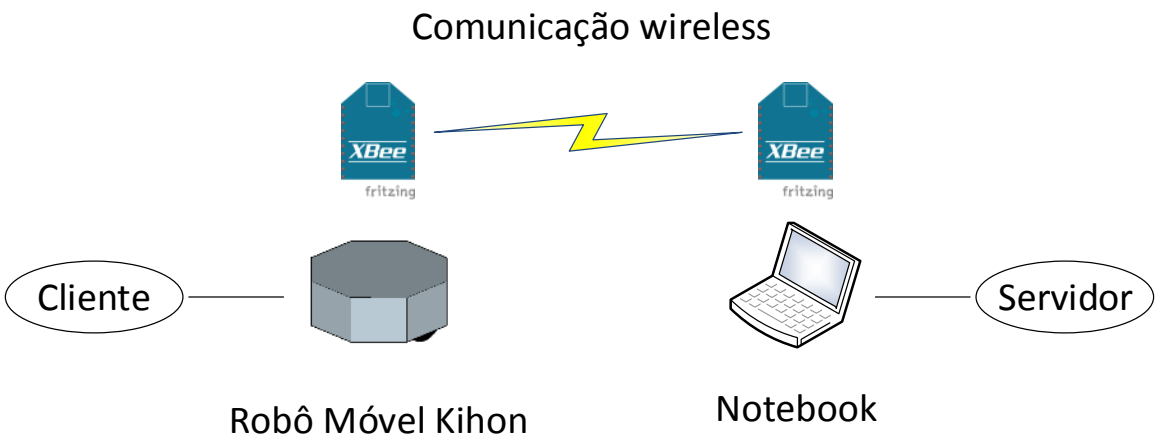


Figura 4.1 – Diagrama cliente-servidor do Experimento 1.

Fonte: Elaborado pelo autor.

Este experimento buscou se aproximar das condições usuais, nas quais os usuários utilizam seus computadores pessoais para desenvolver experimentos envolvendo, por exemplo, técnicas de navegação e locomoção de robôs móveis. A base computacional remota executa o *software* do Sistema de Controle em um computador cujo sistema operacional é o Microsoft Windows 8. Computadores pessoais, em especial os *notebooks*, em conjunto com o sistema operacional da plataforma Windows, costumam ser familiares na maior parte dos usuários, tornando tais ambientes uma alternativa comum para o desenvolvimento de *software*, seja por parte dos usuários, como alunos e, até mesmo, pesquisadores. A linguagem de programação empregada foi a C++, uma vez que a API do Controlador já havia sido concebida anteriormente nessa linguagem. Além disso, o C++ frequentemente se apresenta

como uma linguagem didática no desenvolvimento de algoritmos, sendo comum em disciplinas da graduação e, também, como linguagem para desenvolvimento de *software*.

Para realizar este experimento, foi utilizado um sensor de distância (Sonar) acoplado ao servomotor para realizar a varredura da parte frontal do robô móvel e auxiliado pelos três sensores de proximidade frontais. A Figura 4.2 ilustra a vista superior do Kihon e a disposição dos sensores frontais de proximidade, sonar e do servomotor.

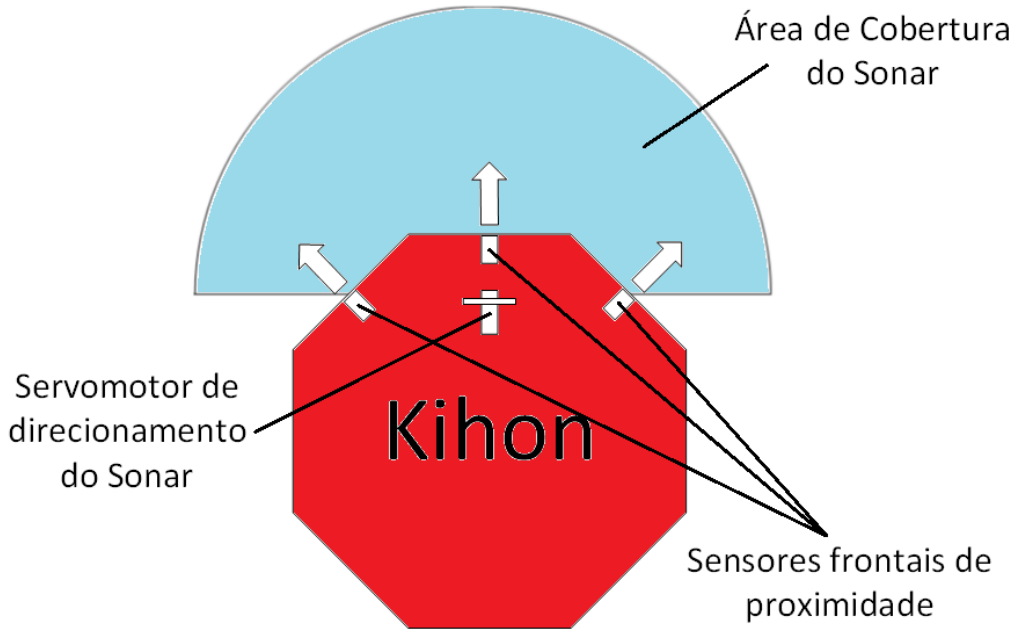


Figura 4.2 – Disposição dos sensores do Kihon utilizados no Experimento 1.

Fonte: Elaborado pelo autor.

A arquitetura de controle concebida para o experimento se enquadra na arquitetura de subordinação (ou subsunção) apresentada por Brooks (1986), característico do paradigma comportamental, uma vez que baseia-se no comportamento observado em animais. A partir disso, um simples algoritmo de seguir parede foi implementado. Neste algoritmo, uma série de 5 medições do sonar é coletada ao longo dos 180 graus (sendo uma medição a cada 45 graus) da parte frontal da base do robô móvel Kihon, realizado com auxílio do servomotor. Essas medições são feitas de modo a detectar a presença da parede em conjunto com leituras dos sensores de proximidade, na detecção de uma colisão frontal iminente. A partir das medições levantadas pelos sensores, o robô navega contornando e acompanhando a parede

encontrada. A velocidade de tração dos motores foi controlada por modulação de largura de pulso – PWM (acrônimo do termo em inglês *Pulse Width Modulation*). O diagrama de execução do algoritmo usado é mostrado na Figura 4.3.

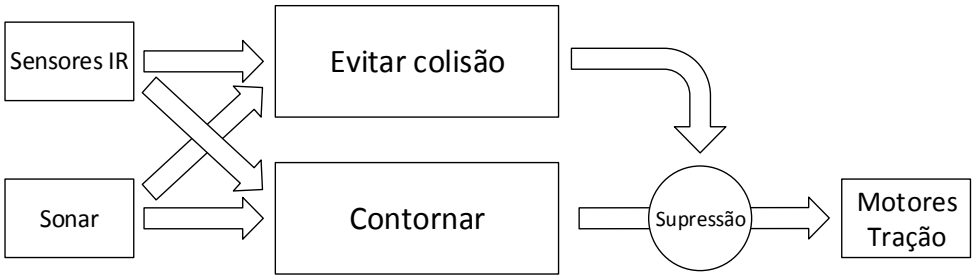


Figura 4.3 – Diagrama do algoritmo reativo de navegação.

Fonte: Elaborado pelo autor.

Por meio de chamadas às rotinas contidas na Classe Controlador da API (descrito em 3.4.3), o algoritmo de navegação é capaz de ter acesso aos módulos de *hardware* do Kihon. Algumas das chamadas utilizadas neste experimento são exemplificadas na Tabela 4.1.

Tabela 4.1 – Exemplos de chamadas às rotinas da Classe Controlador API utilizadas no Experimento 1.

Exemplo de Requisição	Chamada à API	Retorno
Config. Angulo do servo motor 0 a 35º	controlador->servoAngulo(0, 35)	-----
Medição do Sonar	controlador->Sonar()	Distância em cm
Config. Movimentação Kihon p/ frente	controlador->pwmControl(frente)	-----
Config. Ciclo PWM: 85% Esq e 65% Dir	Controlador->pwmCiclo(0.85, 0.65)	-----
Leitura Portas I/O Dig. (sensors prox.)	Controlador->ioRead()	Estado de todas I/O Digitais

O trajeto percorrido pelo Kihon neste experimento com o algoritmo implementado é mostrado na Figura 4.4.

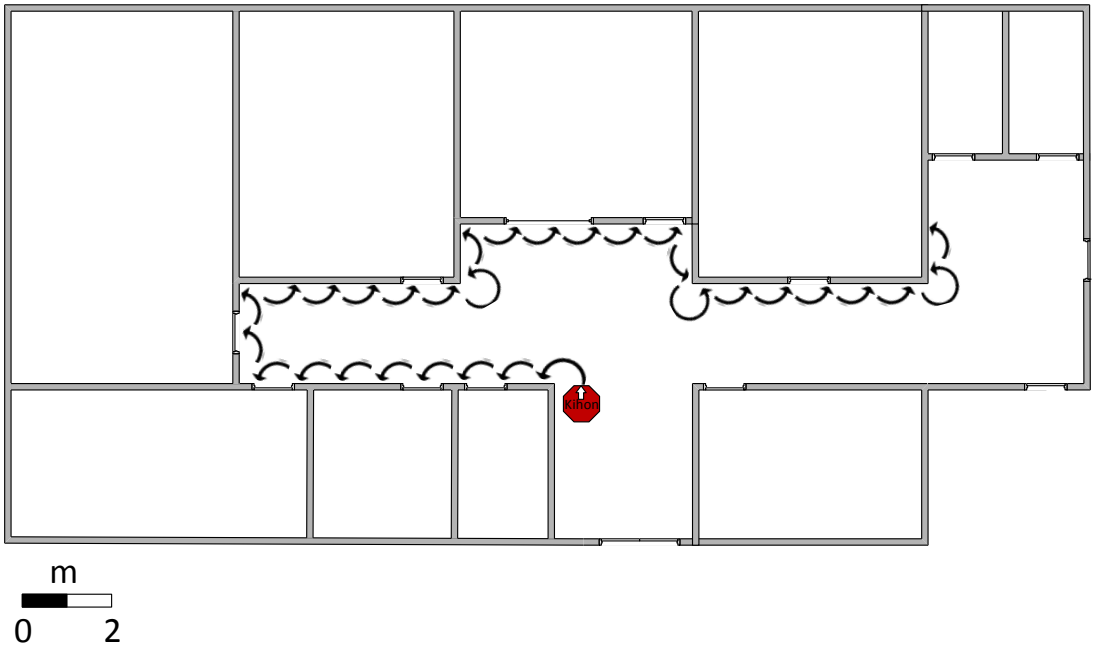


Figura 4.4 – Trajeto percorrido pelo Kihon com o algoritmo implementado.

Fonte: Elaborado pelo autor.

A Figura 4.5 ilustra o Experimento 1 em andamento com o Kihon navegando através do ambiente.

Figura 4.5 – Kihon em execução do Experimento 1.

Fonte: Elaborado pelo autor.

4.1.1 Considerações

Analisando a capacidade da plataforma em oferecer suporte a uma aplicação hipotética envolvendo navegação de robôs móveis, que de certa forma culminam na escolha da arquietura de *hardware* e também na disposição dos dispositivos de sensoriamento e atuação escolhidos, o robô móvel Kihon se mostrou capaz de se locomover bem através do ambiente, observando-se os seguintes aspectos:

1. Os motores CC empregados no tracionamento se mostraram capazes de operar com ciclos de trabalho acima dos 50% em virtude da elevada frequência PWM gerada pelo Controlador, de aproximadamente 2.93kHz. Esta frequência é a menor possível obtida no microcontrolador utilizado. Por causa disto, o robô apresentou dificuldades em se locomover ou arrancar em baixas velocidades.

2. Os sensores de proximidade frontais empregados no robô móvel apresentam uma sensibilidade cujo limite de detecção de objetos e paredes se mostrou próximo de 5 cm, inferior aos 20 cm nominais do fabricante. Isso, somado ao posicionamento dos mesmos, ocasionou algumas colisões nas laterais do robô móvel. Essas colisões ocorreram em função do ângulo de aproximação do robô móvel em relação às paredes, principalmente nas ocasiões em que as medidas realizadas pelo sonar não foram suficientes para detectá-las. Vale ressaltar que, apesar do sonar utilizar-se do ultra som, e do sensor de proximidade basear-se na emissão de luz infravermelha, tanto a propagação direcional do som como da luz se sujeitam a fenômenos físicos ópticos similares em relação aos ângulos de emissão e reflexão, devido à natureza especular das superfícies.

3. Quanto ao tempo de resposta do sistema, que engloba o processamento remoto e comunicação feita por meio sem fio, isto não comprometeu o objetivo da

aplicação. Apesar do atraso médio de 20ms nas respostas aos estímulos ambientais, ocasionados pelo sistema de comunicação Xbee, o robô móvel Kihon foi capaz de se locomover através do ambiente conforme planejado, evitando colisões na maior parte dos casos, apesar das situações descritas.

4. Por fim, em relação à facilidade de uso, o robô móvel se mostrou capaz de prover conectividade remota de forma descomplicada. A interface de comunicação usada para estabelecer a conexão cliente-servidor apresenta terminais simples e objetivos, com funções estabelecidas como TX, RX e GND. Isso permite aos usuários voltarem seus esforços aos tópicos de interesse, não demandando tempo em extensas documentações apenas para estabelecer uma comunicação.

O intuito deste experimento foi verificar a capacidade da plataforma em oferecer suporte a uma atividade envolvendo a navegação do robô móvel. O Kihon foi capaz de responder aos estímulos do ambiente em tempo real, quando controlado de forma remota, apesar das dificuldades que ocorreram em vista dos sensores de proximidade utilizados e também da limitada velocidade de sua interface de comunicação. É válido mencionar que essa disposição de conexão remota estabelecida para este experimento apresenta diversas vantagens ao usuário, como o fato de poder desenvolver aplicações em um ambiente familiar, como um computador pessoal (*notebook*), e ainda, sem a necessidade de manusear o robô móvel a cada nova reprogramação de seus algoritmos de controle, já que o sistema de controle é executado no próprio *notebook* do usuário. Por outro lado, a reprogramação de um sistema de controle embarcado no robô móvel exige maiores cuidados, além de tempo e trabalho por parte dos usuários.

O robô mostrou ser apto a se locomover em seu ambiente de aplicação apesar do seu sistema de tracionamento se limitar a velocidades acima de 50% dos ciclos PWM. Quanto ao problema envolvendo colisões laterais, uma alternativa para contornar tal situação, além do uso de controle diferente de velocidade de deslocamento ou de outro algoritmo de controle do robô, pode ser a substituição destes sensores digitais por sensores de proximidade analógicos, como é o caso do GP2D12 da Sharp. Apesar de mais caro em relação aos sensores empregados, o GP2D12 é capaz de medir distâncias de 10cm a 80 cm, de acordo com o

fabricante, também pelo processo de reflexão de feixe de luz. Além disso, suas saídas poderiam ser conectadas nas entradas ADC do Controlador, permitindo aquisições dos dados analógicos destes sensores, e não digital (binária), como foi realizado neste experimento. Uma outra alternativa ainda em termos de *hardware*, poderia ser a adoção de sensores analógicos ou ainda, empregando uma quantidade maior de sensores de proximidade. Essa possibilidade em acoplar e experimentar diversos tipos de dispositivos periféricos pode ser vista como um fator positivo, uma vez que estimula os usuários a desenvolverem e implementarem suas próprias soluções. Tal possibilidade somada à facilidade de uso obtida através da conectividade remota, mostram-se interessantes, principalmente em ambientes de desenvolvimento, na qual é capaz de permitir e motivar os usuários da plataforma a explorarem na prática, outras atividades envolvendo assuntos relacionados à locomoção e navegação de robôs móveis utilizando o Kihon.

Já em relação à API do controlador utilizado, este foi capaz de oferecer acesso aos módulos de *hardware* implementados no Kihon de forma descomplicada. Essa facilidade apresenta-se como um fator positivo na utilização da plataforma concebida.

4.2 Experimento 2

O segundo experimento teve como objetivo averiguar a capacidade da plataforma em oferecer suporte à uma aplicação científica. Nesse contexto, este experimento foi baseado em um trabalho realizado por Voltan et. al. (2011), onde os autores coletaram e estudaram a umidade relativa do ar dentro de uma estufa agrícola, esta que teve sua área mapeada e dividida em 114 pontos de medição. De modo a se obter sua variabilidade espacial, cada um desses pontos tiveram suas medições coletadas manualmente em três diferentes alturas em relação ao solo, sendo estas, 30cm, 120cm e 200cm em três horários ao longo do dia. Nesse contexto, foi atribuído ao robô móvel, a missão de executar uma tarefa em um cenário similar, seguindo um circuito determinado com linhas pretas no chão e coletando amostras de temperatura em três diferentes alturas, e em determinados pontos de seu plano de navegação. A partir dos dados levantados, é possível, por exemplo, fazer um levantamento

espacial da variação de temperatura de uma estufa ou um ambiente qualquer, em diferentes condições como horários, estações, climatizações, etc. Tanto a objetividade como a análise dos dados coletados podem ser vistas em Voltan et. al. (2011), já que fogem do escopo proposto neste trabalho e, portanto, não serão aprofundados ou discutidos aqui.

Adicionalmente, este experimento foi elaborado de forma a avaliar a flexibilidade da plataforma de robô móvel em termos de tecnologias de sistemas de controle e sua usabilidade. Para tal, empregou-se um RLRduino para realizar o Sistema de Controle. Produzido pela RLRobotics, o RLRduino é uma versão brasileira e equivalente ao Arduino Uno R3. Adicionalmente, o sistema de controle foi embarcado no robô móvel, fixando-o sobre o suporte do robô móvel Kihon. E por meio de seu porto serial, o RLRduino foi conectado diretamente à interface de comunicação do robô móvel Kihon. A disposição dos dispositivos utilizados neste experimento no Kihon são mostrados na Figura 4.6.

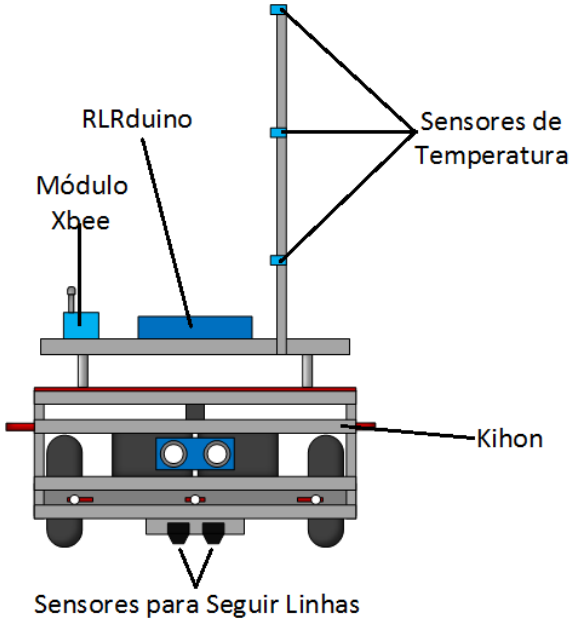


Figura 4.6 – Disposição dos dispositivos utilizados no Experimento 2.

Fonte: Elaborado pelo autor.

Para a realização deste experimento, três sensores de temperatura foram adicionados ao robô móvel, de modo que o permita coletar amostras de temperatura do ambiente em três alturas diferentes a partir do solo. Para isso, uma coluna de alumínio foi fixada sobre o

suporte da plataforma do Kihon e cada sensor foi fixado na coluna de modo que sua posição em relação ao solo corresponda às alturas de 50cm, 100cm e 150cm, respectivamente. O sensor de temperatura usado foi o LM35DZ, um modelo calibrado em graus Celsius que apresenta resolução de 10mV/°C e erro típico de ±0,9°C em temperaturas superiores a 25°C. Os sensores de temperatura tiveram as saídas de seus circuitos conectadas às entradas ADC do RLRduino, sendo estas A0, A1 e A2. As entradas analógicas do RLRduino possuem resolução de 10 bits que, no intervalo de 0V a 5V, assumem resolução (degrau) de 4,88mV/degrau.

Apesar do Controlador basear-se no microcontrolador PIC18F4550, que nesta utilização, dispõe de 8 canais de conversão AD de 10 bits de resolução e dos quais 5 ainda disponíveis para uso, escolheu-se por utilizar os conversores do RLRduino. A leitura dos canais AD do RLRduino levam cerca de 100 μs, o que permite efetuar até 10 mil medições por segundo. Já a leitura dos canais AD do Controlador, que tem como base o PIC18F4550, e que devido aos processos de chaveamento e pelas configurações dos seus registradores do módulo AD, o ciclo completo de conversão AD do microcontrolador demanda 20μs, permitindo efetuar até 50 mil medições por segundo. A decisão por utilizar o ADC do RLRduino se deu uma vez que, apesar da plataforma de robô móvel ainda ter disponíveis recursos vagos de entrada e saída, pode haver a necessidade de empregar outros tipos de dispositivos não implementados no Kihon, como por exemplo, uma câmera de vídeo, ou um sistema GPS. Portanto, apesar da plataforma dispor de recursos similares neste caso, ou seja, de conversores ADC, decidiu-se por utilizar recursos do próprio RLRduino e externos ao Kihon.

Como parte da tarefa da aplicação, determinou-se que o robô deveria seguir um circuito determinado por uma linha sobre seu plano de navegação. O circuito possui demarcações que determinam os locais onde devem ser coletadas medições de temperaturas do ambiente. A linha foi demarcada com fita adesiva preta de forma contínua, colada sobre a superfície de navegação, sendo esta plana e de cor clara, de modo a se obter suficiente contraste. Foi definido que tal traçado não possui curvas acentuadas e nem bifurcações, no entanto, a cada trecho, o percurso possui demarcações que determinam os locais onde devem ser coletadas

medições de temperatura do ambiente. Para seguir o traçado, foram utilizados os dois sensores de seguir linhas implementados no robô móvel Kihon. Como descrito no item 3.4.2.5, a saída destes dois sensores são conectados às entradas dos canais ADC do Controlador. A Figura 4.7 ilustra o circuito elaborado para o experimento.

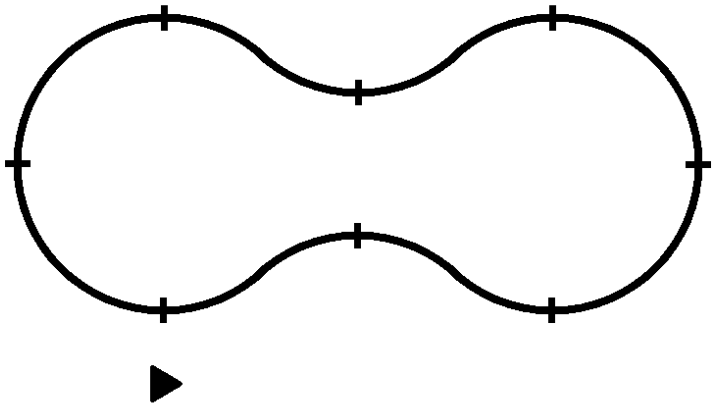


Figura 4.7 – Vista superior do Circuito elaborado para ser percorrido pelo Kihon.
Fonte: Elaborado pelo autor.

A seta indica o sentido e a posição inicial do circuito. Em cada trecho demarcado do trajeto, o robô realiza uma parada para coletar 10 amostras de temperatura de cada um dos três sensores, calcular suas respectivas médias e enviar os resultados obtidos a uma base de coleta de dados remota; então o robô móvel prossegue pela linha até o próximo ponto demarcado, sucessivamente, até concluir o trajeto e retornar ao ponto inicial. O envio dos resultados foi necessário, uma vez que não havia possibilidade de armazenamento de dados em um arquivo diretamente no RLRduino, já que a manipulação de arquivos no RLRduino requer o uso do *shield SD card*, indisponível no LISDI. Essa transmissão dos dados foi realizada por meio de um dispositivo Xbee, que foi conectado a um segundo porto serial emulado via *software* pelo RLRduino. Para a recepção dos dados foi utilizado outro módulo Xbee conectado a um *notebook*, onde um *software* ficou responsável por armazenar os dados recebidos em um arquivo de registro (*log*), este que também registra a data e a hora do início de cada volta. A Figura 4.8 ilustra a arquitetura cliente-servidor, desta vez embarcada, adotada para o Experimento 2.

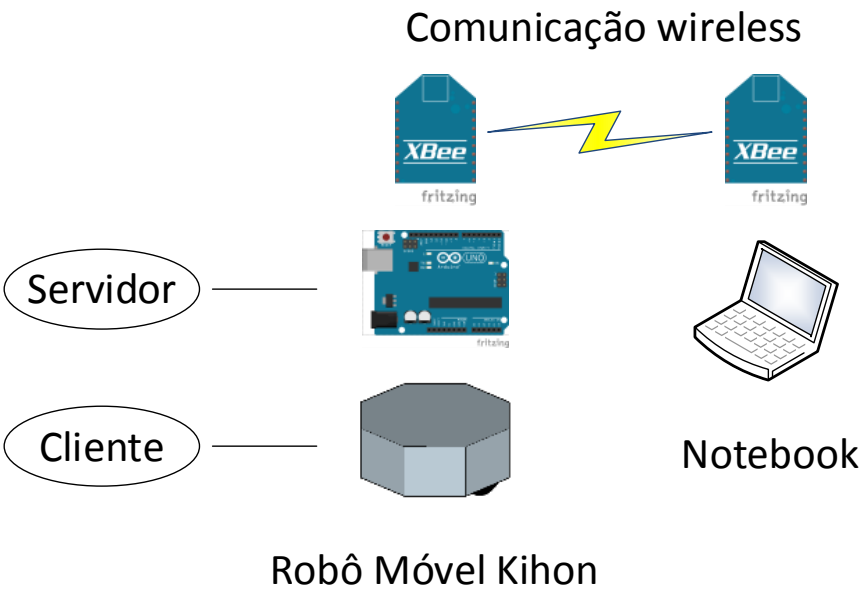


Figura 4.8 – Diagrama da arquitetura cliente-servidor do Experimento 2.

Fonte: Elaborado pelo autor.

As Figuras 4.9 (a) ilustra o ambiente onde foi realizado o Experimento 2 e a Figura 4.9 (b) ilustra o robô móvel Kihon executando o Experimento 2.

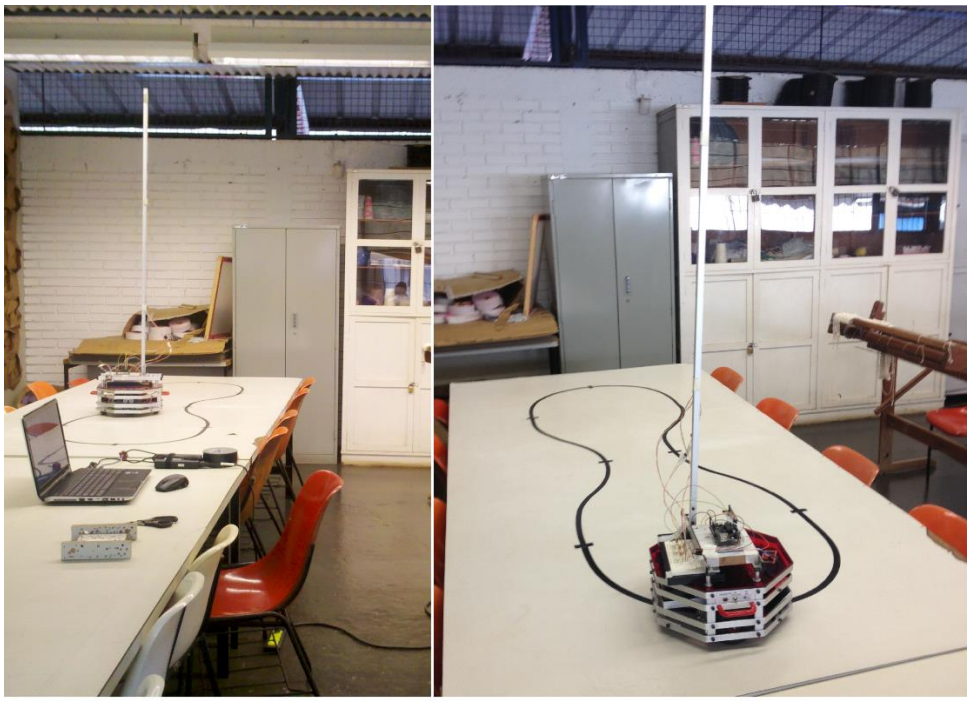


Figura 4.9 – (a) Vista geral do ambiente onde foi realizado o Experimento 2; e (b) Robô móvel Kihon executando o Experimento 2.

Fonte: Elaborado pelo autor.

Em relação ao algoritmo, utilizou-se a programação em linguagem baseada em C/C++ adaptada para o ambiente Arduino, já que o RLRduino utilizado é 100% compatível à IDE disponibilizada para o Arduino. A escolha por essa linguagem C++ se deu em vista da familiaridade em relação à ela, e também porque a API do Controlador já havia sido previamente concebida em C++, exigindo pequenas alterações. Apesar de ser possível implementar outra arquitetura de controle, pela própria facilidade de implementação, este experimento enquadrou-se novamente no paradigma comportamental, conforme Brooks (1986).

De modo similar ao modo como foi realizado no Experimento 1, por meio de chamadas às rotinas contidas na Classe Controlador da API (descrito em 3.4.3) adaptadas ao Arduino, o algoritmo de navegação foi capaz de ter acesso aos módulos de *hardware* do Kihon. Algumas das chamadas utilizadas neste experimento são exemplificadas na Tabela 4.2.

Tabela 4.2 – Exemplos de chamadas às rotinas da Classe Controlador API utilizadas no Experimento 2.

Exemplo de Requisição	Chamada à API	Retorno
Leitura dos Sensores de Solo	controlador->leADC()	Leitura de todos canais AD
Config. Movimentação Kihon p/ parar	controlador->pwmControl(bloqueio)	-----
Config. Ciclo PWM: 75% Esq e 75% Dir	Controlador->pwmCiclo(0.75, 0.75)	-----

Os dados obtidos no Experimento 2 são apresentados na Tabela 4.3.

Tabela 4.3 – Valores de temperatura (em °C) obtidos no Experimento 2.

	Volta 1 - hora: 16:30			**Volta 2 - hora 16:55**			**Volta 3 - hora 17:30**		
	Sensor 0.5m	Sensor 1.0m	Sensor 1.5m	Sensor 0.5m	Sensor 1.0m	Sensor 1.5m	Sensor 0.5m	Sensor 1.0m	Sensor 1.5m
Medição pto 1	31.04	31.28	30.79	30.79	30.79	30.30	30.30	30.40	29.91
Medição pto 2	30.99	31.28	30.79	30.74	30.74	30.30	30.30	30.50	30.16
Medição pto 3	30.79	31.28	30.79	30.40	30.74	30.30	30.30	30.55	30.21
Medição pto 4	30.89	31.28	30.79	30.35	30.74	30.30	30.30	30.45	30.25
Medição pto 5	30.79	31.28	30.79	30.30	30.74	30.30	30.30	30.30	30.25
Medição pto 6	30.79	31.28	30.79	30.30	30.74	30.30	30.30	30.30	30.21
Medição pto 7	30.79	31.28	30.79	30.30	30.65	30.30	30.30	30.30	30.06
Medição pto 8	30.79	31.28	30.79	30.30	30.74	30.30	30.30	30.30	30.11

4.2.1 Considerações

Os resultados obtidos por meio deste experimento mostraram que o robô móvel Kihon foi capaz de realizar o experimento satisfazendo os requisitos levantados pela tarefa estipulada, observando-se os seguintes itens:

1. Quanto à capacidade em oferecer suporte aos experimentos voltados à pesquisa, o robô móvel foi capaz de fazer uso adequado de seus sensores de solo ao seguir corretamente o traçado estipulado e, em conjunto com sensores acoplados

diretamente à plataforma de controle (realizada pelo RLRduino), a aplicação foi capaz de realizar as medições de temperatura nos determinados pontos do plano de navegação. Essa capacidade da plataforma proposta de atuar como uma base de robô móvel o torna apto a suportar experimentos envolvendo a robótica móvel, uma vez oferece funcionalidades essenciais de navegação, locomoção e ainda, sua arquitetura de *hardware* aberta o possibilita operar em conjunto com dispositivos externos, que neste caso foram a base de coleta de dados remota, e três sensores de temperatura conetados diretamente ao Sistema de Controle, mas que em outro cenário, poderia ter sido uma câmera de vídeo ou um módulo GPS.

2. A flexibilidade oferecida em termos de conectividade permitiu que uma tecnologia de Sistema de Controle fosse embarcada no robô móvel de forma descomplicada e rápida, por meio de seu conector de interface serial. Por apresentar uma interface de conexão serial genérica e um protocolo enxuto, a plataforma de robô móvel se apresenta como uma alternativa interessante, e ainda, sem comprometer a viabilidade técnica e financeira. Esse aspecto é interessante na robótica móvel e afeta diretamente na usabilidade da plataforma, uma vez que tal liberdade aliada à simplicidade pode oferecer aos pesquisadores condições de escolher diversos aspectos do desenvolvimento de suas aplicações, como empregar o sistema de controle de sua escolha, a própria linguagem de programação, a arquitetura de controle desejada, e seu ambiente de desenvolvimento (IDE).

Com o sistema de controle embarcado, o robô móvel foi capaz de executar adequadamente a tarefa determinada pelo experimento. Além disso, o envio e recepção dos dados coletados a uma base de dados remota pôde ser estabelecida sem dificuldades com o Sistema de Controle embarcado, já que o uso da tecnologia Xbee de comunicação sem fio empregada já era familiar. A conexão remota entre o Arduino e o *Notebook* foi estabelecida por meio de um porto serial, replicado utilizando a biblioteca SoftwareSerial do Arduino. Já a conexão do Arduino, como Sistema de Controle, ao robô móvel, como base da plataforma

de robô móvel, foi estabelecida diretamente através do porto serial USART nativo do Arduino. Ainda que o porto serial tenha uma taxa de comunicação limitada para os padrões modernos de comunicação, a interface de comunicação apresentou desempenho suficiente para a transmissão dos dados deste experimento, já que apenas 3 pares de bytes foram transmitidos a cada ponto de medição do circuito.

4.3 Conclusões do capítulo

Da forma como a plataforma foi concebida, ela ainda apresenta limitações tanto em termos de flexibilidade como em seu tempo de resposta. Caracterizando um sistema fortemente acoplado, seu Controlador foi concebido tendo como base um único microcontrolador, responsável tanto pelo gerenciamento de todos os recursos (sensores e atuadores), como pela comunicação com o Sistema de controle. Esse gerenciamento se torna custoso ao microcontrolador, na medida que lhe é atribuída a tarefa de efetuar o processamento das mensagens e ainda ter de se preocupar com todas as atividades dos periféricos, como interrupções externas (contador), geração dos ciclos PWM, geração dos tempos dos servomotores, conversão ADC, etc. Essa abordagem traz consigo certas limitações, ainda que, a princípio, reduza o custo e simplifique o entendimento, e que foram parte dos objetivos deste trabalho.

Apesar de oferecer suporte às iniciativas essenciais envolvendo a robótica móvel, a plataforma ainda tem muito a melhorar, como por exemplo, com a exploração mais granularizada oferecida por uma arquitetura fracamente acoplada. Uma plataforma na qual possua, por exemplo, um dispositivo microcontrolador dedicado à coordenação dos recursos de *hardware* e microcontroladores dedicados à cada dispositivo periférico, podendo oferecer ainda, maior liberdade principalmente em relação à flexibilidade de *hardware*. Essa flexibilidade diz respeito também às possibilidades de adicionar ou remover *hardware* de periféricos, podendo apresentar características de operabilidade PnP (do termo em inglês: *plug and play*, conectar e usar). Essa arquitetura fracamente acoplada poderia ser explorada em trabalhos futuros.

Um dos itens que se exigiu especial atenção na concepção do robô móvel Kihon foi a embarcação de dispositivos padrões de sensores e atuadores, de forma que o tornasse operacional, permitindo-o experimentar diferentes conceitos práticos relacionados à robótica móvel. Essa tarefa compreendeu tanto a escolha como a disposição dos itens de *hardware* para compor o robô móvel, sem perder de vista o foco no reduzido custo dos periféricos escolhidos, como pode ser verificada em 3.4.2.

No entanto, devido às restrições de tempo e orçamento, o *hardware* implementado não foi explorado totalmente. O Controlador do robô móvel, apesar de suas limitações em virtude da arquitetura adotada, suporta uma quantidade maior de dispositivos periféricos do que foi implementado neste trabalho. Portanto, em trabalhos futuros e ainda explorando a arquitetura fortemente acoplada implementada neste trabalho, outros dispositivos sensores e atuadores poderão ser implementados e embarcados no robô móvel para aumentar sua funcionalidade, de modo similar à forma como foi realizado no Experimento 2. Existem inúmeras possibilidades para tais dispositivos periféricos, tais como sensores de choque, sensores de luz, servomotores para compor um braço mecânico, dentre outros.

5 CONSIDERAÇÕES FINAIS

Com os recentes avanços tecnológicos e a redução dos custos de produtos tecnológicos, a robótica móvel tem se mostrado uma área em crescimento e com isso, tem despertado interesse cada vez maior em diversas aplicações, como domésticas, agrígolas e industriais. Em muitas dessas aplicações, os robôs são desenvolvidos de modo a realizarem tarefas tediosas e até mesmo perigosas para os seres humanos. Exemplos comerciais ou científicos desses esforços não faltam, como robôs que aspiram o chão da casa, veículos inteligentes como os desenvolvidos para o desafio DARPA e o *Google Self-Driving Car Project*, robôs de exploração de solo marciano com o *Mars Exploration Rover* da NASA, dentre outros. No entanto, a robótica móvel ainda apresenta diversos desafios para a comunidade científica. Um dos desafios é que, apesar dos diversos trabalhos propostos na literatura, ainda não há uma padronização de técnicas, arquiteturas de controle ou *software* e nem mesmo uma linguagem de programação estabelecida para a robótica móvel. Essa ausência de um consenso é justificável, uma vez que cada aplicação possui necessidades específicas que podem ser alcançadas de uma maneira, e outras aplicações que exigem outra abordagem. Neste contexto se inseriu a proposta deste trabalho, que é uma arquitetura de controle e *software* aberta e flexível, destinada ao desenvolvimento de iniciativas envolvendo a robótica móvel. Em virtude de sua destinação, aspectos como simplicidade e baixo custo foram consideradas, de modo a tornar seu uso mais acessível e motivador. Por apresentar uma solução de baixo custo, seu *hardware* implementado apresenta poucos componentes, como sensores e atuadores, os quais podem ser encontrados sem dificuldade no mercado. Logo, uma replicação da plataforma apresentada pode facilmente ser efetuada até mesmo por estudantes. E, ainda, com a facilidade oferecida pelo ICSP, é possível fazer a reprogramação do Controlador utilizado sem a necessidade de desmontar o robô móvel e remover o microcontrolador do circuito.

A plataforma proposta foi averiguada através da concepção do robô móvel Kihon. O Kihon apresenta, dentro das limitações já discutidas em 3.4 e 3.5, flexibilidade para implementação de um conjunto de dispositivos periféricos adicionais, propriedade verificada

no Experimento 2 (ver 4.2). E de modo a suportar tais iniciativas, a implementação do Kihon buscou oferecer condições essenciais de operação, como navegabilidade, sem perder de vista a facilidade de se efetuar manutenção e usabilidade, características estas que foram averiguadas nos dois experimentos realizados (capítulo 4). Apesar do Kihon ter apresentado algumas dificuldades no experimentos realizados, alguns se deram em virtude das limitações apresentadas, devido às restrições de orçamento, com o emprego de sensores de baixo custo e em pouca quantidade. Sendo assim, em termos de *hardware*, ainda é possível por exemplo, experimentar o uso de outros componentes tanto sensores como atuadores, ou ainda, adicionar módulos de *hardware*, uma vez que a quantidade de entradas e saídas do controlador não foi totalmente explorada neste trabalho.

A interface de comunicação adotada mostrou-se suficiente para realizar a comunicação de dados entre a plataforma e o *software* de controle sem comprometer sua reatividade. Apesar das evidentes limitações trazidas com a escolha de um padrão legatório, o resultado satisfatório se deu em virtude do simples protocolo de comunicação e também com a abstração das requisições e dos dados. Além disso, atribuindo à API do Controlador a tarefa de efetuar cálculos e conversões permitiu aliviar a carga de processamento sobre o microcontrolador, otimizando seu tempo de resposta aos comandos (requisições) de controle. A interface de comunicação adicional oferecida por meio de um segundo conector físico mostrou ser uma alternativa interessante para conexão de dispositivos que não utilizam o padrão de conector DB9, além de oferecer opções para alimentação de dispositivos de baixa potência.

A API do Controlador foi capaz de realizar a interface do *software* de controle com a plataforma de forma adequada, obtendo resultados dentro do esperado, averiguados através dos resultados obtidos com os Experimentos 1 e 2 (ver 4.1 e 4.2). Essa API pode ser reaproveitada para desenvolver novas aplicações, implementar novas funcionalidades à plataforma, assim como ser adaptada para outras linguagens de programação de modo a suportar outras plataformas e tecnologias para o controle da plataforma.

Ao deixar em aberto os paradigmas de controle e de *software*, a plataforma oferece maior liberdade, permitindo a exploração de diferentes paradigmas no desenvolvimento de

suas aplicações. Situação esta que seria dificultada com a definição e adoção de uma determinada arquitetura de controle ou de *software*, já que acabaria restringindo as possibilidades de desenvolvimento, manutenção e reutilização do *software*, além de acabar limitando a plataforma a algumas tecnologias e ambientes de desenvolvimento.

Através dos resultados obtidos, é possível concluir que a plataforma concebida neste trabalho é capaz de se apresentar como uma alternativa viável para facilitar o desenvolvimento de aplicações envolvendo a robótica móvel. Com a adoção de conceitos simples e ainda com o emprego de um número reduzido de componentes, é possível obter uma plataforma que apresente, dentro de suas limitações, um certo grau de flexibilidade e ainda oferecer recursos, de modo a torná-la funcional para execução de aplicações envolvendo locomoção e navegação. Conforme já considerado em 3.5 e 4.3, este trabalho ainda pode ser melhorado significativamente, com a adoção de outras tecnologias para a comunicação de dados, ampliação e otimização do protocolo de comunicação, de modo que a plataforma suporte uma variedade maior de dispositivos de *hardware*, implementação de sua API em outras linguagens de programação e ainda, com uma abordagem que explore a moduralização dos dispositivos de *hardware* da plataforma, ainda isto signifique aumento da complexidade e custo.

O presente trabalho contribui com a implementação da API do Controlador de Robô Móvel, suas bibliotecas básicas de controle, com protocolo básico de comunicação, e com o robô móvel Kihon concebido, de modo a averiguar suas arquiteturas abertas de controle e de *software* simples e de baixo custo, em conformidade com a proposta apresentada. Este trabalho, junto ao robô móvel Kihon, apresenta-se como uma base para trabalhos futuros sob a perspectiva computacional para o desenvolvimento de aplicações e experimentos envolvendo não apenas a robótica móvel, mas também de tópicos relacionados.

REFERÊNCIAS

ALVES, S.; ROSÁRIO, J. M.; FILHO, H. F.; NUNES, I.; **TAO** - a software plataform for autonomous mobile robots. IEEE International Conference on Control, Automation and systems (ICC), out 2012.

ANDO, N.; SUEHIRO, T.; KITAGAKI, K.; KOTOKU, T.; YOON W. **RT-Middleware**: distributed component middleware for RT (Robot Technology), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), ago 2005.

BARBER K. S.; MARTIN C. E. **Specification, measurement, and adjustment of agent autonomy**: theory and implementation**.** Autonomous Agents and Multi-Agent Systems, may 1999.

BATLLE, J. A.; BARJAU, A. **Holonomy in mobile robots**. Robotics and Autonomous Systems, v. 57, n. 4, p. 433-440, 30, abr 2009.

BEKEY, G. A. **Autonomous robots**: from biological inspiration to implementation and control. [S.l.]: MIT Press, 2005.

BETKE, M.; GURVITS, L. **Mobile robot localization using landmarks**. abr 1995.

BLANK D.; KUMAR D.; MEEDEN L.; YANCO H. **Pyro**: an integrated environment for robotics education. Proceedings of the National Conference on Artifical Intelligence, Vol. 20, No. 4, 2005.

BLANK D.; KUMAR D.; MEEDEN L.; YANCO H. **Pyro**: a Python-based versatile programming environment for teaching robotics. Journal on Educational Resources in Computing, Vol. 3, No. 4, dez 2003.

BONARINI, A.; MATTEUCCI, M.; MIGLIAVACCA, M.; RIZZI, D. **R2P**: an open source hardware and software modular approach to robot prototyping. Robotics and Autonomous Systems, 2013.

BORENSTEIN, J.; EVERETT, H. R.; FENG, L. **Where am I?** Sensors and techniques for mobile robot positioning. University of Michigan, abr 1996.

BROOKS, R. A. **A robusted layered control system for a mobile robot**. IEEE Journal of Robotics and Automation, Vol. RA2, No. 1, mar 1986.

BROOKS, R. A. **Intelligence without reason**. Proceedings of 12th International Joint Conference on Artificial Intelligence, p. 569-595, ago 1991.

CHATILA, R.; CAMARGO, R. F. **Open architecture design and inter-task/inter-module communication for an autonomous mobile robot**. IEEE International Workshop on Intelligent Robots and Systems - IROS, 1990.

DUDEK, G.; JENKIN, M. **Computational principles of mobile robotics**. [S.l.]: Cambridge University Press, 2010.

ELFES, A. **A distributed control architecture for an autonomous mobile robot**. Artificial Intelligence in Engireering, Vol. 1, No. 2, p. 99-108, 1986.

ELKADY, A.; SOBH, T. **Robotics middleware**: a comprehensive literature survey and attribute-based bibliography. Journal of Robotics, 2012.

FAYEK, R. E.; LISCANO, R.; KARAM, G. M. **A system architecture for a mobile robot based on activities and a blackboard control unit**. Proceedings of 1993 IEEE International Convference on Robotics and Automation, p. 267-274, 1993.

FUJITA, M.; KAGEYAMA, K. **An open architecture for robot entertainment**. D21 Lboratory, Sony Corporation, 1997.

GERKEY B.; VAUGHAN R.; HOWARD A. **The Player/Stage project**: tools for multi-robot and distributed sensor systems. Proceedings of the 11th International Conference on Advanced Robotics, jun 2003.

HABIB, M. K. **Bioinspiration and robotics**: walking and climbing robots**.** I-Tech Education and Publishing, p. 227-317 set 2007.

KALE, S.; SHRIRAMWAR, S. S. **FPGA-base controller for a mobile robot**. International Journal of Computer Science and Information Security, Vol. 3, No. 1, 2009.

KRAMER, J; SCHEUTZ, M. **Development environments for autonomous mobile robots**: a survey. Autonomous Robots 22.2, p. 101-132, 2007.

MARTINS, N. A. **Controle adaptativo e robusto de robôs móveis com rodas**. Florianópolis, Universidade Federal de Santa Catarina, 2010.

MATARIĆ, M. J. **Interaction and intelligent behavior**. Ph. D. Thesis - MIT, Mai 1994.

MATARIĆ, M. J. **The robotics primer**. [S.l.]: MIT Press, 2007.

MEDEIROS, A. A. D. **A survey of control architectures for autonomous mobile robots**. Journal of the Brazilian Computer Society, Vol. 4, No. 3, abr 1998.

MERTEN, M.; GROSS, H-M. **Highly adaptable hardware architecture for scientific and industrial mobile robots**. Proceedings of 2008 IEEE Conference on Robotics, Automation and Mechatronics (RAM 2008), p. 1130-1135, 2008.

MEYER-DELIUS, D.; BEINHOFER, M.; KLEINER, A.; BURGARD, W. **Using artificial landmarks to reduce the ambiguity in the environment of a mobile robot**. IEEE Conference on Robotics and Automation (ICRA 2011), p. 5173-5178, 2011.

MIGLIAVACCA, M.; BONARINI, A.; MATTEUCCI, M. **RTCAN**: A real-time CAN-bus protocol for robotic applications. Proceedings of 10th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2013), 2013.

MOHAMED, N.; AL-JAROODI, J.; JAWHAR, I. **Middleware for robotics**: a survey. IEEE 2008.

MOK, S. M.; WU, C.-HAUR. **Automation integration with UPnP modules**. Third IEEE International Workshop on Electronic Design, Test and Applications, 2006.

MURPHY, R. **Introduction to AI robotics**. [S.l.] MIT Press, 2000.

NAMOSHE, M.; TLALE, N. S.; KUMILE, C. M.; BRIGHT, G. **Open middleware for robotics**, 15th International Conference on Mechatronics and Machine Vision in Ptractice, dez 2008.

NEVES, M. C.; OLIVEIRA, E. **A control architecture for an autonomous mobile robot**. Proceedings of the First International Conference on Autonomous Agents, ACM, p. 193-200, fev 1997.

Object Management Group (OMG). **The common object request broker**: architecture and specification, Revision 2.0, 1995.

PESSIN, G.; OSÓRIO, F. S.; UEYAMA, J.; WOLF, D. F.; BRAUN, T. **Mobile robot indoor localization using artificial neural network and wireless networks**. First Brazilian Conference on Critical Embedded Systems (I CBSEC), p. 89-94, 2011.

QUIGLEY, M.; CONLEY, K.; GERKEY, B.; FAUST, J.; FOOTE, T.; LEIBS, J.; BERGER, E.; WHEELER, R.; NG, A. **ROS**: an open-source robot operating system. ICRA workshop on open source software, Vol. 3, No. 3.2, mai 2009.

RAGHAVAN, A.; ANANTHAPADMANBAN, H.; SIVAMURUGAN, M.; RAVINDRAN, B. **Accurate mobile robot localization in indoor environments using bluetooth**. IEEE International Conference on Robotics and automation (ICRA), p. 4391-4396, mai 2010.

RODRIGUES, M. L.; VIEIRA, L. F. M.; CAMPOS, M. F. M. **Mobile robot localization in indoor environments using multiple wireless technologies**. IEEE Brazilian Robotics Symposium and Latin American Robotics Symposium (SBR-LARS), 2012.

ROSS, J. **The book of wireless**: a painless guide to Wi-Fi and broadband wireless. No Starch Press, 2008.

SANTIAGO, G. S.; MEDEIROS, A. A. **A proposed hardware and software architecture for a robotic system**. development 1: 2, 2011.

SARANLI U.; Avci, A.; Öztürk, M.C. **A modular real-time fieldbus architecture for mobile robotic platforms**. IEEE Transactions on Instrumentation and Measurement, Vol. 60, mar 2011.

SIEGWART, R.; NOURBAKHSH, I. R. **Introduction to autonomous mobile robots**. MIT Press, 2004.

SPONG, MARK. W.; HUTCHINSON, S.; VIDYASAGAR, M. **Robot modeling and control**. Wiley, 2005.

THUEER, T. **Mobility evalutation of wheeled all-terrain robots**. Robotics and Autonomous Systems, 2009.

UTZ, H.; SABLATNÖG, S.; ENDERLE, S.; KRAETZSCHMAR, G. **MIRO** - middleware for mobile robot applications. IEEE Transactions on Robotics and Automation, Vol. 18, No. 4, 2002.

VOLPE, R.; NESNAS, I.; ESTLIN, T.; MUTZ, D.; PETRAS, R.; DAS, H. **The CLARAty architecture for robotic autonomy**. Proceedings of 2001 Conference in Aerospace Conference, Vol. 1, p1-121, 2001.

VOLTAN, D. S.; BARBOSA, R. Z.; MARTINS, J. E. M. P.; ZIMBACK, C. R. L. **Análise da distribuição espacial da temperatura do ar em uma casa de vegetação**. II Simpósio de Geoestatísitica Aplicada em Ciências Agrárias, mai 2011.

YAMAMOTO, Y.; YUN, X. **Coordinating locomotion and manipulation of a mobile manipulator**. Automatic Control, IEEE Transactions on, Vol. 39, No. 6, p. 1326-1332, 1994.

YOO, J.; KIM, S.; HONG, S. **The robot software communications architecture (RSCA)**: QoS-aware middleware for networked service robots. SICE-ICASE Internation Joint Conference, 2006.

APÊNDICE A – MODELO CINEMÁTICO DE UM ROBÔ COM TRAÇÃO DIFERENCIAL

O modelo cinemático desenvolvido em coordenadas retangulares descreve as características geométricas do robô, e é obtido a partir de estudos relacionados ao deslocamento realizado por seus atuadores. A Figura 1 ilustra o modelo matemático do sistema de tração diferencial, conforme Martins (2010).

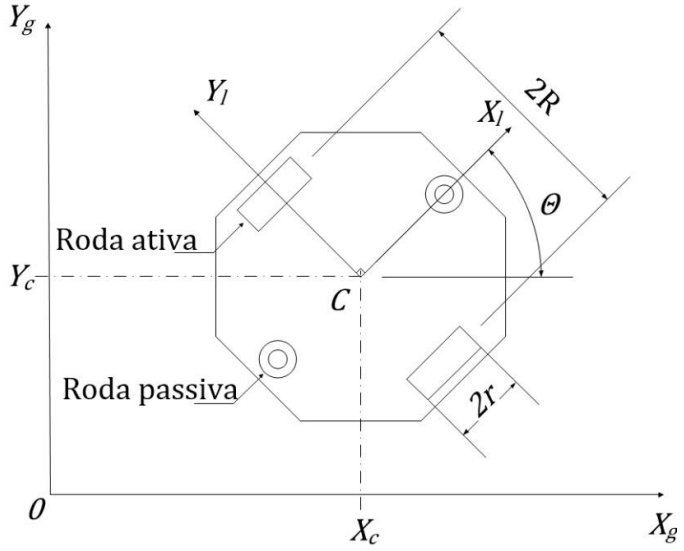


Figura 1 - Modelo matemático de robôs móveis que adotam tração diferencial.

Fonte: Adaptado de Martins (2010).

De acordo com a notação utilizada para ilustrar o modelo matemático da Figura 1, OX_gY_grepresenta o sistema de coordenadas global ou inercial $\mathcal{F}_g$ do robô, e CX_lY_l representa o sistema de coordenadas local $\mathcal{F}_l$do robô; C é o centro de massa do robô; r é o raio das rodas ativas; R é a distância entre as rodas e o eixo de simetria; $\dot{\varphi}_i$ e v_i correspondem, respectivamente, às velocidades angular e linear para a roda i; v é a velocidade linear do robô e ω é a velocidades angular do robô, tendo $\mathcal{V} = (v, \omega)^{\mathrm{T}}$; e q é o vetor de coordenadas generalizadas onde $q = [q_1 q_2 \dots q_n]^{\mathrm{T}} \in \Re^n$. A posição do robô é dada por (x_c, y_c, θ), na qual x_c e y_c são as coordenadas de C em $\mathcal{F}_g$ e θ é o ângulo de $\mathcal{F}_l$ em relação a $\mathcal{F}_l$, onde $(x_c, y_c) = (x_g, y_g)$.

Em termos cinemáticos, o modelo e suas restrições são discutidos por Siegwart e Nourbakhsh (2004). A tração diferencial apresenta três restrições cinemáticas (YAMAMOTO; YUN, 1994), onde a primeira restringe sua movimentação apenas na direção normal ao eixo das rodas ativas, como mostra a equação (3), e as outras duas restrições dizem respeito à restrição de rolamento puro das rodas esquerda e direita, respectivamente, conforme as equações (4) e (5).

$$\dot{y}_c \cos\theta - \dot{x}_c \operatorname{sen}\theta = 0 \tag{3}$$

$$\dot{x}_c \cos\theta + \dot{y}_c \operatorname{sen}\theta + R\dot{\theta} - r\dot{\varphi}_d = 0 \tag{4}$$

$$\dot{x}_c \cos\theta + \dot{y}_c \operatorname{sen}\theta + R\dot{\theta} - r\dot{\varphi}_e = 0 \tag{5}$$

Uma vez que $v = \dot{x}_c \cos\theta + \dot{y}_c \operatorname{sen}\theta$ e que $\dot{\theta} = \omega$, é possível obter as velocidades linear v e angular ω do centro de massa C do robô a partir das equações (4) e (5), podendo assim, ser descrito como:

$$\begin{bmatrix} \dot{\varphi}_d \\ \dot{\varphi}_e \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & \frac{R}{r} \\ \frac{1}{r} & -\frac{R}{r} \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{6}$$

Uma outra forma de expressar o modelo cinemático é através das velocidades nos eixos do plano e de rotação, ou seja, $\dot{x}_c$, $\dot{y}_c$ e $\dot{\theta}$, resultando em:

$$\begin{aligned} \dot{x}_c &= v\cos\theta \\ \dot{y}_c &= v\operatorname{sen}\theta \\ \dot{\theta} &= \omega \end{aligned} \tag{7}$$

Ainda é possível expressar em termos de coordenadas generalizadas q, como mostra a equação (8), ou seja:

$$\dot{q} = S(q)v(t) \quad , \tag{8}$$

onde

$$S(q) = \begin{bmatrix} \cos(\theta) & 0 \\ \text{sen}(\theta) & 0 \\ 0 & 1 \end{bmatrix} \tag{9}$$

De acordo com as especificações do motor CC com caixa de redução empregado, 127 rotações equivalem a um giro completo da roda que está conectada após a caixa de redução. Considerando apenas a rampa de subida da onda quadrada gerada pelo sensor de odometria, são necessários 127 pulsos para completar uma rotação completa da roda. Com isso, é possível determinar o quanto o eixo do motor girou para um dado sentido, conforme mostra a equação (10). Considerando o diâmetro da roda empregada, de 8,89cm, cada transição corresponde a:

$$P = 2\pi r, onde\ r = 4{,}445\ cm \tag{10}$$

$$\varphi = \frac{360°}{d} \quad , \tag{11}$$

onde $d = 127$ pulsos, logo

$$\varphi = 2{,}8346° \tag{12}$$

Proporcionalmente, é possível determinar o quanto cada roda deslocou o robô. Como o deslocamento *d* da roda para uma volta completa é dada por $d = 2\pi r$, o deslocamento *d* é de 27,928 cm a cada giro completo da roda.

E o erro máximo que cada roda apresentará a 2,8346° será:

$$P = \frac{2{,}8346° * 27{,}928}{360°} = 0{,}2199\ cm \tag{13}$$

A velocidade nominal de operação da saída do motor CC utilizado, após sua caixa de redução, é de 80rpm. Através da Equação (14) obtém-se sua velocidade angular de 1,33 rad/s. Com isso, é possível determinar a velocidade linear máxima v_{max} de deslocamento do robô, de 34,91cm/s, obtida por meio da equação (15).

$$\omega = \frac{75rpm}{60s} = 1{,}33\ rad/s \tag{14}$$

$$v_{max} = \frac{1{,}33 * 27{,}928cm}{1s} = 37{,}144\ cm/s \tag{15}$$

Esta v_{max} de 37,144cm/s corresponde à velocidade plena do robô móvel. Nela são desconsideradas as perdas e variações, como as atribuídas à carga ou resistência mecânica sobre o motor CC, ou o robô locomovendo-se em superfícies irregulares ou inclinadas, assim como outros fatores como patinação e deformações da roda.

APÊNDICE B – CONTROLADOR DE ROBÔS MÓVEIS

A plataforma de robô móvel proposta opera sobre uma arquitetura cuja função é atuar como um agente de interface entre o sistema de controle e a camada de *hardware* do robô móvel. Essa interface é estabelecida através de uma conexão serial, cuja programação é facilitada por uma API escrita em linguagem C++.

A princípio, o *hardware* da plataforma foi desenvolvido e implementado de modo a oferecer recursos, como atuadores e sensores, para prover suporte às funcionalidades essenciais relacionadas à mobilidade e navegação, tais como sistema de tracionamento e sensores de proximidade, sensores de solo, odometria e sonar. Essa configuração inicial é suficiente para prover condições da plataforma em ensaiar experimentos fundamentais de robótica, e ainda, apresentar flexibilidade de modo a permitir alterações de *hardware*. Para tornar tais recursos funcionais, uma API implementada em C++ é responsável tanto pela comunicação de dados, como pela abstração dos comandos, e tanto de leitura dos sensores como de atuação. O tipo de interface de comunicação escolhido foi o tipo serial, pois o padrão de comunicação serial RS-232, apesar de suas limitações, pode facilmente ser implementado ou adaptado em vista da sua simplicidade.

O microcontrolador

Como mencionado em 3.4.2, para o desenvolvimento do *hardware* do controlador, optou-se por empregar o microcontrolador PIC18F4550 da Microchip®. A opção por este componente em particular se deu devido a alguns fatores, como seu enquadramento ao escopo do projeto, baixo custo, facilidade de aquisição no mercado brasileiro e, ainda, por oferecer diversos dispositivos de entrada e saída, dentre os quais se destacam:

- Um conversor analógico-digital (ADC) de 13 canais e 10 bits de resolução;

- Até trinta e seis linhas de entrada e saída digitais reconfiguráveis;

- Quatro temporizadores, um de 8 bits e três de 16 bits;

- Suporte aos barramentos USB, I2C, Master Synchronous Serial Port (MSSP) e SPI;

- Dois módulos de Modulação por Largura de Pulso (PWM).

Além das características mencionadas, outros fatores influenciaram de forma decisiva na escolha deste microcontrolador. Em primeiro lugar, a familiaridade com o desenvolvimento do *software* embarcado com microcontroladores da família PIC da Microchip®. Em segundo lugar, a disposição imediata destes componentes para o uso no laboratório onde o trabalho de desenvolvimento foi realizado – LISDI (Laboratório de Integração de Sistemas e Dispositivos Inteligentes). Por último, o laboratório dispunha de materiais e dispositivos necessários para a gravação dos firmwares e manuseio destes microcontroladores.

Desenvolvimento do circuito

A partir dos recursos oferecidos pelo microcontrolador empregado, foram selecionados os periféricos necessários para que o controlador oferecesse suporte aos diferentes tipos de sensores e atuadores, e ainda possibilitar a comunicação com o sistema de controle. Assim, o circuito desenvolvido oferece suporte aos seguintes periféricos:

- Oito entradas para conversão analógico-digital com resolução de 10 bits (ADC0-ADC7);

- Oito canais de dados digitais reconfiguráveis (DADOS0-DADOS7);

- Quatro canais de saída de dados digitais (CONTROLE0-CONTROLE3);

- Dois módulos de PWMs (PWM0-PWM1);

- Quatro módulos para servomotores (SERVO0-SERVO3);

- Três contadores digitais de 16 bits (CONTADOR0-CONTADOR2);

A geração dos pulsos de clock para o PIC 18F4550 é feita a partir de um cristal de 20MHz, que o transforma em 48MHz por meio de um circuito PLL (Phase-Locked Loop). Desta forma, o microcontrolador é capaz de operar a 12MIPs.

De modo a permitir a gravação do firmware do microcontrolador, foi implementada a funcionalidade ICSP no circuito. O ICSP (acrônimo de *In Circuit Serial Programming*) permite que operações de leitura e gravação do microcontrolador sejam realizadas sem a necessidade de desmontar o robô móvel Kihon, e sem remover o microcontrolador da placa de circuito. Desta forma, uma interface ICSP foi disposta na lateral do robô móvel Kihon, permitindo fácil acesso às operações de leitura e gravação do PIC utilizado. Para tal, basta conectar qualquer dispositivo gravador, programador e depurador, como o Pickit™ 2 ou o Pickit™ 3 da Microchip®. O diagrama do circuito elétrico é ilustrado na Figura 1 .

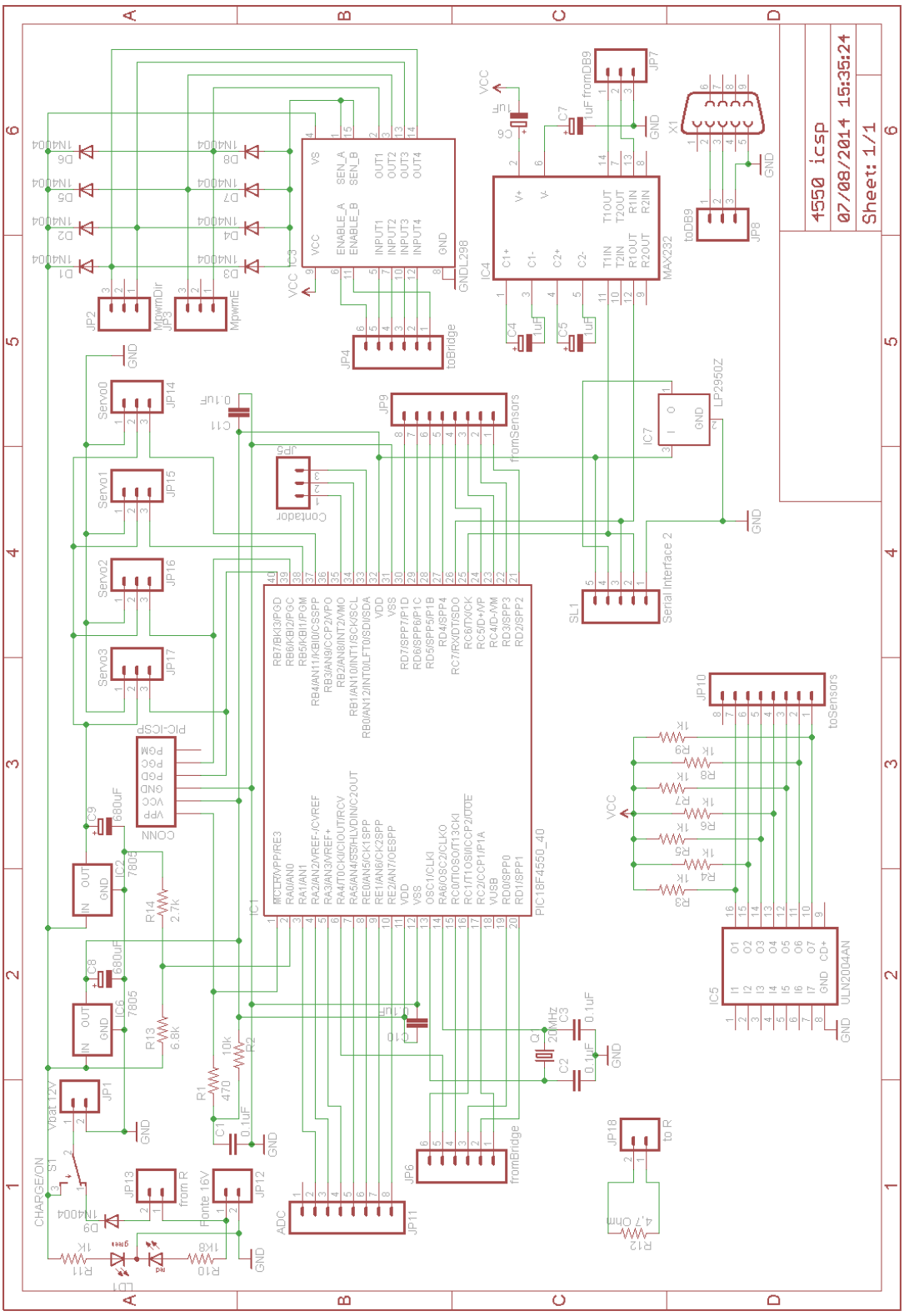


Figura 1 - Diagrama do circuito elétrico do Controlador da plataforma de robô móvel.

Fonte: Elaborado pelo autor.

Firmware do controlador

Com o circuito definido, foi dado início ao desenvolvimento do firmware. A linguagem de programação escolhida foi o C, uma vez que a maior parte dos compiladores C possuem bibliotecas consolidadas que permitem a utilização do porto serial. Optou-se pela utilização do compilador C18 da Microchip®, já empregado pelo laboratório LISDI no desenvolvimento de outros trabalhos e atividades.

Por meio da implementação do recurso ICSP (acrônimo do termo em inglês: *In Circuit Serial Programming*), permitiu-se a possibilidade de gravação sem a necessidade de desmonte do Kihon e remoção do microcontrolador da placa de circuito. O ICSP permite que o *firmware* do Controlador possa ser lido, escrito ou reescrito de forma fácil e descomplicada, uma vez que se disponha do dispositivo de gravação que suporte a família de microcontroladores PIC18 da Microchip®, na qual está incluso o PIC18F4550. A Figura 2 ilustra a interface de conexão ao ICSP do microcontrolador PIC no Kihon.

Figura 2 – Conector ICSP no Kihon.
Fonte: Elaborado pelo autor.

APÊNDICE C – PROTOCOLO DE COMUNICAÇÃO

Esta seção apresenta os detalhes da comunicação de dados, ou seja, como é feita a estruturação dos dados na arquitetura cliente-servidor concebida, representados respectivamente pelo Controlador e pelo Sistema de Controle. A visão geral do protocolo de comunicação é apresentado em 3.4.3. A seguir serão detalhados cada um dos elementos.

Leituras das entradas ADC

Ao receber uma requisição de leitura dos conversores analógico-digital, o PIC efetua a aquisição dos 8 primeiros canais (AN0-AN7) e os serializa para retornar ao Sistema de controle. Ainda que fosse possível solicitar a leitura de apenas um canal, optou se apenas por realizar a leitura de todos os canais utilizados por uma questão de otimização de tempo. Desconsiderando a taxa de comunicação, uma série de requisições enviadas pelo Sistema de Controle, de modo a obter as leituras de cada canal analógico-digital individualmente, demandaria 5 bytes para concluir a comunicação, dos quais seriam 3 bytes para a requisição e 2 bytes para a resposta ao Sistema de Controle, que multiplicada pelos 8 canais, totalizaria 40 bytes. Por outro lado, utilizando uma única requisição para obter a leitura de todos os 8 canais analógico-digitais, seriam necessários apenas 19 bytes, dos quais seriam 3 bytes da requisição e 16 bytes de resposta ao Sistema de Controle.

O comando de requisição de leitura utiliza apenas o byte identificador, que contém o valor 0 ASCII. A resposta enviada pelo Controlador tem 16 bytes de comprimento, dos quais cada dupla de bytes corresponde à leitura de um dos canais. O comando de requisição e a resposta são mostrados na Tabela 1 e Tabela 2 abaixo.

Tabela 1 – Organização dos bytes da requisição de leitura dos canais ADC.

Byte	Descrição
0	Código ASCII 0

Tabela 2 – Organização dos bytes da resposta da leitura dos canais ADC.

Byte	Descrição
0	Canal AN0: byte mais significativo
1	Canal AN0: byte menos significativo
2	Canal AN1: byte mais significativo
3	Canal AN1: byte menos significativo
4	Canal AN2: byte mais significativo
5	Canal AN2: byte menos significativo
6	Canal AN3: byte mais significativo
7	Canal AN3: byte menos significativo
8	Canal AN4: byte mais significativo
9	Canal AN4: byte menos significativo
10	Canal AN5: byte mais significativo
11	Canal AN5: byte menos significativo
12	Canal AN6: byte mais significativo
13	Canal AN6: byte menos significativo
14	Canal AN7: byte mais significativo
15	Canal AN7: byte menos significativo

Foi implementado em *hardware* um circuito para medir o nível de tensão da bateria obtidos pela conexão serial de 2 baterias de 6V empregadas para alimentar toda a plataforma de robô móvel. O circuito divisor de tensão foi obtido por meio de uma associação simples de resistores, pode ser observado no diagrama elétrico do Controlador (Figura 1 do Apêndice B), cuja saída é conectada à entrada do canal AN0 do conversor analógico-digital do PIC. Assim, cada vez que uma requisição de leitura dos canais ADC é recebida pelo Controlador, os bytes de resposta 0 e 1 correspondentes às leituras do Canal 0, sempre retornarão o nível de tensão da alimentação da plataforma de robô móvel.

Leitura dos contadores

Ao receber uma requisição para leitura dos contadores, o Controlador serializa o valor dos três contadores e, após o envio, reinicia todos os contadores. Assim como descrito anteriormente, a leitura unificada dos contadores foi preferida em relação à leitura isolada para obter melhor proveito da comunicação serial.

O comando de requisição de leitura utiliza apenas o byte identificador, que contém o valor 1 ASCII. A resposta enviada pelo Controlador possui 6 bytes de comprimento, onde cada dupla de bytes corresponde ao valor de cada um dos contadores. O comando de requisição e a resposta são mostrados, respectivamente, na Tabela 3 e Tabela 4.

Tabela 3 – Organização dos bytes da requisição de leitura dos contador.

Byte	Descrição
0	Código ASCII 1

Tabela 4 – Organização dos bytes da requisição de leitura dos canais ADC.

Byte	Descrição
0	Contador 0: byte mais significativo
1	Contador 0: byte menos significativo
2	Contador 1: byte mais significativo
3	Contador 1: byte menos significativo

Medição do Sonar

Ao receber uma requisição de medição do Sonar, o Controlador retorna como resposta 2 bytes contendo a medição do Sonar. O comando de requisição e resposta é mostrado na Tabela 5 e Tabela 6.

Tabela 5 – Organização dos bytes da requisição de leitura dos contador.

Byte	Descrição
0	Código ASCII 2

Tabela 6 – Organização dos bytes da requisição de leitura dos canais ADC.

Byte	Descrição
0	Contador 2: byte mais significativo
1	Contador 2: byte menos significativo

O cálculo necessário de conversão do valor numérico obtido para metros é apresentado na Equação 1.

$$D = \frac{1}{48Mhz} * 4 * 8 * TMR1 * 340\, m/s * \frac{1}{2} \quad , \tag{1}$$

onde:

D corresponde à distância em metros;

$TMR1$ corresponde à quantidade de interrupções contabilizadas.

A utilização do Sonar está associada ao Servo 0, uma vez que o direcionamento do Sonar é obtido através do posicionamento do eixo do Servo motor 0.

Configuração da frequência PWM

A frequência de operação dos PWMs do PIC18F4550 é gerada pelo temporizador Timer2. O Timer2 pode ser modificada alterando-se os valores dos registradores PR2 e TCON. O registrador PR2 armazena o período de contagem do Timer2, podendo variar no intervalo

de 0 a 255. Já o registrador TCON armazena o valor de prescaler do Timer2. O prescaler é uma forma de reduzir a frequência de clock dos temporizadores do PIC, possibilitando que os temporizadores sejam capazes de aguardar intervalos de tempo mais longos.

O cálculo da frequência é apresentado na Equação 2.

$$F = \frac{1}{\big((PR2) + 1\big) * 4 * T_{osc} * (TMR2\ Prescale)} \quad , \tag{2}$$

onde

F corresponde à frequência de operação do PWM

$T_{osc} = (48 * 10^6 Hz)^{-1}; 0 \leq PR2 \leq 255, PR2 \in \mathbb{Z}; TMR2\ Prescale \in \{1,4,16\}$

Considerando que a frequência de clock do PIC18F4550 utilizado é 48MHz (resultando em um período aproximado de T_{osc} =20,83ns), as frequências mínimas e máximas dos módulos de PWM são de, respectivamente 2929,6875MHz e 12MHz. No entanto, quanto maior a frequência, menor será a resolução do ciclo de trabalho dos PWMs, de modo que a frequência de 12MHz, ainda que possível, não apresentará resultados satisfatórios.

O comando de configuração da frequência dos PWMs passa ao Controlador como parâmetros o valor do período do temporizador Timer2, e o valor de prescale. A organização dos bytes é mostrada na Tabela 7. O controlador não envia nenhum byte de resposta para este comando.

Tabela 7 – Organização dos bytes da requisição de configuração dos PWMs.

Byte	Descrição	
0	Código ASCII 3	
1	Byte que será copiado para registrador PR2	
2	Bits 7-2	Não usados
	Bits 1-0	TCON bits: T2CKPS1 e T2CKPS0

Configuração dos ciclos PWM

Os ciclos de trabalhos PWM0 e PWM1 podem ser alterados através da escrita nos registradores CCPR1L e CCP1CON para o PWM1, e CPPR2L e CCP2CON para o PWM2. A resolução em bits dos PWMs varia de acordo com a frequência e é dada pela Equação 3. Já o cálculo do ciclo de trabalho dos PWMs é dado pela Equação 4.

$$Resol_{PWM_{(max)}} = \frac{\log(\frac{F_{osc}}{F_{PWM}})}{\log(2)} bits \quad (3)$$

$$Ciclo\ de\ trabalho\ _{PWM} = (CCPRxL{:}\ CCPxCON < 5{:}\ 4 >) * T_{osc} * (TMR2\ prescale) \quad (4)$$

O comando de configuração do ciclo de trabalho do PWM envia ao firmware uma única requisição, passando como parâmetro 3 bytes, dispostos de modo a conter 2 sequências de 10 bits – uma sequência para o PWM0 e uma sequência para o PWM1. A organização dos bytes é mostrada na Tabela 8.

Tabela 8 – Organização dos bytes da requisição de configuração dos ciclos PWMs.

Byte		Descrição
0		Código ASCII 4
1	Bits 7-4	Não usados
	Bits 3-0	4 Bits mais significativos do PWM0
2	Bits 7-2	6 Bits menos significativos do PWM0
	Bits 1-0	2 Bits mais significativos do PWM1
3	Bits 7-0	8 Bits menos significativos do PWM1

Configuração de controle PWM

O controle de estado dos PWM pode ser configurado através de uma requisição de escrita nas portas digitais que foram destinadas a essa função. O controle PWM nada mais é que uma requisição de escrita direta sobre o estado de saída lógica de 4 portas digitais. Quando relacionada ao PWM, esta requisição tem como objetivo determinar o sentido de rotação dos PWM0 e PWM1.

O comando de configuração do controle PWM permite que o PWM0 e PWM0 possuam individualmente 4 estados: bloqueio, parado, para frente e para trás. Assim, o único parâmetro passado a essa requisição é um byte, dentre os quais foram destinados os quatro bits menos significativos para definir tais estados, dos quais os bits 3 e 2, correspondem aos estados do PWM0, e os bits 1 e 0 correspondem aos estados do PWM1. A organização dos bytes é mostrada na Tabela 9. O controlador não envia nenhum byte de resposta para este comando.

Tabela 9 – Organização dos bytes da requisição de configuração de Controle PWM.

Byte		Descrição
0		Código ASCII 5
1	Bits 7-4	Não usados
	Bits 3-2	Estado do PWM0
	Bits 1-0	Estado do PWM1

Configuração do tempo dos servomotores

O PIC18F4550 não possui nenhum módulo de *hardware* que suporte o controle de servomotores. Para que o Controlador fosse capaz de oferecer tal suporte, foi necessário escrever rotinas próprias de geração de sinais de controle. Essas rotinas permitem gerar

pulsos no intervalo de 920us a 2120us com resolução de 333,33 ns. Para isso, as rotinas fazem uso do temporizador Timer0, de 16 bits de resolução do PIC.

O comando de configuração do servomotor indica qual servo será alterado e qual sua nova constante de tempo (com 16 bits). Ao recebê-lo, o Controlador atualiza as constantes de tempo para o servomotor especificado. A descrição deste comando é mostrada na Tabela 10. O controlador não envia nenhum byte de resposta para este comando.

Tabela 10 – Organização dos bytes da requisição dos servo motores.

Byte	Descrição
0	Código ASCII 6 para o Servo motor 0
	Código ASCII 7 para o Servo motor 1
	Código ASCII 8 para o Servo motor 2
	Código ASCII 9 para o Servo motor 3
1	Byte mais significativo da nova constante de tempo
2	Byte menos significativo da nova constante de tempo

Configuração I/O digital

O comando de configuração das portas de entrada e saída digitais permite que se altere a direção do das portas digitais, ou seja, que se alterne a função das portas entre saída e entrada. Ao todo, são 8 canais de dados digitais bidirecionais. O comando de requisição de configuração I/O digital envia ao firmware passando como parâmetro um byte no qual cada bit corresponde à configuração da respectiva porta digital sendo o bit com nível lógico 0 para saída e o bit com nível lógico 1 para entrada. A organização dos bytes deste comando é mostrada na Tabela 11. O controlador não envia nenhum byte de resposta para este comando.

Tabela 11 – Organização dos bytes da requisição de configuração I/O das Portas Digitais.

Byte		Descrição
0		Código ASCII 10
1	Bit 7	Direção da Porta Digital 7
	Bit 6	Direção da Porta Digital 6
	Bit 5	Direção da Porta Digital 5
	Bit 4	Direção da Porta Digital 4
	Bit 3	Direção da Porta Digital 3
	Bit 2	Direção da Porta Digital 2
	Bit 1	Direção da Porta Digital 1
	Bit 0	Direção da Porta Digital 0

Leitura I/O digital

As portas I/O digitais cujas configurações de direção foram definidas como entrada poderão ter seus níveis binários de estados lógicos (0 ou 1) lidos através do comando de requisição de leitura I/O Digital. Ainda que a direção de uma dada porta digital esteja configurada como saída, a requisição de leitura I/O Digital retornará o último valor lógico escrito na respectiva porta digital. Ao receber o comando de leitura I/O Digital, o Controlador serializa e envia ao Sistema de Controle o estado atual presente de todos os canais digitais em seu respectivo bit.

O comando de leitura I/O Digital e a respectiva resposta são mostradas na Tabela 12 e Tabela 13, respectivamente.

Tabela 12 – Organização dos bytes da requisição de Leitura I/O Digital.

Byte	Descrição
0	Código ASCII 11

Tabela 13 – Organização dos bytes da resposta da requisição de Leitura I/O Digital.

Byte		Descrição
0	Bit 7	Estado lógico da Porta Digital 7
	Bit 6	Estado lógico da Porta Digital 6
	Bit 5	Estado lógico da Porta Digital 5
	Bit 4	Estado lógico da Porta Digital 4
	Bit 3	Estado lógico da Porta Digital 3
	Bit 2	Estado lógico da Porta Digital 2
	Bit 1	Estado lógico da Porta Digital 1
	Bit 0	Estado lógico da Porta Digital 0

Escrita I/O digital

As portas I/O digitais cujas configurações de direção foram definidas como saída poderão ter seus níveis binários de estados lógicos (0 ou 1) alterados através do comando de requisição de escrita I/O Digital. O comando de escrita I/O Digital envia como parâmetro um byte contendo os novos estados lógicos binários, cada um em sua respectiva posição no byte. Ao receber uma requisição de escrita I/O Digital, o Controlador escreve na respectiva porta digital seu novo valor binário recebido. Se a direção de uma dada porta digital estiver configurada como entrada, o respectivo bit de escrita não terá efeito de escrita sobre o estado lógico da porta digital de entrada, mantendo inalterado o seu estado lógico.

O comando de escrita é mostrado na Tabela 14. O controlador não envia nenhum byte de resposta para este comando.

Tabela 14 – Organização dos bytes da requisição de Escrita I/O Digital.

Byte		Descrição
0		Código ASCII 12
1	Bit 7	Novo valor lógico da Porta Digital 7
	Bit 6	Novo valor lógico da Porta Digital 6
	Bit 5	Novo valor lógico da Porta Digital 5
	Bit 4	Novo valor lógico da Porta Digital 4
	Bit 3	Novo valor lógico da Porta Digital 3
	Bit 2	Novo valor lógico da Porta Digital 2
	Bit 1	Novo valor lógico da Porta Digital 1
	Bit 0	Novo valor lógico da Porta Digital 0