UFBA - UNIVERSIDADE FEDERAL DA BAHIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PPGEE - PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

MARCIO FIGUEIREDO GARCIA

DISSERTAÇÃO DE MESTRADO

AXEBOT II - UMA PLATAFORMA MODULAR PARA SISTEMAS MULTI-ROBÔS

Salvador-Ba

Setembro de 2020.

MARCIO FIGUEIREDO GARCIA

AXEBOT II - UMA PLATAFORMA MODULAR PARA SISTEMAS MULTI-ROBÔS

DISSERTAÇÃO DE MESTRADO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal da Bahia, como requisito parcial para obtenção do título de Mestre em Engenharia Elétrica.

Orientadores:
Prof. Augusto Cesar Pinto Loureiro da Costa, D.Sc - UFBA
Prof. Tiago Trindade Ribeiro, D.Sc - UFBA
Prof. Jés de Jesus Fiais Cerqueira, D.Sc - UFBA

Salvador-Ba

Setembro de 2020.

Ficha catalográfica elaborada pela Biblioteca Universitária Bernadeth Sinay Neves da Escola Politécnica (SIBI/UFBA) com dados fornecidos pelo autor

G216 Garcia, Marcio Figueiredo.
Axebot II: Uma plataforma modular para sistemas multi-robôs / Marcio Figueiredo Garcia. – Salvador, 2020.
169 f. : il. color.

Orientador: Prof. Dr. Augusto Cesar Pinto Loureiro da Costa.
Coorientador: Prof. Dr. Tiago Trindade Ribeiro.

Dissertação (mestrado) – Universidade Federal da Bahia. Escola Politécnica, 2020.

1. Robôs móveis. 2. Sistemas de controle inteligente. 3. Microcontroladores. 4. Inteligência artificial. I. Costa, Augusto Cesar Pinto Loureiro da. II. Ribeiro, Tiago Trindade. III. Universidade Federal da Bahia. IV. Título.

CDD.: 629.892

MARCIO FIGUEIREDO GARCIA

AXEBOT II - UMA PLATAFORMA MODULAR PARA SISTEMAS MULTI-ROBÔS

Dissertação de Mestrado aprovada em 22 de setembro de 2020 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Augusto Cesar Pinto Loureiro da Costa
DEE/UFBA

Prof. Dr. Tiago Trindade Ribeiro
DEE/UFBA

Prof. Dr. Jés de Jesus Fiais Cerqueira
DEE/UFBA

Prof. Dr. Vitor Leão Filardi
Examinador externo - IFBA

Dedico este trabalho a minha filha querida Laura Nunes Garcia
que sem saber, foi minha motivadora por esse acontecimento.

Agradecimentos

- Primeiramente agradeço a Deus, pela essência de todas as coisas e por completar o significado deste trabalho. Agradeço ao senhor por iluminar o meu caminho e tudo no seu tempo, pois a sua graça se faz presente em todos os momentos da minha vida;
- Ao Professor orientador o Dr. Augusto César Pinto Loureiro da Costa por ter me aceito como orientando, pela paciência e apoio;
- À CAPES pela concessão de uma bolsa de estudos durante a realização do curso;
- A minha família pelo apoio, compreensão, paciência e carinho durante essa jornada, em especial a minha esposa Vivian Nunes Garcia e a minha mãe Marinalva Figueiredo Garcia, na qual acreditou que esse sonho seria possível;
- Ao grande amigo Volker Kible, que tive o prazer de conhecer durante a realização desse trabalho pelas dicas;
- Aos Professores deste programa de Pós Graduação, a exemplo dos Professores Dr. Jés de Jesus Fiais Cerqueira e Dr. André Gustavo Scolari pelo aprendizado nas disciplinas essenciais ao andamento e conclusão desse trabalho;
- Aos comentários e as sugestões dos membros da banca examinadora desta dissertação: Professor Dr. Vitor Leão Filardi do IFBA e o Professor Dr. Tiago Ribeiro Trindade da UFBA;
- Ao amigo e Prof. Dr. Ivanoe Rodowanski da UFRB pelas dicas;
- Aos grandes amigos Wecio Santana Machado e Wellinghton Passos que tive a oportunidade de conhecer durante essa pós-graduação pelas dicas e apoio;
- Ao colega e Professor Diego Stéfano, MSc. pelo apoio;
- Também gostaria de agradecer aos colegas do LAB 54 (a Bat Caverna) desta Universidade pelas trocas de informações bastante úteis;
- Aos funcionários do Departamento de Engenharia Elétrica, especialmente a Jaime e Agatha que sempre me atenderam com carinho e dedicação;
- A todos que contribuíram para que este árduo trabalho chegasse ao fim. Agradeço de coração e deixo aqui o meu, *muito obrigado!*

“ Não importa quanto a vida possa ser ruim,
sempre existe algo que você pode fazer e triunfar.
Enquanto há vida, há esperança ”.
(Sthepen Hawking)

Resumo

Robôs autônomos são dotados de mecanismos para desempenhar determinada tarefa com mínima intervenção humana. Entretanto, o desempenho dessas tarefas não seria possível se não fosse incorporado aos robôs a capacidade de movimentação autônoma. Contudo, para que estudos em sistemas embarcados seja possível, é necessário um ambiente experimental confiável, com possibilidades de alterações e adaptações a nível de software e hardware adequados às pesquisas que abordam. As plataformas para desenvolvimento prático, apresentam-se como uma alternativa à experimentação com robôs reais, já que concedem a realização de experimentos em um hardware apropriado. Devido a isso, foi desenvolvido no presente trabalho, uma arquitetura robótica modular multiplataforma composta por uma PCB provida de um hardware reconfigurável e expansiva. Nessa arquitetura, à qual também é composta pelo robô omnidirecional, foi configurada para execução de um modelo de agente autônomo cognitivo embarcado, utilizando-se três plataformas conectadas a barramento CAN (*Controller Area Network*). A arquitetura do agente é composta por três níveis, a saber: o nível reativo (percepção-ação) executado pela plataforma PSoC® 5L consistindo de comportamentos criados sobre um controlador cinemático, o nível instintivo (coordenação) composto pela plataforma LTeK® IoT LPC546xx e o nível cognitivo (planejamento) composto pela plataforma RENESAS® GR-PEACH, que são executados concorrentemente. Além disso, a arquitetura do agente cognitivo foi configurada e programada para mesclar comportamentos reativos no nível instintivo usando lógica difusa onde foram realizados experimentos com o AAC embarcado em que o robô foi capaz de realizar uma trajetória sem falhas. Algumas melhorias futuras para este trabalho também são mencionadas

Palavras-chave: Robôs móveis, Sistemas de controle inteligente, Microcontroladores, Inteligência artificial.

Abstract

Autonomous robots are equipped with mechanisms to perform a certain task with minimal human intervention. However, the performance of these tasks would not be possible if the ability to autonomously move was not incorporated into the robots. However, for studies on embedded systems to be possible, it is necessary to have a reliable experimental environment, with the possibility of changes and adaptations in terms of software and hardware appropriate to the research they address. Platforms for practical development are presented as an alternative to experimentation with real robots, since they allow the realization of experiments on appropriate hardware. Because of this, a multiplatform modular robotic architecture composed of a PCB with reconfigurable and expansive hardware was developed in the present work. In this architecture, which is also composed by the omnidirectional robot, it was configured to execute an embedded cognitive autonomous agent model, using three platforms connected to the CAN bus (*Controller Area Network*). The agent architecture consists of three levels, namely: the reactive level (perception-action) executed by the PSoC® 5LP platform consisting of behaviors created on a kinematic controller, the instinctive level (coordination) composed by the platform LTeK® IoT LPC546xx and the cognitive level (planning) composed by the RENESAS® GR-PEACH platform, which are executed concurrently. In addition, the architecture of the cognitive agent was configured and programmed to merge reactive behaviors at the instinctual level using fuzzy logic where experiments were carried out with the embedded AAC in which the robot was able to perform a flawless trajectory. Some future improvements for this work are also mentioned.

Keywords: Mobile robots, Intelligent control systems, Microcontrollers, Artificial Intelligence.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

IA	Inteligência Artificial
ROS	*Robot Operating System*
AAC	Agente Autônomo Concorrente
FLC	*Fuzzy Logic Control*
API	*Application Programming Interface*
AVR	linha de microcontroladores de 8*bits* fabricada pela ATMEL
USB	*Universal Serial Bus*
UDB	*Universal Digital Blocks*
CAN	*Controller Area Network*
CPU	*Central Processing Unit*
UART	*Universal Asynchronous Receiver/Transmitter*
HPS	*Hard Processor System*
RTR	*Remote Transmit Request*
UDP	*User Datagram Protocol*
ACK	*Acknowledge*
IP	*Internet Protocol*
SBC	Sistema Baseado em Conhecimento
MAC	*Medium Access Control*
PSoC	*Programmable System-on-a-Chip*
RISC	*Reduced Instruction Set Computer*
CRC	*Cyclic Redundancy Check*
MOSFET	*Metal Oxide Semiconductor Field Effect Transistor*
DOF	*Degrees Off Freedom*

AGV	*Automated Guided Vehicle*
ARM	*Advanced RISC Machine*
IoT	*Internet of Things*
GPIO	*General Purpose Input/Output*
GPS	*Global Positioning System*
I2C	*Inter-Integrated Circuit*
SPI	*Serial Peripheral Interface*
PID	*Proporcional-Integral-Derivativo*
PWM	*Pulse Width Modulation*
IDE	*Integrated Development Environment*
TOMR	*Three-Wheeld Omnidirectional Mobile Robot*
CPR	Ciclos por Revolução
ECR	Eventos Contáveis por Revolução
RT	Relação de Transmissão
RPM	Rotação Por Minuto
CMOS	*Complementary Metal-Oxide-Semiconductor*
FPGA	*Field Programmable Gate Arrays*
FT	*Function Transfer*
eFLL	*Embedded Fuzzy Logic Library*
PPTC	*Polyer Temperature Positivo Coefficent*
I/O	*Input/Output*
RTOS	*Real Time Operating System*
KBS	*Knowledge Based System*
MCU	*Micro Controller Unit*
PCB	*Printed Circuit Board*
SMD	*Surface Mount Device*
CAD	*Computer Aided Design*

Sumário

1 Introdução

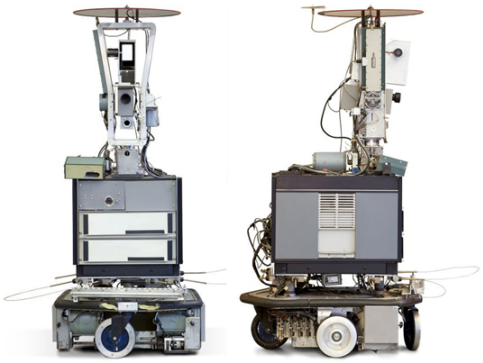
Com as grandes descobertas no século passado, como os materiais semicondutores, destacando a invenção do transistor em 1948 e do circuito integrado em 1958, a eletrônica colaborou excepcionalmente para o desenvolvimento da robótica. Devido a esses avanços tecnológicos, o desenvolvimento de áreas como a eletrônica e a mecânica, tornou-se real as possibilidades de colocar em prática sistemas capazes de operar de maneira que dispensam a supervisão humana, ou seja, de forma autônoma. Tais sistemas são denominados robôs.

Em meio a esses avanços, em 1954 o americano George Devol, criou o primeiro braço robótico industrial controlado por computador, no qual foi intitulado Máquina de Transferência Programada (do inglês, *Programmed Article Transfer*). Sua função principal era de transferir objetos de um local para outro. Devol ainda criou uma memória de computador e um sistema de controle chamado de *Universal Automation* e foi com o engenheiro Joseph Engelberger da Columbia University que em 1956 desenvolveram o primeiro robô industrial com a empresa *Unimation Inc.*, um projeto para substituir os trabalhadores da fábrica por máquinas robóticas.

Para alcançar tal autonomia, os robôs devem ser dotados de sistemas e mecanismos capazes de tomar suas próprias decisões, através de algum tipo de inteligência, ou seja, precisam interpretar informações sensoriais, aprendizado, interação com seres humanos, dentre outras ações. Devido a isso, vem surgindo sistemas capazes de suprir a essas necessidades através de dispositivos integrados a sistemas inteligentes, como por exemplo: planejamento, inferência, representação do conhecimento e até visão computacional.

Após mais alguns anos de estudos e desenvolvimentos no Instituto de Pesquisas de Stanford nos Estados Unidos, foi desenvolvido o robô móvel SHAKEY, mostrado na Figura 1.

Figura 1 – Robô Móvel SHAKEY.



Fonte: [1].

Este robô era dotado de mecanismos de raciocínio automático, permitindo assim, a tomada de decisão sobre quais ações deveriam ser executadas. Além disso, tinha um importante diferencial com relação a outros robôs da época, necessitavam receber instruções detalhadas dos passos que deveriam executar, enquanto que este, interpretava os comandos dados por seus operadores e armazenavam as informações obtidas do ambiente através de seus componentes eletrônicos embarcados, tais como: sensores de visão e contato, câmera e link de rádio. Sendo considerado como um dos primeiros projetos da área de Inteligência Artificial no qual combinavam raciocínio lógico e ações físicas, o robô SHAKEY possuía módulos de visão computacional e processamento de linguagem natural.

Em decorrência do surgimento desses protótipos, a robótica móvel já encarava alguns desafios como de tentativas de construir sistemas robóticos capazes de interagir autonomamente com seus ambientes, planejando um determinado caminho através de movimentos de rotação e translação sem colisões. Um desses desafios consistia no problema do planejamento de movimento. Com isso, houveram pesquisas intensas sobre o planejamento do movimento de robôs, sendo que a maior parte desses estudos se deu em planejar locomoção entre obstáculos estacionários, [2], [3]. Atualmente, o problema de planejar movimentos entre obstáculos em movimento atraiu intensos estudos, visto o interesse cada vez maior de construir sistemas autônomos utilizando-se comportamentos reativos através de sensores e atuadores.

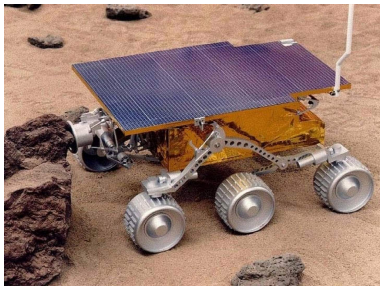
Mais tarde, [4] publicou alguns artigos mostrando uma arquitetura reativa, fundamentada em esquemas motores, em que cada comportamento influenciava no movimento do robô, por meio de um vetor força artificial, a qual tornou-se mais tarde uma arquitetura híbrida denominada de AuRA (*Autonomous Robot Architecture*).

Segundo [5] as arquiteturas reativas evitam o uso de um modelo do mundo. Um dos lemas da abordagem puramente reativa é o de que: *"o mundo é a melhor representação dele mesmo"*. Dessa forma, o sistema de controle do robô depende fortemente das informações locais obtidas por meio dos sensores, ou seja, de como o robô vê o mundo ao seu redor em determinado instante [6]. Uma arquitetura reativa define a maneira como a informação sensorial é mapeada em uma ação ou resposta, e também define como é realizada a coordenação dos diversos pares percepção-ação, ou estímulo-resposta [6].

A princípio, com o avanço das áreas de sensores, Inteligência Artificial e processamento de imagens, a robótica se deparou com a complexidade envolvida no desenvolvimento de sistemas móveis que fossem robustos e adaptáveis. Devido a esses grandes avanços, na década de 1990, o robô móvel SOJOURNER da NASA, ilustrado na Figura 2 toca o solo de Marte, que apesar de ter suas limitações de deslocamento, foi capaz de realizar um planejamento *off-line*, onde explorou a superfície marciana por cerca de três meses. Esse robô possuía seis rodas, era composto por câmeras frontal e trazeira, hardware para realizar várias experiências científicas e tecnológicas e transmitir imagens e dados de volta

a sonda *LANDER*.

Figura 2 – Projeto *Mars Sojourner Rover*.



Fonte: [7].

Já em 2004, outros dois robôs móveis, conhecidos como *rovers* SPIRIT e OPPORTUNITY pousam em Marte, desta vez com maior sucesso, conseguindo explorar maiores áreas devido a uma melhor autonomia de suas baterias.

Daí em diante, a robótica também mostra-se mais crescente e presente em serviços domésticos (aspiradores de pó e cortadores de grama), nas industrias, no qual podemos citar a implantação de robôs móveis tais como os veículos AGV (do inglês, *Automated Guided Vehicles*) para transportes automatizados de cargas, na área de defesa civil e militar (resgate e exploração em ambientes de alto risco) e de pesquisas científicas, como os robôs da NASA enviados a marte mais recentemente. A Figura 3 ilustra alguns desses tipos de robôs.

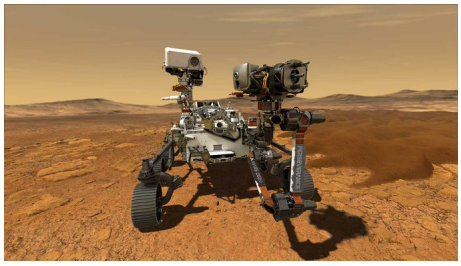
Figura 3 – Aplicações da robótica móvel.

(a) Robô Aspirador.

(b) Kiva Robot.

(c) Robô Telemax.

(d) Perseverance Rover Robot.



Fontes: a) [8]; b) [9]; c) [10]; d) [11].

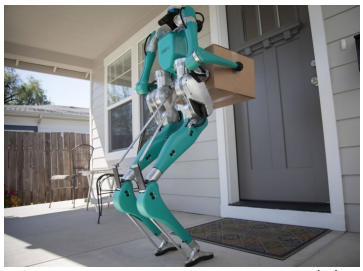
Esses robôs se apresentam numa grande variedade de formas, tamanhos e tipos de mobilidade que lhes permitem funcionar em seu ambiente. A robótica, enquanto linha de pesquisa, permite o desenvolvimento de sistemas robóticos para diversos tipos de ambiente, além disso integra diversos segmentos de pesquisa, dos quais pode se destacar comunicação, teoria de controle e sistemas multi-agentes.

Para executar uma tarefa, é necessário que o robô supere as dificuldades acerca do ambiente (obstáculos estáticos ou dinâmicos), interação com outros robôs ou seres humanos. Sendo assim, o sistema inteligente que governa a sua atuação deve ser capaz de superar situações divergentes que podem comprometer o cumprimento da tarefa. Segundo [12] essa é uma tarefa desafiadora e um dos problemas fundamentais da robótica autônoma.

Para superar essas dificuldades, houve o surgimento de diversas configurações para os robôs móveis. Podem se deslocar-se no solo através de rodas, esteiras, pernas, etc, outros podem deslocar-se no ar, como um helicóptero, balão, drone e alguns na água, como um robô submarino. Enfim, robôs móveis são definidos como sendo um veículo capaz de movimentação autônoma, equipado com atuadores controlados por um computador embarcado. Algumas dessas configurações podem ser observadas na Figura 4.

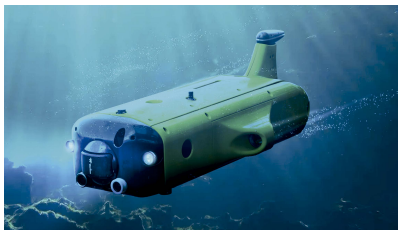
Figura 4 – Exemplos de robôs móveis.

(a) Meet Digit Robot.



(b) Drone.

(c) FlatFish Robot.

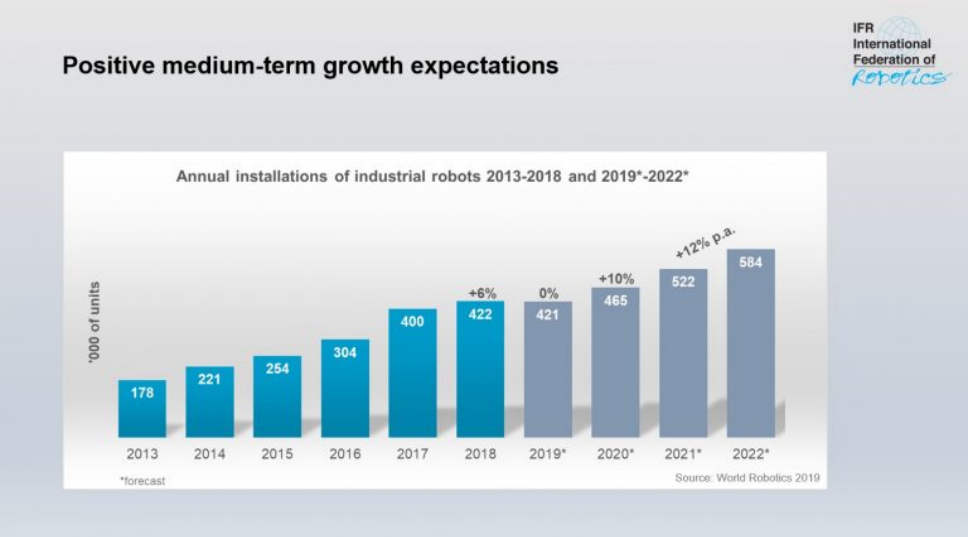


Fontes: a) [13]; b) [14]; c) [15].

Diante de recentes e avançadas pesquisas para aplicações em sistemas embarcados, robótica e inteligência artificial, vistas em conjunto, tem sido investigada largamente no meio acadêmico. De acordo com [16], a criação de novas necessidades e mercados demandam pesquisa e desenvolvimento em outras áreas da robótica, como a robótica móvel [17], controle não linear, geometria computacional, inteligência em ambientes não estruturados, dentre outros [18]. Em consequência disso, são vastas suas aplicações.

De acordo com o RBR (do inglês, *Robotics Business Review*), na categoria de robôs de serviço, informa que o valor de vendas de robôs para uso profissional aumentou de forma significativa. Os robôs utilizados em sistemas de logística, como os AGV's, representam a maioria de todas as unidades comercializadas. A Figura 5 ilustra as perspectivas de crescimento das instalações anuais de robôs industriais.

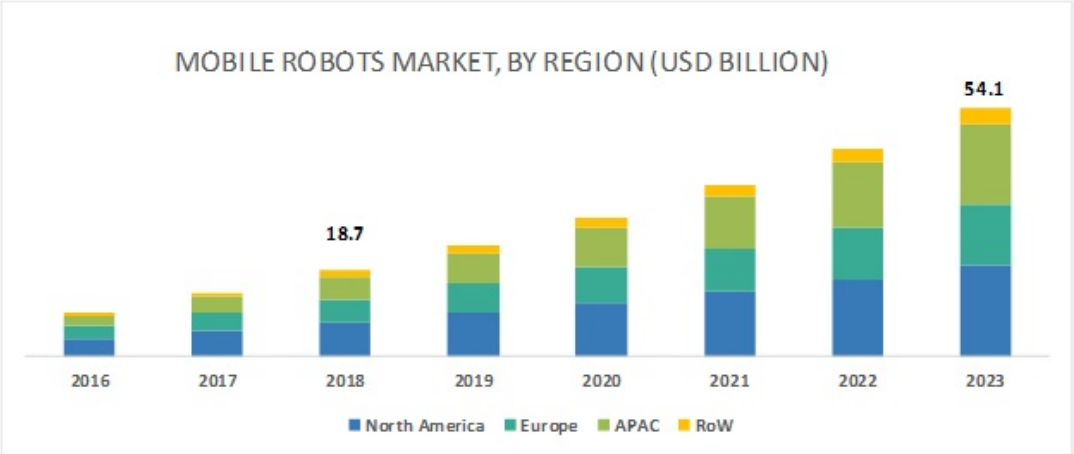
Figura 5 – Instalações anuais de robôs industriais.



Fonte: [19].

Além do interesse acadêmico, existe um significativo interesse comercial, visto que o mercado de robôs móveis tende a crescer a uma taxa bastante significativa, como ilustrado na Figura 6.

Figura 6 – Projeção do mercado de robôs móveis por região.



Fonte: [20].

Mais recentemente, devido aos intensos avanços da microeletrônica, destacando-se os microcontroladores e microprocessadores, percebeu-se um desenvolvimento crescente, evidenciado principalmente pela indústria de automóveis, na utilização dos sistemas robóticos. Além disso, a evolução desses sistemas permitiu que ocupem hoje espaços distintos dos idealizados inicialmente, e cada vez mais frequente a utilização de robôs em contexto hospitalar, doméstico, na agricultura, educação e monitoramento de áreas urbanas e rurais.

Sendo assim, a robótica móvel, por exemplo, tem atraído a atenção de pesquisadores e entusiastas, estendendo a acessibilidade dessas tecnologias para o alcance de boa parte da população, trazendo desafios e centenas de possibilidades aos desenvolvedores dessas incríveis máquinas. Além disso, na área acadêmica, os robôs vem se firmando como plataformas para desenvolvimento e implementação de técnicas que até então ficavam restritas às simulações de forma virtual. Códigos para sistemas multiagentes, técnicas de inteligência computacional, de controle e navegação, são alguns das áreas que essas plataformas de desenvolvimento são utilizadas. Algumas incluem procedimentos (deliberativos/reativos) como o trabalho desenvolvido em [6] para robótica móvel com interação humana, onde a arquitetura desenvolvida para tarefas de navegação, permite que o robô seja operado em diferentes níveis de autonomia.

1.1 Motivação

Sistemas embarcados estão presentes em diversas áreas e apresentam inúmeras aplicações, dentre elas, a robótica móvel. Contudo, apontar-se o elevado custo como um ponto a ser destacado para a aquisição de plataformas prontas para desenvolvimento dessas aplicações. Às limitações encontradas em plataformas para desenvolvimento de algoritmos em robótica móvel, torna-se inacessível em alguns casos, modificações de hardware para outras aplicações. Um hardware reconfigurável oferece equilíbrio entre eficiência e flexibilidade na hora de implementar um sistema. Além disso, os crescentes requerimentos de flexibilidade, como a necessidade de um dispositivo conseguir se adaptar a certas condições de operação, ou seja, precisam ser reconfiguráveis às aplicações de pesquisas que abordam.

Este trabalho faz parte do escopo do projeto “Mecateam”, com a Universidade Federal da Bahia, que visa o desenvolvimento e aprimoramento para uma nova estrutura física e eletrônica do robô móvel AxeBot [21]. O hardware desenvolvido nesse trabalho, além de ser configurado para embarcar um agente autônomo em uma rede de CAN, estará apto ao desenvolvimento de outras aplicações em robótica móvel, tanto para uma estrutura robótica omnidirecional, quanto para robô móvel diferencial e/ou manipuladores articulados, esses últimos podendo ser configurados em conjunto.

Por outro lado, é relevante pela falta de projetos similares disponíveis atualmente utilizando plataforma modular, na qual servirá de base para projetos de diversos propósitos nessa área. Ademais, este projeto se destaca por dotar de diversas opções e tipos de controle disponíveis através da fácil e documentada configuração física, provendo uma solução reconfigurável, prática e expansiva referente ao quesito de software e hardware.

1.2 Objetivo Geral

O Principal objetivo deste trabalho é desenvolver uma arquitetura robótica de característica modular, multiplataforma e aberta para estudos e pesquisas em sistemas multi-robôs: O *AxeBot II*. A Arquitetura proposta deve apresentar flexibilidade para utilização de diferentes plataformas de hardware, reconfigurável e expansível para robôs com rodas de diferentes tipos de locomoção.

1.2.1 Objetivos Específicos

Este trabalho possui alguns dos principais objetivos específicos aos quais destaca-se:

- Projeto e remodelagem para uma nova estrutura mecânica;
- Projeto e remodelagem para uma nova estrutura eletrônica e computacional;
- Modelagem de novos motores de baixo custo.

1.3 Justificativa

A robótica é uma área interdisciplinar que estuda o desenvolvimento de soluções para problemas em que a ação humana deve ser substituída ou otimizada para acelerar a operação de um processo físico de maneira segura. A natureza dessas soluções permite subdividir a robótica em subáreas não disjuntas: robótica de manipuladores, móvel, tele operada, autônoma, etc.

A robótica móvel é uma dessas subáreas em intenso desenvolvimento, abrangendo estudos e pesquisas de característica multidisciplinar. Conforme [22], "... é um campo novo" e, de acordo com [17], "... o avanço de sistemas que mostra mobilidade é um problema chave...". Além da robótica móvel autônoma ser o contexto deste trabalho, ela pode ser usada para o desenvolvimento de outras áreas da robótica, com projetos ainda mais complexos.

Para desenvolver aplicações em sistemas embarcados existem várias plataformas de desenvolvimento e módulos comerciais e com tendência acentuada de surgimento de

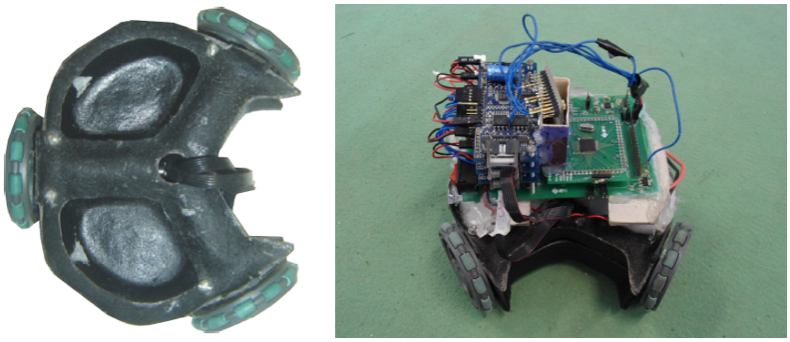
outros(as) que possibilitam o desenvolvimento de trabalhos e pesquisas em robótica na sua vasta área.

Em áreas como engenharia da computação, sistemas em tempo real, visão computacional, agentes autônomos e multi-agentes, são exemplos na qual a robótica está inserida. Estes sistemas podem ser vistos de forma didática, quando plataformas robóticas são empregadas na aprendizagem, como por exemplo, os *kits* de robótica educacional da LEGO® MINDSTORMS interativos, dotados de sensores e atuadores utilizado em ambiente acadêmico ou na indústria, onde eles realizam tarefas como pintura, soldagem, empilhamento e outros processos como transportar objetos de um lugar para outro.

Já nas escolas técnicas e de nível superior, outros *kits* mais avançados de robótica como, KOALA e KHEPERA da empresa suíça *K-Team Corporation* e o robô PIONEER 3-DX da empresa americana *Adept MobileRobots* auxiliam no desenvolvimento de pesquisas relacionadas a robótica móvel facilitando a validação de algoritmos de navegação, controle, sistemas multi agentes, técnicas de Inteligência Artificial, dentre outras aplicações. Esses robôs móveis podem ser capacitados a efetuarem tarefas de maneira individual ou em conjunto com outros robôs.

Em aplicações acadêmicas de pesquisas, tais como trabalhos desenvolvidos na Universidade Federal da Bahia como em [23] e [24] utilizam uma base omnidirecional de três rodas, como ilustrado na Figura 7 para desenvolver e validar técnicas variadas de controle.

Figura 7 – Robô AxeBot.



Fonte: UFBA.

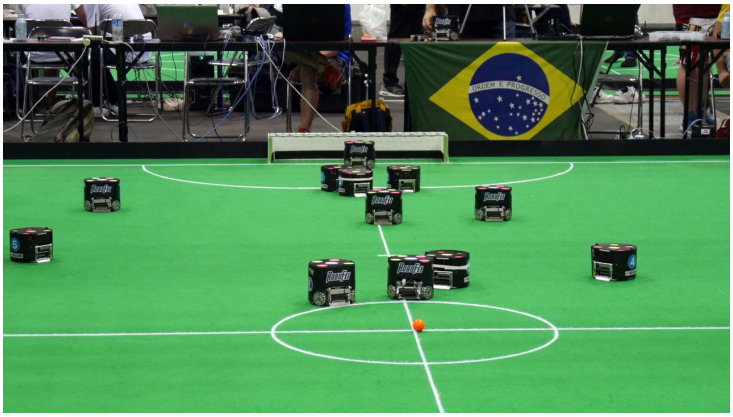
Estes trabalhos investem tanto ao desenvolvimento de robôs comerciais, como em [25] que desenvolve um protótipo robótico para aplicação em serviços domésticos, bem como plataforma para finalidades de pesquisas acadêmicas como em [26], [27], [28], [29] e mais recentemente em [30].

Segundo [18], aplicações em robótica necessitam de mais pesquisas e afirma que: *“ ... olhando para o futuro há muitos obstáculos na robótica, pois diversas aplicações*

desenvolvidas até aqui estão no início e terão crescimento significativo. Outras áreas mais desenvolvidas terão crescimento continuado ...".

Um ambiente de progresso (do inglês, *TestBed*) para pesquisadores de IA bastante utilizada para a realização de trabalhos em robótica móvel é a *RoboCup* [1], conforme visto em [31] e [32], onde os autores ressaltam a multidisciplinaridade do futebol de robôs e os desafios enfrentados na implementação de um agente inteligente robótico capaz de superar as dificuldades inerentes ao ambiente. Esses principais segmentos de pesquisa é parte integrante da *RoboCup*, onde atividades voltadas a competição para o avanço de pesquisa em robótica e inteligência artificial são realizadas e onde iniciativas científicas internacionais, como as utilizadas nas competições do futebol de robôs ilustrado na Figura 8 são desenvolvidas com o objetivo de aprimorar o estado da arte dos robôs inteligentes.

Figura 8 – RoboCup Soccer - *Small Size League* F180.



Fonte: [33].

O *Small Size Soccer* F180 é uma das divisões da *RoboCup League* e enfoca o problema da cooperação e controle multi-agente inteligente em um ambiente altamente dinâmico. Essa comunidade vem sendo consultada para solução de problemas e estudos por vários pesquisadores, utilizado cada vez mais para a evolução da inteligência artificial, por conter meios necessários para as pesquisas de cooperação entre robôs e estudos de sistemas autônomos [34].

Inicialmente a *RoboCup* foi criado pelo Dr. Itsuki Nota em 1993 [35] para ser um meio de divulgação da robótica e da pesquisa em Inteligência Artificial. Mas segundo [36] a comunidade deu a oportunidade para os pesquisadores trabalharem em algumas áreas, como por exemplo, em sistemas multi-agentes, estratégia, aprendizado, visão, redes neurais, controle, dentre outras.

[1] RoboCup é um projeto internacional conjunto para promover a Inteligência Artificial (IA), robótica e campos relacionados. É uma tentativa de promover IA e pesquisas em robótica inteligente, fornecendo um problema padrão onde uma vasta gama de tecnologias podem ser integradas e examinadas.

Embora o objetivo principal da *RoboCup League* seja uma copa do mundo com robôs reais, a comunidade oferece um ambiente de softwares para pesquisas. Essa liga também promove pesquisas sobre interações multi-agentes baseadas em computação gráfica e animações fisicamente realistas e um conjunto de tecnologias potencialmente providas, promove o uso avançado da internet.

A primeira edição realizada pela *RoboCup League* aconteceu em 1997 em Nagoia no Japão e vem sendo realizadas competições anuais até hoje sempre em locais distintos. O futebol de robôs vem sendo utilizado como laboratório para pesquisas desenvolvidas nas áreas de Robótica Móvel e Inteligência Artificial no Brasil desde 1996 [37].

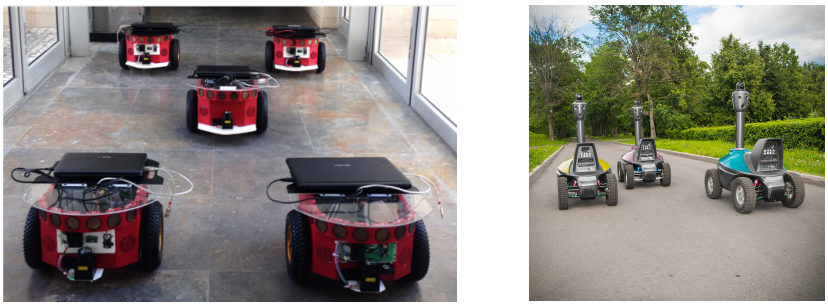
O Brasil começou a participar na categoria *Small Size* F180 com a equipe do Centro Universitário FEI (Faculdade de Engenharia Industrial) no mundial em 2009, na Áustria e em 2012, chegou as quartas de final, na disputa no México, figurando entre as oito melhores equipes do mundo.

Algumas equipes brasileiras surgiram em 1998 com um time de futebol de robôs simulados, como a UFBA (MecaTeam), que mostrou uma evolução na arquitetura do agente apresentado pela UFSC-*Team*-98 [38] na *RoboCup'98* chamado de Agente Autônomo Concorrente [38]. O *MecaTeam F-180 Multi-Robot System* apresenta uma abordagem de sistema multiagente para um sistema multi-robô de controle distribuído sob a *RoboCup Soccer Small Size League*. Este sistema multiagente utiliza a arquitetura do agente Autônomo Concorrente, onde o nível reativo encapsula a geração e o controle da trajetória como um comportamento, e a estrutura gerada pelo sistema de visão é usada pelo nível de decisão instintivo.

Os sistemas multirobôs (SMR) se caracterizam, pela existência de um comportamento cooperativo, envolvendo o controle coordenado de vários robôs, que emerge da interação de uma equipe de robôs na realização de uma tarefa, no qual pode ser utilizado um sistema de controle centralizado ou distribuído. As Figuras 9 a) e b) mostra exemplos de sistemas multirobôs.

Figura 9 – Sistemas multirobôs.

(a) **(b)**



Fontes: a) [39], b) [40].

O sistema de controle centralizado utiliza um único sistema, sendo responsável por determinar as ações de todos os robôs da equipe e pela sincronização destas ações, visando a realização de uma tarefa [38]. Já num sistema de controle distribuído cada um dos robôs que compõe a equipe é dotado de autonomia para determinação de suas ações e de habilidades para interagir com outros robôs visando a execução de um plano composto por ações de múltiplos robôs que cooperam para a realização de uma dada tarefa [38]. Algumas das importantes vantagens de se utilizar sistemas multirobôs são: a execução de tarefas complexas com maior eficiência, menor custo e maior tolerância a falhas. Além disso, pode ser realizado sensoriamento e mapeamento de grandes áreas, localização de minas terrestres, busca, resgate e vigilância de grandes áreas.

Consequentemente, um dos problemas fundamentais para os sistemas multirobôs é a necessidade de planejar e controlar as tarefas e as trajetórias dos robôs tal que eles não colidam entre si nem com os eventuais obstáculos. Para tal, é essencial a necessidade de algum tipo de coordenação [41].

O presente trabalho objetiva a reconstrução de uma nova arquitetura de hardware para o robô móvel AxeBot, com algumas semelhanças do projeto anterior desenvolvido em [26], onde adaptações ao sistema de controle foi remodelado. Ademais, também se concebeu uma estrutura mecânica e eletrônica para um novo robô móvel omnidirecional, no qual estará de acordo com as dimensões exigidas para a categoria *Small Size Soccer* F180 do futebol de robôs.

Desse modo, para proporcionar ao robô desenvolvido nesse trabalho executar tarefas de maneira autônoma, a sua arquitetura foi configurada para atuar através de um agente autônomo em conformidade com as exigências de hardware para execução em um robô móvel omnidirecional de três rodas. No entanto, consiste apenas em um subconjunto de seus recursos: àqueles necessários para executar um agente cognitivo embarcado. Além disso, o projeto demonstra que a plataforma desenvolvida é mais avançada do que foi somente configurada para essa tarefa, ou seja, vai além de tentativas anteriores como em [29]. Ademais, pode ser aplicada como base para estudos de tópicos relacionados à inteligência artificial e sistemas multi-agentes utilizando outras plataformas de tipos e modelos distintos das que foram aqui utilizadas.

Com o trabalho desenvolvido, além da reconstrução e aperfeiçoamento do robô omnidirecional AxeBot com uma nova estrutura, surge uma arquitetura prática para o desenvolvimento de pesquisas em sistemas embarcados, tornando-a de grande valia para atividades acadêmicas. Adicionalmente, foi possível concluir que a necessidade da sociedade acadêmica atual pela robótica móvel, diferentemente da robótica a nível industrial, apresenta-se em crescimento, o que justifica maiores necessidades de pesquisas nesta área. Também foi possível perceber a falta de trabalhos que se proponham a atendam aos mesmos objetivos que este, justificando a relevância do mesmo.

1.4 Organização do Trabalho

Além do capítulo introdutório este trabalho está estruturado da seguinte forma:

- O Capítulo 2 se inicia com o Estado da Arte de robôs móveis utilizados em pesquisas. Em seguida, aborda-se conceitos sobre as definições de robótica móvel, no qual conduzirá parte do conteúdo dessa dissertação;
- O Capítulo 3 apresenta conceitos gerais de sistemas embarcados, agentes inteligentes artificiais, o protocolo CAN e os elementos que constituem a arquitetura de hardware proposta, configurada para embarque do Agente Autônomo Concorrente;
- O Capítulo 4 tem como objetivo apresentar o desenvolvimento do robô;
- O Capítulo 5 apresenta as implementações do modelo cinemático e do agente autônomo;
- O Capítulo 6 apresenta as considerações finais, bem como sugestões para trabalhos futuros.

2 Robótica Móvel

Este capítulo apresenta a contextualização da robótica móvel, abordando alguns conceitos importantes que serão tratados neste trabalho, retratando critérios que norteiam assuntos relacionados ao tema proposto, para compreensão dos objetivos. Inicialmente, será abordado o estado da arte de plataformas robóticas utilizadas em pesquisas. Dando seguimento ao Capítulo, definições de robótica são apresentadas e por fim é abordada uma breve descrição sobre controle de robôs móveis.

2.1 Estado da Arte de Robôs Utilizados em Pesquisas

Recentemente, desenvolveram-se plataformas para robótica móvel para atenderem as necessidades de estudos, pesquisas e para utilização comercial. Algumas dessas plataformas possuem *hardware* para navegação de alto nível, contendo inúmeros recursos para tal finalidade, como sensores, atuadores, câmeras, dentre outros. Outras, oferecem inúmeras possibilidades de programação para o rastreamento de trajetória, como para mobilidade diferencial e omnidirecional (não-holonômicos e holonômicos), no qual são podem ser configurado até mesmo um sistema de visão computacional.

Existem também plataformas que disponibilizam um sistema operacional embarcado que englobam a comunicação de elementos como sensores a laser 3D permitindo aplicação de técnicas avançadas para utilização da robótica aplicada a processos de mineração em ambientes hostis, como linha férrea e minas terrestres, como por exemplo alguns robôs dedicados às tarefas de inspeção, desenvolvidos no Instituto Tecnológico da Vale, em parceria com a UFOP - Universidade Federal de Ouro Preto-MG [42]. Além disso, aponta-se o alto custo como ponto importante para a aquisição dessas avançadas plataformas. Geralmente as plataformas para robótica móvel inteligente comerciais possuem dois níveis de camadas bem definidas, dentre as quais destacam-se:

- Uma camada composta com os drivers para acionamento dos atuadores (motor com sensor) para informações de sensoriamento de velocidade;
- Uma camada que trata do processo decisório (microcontrolador), no qual este geralmente possui um sistema operacional embarcado.

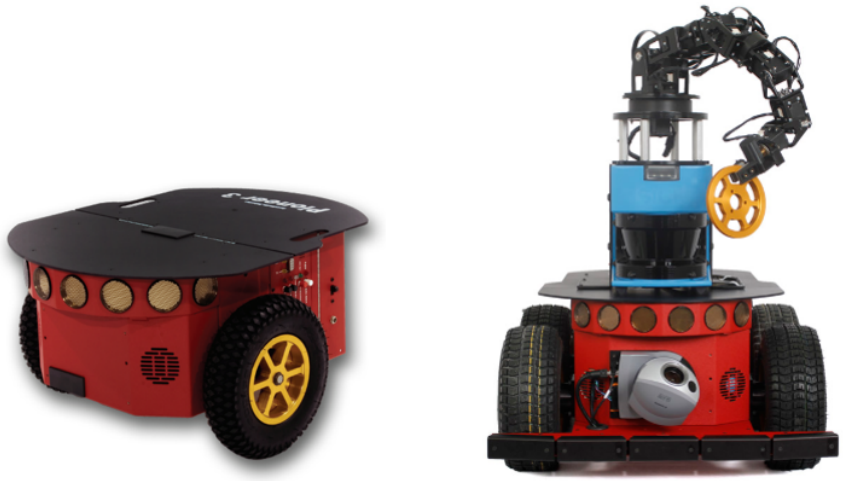
Essa camada, que contém o microcontrolador, permite que o usuário embarque aplicações diversas no robô. Além disso, muitos utilizam câmeras para se guiar através de um sistema de visão para realização de determinada tarefa como em [43]. Dentre algumas companhias que desenvolvem plataformas robóticas com rodas para utilização em projetos

e pesquisas em universidades estão a *Clearpath Robotics Inc.*, *Adept Mobilerobots*, *K-Team*, dentre outras. Nas subseções a seguir, será abordada algumas dessas principais plataformas, citando algumas das suas principais características e alguns trabalhos desenvolvidos com o tema tratado.

2.1.1 Adept MobileRobots

A *Adept MobileRobots* é uma companhia que desenvolve robôs voltados para pesquisadores. Fundada em 1995, têm em seu portfólio uma série de robôs de auto desempenho, fornecendo soluções para organizações como a *Hitachi*, *Microsoft*, MIT, *Siemens*, exército dos Estados Unidos e NASA. Em sua gama de robôs, podemos citar as famílias: *Pioneer*, *PeopleBot*, *PowerBot*, etc. Todas essas plataformas são utilizadas em pesquisa e desenvolvimento, com aplicações em áreas de pesquisas como localização, monitoramento, visão computacional, manipulação, teleoperação, cooperação multi-robôs, navegação autônoma, sendo essa última, o foco deste trabalho. O robô PIONEER é ilustrado na Figura 10.

Figura 10 – Robô móvel PIONEER 3-DX.



Fonte: [44].

Dentre os modelos citados, o robô móvel com tração diferencial PIONEER 3-DX é um dos robôs mais utilizados em pesquisas no ambiente acadêmico, devido a sua versatilidade e confiabilidade. Esse robô possui duas rodas tracionadas de 19cm com dois motores acoplados a encoders de 500 *ticks* de precisão com uma roda livre (tipo caster), é capaz de detectar objetos a uma distância de até 7m e pode atingir uma velocidade de 1,4m/s. Em outras versões pode ser aprimorado, compondo um manipulador embarcado de 7-DOF e efetuador de 2-DOF.

Em sua primeira versão, o robô PIONEER P3–DX, utilizava um microcontrolador da Motorola, o 68HC11 com sistema operacional próprio, o PSOS (*Pioneer Server Operating System*). Atualmente, o PIONEER utiliza o microcontrolador da *Renesas* SH2 de 32-bit e 44 MHz juntamente com o sistema operacional ARCOS (*ActivMedia's Robot Control and Operations Systems*).

Um trabalho desenvolvido com esse tipo de robô foi realizado em [45], onde o autor investiga se sistemas de controle *fuzzy* são mais robustos do que sistemas de controle neurais, ambos otimizados por um algoritmo genético em simulação e embarcado para o robô real na tarefa de navegação autônoma evitando obstáculos.

2.1.2 Clearpath Robotics

A empresa canadense *Clearpath Robotics inc.* criaram o robô HUSKY A200, uma plataforma de desenvolvimento robótico de tamanho médio e robusto com tração 4x4, projetado para ambientes externo agressivo personalizado para atender às necessidades de pesquisas. Esse protótipo possui câmeras, sensor a laser3D (*Lidar sensor*), GPS (do inglês, *Global Positioning System*), em outros modelos e versões podem incluir módulos opcionais. Além disso, o HUSKY foi a primeira plataforma robótica de campo a suportar o ROS (do inglês, *Robotic Operating System*) a partir de suas configurações de fábrica. A Figura 11 ilustra o robô.

Figura 11 – Robô móvel HUSKY A200.

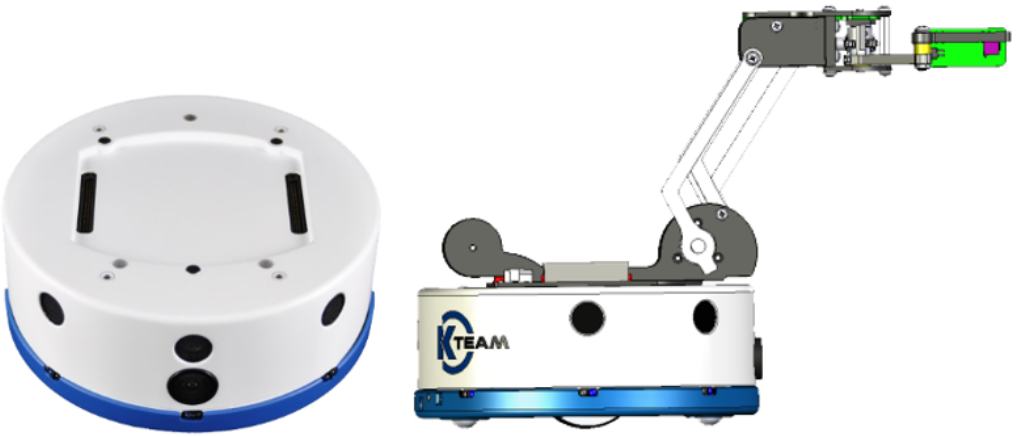
Fonte: [46].

O HUSKY utiliza um protocolo serial de código aberto e possui suporte à API (do inglês, *Application Programming Interface*) para ROS e LABVIEW e opções para *C++* e *Python*. Com essa plataforma, vem sendo desenvolvidos na Universidade de Toronto no Canadá (Departamento Aeroespacial) trabalhos na área de robótica espacial, mecatrônica e desenvolvimento de microssatélites com tecnologias para rovers lunares, com pesquisas voltadas para a criação de algorítmos aeroespaciais para operação de robôs que podem locomover-se sem GPS ou outra assistência de posicionamento global no espaço.

2.1.3 K-Team

Dentre as companhias que produzem plataformas robóticas citadas existe também a K-TEAM, uma desenvolvedora suíça de robôs dedicada a pesquisas, ensino e desenvolvimento. Atualmente essa companhia possui os modelos: *KiloBot*, *Khepera IV* e *Koala* 2.5. Entre as citadas, a mais utilizada é a plataforma modular e multi funções KHEPERA ilustrado na Figura 12 que atualmente está em sua quarta versão, na qual vem dotada de suporte para comunicação com softwares como o Labview e MATLAB®.

Figura 12 – Robô móvel KHEPERA IV.



Fonte: [47].

Nesse modelo, existe a possibilidade de expansão através da conexão de outros módulos com a necessidade da pesquisa, como módulos de visão computacional, garra articulada e módulo de comunicação via rádio, além disso integra um núcleo Linux com WiFi, Bluetooth, acelerômetro e giroscópio. O módulo da base móvel possui duas rodas tracionadas, mais dois apoios composto por rodas castor de deslizamento com a superfície de movimento, na qual contribuem para o equilíbrio da base. Esse robô é utilizado em áreas de pesquisas como: navegação, sistemas multi-agentes, visão e inteligência artificial.

Um trabalho desenvolvido com esse tipo de robô foi realizado em [48], onde os autores faz uma abordagem incremental usada para simular a evolução dos controladores neurais para evitar obstáculos estáticos e com isso provou ser mais eficiente do que uma abordagem direta da concorrência.

Dentre as plataformas citadas, podemos apontar o alto custo como ponto significativo. Uma alternativa adotada seria a montagem dessas plataformas utilizando *kits* e módulos comerciais, como por exemplo as que serão citadas e abordadas nesse trabalho.

2.2 Definições de Robótica

Segundo a Associação das Indústrias de Robótica RIA (do inglês, *Robotic Industries Association*) define-se um robô industrial como sendo: "Um manipulador multifuncional, reprogramável projetado para mover material, ferramentas ou dispositivos especializados através de movimentos programáveis variados para desenvolver uma variedade de tarefas". A norma ISO (do inglês, *International Organization for Standardization*) 10218, define robô como sendo: "uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional, que pode ter base fixa ou móvel para utilização em aplicações de automação industrial".

Outra importante definição de robôs móveis é descrita em [49] onde: "Um robô móvel pode ser definido como sendo um dispositivo mecânico implementado sobre uma base não fixa, que age sob o controle de um sistema computacional, equipado com sensores e atuadores, capaz de exercer interação com o ambiente".

De acordo com [50], tais interações, pode se obter no decorrer de ciclos de percepção-ação, que contitui-se em três fundamentos básicos:

- Aquisição de informações através de sensores;
- Processamento das informações adquiridas e seleção de ações a serem executadas;
- Execução das ações planejadas através do acionamento dos atuadores.

Ciclos de percepção-ação, através do controle de seus movimentos é ilustrado na Figura 13. O intuito é de obter informações interpretadas, adquirir autonomia e poder modificar seu estado no ambiente para assim realizar seus objetivos [22].

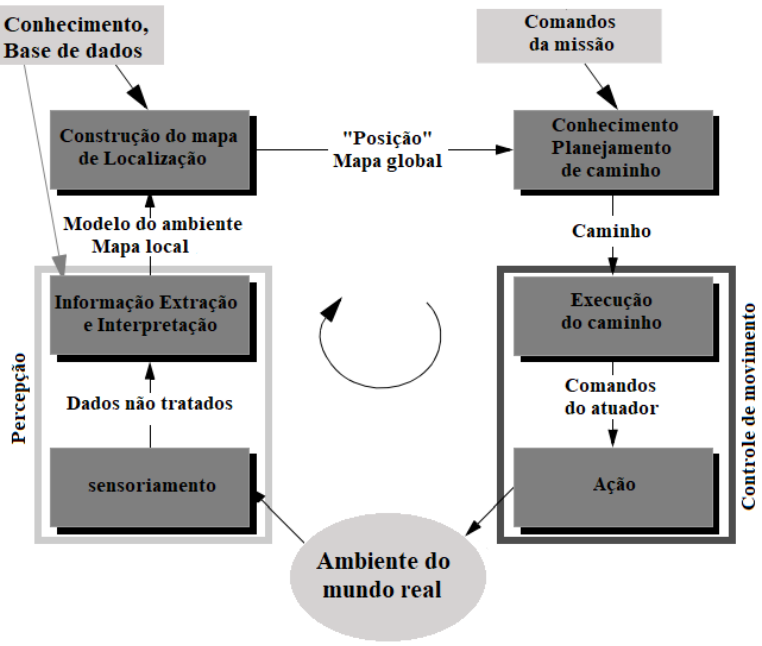
Segundo [51] uma definição mais genérica é que: *"um robô é um sistema autônomo que existe no mundo físico, que reconhece o ambiente a volta e poder agir sobre ele para alcançar suas metas"*.

O êxito no desenvolvimento de qualquer sistema de controle está profundamente condicionado a uma boa caracterização das suas partes, em especial uma descrição confiável do processo a ser controlado e uma boa caracterização dos dispositivos sensores e atuadores [52]. Outras importantes definições de robótica industrial são apresentadas em [53].

2.2.1 Locomoção de robôs móveis autônomos

No momento em que são solicitados algum nível de autonomia para um robô executar determinado tipo de tarefa, como por exemplo: percepção e atuação, reconhecimento do meio e localização, supervisão e controle, torna-se necessária a utilização de um agente móvel autônomo projetado para tal finalidade. Uma abordagem decisória, provém de

Figura 13 – Elementos de um sistema de controle para robôs móveis.



Fonte: Baseado em [22].

alguma forma, se basear com o processo de tomada de decisão do ser humano, para que um agente robótico realize suas tarefas.

Os fundamentos teóricos necessários ao desenvolvimento destes agentes são em parte comuns àqueles utilizados para agentes estacionários, porém, com peculiaridades nos domínios mecânicos e elétricos [52]. A função mais básica de um robô móvel é a sua locomoção, que geralmente é regida por um planejamento de trajetória e executada por uma estratégia de controle.

Conforme [54], algumas etapas necessitam serem realizadas por um robô móvel autônomo para que o planejamento de trajetória seja alcançado com eficiência, levando-se em consideração um comportamento de navegação deliberativa. Estas etapas estão determinadas em níveis hierárquicos de abstração como:

- Mapeamento e a representação do ambiente;
- Planejamento da trajetória;
- Geração e controle da trajetória.

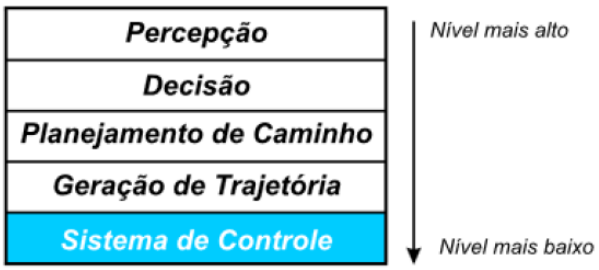
Ainda segundo [54], a maioria dos mecanismos de locomoção usados nos robôs móveis é inspirada em modelos biológicos. Porém existe uma categoria que não segue este modelo, os robôs com rodas, também conhecidos como veículos, extremamente eficientes em superfícies planas. Existe uma grande variedade de movimentos que podem ser executados

por um robô e a escolha de um sistema ideal de locomoção é um aspecto importante a ser considerado no projeto.

No sistema de controle de um robô, o conhecimento é armazenado em um modelo interno do mundo que pode ser construído a partir do conhecimento por suposição sobre o ambiente e de informações adquiridas pelos sensores do robô [6]. Esse modelo que o robô possui do mundo pode ser exibido de diversas formas, como por exemplo um tipo simbólico baseado em lógica, como os tipicamente utilizados em inteligência artificial [55].

Uma Visão mais geral em [56] define uma arquitetura de navegação de robôs móveis, onde cada camada é composta por um nível de pesquisa e juntas realizam a navegação do robô como ilustrado na Figura 14.

Figura 14 – Camadas de execução na navegação do robô.



Fonte: [56].

A primeira é a percepção do ambiente pelo robô, através de sensores. Após tratar esses dados adquiridos do ambiente ao qual está inserido, é executado o nível de decisão, onde o robô avalia a informação e decide que ações executar. Essa camada é o cérebro da máquina e pode possuir algoritmos de inteligência artificial.

2.2.2 Estrutura física de robôs com rodas

Atualmente existe um amplo campo de pesquisa e desenvolvimento para cada forma de deslocamento do robô móvel. No caso dos robôs com rodas, o tipo de roda e a estrutura do chassi são consideradas para a mobilidade no qual o protótipo é estruturado.

Nos robôs móveis composto por rodas, encontram-se duas limitações para cada tipo: a primeira limitação refere-se ao rolamento, onde o giro ocorre no sentido apropriado. A segunda limitação caracteriza o deslizamento lateral em que a roda deve deslizar ortogonalmente ao seu plano, normalmente aplicado com rodas omnidirecionais. Esta última, será abordada com mais detalhes nas próximas subsecções por ser parte integrante do desenvolvimento da arquitetura robótica modular desse trabalho.

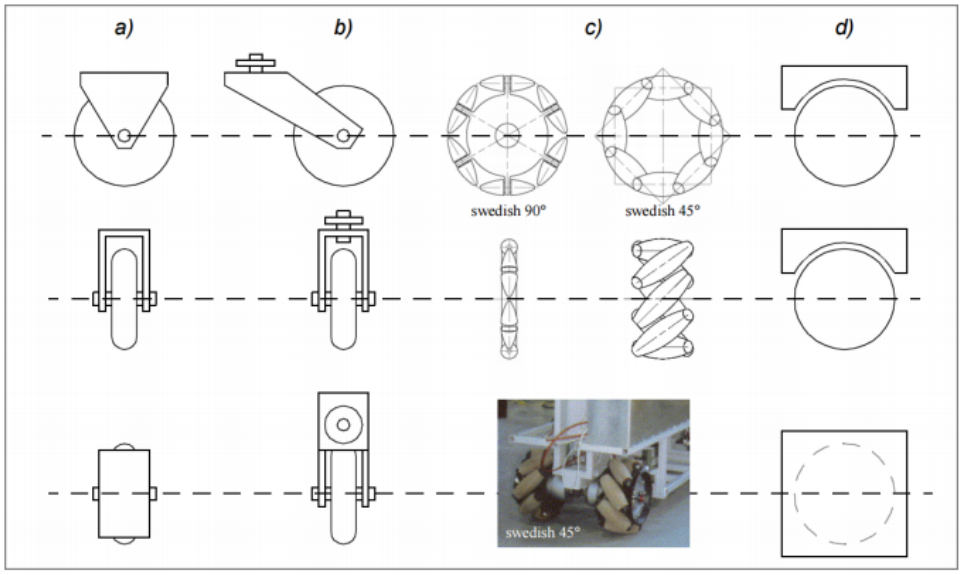
De acordo com [17], os robôs podem ser divididos em quatro categorias:

- Aquáticos;
- Terrestres;
- Espaciais;
- Aéreos.

Na categoria de robôs terrestres com rodas, existem algumas configurações associadas a parte mecânica de locomoção, que é o artifício responsável pelo movimento do robô, como: diferencial, triciclo, Ackerman, e omnidirecional [57].

Segundo [22], é possível dar estabilidade ao robô com duas ou três rodas e ao usar quatro ou mais é fundamental um sistema de suspensão para comportar que todas as rodas toquem o chão quando o robô estiver em terreno irregular. Em [58] são listados quatro tipos de rodas mais utilizadas. A Figura 15 mostra uma ilustração dos tipos de rodas citados.

Figura 15 – Tipos de rodas.



Fonte: [22].

(a) Rodas padrão: dois graus de liberdade; rotação ao redor do eixo (motorizado) da roda e do ponto de contato;

(b) Rodas castor orientável: dois graus de liberdade; rotação ao redor de uma junta deslocada;

(c) Rodas sueca: três graus de liberdade; rotação ao redor do eixo (motorizado) da roda;

(d) Rodas esféricas: dois graus de liberdade; possui limitações técnicas [22].

A quantidade de liberdade dos movimentos da estrutura de um robô, são definidas pelas rodas, pois cada modelo possui diferentes graus de liberdade DOF (do inglês, *Degrees Of Freedom*). Por exemplo, para fazer um robô omnidirecional se movimentar para qualquer lado sem a necessidade de reorientação, costuma-se usar rodas suecas ou esféricas, mas também é permitido montar com rodas orientáveis centralizadas [58].

As rodas omnidirecionais destaca por sua capacidade de se mover em qualquer sentido, ou seja, para frente, para trás, para os lados e ainda tem a capacidade de girar em torno do seu próprio eixo utilizando-se um motor para cada roda. Esse tipo de roda, são as utilizadas nos robôs utilizados em competições de futebol de robôs da liga *RoboCup* na categoria *Small Size* F180.

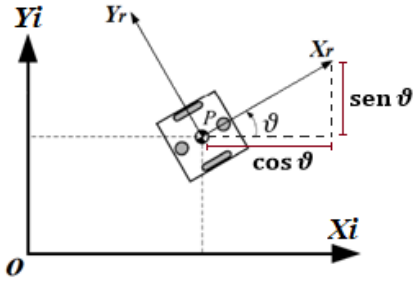
2.2.3 Representação de posição e orientação de um robô móvel

De acordo com [22], os robôs móveis terrestres podem ser modelados como corpos rígidos sobre rodas operando sobre um plano horizontal. Nesse plano, a base é composta de três dimensões: duas para a posição no plano e uma para a orientação ao longo do eixo vertical, sendo esse ortogonal ao plano.

Considerando-se que as rodas são rígidas e isentas de deformações e não há escorregamento, ou seja, todo movimento produzido pela roda é transmitido em movimento para estrutura móvel do robô. Independentemente do tipo de robô a se considerar, é necessária uma breve descrição das relações entre os diversos sistemas de coordenadas dos sistemas locais fixados nos centros de massa de suas junções e de um sistema global, no qual é constituído por um referencial inercial distinto e trivial como sistema de referência do mundo.

A indicação da posição do robô sobre o plano especifica uma associação entre a referência local do robô e a referência global do plano e a orientação é dada por uma matriz R_w^r comumente chamada de matriz de rotação. Como ilustrado na Figura 16, os eixos X_w e Y_w determinam uma base inercial em relação ao plano, referindo-se ao sistema global desde a sua origem **O**.

Figura 16 – Coordenadas das Referências Global e Local do Robô Móvel.



Fonte: Baseada em [22].

Para estabelecer a posição do robô, é determinado um ponto **P** de referência sobre sua base. Os sistemas de coordenadas e as relações entre essa referência geralmente são dadas pela matriz de rotação R_r^i, obtidas por:

$$R_r^i = [X_r^i \mid Y_r^i] \tag{2.1}$$

Sendo que X_r^i e Y_r^i são as coordenadas dos vetores X_r e Y_r de $O_rX_rY_r$ com relação a $O_iX_iY_i$. Desse modo:

$$X_r^i = \begin{bmatrix} cos(\vartheta) \\ sen(\vartheta) \end{bmatrix}, Y_r^i = \begin{bmatrix} -sen(\vartheta) \\ cos(\vartheta) \end{bmatrix} \tag{2.2}$$

Ao realizar a substituição das Matrizes de (2.2) em (2.1), temos:

$$R_r^i = \begin{bmatrix} cos(\vartheta) & -sen(\vartheta) \\ sen(\vartheta) & cos(\vartheta) \end{bmatrix} \tag{2.3}$$

A base do robô é indicada pela referência local, dada por $[X_r;Y_r]$, que são os eixos referente ao ponto $\boldsymbol{P}$. Esse ponto é representado pela expressão:

$$P^i = R_r^i \cdot p_r \tag{2.4}$$

Sendo a posição de $\boldsymbol{P}$ na referência global especificada pelas coordenadas X e Y, a diferença angular entre as duas referências é dado por ϑ. Realizadas as definições, a posição do robô pode ser descrita como um vetor com três elementos, onde é definido que a base da posição está supramencionada globalmente.

$$\zeta_i = \begin{bmatrix} X \\ Y \\ \vartheta \end{bmatrix} \tag{2.5}$$

Sendo que X e Y é a posição do centro de massa do robô e ϑ a orientação. Geralmente é utilizado o termo pose (postura) em robótica, no qual é empregado para discriminar um grupo de pontos que indicam a posição e orientação do robô em relação a um determinado sistema de coordenadas. Sabe-se também que existe uma rotação entre o sistema de coordenadas desse centro de massa e o sistema de coordenadas do referencial inercial, no qual é obtida a matriz de rotação ortogonal elementar em torno do eixo Z:

$$R(\vartheta(t)) = \begin{bmatrix} cos(\vartheta(t)) & sen(\vartheta(t)) & 0 \\ -sen(\vartheta(t)) & cos(\vartheta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

As matrizes de rotação pertencem a um grupo ortogonal e possuem importantes propriedades, as quais foram analisadas em [59]. Esta matriz de rotação é utilizada para esquematizar o movimento da referência global $[X_i,Y_i]$ em relação à referência local $[X_r,Y_r]$ em função de ϑ.

$$\zeta_r = R(\vartheta(t)) \cdot \zeta_i = \begin{bmatrix} cos(\vartheta(t)) & sen(\vartheta(t)) & 0 \\ -sen(\vartheta(t)) & cos(\vartheta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ \vartheta_i \end{bmatrix} \tag{2.7}$$

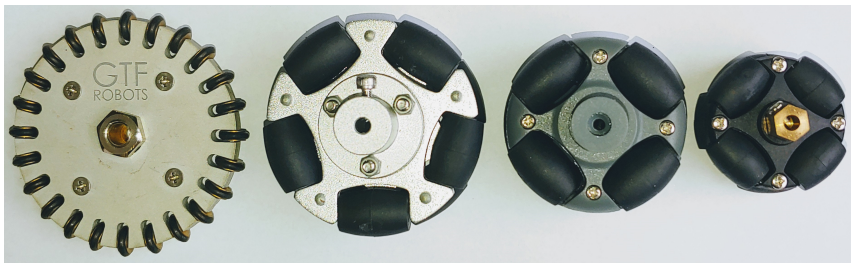
2.2.4 Rodas omnidirecionais

As rodas omnidirecionais ou suecas foram desenvolvidas pelo sueco Bengt Ilon na década de 1970. A rotação dessa roda é realizada sobre seu eixo principal, mas pode mover-se com pouco atrito em todas as direções, devido a sua forma e os pequenos rolamentos que constituem a roda principal maior [60], [61], [62], [63].

Os robôs móveis omnidirecionais utilizam rodas onde possuem rolamentos sobre sua superfície de contato, podendo ser capaz de traçar qualquer caminho no ambiente para atingir os locais necessários.

Segundo [22], esse tipo de roda possui algumas desvantagens que podem ser causadas pelas suas rodas auxiliares rolantes, que aumentam os graus de liberdade da roda, resultando em acúmulo de derrapagem, redução da precisão da odometria, aumento na complexidade do projeto. A Figura 17 ilustra quatro exemplos de rodas omnidirecionais, uma delas é utilizada para o este projeto, como será visto na próxima secção.

Figura 17 – Exemplos de rodas omnidirecionais.



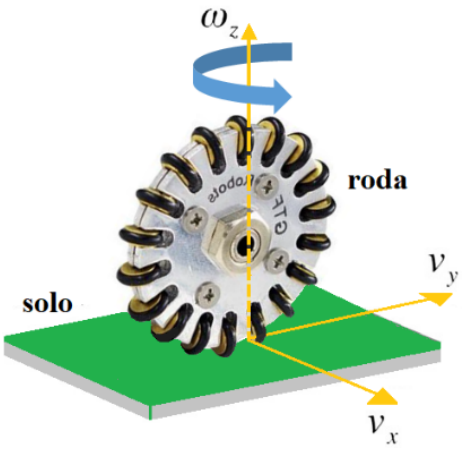
Fonte: Do autor.

Esses rolamentos auxiliares diminuem o atrito de deslizamento lateral da roda, permitindo um grau de liberdade a mais, possuindo três graus de liberdade. Isso permite a um robô omnidirecional movimentar-se em qualquer direção no plano horizontal e a capacidade de transladar em duas direções e girar em relação ao seu centro de massa.

O estudo de movimento do robô é fundamentado em sua geometria, considerando que o robô está se movendo sobre uma superfície plana e não há componentes maleáveis

na estrutura do robô e o atrito não é considerado, as rodas se comportam como uma junta plana considerando graus de liberdade DOF (do inglês, *Degrees Of Freedom*). A Figura 18 ilustra a roda omnidirecional da GTF *Robots* utilizada nesse projeto em contato com o solo e o sistema de referência.

Figura 18 – Coordenadas da roda omnidirecional no solo.



Fonte: Do autor.

Considerando que a roda é um componente inflexível, ela está constantemente tocando o solo em um ponto único e adequa-se como origem para o sistema de coordenadas ilustrado na Figura 18. A direção ν_y indica a velocidade na direção normal da roda, ν_x mostra a velocidade na direção lateral.

O deslizamento ω_z é a velocidade de rotação quando o robô gira. A roda omnidirecional é definida como uma roda padrão fornecido por uma matriz de rolamentos, cujo eixo gira perpendicularmente para a direção normal da roda [64]. Com isso, observa-se uma relação de rotação simples **R** ao redor do centro de massa **P** do robô ilustrado na 22, na qual a Matriz 2.8 de rotação é obtida.

$$R(\theta) = \begin{bmatrix} cos(\theta) & -sen(\theta) & 0 \\ sen(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.8}$$

A partir da geometria dos sistemas de coordenadas, as equações da cinemática[2] do movimento do robô podem ser obtidas. Dessa forma, o modelo cinemático bem conhecido de um robô omnidirecional localizado a $[X_i, Y_i, \theta_i]$ pode ser escrito como: $x_i(t) = dx(t)/dt$; $y_i(t) = dy(t)/dt$; $\omega_i(t) = d\theta(t)/dt$.

[2] De acordo com [65] a cinemática é a ciência que trata do movimento, sem considerar as forças que causam. Dentro da cinemática, se estudam a posição, velocidade, aceleração e todas as derivadas de maior ordem das variáveis de posição (em relação ao tempo ou a qualquer outra variável).

2.3 Robôs móveis omnidirecionais

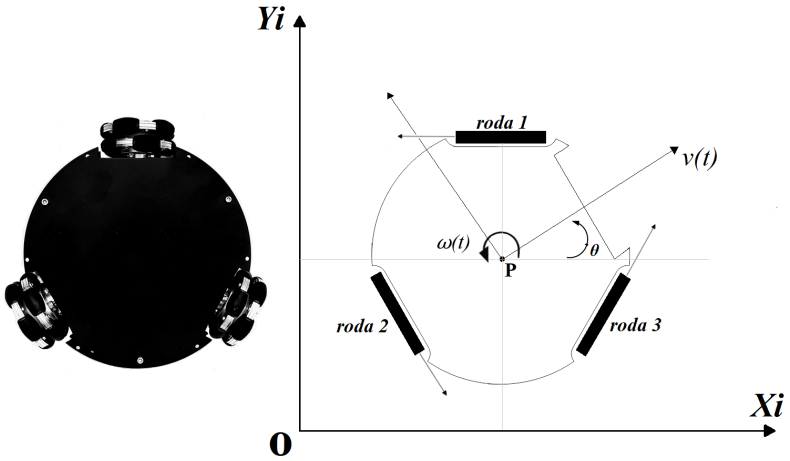
O espaço de trabalho de um robô móvel é importante porque define a gama de poses possíveis que o robô móvel pode alcançar em seu ambiente [22]. Uma das coisas importantes ao definir o espaço de trabalho de um robô móvel é que primeiramente se examinam suas velocidades admissíveis [66]. Devido a isso, ao descrever o espaço de trabalho de um robô móvel omnidirecional, frequentemente o conceito de holonomicidade é utilizado.

Segundo [67], um robô móvel omnidirecional holonômico[3] provê simplicidade de controle em várias aplicações, como em navegação autônoma, manipulação móvel e controle de reboque.

De acordo com [68], robôs omnidirecionais tem "[...] total mobilidade no plano, o que ele pode se movimentar a cada instante em qualquer direção sem qualquer reorientação".

Assim, um veículo omnidirecional ilustrado na Figura 19 tem seu posicionamento definido por três dimensões: duas para representar a sua posição no plano e uma para a rotação em relação ao seu eixo vertical, que é ortogonal ao plano de movimentação.

Figura 19 – Representação do robô omnidirecional.



Fonte: Do autor.

De acordo com [69], existe outra maneira de se descrever um robô holonômico que é pela relação entre os graus de liberdade de seus atuadores e os graus de liberdade do seu espaço de trabalho através do ("número de coordenadas necessárias para especificar a configuração do sistema").

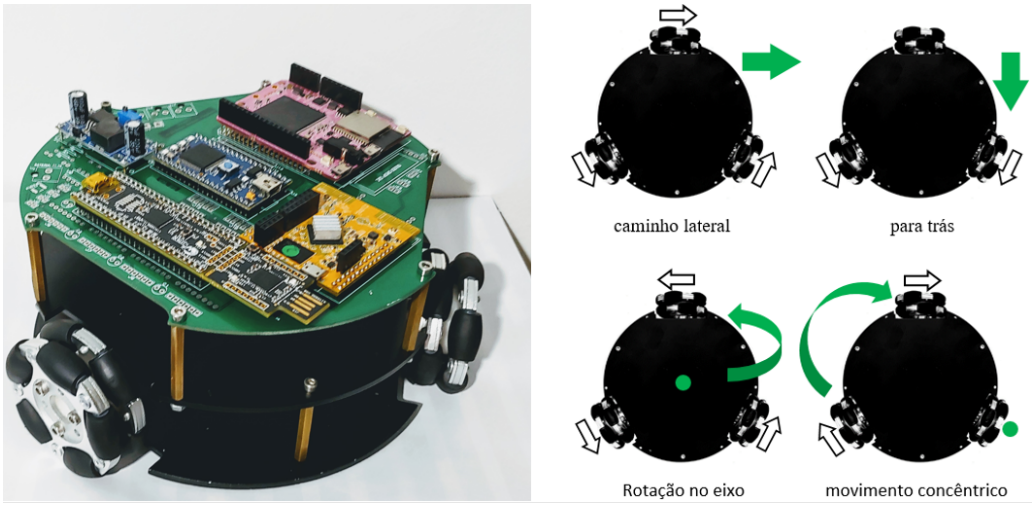
Um robô omnidirecional é, portanto, um robô holonômico com três graus de liberdade nos atuadores [22]. Esta definição desconsidera a composição física do mecanismo,

[3] O termo "holonômico" possui muitas aplicações para diversas áreas da matemática, mas, para a robótica móvel, o termo se refere exclusivamente às restrições cinemáticas do chassi do robô. Um robô holonômico é um robô com zero restrições cinemáticas não holonômicas. Inversamente, um robô não holonômico é um robô com uma ou mais restrições cinemáticas não holonômicas [22].

que pode ser criado de diversas maneiras diferentes para atingir a holonomia [69].

Devido a dinâmica de movimentação, o desenvolvimento de uma plataforma robótica omnidirecional holonômica, como por exemplo o robô da Figura 20, se torna útil para aplicações em pesquisas em robótica móvel.

Figura 20 – Robô móvel omnidirecional.



Fonte: Do autor.

Existem diversas aplicações que utilizam plataformas omnidirecionais. Algumas apresentam uma configuração com 4 rodas proporcionando movimentos em todas as direções para facilitar manobras em espaços reduzidos. Recentemente, a empresa *Vehicle Technologies Inc.* apresentou no mercado industrial a empilhadeira *Sidewinder* ATX-3000 ilustrado na Figura 21 (a). Existem também plataformas desenvolvidas para aplicações de pesquisas em robótica móvel, como a plataforma omnidirecional holonômica composta de manipulador articulado da KUKA ilustrado na Figura 21 (b).

Figura 21 – Robôs móveis omnidirecionais.

(a) **(b)**

Fonte: a) [70], b) [71].

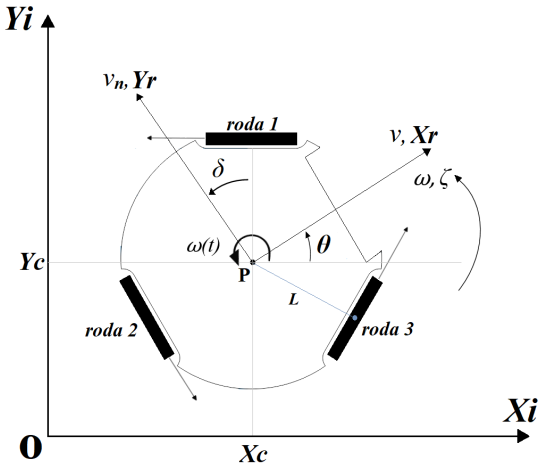
Segundo [67], uma plataforma móvel omnidirecional holonômica provê simplicidade de controle em várias aplicações, como em navegação autônoma, manipulação móvel e controle de reboque.

2.3.1 Cinemática dos robôs omnidirecionais

O modelo cinemático dos robôs móveis são de extrema importância para o projeto do controlador de seus atuadores, principalmente quando os robôs executam trajetórias com velocidades elevadas ou trajetórias com mudanças bruscas de sentido e direção, exigindo força máxima dos atuadores [72].

A Figura 22 ilustra essa relação.

Figura 22 – Representação do robô no sistema de coordenadas.



Fonte: Do autor.

Inicialmente é definido dois sistemas de coordenadas:

- $[Xi; Yi]$ sistema de coordenadas global fixo no ambiente;
- $[Xc; Yc]$ sistema de coordenada do robô.

A orientação do robô é definida pelo ângulo θ que corresponde ao ângulo entre os dois sistemas de coordenadas.

Alguns estudos da modelagem cinemática do robô móvel AxeBot semelhante ao apresentado neste trabalho pode ser visto em [73] e [54], na qual comprovam que o modelo cinemático de um robô móvel omnidirecional é dado pela relação matemática entre as velocidades lineares e angulares dos seus atuadores e a velocidade de seu centro de massa, ou seja, esse modelo representa a equação de seu movimento em função das velocidades de suas rodas sem conceituar as forças que agem sobre o mesmo. Uma representação mais

detalhada da cinemática e modelagem do robô omnidirecional desenvolvido nesse projeto é abordada do Capítulo 4.

2.4 Odometria

Esta seção introduz a odometria, técnica amplamente usada por sistemas de navegação como por exemplo: Algoritmos de localização, mapeamento e exploração. Permite a integração incremental do movimento do robô através dos encoders acoplados aos motores e tem precisão bastante considerável em curtas distâncias. A implementação mais simples de *dead reckoning*[4] é as vezes chamada de odometria. A vasta maioria de sistemas robóticos terrestres em uso hoje dependem de *dead reckoning* para formar a base de seus sistemas de navegação.

O método da utilização da odometria é baseada na ideia de que as revoluções das rodas do robô podem ser convertidas em deslocamento linear relativo à superfície. O termo implica que o deslocamento de veículo ao longo do caminho do movimento é diretamente derivado de algum "odômetro" a bordo. Maneiras comuns de implementação de odometria envolve encoder incremental diretamente acoplados à armação do motor ou eixos das rodas como os utilizados neste projeto, no qual será abordado no Capítulo 4.

2.5 Controle de robôs móveis omnidirecionais

Na estratégia de controle sem realimentação (malha aberta), o controle se torna um problema de estimar com antecedência ao movimento o perfil de velocidade a ser executado [75]. As Soluções mais precisas utilizam controle por realimentação (malha fechada), e evidentemente dependem principalmente do bom funcionamento do sistema de odometria [76].

Aplicar controle com realimentação em robôs omnidirecionais é parcialmente simples, visto que esse tipo de plataforma robótica apresenta controlabilidade ou seja, sempre há um conjunto de velocidades para as rodas que ocasiona certas velocidades (rotacional e translacional) para o robô [77]. Segundo [75], o controle de posição de um robô móvel pode ser de três tipos, como:

- Seguir um caminho geométrico;
- Seguir uma trajetória dependente do tempo;
- ou pode-se atingir uma certa configuração estática.

[4] (derivado de *Deduced Reckoning* dos dias de navegação à vela), é um procedimento matemático simples para determinar a localização atual de uma embarcação pelo avanço de uma posição anterior através de informações de curso e velocidade conhecidos sobre um dado período de tempo [74].

Em [78] é apresentado o modelo cinemático e dinâmico de um robô de quatro rodas, incluindo a estimação de parâmetros, baseados no método dos mínimos quadrados. Em [79] também foi utilizado esse modelo para projetos de Controladores Preditivos, no qual foi utilizado um robô omnidirecional de três rodas para seguimento de trajetória controlado por um algoritmo MPC (do inglês, *Model Predictive Control*), no qual realiza um controle de velocidade e o modelo da cinemática inversa para gerar a referência de velocidade a partir da posição atual do robô.

Estas abordagens da literatura fortalecem a importância de se obter um modelo prático e com capacidade de se adaptar as melhores condições de rastreamento. Como base bibliográfica foram utilizados trabalhos realizados pelos pesquisadores e alunos do Laboratório de Robótica do Departamento de Engenharia Elétrica da UFBA e artigos da literatura especializada [80]; [26]; [52].

2.5.1 Controle fundamentado na cinemática

O Tipo de controle mais comumente utilizado, pois leva em consideração o modelo cinemático e considera-se que o rastreamento de trajetória está adequado ao projeto do controlador. O objetivo desse tipo de controle é de encontrar valores de entradas de velocidade ($\dot{\zeta}$) que estabilize o controle em malha fechada, dada uma pose de referência da base (ζ). Mas devido a algumas características físicas do robô como massa, características mecânicas e elétricas dos atuadores, muitas vezes na prática não se obtém um rastreamento de velocidade adequado. Segundo [81], este tipo de controle funciona bem em baixas velocidades e com robôs de estruturas reduzidas.

2.5.2 Controle fundamentado na dinâmica

O controle baseado na cinemática geralmente não é adequado para configurar com precisão o desempenho do robô. Com isso, diferentemente da cinemática, as características de sua dinâmica poder sofrer alterações quando o robô se encontra em aceleração e velocidades elevadas e quando este é de grande porte ou de estruturas maiores. Esse tipo de controle tem como objetivo fazer o robô rastrear uma determinada velocidade se baseando nos torques (T) entregues a seus atuadores. [54], em uma pesquisa de mestrado, utiliza um controlador por realimentação de estados, baseado na dinâmica de um robô omnidirecional. Outros trabalhos envolvendo controladores deste tipo de controle são encontrados em [82]. [83] utiliza um controlador adaptativo, baseado na dinâmica, para um robô omnidirecional.

2.5.3 Controle de trajetória

Em robótica móvel, existem diversas técnicas de rastreamento de trajetória, com diversos níveis de complexidade utilizando microcontroladores. As implementações mais

triviais podem controlar uma trajetória em linha reta de um determinado ponto a outro, ou seja, de origem a destino. No controle da trajetória, é importante analisar o perfil da velocidade executado pelo robô para garantir uma odometria eficiente para as características gerais do projeto.

Diversos trabalhos na área de robótica móvel autônoma, é direcionado a estudos e pesquisas para o desenvolvimento de técnicas para um controle de trajetória eficiente. O principal objetivo do controle de trajetória é primeiramente guiar o robô sobre um percurso (trajetória) definido pelo controlador projetado para essa finalidade. O controlador de trajetória está diretamente ligado aos modelos cinemático e dinâmico de robô. Com isso, algumas leis de controle para robótica móvel utilizam alguns procedimentos que são:

- **Rastreamento de Trajetória**: Com base nas coordenadas de postura $\zeta = (x, y, \theta)$ e de referência $\zeta_c = (x_c, y_c, \theta_c)$ do robô, deve-se encontrar uma lei de controle para as velocidades angular e linear de modo que o $lim_{t\to\infty}(x(t) - x_c(t)) = 0$, $lim_{t\to\infty}y(t) - y_c(t)) = 0$ e o $lim_{t\to\infty}(\theta(t) - \theta_c(t)) = 0$ [83];
- **Estabilização em um ponto**: Esse procedimento é adotado quando se deseja obter uma lei de controle de velocidade no qual o $lim_{t\to\infty}(\zeta - \zeta_c) = 0$, sendo $\zeta = (x, y, \theta)$ a postura do robô e $\zeta_c = (x_c, y_c, \theta_c)$ a postura de referência a ser seguida pelo robô;
- **Seguimento de caminhos**: Sendo que o caminho no plano ζ, e assumindo que ϵ_θ e ϵ_{xy} são o erro de orientação e a distância entre um ponto de referência no robô e o plano ζ, respectivamente, objetiva-se encontrar uma lei de controle de velocidade a qual o $lim_{t\to\infty} \mid \epsilon_\theta \mid$ e o $lim_{t\to\infty} \mid \epsilon_x y \mid = 0$ [83].

De acordo com [52], os requisitos operacionais de um sistema robótico vêm se tornando cada vez mais rigorosos, de modo que deve-se avaliar algumas estratégias de controle para que se tenha embasamento na escolha daquelas que apresentem melhor desempenho em determinada aplicação. Ainda segundo [52], nos diversos níveis estruturais do robô podem-se aplicar variadas técnicas de controle, sejam estas as consolidadas estratégias clássicas ou até as mais atuais estratégias de controle ótimo, sendo assim justificada a grande valia do desenvolvimento de uma plataforma experimental que possibilite uma avaliação criteriosa das diversas técnicas existentes. Para alguns processos no qual utilizam-se controle clássico, as ações PI (Proporcionais e Integrais) em conjunto são suficientes para atender aos objetivos de controle, como o que será utilizado neste projeto.

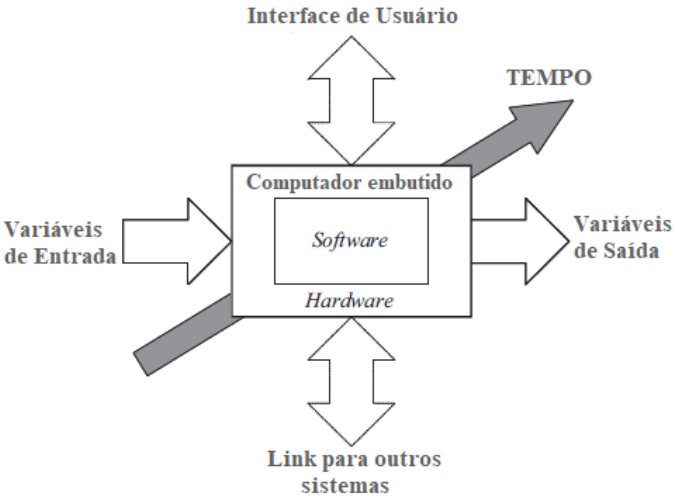
3 Fundamentação Teórica

Este capítulo abordará conceitos gerais sobre sistemas embarcados e de agentes inteligentes, ressaltando algumas definições da Inteligência Artificial. Em seguida o Agente Autônomo Concorrente e sua arquitetura cognitiva será descrita, com base na sua implementação original em [38]. Dando seguimento ao Capítulo, serão apresentadas as arquiteturas do Agente Autônomo Concorrente embarcado, a composição da estrutura de hardware e sua configuração. Em seguida, os conceitos gerais sobre rede CAN, no qual são compostos os barramentos de comunicação, proposto para embarque do AAC. Por fim, a lógica difusa será apresentada para implementar um sistema de inferência no agente embarcado composto pela arquitetura de hardware proposta.

3.1 Conceito geral de sistemas embarcados

Um sistema embarcado pode ser considerado um sistema operacional com propósitos específicos, normalmente construído em dimensões reduzidas, que permite que um veículo funcione de forma autônoma [81]. Um computador embarcado, normalmente um microcontrolador, realiza a execução de um planejamento destinado a esta finalidade, permanentemente armazenado em sua memória [84]. Segundo [85] um requisito normalmente presente em sistemas embarcados é a execução em tempo-real. Com isso, o tempo é outra variável que afetará tudo o que executa-se nesses sistemas, sendo representado por uma flecha que atravessa a imagem mostrada na Figura 23, onde é ilustrada uma visão geral de sistema embarcado como um esquema básico.

Figura 23 – O Sistema Embarcado.



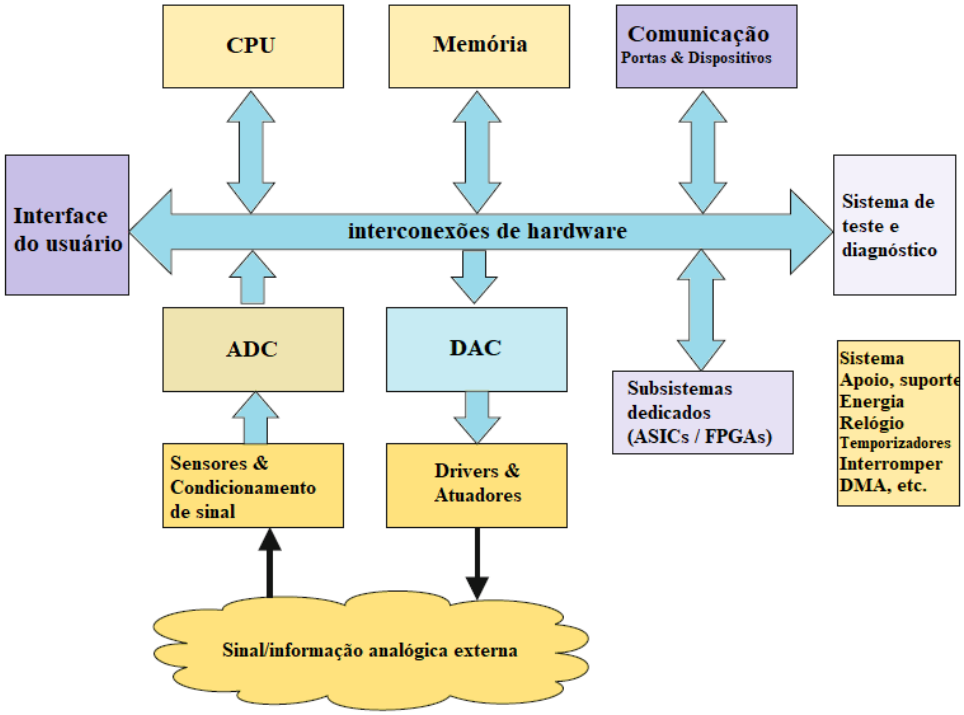
Fonte: [84].

Com o sistema baseado em informações de entrada, o microcontrolador estabelece as saídas, que são ligadas a efetuadores anexados ao sistema. Um sistema eletrônico real, juntamente com um componente eletromecânico, é continuamente chamado de hardware (gerenciado pelo *firmware*), composto por um controlador de entrada e saída de baixo nível e o programa que será executado é regularmente chamado de software.

Ainda segundo [84], também pode haver interação com um usuário por meio de um teclado e monitor e interações com outros subsistemas em outros lugares, mesmo que esses subsistemas não sejam essenciais para o conceito geral.

Geralmente, os componentes do hardware de um sistema embarcado possuem as plataformas e módulos eletrônicos necessários para o sistema executar a tarefa para a qual foi projetado [86]. Segundo o autor, um sistema embarcado possui três componentes principais de hardware: a Unidade Central de Processamento (CPU), a memória do sistema e um conjunto de portas de entrada e saída. A Figura 24 a seguir ilustra os elementos principais de um sistema embarcado.

Figura 24 – Elementos de hardware em um sistema embarcado.



Fonte: Baseado em [86].

A CPU executa instruções de software para processar as entradas e tomar as decisões que orientam a operação do sistema. A memória armazena programas e dados necessários para a operação do sistema. As portas de entradas e saídas E/S permitem transmitir sinais entre a CPU e o ambiente externo.

Os componentes de software de um sistema embarcado possuem os programas necessários para dar funcionalidade ao hardware do sistema. Esses programas, comumente chamados de *firmware*, são armazenados em algum tipo de memória não volátil [86]. Normalmente esses *firmwares* não devem ser modificado pelos usuários, embora alguns sistemas possam fornecer meios de realizar modificações.

3.2 Agentes Inteligentes Artificiais

Com o desenvolvimento da informática, internet e principalmente da robótica na década de 1990, intensificam-se pesquisas e determinam-se alguns agentes inteligentes para interfaces colaborativas [87]. Com o advento dessas tecnologias surgem unidades computacionais autônomas capazes de realizar determinado tipo de tarefa para auxiliar usuários de outros agentes [88].

Com base em [88], reformular conceitos do costume do ser humano em máquinas inteligentes é um dos objetivos da inteligência artificial. Esse fato cooperou em muitas razões para o desenvolvimento da tecnologia de agentes inteligentes artificiais.

Agentes se apresentam para alcançar melhores objetivos de si mesmos ou da sociedade/sistema em que eles existem. Uma importante definição para agentes autônomos foi argumentada em [89], onde afirmam que:

« *"Agentes autônomos são sistemas computacionais que habitam algum ambiente dinâmico complexo, detectam e agem de forma autônoma nesse ambiente e, ao fazer isso, realizam um conjunto de objetivos ou tarefas para os quais foram projetados".* »

Esses objetivos podem ou não serem celebrados pelos agentes sem restrições, conforme se os agentes são ou não fundamentados em objetivos. A sociabilidade pode permitir que os agentes articulem suas ações e comportamentos, tornando-se sistemas mais coerentes.

Um agente é qualquer entidade estabelecida a realizar determinada função, que possa perceber o ambiente e agir sobre o mesmo e poder obter um resultado, ou se existe incerteza, o melhor resultado esperado [90]. Segundo eles, num agente racional ideal para cada sequência de percepções, o agente escolhe a ação que maximiza seu desempenho baseado nas informações de percepção e de seu conhecimento adquirido sobre o meio.

[55], detalha que "Um agente é algo de pode ser visto como percebendo seu ambiente através de sensores e que age nesse meio através de atuadores". Ainda de acordo com os autores, o conceito de agentes racionais é um dos principais fatores para pesquisas em inteligência artificial e designa que o julgamento de racionalidade de determinado agente é dependente de quatro fatores como:

- A medida do desempenho que define o êxito do agente;
- A sequência de percepções obtidas do agente;
- O aprendizado que o agente adquiriu sobre o ambiente;
- As ações que o agente poderá realizar.

Segundo [55], existem diversas e possíveis definições para a Inteligência Artificial, no qual pertencem a categorias. Na Tabela 1 temos algumas definições de IA, logradas ao longo de duas influências. Os títulos dessas definições estão associados com os métodos de pensamento e raciocínio em termos de autenticidade ao desempenho humano. Comenta o autor que *"Um sistema é racional quando se faz a coisa certa, dado o que se sabe"*.

Tabela 1 – Definições da Inteligência Artificial.

Pensando Humanamente	**Pensando Racionalmente**
"Atividades que nos associa com o estudo das faculdades mentais tomada de decisões, resolução de problemas, aprendizagem" [92]; "O novo e empolgante esforço para tornar os computadores pensar... máquinas com mentes, no sentido completo e literal".[94].	"O estudo das faculdades mentais através do uso de modelos computacionais". [91]; "O estudo dos cálculos que fazem, é possível perceber, raciocinar e agir". [93].
Sistemas que agem como os seres humanos	**Sistemas que agem racionalmente**
"A arte de criar máquinas que executam funções que exigem inteligência quando realizado por pessoas." [95]; "O estudo de como fazer computadores coisas em que, no momento, as pessoas são melhores." [97].	"Inteligência Computacional é o estudo do design de agentes inteligentes." [96]; "AI... está preocupado com o comportamento inteligente em artefatos." [98].

Fonte: [55].

Com isso, o conceito de agente tem forte influência tanto na Inteligência Artificial (IA), quanto nas áreas da ciência da computação, automação e controle, processamento de imagens e muito precisamente em robótica.

De acordo com [87], as pesquisas em sistemas multi-agentes, concentrou-se em estratégias de negociação e cooperação usadas por agentes autônomos que precisam competir por recursos escassos. Ainda conforme [87], o domínio e as capacidades de cooperação dos agentes é porque eles precisam acessar um recurso compartilhado ou ter várias metas sobrepostas.

Uma descrição geral que abrange agentes foi argumentado em [99] e [100] onde afirmam que:

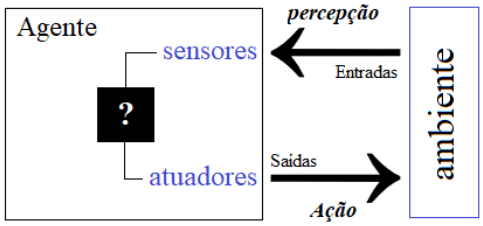
« “Um agente é uma organização real, ou virtual, inserida em determinado local onde o mesmo poderá executar algumas ações, estar capaz de receber e retratar relativamente este meio, podendo ainda interar-se com outros agentes desse local. Este agente expõe uma atitude autônoma, no qual consiste das consequências de seus argumentos, do conhecimento registrado e das interações com os demais agentes desse ambiente.” »

De acordo com [101], os agentes podem ser classificados em concordância com sua complexidade computacional. Agentes com complexidade computacional menor são chamados de Agentes Reativos, e aqueles com alta complexidade são chamados de Agentes Cognitivos ou Agentes Inteligentes.

3.2.1 Arquitetura de um Agente Inteligente

Conforme ilustrado na Figura 25, os agentes reativos estabelecem suas ações com base no conhecimento atual, desconsiderando o excesso do histórico de percepções [55]. Os agentes reativos não possuem memória de trabalho e não buscam elaborar suas ações futuras, ou seja, o modelo seguido por esse agente é baseado no funcionamento de estímulo-resposta.

Figura 25 – Interação de um agente reativo com o ambiente.

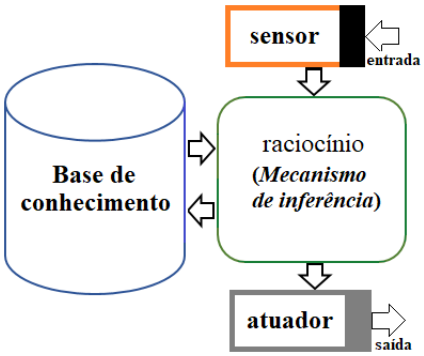


Fonte: Baseado em [55].

Estes agentes não possuem uma representação explicita do ambiente nem um modelo de comunicação de alto nível ou conhecimento de outros agentes quando em sociedade.

Um simples agente inteligente deve ser capaz, por exemplo, de conhecer e representar (estados, ações, etc.), incorporar novas percepções e atualizar representações internas do mundo [102]. Um agente baseado em conhecimento possui conhecimento sobre o mundo e sobre suas ações, raciocina para refletir uma maneira de atingir seus objetivos, ou seja, raciocina sobre suas próprias ações. De acordo com [103], os agentes autônomos baseados em sistemas computacionais possuem arquitetura similar ao da Figura 26, onde mostra um agente composto por uma base de conhecimento.

Figura 26 – Arquitetura básica de agentes baseados em computador.



Fonte: Baseado em [103].

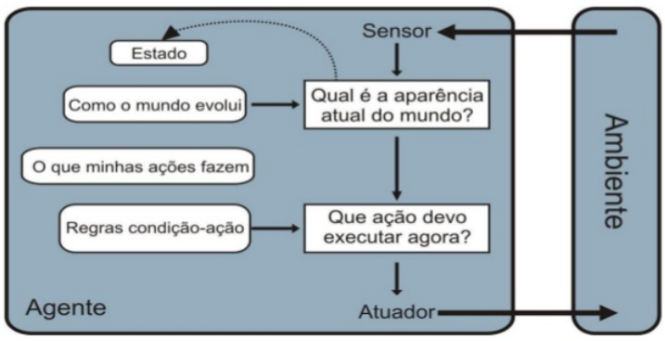
Segundo [104], um agente é construído a partir de técnicas e algoritmos que são usados por uma metodologia específica que irá definir se ele irá agir isoladamente ou em grupo, definindo os funcionamentos de seus sensores e estados internos.

3.2.2 Agentes cognitivos

Segundo [105], o conhecimento sobre o meio pode ser usado pelo agente para relacionar percepções e ações de maneira adequada, um exemplo é um agente cognitivo. Os Agentes Cognitivos como ilustrado na Figura 27, dispõe de uma reprodução conhecida do meio e dos demais agentes quando estão presentes em determinado ambiente.

Esses agentes são dotados de uma memória de trabalho, no qual podem raciocinar sobre ações adquiridas anteriormente para poder planejar suas futuras ações. Sendo assim, dispõem de um alto nível de complexidade computacional e caracterizam-se por possuir um comportamento inteligente [101] e são baseados no modelo de organização social humano.

Figura 27 – Representação de um agente cognitivo.

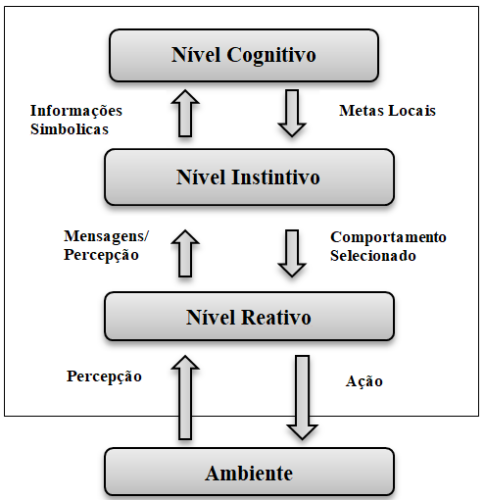


Fonte: [55].

3.3 O Agente Autônomo Concorrente

Um exemplo de agente cognitivo é o Agente Autônomo Concorrente no qual consiste de uma arquitetura autônoma de agentes para robôs móveis, provado ser eficiente em [106]; [38]; [107]; [108] e mais recentemente em [29]. A arquitetura do Agente Autônomo Concorrente é ilustrada na Figura 28

Figura 28 – Arquitetura geral do Agente Autônomo Concorrente.



Fonte: Do autor.

Essa arquitetura, foi inspirada no modelo genérico para agentes cognitivos proposto em [109] para uma arquitetura de AAI's utilizado na *RoboCup League*. Esse modelo consiste em uma arquitetura cognitiva geral utilizada para modelar agentes de qualquer natureza.

Inicialmente, numa atividade proposta na categoria de robôs simulados da *RoboCup'98*, o *UFSC-Team* retratou uma abordagem concorrente em uma arquitetura de agente cognitivo [38].

A princípio era a de se realizar percepção, ação, cooperação, comunicação, elaboração e tomada de decisão, aplicando a programação a nível concorrente. Essa elaboração inicial com atuação decisória do agente centralizado, apresentou alguns problemas de concordância entre o agente e o ambiente, e a resposta em tempo real ficou limitada [110].

3.4 Composição da arquitetura do AAC embarcado

A seguir é abordado uma breve descrição de cada um dos níveis do Agente Autônomo Concorrente para melhor entendimento do Capítulo 4, no qual será abordado cada nível individualmente relacionados ao hardware e onde cada um serão concretizados.

3.4.1 Nível cognitivo

O nível cognitivo é onde o planejamento acontece. Esse nível também possui um SBC (Sistema Baseado em Conhecimento), mas desta vez é usado como um mecanismo de pesquisa para o algoritmo de planejamento. Um modelo lógico do mundo é mantido e sua base de conhecimento é dividida em local e social, porque esse nível também é responsável por estabelecer comunicação com outros agentes.

3.4.2 Nível instintivo

Tradicionalmente, o nível instintivo possui um SBC que, com base no objetivo local enviado pelo nível cognitivo, nas informações sensoriais enviadas pelo reativo e em sua base de regras, infere qual comportamento deve ser ativo no nível reativo. Sua base de conhecimento possui uma coleção de bases de regras chamadas planos, e a base de regras atual é selecionada pelo nível cognitivo e seu procedimento de planejamento. O conhecimento desse nível é representado por uma linguagem específica de domínio de primeira ordem e a inferência é realizada por um encadeamento de regras [29].

Diferentemente de [29], neste trabalho, foi utilizada a lógica difusa como formalização de representação do conhecimento. Como esse nível é responsável, entre outras tarefas, por coordenar a seleção de comportamentos no nível reativo, um KBS (*Knowledge Based System*) difuso poderá mesclar comportamentos de maneira difusa ponderada.

3.4.3 Nível reativo

O nível reativo interage com o ambiente e executa o ciclo de percepção-ação do agente. Em [29], comportamentos reativos simples foram implementados: as direções Norte (N), Nordeste (NE), Leste (L), Sudeste (SE), Sul (S), Sudoeste (SO) e Oeste (O) constantes. Esses comportamentos foram efetuados por um controlador de velocidade cinemático.

Neste trabalho, o agente pode se mover em qualquer direção, portanto, existem infinitos comportamentos possíveis. Isso é consequência do uso de um sistema de inferência difusa no nível instintivo executado em um robô omnidirecional como será descrito mais adiante. O nível instintivo recebe as leituras do sensor realizadas pelo nível reativo e relata qual comportamento deve estar ativo.

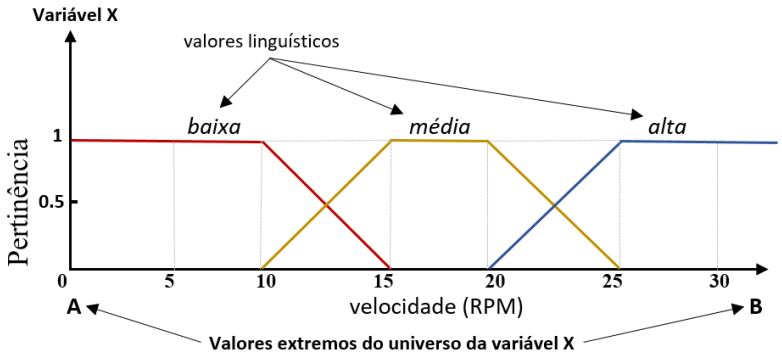
3.5 Lógica difusa

A Lógica *fuzzy* é uma extensão da lógica booleana, introduzida pelo Dr. Lofti Zadeh da Universidade da Califórnia/Berkeley no ano de 1965. Foi desenvolvida para expressar o conceito de verdade parcial, de maneira que se possam determinar valores entre o limite 0 “completamente falso” e 1 “completamente verdadeiro”. Ou seja, a lógica *fuzzy* lida com

a imprecisão e as relações complexas entre as variáveis envolvidas no processo. Segundo [111], essas variáveis podem ser representadas por termos linguísticos.

Esses termos, também conhecidos como conjuntos *fuzzy* são figurados por funções de pertinência, que definem o grau de possibilidade dos valores analíticos das variáveis dentro no seu termo linguístico correspondente. As funções de pertinência são comumente trapezoidais, triangulares ou gaussianas. A Figura 29 ilustra uma função de pertinência difusa típica.

Figura 29 – Exemplos de funções de pertinência difusa.



Fonte: Do autor.

O formato e a quantidade das funções de pertinência podem ser alteradas conforme o fenômeno e natureza avaliados pelo humano, pois quanto maior for o número de funções de pertinência, maior a precisão e exigência computacional. Segundo [112] o grau de superposição entre as funções de pertinência é outro fator que influencia na precisão, no qual recomenda-se um mínimo de 25% e máximo de 75%. Além disso, é essencial que as funções abranjam todo o universo de discurso.

Segundo [113], as teorias mais conhecidas para tratar essa imprecisão e da incerteza são, respectivamente, a teoria dos conjuntos e a teoria de probabilidades. Estas teorias, embora muito úteis, nem sempre conseguem captar a riqueza da informação fornecida por seres humanos.

Essas características permite a lógica *fuzzy* tratar de informações imprecisas, vagas ou ambíguas presentes na linguagem humana como: “pouco” e “muito”, afirmações comuns e perceptíveis do tipo: “muito veloz”, “pouco erro”, etc, e proferir modelos aproximados de pertinência de unidades em conjunto. É baseado em medidas como estas que, muitas vezes, os especialistas humanos expressam seus conhecimentos.

De acordo com [114] a lógica difusa pode ser vista como uma linguagem que possibilita traduzir estruturas sofisticadas da linguagem natural dentro de um formalismo matemático. A lógica *fuzzy* torna-se importante na medida em que o mundo não é constituído por fatos absolutamente verdadeiros ou falsos.

3.5.1 Sistema *Fuzzy* - baseado em regras

Segundo [112] a lógica *fuzzy* foi desenvolvida com base na necessidade de se obter um método capaz de expressar de maneira sistemática quantidades imprecisas. Ainda de acordo com [112] os controladores industriais elaborados a partir da lógica *fuzzy*, podem incorporar conhecimento experimental de operadores humanos já treinados, tornando a sua ação de controle tão boa quanto a deles (em geral melhor) e de forma consistente.

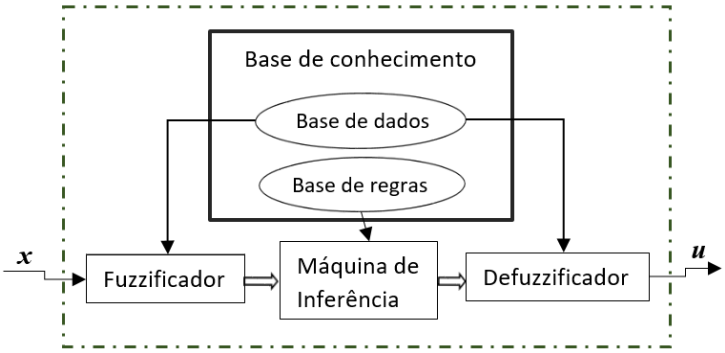
Um tipo de sistema *fuzzy* que possui uma aprovação mais ampla em aplicações industriais e de controle, chamado de controlador *fuzzy singleton* [115]. No campo da inteligência artificial (inteligência de máquina), existem várias maneiras de representar o conhecimento. Talvez a maneira mais comum de representar o conhecimento humano seja formar expressões de linguagem natural do tipo:

SE *premissa (antecedente),* ***ENTÃO*** *conclusão (consequente)*

Tipicamente expressa uma inferência tal que se conhece um fato (premissa, hipótese, antecedente), então podemos inferir, ou derivar, outro fato chamado conclusão (consequente), comumente referido como o formulário baseado em regras IF-THEN definindo dessa maneira a base de conhecimento de um dos modelos difuso proposto.

Um sistema lógico nebuloso FLC (do inglês, *Fuzzy Logic Control*) no qual o controle *fuzzy* é uma aplicação especial e uma extensão natural da teoria dos conjuntos nebulosos das relações entre conjuntos nebulosos e regras [116]. Esse sistema é caracterizado por quatro módulos: fuzzificador, mecanismo de inferência, base de conhecimento e defuzzificador. Uma representação esquemática em blocos do FLC é ilustrado na Figura 30.

Figura 30 – Estrutura básica de um controlador difuso.



Fonte: Baseado em [116].

Os principais elementos deste agente inteligente são: o fuzzificador, a base de conhecimentos difusa, a máquina de inferência e o defuzzificador. Os passos do raciocínio *fuzzy* (operações de inferência sobre as regras *fuzzy* do IF-THEN) são realizadas comparando-se

as variáveis de entrada com as funções de associação na parte antecedente para obter os valores de associação de cada rótulo linguístico (este passo é frequentemente chamado de fuzzificação). Usando regras *fuzzy* IF-THEN, o motor de inferência converte a entrada difusa na saída difusa. Por fim, a defuzzificação é responsável por agregar os consequentes qualificados para produzir uma saída nítida.

Segundo [117] o suporte de um conjunto difuso A no universo do discurso U é um conjunto nítido que contém todos os elementos de U que possuem valores de associação diferentes de zero em A. A Equação 3.1 define como:

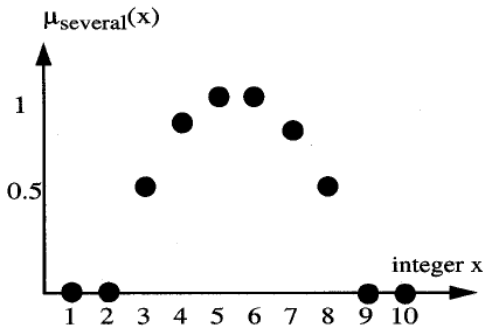
$$support(A) = x \in U | \mu A(x) > 0. \tag{3.1}$$

Baseando-se na Figura 31, temos que o suporte do conjunto difuso é formado por:

- $support(A)$: Define o suporte do conjunto difuso A;
- $\mu_s(x)$: Conjunto de números inteiros (3, 4, 5, 6, 7, 8).

Se o suporte de um conjunto difuso estiver vazio, ele será assim chamado. Um *singleton* difuso é um conjunto difuso cujo suporte é um ponto único em U [117].

Figura 31 – Função de associação para o conjunto difuso.



Fonte: [117].

Devido aos seus benefícios, como regras próximas da linguagem natural, fácil manutenção, simplicidade estrutural, os modelos baseados em sistemas *fuzzy* são validados com maior precisão e a confiança destes modelos tendem ao crescimento [118].

3.5.2 Sistema de inferência difusa

Como mencionado na secção anterior, de acordo com [117], a lógica difusa é usada para representar informações imprecisas. Aplica o conceito de associação para denotar quanto um determinado número (um valor da variável linguística) pertence a um dado

conjunto difuso. Dessa forma, ainda segundo os autores, uma pertinência teste como IF (exemplo: temperatura IS LOW), na verdade retorna um número, que denota a associação do valor assumido pela variável temperatura linguística ao conjunto *fuzzy* LOW. Estes testes podem ser combinados por conjunções e disjunções difusas na premissa de uma regra *fuzzy*, que são realizadas por operações chamadas *t-norms* e *t-conorms*, respectivamente. Exemplos de t-normas são o mínimo e o produto, operações e t-conorms comuns são máximos e a soma probabilística.

O consequente de uma regra *fuzzy* é um ou mais conjuntos *fuzzy*, que são ponderados pela combinação (conjunções e disjunções) de testes de variáveis linguísticas. Os pesos são aplicados através de uma operação de implicação *fuzzy*, que também é uma norma t. Este processo de atribuir um valor numérico (não *fuzzy*) a variáveis linguísticas numa premissa e obter um conjunto difuso como resultado é chamado de fuzzificação [117].

Quando uma coleção de regras difusas é reunida, obtém-se uma base de regras difusa. O resultado final de um ciclo de inferência através de uma base de regras é a soma ponderada de uma propriedade geométrica (área, centróide, etc.) dos conjuntos difusos nos consequentes de todas as regras; os pesos são a combinação (conjunções e disjunções) das premissas. Este é o passo de defuzzificação.

3.6 Rede dos barramentos de comunicação

3.6.1 O Protocolo CAN

O barramento de comunicação serial CAN (do inglês, *Controller Area Network*) foi preliminarmente elaborado pelo fornecedor de sistemas automotivos, o alemão Robert Bosch em meados da década de 1980 para funcionalidades de transmissão de mensagens com capacidade multi-mestre, sendo que todos os nós podem solicitar acesso ao barramento, como uma maneira de permitir a comunicação serial robusta a uma taxa máxima de sinalização de 1Mbps. O protocolo CAN também comporta o conceito de *multicast*, ou seja, permite que uma mensagem seja transmitida a um conjunto de receptores de forma simultânea.

Desde quando foi criado, o protocolo CAN ganhou popularidade em aplicações de automação industrial e automotiva, no qual foi incorporado a um sistema de controle central. Projetado para permitir que microcontroladores e dispositivos se comuniquem entre si dentro de um veículo sem um computador *host*, este padrão de barramento inclui seu próprio protocolo de mensagens para comunicações entre nós de uma rede em sistemas embarcados. Ao contrário de uma rede tradicional, como por exemplo a rede Ethernet, o CAN não envia grandes blocos de dados ponto-a-ponto (de um nó A para um nó B) sob a supervisão de um mestre de barramento central. Em uma rede CAN, muitas mensagens

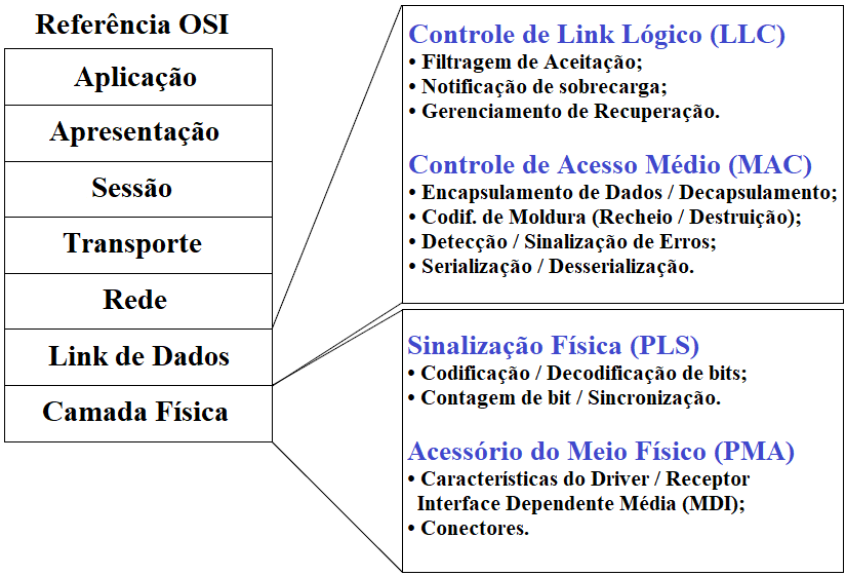
curtas contendo um máximo de 8 *bytes* de informação útil, como, temperatura e RPM por exemplo, são transmitidas para toda a rede, o que fornece consistência de dados em todos os nós do sistema.

As mensagens numa rede CAN não são endereçadas aos destinatários no sentido convencional, em vez disso, são transmitidas mensagens com um determinado identificador. Sendo assim, um nó emissor envia uma mensagem a todos os outros nós do barramento e cada um decide com base no seu identificador recebido, se deve ou não executar o que está contido na mensagem. Uma característica importante desse identificador é a de definir a prioridade da mensagem ao competir com outras pelo acesso ao barramento.

Desde então, o protocolo CAN passou a ser utilizado amplamente no contexto de automação industrial, até que, em 1993, se tornou um padrão interncaional: o ISO (do inglês, *International Standards Organization*) 11898 [119]. Além de consistência na transmissão de informações, outras principais características no qual levou ao protocolo a se tornar padrão internacional foi a alta imunidade à interferência elétrica e a capacidade de auto diagnosticar e reparar erros de dados.

O protocolo de comunicação CAN, ISO-11898: 2003, descreve como as informações são transmitidas entre os dispositivos em uma rede e estão de acordo com o modelo **OSI** (do inglês, *Open Systems Interconnection*) que é definido em termos de camadas. A comunicação real entre dispositivos conectados pelo meio físico é definida pela camada física do modelo. A arquitetura ISO-11898 define as duas camadas inferiores do modelo **OSI/ISO** de sete camadas como a camada física e a camada de enlace de dados ilustrado na Figura 32. A camada física trata de especificações do meio físico.

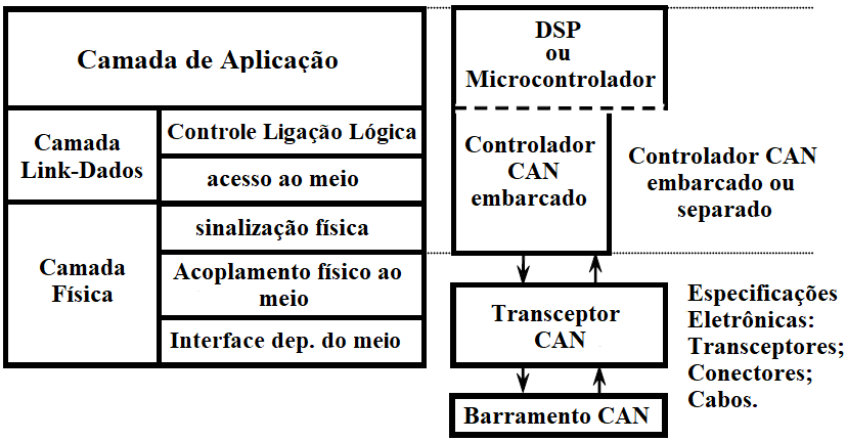
Figura 32 – Modelo de referência **ISO/OSI**.



Fonte: Baseado em [120].

Em termos gerais, a camada do link de dados é responsável pela manutenção de um enlace lógico entre os nós. A Figura 33 mostra as camadas **OSI** acima referidas e os elementos da rede CAN que às implementam [119].

Figura 33 – Arquitetura padrão em camadas ISO-11898.

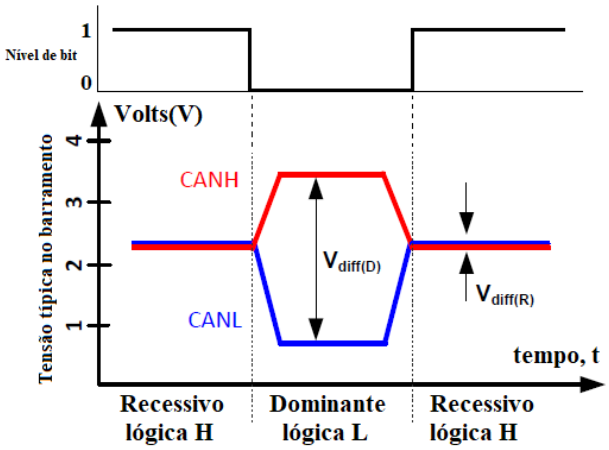


Fonte: [119].

O restante da especificação da camada física, isto é, a conexão com o Meio Físico **PMA** (do inglês, *Physical Medium Attachment*) e a Interface Dependente do Meio **MDI** (do inglês, *Medium Dependent Interface*), são definidos no padrão ISO-11898, que também engloba a especificação CAN descrita acima [121].

Uma característica fundamental da CAN mostrada na Figura 34, onde é ilustrado o sinal diferencial elétrico da rede é o estado lógico oposto entre o barramento e a saída do receptor e da entrada do driver. Normalmente, um nível lógico alto é associado a 1, e um nível lógico baixo é associado a 0, mas não tão em um barramento CAN.

Figura 34 – Estados do barramento CAN (representação de bit físico).



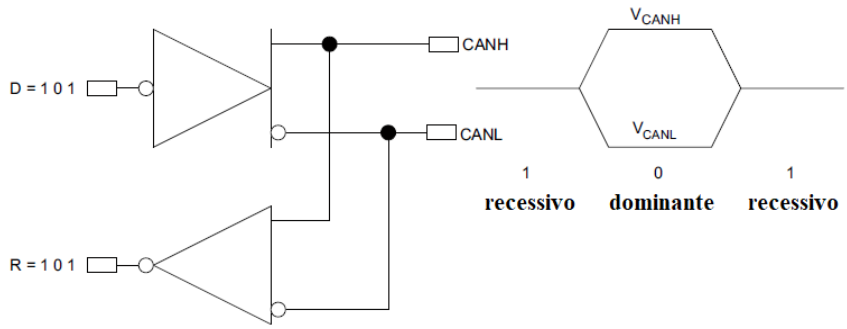
Fonte: Adaptado de [122].

No caso da camada física, o protocolo CAN não define um meio de transmissão, apenas que os sinais devem ser transmitidos manipulando uma codificação diferencial, o que representa que os valores lógicos no barramento serão codificados de decisão com a diferença de tensão entre duas linhas, CANH (nível lógico alto - $HIGH$) classificado como recessivo e CANL (nível lógico baixo - LOW) classificado como dominante. Em um barramento CAN se um nó tentar estabelecer o nível lógico alto sobre o barramento, quando o outro, ao mesmo tempo tentar estabelecer o nível lógico baixo, o último predominará [123].

Sendo assim, mensagens com mais zeros (ou seja, mais bits dominantes) no início, possuem maior prioridade [124]. Essa característica é observada no circuito da Figura 35.

A permissão de acesso ao barramento é direcionada a eventos e acontece de forma arbitrária. Se dois nós buscarem ocupar o barramento simultaneamente, o acesso será implementado com uma arbitragem bit a bit não destrutiva, ou seja, que a arbitragem vencedora do nó continua com a mensagem, sem que a mesma seja destruída ou corrompida por outro nó. A função de preferência às mensagens no identificador é um recurso do CAN que o torna particularmente atraente para uso em um ambiente de controle em tempo real [124].

Figura 35 – A lógica invertida de um barramento CAN.

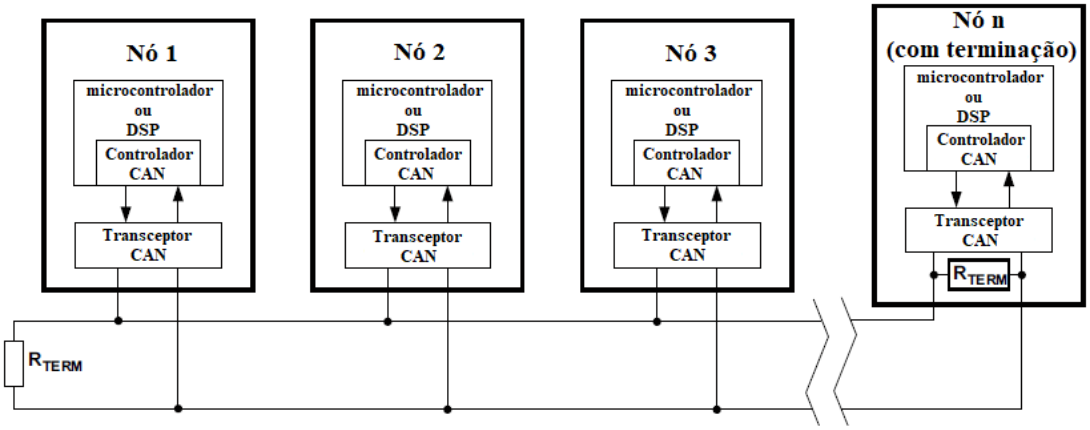


Fonte: Baseado em [124].

Um identificador que representa inteiramente em zeros é a mensagem de prevalência mais alta em uma rede porque preserva o barramento dominante por mais tempo. Desse modo, se dois nós iniciarem a transmitir ao mesmo tempo, o nó que envia um último bit identificador como zero (dominante) quando os outros nós enviam um (recessivo) retém o controle do barramento CAN e continua a integrar sua mensagem. Em um barramento CAN, quanto menor o número do identificador da mensagem binária, maior a sua prioridade e um bit dominante sempre sobrescreve um bit recessivo.

A taxa de transmissão de dados numa rede CAN quando se utiliza transceptores, pode chegar a 1Mbps e são projetadas para mensagens curtas com comprimento de dados de até 8 bytes. A topologia deste barramento é ilustrada na Figura 36.

Figura 36 – Barramento CAN típico.



Fonte: Adaptado de [122].

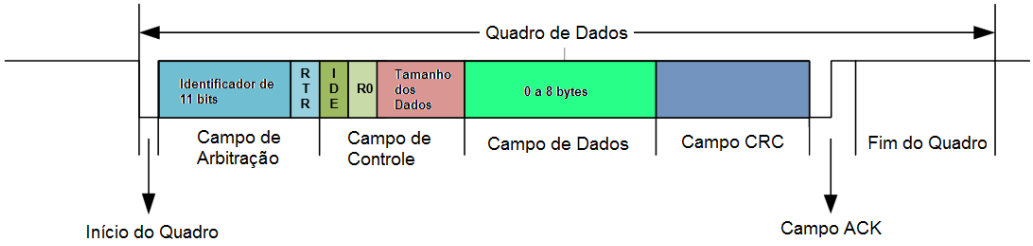
Nesta ilustração é ressaltado o papel do transceptor como o elemento que é fisicamente conectado ao barramento. Além disso, fica explícito que cada nó deve possuir um transceptor. O controlador CAN é mais frequentemente integrado ao hardware dos nós da rede, enquanto que os transceptores são geralmente externos a estes últimos.

Qualquer nó ligado ao barramento, verificando que este último se encontra livre, pode iniciar a transmissão das mensagens. Estas mensagens são formadas em pacotes de dados intitulado quadros. O protocolo CAN é composto por quatro tipos de quadros:

- Quadro de dados, utilizado para transferir dados entre nós;
- Quadro remoto, utilizado para os nós fazerem requisições de dados;
- Quadro de erro, que pode ser transmitido por qualquer nó quando um erro é detectado no barramento;
- Quadro de sobrecarga, utilizado entre dois quadros de dados ou remotos para prover um atraso adicional.

O quadro de dados é ilustrado na Figura 37.

Figura 37 – Quadro de dados do protocolo CAN.



Fonte: [123].

O campo de arbitração carrega o significado da mensagem, e cada nó do barramento decide, ao ler este campo, se deve aceitar essa mensagem. Também é utilizado para arbitrar o acesso ao meio: em caso de colisão, o nó cuja mensagem possui o menor identificador de 11 bits tem a prioridade. O campo de controle contém o tamanho da mensagem, o campo de dados contém os dados da mensagem e o campo **CRC** (do inglês, *Cyclic Redundancy Check*) é utilizado pelos nós do barramento para verificar erros no quadro [123].

Ainda se referindo à Figura 37, a camada de enlace divide suas responsabilidades entre a subcamada de controle da ligação lógica LLC (do inglês, *Logical Link Control*) e a subcamada de controle de acesso ao meio MAC (do inglês, *Medium Access Control*). A camada inferior LLC adota o recebimento de mensagens através de um artifício de filtragem, notificação de sobrecarga e gestão de restauração. Já as funções de detecção de erros, encapsulamento de dados em pacotes, validação e gestão de acesso ao meio são atribuições da subcamada MAC [119].

3.7 Projetos desenvolvidos com o AAC

Em [29] uma rede de três microcontroladores foi projetada para embarcar o AAC no robô omnidirecional AxeBot, ao qual esse projeto se baseia para embarque do agente na nova arquitetura de hardware desenvolvida. A rede foi concebida mimetizando a rede funcional do Agente Autônomo Concorrente. Os três níveis deste último foram embarcados em cada nó da rede. O nível reativo, consistia em um controlador cinemático de posição baseado no modelo do robô omnidirecional AxeBot, o mesmo foi embarcado no microcontrolador PSoC® 5LP.

O nível instintivo foi embarcado em um módulo microcontrolador baseado no NXP LPC1768. Um sistema baseado em conhecimento com uma base de regras em LPO (Lógica de Primeira Ordem) foi implementado neste nível com o intuito de coordenar a seleção de comportamentos no nível reativo. Já o nível cognitivo, foi implementado em um módulo baseado DIL/NetPC 2486. Nesse projeto, foi implementado também um sistema baseado em conhecimento com um motor de inferência que pôde utilizar LPO (Lógica de Primeira Ordem), quanto LTP (Lógica Temporal Proposicional).

Em [110], o nível reativo do AAC foi embarcado no robô omnidirecional AxeBot. Neste caso, o nível reativo foi composto por um módulo responsável pela implementação da instrumentação do sistema, encapsulando um sistema de controle em cascata para controle da posição do centro de massa do robô.

A arquitetura do Agente Autônomo Concorrente possui a característica de poder ser implementado em qualquer sistema robótico, para isso, é apenas necessário a criação de bases de regras para os níveis instintivo e cognitivo, além da modificação da sua camada reativa, levando-se em conta a característica de hardware de cada robô.

4 Desenvolvimento

Apresenta-se neste Capítulo uma nova arquitetura de hardware reconfigurável composta por três avançadas plataformas de desenvolvimento nas quais utilizou-se como base microcontroladores ARM®. O propósito dessa nova arquitetura, é estender opções para desenvolvimento de estudos e pesquisas em robótica móvel. Na questão da prototipagem, descrevem-se alguns métodos que permite que o robô desenvolvido evidencie as caraterísticas adaptadas às competições da liga *RoboCup* em causa, nomeadamente às estruturas modular mecânica e eletrônica inseridas na arquitetura geral de hardware.

Após o desenvolvimento, a arquitetura foi configurada apresentando um sistema para execução do agente cognitivo, baseando-se no projeto desenvolvido em [29]. Diferentemente do projeto anterior, no presente projeto utilizou-se somente barramento CAN composto por transceptores, no qual compõem o núcleo de comunicação entre os níveis a saber: cognitivo (planejamento), instintivo (coordenação) e reativo (percepção-ação). A rede embarcada foi concebida para fazer o agente autônomo funcionar como um sistema compartilhado em que um comportamento inteligente surge da interação entre os três níveis do agente de forma confiável e com uma maior velocidade de comunicação entre seus barramentos através do protocolo CAN.

4.1 O Robô AxeBot II - Arquitetura Geral de Hardware

Além da modernização do robô *AxeBot*, tanto na estrutura mecânica quanto na eletrônica, a proposta desse trabalho é possuir uma PCB com opções multiplataforma reconfiguráveis e substituíveis, as quais permitirá o desenvolvimento de aplicações de estudos e pesquisas no âmbito da robótica móvel. O estudo para desenvolvimento do robô foi realizado para três aplicações de interesse, tais como:

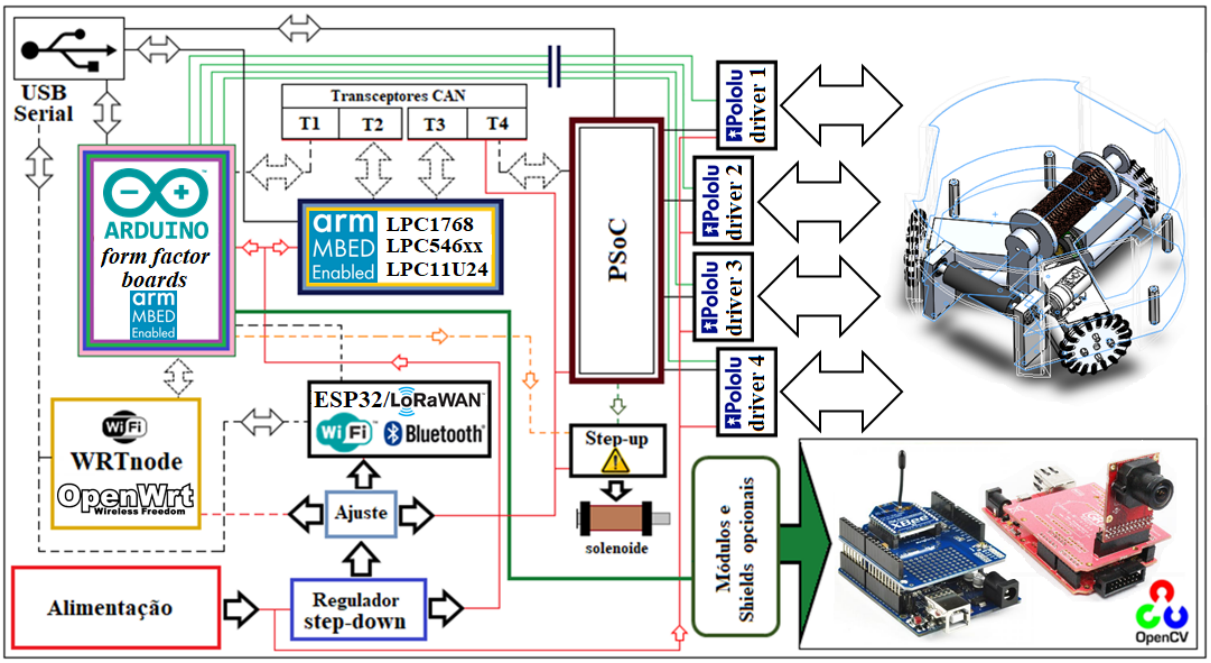
- Robótica móvel omnidirecional;
- Robótica móvel diferencial;
- Manipuladores articulados.

A primeira aplicação, ao qual trata de robôs móveis omnidirecionais, em parte, diz respeito às competições do futebol de robôs, a qual é foco da contextualização à proposta realizada nesta dissertação, mais especificamente àquelas referentes a robôs da liga RoboCup *Small Size* F180. As duas últimas ficam como aplicações opcionais.

4.1.1 Diagrama funcional

O diagrama da Figura 38 apresenta a arquitetura robótica proposta neste trabalho, bem como sua divisão em módulos e as interfaces utilizadas para comunicação das plataformas para configuração do agente em questão. No diagrama, é possível perceber a existência da possibilidade de utilização de plataformas e extensores opcionais para outras aplicações de pesquisas, como por exemplo, módulo de comunicação via rádio e plataformas com suporte a câmeras.

Figura 38 – Diagrama em blocos do Hardware.



Fonte: Do autor.

Nessa arquitetura modular é possível utilizar 3 plataformas principais, dois módulos de comunicação, drivers dos motores, no qual podem ser configurados robôs diferenciais ou omnidirecionais. Alternativamente, é possível também configurar manipuladores articulados. Além disso, a PCB permite a expansão de suas funcionalidades com a utilização de diversos extensores (do inglês, *shields*), vastamente disponíveis comercialmente.

Devido ao intenso crescimento de plataformas com o fator de forma Arduino, estima-se que é possível a utilização de dezenas de plataformas na arquitetura de hardware desenvolvida nesse trabalho. Essas plataformas apresentam-se como uma alternativa essencial à experimentação com robôs reais, já que concedem a realização de experimentos em um ambiente organizado.

A Tabela 2 a seguir, lista as três plataformas utilizadas para embarcar o AAC no robô omnidirecional, configuradas com transceptores em seus 2 barramentos CAN.

Tabela 2 – Plataformas utilizadas.

Plataforma	**Tipo de arquitetura**
GR-PEACH RZ/A1H	ARM® Cortex®-A9;
IoT/FF-LPC546xx	ARM® Cortex®-M4;
PSoC® 5L	ARM® Cortex®-M3.

Fonte: Do autor.

Duas das plataformas listadas, a (LTeK® LPC546xx e GR-PEACH), das três utilizadas para embarcar o AAC possui suporte para ambiente de programação *mbed* ou *mbed* OS, exceto a PSoC® (sistema em um chip programável). A plataforma PSoC® contém uma arquitetura composta por funções digitais, analógicas e mistas, além de controladores, algorítmo de leitura das variáveis analógicas de acionamento dos drivers dos motores e dos encoders para leitura das velocidades.

Uma das justificativas para utilização de microcontroladores ARM® (do inglês, *Advanced RISC Machine*) para embarque do AAC, é que são baseados em arquitetura RISC (do inglês, *Reduced Instruction Set Computer*) que visam Unidades de Controle (μC) mais simples, rápidas e de baixo custo. Essa arquitetura geralmente optam por instruções mais simples, com pouca complexidade e simplicidade em endereçamento. Além disso, tem seu projeto focado em baixo consumo de energia e apresentam tamanho reduzido. Importante salientar, que a plataforma GR-PEACH possui fator de forma compatível com o Arduino UNO, na qual poderá ser substituída por outras com mesmo fator de forma (Arduino *pin header*), características que culminaram para escolha da plataforma.

A evolução da arquitetura desde o seu surgimento trouxe diversas capacidades ao processador ARM®, tornando-o ainda mais utilizado e fazendo dele objeto de interesse em projetos acadêmicos cada vez mais elaborados [125]. Com isso, é possível perceber pelas decisões de projeto, a simplicidade inicial dos processadores, seguida da adição de funcionalidades opcionais oferecidas pelas extensões e pela evolução constante da arquitetura. Neste Capítulo, também será abordado em detalhes sobre as plataformas estabelecidas em microcontroladores ARM® utilizadas no projeto, na qual fazem parte três plataformas que serão configuradas com seus devidos níveis do agente autônomo embarcado em questão listadas na Tabela 3, ou seja:

Tabela 3 – Plataformas e devidos níveis.

Plataforma	**nível do sistema embarcado**
GR-PEACH RZ/A1H	Cognitivo;
IoT/FF-LPC546xx	Instintivo;
PSoC® 5L	Reativo.

Fonte: Do autor.

4.2 Etapas de Comunicação

Além das plataformas utilizadas para embarcar o agente as quais serão detalhadas nas próximas secções, na arquitetura de hardware proposta é possível utilizar mais dois módulos para etapa de comunicação opcionais, os quais são descritos na Tabela 4.

Tabela 4 – Plataformas de comunicação.

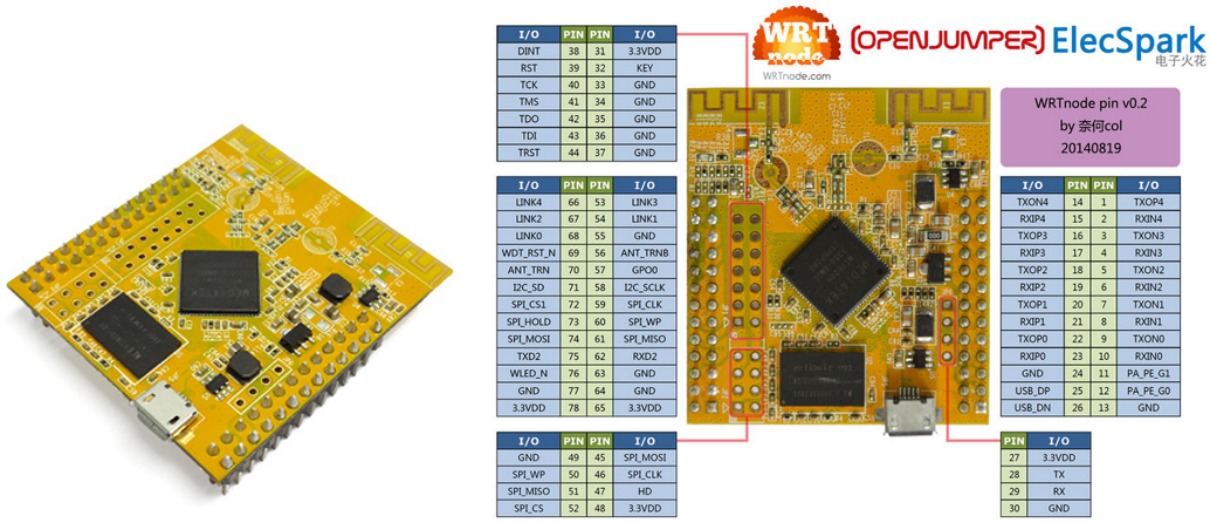
Plataforma	**Interface**
WRTnode	Wi-Fi / OpenWRT *Development*;
ESP-WROOM-32	Wi-Fi e Bluetooth.

Fonte: Do autor.

4.2.1 WRTnode - Interfaces Wi-Fi e IP

O WRTnode, é uma plataforma de desenvolvimento de hardware baseado no WiFi (do inglês, *Wireless Fidelity*) $AP-SoC$ de código aberto com dimensões de 45mm x 50mm que permite incorporar periféricos como sensores e câmera com IP (*do inglês, Internet Protocol*) e WiFi. A plataforma WRTnode é ilustrada na Figura 39.

Figura 39 – Placa WRTnode e diagrama dos terminais.



Fonte: [126].

Nesse módulo, é possível utilizar o *OpenWRT Project* que é um sistema operacional Linux direcionado a dispositivos embarcados. Esse firmware fornece um sistema de arquivos gerenciável, no qual permite personalizar o dispositivo através do uso de pacotes para se adequar a diversos aplicativos, como por exemplo o *OpenCV*[5], sistema operacional em

[5] OpenCV (*Open Source Computer Vision Library*). Originalmente, desenvolvida pela Intel, em 2000, o *OpenCV*

tempo real *RTOS* (do inglês, *Real Time Operating System*) para cooperação entre robôs, dentre outros. As especificações do módulo são vistas na Tabela 5.

Tabela 5 – Características do WRTnode.

Recurso	**Características**
Processador	*Mediatek* MT7620N 580MHz MIPS (MIPS24KEc)
Memória RAM	64MB DDR2
Memória Flash	SPI de 16 MB
Conectividade	Wi-Fi 2T2R 802.11n 2.4 GHz até 300Mbps
Portas	composto por: 23GPIO's, JTAG, SPI, UART Lite, host USB2.0

Fonte: Do autor.

Soluções *IoT* (do inglês, *Internet of Things*) têm sido desenvolvidas utilizando alguns tipos de plataformas, muitas vezes com um sistema WiFi externo a placa de controle. Devido a isso, o *OpenWrt* é a estrutura para projetos de aplicativos sem precisar criar um firmware completo em torno da plataforma.

4.2.2 ESP32 - Interface WiFi e SPI

A outra plataforma utilizada no presente projeto é a ESP32, na qual fará parte da etapa de comunicação. Essa plataforma tem como principal característica o poder de comunicação wireless, já integrado à placa e poderá ser configurada para executar as regras do agente, conforme mostrado na Figura 85.

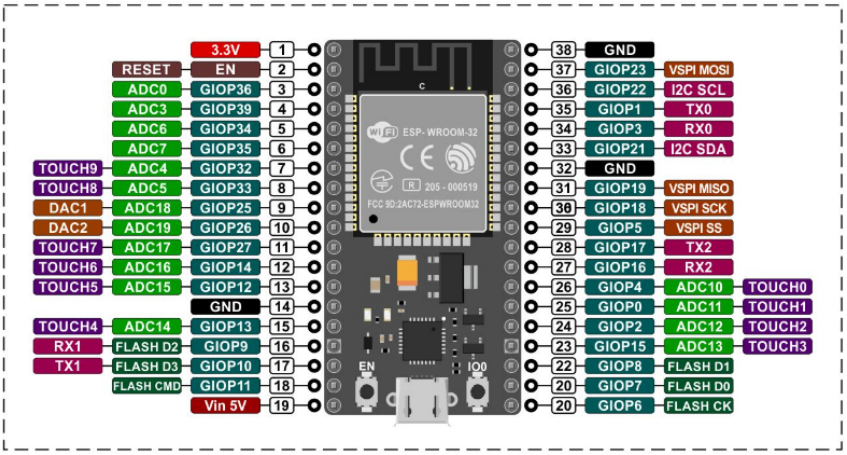
O ESP32 é o sucessor do ESP8266, lançado em 2014 pela empresa chinesa *Espressif*. Esse chip é um microcontrolador que possui tanto interface WiFi (protocolo 802.11 b/g/n) a 2.4GHz e *bluetooth* de 2.5GHz quanto interface periférica serial (SPI), o que o torna útil para realizar uma conexão sem fio entre outros dispositivos remotos que também possuam protocolo WiFi. Além disso, permite a comunicação bluetooth para configurar a rede WiFi, eliminando a necessidade de utilizar cabos. A utilização da comunicação *Bluetooth* para configurar a rede WiFi apresenta algumas vantagens, tais como:

- Protocolo aberto e seguro;
- É possível descobrir outros dispositivos próximos de mesma tecnologia;
- Possibilidades de comunicação quando a rede WiFi está inativa;
- Possibilidades de integração com *smartphones*.

é uma biblioteca de programação multiplataforma, totalmente livre ao uso acadêmico e comercial, para o desenvolvimento de aplicativos na área de visão computacional, vinculado ao modelo da licença de código aberto BSD (do inglês, *Berkeley Software Distribution*) (um sistema derivado do Unix) Intel.

A funcionalidade dotada pelo módulo composto pelo ESP32-WROOM com flash integrado, torna o dispositivo de extrema importância, útil considerável no contexto da internet das coisas *IoT* e robótica, pois facilita-rá a conectividade, a troca de dados e o controle remoto de outros dispositivos, solicitando o mínimo de infra-estrutura de redes sem fio, facilmente conseguida com os computadores. A Figura 40 ilustra o componente e a configuração de seus terminais.

Figura 40 – Plataforma ESP32-WROOM e diagrama dos terminais.



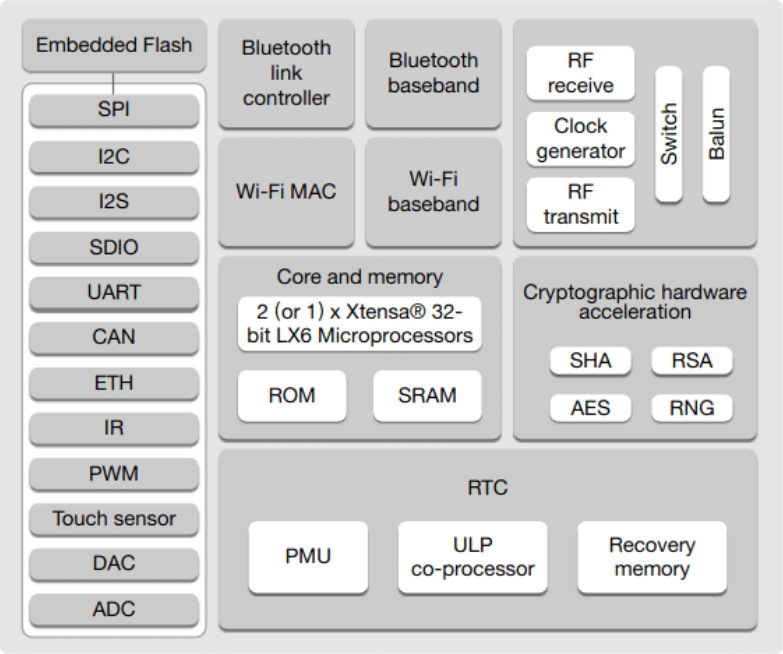
Fonte: [127].

As principais características do ESP32-WROOM estarão disponíveis no *SoC*, resumidas na lista a seguir:

- Microprocessador *dual − core* da *Tensilica* LX6 240 MHz e arquitetura RISC de 16bits;
- 520kB de memória SRAM e 16 MB de memória $flash$ na placa;
- Tensão de operação: 2.2 a 3.3V;
- O protocolo WiFi pode ser usado com TCP ou UDP;
- Possui suporte aos protocolos WiFi (IEEE 802.11 g/b/n), I2C, SPI e UART;
- 32 terminais GPIO, que podem ser multiplexados com outras funções (UART, I2C, PWM, entre outras);
- Variedades de periféricos: *Touch* capacitivo, ADCs, DACs, I2C. UART, SPI, SDIO, I2S, RMII, PWM.

A Figura 41 ilustra a arquitetura organizacional interna da plataforma, representada por diagrama em blocos.

Figura 41 – Diagrama em blocos do ESP32-WROOM.



Fonte: [128].

O interesse despertado pelo ESP32 resultou na elaboração de uma biblioteca para o chip em códigos com funções muito similares às das tradicionais plataformas Arduino. O download dessas bibliotecas pode ser feito através do repositório que a *Espressif* mantém no GitHub, disponível em [129].

Além das plataformas citados, adicionalmente, a PCB é composta por módulos listados na Tabela 6 a seguir.

Tabela 6 – Módulos utilizados.

Módulo	**Recurso**
LM2596 e MP1584EN	Ajuste DC/DC (*step-down*) 16-6V/6-3.3V
Conversor *boost*	Ajuste DC/DC (*step-up*) 3-5V para 300-1200V
TJA1050	Transceptor CAN de 5V
Driver 18V17	Ponte H (18 volts e 17A)
zigbee S2C	Módulos de comunicação via rádio
Bateria Li-Po (*íon*-polímero)	Tensão: 11.1V, corrente: 1500mAh

Fonte: Do autor.

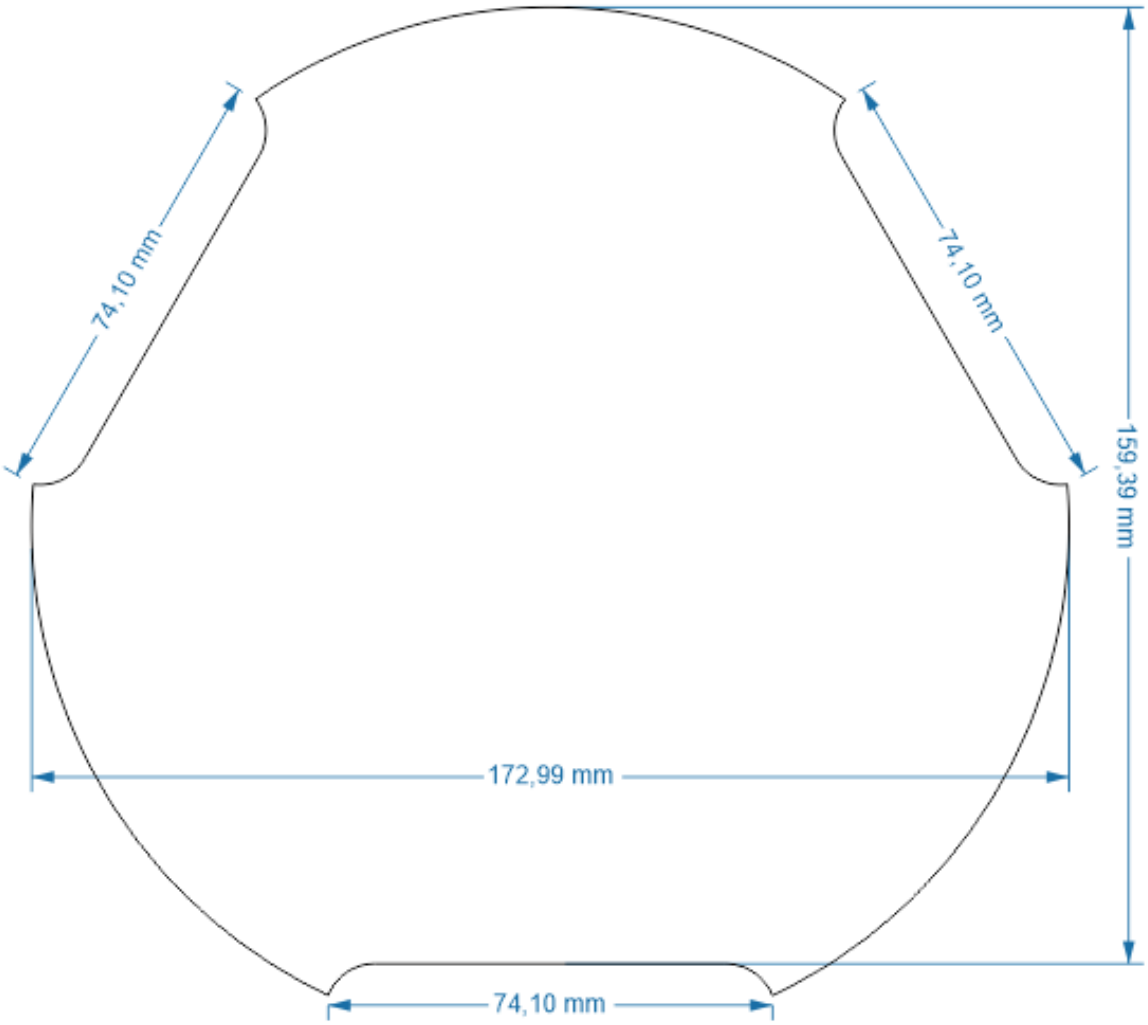
4.2.3 Dimensões e Circuito Eletrônico da PCB

Inicialmente as dimensões e formato da PCB (do inglês, *Printed Circuit Board*) foi desenvolvida no software AUTOCAD 2D e depois importada para o ambiente de

desenvolvimento do Proteus ARES em formato .DXF (do inglês, *Drawing Exchange Format*), que é um arquivo de intercâmbio para modelos de CAD (do inglês, *Computer-Aided Design*), definindo assim a borda da PCB onde corresponderá o espaço delimitado para inserção dos componentes.

As dimensões da PCB tem aproximadamente 173mm de diâmetro onde possui três "cavas" de 74.10mm onde ficam os espaços paralelos das três rodas omnidirecionais de 50mm de diâmetro. A Figura 42 ilustra essas dimensões na área de trabalho do software.

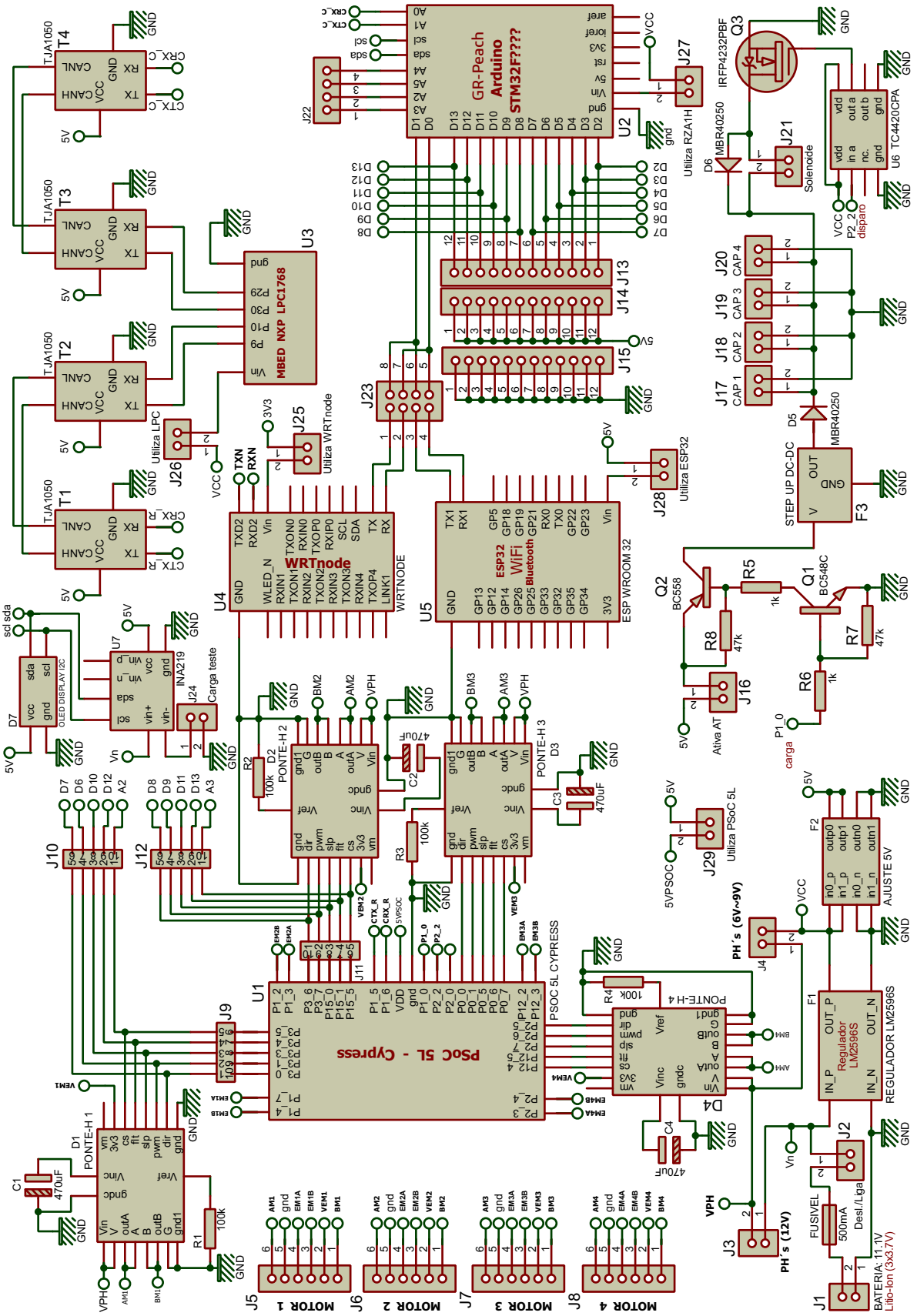
Figura 42 – Dimensões da PCB.



Fonte: Do autor.

O software Proteus ISIS versão 8.6 SP2 foi usado para montagem do circuito ilustrado na Figura 43 a seguir.

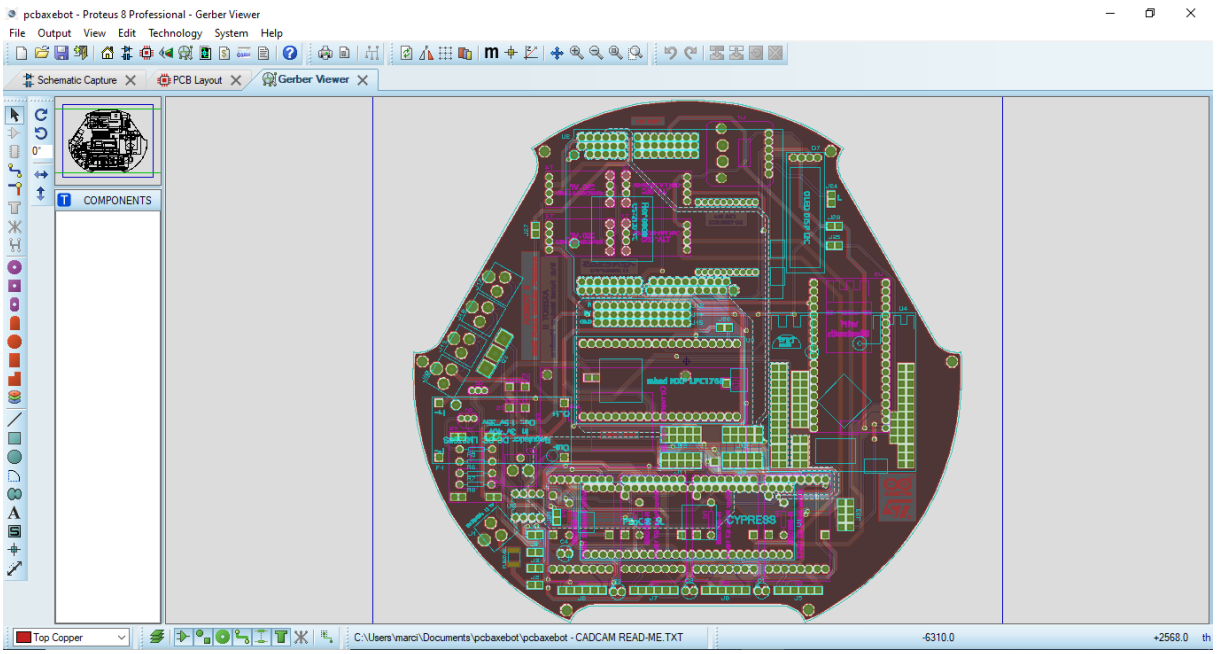
Figura 43 – Circuito geral da PCB no Proteus ISIS.



Fonte: Do autor.

O Proteus ARES *Design Suite* foi o software usado na montagem dos módulos do projeto da PCB. O Recurso utilizado para o roteamento das trilhas foi o *autorouter*, no qual consiste num roteamento baseado em formato padrão. O roteador do ARES utiliza algoritmos avançados de redução de conflitos baseados em custo, ou seja, é realizado o traçado no menor caminho possível, comprovados para maximizar as taxas de conclusão mesmo nas placas mais densamente compactadas, ou seja, com placas com mais camadas. A Figura 44 ilustra o circuito final na área de trabalho no software Proteus ARES em formato Gerber RS274X.

Figura 44 – PCB em formato Gerber.



Fonte: Do autor.

Após a configuração e execução desse procedimento para o roteamento da PCB em quatro camadas (*top layer*, *inner1*, *inner2* e *botton Layer*) um ajuste manual foi necessário para retirar trilhas desnecessárias e auxiliar no posicionamento tanto dos módulos quanto os componentes discretos, como, resistores, transistores, conectores e circuitos integrados.

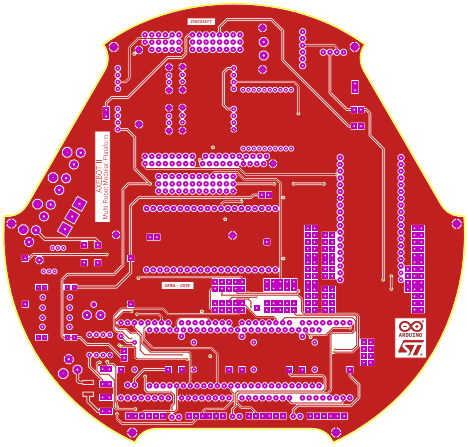
No desenvolvimento da PCB foi utilizado o sistema de Regras de Design (do inglês, *Design Rules*), que impõe o roteamento de trilhas compatíveis com o projeto, verificação e correção de violações dos pontos críticos antes da geração dos arquivos Gerber, no formato RS274X.

No roteamento levou-se em consideração padrões importantes adotados com relação a distância (*Clearence*) da malha de terra para as trilhas de sinal, devido a PCB possuir um circuito em que trabalhará com valores de tensões distintas, ou seja, um circuito em que a tensão pode exceder os 200V, por utilizar um circuito de carregamento de capacitores

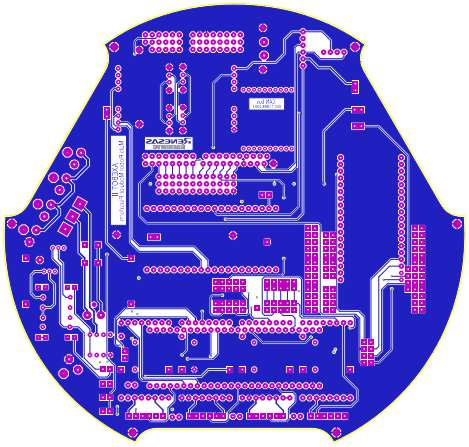
e por circular uma corrente próximo dos 30A (circuito de disparo da solenoide) e baixas tensões de sinal (3.3V e 5V), provenientes dos microcontroladores e outros módulos. A Figura 45 ilustra as camadas da PCB.

Figura 45 – Camadas superior, inferior e internas.

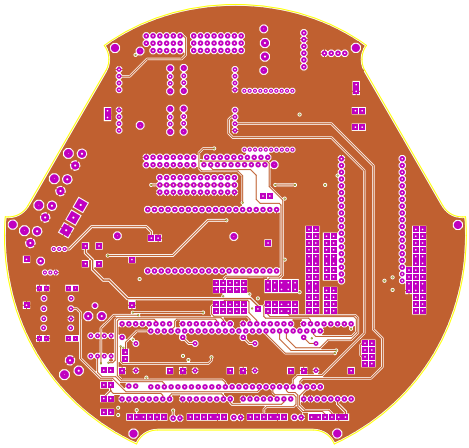
(a) Camada superior.



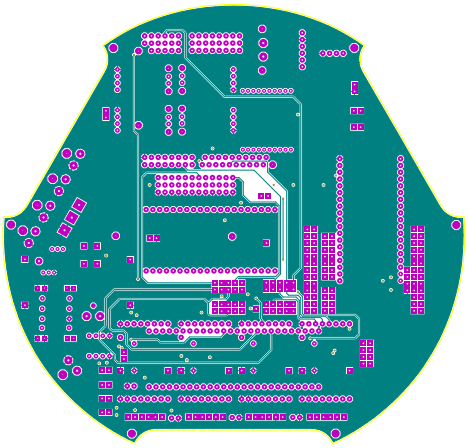
(b) Camada inferior.



(c) Camada superior interna.



(d) Camada inferior interna.



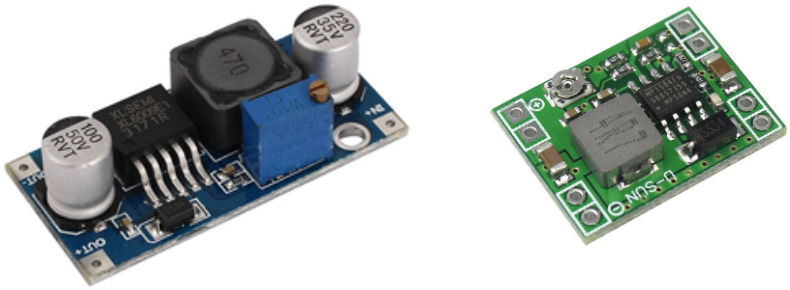
Fonte: Do autor.

4.3 Circuito de Alimentação

Na etapa de alimentação foram utilizados dois módulos regulador abaixador de tensão DC-DC, um LM2596 no qual está diretamente ligados nos terminais de entrada *Vin* das plataformas GR-PEACH RZ/A1H e a IoT/FF-LPC546xx. O módulo MP1584EN com mesmo recurso abaixador de tensão é utilizado para alimentação dos transceptores e a plataforma PSoC® 5L. A Figura 46 a) e b) mostram os módulos da etapa de alimentação.

Figura 46 – Módulos regulador de tensão DC-DC *step-down*.

(a) LM2596. **(b)** MP1584EN.



Fonte: Do autor.

Esses módulos reguladores trabalham como um conversor DC-DC no modo *Step-down*. Reguladores de tensão do tipo *step-down* são componentes ativos os quais delimitam a tensão de saída e mantém uma corrente de saída eficiente. Especificações dos módulos são apresentadas na Tabela 7 a seguir.

Tabela 7 – Especificações dos módulos reguladores (*step-down*.)

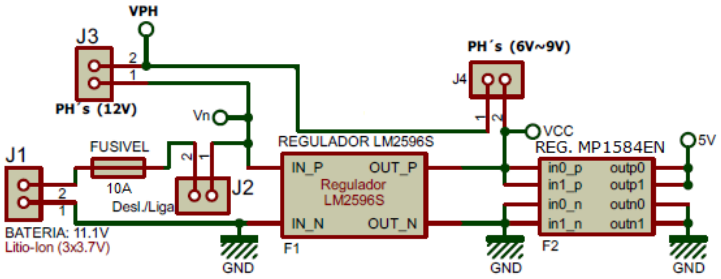
a) Regulador LM2596 DC-DC	**b) Regulador MP1584 DC-DC**
Tensão de entrada: +4.75V a 35V	Tensão de entrada: +4.75V a 28V
Tensão de saída: +1.25V a 26V	Tensão de saída: +0.8V a 20V
Corrente de saída: 2A(nom.) 3A(máx.)	Corrente de saída: 1.8A(nom.) 2.5A(máx.)
Eficiência de conversão: 92%	Eficiência de conversão: 96%
Freq. de chaveamento: 150KHz	Freq. de chaveamento: 1MHz a 1.5MHz
Regulação da carga: +-0.5%	Regulação da carga: +-0.5%
Regulação da tensão: +-0.5%	Regulação da tensão: +-2.5%
Dimensões do módulo: 44mm x 22mm	Dimensões do módulo: 22mm x 17mm

Fonte: Dados dos fabricantes.

Alguns detalhes das configurações de alimentação foram projetados no sentido de que opções seguras seriam necessárias para programação, simulação e testes, utilizando qualquer plataforma da PCB separadamente. Esse procedimento consiste em a tensão regulada de entrada que alimenta essas plataformas, através de conectores PH2.0mm mini *jumper* inseridos na PCB, sejam utilizados caso o cabo USB esteja conectado em qualquer

plataforma inserida na PCB. Com isso também se evita conflitos entre a alimentação do computador e a tensão de entrada. A Figura 47 ilustra o circuito de alimentação.

Figura 47 – Circuito de alimentação.



Fonte: Do autor.

- J1: Entrada da alimentação da bateria;
- J2: Chave liga/desliga;
- J3: Chave de alimentação dos drivers dos motores (12V);
- J4: Chave de alimentação para motores (6V a 9V).

Na PCB também pode ser verificado no circuito de alimentação a inclusão de um fusível montado em superfície SMD (do inglês, *Surface Mount Device*) auto-redutor *PPTC* (do inglês, *Polyer Temperature Positivo Coefficent*) de 10A. Nesse componente, quando a corrente do circuito é muito alta, o fusível reinicializável PPTC mudará de baixa resistência para alta resistência sob o efeito da temperatura dentro de um certo intervalo de tempo, evitando assim o grande fluxo de corrente, protegendo todo o circuito modular.

A escolha do valor do fusível se deu devido ao valor máximo de corrente quando o motor está com o eixo bloqueado (*stall*) que é de 4.9A. Sendo assim, considerou-se um fusível com um valor de duas vezes ao valor da corrente de bloqueio do motor DC utilizado.

4.3.1 Bateria de Lithium - Li-Po

Para suprir a corrente necessária para os 4 motores DC, plataformas, módulos de comunicação, circuito *Step-up* e transceptores foi necessário uma bateria com boa relação entre carga/peso e corrente máxima de descarga. A escolha da bateria foi feita admitindo-se corrente máxima e peso. A Figura 48 mostra a bateria utilizada.

Figura 48 – Bateria de Li-Po da *Vok Power*.

Fonte: Do autor.

Baterias de Lithium Polímero (Li-Po) são geralmente mais utilizadas em projetos de robótica, por terem boa capacidade de carga e serem recarregáveis. Sendo assim, optou-se por uma bateria de 11.1V (3 células de 3.7V) com 1500mAh de capacidade de corrente e taxa de descarga de 25C (capacidade nominal da bateria) que supre os requisitos necessários. Para cálculo da obtenção da corrente máxima de descarga da bateria utilizada, tem-se que:

$$D_{mA} = C \cdot I_{mAh} \Rightarrow D_{mA} = 37.5A \tag{4.1}$$

Para a PCB desenvolvida, incluindo os motores, estimou-se um consumo de corrente máxima de 6A, com isso, obtém-se o valor máximo da corrente de descarga que a bateria suporta em ampères, esse valor tem de ser maior do que o consumo máximo do robô.

$$I_{máx} < C \cdot I_{mAh} \Rightarrow 6A < 37.5A \tag{4.2}$$

Para cálculo do tempo de descarga, considerou-se que o robô irá consumir uma corrente média de 1.5A. Com a Equação 4.3, tem-se que:

$$T_D = \frac{I_(mAh)}{I_(média)} \cdot 60 \tag{4.3}$$

Com isso, obtém-se o tempo aproximado de descarga da bateria.

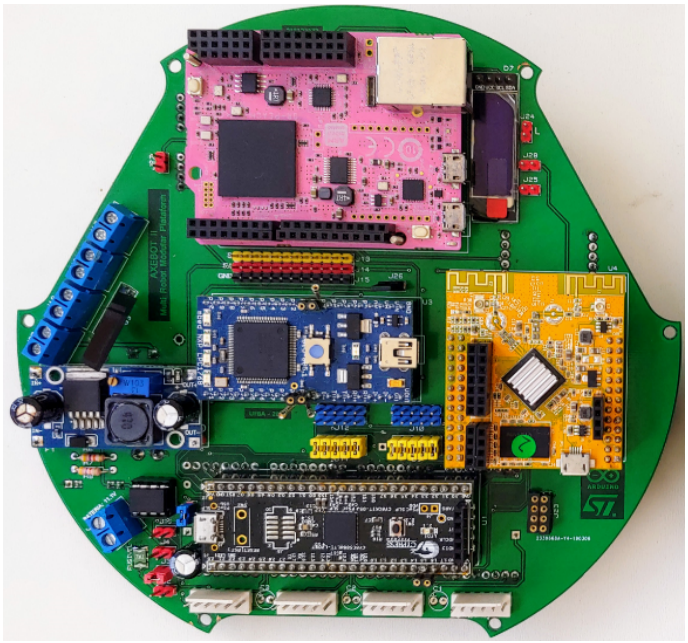
$$T_D = \frac{1.5A}{1.5A} \cdot 60 = 60_{min} \tag{4.4}$$

4.4 Camadas inferior e superior da PCB

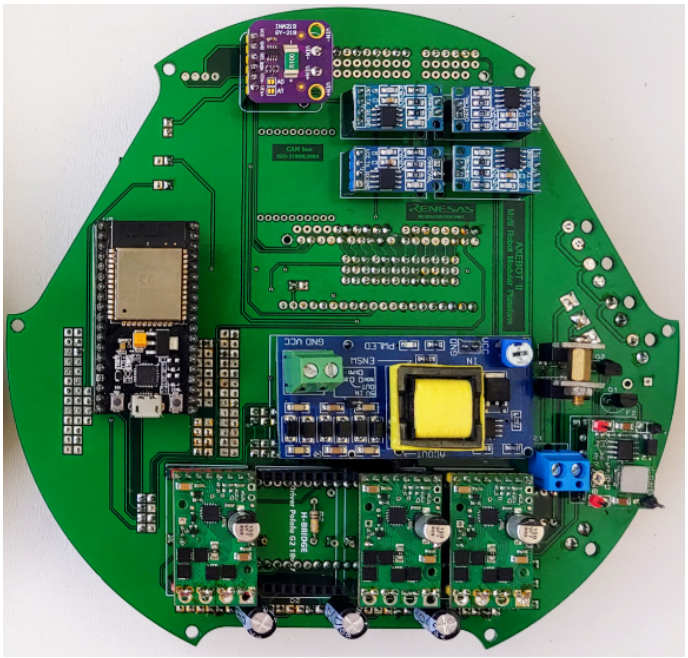
A PCB desenvolvida possui característica modular, na qual permitirá a substituição de qualquer plataforma ou módulo facilmente. Toda a PCB é composta por conectores *pin header conector socket* fêmea com um espaçamento padrão entre os terminais de 2,54mm. As Figuras 49 a) e b) ilustra os dois lados da PCB (superior e inferior), respectivamente.

Figura 49 – PCB desenvolvida.

(a) Camada superior.



(b) Camada inferior.

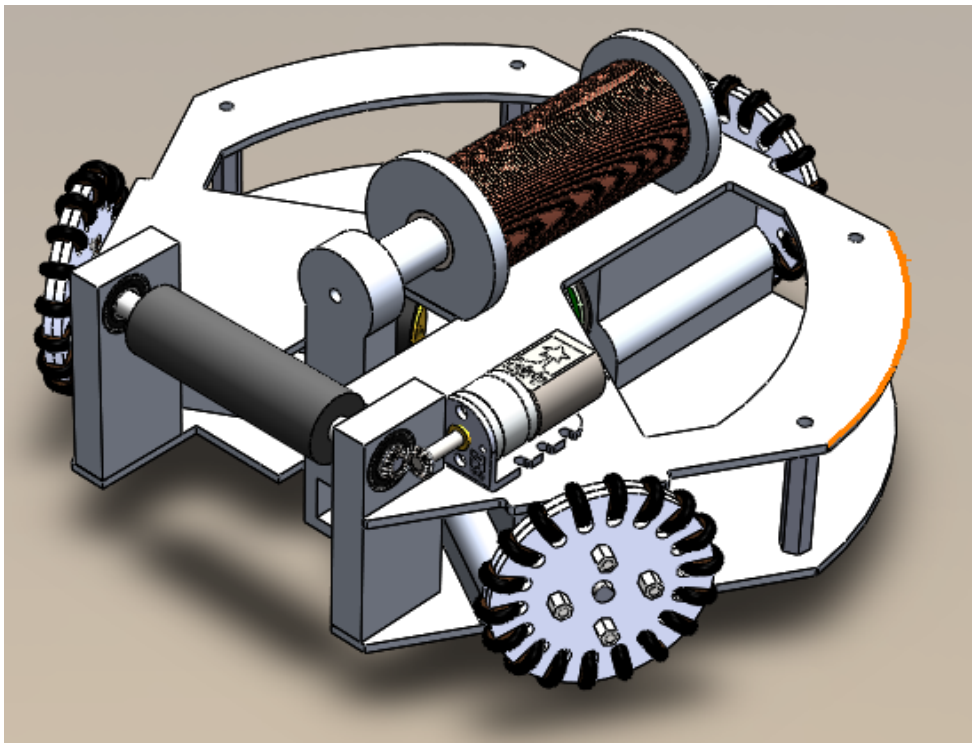


Fonte: Do autor.

4.5 Estrutura mecânica

Como mencionado no Capítulo 1, a estrutura mecânica dessa plataforma segue os padrões da *RoboCup League* para competições do futebol de robôs na categoria *Small Size League* F180. O robô com sua carapaça (*shell*) tem aproximadamente 175mm de diâmetro e 145mm de altura, possui duas partes mecânicas em alumínio de 3mm de espessura, espaçadas por 35mm de uma base para outra e na parte interna. Essas bases contém dois espaços para comportar as baterias e capacitores do circuito de acionamento da solenoide. A Figura 50 ilustra a estrutura mecânica do robô.

Figura 50 – Estrutura mecânica do robô no SOLIDWORKS®.

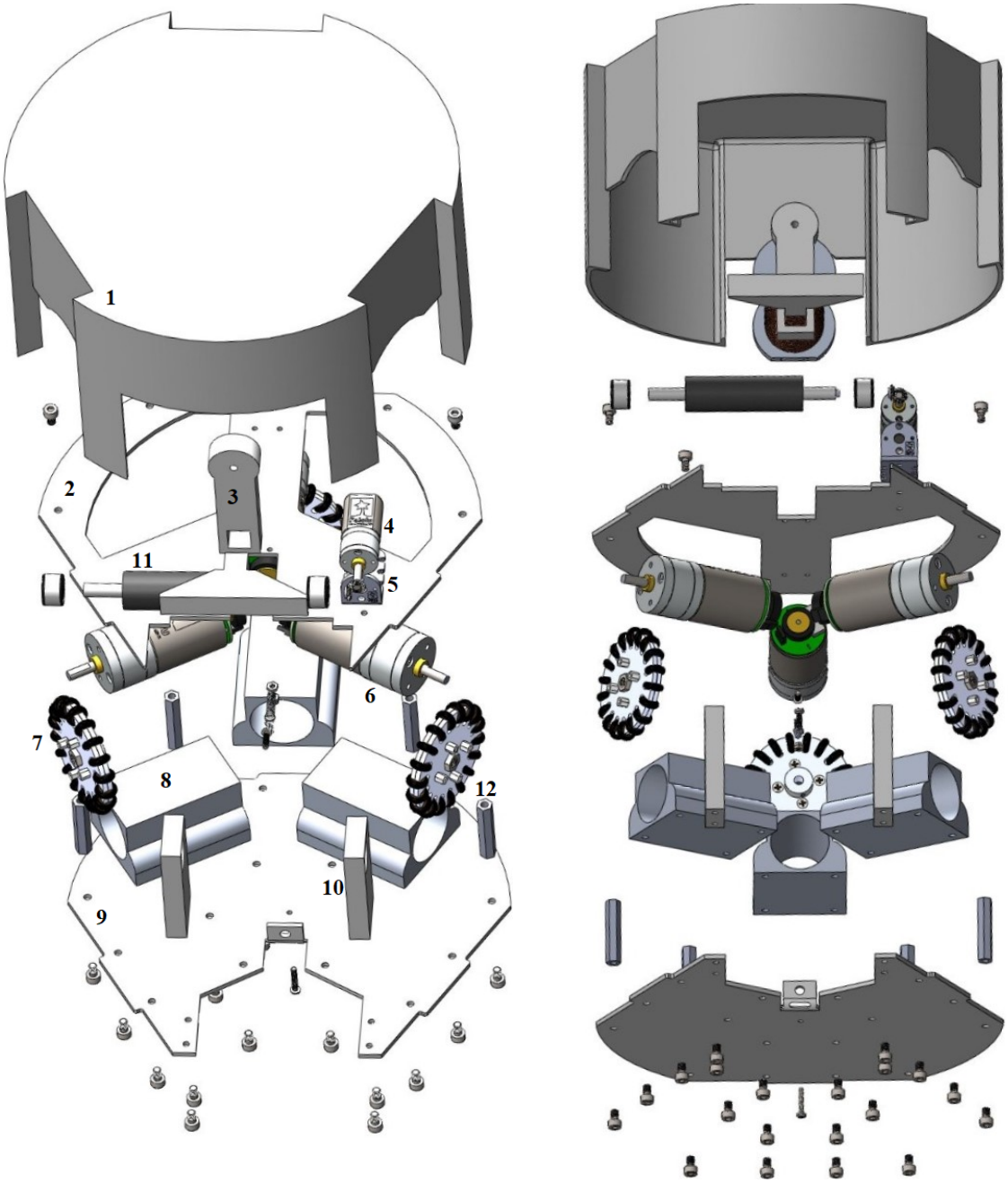


Fonte: Do autor.

Essas partes foram prototipadas e construídas utilizando o SOLIDWORKS® Premium e o AUTOCAD® 2D. O robô está montado com três rodas omnidirecionais, cada uma acoplada por um motor DC composta por caixa de redução, descrito mais detalhadamente nas próximas subseções deste Capítulo. Os ângulos entre cada motor são exatamente 120°, na qual proporciona eficiência e baixo atrito. Cada roda omnidirecional da GTF *Robots* tem 50mm de diâmetro composta por 18 sub-rodas em sua extremidade. A Figura 51 ilustra a vista geral do robô. O projeto modular, consiste em dois sistemas para serem utilizados em competições do futebol de robôs: um dispositivo de drible (frontal com base entre um dos três ângulos de 120°) e um dispositivo de chute, composto por um efetuador linear a um desses ângulos. A Figura 53 ilustra a vista geral do robô.

Figura 51 – Vista explodida.

(a) Vista superior. **(b)** Vista inferior.

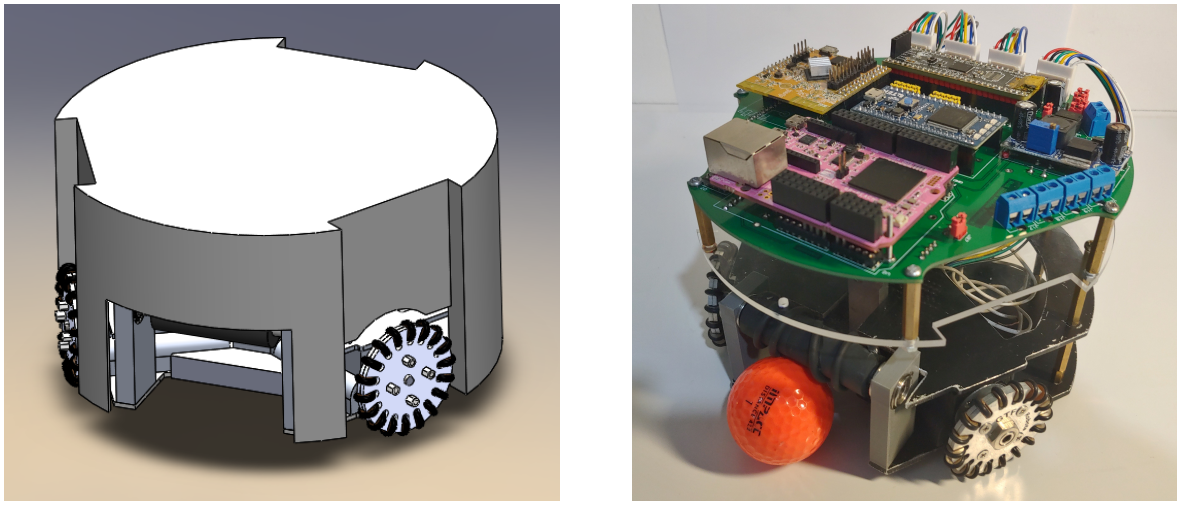


Fonte: Do autor.

Nº	Descrição	Nº	Descrição
1	carapaça	7	roda omnidirecional
2	chassis superior	8	suporte do motor das rodas
3	mecanismo chute	9	chassis inferior
4	motor do sistema de drible	10	suporte do *roller ball*
5	suporte do motor do sistema de drible	11	*roller ball*
6	motor das rodas	12	parafuso extensor

Figura 52 – Estrutura com a capa externa.

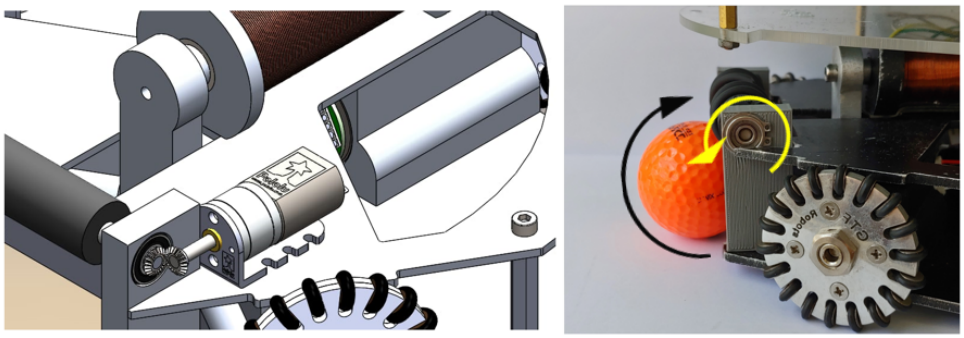
(a) Modelo completo no SolidWoroks. **(b)** Estrutura interna.



Fonte: Do autor.

Na plataforma desenvolvida também foi possível projetar um sistema de drible, comumente utilizado em competições do futebol de robô. Esse sistema é composto por um motor DC de 12V com redução de 34:1, com engrenagens e uma barra tubular em alumínio revestida de borracha para contato giratório com a bola. A Figura 53 ilustra esse sistema.

Figura 53 – Sistema de drible.



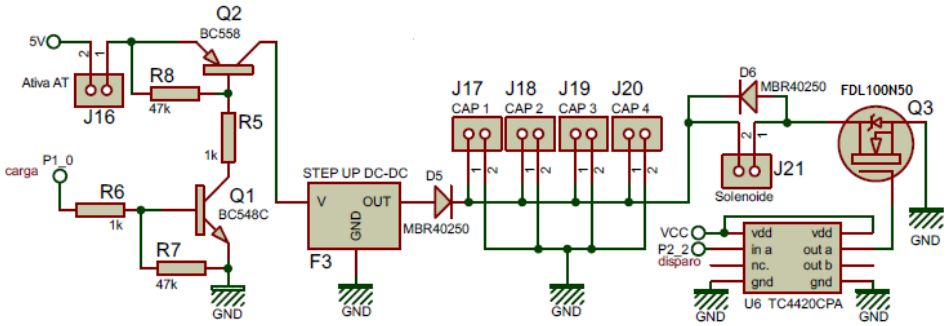
Fonte: Do autor.

4.5.1 Circuito de acionamento da solenoide: *Kicker circuit*

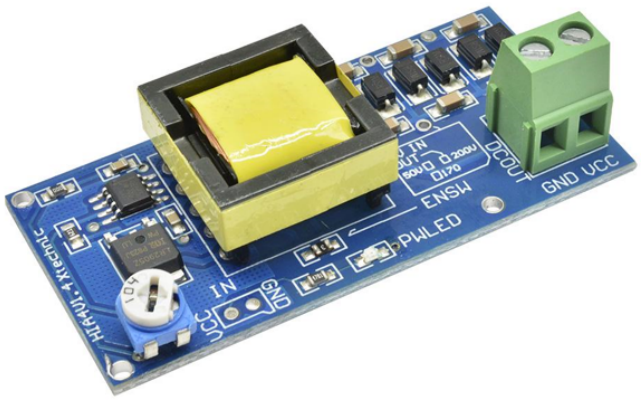
Outro mecanismo, também desenvolvido para o futebol de robôs, consiste num sistema de disparo linear através de uma solenoide tubular, com um curso de 4cm que será aplicado à bola. Esse circuito é acionado pelo elevador de tensão (do inglês, *step up*) aplicando cerca de 200V ao êmbolo após a descarga de quatro capacitores de 2200μF/250V. A Figura 54 a) e b) mostra o circuito e a placa conversora.

Figura 54 – Circuito de alta tensão.

(a) Circuito no Proteus.



(b) Placa do conversor.



Fonte: Do autor.

Ao ser acionado, será capaz de fornecer cerca de 30A de corrente à solenoide, provocando um impacto suficiente para deslocar a bola por cerca de 10 metros. Para controlar a alta corrente usando uma tensão de entrada de 5V à placa do conversor elevador de tensão, é utilizado um mosfet FDL100N50 especificado para suportar até 500V/100A, no qual é chaveado pela porta *gate* através do circuito integrado TC4420CA.

Alguns problemas podem ser causados utilizando-se alta tensão e corrente da conexão de circuitos envolvendo toda a PCB com o mesmo plano de aterramento. Para diminuir tais problemas, optou-se pela utilização de um diodo *SCHOTTKY* MBR40250G de 40A/250V, ou seja, suporta uma corrente direta de 40A e uma tensão de bloqueio de até 250V.

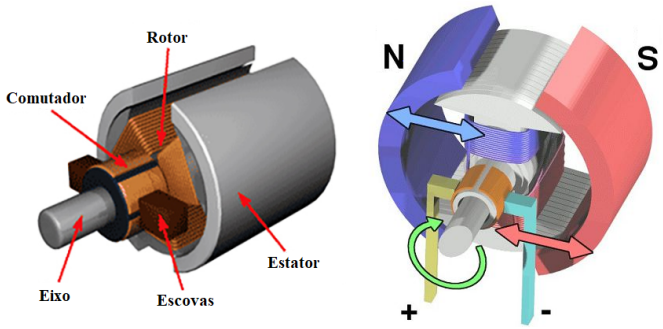
4.6 Motores DC utilizados

Motores de corrente contínua ou mais comumente chamado de motores DC têm amplas aplicações em sistemas de controle porque são mais simples de modelar e controlar. Um projeto de otimização do sistema de controle, muitas vezes um modelo preciso do motor DC usado em um sistema de controle é de extrema importância. Neste caso, a depender

do fabricante, alguns parâmetros importantes para modelagem não são disponibilizados e alguns valores para se modelar o motor dados nas especificações, geralmente fornecido pelo fabricante do motor, não são suficientes ou pode não ser considerado adequado, especialmente para motores DC de baixo custo, que tendem a tolerâncias relativamente grandes em suas partes elétrica e mecânica [130].

Os motores DC utilizados neste projeto são com imã permanente. Estes motores são formados por duas estruturas internas principais: estator (enrolamento do campo ou imã permanente) e rotor (enrolamento da armadura). A Figura 55 ilustra a estrutura interna de um motor DC.

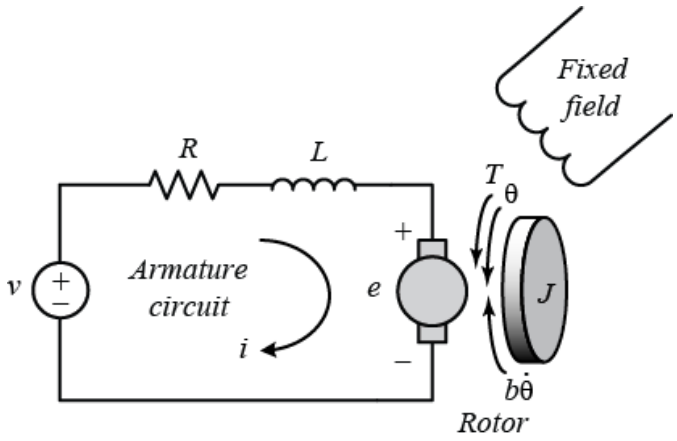
Figura 55 – Estrutura interna de um motor DC.



Fonte: Do autor.

Para gerar um modelo de um motor DC, foi considerado uma versão básica de seu funcionamento. A Figura 56 representa um circuito equivalente elétrico da armadura e o diagrama do rotor, no qual o rotor e o eixo são considerados rígidos.

Figura 56 – Esquema representativo de um motor DC.



Fonte: Baseado em [131].

Nesse esquema, considerou-se a fonte de tensão ν aplicada à armadura do motor DC como a entrada e a velocidade $\frac{d\theta}{dt}$ de rotação do eixo como a saída. Para fins de modelagem,

a velocidade de um motor DC é controlada diretamente pela tensão aplicada nos terminais de armadura, como apresentado na Equação 4.5 e o torque é controlado pela corrente, o que é mostrado na Equação 4.6. Já o sentido de giro é definido pela polarização da tensão ν nos terminais do motor. Uma vez invertida a polaridade nos terminais, o motor tem o giro de seu eixo invertido devido a inversão de polaridade, na qual se baseia a Equação 4.7.

$$\omega = \frac{\nu - Ri_d + L\frac{di_d}{dt}}{K\Phi} \tag{4.5}$$

$$T = K\Phi \cdot id \tag{4.6}$$

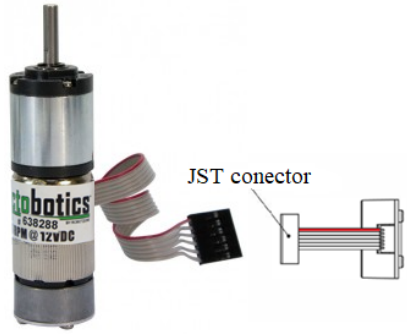
$$e = K\Phi \cdot \theta \tag{4.7}$$

As Equações 4.5 e 4.6 descrevem a velocidade de rotação e torque do motor de acordo seu modelo elétrico. Sendo que:

- ν Tensão da fonte;
- R: Resistência da armadura;
- i_d: Corrente de armadura;
- L: Indutância do enrolamento da armadura;
- Φ: Fluxo de entreferro;
- K: Constante do motor (determinada por características construtivas);
- T: Torque;
- J: Momento de inércia;
- b: Coeficiente de atrito;
- e: Campo elétrico.

No presente projeto, foram utilizados motores da *Actobotics* ref. 638304 com caixa de redução planetária fabricado pela SERVOCITY®. Trata-se de um conjunto composto pelo motor modulado a uma caixa de redução em sua parte dianteira e um codificador magnético DE3 – *Hennkwell Ind. Co., Ltd.* embutido em sua parte traseira. A Figura 57 ilustra o motor.

Figura 57 – Motor *Actobotics*® de 12V/624RPM.



1	vermelho	motor +
2	cinza	canal B
3	cinza	codificador (Vcc)
4	cinza	codificador (GND)
5	cinza	canal A
6	cinza	motor -

Fonte: [132].

O conjunto modular mecânico do seu eixo com a caixa redutora permite uma redução na velocidade de rotação de 19.225:1, ou seja, uma rotação completa do eixo externo equivale a aproximadamente 19 rotações do eixo do motor.

Nota-se que em um sistema mecânico, este conjunto modular está sujeito a desgastes o que podem modificar o comportamento dinâmico do motor relativo à sua parte mecânica, quando atuadores estão sujeitos a utilizações intensas sob pena de causar-se danos ao seu enrolamento.

No presente trabalho, por ter sido utilizado motor DC de baixo custo, alguns procedimentos foram realizados a fim de obter confiabilidade dos dados (ν e RPM) fornecidos pelo fabricante desses atuadores, como um ensaio de entrada de tensão (Volts) e saída em velocidade (RPM). No motor utilizado, o valor informado pelo fabricante é de 12V/624RPM com margem de erro de ±62RPM a uma corrente de 0.19A operando sem carga.

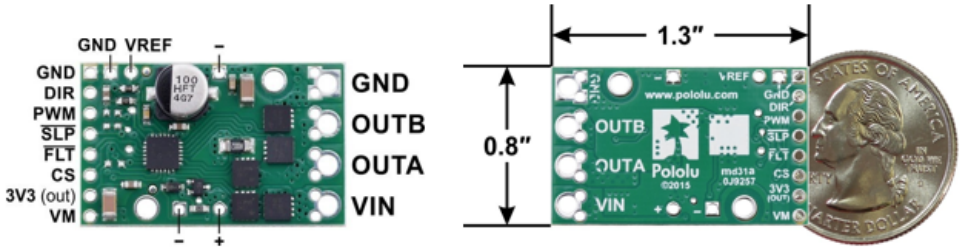
As principais justificativas para a realização do procedimento de verificação da confiabilidade é de que o fabricante não disponibiliza os parâmetros principais como resistência (R), indutância (L), coeficiente de atrito (B), momento de inércia (J) e a constante (K), que são parâmetros imprescindíveis para obtenção da função de transferência para consequentemente se desenvolver o projeto do controlador.

Alguns métodos de identificação do sistema como citados em [133], [134], [135] e [136] pode ser aplicado à identificação do modelo do motor DC. Após identificados, os modelos do motor DC são frequentemente usados posterior projeto do controlador, como desenvolvidos em [137] e [138]. Outro importante fator a considerar é o tempo de utilização desses atuadores e os níveis operacionais de potência de entrada, situações que podem inviabilizar a utilização de informações oriundas de outros trabalhos com propósitos semelhantes.

4.6.1 Driver de Acionamento dos Motores: Configuração em Ponte H

O driver utilizado no projeto é um módulo Pololu G2 *Motor Driver* 18v17 composto por MOSFET's (do inglês, *Metal Oxide Semiconductor Field Effect Transistor*) de canal N, o AON7418 da *Alpha and Omega Semiconductor*, ilustrado na Figura 58.

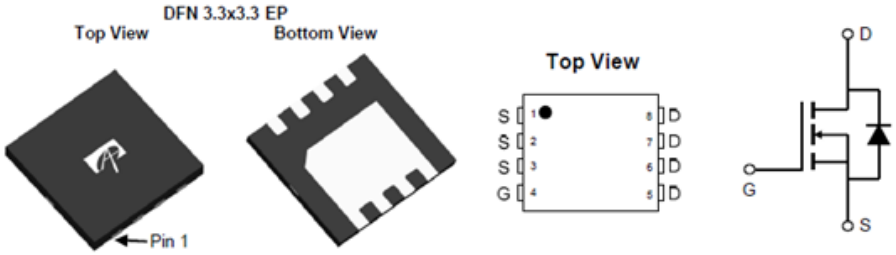
Figura 58 – Pololu G2 High-Power Motor Driver 18v17 e *pinout*.



Fonte: [139].

Este driver em configuração ponte H permite o controle bidirecional de um motor DC escovado a uma faixa de tensão de 6.5V a 30V, eficiente para fornecer uma corrente de até 17A contínua. Recursos adicionais deste driver incluem proteção de tensão reversa, detecção e limitação de corrente. As conexões lógicas são projetadas para interface com sistemas de 1.8V a 5V. A Figura 59 ilustra o componente.

Figura 59 – MOSFET AON7418 da Alpha & Omega Semiconductor.



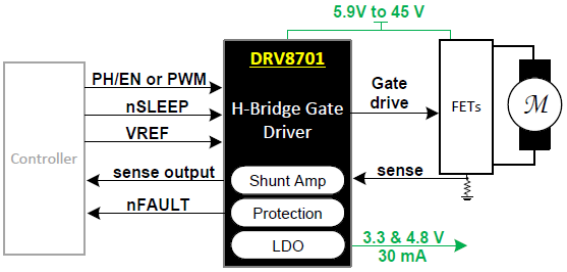
Fonte: [140].

Parte do circuito desse módulo contém um circuito de referência DRV8701E *Brushed DC Motor Full-Bridge Gate Driver* da *Texas Instruments* para receber entradas de sinal e controlar esses MOSFET's. Esse dispositivo pode ser alimentado com uma tensão de alimentação entre 5.9V e 45V e uma interface PH/EM (8701E) permite a conexão com os circuitos do controlador e um amplificador interno, no qual permite o monitoramento e controle de corrente ajustável do motor.

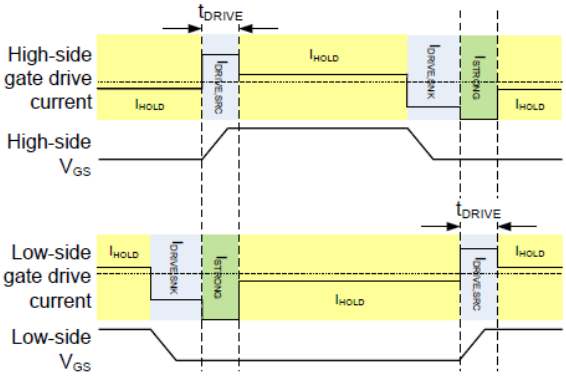
O DRV8701E direciona os FET's (do acrônimo em inglês, *Field Effect Transistor*) de alta e de baixa intensidade com o gate driver V_{GS}. O diagrama em blocos do componente e seu gráfico de corrente são ilustrados na Figuras 60 a) e 60 b), respectivamente.

Figura 60 – Diagrama simplificado e gráfico de corrente do módulo.

(a) Diagrama de funcionamento.



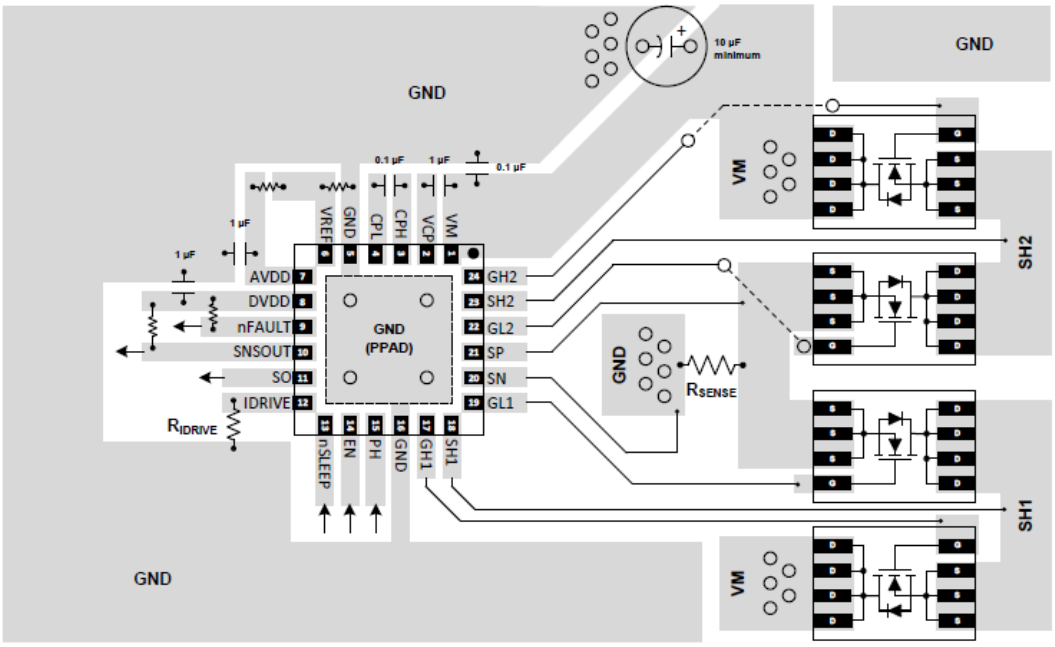
(b) Gráfico de corrente.



Fonte: [141].

O driver composto pelo DRV8701E integra FET's para controlar quatro MOSFET's de canal N externos em configuração ponte H a um motor DC escovado bidirecional de 6V a 18V. Nesse drive também é fornecido um modo de suspensão de baixa potência que desliga os circuitos internos para obter um consumo de corrente inativo baixo. Este modo de suspensão pode ser definido no terminal $nSLEEP$ e as condições de falha são indicadas no terminal $nFAULT$. Uma configuração padrão para utilização desse driver com MOSFET's externos, no qual faz parte do módulo composto pelo driver do motor utilizado no projeto é ilustrada na Figura 61.

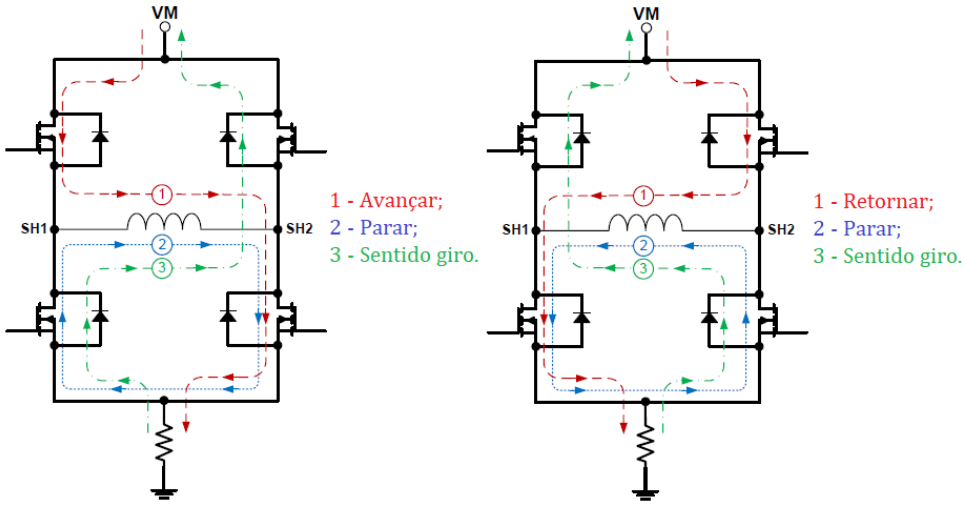
Figura 61 – Configuração padrão de *layout* para utilização do DRV8701E.



Fonte: [141].

A Figura 62 ilustra o sentido de giro do motor.

Figura 62 – Estados operacionais da configuração em ponte H do driver.



Fonte: [141].

4.7 Controle do Motor DC

Por padrão, o driver Pololu G2 18V17 está em nível lógico baixo e o terminal *SLP* deve ser acionado pelo microcontrolador ou alimentado por uma tensão lógica de no máximo 5.5V para habilitar o driver. Tensão abaixo desse valor, ou seja, de 3.3V também pode ser utilizada.

Em uma configuração típica, a exemplo do projeto aqui desenvolvido, apenas dois outros terminais serão necessários, o PWM (*Pulse Width Modulation*) e DIR (*Direction*). Se o terminal PWM estiver com nível lógico baixo (LOW), ambas as saídas do motor (OUTA e OUTB) também serão mantidas baixas, o driver se comportará como uma operação de frenagem. Caso o PWM esteja com nível lógico alto (HIGH), as saídas do motor serão acionadas de acordo com a entrada no terminal DIR. Isso permite dois modos de operação:

- Sinal-magnitude, em que o ciclo de trabalho *Duty Cycle* PWM controla a velocidade do motor;
- e DIR controla a direção e a frenagem, na qual um sinal modulado por largura de pulso é aplicado ao terminal DIR com PWM mantido alto.

Na operação de frenagem, um ciclo de trabalho baixo aciona o motor em uma direção, um ciclo de trabalho alto aciona o motor na outra direção e um ciclo de trabalho de 50% desliga o motor. Quando DIR tem nível alto (HIGH), a corrente fluirá de OUTA

para OUTB e quando está baixo, a corrente fluirá de OUTB para OUTA. A Tabela 8 a seguir mostra as possíveis configurações de acionamento do driver.

Tabela 8 – Níveis lógicos de operação do driver.

PWM	DIR	OUTA	OUTB	Operação
HIGH	HIGH	HIGH	LOW	Avançar
HIGH	LOW	LOW	HIGH	Retornar
LOW	X	LOW	LOW	Parar

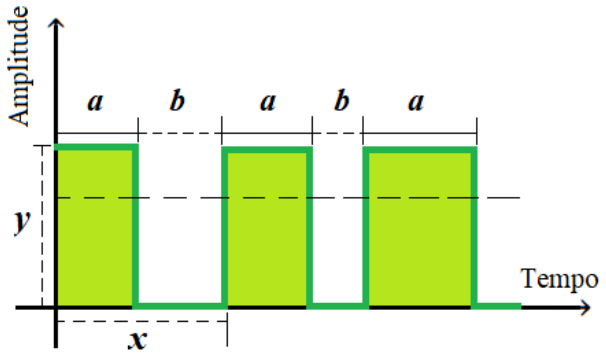
Fonte: [139].

4.7.1 Frequência PWM do *driver*

A Modulação PWM (do inglês, *Pulse Width Modulation*) consiste numa onda quadrada modulada, caracterizada por sua frequência e pelo ciclo de trabalho ativo (do inglês, *Duty Cycle*). É uma maneira de controlar circuitos analógicos usando os terminais digitais dos microcontroladores variando o seu ciclo de trabalho desse sinal digital a uma frequência determinada, e dessa maneira obter o valor médio médio caracterizado pelo evento periódico.

A relação entre o tempo em que ocorre o pulso e a duração de um ciclo completo de operação define o ciclo ativo do driver, como ilustrado na Figura 63.

Figura 63 – Sinal PWM.



Fonte: Do autor.

De acordo com a Figura 63, temos que:

- x: Período [segundos];
- y: Amplitude [Volts];
- a: *Duty Cycle* ativo [%];

- b: *Duty Cycle* inativo [%];
- f: Frequência [Hertz];
- Vm: Tensão média fornecida ao motor [Volts].

Sendo que,

$$f = \frac{1}{x} \tag{4.8}$$

$$Duty\ Cycle(\%) = \frac{a}{x} \cdot 100 \tag{4.9}$$

Ou seja,

$$Duty\ Cycle = \frac{b}{b+a} \cdot 100 \tag{4.10}$$

Por fim, a tensão média fornecida à carga é diretamente proporcional ao *Duty Cycle*, e é dada pela Equação 4.11:

$$Vm = y \cdot Duty\ Cycle \tag{4.11}$$

O driver do motor suporta frequências do PWM maiores que 100kHz, mas as perdas de comutação no driver serão proporcionais à frequência do PWM. Normalmente, cerca de 20kHz é uma escolha ideal para a operação desse sinal, no qual resulta em uma operação isenta de ruídos.

Um pulso no terminal PWM deve ser em nível alto por um tempo mínimo de aproximadamente 0.5µs e pulsos de entrada com tempo menor não produz mudança na saída, portanto *Duty Cycle* baixos tornam-se indisponíveis em altas frequências. Por exemplo, numa frequência de 100kHz, o período de pulso é 10µs e o ciclo de trabalho ativo mínimo é de $\frac{0.5}{10}$ ou 5%.

4.7.2 Sensor de corrente

O terminal de detecção do sensor de corrente CS do *driver* gera uma tensão proporcional à corrente do motor enquanto o driver está em funcionamento. A tensão de saída é cerca de 20mV/A mais um pequeno deslocamento (*offset*), que normalmente é de cerca de 50mV, podendo haver variações de um módulo para outro.

Os valores de tensão de *offset* medido nos *drivers* utilizados foram diferentes nos três módulos. Esses valores foram medidos alimentando o módulo com a tensão que será

utilizada pelos motores, ou seja, 12V. Na Tabela 9 a seguir, listamos os valores de *offset* de cada driver e suas variações devido a oscilação na medição da tensão com o multímetro.

Tabela 9 – Valores de *offset* do sensor de corrente.

Driver 1		**Driver 2**		**Driver 3**	
mín	máx	mín	máx	mín	máx
71.2mV	72.4mV	56.7mV	57.9mV	34.1mV	35.3mV
Variação(%)		**Variação(%)**		**Variação(%)**	
1.69		2.12		3.52	
Valor médio (mV)		**Valor médio (mV)**		**Valor médio (mV)**	
71.8		57.3		34.7	

Fonte: Do autor.

Com base nos valores medidos, foi utilizado uma média entre os valores mínimo e máximo, devido a oscilação dos baixos valores medidos em mili-volts (mV), definido como o valor de *offset*. Um dos pontos positivos com relação a esses valores foi o fato de que essas variações não ultrapassaram 5%, no qual torna o *driver* preciso para utilizá-los em diversos projetos.

Antes do acionamento do *driver*, porém alimentado pela tensão da bateria, a saída CS está ativa somente para a leitura do valor de *offset*, está em nível lógico alto (HIGH) enquanto o *driver* está no modo de funcionamento e está inativo (LOW) quando o *driver* está no modo de frenagem (*break*), no qual acontece quando a entrada PWM está baixa ou a limitação de corrente está ativa.

Para verificar a precisão do sensor de corrente, foi realizado um experimento, não usando somente os motores, mas também utilizando um resistor de alta potência no valor de 12Ω/10W para uma breve análise do valor de corrente circulante entre os terminais do componente utilizando a Equação 4.12 a seguir.

$$V_{CS(mV)} = V_{resolução(mV)} \cdot I_R(A) + V_{offset(mV)} \tag{4.12}$$

A intenção de se utilizar uma medição utilizando um resistor, foi do fato de que ao utilizar o motor para verificação do valor de tensão do terminal **CS**, não foi possível verificar com precisão esse valor visto que o eixo do motor tinha de ser bloqueado para que o valor de corrente atingisse um valor próximo a 1A de corrente. A escolha desse valor de resistor se deu devido a facilitar a leitura do sinal de saída do terminal **CS** utilizando a Lei de Ohm na Equação 4.13, visto que a tensão utilizada pelo motor será de 12V, e que o intuito era de obter uma corrente de 1A entre os terminais $OUTA$ e $OUTB$ do *driver*,

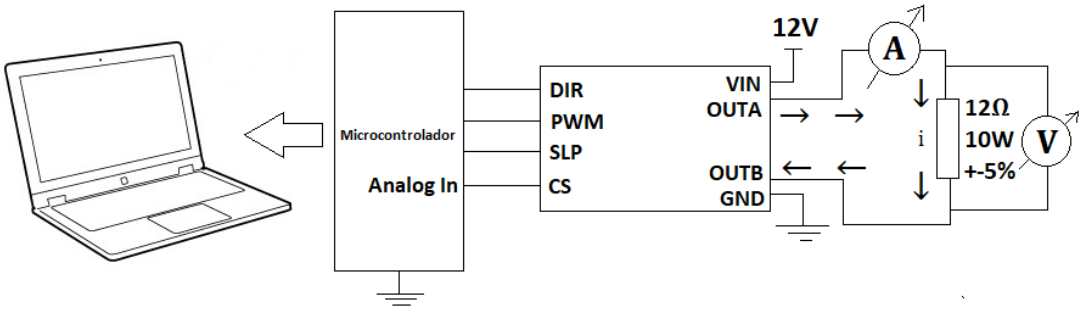
analisando a resolução desse valor em $20mV/A$ informado pela folha de dados do módulo.

$$I_R(A) = \frac{Vin(V)}{R_{AB}(\Omega)} \tag{4.13}$$

Para verificação da precisão do sensor de corrente dos três *drivers* dos motores, o módulo foi alimentado com uma fonte de alimentação *Minipa MPS-3005A*, na qual tem a capacidade de gerar até 30 volts e 5A de corrente.

A Figura 64 ilustra o circuito que foi utilizado para medir o valor de tensão no terminal CS_{pin}.

Figura 64 – Circuito para aferir a precisão do sensor de corrente.



Fonte: Do autor.

Em análise realizada quando o código foi executado, e fazendo o cálculo pela Equação 4.12, foram obtidos os resultados listados na Tabela 10:

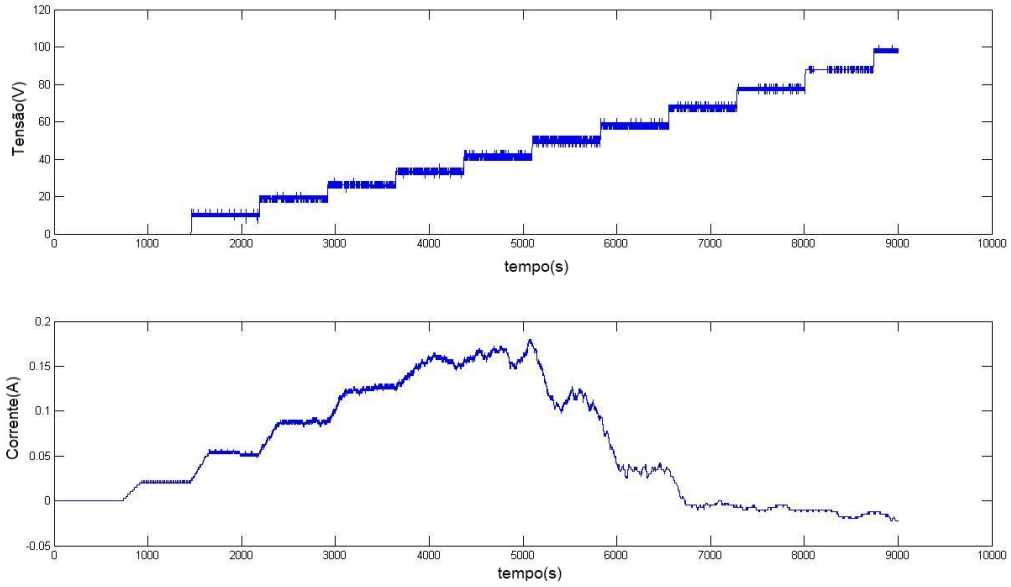
Tabela 10 – Valores de teste de precisão do *offset* do sensor de corrente.

Driver 1		**Driver 2**		**Driver 3**	
I_{off}=0A	V_{off}=71.9mV	I_{off}=0A	V_{off}=57.3mV	I_{off}=0A	V_{off}=34.7mV
I_{on}=1A	V_{on}=91.8mV	I_{on}=1A	V_{on}=77.3mV	I_{on}=1A	V_{on}=54.7mV
range=$V_{on} - V_{off}$		**range**=$V_{on} - V_{off}$		**range**=$V_{on} - V_{off}$	
V_{CS}=19.9mV		V_{CS}=20mV		V_{CS}=20mV	

Fonte: Do autor.

Para uma breve análise realizada com o motor, o mesmo circuito ao utilizado com o resistor ilustrado na Figura 64, no qual obteve-se o gráfico ilustrado na Figura 65 a seguir.

Figura 65 – Corrente proporcional obtida com o *driver*.

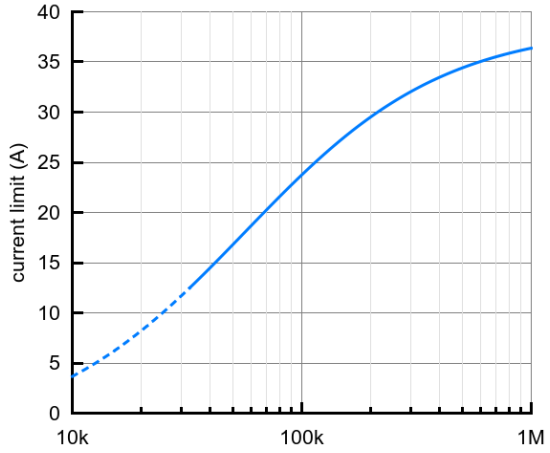


Fonte: Do autor.

O limite máximo que o driver suporta é configurado para uma faixa de 40A por padrão. Para diminuir esse limite, no projeto da PCB foi adicionado um resistor adicional de 100KΩ entre o terminal V_{REF} e o terra (GND) adjacente, reduzindo para aproximadamente 24A, como informa a folha de dados do fabricante.

No gráfico ilustrado da Figura 66, observa-se que essa limitação de corrente é menos precisa em configurações mais baixas, indicadas pela porção tracejada da curva no gráfico.

Figura 66 – Limites de corrente (A) versus o valor do resistor $k\Omega$ em V_{REF}.



Fonte: [139].

Os MOSFET's podem suportar grandes picos de corrente por curto espaço de tempo (por exemplo, acima dos 40A por alguns milissegundos), e o corte de corrente do driver manterá a corrente média abaixo do limite definido.

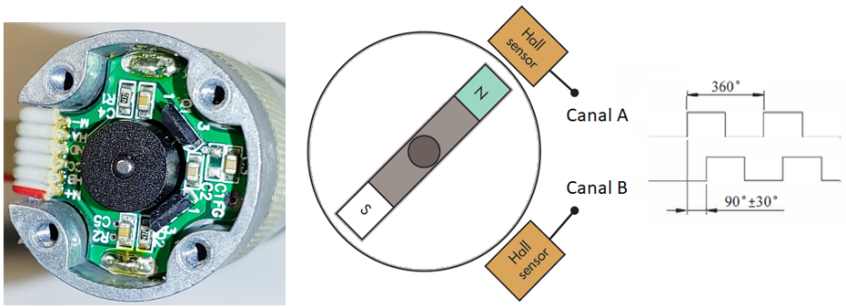
4.7.3 Encoder rotativos

Os codificadores rotativos são extremamente abrangentes em automação industrial, como no setor automotivo, em robótica móvel e até mesmo na área médica. Consiste em uma forma comum para medir a velocidade ou a posição angular para fins de controle e *feedback* em diversas aplicações.

Codificadores mais modernos permitem métodos magnéticos ou ópticos de leitura sem contato, no qual tem maior confiabilidade e desempenho, a exemplo dos sensores de efeito *Hall*[6] ao qual fazem parte dos codificadores acoplados aos motores do presente projeto.

Com um sensor, defasados a 90° do outro, torna-se possível detectar a direção e a velocidade de rotação a partir destes sinais. A Figura 67 ilustra esses sinais característicos.

Figura 67 – Mudança de fase do encoder por efeito *Hall*.



Fonte: Do autor.

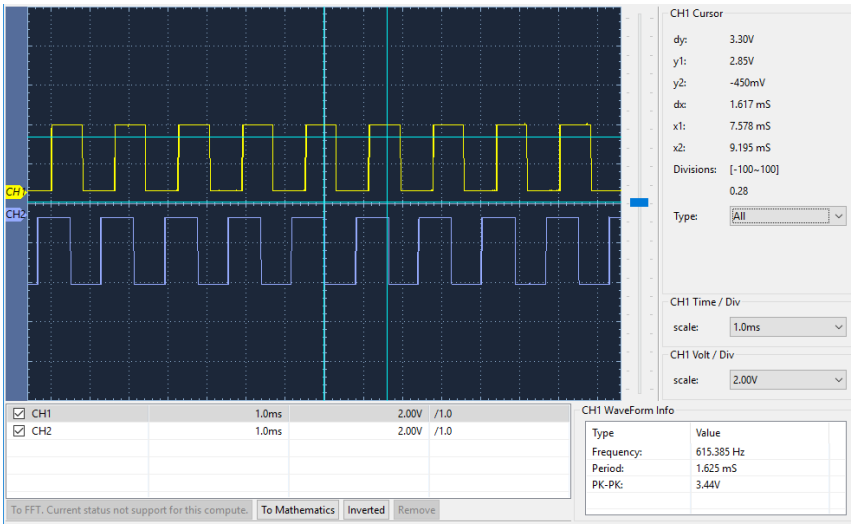
Esses componentes emprega um codificador magnético de quadratura para gerar os sinais de saída (canal A e canal B), ou seja, quando o eixo do motor composto pelo disco magnético está se movendo a uma velocidade constante, o ciclo de trabalho de cada pulso é 50% a uma forma de onda quadrada e existe uma diferença de fase de 90° geralmente referenciados como um "trem de pulsos" separados entre os terminais A e B.

Em um codificador rotativo, a frequência indica a velocidade de rotação do eixo do codificador. Utilizando um osciloscópio $OWONSDS1102$ foram obtidas as amostras dos sinais de saída com o motor alimentado com a tensão nominal (12V) e com os codificadores

[6] O Sensor *Hall* tem seu princípio de funcionamento baseado no Efeito *Hall*, descoberto em 1889, por Edwin Hall. Esse efeito é uma propriedade que se apresenta em um condutor quando um campo magnético perpendicular ao fluxo de corrente é aplicado sobre ele. Quando isso ocorre, uma diferença de potencial no condutor é gerada, chamada de "Tensão Hall"

alimentados a 3.3V. O *Duty Cycle* foi gerado automaticamente a uma valor de 49,7% para o canal A (ch 1 Osciloscópio) e 52,5% para o canal B (ch 2 Osciloscópio) esses sinais estão ilustrados na Figura 68.

Figura 68 – Mudança de fase do encoder com o giro do motor.



Fonte: Do autor.

De modo preciso, as informações de posição são geradas usando-se uma série de pulsos em quadratura de fase, de modo que a direção do percurso possa ser determinada. Em qualquer momento específico, a diferença de fase entre os sinais A e B será positiva ou negativa, dependendo da direção do movimento do codificador.

No caso do codificador rotativo acoplado ao motor desse trabalho, a diferença de fase é de $+90° \pm 30°$ para rotação no sentido horário e de $-90° \pm 30°$ para rotação no sentido anti-horário. A frequência dos pulsos na saída A ou B é diretamente proporcional à velocidade do encoder (taxa de mudança de posição) e as frequências mais altas indicam movimento rápido, enquanto frequências mais baixas indicam velocidades mais lentas.

4.8 Modelagem dos Motores DC

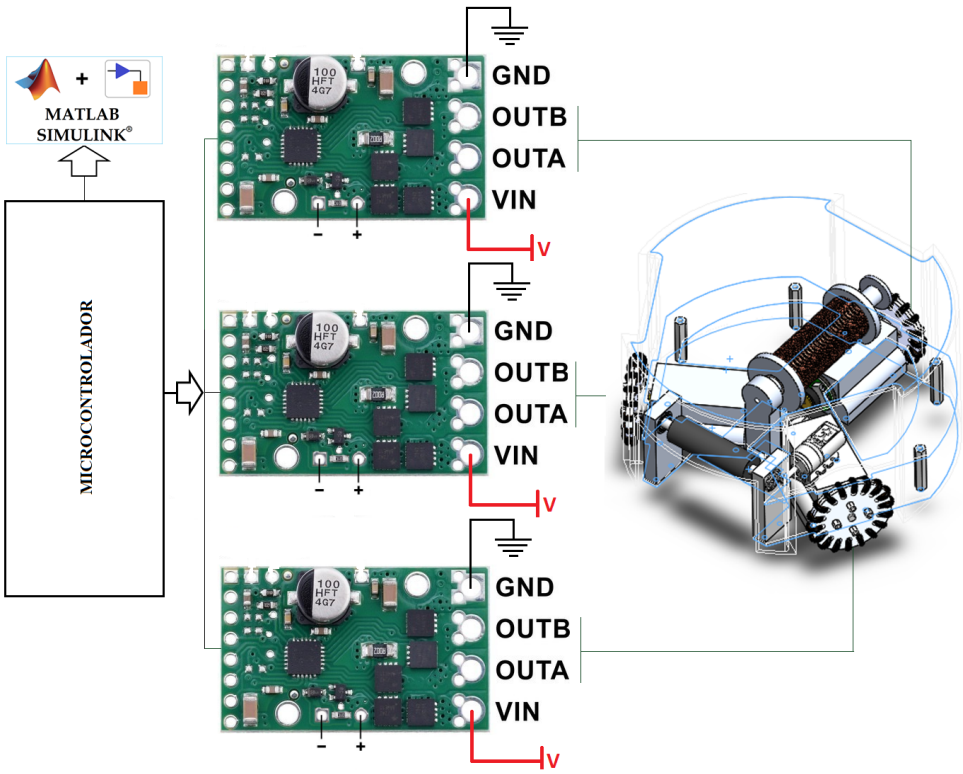
Modelar um motor concede estudar e simular o desempenho de seu sistema no decorrer do tempo, sem a utilidade de sua implementação física. Esse processo, concebe um grupo de técnicas que viabiliza alcançar um modelo a um determinado processo, por suposição, com um mínimo ou nenhum entendimento em relação as leis físicas que o regem.

Dentre algumas ferramentas para auxílio na área de controle de processos, o MATLAB® destaca-se como programa interativo para cálculos matemáticos e científicos de engenharia, onde é muito utilizado no meio acadêmico por alunos e professores. A família de programas deste software inclui além do programa principal, uma variedade de

ferramentas (*Toolboxes*) para auxilio em simulações, a exemplo dos arquivos especiais, os *scripts*, também chamados de (*m-files*) que estendem a utilidade do programa principal. Juntos, o programa principal e o *Control System Toolbox* propiciam a capacidade para analisar e projetar sistemas de controle [142].

No projeto, foi realizado um experimento escrito em um *script* no software MATLAB®, no qual a função *rotaryEncoder()* foi utilizada para obter as amostras dos encoders rotativos acoplados aos motores DC. Essa função está disponível no software a partir da versão 2017a, composta por uma ferramenta de suporte e conexão para uso das plataformas Arduino em seus diversos modelos. A Figura 69 a seguir ilustra de forma resumida de como os drivers estão configurados na PCB desenvolvida.

Figura 69 – Verificação da velocidade em RPM.



Fonte: Do autor.

A obtenção desses valores permite a comparação do parâmetro RPM em relação a tensão aplicada, a fim de prosseguir com uma sintonia precisa e confiante. Nesse experimento utilizou-se um Arduino DUE, uma plataforma baseada em um microcontrolador de 32*bits*, o *Atmel*® SAM3X8E ARM® Cortex®-M3 que se deu devido a essa plataforma obter medidas compatíveis com projeto da PCB aqui proposta. Além disso, é uma plataforma do tipo ARM® utilizada devido ao seu poder de processamento, funções de interrupções, dentre outras.

Para utilização dos drivers dos motores, pode-se utilizar na PCB desenvolvida

diversas plataformas com o fator de forma Arduino, assim como a GR-PEACH, nas quais vão estar diretamente ligadas aos 4 drivers dos motores, além da plataforma PSoC® que será utilizada pelo nível reativo, na qual será detalhada nas próximas secções.

As conexões lógicas são projetadas para interface com sistemas de 1.8V a 5V (5.5V máx.). Por padrão, o driver está em nível lógico baixo e o terminal SLP deve ser acionado ou conectado a uma tensão lógica para habilitar o acionamento do módulo. Em uma configuração típica, padrão para utilização dos drivers dos motores, do tipo a utilizada para a obtenção dos dados de tensão (Volts) e velocidade (RPM), apenas dois outros terminais de cada driver são necessários, o PWM (*speedPin*) e DIR (*direction*).

Para verificação dos dados do encoder acoplado aos motores, levou-se em consideração os seguintes dados do fabricante:

- Tensão nominal: 12V;
- Velocidade sem carga: 624RPM;
- Relação de transmissão (RT): 19.225:1;
- Ciclos por Revolução (CPR) no eixo do motor: 3;
- Ciclos por Revolução (CPR) no eixo de saída: 57.667;
- Eventos contáveis por revolução (ECR) no eixo do motor: 12;
- Eventos contáveis por revolução (ECR) no eixo de saída: 230.7;

Para verificar se de acordo com a tensão aplicada indicada pelo fabricante obtém-se a velocidade em RPM como informavam em sua folha de dados, foi utilizada a informação de Ciclos Por Revolução (CPR) no eixo de saída, com um valor de 57.667, visto que esse valor é obtido utilizando a Equação 4.15, fazendo-se:

$$RT = \frac{ECR_{saída}}{ECR_{motor}} \tag{4.14}$$

Quando,

$$CPR_{(saída)} = CPR_{(motor)} \cdot RT \tag{4.15}$$

Esses valores dos Ciclos por Revolução no eixo de saída, ou seja, 57.667 será mantido na programação. O valor de CPR do eixo de saída será utilizado como referência para execução do código no MATLAB® utilizando um *script* com a função *rotaryEncoder()*, como será detalhado na próxima subsecção onde o sistema será identificado.

4.8.1 Identificação do Sistema

O campo de identificação de sistemas usa métodos estatísticos para construir modelos matemáticos de sistemas dinâmicos a partir de dados medidos [143]. Esse processo, também inclui o design ideal de experimentos para gerar dados informativos eficientemente para a montagem do modelo. Um método comum é partir de medições do comportamento do sistema e das influências externas, referente às suas entradas e saídas e determinar uma relação matemática entre eles sem entrar em muitos detalhes do que realmente está acontecendo no sistema.

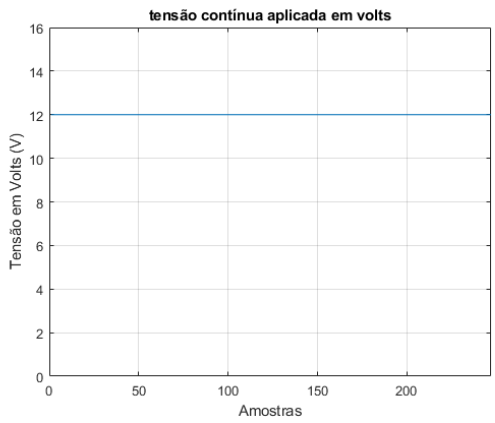
Para obter do modelo do motor DC, as amostras obtidas foram consideradas para fins de identificação do modelo do motor e obtenção da função de transferência através do MATLAB, utilizando a ferramenta de identificação do sistema (do inglês, *System Identification Tools*). Esse software é acessado através da área de trabalho principal do MATLAB®, através do comando *systemIdentification*.

Com esse *ToolBox*, é possível criar o modelo de sistemas dinâmicos lineares e não lineares a partir de dados medidos de entrada e saída. É utilizado também para obtenção de modelos matemáticos a partir de informações de entrada e saída de um sistema, no qual não podem ser modelados por métodos convencionais.

Neste trabalho, optou-se por aplicar a tensão nominal dos motores, ou seja, de 12V (+3.3V de referência do PWM quando utilizada a plataforma Arduino citada) para obter a velocidade em RPM num período de tempo de 10 segundos, tanto para obter os dados, quanto para se adquirir um número significativo de amostras durante o estado em regime permanente do motor.

Utilizando-se do *ToolBox*, foram definidos dois vetores com dados de entrada, considerando a tensão [V] nominal contínua dos motores em 12 Volts, como pode ser observado na ilustração da Figura 70.

Figura 70 – Amostra da tensão DC.

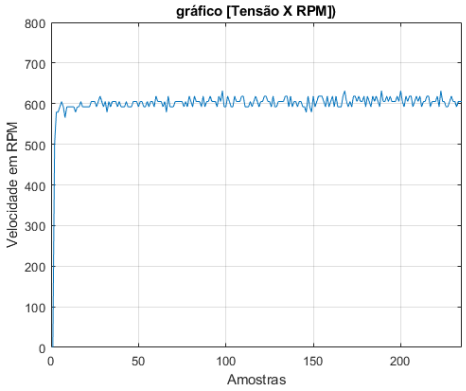


Fonte: Do autor.

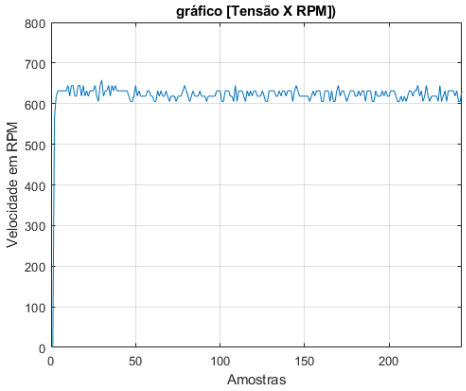
Com isso, foi obtida a saída de velocidade em [RPM] com o valor alvo (*setpoint*) mantido em 624 RPM. As Figuras 71 *a*), *b*) e *c*) ilustram graficamente os resultados obtidos para os três motores, respectivamente.

Figura 71 – Amostras obtidas dos motores.

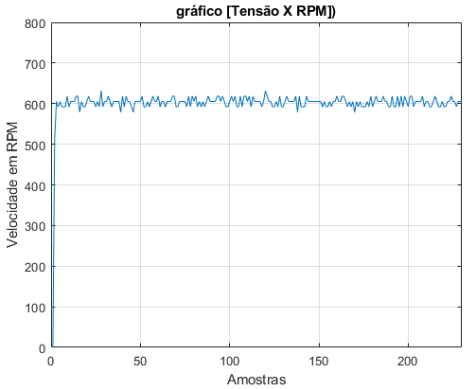
(a) motor 1.



(b) motor 2.



(c) motor 3.



Fonte: Do autor.

Foram percebidas pequenas variações nas amostras com valores oscilando, porém condizentes e dentro do limite estabelecido pelo fabricante obedecendo a sua margem de erro (±62 RPM). Os valores médios das velocidades são descritos na Tabela 11 a seguir.

Tabela 11 – Valores médios obtidos em RPM com MATLAB.

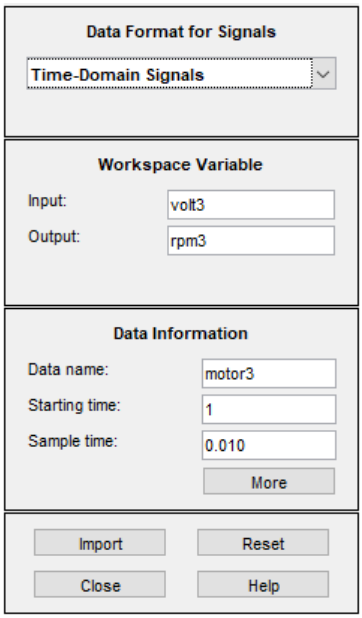
Tensão(V)	**Motor 1 (RPM)**	**Motor 2 (RPM)**	**Motor 3 (RPM)**
12	618.42	631.58	618.42

Fonte: Do autor.

De posse dos vetores e dos dados neles contidos obtidos através da porta serial, iniciou-se o processo de estimação das funções de transferência dos motores com o *systemIdentification* no MATLAB, utilizando a versão 2018b. Os seguintes procedimentos

para identificação do sistema consistiram em selecionar dados no domínio do tempo na aba *import data*, indicada pela janela na ilustração da Figura 72.

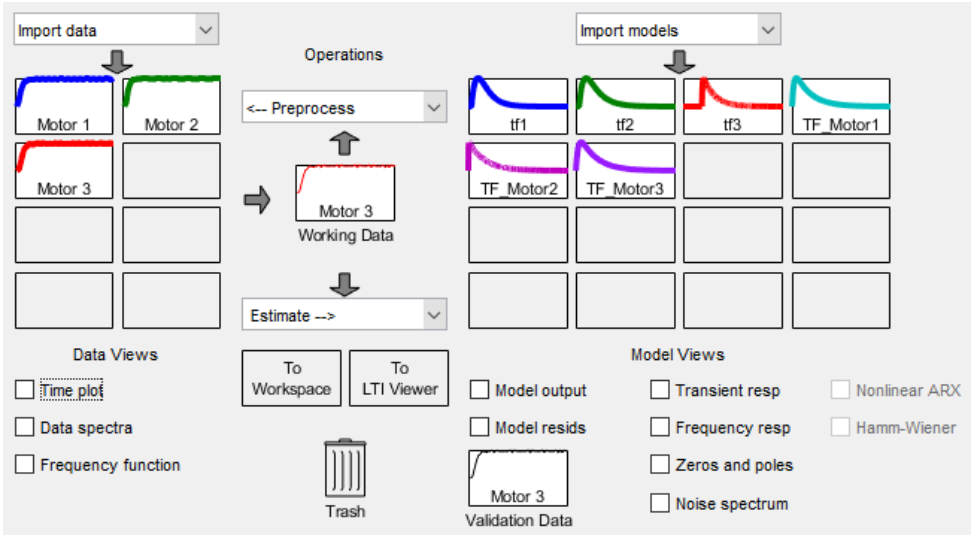
Figura 72 – Dados importados destinados às análises.



Fonte: Do autor.

Nesse processo, pôde-se importar os dados de entrada $[Volts]$ e de saída $[RPM]$ do sistema real definindo-se um o tempo de amostragem de 20ms a 500 amostras que serão utilizados para identificação do sistema, a fim de obter o modelo de melhor ajuste para o motor. Nas janelas indicadas pela ilustração da Figura 73 são ilustrados os modelos obtidos.

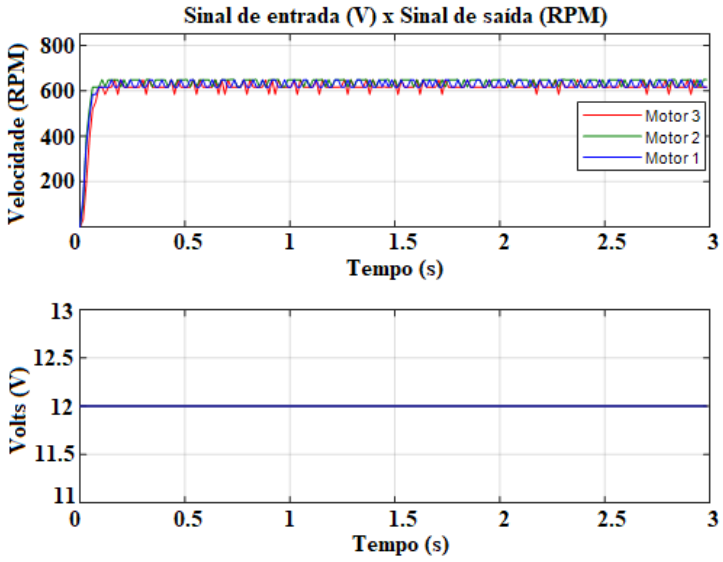
Figura 73 – Interface do *System Identification Tool*.



Fonte: Do autor.

O gráfico dos dados é visualizado ao marcar a opção *"Time plot"*. Após realizar a aquisição de dados de validação, iniciou-se o procedimento de importação dos dados de estimação, nomeados como: *motor*1, *motor*2 e *motor*3. A Figura 74 ilustram os gráficos dos modelos de cada motor e de suas funções de transferência obtidas.

Figura 74 – Modelos gerados.



Fonte: Do autor.

Após o processo de estimação, foi realizado a de identificação da função de transferência, no qual foi exibido no progresso de identificação dos dados dos motores e o percentual de estabilidade aplicada ao modelo da função de transferência estimada. Os resultados foram obtidos para uma função de transferência de primeira ordem com 1 polo e o ajuste aos dados da estimativa de estabilidade aplicada de cada motor são apresentados na Tabela 12, onde mostra os valores da covariância entre os dados de entrada [Volts] e de saída [RPM] das amostras. A covariância, ou variância conjunta é uma medida que busca avaliar o modo com que as duas variáveis se inter-relacionam linearmente, ou seja, como a tensão aplicada varia em relação a uma determinada variação de velocidade.

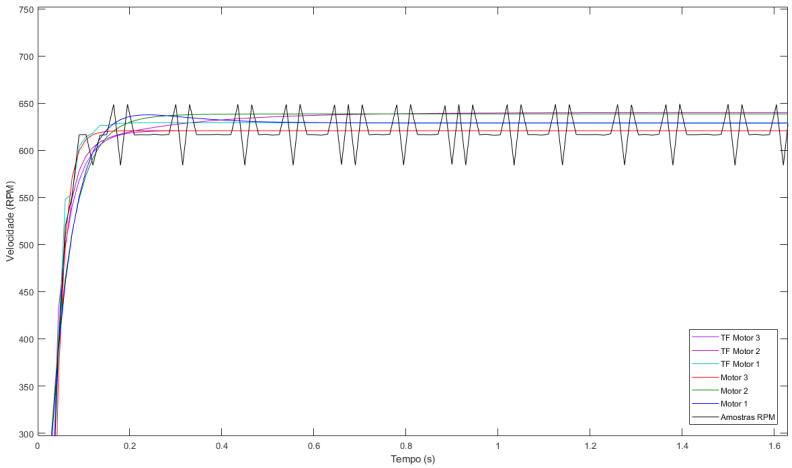
Tabela 12 – Estabilidade aplicada.

Motor 1(TF1)	**Motor 2(TF2)**	**Motor 3(TF3)**
78.86%	77.83%	79.58%

Fonte: Do autor.

Nesse processo de identificação, notou-se claramente que a resposta do modelo possui uma característica semelhante à resposta obtida pelos codificadores (*encoders*). Com isso, esses modelos foram validados para representar os motores.

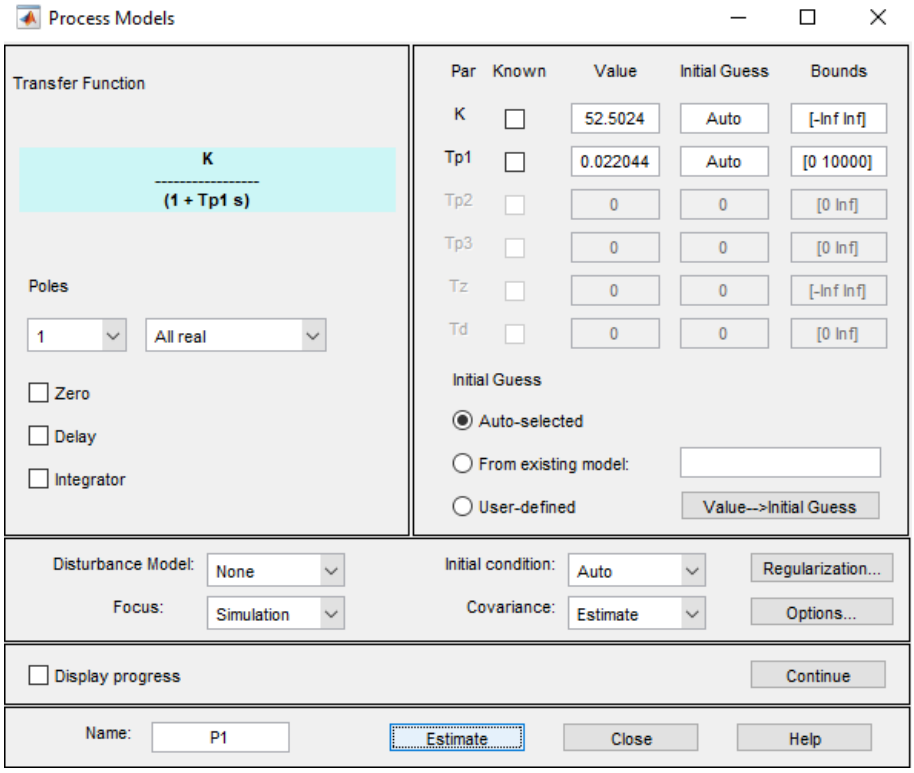
Figura 75 – Respostas dos modelos medidos e simulados.



Fonte: Do autor.

Com base na identificação realizada, foi obtido o modelo de primeira ordem da função de transferência de tempo contínuo do motor DC para 1 polo como ilustra a janela de modelo do processo na Figura 76.

Figura 76 – Modelo do processo.



Fonte: Do autor.

Com base nas informações desses dados, foram obtidas as funções de transferência com valores aproximados para cada motor do tipo:

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{K}{(1 + p1s)}[\frac{RPM}{V}] \tag{4.16}$$

Função de Tansferência do motor 1:

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{52.50}{(1 + 0.022s)}[\frac{RPM}{V}] \tag{4.17}$$

Função de Tansferência do motor 2:

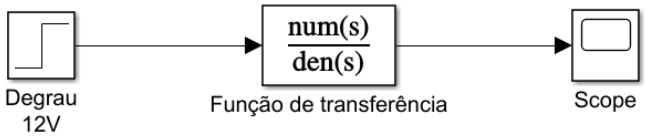
$$G(s) = \frac{\omega(s)}{V(s)} = \frac{53.35}{(1 + 0.020s)}[\frac{RPM}{V}] \tag{4.18}$$

Função de Tansferência do motor 3:

$$G(s) = \frac{\omega(s)}{V(s)} = \frac{51.72}{(1 + 0.028s)}[\frac{RPM}{V}] \tag{4.19}$$

Após a obtenção das funções de transferência aplicou-se um degrau unitário através do software Simulink®. Esse software é uma ferramenta para modelagem, simulação e análise de sistemas dinâmicos. Possui uma interface que permite obter diagramas gráficos através bibliotecas e blocos customizáveis. Com isso, também é possível a integração com o próprio ambiente do MATLAB®, onde é amplamente utilizado em processamento digital de sinais e teorias de controle para projetos de simulação multi-domínios. A Figura 77 ilustra o diagrama em blocos.

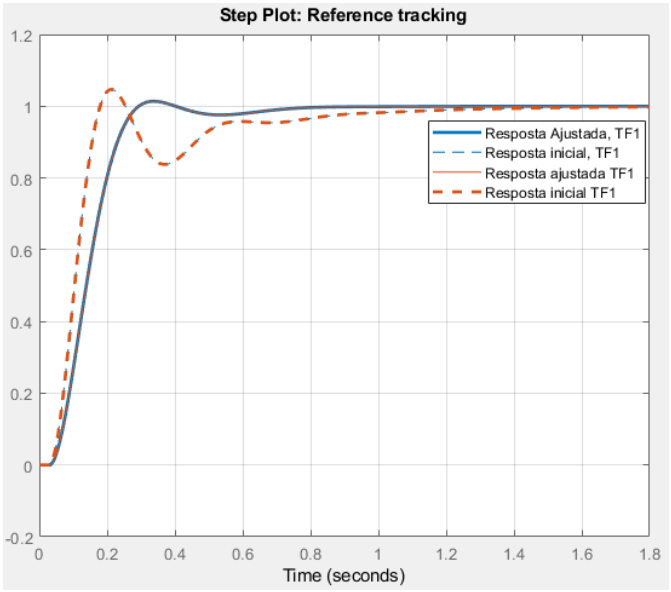
Figura 77 – Modelo no Simulink.



Fonte: Do autor.

Com o aparecimento dos computadores digitais, foi possível a aplicação da simulação dinâmica desde o projeto até a operação do sistema, integrando os estudos de viabilidade dos processos e a utilização de técnicas para obtenção de modelos matemáticos de forma prática, relacionando as entradas com as saídas de um determinado sistema. O correto ajuste desses parâmetros levando-se em consideração a margem encontrada, implica em uma resposta do sistema ajustada para obter a saída desejada, permitindo obter um processo o mais eficiente possível. A Figura 78 ilustra o gráfico de ajuste.

Figura 78 – Gráfico do ajuste no PID *Tuner* do MATLAB.



Fonte: Do autor.

A Figura 79 a seguir ilustra os parâmetros de controle e performance antes e após os ajustes no PID *Tuner*.

Figura 79 – Parâmetros obtidos antes e após o alinhamento das constantes.

Controller Parameters

	Tuned	Baseline
Kp	0.017678	0.033457
Ki	0.12739	0.12255
Kd	n/a	n/a
Tf	n/a	n/a

Performance and Robustness

	Tuned	Baseline
Rise time	0.161 seconds	0.0982 seconds
Settling time	0.605 seconds	0.949 seconds
Overshoot	1.38 %	4.69 %
Peak	1.01	1.05
Gain margin	13.2 dB @ 24.8 rad/s	9.22 dB @ 27.2 rad/s
Phase margin	66 deg @ 7.17 rad/s	60 deg @ 11.9 rad/s
Closed-loop stability	Stable	Stable

Fonte: Do autor.

Por fim as constantes PI de cada motor foram obtidas para projeto do controlador através do *PID Tuner* no software MATLAB®, onde são descritos na Tabela 13 os valores

dessas constantes antes e após o alinhamento. De posse dessas condições foram listados os parâmetros mais apropriados para o controlador.

Tabela 13 – Constantes PI obtidas para os motores de 624RPM.

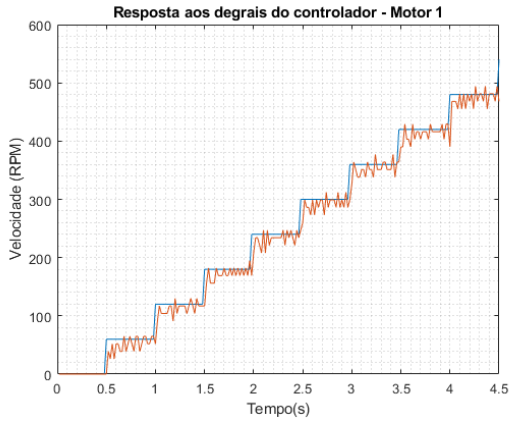
	PIDTune (automático)		**PIDTuner (ajustado)**	
	Kp	Ki	Kp	Ki
Motor 1	0,003345	0,12255	0,017678	0,12739
Motor 2	0,022654	0,15691	0,034885	0,12699
Motor 3	0,005712	0,32606	0,006357	0,33257

Fonte: Do autor.

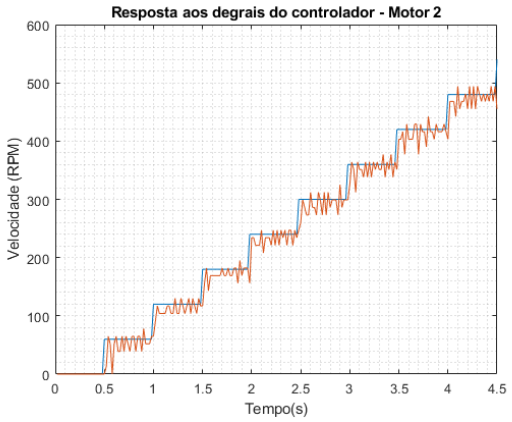
As constantes obtidas foram utilizadas no controlador baseando-se em [29] e a validação foi realizada com a estrutura completa do robô com os motores funcionando sem carga utilizando-se um *script* do MATLAB® para obtenção da resposta ao degrau vista na Figura 80 a), b) e c) a seguir.

Figura 80 – Respostas aos degraus dos motores 1, 2 e 3.

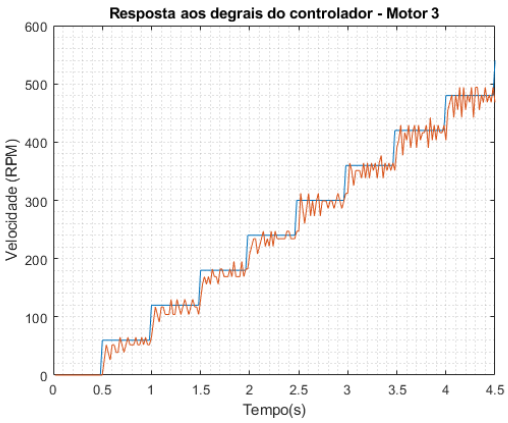
(a) Motor 1.



(b) Motor 2.



(c) Motor 3.



Fonte: Do autor.

4.9 Modelagem do robô

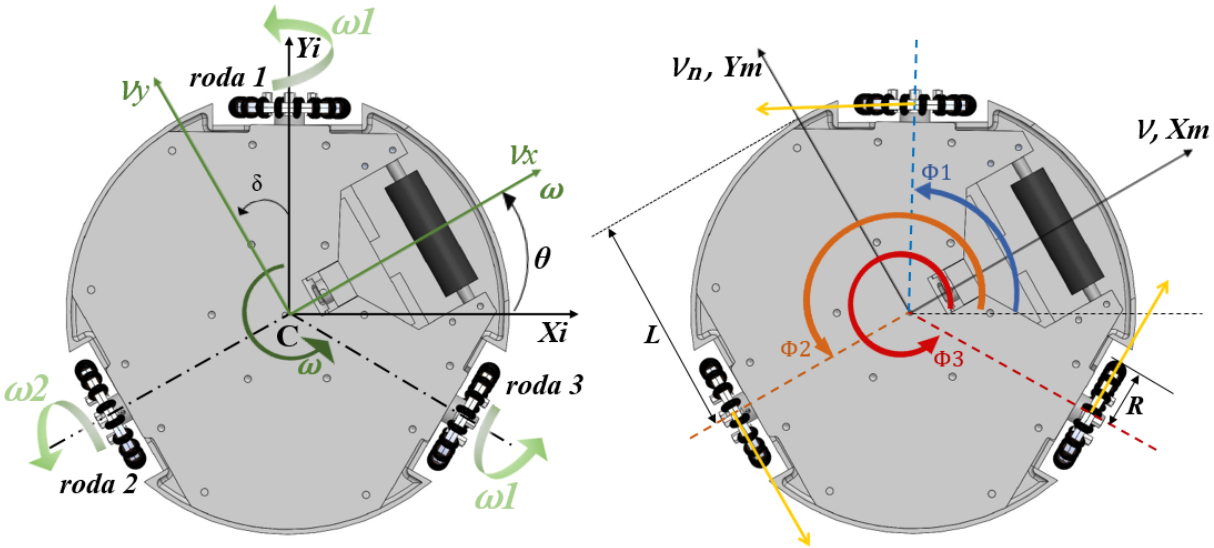
Conceitos de natureza elétrica e mecânica e a organização das partes físicas que constituem uma estrutura robótica são aspectos extremamente importantes na implantação de seus controladores, e, devido a isso, a eficácia da movimentação e as equações que regem o comportamento de um robô são influenciadas pelas questões relacionadas a sua geometria.

Nesta seção, se descreverá como o modelo cinemático do novo robô móvel omnidirecional desenvolvido será utilizado no nível reativo composto pelo controlador cinemático, no qual servirá de base para a implementação dos demais comportamentos aqui descritos.

4.9.1 Modelo cinemático

O modelo cinemático é obtido a partir da relação geométrica entre um sistema de referência global $[Xi,Yi]$ e um sistema de referência da estrutura do robô $[Xm,Ym]$. Estes sistemas de referência para o robô omnidirecional AxeBot II, cuja base é mostrada na Figura 81, é composto por três rodas omnidirecionais dispostas simetricamente a 120º uma da outra, onde cada roda é conduzida por um motor DC e tem uma mesma distância L do centro de massa C do robô para o centro da roda, R é o raio da roda. O ângulo θ indica a orientação do robô, que é o ângulo em direção ao eixo Xm no sistema de coordenadas global. O ângulo Φ indica a direção da velocidade de translação do robô. Sabe-se também que existe uma rotação entre o sistema de coordenadas do centro de massa e o sistema de coordenadas do referencial inercial.

Figura 81 – Sistemas de Coordenadas do AxeBot II para modelagem cinemática.



Fonte: Do autor.

- Xi, Yi, θ - Posição do robô $[Xi, Yi]$ e ângulo θ de orientação do robô;

- ν_x e ν_y - Velocidade linear das rodas dados em $[m/s]$;

- $\omega 1$, $\omega 2$, $\omega 3$ - Velocidade angular das rodas dados em [rad/s];

A posição e orientação da base em relação ao eixo global são encontradas integrando as equações matriciais de movimento em relação ao tempo. As velocidades do centro de massa (base) são calculadas a partir das velocidades das rodas, como as velocidades linear e angular utilizando as matrizes do modelo cinemático. No entanto, temos de analisar robô no espaço, calcular sua trajetória e derivar a velocidade de cada roda individualmente, no qual é representada pelos vetores de rotação e translação da estrutura do robô, ilustrado na Figura 81.

A partir dessa geometria e dos sistemas de coordenadas inseridos, as equações foram obtidas agrupando-se as velocidades de cada roda individualmente no vetor $(\nu_1, \nu_2, ...\nu_n)^T$ e a velocidade tangencial de rotação do robô no vetor (v_x, v_y, ω). A Equação 4.20 matricial baseada em [144] será referenciada para obter o modelo cinemático de um robô com $(R_{1,2,...,n})$ n-*ésima* rodas.

$$\begin{pmatrix} \nu_1 \\ \nu_2 \\ \vdots \\ \nu_n \end{pmatrix} = \begin{pmatrix} -sen(\Phi_1) & cos(\Phi_1) & 1 \\ -sen(\Phi_2) & cos(\Phi_2) & 1 \\ \vdots & \vdots & \vdots \\ -sen(\Phi_n) & cos(\Phi_n) & 1 \end{pmatrix} \begin{pmatrix} \nu_x \\ \nu_y \\ \omega \end{pmatrix} \tag{4.20}$$

De posse dessa equação e dos ângulos $\Phi_1 = 90\,^\circ$, $\Phi_2 = \Phi_1 + 120\,^\circ$ e $\Phi_3 = \Phi_2 + 120\,^\circ$ determinados para orientação inicial do robô e sendo: L a distância entre o centro de massa e as rodas, $R_{1,2,3.}$ o raio das rodas e ω_i (para $i = 1, 2, 3.$) a velocidade de rotação, obtem-se a Equação 4.21.

$$\begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{pmatrix} = \frac{1}{R} \begin{pmatrix} -sen(\Phi_1) & cos(\Phi_1) & L \\ -sen(\Phi_2) & cos(\Phi_2) & L \\ -sen(\Phi_3) & cos(\Phi_3) & L \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ \omega \end{pmatrix} \tag{4.21}$$

Daí, a matriz M resultante é dada por:

$$M = \frac{1}{R} \begin{bmatrix} -sen(\frac{\pi}{2}) & cos(\frac{\pi}{2}) & L \\ -sen(\frac{7\pi}{6}) & cos(\frac{7\pi}{6}) & L \\ -sen(\frac{11\pi}{6}) & cos(\frac{11\pi}{6}) & L \end{bmatrix} \tag{4.22}$$

4.9.2 Cinemática direta

De acordo com a ilustração da Figura 81, a cinemática direta do robô pode ser expressa através da Equação 4.23 matricial como:

$$\begin{bmatrix} \nu_x \\ \nu_y \\ \omega \end{bmatrix} = \frac{R}{3} \begin{bmatrix} cos(\pi) & sen(\frac{\pi}{6}) & sen(\frac{5\pi}{6}) \\ 0 & sen(\frac{4\pi}{3}) & cos(\frac{\pi}{6}) \\ L & L & L \end{bmatrix} \begin{bmatrix} \omega 1 \\ \omega 2 \\ \omega 3 \end{bmatrix} \tag{4.23}$$

O modelo cinemático direto completo é apresentado na Equação 4.24.

$$\begin{bmatrix} \nu_x \\ \nu_y \\ \omega \end{bmatrix} = M \begin{bmatrix} \omega 1 \\ \omega 2 \\ \omega 3 \end{bmatrix} \tag{4.24}$$

4.9.3 Cinemática inversa

Para o cálculo da cinemática inversa, a matriz cinemática direta **P** pode ser invertida para fornecer a Equação 4.25 matricial.

$$\begin{bmatrix} \omega 1 \\ \omega 2 \\ \omega 3 \end{bmatrix} = M^{-1} \begin{bmatrix} \nu_x \\ \nu_y \\ \omega \end{bmatrix} \tag{4.25}$$

Controlador Cinemático

De propriedade do modelo cinemático, é possível aplicar este modelo para fazer um controlador que possibilize realizar um rastreamento de trajetória e estabelecer o robô em uma determinada posição esperada (ponto-a-ponto).

De posse do controlador, baseando-se em [52] é possível implementar controle de velocidade ou de posição. Com isso, será referido o controle de posição, expondo mais adiante como adaptar este mesmo controlador para efetuar o controle da velocidade. A entrada do controlador de posição é a trajetória ou a posição desejada, isto é, o *setpoint*.

Sendo assim, o controlador calcula o vetor de diferença entre a posição real do centro de massa do robô (Xi, Yi e θ) e a posição de referência (X_i^C, Y_i^C e θ^C). Com isso, a execução de controle gera o vetor de erro, como na Equação 4.26.

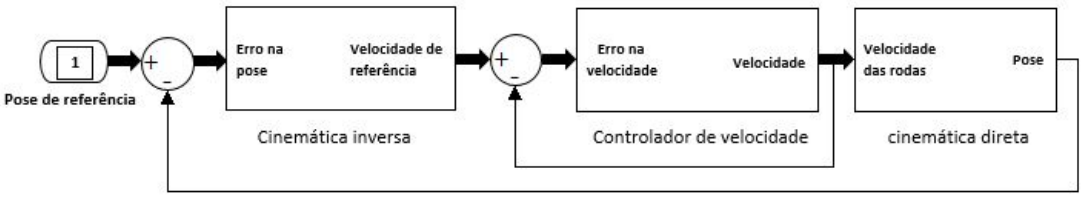
$$e(t) = \begin{bmatrix} X_i^C \\ Y_i^C \\ \theta^C \end{bmatrix} - \begin{bmatrix} X_i(t) \\ Y_i(t) \\ \theta(t) \end{bmatrix} \tag{4.26}$$

Segundo [145], para o controle ponto-a-ponto, a ação de controle é calculada de acordo com a Equação 4.27, onde K_P e K_I são matrizes 3 x 3 simétricas e definidas. Este vetor de erro é usado para calcular uma ação de controle Proporcional-Integral (PI).

$$u(t) = K_P e(t) + K_I \int_0^t e(\gamma) d\gamma \tag{4.27}$$

O diagrama em blocos do controlador é apresentado na Figura 82.

Figura 82 – Diagrama do controlador cinemático.



Fonte: Baseado em [29].

Para o rastreamento de trajetória, levou-se em consideração o vetor referência $p^T(t) = [X_i^C(t); Y_i^R(t); \theta^C(t)]$ que depende do tempo e consequentemente a ação do controle é modificada pela adição da derivada do vetor de referência, conforme a Equação 4.28.

$$u(t) = K_P e(t) + K_I \int_0^t e(\gamma) d\gamma + \dot{p}(t) dt \tag{4.28}$$

Esse vetor de referência é aplicado na Equação 4.29 da cinemática inversa para a obtenção das velocidades angulares desejadas, ou seja, o *setpoint* dos atuadores. As velocidades das rodas devem ser ajustadas aos valores obtidos para que a posição da base seja corrigida.

$$\begin{bmatrix} \dot{\Phi}_1 \\ \dot{\Phi}_2 \\ \dot{\Phi}_3 \end{bmatrix} = M^{-1}(\theta) u(t). \tag{4.29}$$

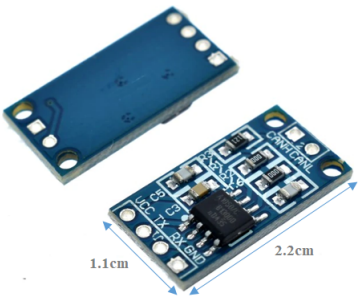
Em seguida a velocidade real das rodas é obtida pelos *encoders* e finalmente aplicadas a Equação 4.23 da cinemática direta, resultado-o nas velocidades lineares e angular e consequentemente a posição do robô. A realimentação efetuada pelo controlador é responsável pelo *loop* para novo cálculo do erro e continuação do ciclo. Esse controlador desenvolvido em [52] foi utilizado como base para implementação dos comportamentos do nível reativo no PSoC® 5L com parâmetros dimensionais readaptados para a versão atual do novo robô AxeBot desenvolvido nesse trabalho. Estes comportamentos integraram para do movimento nas direções cardeais: norte, nordeste, leste, sudeste, sul, sudoeste, oeste e noroeste.

4.10 Configuração do robô para embarque do AAC

Uma arquitetura contendo três microcontroladores e um módulo de comunicação, porém com algumas diferenças da projetada anteriormente em [29] compõe um hardware configurado para comportar o agente. A arquitetura de microcontroladores atual, muda as características do sistema anteriormente citado, ou seja, não foi utilizado Ethernet, pois somente foram utilizados dois barramentos CAN conectados entre as três principais plataformas.

Para o projeto foram utilizados 4 módulos compostos pelo transceptor da *NXP semiconductors* com o circuito TJA1050 como ilustrado na Figura 83. O transceptor fornece capacidade de transmissão diferencial para o barramento e capacidade de recepção diferencial para o controlador CAN. Esse transceptor possui características importantes para o projeto como: Compatibilidade com o padrão ISO11898, Emissão Eletromagnética (EME) baixa e possui níveis lógicos de entrada compatíveis com plataformas de 3.3V e 5.0V.

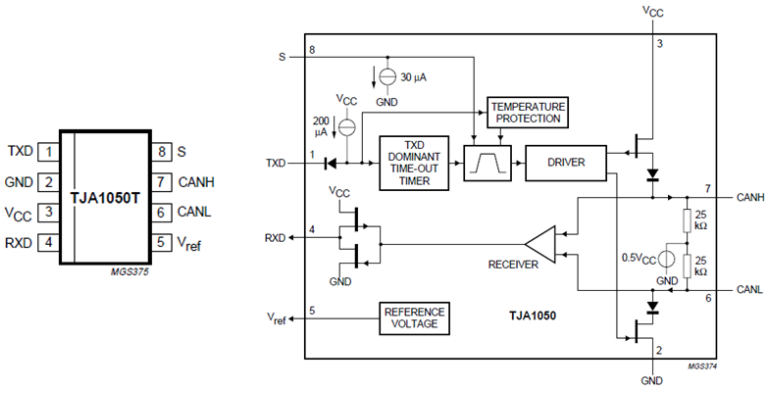
Figura 83 – Módulo composto pelo transceptor TJA1050 da *NXP*.



Fonte: [146].

A Figura 84 mostra o *pinout* e o diagrama do circuito integrado $TJA1050$.

Figura 84 – Terminais e diagrama do TJA1050 da *NXP*.



Fonte: [147].

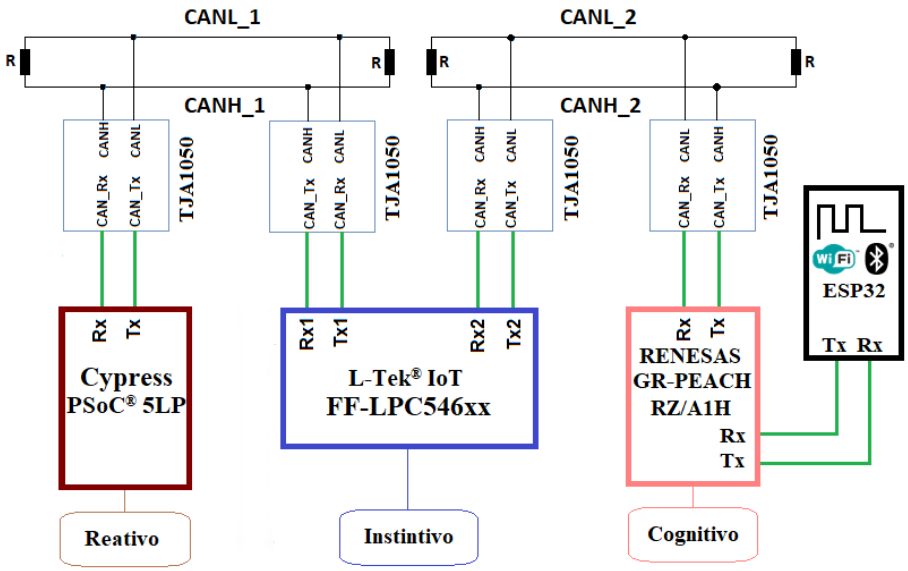
Tabela 14 – Funções dos terminais do TJA1050.

Referência	Terminal	Descrição
TXD	1	transmite entrada de dados
GND	2	terra
Vcc	3	tensão de alimentação
RXD	4	recebe saída de dados
Vref	5	tensão de referência de saída
CANL	6	barramento CAN de nível baixo
CANH	7	barramento CAN de nível alto
S	8	seleção do modo de entrada

Fonte: [147].

Algumas mudanças de conexão a nível de hardware foram feitas, na qual serão utilizados somente barramento CAN composto por esses transceptores para conexão dos três microcontroladores que compõem a rede. O transceptor TJA1050 é a interface entre o controlador de protocolo CAN e o barramento físico, como ilustra na Figura 85 a seguir, caracterizando a nova arquitetura configurada para embarque do AAC.

Figura 85 – Configuração do barramento CAN proposto.



Fonte: Do autor.

Com essa nova configuração de comunicação, contendo dois barramentos CAN separadamente, permite-se utilizar duas taxas de comunicação distintas entre seus níveis, visto que a frequência da troca de mensagens entre o nível instintivo e reativo é mais alta do que a frequência entre os níveis instintivo e o cognitivo.

Conforme pode-se observar, os nós computacionais da rede são compostos e configuradas pelas seguintes plataformas de desenvolvimento conectadas aos dois barramentos,

descritas a seguir:

- PSoC® 5L da *Cypress Semiconductors*: ARM® Cortex®-M3;
- mbed IoT/FF-LPC546xx da L-Tek®: ARM® Cortex®-M4;
- GR-PEACH-RZ/A1H da RENESAS: ARM® Cortex®-A9.

Do projeto desenvolvido em [29], foram substituídas duas plataformas, devido as mesmas possuírem algumas limitações de software e hardware. Naquele projeto, considerado uma rede heterogênea por utilizar duas interfaces de comunicação digital diferentes entre seus dois barramentos: uma rede CAN (*Controller Area Network*) conecta o mbed LPC1768 e o PSoC® 5LP, e uma rede *Ethernet* conecta o mbed LPC1768 e o *DIL-NetPC* DNP 2486, da *SSV Embedded Systems*.

No presente trabalho, um barramento CAN conecta o *mbed* L-Tek® e o PSoC® 5LP, e outro barramento conecta o *mbed* L-Tek® e a GR-PEACH, no qual também possui suporte *mbed*.

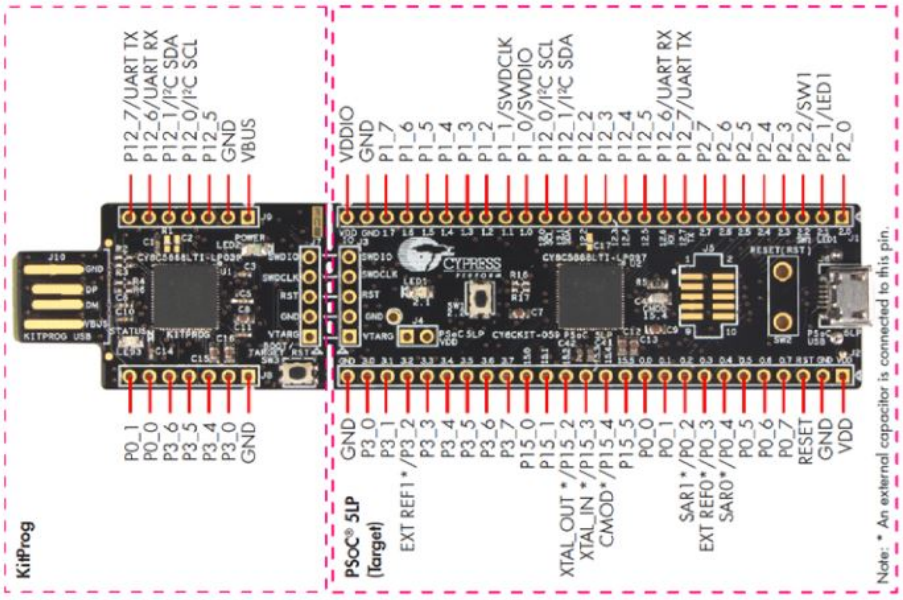
Essa nova rede foi projetada desta forma para permitir que o AAC seja embarcado sem que haja mudança em suas características de comunicação entre seus níveis de abstração com relação ao projeto anterior. Neste projeto, não será utilizado o protocolo Ethernet, devido a custos e disponibilidade, nos quais são dois dos principais fatores. No domínio em que a CAN se destaca, não é necessário que o sistema seja dotado de comunicações de altíssimas velocidades, como 10, 100 ou 1000 Mbit/s. Além disso, embora a Ethernet esteja se tornando comum em microcontroladores, ela ainda requer um transceptor externo muito mais caro, e este, é normalmente conectado a um comutador, não a um barramento, sendo necessário a adição de mais um dispositivo, o qual tornaria o circuito mais complexo.

De acordo com [29], o agente autônomo requer concorrência, esta concorrência só pode ser alcançada perfeitamente se cada nível tiver um núcleo computacional dedicado à sua execução.

4.10.1 Nível Reativo: A Plataforma PSoC® 5LP

PSoC® é a abreviação de (*Programmable System-on-Chip*), ou seja, Sistema em um Chip programável. Esse sistema permite desenvolver modelos com dados periféricos integrados ao microcontrolador, semelhantes aos utilizados em Dispositivos Lógicos Programáveis com microcontroladores mais atuais. O kit de prototipagem CY8CKIT-059 PSoC® 5LP ilustrado na Figura 86 apresenta a plataforma composta pelo circuito integrado CY8C5888LTI-LP097 que utiliza uma CPU ARM® Cortex®-M3, no qual é o microcontrolador que irá compor o barramento da rede onde o nível reativo do AAC foi embarcado.

Figura 86 – Plataforma PSoC® 5LP e *pinout*.



Fonte: [148].

Algumas das principais características do PSoC® 5LP são listadas na Tabela 15.

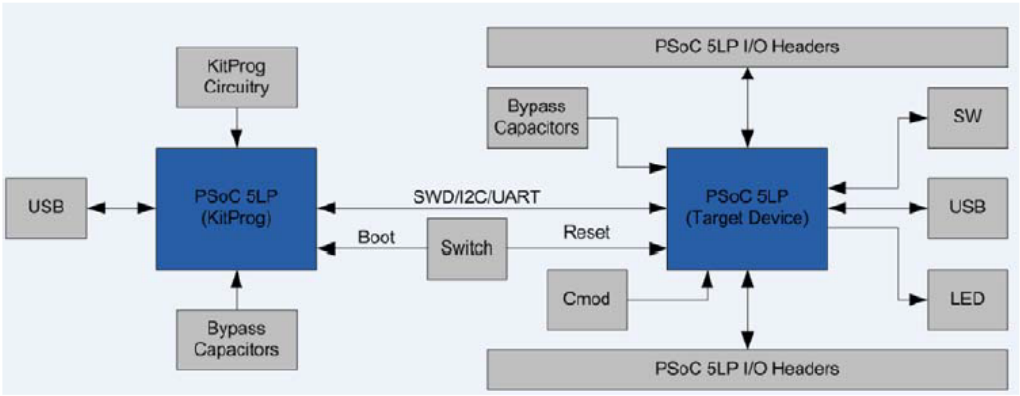
Tabela 15 – Principais Características da PSoC® 5L.

Recurso	Características
Microcontrolador	ARM® Cortex®-M3 de 32bits, 1.25DMIPS/MHz
Memória FLASH de programa	256KB
Memória FLASH adicional	32KB
Memória memória RAM	64KB
Memória EEPROM	2kB

Fonte: Do autor.

A plataforma PSoC® 5LP vem configurado como USB-UART no CY8CKIT-059, de modo a prover uma porta de comunicação COM, utilizada como mecanismo de *bootloader* via UART por meio do *Bootloader Host* do software *PSoC Creator* ou do *PSoC Programmer*. Na plataforma é possível carregar o código de programa via USB por meio do CY8C5888LTI-LP097. A parte da placa que contém o *kitprog* composto pelo conector USB e o circuito CY8C5868LTI-LP03 pode ser removido, de modo a tornar possível também o desenvolvimento de aplicações voltadas para o dispositivo. A Figura 87 ilustra o diagrama de blocos do PSoC® 5LP.

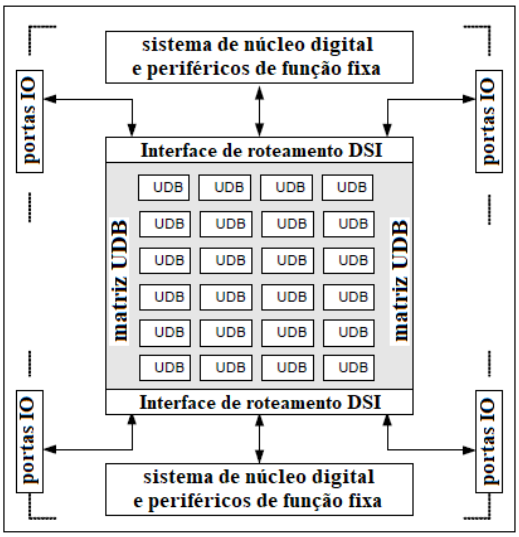
Figura 87 – Diagrama de Blocos do PSoC® 5LP.



Fonte: [148].

Assim como em [29], para esse projeto também manteve-se o PSoC® 5LP como plataforma para embarcar o nível reativo, ao fato de que este microcontrolador, juntamente com a PCB desenvolvida, incorpora uma arquitetura apropriada para possíveis evoluções para o modelo genérico de agentes cognitivos. O diagrama de blocos e a arquitetura digital programável da PSoC® 5LP é mostrada na Figura 88.

Figura 88 – Arquitetura Digital Programável do PSoC®



Fonte: [148].

Nessa arquitetura, os controladores difusos por exemplo, foram implementados nos UDB's (do inglês, *Universal Digital Blocks*) do PSoC®, liberando o processador para tarefas de comunicação, ou seja, nesse subsistema digital composto por blocos, implementam recursos digitais no hardware independentes da execução do processador. Além disso, permitem o desenvolvimento de funções periféricas digitais programáveis, consistindo em

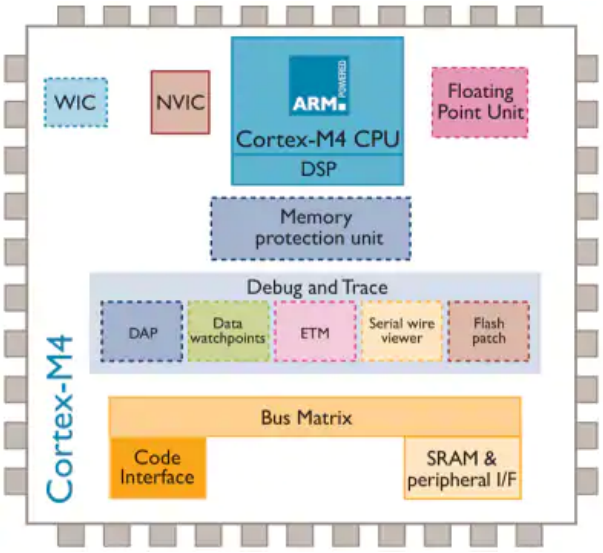
uma combinação de lógica semelhante a Dispositivos Lógicos Programáveis DLP's (do inglês, *Programmable Logic Devices*).

A versão da PCB desenvolvida nesse trabalho, também poderá ser estendida para adaptar-se a outras estruturas robóticas, reprojetando também na arquitetura do nível reativo, utilizado subsistemas analógico e digital do PSoC® para as tarefas de controle ou utilizando outras plataformas para a mesma finalidade.

4.11 A Arquitetura ARM® *mbed*

O *mbed* é fundamentalmente um ambiente de desenvolvimento que é destinado a programação em plataformas estabelecidas com microcontroladores ARM®, que é um acrônimo de *Advanced RISC Machine*. Os microcontroladores ARM® Cortex®-M4 foram desenvolvidos recentemente para atender especificamente a controles de sinais digitais com alta eficiência e baixa potência. A Figura 89 a seguir ilustra um diagrama em blocos da arquitetura ARM® Cortex®-M4.

Figura 89 – Diagrama em blocos da arquitetura ARM® Cortex®-M4.

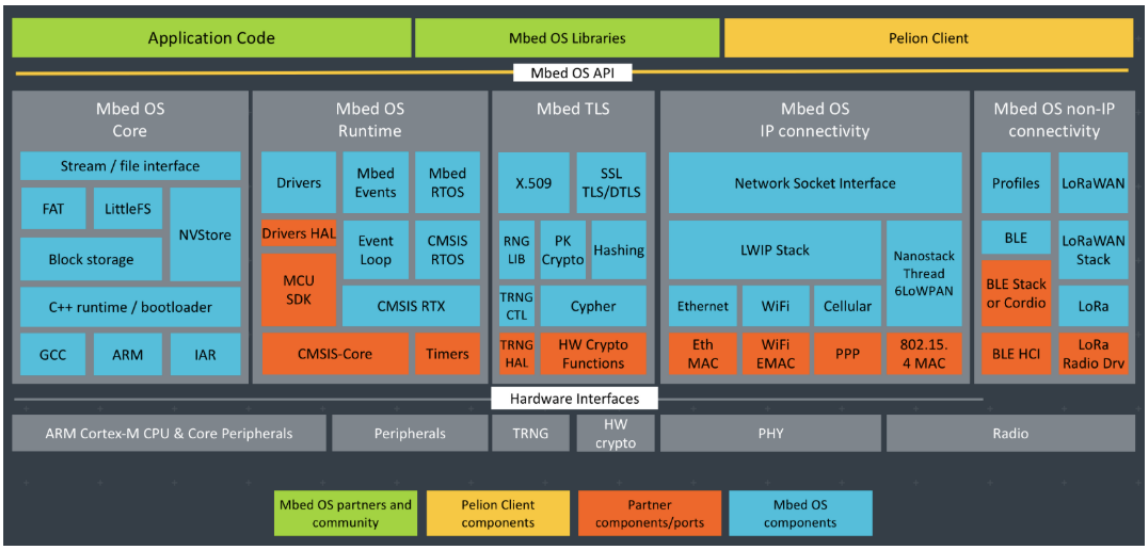


Fonte: [149].

A concepção *mbed* está agora no avanço do desenvolvimento de sistemas de Internet das Coisas *IoT* (do inglês, *Internet of Things*) da ARM® (do inglês, *Advanced Risk Machine*). Isso tem enormes implicações para o conceito *mbed*, pois as ferramentas e processos de design são adaptados às necessidades dos aplicativos *IoT*. Como parte dessa evolução, as necessidades da *IoT* são, até certo ponto, divergentes daquelas do sistema embarcado convencional.

Disponibilizada atualmente para desenvolvimento de *firmware* em ambiente *on-line* e *off-line*, o *mbed* OS utiliza uma camada de abstração de hardware HAL (do inglês, *Hardware Annotation Library* para suportar as partes mais comuns de microcontroladores, como os temporizadores. A Figura 90 a seguir ilustra o diagrama em blocos da arquitetura de uma placa *mbed*.

Figura 90 – Arquitetura básica de uma placa *mbed*.



Fonte: [150].

Essa camada permite a utilização de aplicativos através de uma *API* (do inglês, *Application Program Interface*) a um grupo de interfaces para programação de softwares compostos por bibliotecas e suporte de drivers necessários para periféricos MCU (do inglês, *Micro Controller Unit*) padrão, como I2C, serial e SPI. Além disso, também é utilizada como ambiente inicial à adição de suporte para novas comunicações ou recursos às comunicações existentes.

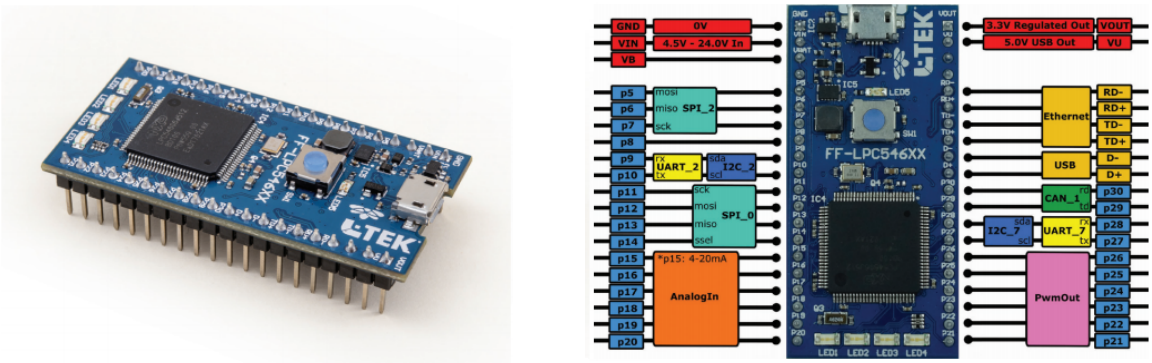
No projeto desenvolvido em [29] foi utilizado uma plataforma composta pelo microcontrolador NXP LPC1768 para o mesmo nível de abstração composto por esse trabalho, baseado em um microcontrolador ARM® Cortex®-M3 como CPU. Essa plataforma poderá também ser utilizada na PCB desenvolvida do presente projeto, devido a possuir as mesmas dimensões e configuração de terminais (*pinout*), semelhantes a plataforma FF-LPC546xx aqui utilizada. Além disso, para as conexões dessa placa na PCB, pode-se utilizar a plataforma mbed LPC11U24, uma ARM® Cortex-M0, também disponível comercialmente. A mudança dessa plataforma para embarque do AAC no projeto se deu devido ao maior poder de armazenamento da memória SRAM da plataforma ARM® Cortex®-M4 L-Tek®, visto que a expansão dessa memória para execução do nível atribuído a ela, traria alguns benefícios tais como solucionar problemas referente a esse quesito enfrentados pelo nível instintivo em [29].

4.11.1 Nível Instintivo: Utilização do ARM® *mbed* L-Tek® FF-LPC54606

A plataforma IoT FF-LPC546xx da L-Tek® utilizado é baseado no microcontrolador LPC54606 da *NXP Semiconductors* de baixa potência. Na Figura 91(b) um diagrama completo de seus terminais (*Pinout*) e seus recursos de comunicação são ilustrados, que por sua vez, utiliza o microprocessador ARM® Cortex®-M4 como CPU (do inglês, *Central Processing Unit*). Esse módulo *mbed* é ilustrado na Figura 91(a) com os seus 40 terminais dispostos em duas linhas de 20 destinados a prototipagem de aplicações de entradas e saídas de propósitos gerais GPIO (do inglês, *General Purpose Input/Output*).

Figura 91 – ARM® *mbed* L-Tek® FF-LPC546xx.

(a) ARM® *mbed L-Tek* ® *IoT FF-LPC546xx* **(b)** diagrama de terminais (*Pinout*)



Fonte: [151].

Da série de microcontroladores *mbed*, essa plataforma suporta alguns periféricos externos como o transceptor Ethernet LAN8720A, USB e memória FLASH serial externa SPI (do inglês, *Serial Peripheral Interface*) AT45DB321E de 32-M*bit on-board* que é um protocolo que permite a comunicação do microcontrolador com outras plataformas e/ou módulos entre si em modo *full duplex* formando a rede.

Algumas das principais características do Microcontrolador NXP LPC54606 são listadas na Tabela 16.

Tabela 16 – Principais Características da L-Tek® IoT/FF-LPC546xx.

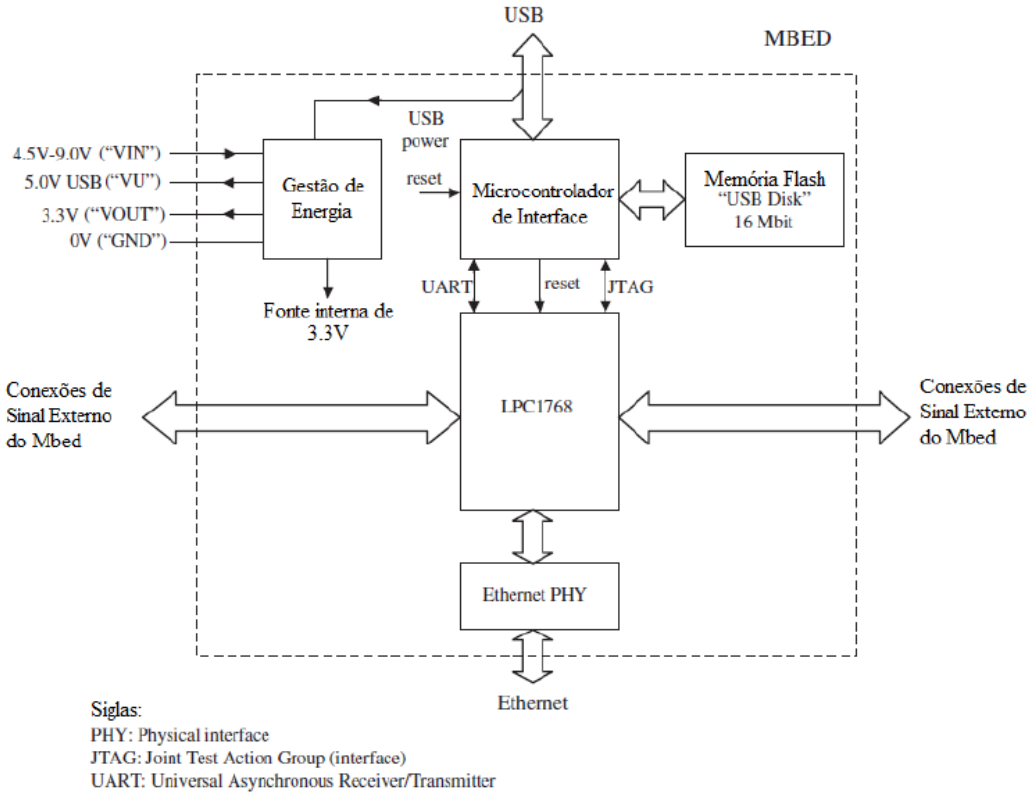
Recurso	**Características**
Microcontrolador	ARM® Cortex®-M4 de 220Mhz
Memória FLASH de programa	512kB
Memória SRAM	200kB
Memória EEPROM	16kB

Fonte: Do autor.

As entradas e saídas digitais do *mbed*, assim como os periféricos que este disponibiliza, são os do microcontrolador LPC1768, plataforma que também poderá ser utilizada

neste projeto, como citado anteriormente neste Capítulo. A Figura 92 apresenta um diagrama de blocos do princípio de funcionamento do *mbed*.

Figura 92 – Diagrama de blocos do *mbed*



Fonte: [84].

Esse ambiente dispõe de um microcontrolador para conexão e gerenciamento da comunicação USB com o PC. Com isso, o microcontrolador embutido faz o PC reconhecer o *mbed* como um dispositivo de armazenamento e gerencia a transferência do arquivo executável *.bin* (extensão de arquivos com formatos de dados binários) para uma memória flash do módulo de 32M*bits* (plataforma atual). Ao utilizar o *reset*, o microcontrolador de interface concede para a memória *flash* da plataforma o arquivo executável, e assim inicia a execução [84].

Com a nova estrutura de hardware do presente projeto, uma das interfaces CAN do módulo *mbed* FF-LPC546xx é utilizada para a comunicação através da rede CAN com o PSoC® 5LP. Com isso, essa plataforma composta pelo nível reativo interage com o *mbed* informações referentes ao estado do ambiente e do agente e recebe comandos de seleção de comportamentos. A outra interface CAN da LTeK® FF-LPC546xx é utilizada para comunicação com o nível cognitivo, na qual está embarcada no GR-PEACH. As tarefas utilizadas na implementação do nível instintivo foram basicamente o envio e recebimento de mensagens pela rede CAN.

Tradicionalmente, o nível instintivo possui um Sistema Baseado em Conhecimento (SBC) que, com base no objetivo local enviado pelo nível cognitivo, nas informações enviadas pelo reativo e em sua base de regras, infere qual comportamento deve ser ativo no nível reativo. Sua base de conhecimento possui uma coleção de bases de regras chamadas planos, e a base de regras atual é selecionada pelo nível cognitivo e seu procedimento de planejamento.

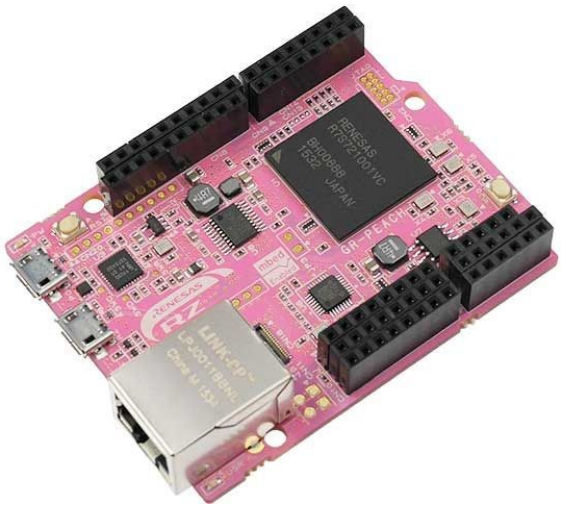
Assim como o trabalho a que este se baseia em [29], o conhecimento foi representado por uma linguagem específica e a inferência foi realizada por um encadeamento de regras. Além disso, a representação de conhecimento usada neste nível foi alterada para execução baseando-se na lógica difusa. Como esse nível é responsável, entre outras tarefas, por coordenar a seleção de comportamentos no nível reativo, um KBS (do inglês, *Knowledge Based System*) difuso pode ser capaz de mesclar comportamentos de maneira difusa ponderada.

4.11.2 Nível Cognitivo: utilização do ARM® *mbed* GR-PEACH - RZ/A1H

O nível cognitivo é onde o planejamento acontece. Esse nível também possui um sistema baseado em conhecimento, mas desta vez é usado como um mecanismo de pesquisa para o algoritmo de planejamento. Um modelo lógico do mundo é mantido e sua base de conhecimento é dividida em local e social, porque esse nível também é responsável por estabelecer comunicação com outros agentes. Esse nível é implementado no GR-PEACH, uma plataforma baseada no microcontrolador ARM® Cortex®-A9 da RENESAS, o R7S721001VCBG da família RZ/A1H.

A Figura 93 ilustra a plataforma.

Figura 93 – ARM® *mbed* GR-PEACH.



Fonte: [152].

A plataforma GR-PEACH suporta até 2 displays com resolução WXGA (1280x800) sem a necessidade de memória externa. Essa plataforma também permite projetos embarcados, como utilizar uma distribuição na versão XIP *Linux Kernel*, possui suporte RTOS, além de ser possível projetar dispositivos de *IoT End Point* inteligentes com alto desempenho.

Além disso, essa plataforma preserva o suporte Ethernet, características que supriria às necessidades de sua utilização em [29]. Algumas das suas principais características são listadas na Tabela 17.

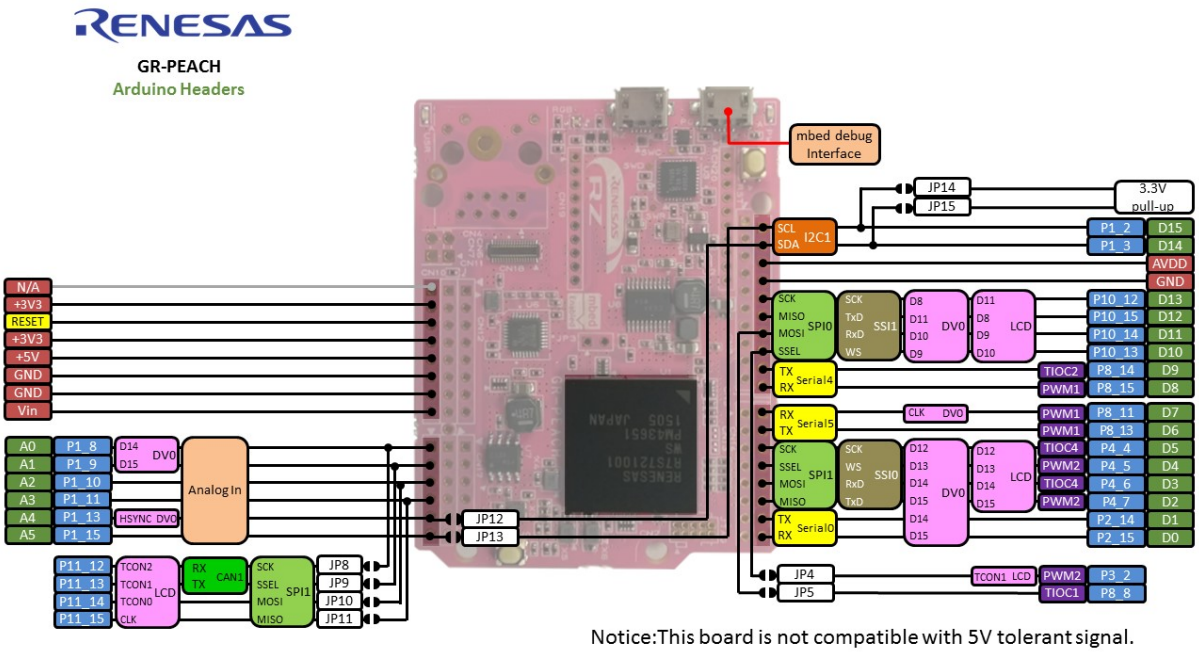
Tabela 17 – Principais Características da GR-PEACH.

Recurso	Características
Microcontrolador	ARM® Cortex®-A9 de 400Mhz
Memória SRAM	10MB *on-chip* expansível até 128MB
Memória Flash	8MB
Memória cache de instruções	32kB
Memória cache de dados	32KB

Fonte: Do autor.

A Figura 94 ilustra a configuração padrão de terminais I/O (*Input/Output*) padrão Arduino UNO (*pin header*), no qual outras plataformas de marcas e modelos semelhantes e compatíveis com o mesmo fator de forma podem ser utilizadas.

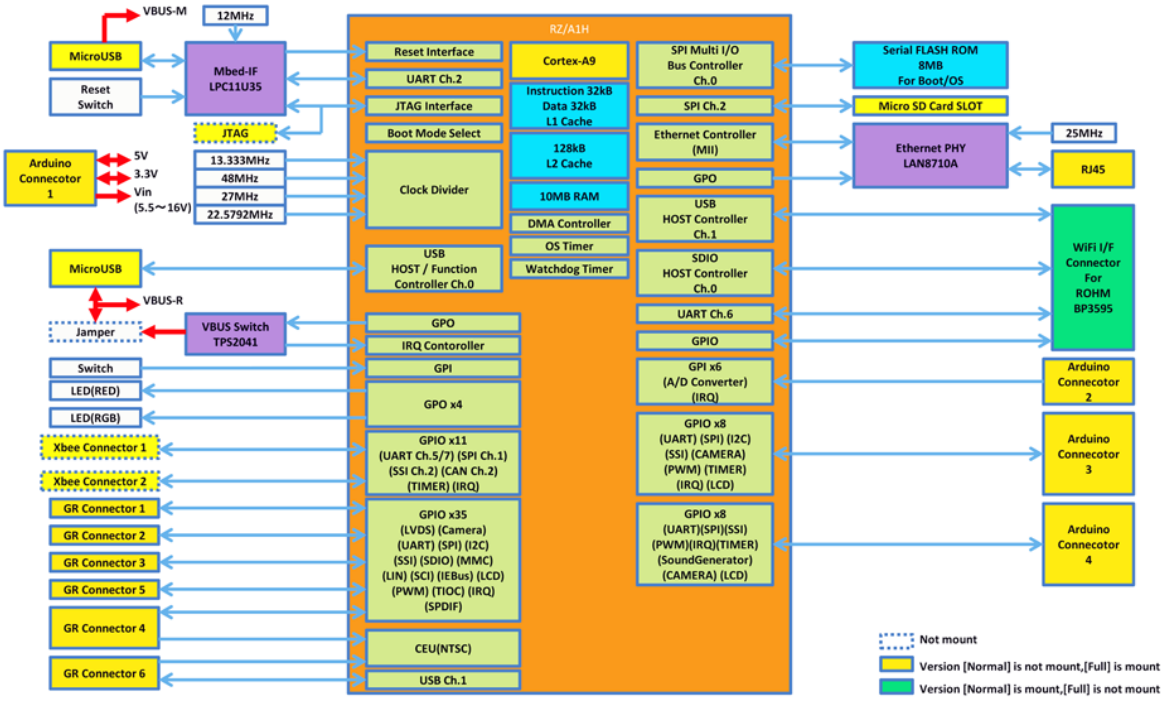
Figura 94 – Diagrama de terminais da GR-PEACH (Arduino *headers*).



Fonte: [153].

A Figura 95 a seguir ilustra o diagrama em blocos da plataforma.

Figura 95 – Diagrama em blocos da GR-PEACH.



Fonte: [154].

Outra importante característica que culminou para a escolha da plataforma GR-PEACH, foi de que possui dimensões físicas idênticas ao Arduino UNO, importante aspecto levado em consideração para utilização neste trabalho, o que permite o uso de outras centenas de plataformas compatíveis com a PCB aqui desenvolvida. Além disso, são vastas as possibilidades para utilização de módulos expansivos ou extensores/escudos (do inglês, *shields*) disponíveis comercialmente, no qual facilita a expansão da funcionalidade da PCB.

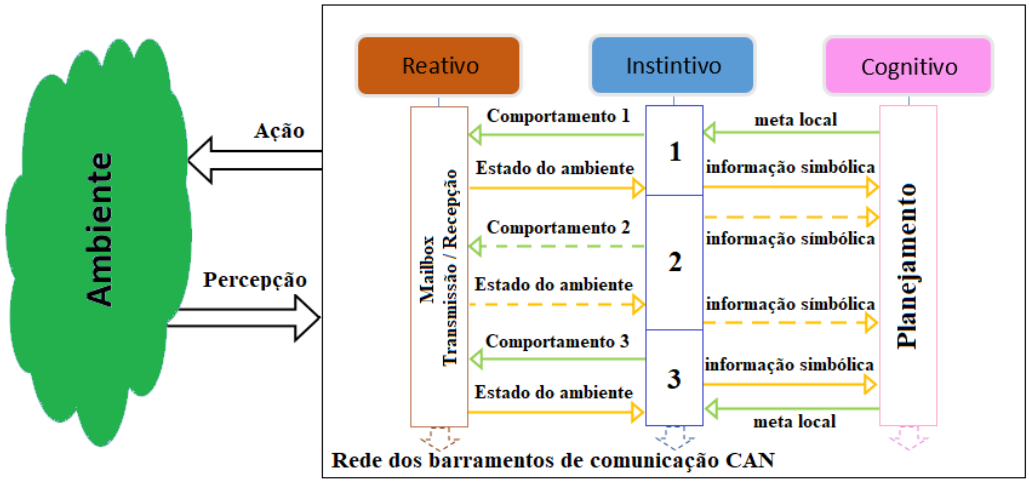
As plataformas Arduino originais utilizam uma série de chips megaAVR, especialmente os ATmega8, ATmega168, ATmega328 e a ATmega1280, porém muitas outras empresas utilizaram outros microcontroladores para evolução da plataforma, mantendo a maneira padrão como os conectores são expostos, permitindo funções mais avançadas como no caso dos microcontroladores ARM® a qual esse trabalho se baseia, configurando-as para embarque de um agente autônomo.

Devido a mais esses motivos, o projeto concebe a PCB projetada a possuir característica modular multiplataformas, objetivo principal desse trabalho. Alguns exemplos de plataformas e suas características principais, são citadas no Apêndice B desta dissertação onde uma vasta gama de possibilidades podem ser utilizadas.

4.12 Barramentos de comunicação CAN

Na Figura 96, o diagrama dos barramentos de comunicação da rede CAN configurada é apresentada, ilustrando como o agente funciona. Esta figura mostra a troca de mensagens entre os três níveis do AAC juntamente com estado do mundo.

Figura 96 – Sequência de informações do barramento.



Fonte: Do autor.

O agente inicia com um comportamento de planejamento com o envio de uma meta local (objetivo final) associado a um plano (trajetória) do nível cognitivo para o nível instintivo. Este, por sua vez escolhe um plano na sua base de conhecimento e envia ao nível reativo, no qual obtém as leituras dos encoder's dos atuadores para realizar os movimentos através do controlador cinemático. Este nível (reativo) converte as percepções/ações recebidas em informação simbólica (palavra/texto) a respeito dos estados do agente e do mundo e as envia de volta ao nível instintivo. A informação simbólica é usada pelo motor de inferência do nível instintivo para decidir se deve mudar o comportamento ativo no reativo, e as envia de volta ao nível reativo. Em seguida o nível instintivo utiliza a informação simbólica no seu motor de inferência para gerar uma nova meta local, caso necessário.

Quando o nível instintivo decide que o comportamento atual não deverá ser mudado, ele simplesmente não envia nenhuma mensagem ao reativo, como se observa no comportamento 2. De forma similar, o nível cognitivo pode decidir não atualizar a meta local especialmente se a informação simbólica recebida do nível instintivo não informa que a meta local atual foi cumprida, como também pode-se observar em 2.

4.13 Conclusão do Capítulo

Neste Capítulo concebeu-se uma estrutura completa de hardware composta por uma PCB de característica modular multi-plataforma. Nesta PCB, configurou-se uma estrutura composta por uma rede de microcontroladores interconectados a barramento CAN tendo como base o trabalho realizado em [29], configurada para embarcar o AAC em um novo robô móvel omnidirecional. Os dois barramentos da rede foram configurados da seguinte forma: o PSoC® 5LP (nível reativo), o *mbed* (nível instintivo) e o GR-PEACH (nível cognitivo), se comunicam através do protocolo CAN entre o PSoC® e o FF-LPC546xx e entre o LTeK® FF-LPC546xx e o GR-PEACH. Isso se faz necessário porque um barramento corresponde a coordenação da execução do planejamento, já o segundo barramento precisa comportar o tráfego de informações do ciclo percepção-ação.

Uma placa de circuito impresso comportará todos os elementos do sistema como será mostrado no próximo Capítulo e, portanto, toda a comunicação será embarcada: a camada física dos barramentos consistirá de transceptores CAN e as linhas de CAN-Tx e CAN-Rx, CAN-HIGH e CAN-LOW, tal como ilustrado na Figura 85, implementadas na PCB.

Conforme discorreu-se no Capítulo 3, a nova rede pode executar tanto sistemas LTP e LPO utilizados em [29], quanto uma representação de conhecimento baseada em lógica difusa, na qual foi utilizada neste trabalho.

5 Resultados Experimentais da Arquitetura de Hardware

Como foi visto no Capítulo 4, na Figura 49, uma das características principais a se destacar no projeto é sua estrutura modular, na qual permite a substituição de plataformas ou configuração para diversos experimentos de acordo com o estudo realizado. A implementação e configuração de um experimento, destinado a execução do AAC, realizados com a nova arquitetura de hardware reconfigurável, serão apresentados a seguir.

Inicialmente é mostrado a implementação do modelo cinemático. Em seguida, foi configurada na PCB uma rede CAN utilizando-se quatro transceptores, formando dois barramentos distintos entre as três principais plataformas ARM® que compõem o projeto. Essa configuração foi realizada no intuito de embarcar o agente autônomo e esse experimento foi executado alterando-se a representação de conhecimento utilizada em [29] para a lógica difusa, na qual será utilizada pela inferência do instintivo.

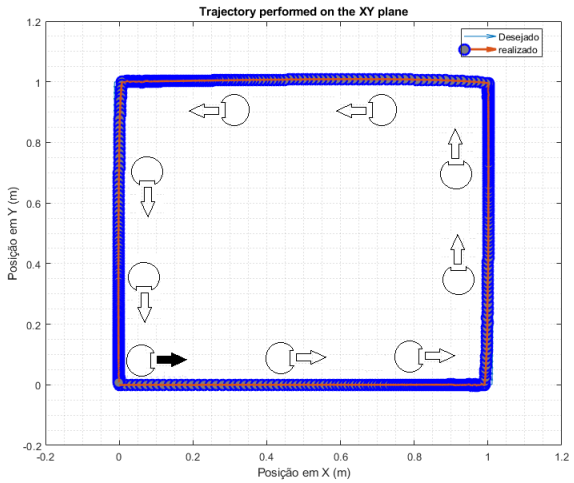
5.1 Implementação do Controle Cinemático

Nesta seção, serão apresentados alguns resultados que serviu de base para embarcar o controlador cinemático. Com isso, foi utilizado o robô com apenas um microcontrolador, tendo como base a arquitetura do diagrama do microcontrolador completo ilustrado na Figura 82 mostrada Capítulo 4. Para testes de controle realizou-se experimentos de estabilização ponto-a-ponto e de simulação do rastreamento de trajetória na qual serão vistos a seguir.

Estabilização Ponto-a-Ponto

A tarefa de controle PI (Proporcional Integral) de velocidade dos motores foi executada em um *loop* do controlador cinemático num período de $T = 20ms$, o que é suficiente para estabilização com erro pequeno. Os valores das constantes proporcional P e integral I utilizadas foram obtidas no *systemIdentification* utilizando o *PIDTuner*, onde foram utilizados os valores médio ta Tabela 13 no Capítulo 4, sendo: Kp = 0.01964 e Ki = 0.19565 respectivamente. A referência utilizada para a estabilização ponto-a-ponto foi um quadrado de $1m$ de comprimento de cada um dos lados e a matriz onde caracteriza-se o *setpoint* é dada por: $[1, 1, 0, 0; 0, 1, 1, 0; 0, 0, 0, 0]$. Os gráficos mostram o percurso realizado pelo robô e as velocidades das rodas e pôde-se perceber que este foi capaz de realizar corretamente a tarefa determinada sobrepondo o *setpoint* de maneira satisfatória. A Figura 97 ilustra a trajetória do robô.

Figura 97 – Resultado real para estabilização ponto-a-ponto.



As Figuras 98, 99 e 100 ilustram graficamente os resultados obtidos das velocidades das rodas e o erro do controlador para os três motores, respectivamente.

Figura 98 – Velocidade e erro do controlador - Motor 1.

(a) Velocidade. **(b)** Erro do controlador.

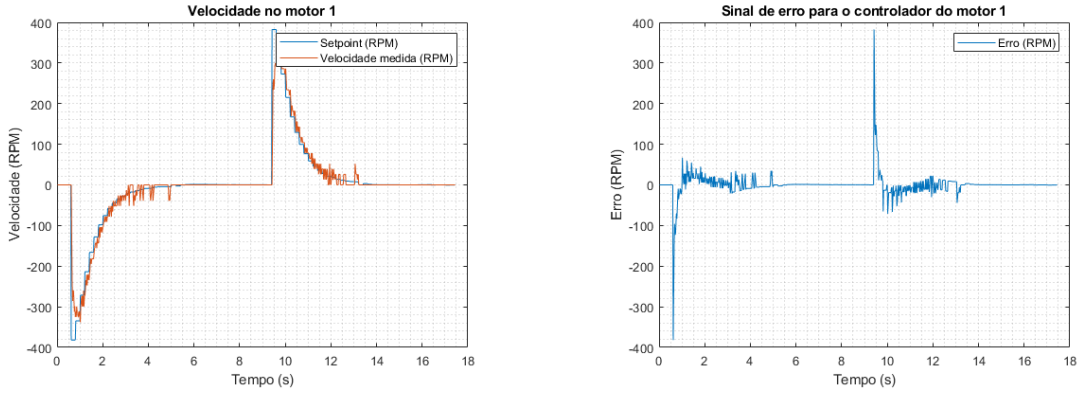


Figura 99 – Velocidade e erro do controlador - Motor 2.

(a) Velocidade. **(b)** Erro do controlador.

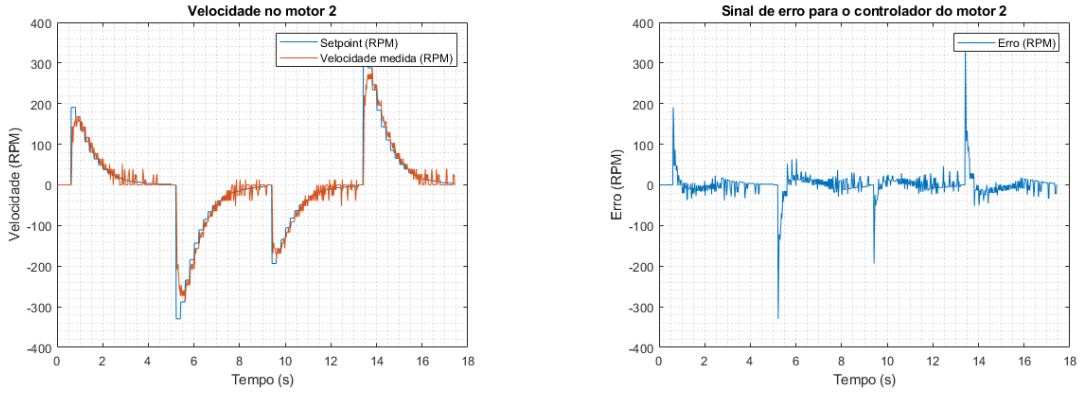
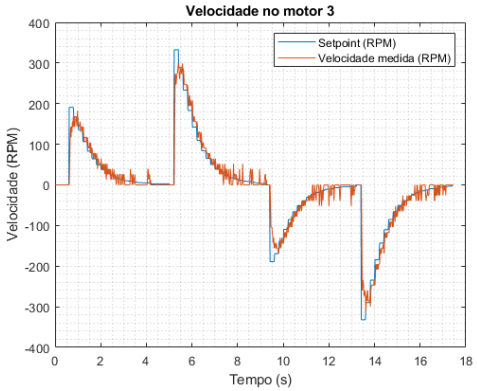
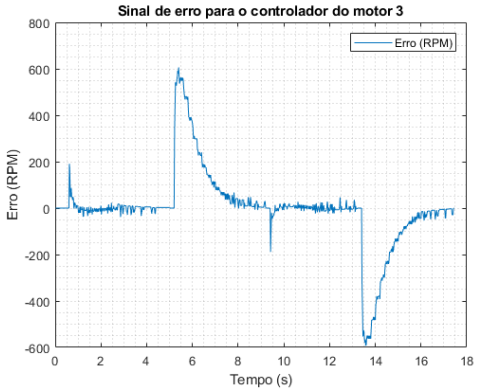


Figura 100 – Velocidade e erro do controlador - Motor 3.

(a) Velocidade.



(b) Erro do controlador.



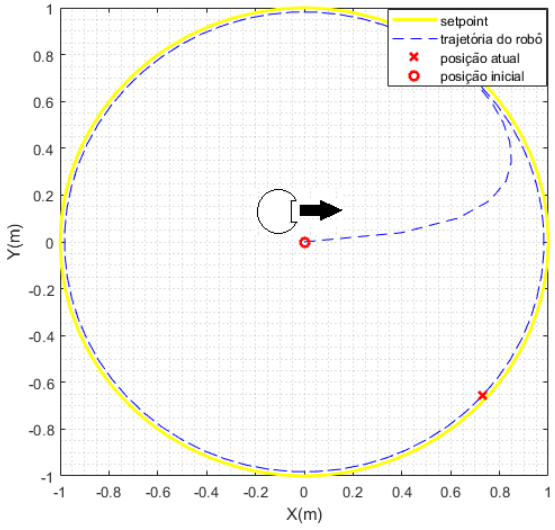
Rastreamento de Trajetória

A referência utilizada para o rastreamento de trajetória foi uma circunferência com raio de $1m$, onde os pontos foram gerados pelo controlador cinemático através da equação paramétrica da circunferência, enquanto a orientação foi definida em $2\pi rad$ e a velocidade angular da trajetória de referência foi de $0.1\frac{rad}{s}$. A Equação 5.1 mostra a expressão paramétrica da trajetória realizada.

$$\begin{bmatrix} X(t) \\ Y(t) \\ \omega(t) \end{bmatrix} = \begin{bmatrix} cos(0.1t) \\ sen(0.1t) \\ 2\pi \end{bmatrix} \tag{5.1}$$

A Figura 101 a seguir ilustra o gráfico da trajetória circular realizada pelo robô.

Figura 101 – Resultado simulado para o rastreamento de trajetória.



Fonte: Do autor.

Os resultados mostraram que no teste prático (rastreamento ponto-a-ponto) com o robô ao solo, a semelhança entre as trajetórias determinada (*setpoint*) e a trajetória realizada pelo robô, verificou-se a estabilização bem próxima entre ambas. Também no teste simulado (rastreamento de trajetória), percebeu-se semelhança entre as trajetórias determinada (*setpoint*) e a real. Com isso, confirma-se com os resultados obtidos que houve eficiência de controle.

5.2 Implementação da Arquitetura do AAC Embarcado

Para a implementação da Arquitetura do Agente Cognitivo o hardware descrito neste trabalho, foi suficiente executar uma tarefa simples de planejamento, porque permitirá avaliar o desempenho de todos os níveis, bem como a comunicação entre eles. Assim, a seguinte base de conhecimento foi usada para o planejamento.

$$
\begin{aligned}
&(go\_to\_p2\\
&\qquad (if\ (logic(robot\ position\ p1)))\\
&\qquad (then\ add((logic(robot\ position\ p2)))\\
&\qquad\qquad rem((logic(robot\ position\ p1)))))\\
&(go\_to\_p3\\
&\qquad (if\ (logic(robot\ position\ p2)))\\
&\qquad (then\ add((logic(robot\ position\ p3)))\\
&\qquad\qquad rem((logic(robot\ position\ p2)))))
\end{aligned}
$$

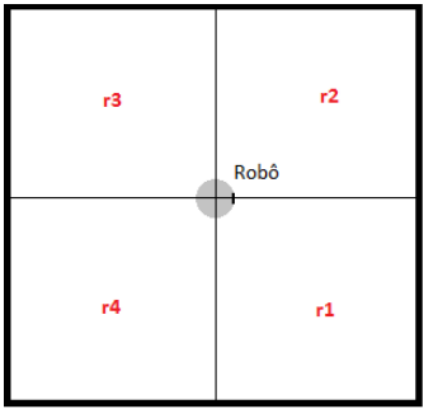
Essa base de conhecimento, escrita no idioma específico do AAC, gera um plano contendo apenas duas ações: vá do ponto $p1$ ao ponto $p2$ e, em seguida, de $p2$ a $p3$. A implementação no nível cognitivo não mudou em relação aos trabalhos anteriores, portanto não havia necessidade de um domínio de planejamento complexo.

O mundo está dividido em quadrantes ao redor do robô, como ilustra a Figura 102. Cada um dos quadrantes $q1$ a $q4$ pode conter um obstáculo ou uma meta assim como em [29]. Essa representação também será adotada no presente trabalho, mas dessa vez, compondo a base de conhecimento nebuloso do nível instintivo com regras que testam em que quadrante é o obstáculo mais próximo e o objetivo, produzindo um comportamento reativo como saída que, por sua vez, é enviada ao nível reativo. A seguir, é apresentado um exemplo de regra do nível instintivo.

$$
\begin{aligned}
IF\ ((obst\_distance_x\ IS\ Negative)\ AND \\
(obst\_distance_y\ IS\ Positive)\ AND \\
(goal\_distance_x\ IS\ Negative)\ AND \\
(goal\_distance_y\ IS\ Positive)) \\
THEN((velocity_x\ IS\ Positive), \\
((velocity_y\ IS\ Zero)))
\end{aligned}
$$

Nesta regra, se a distância do obstáculo na dimensão x for negativo e positivo na dimensão y (obstáculo no quadrante q4), e a distância para a meta segue o mesmo padrão (meta também na q4), a saída seria mover-se na direção x positiva. Os conjuntos nebulosos Negativo e Positivo nos antecedentes são triangulares, variando de -10 a 10. No consequente, Negativo, Zero e Positivo são *singleton*, localizados em -1, 0 e 1, respectivamente. Na Figura 104 é ilustrado o deslocamento do robô.

Figura 102 – Representação do mundo.

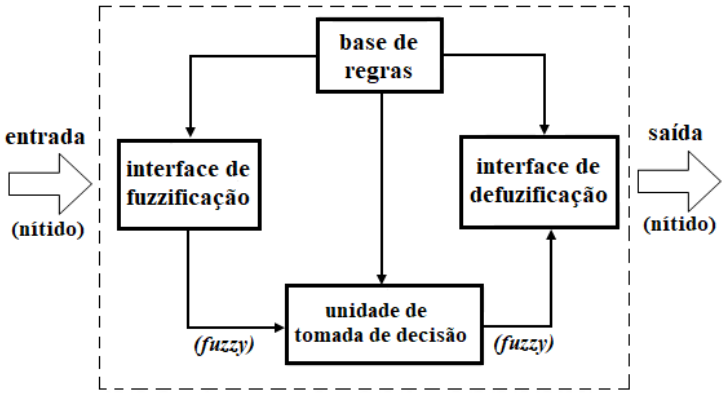


Fonte: Do autor.

Nesse trabalho foi utilizada a biblioteca *eFLL* (*Embedded Fuzzy Logic Library*) que consiste numa biblioteca em configuração básica de um sistema lógico difuso para utilização em sistemas embarcados. Essa biblioteca foi desenvolvida pelo RRG *Robotic Research Group* da Universidade Estadual do Piauí (UESPI-Teresina).

A biblioteca *eFLL* utilizada para implementar a Lógica *Fuzzy* foi escrita em $C++/C$ usa apenas a biblioteca de linguagem C padrão "*stdlib.h*". Portanto o *eFLL* é uma biblioteca projetada não apenas para a plataforma Arduino, mas para qualquer sistema embarcado ou não que tenham seus comandos escritos na linguagem C. A Figura 103 ilustra o diagrama em blocos de um sistema *fuzzy* básico.

Figura 103 – Configuração básica de um sistema lógico difuso.

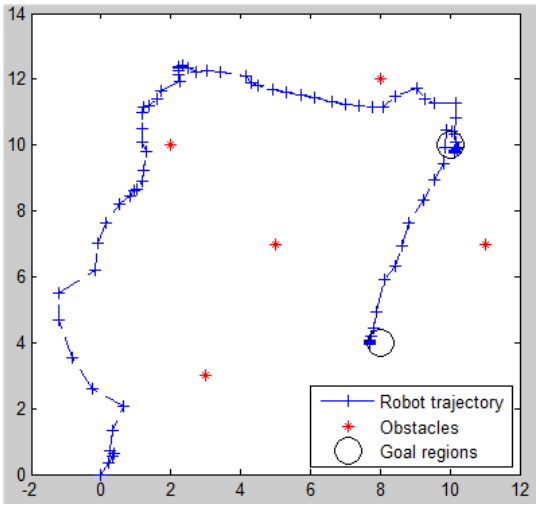


Fonte: [155].

Uma das vantagens da utilização dessa biblioteca é que não possui limitações explícitas quanto à quantidade de regras, entradas ou saídas nebulosas, capacidade de processamento e armazenamento limitados de cada microcontrolador [155].

O experimento realizado consistiu em uma navegação sem colisão em um ambiente simulado com obstáculos estáticos. Embora o ambiente tenha sido simulado, o nível reativo estava realmente controlando as rodas. O barramento CAN foi configurado para a velocidade de $125Kbps$. Os pontos $p1$, $p2$ e $p3$ são dados por (0m; 0m), (10m; 10m) e (8m; 4m). A Figura 104 a seguir ilustra a trajetória do robô.

Figura 104 – Resultados de navegação sem colisão.



Fonte: Do autor.

Durante a trajetória do robô foi possível visualizar as amostras da comunicação, visualizadas pelo "trem de pulsos"entre os dois barramentos, executado em um experimento prático, como o mostrado na Figura 104. O Analisador lógico e de protocolos USB de 8

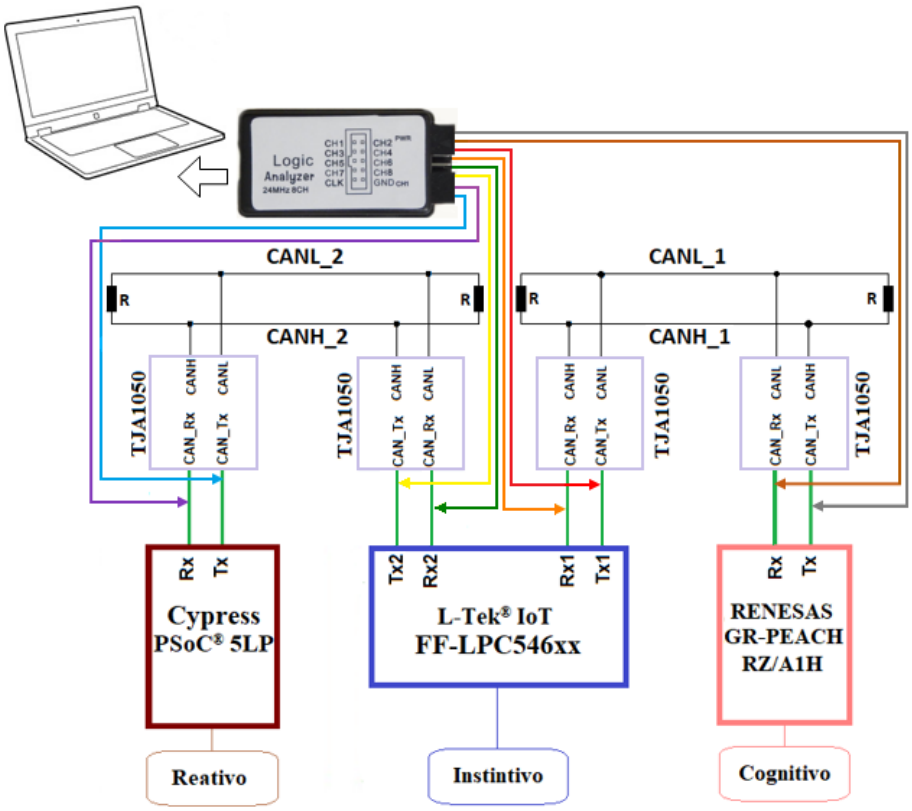
canais mostrado na Figura 105 utilizado é genérico, porém compatível com o da empresa *SALEAE*. O software *Logic* que funciona com o analisador foi capaz de obter as informações presentes nos dois barramentos CAN.

Figura 105 – Analisador lógico.

Fonte: Do autor.

Esse experimento foi suficiente para a visualização dos estados lógicos, caracterizados pelas mensagens presentes nos barramentos a saber: PSoC: Nível reativo (CAN-Tx e CAN-Rx), LTeK: Nível Instintivo (CAN-Tx1 e CAN-Rx1, CAN-Tx2 e CAN-Tx2) e GR-PEACH: Nível Cognitivo (CAN-Tx e CAN-Rx), como ilustra as ligações na Figura 106.

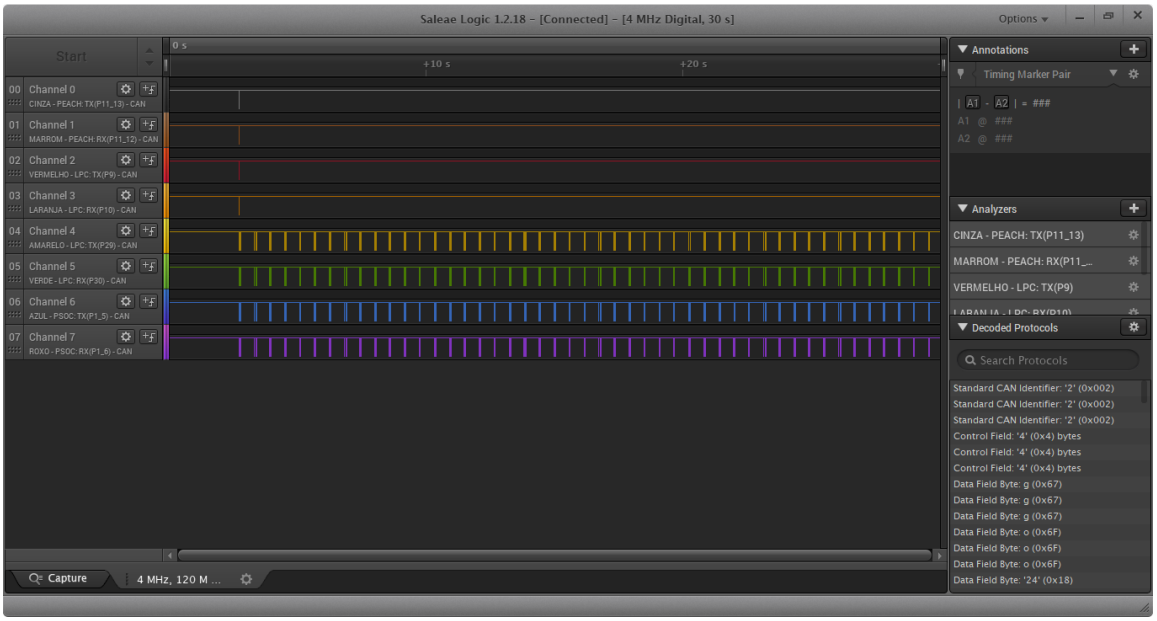
Figura 106 – Circuito para obtenção dos estados lógicos.



Fonte: Do autor.

No experimento, o analisador lógico foi configurado para protocolo CAN a uma velocidade de conexão de $125Kbps$, onde foi possível visualizar os estados lógicos *bit* a *bit* de forma simples e sem erros. A Figura 107 mostra o estado geral dos sinais presente nos dois barramentos, os quais serão detalhados a seguir.

Figura 107 – Estados dos barramentos 1 e 2 durante trajetória.



Fonte: Do autor.

Inicialmente um planejamento é enviado do nível cognitivo ao nível instintivo contendo uma meta, no qual consiste em o robô percorrer uma determinada distância contendo obstáculos de um ponto a outro. A configuração das ligações dos pontos de conexão do barramento 1 são listadas na Tabela 18.

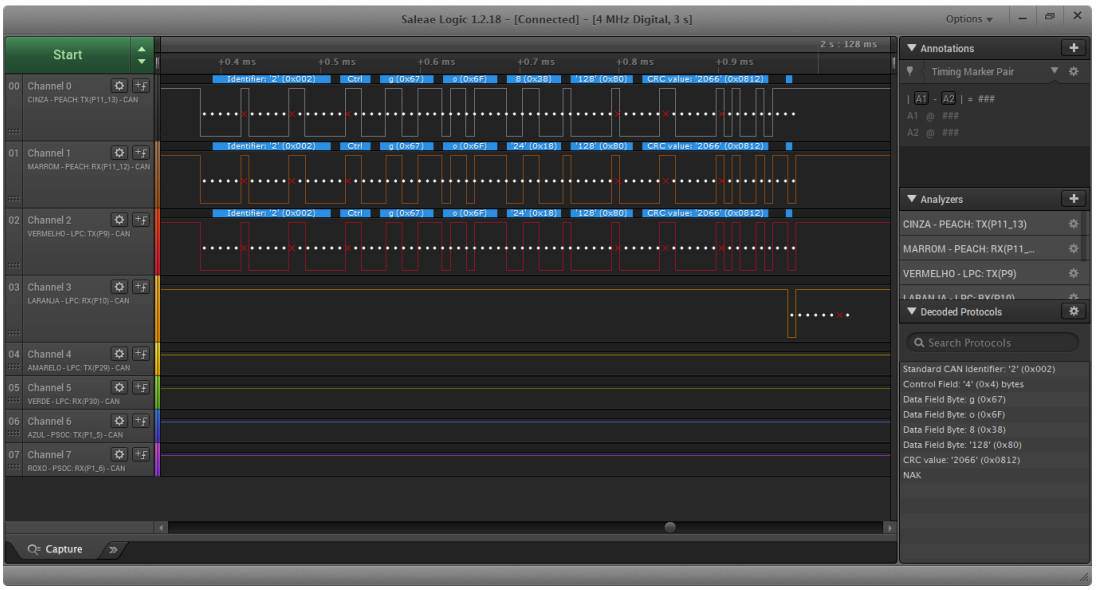
Tabela 18 – Terminais associados a cada canal do barramento 1.

Barramento 1			
GR_PEACH (cognitivo)		LPC546xx (instintivo)	
canal 0	**canal 1**	**canal 2**	**canal 3**
CAN_Tx	CAN_Rx	CAN_Tx1	CAN_Rx1

Fonte: Do autor.

Estes pontos $go1$ e $go2$ caracterizados por *strings* são compostos por um ID. O quadro ilustrado na Figura 108 ilustra o conteúdo da mensagem, no qual é visualizada a informação da meta inicial enviada ao nível instintivo.

Figura 108 – Mensagem contendo a meta enviada.



Fonte: Do autor.

A identificação do padrão utilizada é mostrado na janela de decodificação do protocolo, no caso, foi utilizado o padrão CAN *standard* identificado pelo ID:2(0x002). Esse identificador único da mensagem é utilizado para identificar o conteúdo da mensagem da posição inicial enviada pelo nível cognitivo como meta para o nível instintivo, que, como pode ser observado é composto por 4 *bytes* no campo de controle. Outra característica desse identificador, além de definir o conteúdo da mensagem é também a de estabelecer a prioridade da mesma. Ao receber essa mensagem, o nível instintivo envia uma nova mensagem ao nível reativo composto por um novo ID. A configuração das ligações dos pontos de conexão do barramento 2 são listadas na Tabela 19.

Tabela 19 – Terminais associados a cada canal do barramento 2

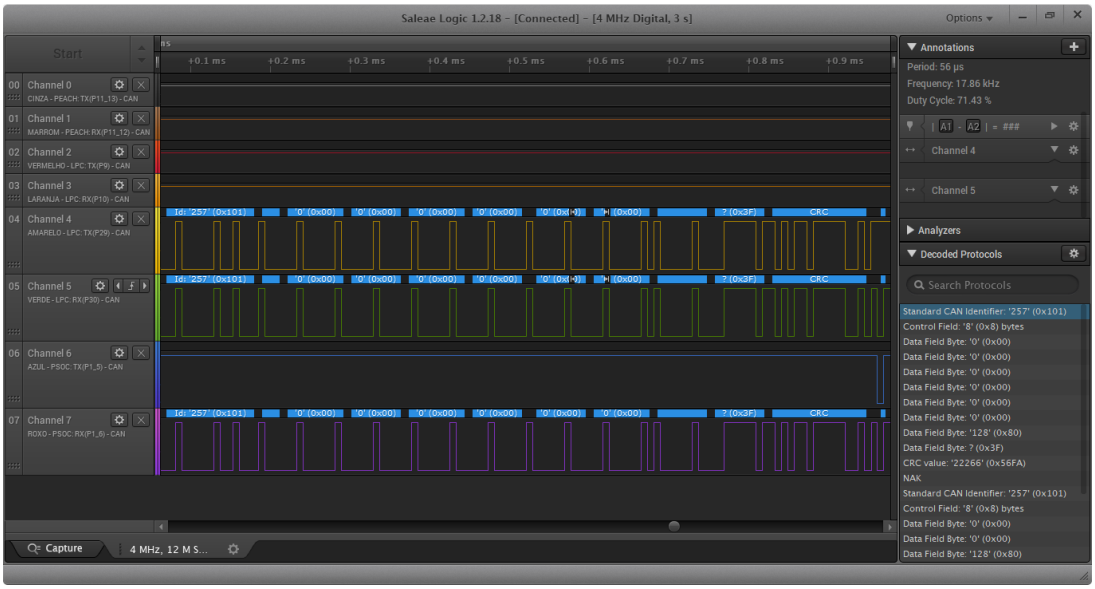
Barramento 2			
LPC546xx (instintivo)		PSoC 5LP(reativo)	
canal 4	**canal 5**	**canal 6**	**canal 7**
CAN_Tx2	CAN_Rx2	CAN_Tx	CAN_Rx

Fonte: Do autor.

Outro campo que contém no *frame* de dados da mensagem CAN e que pôde ser obtida com o analisador lógico é o *CRC* (do inglês, *Cyclic Redundancy Code*). O CAN implementa esse mecanismo de *CRC* no qual permite a um dos *nós* receptor verificar a integridade do *frame* recebido. Se essa sequência de *bits* recebida por um nó transmissor não for idêntica à sequência de *CRC* calculada pelo receptor, então ocorrerá um erro de

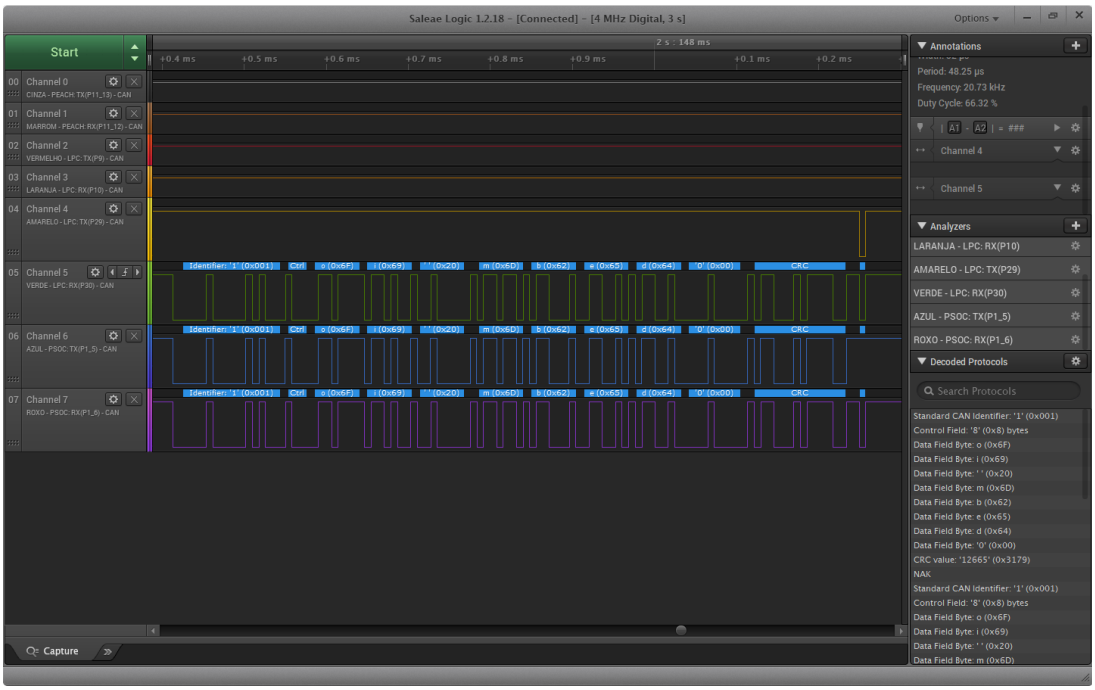
CRC. A Figura 109 a seguir ilustra o quadro composto pela mensagem enviada ao reativo composta pelo ID:1(0x101).

Figura 109 – Mensagem transmitida pelo instintivo.



Fonte: Do autor.

Figura 110 – Mensagem enviada pelo reativo.

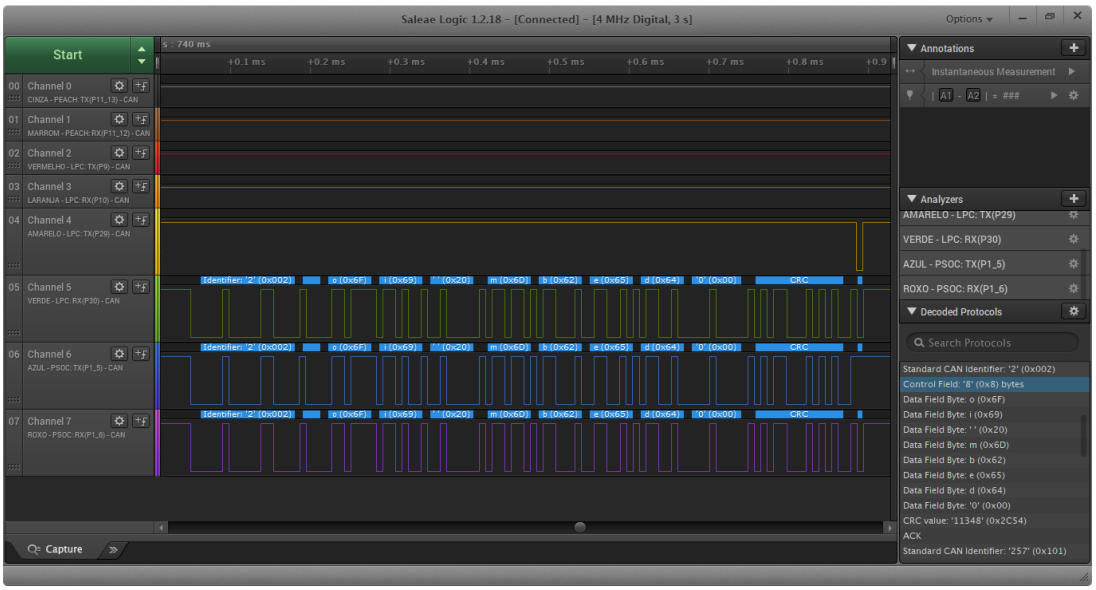


Fonte: Do autor.

Em seguida o quadro das mensagem, no qual refere-se a mensagem com os ID:1(0x001) e ID:2(0x002) composto pela mensagem *"oi mbed"* enviada pelo nível re-

ativo e recebida pelo nível instintivo informando o estado do mundo. As Figuras 110 e 111 ilustram os quadros recebidos.

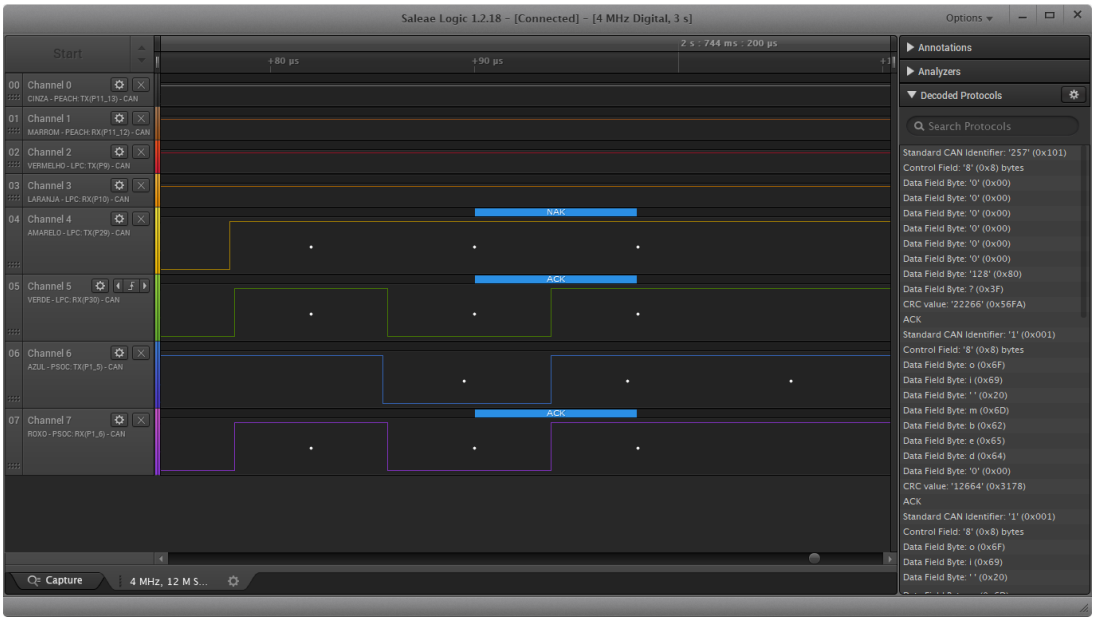
Figura 111 – Mensagem do estado do mundo.



Fonte: Do autor.

A Figura 112 ilustra o campo de reconhecimento de recebimento da mensagem.

Figura 112 – Campo de reconhecimento *ACK*.



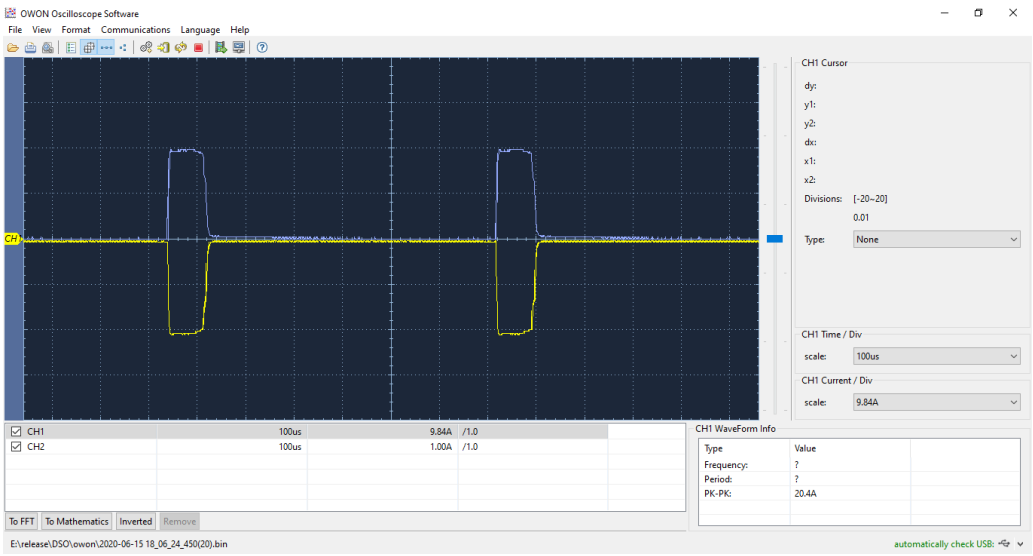
Fonte: Do autor.

Nessa mensagem é possível observar o campo *ACK* (*do inglês, Acknowledge*) o qual identifica o reconhecimento de recebimento da mensagem. Este campo constitui-se de dois

bits, o *slot* ACK e o delimitador ACK, sendo este ultimo "recessivo". Os nós receptores ao receberem corretamente a sequência CRC, enviam o reconhecimento substituindo o *bit* "recessivo"por um *bit* "dominante"no ACK.

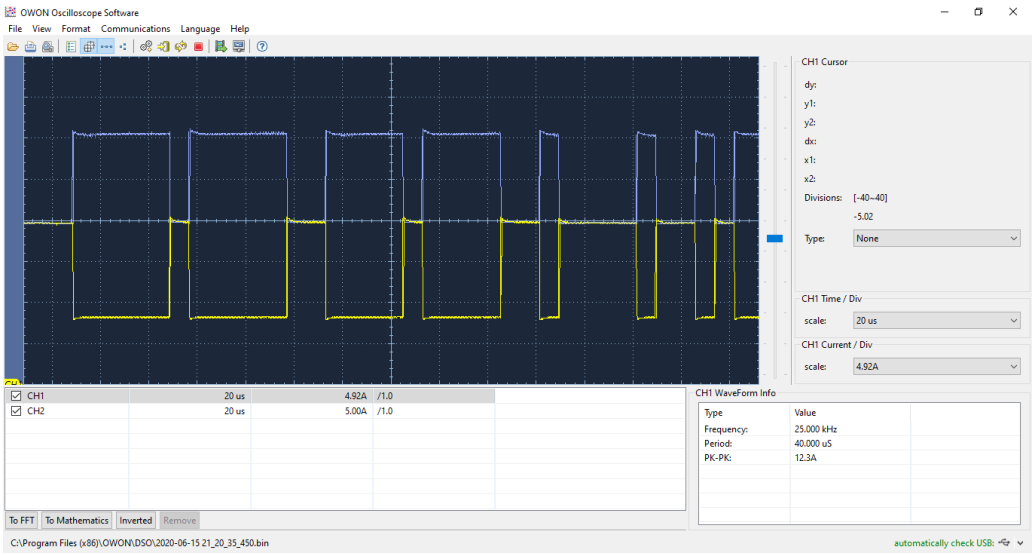
Nos terminais HIGH e LOW dos dois barramentos também foi possível obter as diferenças de tensões entre dois estados dominantes utilizando o osciloscópio, conectado via porta USB do computador utilizando o software próprio, no qual durante a trajetória obteve-se um valor médio de $2.26 Volts$ entre eles. As Figuras 113 e 114 a seguir ilustra esses quadros.

Figura 113 – CAN-HIGH e CAN-LOW: Barramento 1 (cognitivo e instintivo).



Fonte: Do autor.

Figura 114 – CAN-HIGH e CAN-LOW: Barramento 2 (instintivo e reativo).



Fonte: Do autor.

5.3 Conclusão do Capítulo

Neste capítulo, foram descritos alguns resultados com o controlador cinemático, no qual foram obtidos a validação do modelo através do MATLAB® e o robô através da comunicação serial/USB. Os demais experimentos utilizou-se a arquitetura de hardware descrita, configurada para execução do modelo de agente cognitivo mesclando comportamentos reativos no nível instintivo, utilizando a lógica difusa.

Um experimento mostrou que o robô era capaz de navegar sem falhas durante a execução do AAC, com o agente completamente embarcado a uma configuração composta pelas três principais plataformas ARM® da PCB, consistindo em uma rede CAN contendo dois barramentos, utilizando-se quatro transceptores.

Essa proposta, possibilitou-se que o agente se torne mais confiável, robusto e com maior tolerância a falhas, ou seja, a falha de um subsistema, não acarreta a falha em toda arquitetura da rede CAN. Além disso, essa configuração ofereceu boas condições para que se avaliasse o seu desempenho, e com isso, foi possível observar as vantagens que se obtêm utilizando uma arquitetura distribuída embarcada, tais como: velocidade de comunicação distintas entre os níveis, confiabilidade e possibilidades de expansão para outros pontos de controle.

A arquitetura desenvolvida também é capaz de lidar com um grande número de sensores e atuadores permitindo que estes sejam controlados independentemente de qual microcontrolador será utilizado para comunicação através de uma rede CAN, para o propósito desenvolvido nesse projeto ou para outras finalidades de pesquisas. Por fim, a implementação de um barramento CAN envolve conceitos de hardware que depende da estratégia de utilização do microcontrolador e de software.

6 Considerações Finais e Trabalhos Futuros

O desenvolvimento de sistemas robóticos físicos podem apresentar algumas imperfeições e limitações que muitas vezes não são levadas em consideração ou modeladas pelas abordagens convencionais. Dentre as imperfeições em sistemas como em robótica móvel, pode ser destacada com relação a sua estrutura física e até mesmo a arquitetura ao qual o projeto foi embarcado, podendo surgir restrições significativas no sistema, deixando-o limitado a determinada tarefa. Isto possibilita uma visão geral do desempenho do sistema operando em condições próximas às especificadas, mas não a ideal. Com isso, é possível a substituição de módulos, impondo condições especificas a determinada pesquisa.

Sendo assim, uma arquitetura completa de hardware reconfigurável para sistemas embarcados foi desenvolvido neste trabalho, na qual é permitida tanto a substituição das plataformas existentes, quanto é possível a *upgrade* de cada uma, dedicadas às aplicações que visam. Devido ao novo hardware possuir essas características, foi configurado mimetizando a arquitetura funcional do AAC, para execução de um agente autônomo em um robô móvel omnidirecional utilizando o protocolo CAN.

Esse projeto vai além das tentativas anteriores como em [29], quando uma rede de microcontroladores foi projetada para embarque do AAC no robô omnidirecional AxeBot. No presente projeto, amplia-se e moderniza as estruturas mecânicas e eletrônica do robô AxeBot, desenvolvendo uma PCB com característica multiplataforma e com as dimensões do robô. Adicionalmente, outros elementos são possíveis, como possibilidades de utilização de plataformas com fator de forma Arduino em uma de suas opções, na qual permite a expansão de suas funcionalidades através de sua vasta gama de extensores, os chamados *shields*).

A configuração do hardware desenvolvida neste trabalho utilizando-se barramento CAN, no entanto consiste apenas de um subconjunto de seus recursos: àqueles necessários para executar o agente cognitivo em questão. Com isso, este trabalho mostrou que a aplicação prática realizada no novo hardware atendeu satisfatoriamente às demandas de concorrência e comunicação do AAC. Finalmente, a arquitetura mostrou-se flexível e bastante modular, pois permitiu que um novo método de inferência fosse utilizado em um dos níveis do AAC, fato importante para pesquisas posteriores. Além disso, a nova arquitetura de hardware desenvolvida mostra que é mais avançada e pode ser aplicada como base para estudos de tópicos relacionados à inteligência artificial e sistemas multi-agentes, por dotar-se de diversas opções através da fácil e documentada configuração física provendo uma solução reconfigurável e prática referente ao quesito de hardware.

6.1 Perspectiva de Trabalhos Futuros

Utilização da PCB Desenvolvida

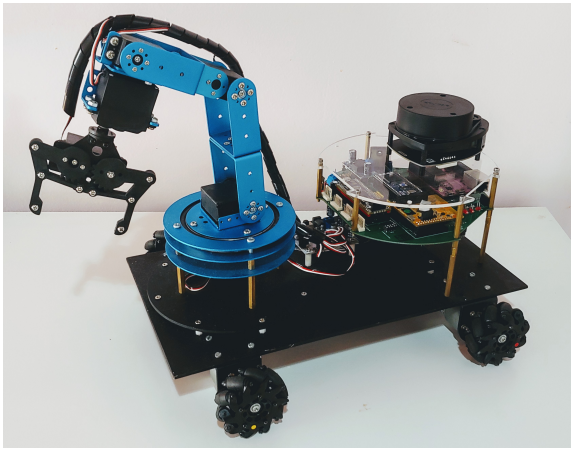
Como trabalhos futuros, pretende-se utilizar a PCB desenvolvida nesse trabalho para embarcar o AAC utilizando um robô móvel com quatro rodas *mecanum* composto por manipulador articulado de 6DOF. Estendendo aplicações específicas para essas duas estruturas robóticas, podemos citar:

- Remodelar os controladores cinemático e dinâmico, essa última não abordada neste projeto;
- Desenvolver práticas e simulações com métodos de controle de maior nível, de modo a garantir um seguimento eficiente por parte dos robôs;
- Utilizar sensores infravermelho e ultrassônico para serem utilizados no nível reativo;
- Inserção de visão computacional e sensor a laser (*Lidar sensor*) utilizando o sistema de código aberto com o *framework* ROS (*Robot Operating System*).

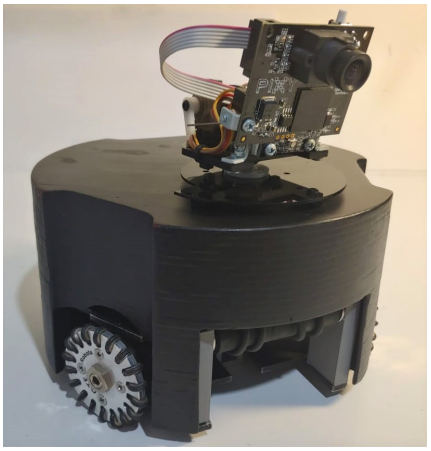
A Figura 115 (a) e (b) mostram os robôs já em fase de desenvolvimento.

Figura 115 – Opções para projetos futuros.

(a) Robô omnidirecional composto por manipulador articulado.



(b) Robô composto por câmera



Fonte: Do Autor.

Utilização de $FPGA$

A grande maioria dos dispositivos eletrônicos presentes no mercado são sistemas embarcados que utilizam hardwares e softwares com lógica digital. Como o crescimento da microeletrônica os principais fabricantes de semicondutores desenvolvem componentes

com dimensões cada vez mais reduzidas e com baixo consumo de energia, como é o caso da família de portas lógicas baseadas na tecnologia $CMOS$ (do inglês, *Complementary Metal-Oxide-Semiconductor*), amplamente utilizada em circuitos digitais. Um hardware digital, do mais simples ao mais complexo, é composto de circuitos lógicos e a maneira como eles são organizados é o que define sua complexidade e funcionalidade.

Devido a isso, alternativamente, pretende-se também o desenvolvimento da arquitetura geral de hardware do presente trabalho com a plataforma $FPGA$ (do inglês, *Field Programmable Gate Arrays*) para desenvolver toda PCB concebida nesse trabalho a nível de circuito integrado. Com o sistema desenvolvido completamente encapsulado em Circuitos Integrados ($CI's$) em FPGA propõe-se realizar experimentos com estruturas mais configuráveis, mais leves e bastante reduzidas, sendo possível ampliar as funcionalidades do hardware, como:

- Desenvolver uma plataforma a nível de circuito integrado, mantendo as funcionalidades existentes e ampliando-as;
- Projetar funcionalidade dessa plataforma a uma drone com 4 motores *brushless* e câmera.

À proporção que mais sistemas embarcados são desenvolvidos usando plataformas $FPGA$, há uma necessidade crescente de oferecer suporte a processadores com essas plataformas. Uma opção é o *softprocessor*, um processador de instruções programável implementado na lógica reconfigurável do $FPGA$. Além disso, atualmente existem plataformas com esse fator de forma para desenvolvimento com FPGA e que podem ser programadas utilizando um *softprocessor* a exemplo da plataforma $XLR8$ utilizando o kit $Intel$ MAX 10 $FPGA$ (50K LE) da *Alorium Technology* e o *DE0-Nano-SoC* um kit da *Atlas-SoC*.

Esses kits apresentam uma plataforma de design de hardware criada em torno do $FPGA$ Altera *System-on-Chip* (SoC) ao qual integra um sistema de processador rígido HPS (do inglês, *Hard Processor System*) baseado em ARM®, que consiste em interfaces, periféricos e de memória ligadas a plataforma $FPGA$. Finalmente, desenvolvimentos e testes futuros com a plataforma FPGA, devem fortalecer ainda mais o uso desta nova arquitetura de hardware, seja com o AAC ou para outras finalidades de pesquisas.

Referências

1 MUSEUM, C. C. H. *SHAKEY*. 1969. <https://www.computerhistory.org/revolution/artificial-intelligence-robotics/13/289/1229?position=1>. Acesso em: 24 jul. 2020. 27

2 C.LAUGIER, F. G. An adaptative collision-free trajectory planner. *85' International Conference on Advanced Robotics.*, 1985. 28

3 KHATIB, O. Real-time obstacle avoidance for manipulators and mobile robots. *The international Journal of Robotics Research. v.5, n.1, p. 90-98.*, 1986. 28

4 ARKIN, R. C. Motor schema based navigation for a mobile robot: An approach to programming by behavior. *Proc. IEEE Int. Conference Robotics and Automation [S.l.: s.n], v.4, p. 264-271.*, 1987. 28

5 BROOKS, R. A. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation. [S.l.: s.n], v.2, n.1, p. 14-23.*, 1986. 28

6 GRASSI, J. V. *Arquitetura hibrida para robôs móveis baseada em funções de navegação com interação humana*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2006. 28, 32, 45

7 AIR, S. N.; MUSEUM, S. *Mars Pathfinder Rover: Sojourner*. 2020. <https://airandspace.si.edu/multimedia-gallery/web12070-2011640jpg>. Acesso em: 02 abr. 2020. 29

8 SAMSUNG ELETRÔNICA DA AMAZÔNIA. *SAMSUNG*. Disponível em: <https://www.samsung.com/br/vacuum-cleaners/>. Acesso em: 10 mar. 2020. 29

9 BUSINESS INSIDER. *AMAZON*. Disponível em: <https://www.businessinsider.com/amazon-doubled-the-number-of-kiva-robots-2015-10>. Acesso em: 10 mar. 2020. 29

10 EXERCITO Brasileiro - Governo Federal. 2020. <http://www.eb.mil.br/web/midia-impressa/noticiario-do-exercito/-/journal_content/56/16541/7571300?refererPlid=16560>. Acesso em: 10 mar. 2020. 29

11 MARS, N. S. *2020 Mission Perseverance Rover*. 2020. <https://mars.nasa.gov/mars2020/>. Acesso em: 02 ago. 2020. 29

12 CHOSET, H. e. a. *Principles of Robot Motion. Theory, Algorithms, and Implementations. 1. ed.* Massachusetts, EUA: The MIT Press, 2005. 30

13 ROBOTICS, A. *MEET DIGIT: THE NEWEST ROBOT FROM AGILITY ROBOTICS*. 2020. <https://www.agilityrobotics.com/meet-digit>. Acesso em: 02 set. 2020. 30

14 WIRED. *New NASA Tech Kills Trespassing Drones Without Touching Them*. 2020. <https://www.wired.com/story/nasa-tech-kills-trespassing-drones/>. Acesso em: 02 set. 2020. 30

15 CIMATEC, S. *FlatFish Project*. 2020. <http://www.senaicimatec.com.br/flatfish/>. Acesso em: 27 jun. 2020. 30

16 ROBOTICS AND AUTOMATION MAGAZINE. VOL. 14, NO. 1. *The Evolution of Robotics Research.* 90-103 p. 30

17 DUDEK G.; JENKIN, M. *Computational Principles of Mobile Robotics*. London: Cambridge University Press, 2010. 30, 33, 45

18 R., K. T. *Robotics and Automation Handbook*. New York: CRC Press, 2010. 30, 34

19 REVIEW, R. R. B. *World Robotics Report: Global Sales of Robots*. 2020. <https://www.roboticsbusinessreview.com/research/world-robotics-report-global-sales-of-robots-hit-16-5b-in-2018/>. Acesso em: 12 mar. 2020. 31

20 MARKETS AND MARKETS. *Mercado de Robótica Industrial*. Disponível em: <https://www.marketsandmarkets.com/Market-Reports/Industrial-Robotics-Market-643.html$>$.Acessoem:10demarçode2020>. Acesso em: 10 mar. 2020. 31

21 BAHIA, U. F. da. *Projeto AxeBot*. 2005. <Http://www.axebot.ufba.br.> Acesso em: 27 mar. 2020. 32

22 SIEGWART R.; NOURBAKHSH, I. R. *Introduction to Autonomous Mobile Robots*. London: Editora Edgard Blucher Ltda, 2004. 33, 43, 44, 46, 47, 49, 51

23 RIBEIRO, T. T. R. F. J. S. e. A. C. A nonlinear mobile robot modelling applied to a model predictive controller. *Symposium on Applied Computing - SAC. Hawaii- Honolulu.*, p. 1186–1187, 2009a. 34

24 CONCEIçãO A. G. S.; CORREIA, M. D. Modeling of a three wheeled omnidirectional robot including friction models. p. 7–12, 2012. 34

25 RODRIGUES, J. A. C. *Plataforma Omnidirecional para Robô de Serviço em Casa*. Dissertação (Mestrado) — Universidade do Minho, 2010. 34

26 NASCIMENTO, T. P. *Controle De Trajetória De Robôs Móveis Omni-Direcionais: Uma Abordagem Multiváriavel*. Dissertação (Mestrado) — Escola Politécnica - Universidade Federal da Bahia., 2009. 34, 37, 55

27 *Desenvolvimento de um robô omnidirecional para fins didáticos usando o kit LEGO MINDSTORMS*. 34

28 SANTOS, J. T. *Projeto e Desenvolvimento de um Sistema Microprocessado Aplicado à Robótica Móvel*. Dissertação (Mestrado) — Universidade Federal da Bahia - PPGEE, 2009. 34

29 FERREIRA, D. S. F. *Embarcando o Agente Autônomo Concorrente em uma Rede de Microcontroladores de um Robô Móvel Omnidirecional*. Dissertação (Mestrado) — Dissertação de Mestrado-Universidade Federal da Bahia, Programa em Pós Graduação em Engenharia Elétrica, Salvador - BA, 2014. 34, 37, 63, 64, 73, 75, 116, 120, 121, 123, 125, 127, 130, 131, 134, 135, 138, 149

30 GARCIA M.F.; COSTA, A. e. F. D. "Projeto e concepção de uma plataforma para robótica cognitiva embarcada". *14º SBAI - Simpósio Brasileiro de Automação Inteligente - Universidade Federal de Ouro Preto-MG*, p. 1–6, 2019. 34

31 KITANO, H. Robocup: A challenge problem for ai. In: SPRING. *IA Magazine*. [S.l.], 1997. p. 73–85. 35

32 KITANO, H.; SANDERSON, A. Robocup: A challenge problem for ai. In: SPRINGER. *IA Magazine*. [S.l.], 1997. p. 73–85. 35

33 2017, R. L. *Futebol de Robôs da Liga Small Size F180*. 2017. <https://www.robocup.org/news/32>. Acesso em: 24 Julho 2020. 35

34 KITANO, H. et al. Robocup: A challenge problem for artificial intelligence. *AI Magazine*, v. 18, n. 1, 1997. 35

35 BOER, R.; KOK, J. *The Incremental Development of a Synthetic Multi-Agent*. Tese (Doutorado) — Amsterdam University, 2002. 35

36 KRAETZCHMAR, G. The ulm sparrows: Research into sensorimotor integration, agency, learning, and multiagent cooperation. In: ROBOCUP. *ROBOCUP WORKSHOP*. [S.l.], 1998. p. 459–465. 35

37 *MecaTeam 2006: Um sistema multiagente reativo para o futebol de robôs simulado*. 36

38 COSTA, A. L. da; BITTENCOURT, G. From a concurrent architecture to a concurrent autonomous agents architecture. p. 274–285, 1999. 36, 37, 57, 63

39 COIMBRA, I. de Sistemas e Robótica Polo de. *Sistemas Multirobôs*. 2020. <https://www.isr.uc.pt/component/content/article/19-contributions/other-contributions/63-cooperative-multi-robot-patrolling>. Acesso em: 01 ago. 2020. 36

40 CORP., S. R. S. *Multi Agent Robotics Systems*. 2020. <https://smprobotics.com/technology_autonomous_mobile_robot/multi-agent-robotics-systems/>. Acesso em: 10 set. 2020. 36

41 PAVEI, J. *COORDENACÃO EM SISTEMAS MULTI-ROBOS UTILIZANDO MÉTODOS BASEADOS EM AUTOMATOS*. Dissertação (Mestrado) — Dissertação de Mestrado (PPGEAS) - UNIVERSIDADE FEDERAL DE SANTA CATARINA, 2011. 37

42 MG, I. T. V. M. O. P. *Robótica Aplicada à Mineração*. 2020. <http://www.itv.org/linha-de-pesquisa/robotica-aplicada-a-mineracao/>. Acesso em: 27 jul. 2020. 39

43 OLIVEIRA, L. L. R. *Controle de Trajetória Baseado em Visão Computacional Utilizando o Framework ROS*. Dissertação (Mestrado) — Universidade Federal de Juiz de Fora, 2013. 39

44 TELEPRESENCE ROBOTS. *Adept MobileRobots*. Disponível em: <https://telepresencerobots.com/adept-mobilerobots>. Acesso em: 02 abr. 2020. 40

45 FARIAS, W. A. *Comparação entre controladores fuzzy e neural desenvolvidos via simulação e transferidos para ambientes reais no âmbito da robótica evolutiva*. Dissertação (Mestrado) — Universidade Federal de Sergipe, 2018. 41

46 CLEARPATH Robotics Inc. Disponível em: <https://clearpathrobotics.com/>. Acesso em: 02 abr. 2020. 41

47 K-TEAM Corporation. Disponível em: <https://www.k-team.com/>. Acesso em: 02 abr. 2020. 42

48 J. CORNE C., H. P. K. J. M. J. C. Incremental evolution of neural controllers for robust obstacle-avoidance in khepera. *Husbands P., Meyer JA. (eds) Evolutionary Robotics. EvoRobots 1998*, Springer, Berlin, Heidelberg, n. Lecture Notes in Computer Science, vol 1468, 1998. 42

49 MARCHI, J. *Navegação de robôs móveis autônomos: Estudo e implementação de abordagens - Dissertação de Mestrado*. Dissertação (Mestrado) — Universidade Federal de Santa Catarina., 2001. 43

50 PIERI, E. R. *CURSO DE ROBÓTICA MÓVEL*. Universidade Federal de Santa Catarina: Programa de Pós-graduação em Engenharia Elétrica, 2002. 43

51 MATARIĆ, M. J. *The Robotics Primer*. [S.l.]: The MIT Press, 2007. 43

52 RIBEIRO, T. T. *Sistema de controle em tempo real aplicado a robótica móvel.* Dissertação (Mestrado) — Escola Politécnica da Universidade Federal da Bahia, 2010. 43, 44, 55, 56, 119, 120

53 ABREU, P. *Robótica Industrial.* Dissertação (Mestrado) — Universidade do Porto: Faculdade de Engenharia - FEUP, 2001. 43

54 COSTA, S. A. F. *GERAÇÃO E CONTROLE DE TRAJETÓRIA DE ROBÔS MÓVEIS OMNIDIRECIONAIS.* Dissertação (Mestrado) — Universidade Federal da Bahia - PPGEE, 2006. 44, 53, 55

55 RUSSELL S. J., N. P. *Artificial Intelligence: A Modern Approach, Second Edition.* [S.l.]: Prentice Hall, 2003. 45, 59, 60, 61, 62

56 VIEIRA, F. C. *Controle Dinâmico de Robôs Móveis com Acionamento Diferencial.* Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, Dissertação de Mestrado, 2005. 45

57 BRAUNL, T. *Embedded Robotics: Mobile Robot Design and Applications with Embedded Systems.* [S.l.]: Springer-Verlag Berlin Heildberg, 2003. 46

58 SECCHI, H. A. *Una Introducción a los Robots Móviles.* [S.l.]: Instituto de Automática - UNSJ: Universidade Nacional de San Juan - Argentina, 2008. 46, 47

59 SPONG M. W., H. M. V. *Robot Modeling and Control.* New York: JOHN WILEY And SONS, INC., 2006. 49

60 NAGARAJAN U., G. K.; HOLLIS, R. L. Trajectory planning and control of an underactuated dynamically stable single spherical wheeled mobile robot. *Robotics and Automation, 2009. ICRA '09. IEEE International Conference*, p. 3743–3748, 2009a. 49

61 NAGARAJAN U., G. K.; HOLLIS, R. L. State transition, balancing, station keeping, and yaw control for a dynamically stable single spherical wheel mobile robot. *Robotics and Automation, 2009. ICRA '09. IEEE International Conference*, p. 998–1003, 2009b. 49

62 INDIVERI, G. Swedish wheeled omnidirectional mobile robots. *Robotics, IEEE Transactions on 25(1)*, p. 164–171, 2009. 49

63 SIEGWART R., I. R. N.; SCARAMUZZA, D. Introduction to autonomous mobile robots - intelligent robotics and autonomous agents. *Mit Press*, 2004. 49

64 SONG, J.-B.; BYUN, K.-S. *Design and Control of an Omnidirectional Mobile Robot with Steerable Omnidirectional Wheels.* Dissertação (Mestrado) — Korea University, Mokpo National University. Republic of Korea, 2008. 50

65 CRAIG, J. J. *Introduction to Robotics.* [S.l.]: 408p. 3.ed. Upper Saddle River: Pearson Education, Inc., 2005. 50

66 LAURA, T. L. *Fuzzy gain scheduling and tuning of multivariable fuzzy control-methods of fuzzy computing in control systems.* Dissertação (Mestrado) — Universidade Federal da Bahia - Dissertação de Mestrado, 2006. 51

67 CHUNG W.; MOON, C.-b. J. C. J. J. Design of the dual offset active caster wheel for holonomic omni-directional mobile robots. *International Journal of Advanced Robotic Systems*, p. [S.l.], v.7, n.4, p.105–110, 2010. 51, 53

68 CAMPION G.; BASTIN, G. D. A.-N. Structural properties and classification of kinematic and dynamic models of wheeled mobile robots. *IEEE Transactions on Robotics and Automation*, p. [S.l.], v.12, n.1, p.47–62, 1996, 1996. 51

69 HOLMBERG R.; KHATIB, O. Development and control of a holonomic mobile robot for mobile manipulation tasks. *International Journal of Robotics Research, Salt Lake City*, p. v.19, n.11, p.1066–1074, 2000. 51, 52

70 AIRTRAX, S. *Omni-directional Lift Truck*. 2014. <http://www.airtrax.com/>. Acesso em: 24 jul. 2020. 52

71 KUKA. *youBot Datenblatt*. 2015. <https://www.kuka.com/youBot_Datenblatt_2015_web.pdf>. Acesso em: 28 jan. 2020. 52

72 CORREIA, M. D. *Modelagem de robôs móveis com rodas omnidirecionais incluíndo modelos estáticos de atrito*. Dissertação (Mestrado) — Universidade Federal da Bahia - PPGEE, 2012. 53

73 CRUZ, A. P. B. D. *Modelagem Dinâmica de uma Estrutura de Base Holonômica para Robôs Móveis: Inclusão das Não-Linearidades de Entrada*. Dissertação (Mestrado) — Universidade Federal da Bahia - PPGEE, 2007. 53

74 BORENSTEIN J.; EVERETT, H. R. F.-L. *'Where am I?' Sensors and Methods for Mobile Robot Positioning*. Dissertação (Mestrado) — University of Michigan, 1996. 54

75 SIEGWART ROLAND ; NOURBAKHSH, I. R.-S. D. *Introduction to Autonomous Mobile Robots*. [S.l.]: London, The MIT Press, 2011. 54

76 SAMANI H. A., A. A. O.-H. D. M. Comprehensive omni-directional soccer player robots. *International Journal of Advanced Robotic Systems*, 2007. 54

77 LYNCH, K. M.; PARK, F. C. *MODERN ROBOTICS MECHANICS, PLANNING, AND CONTROL*. [S.l.]: Cambridge University, 2017. 54

78 CONCEIçãO A.G.S., A. M.; COSTA, P. Model identication of a four wheeled omni-directional mobile robot. *IEEE International Conference on Emerging Technologies and Factory Automation - ETFA*, p. 227–234, 2006. 55

79 BARRETO, J. C. *Controle Preditivo de um Robô Omnidirecional com Compensação de Atrito*. Dissertação (Mestrado) — Escola Politécnica - Universidade Federal da Bahia, 2011. 55

80 RIBEIRO, T. R. F. J. S. e. A. C. Practical approach of modelling and parameters estimation for omnidirectional mobile robots. *IEEE Transactions on Mechatronics 14(3)*, p. 377–381, 2009b. 55

81 RAFFO, G. V. *Algoritmos de controle preditivo para seguimento de trajetórias de veículos autônomos*. Dissertação (Mestrado) — Uiversidade Federal de Santa Catarina - PPGEE, 2006. 55, 57

82 WATANABE, K. Control of an omnidirectional mobile robot. *Second International Conference on Knowledge-Based Intelligent Electronic System.*, p. 51–60, 1998. 55

83 OUBBATI, M. *Neural Dynamics for Mobile Robot Adaptive Control*. Dissertação (Mestrado) — Tese (Doutorado) | Institut fAur Parallele und Verteilte Systeme der UniversitAat Stuttgart, Tag der mAundlichen PrAufung, 2006. 55, 56

84 TOULSON, R. e. T. W. Fast and effctive embedded systems design: applying the arm mbed. *Elsevier*, 2012. 57, 58, 129

85 JUNG C. R.; OSÓRIO, F. S. K. C. R. H. F. J. *Computação Embarcada: Projeto e Implementação de Veículos Autônomos Inteligentes*. [S.l.]: Congresso da Sociedade Brasileira de Computação, 2005. 1359 p. 57

86 JIMéNEZ M; PALOMERA, R. C. I. *Introduction to Embedded Systems*. University of Puerto Rico at Mayaguez: Springer, 2014. 58, 59

87 MAES PATTIE, M. M.; YEZDY, L. Collaborative interface agents. 1994. 59, 60

88 COSER, A. *Utilização de agentes inteligentes no trabalho colaborativo via internet.* Dissertação (Mestrado) — Universidade Federal de Santa Catarina - Florianópolis, 1999. 59

89 MAES, P. *"Artificial Life Meets Entertainment: Life like Autonomous Agents," Communications of the ACM,*. [S.l.: s.n.], 1995. 38, 11, 108-114 p. 59

90 NORVIG, P.; RUSSELL, S. *Inteligência Artificial, 3a. Edição*. [S.l.]: Elsevier, 2014. 59

91 CHARNIAK, E.; MCDERMOTT. *Introduction to Artificial Intelligence*. Massachusetts: Addison-Wesley, Reading, 1985. 60

92 BELLMAN, R. E. *An Introduction to Artificial Intelligence: Can Computers Think?* San Francisco: Boyd and Fraser Publishing Company, 1978. 60

93 WINSTON, P. H. *Artificial Intelligence*. Massachusetts third edition: Addison-Wesley, Reading, 1992. 60

94 HAUGELAND, J. *Artificial Intelligence: The Very Idea*. Massachusetts: MIT Press, Cambridge, 1985. 60

95 KURZWEIL, R. *The Age of Intelligent Machines*. Massachusetts: MIT Press, Cambridge, 1990. 60

96 PROOLE D, M. A. G. R. *Computational Intelligence: a Logical Approach*. England: Oxford University Press, 1998. 60

97 RICH, E.; KNIGHT, K. *Artificial Intelligence*. New York: McGraw-Hill, second edition, 1991. 60

98 NILSSON, N. J. *Artificial Intelligence: A New Synthesis*. Califórnia: Morgan Kaufmann Publishers, Inc., 1998. 60

99 COSTA, A. L. da; BITTENCOURT, G. Dynamic social knowledge: A comparative evaluation. *Lecture notes in computer science.*, Springer., p. 176–185, 2000. 60

100 *Intelligence Artificielle Distribuée*. 60

101 COSTA, A. C. P. L. d.; OTHERS. *Conhecimento social dinâmica: uma estratégia de cooperação para sistemas multiagentes cognitivos*. Tese (Doutorado) — Universidade Federal de Santa Catarina, Florianópolis, SC., 2001. 61, 62

102 BARROS, L. N. de. *Conhecimento e Raciocínio - Agentes Lógicos*. 2020. <https://www.ime.usp.br/~leliane/IAcurso2006/slides/Aula7-LProposicional-I-2006.pdf>. Acesso em: 01 ago. 2020. 61

103 DAVIDSON, P. *"Concept Acquisition by Autonomous Agents: Cognitive Modeling versus Engineering Approach". Lund University Studies 12, ISSN 1101-8453, Lund University, Suécia, 1992*. 2014. <http://fileadmin.cs.lth.se/ai/psfiles/LUCS-12.pdf>. Acesso em: 08 jan. 2020. 61, 62

104 REVISTA ELETRôNICA DA FACULDADE METODISTA GRANBERY. *Princípios e Aspectos Sobre Agentes Inteligentes.* 1-29 p. 62

105 HUHNS, M.; SINGH, M. *Readings in Agents.* [S.l.: s.n.], 1998. 62

106 BARBOSA, L. R. T. et al. *Um sistema multiagente para monitoramento atmosférico.* Dissertação (Mestrado) — Universidade Salvador, 2005. 63

107 BITTENCOURT, G.; COSTA, A. da. Hybrid cognitive model. 2001. 63

108 CERQUEIRA R. G., A. L. d. C. S. G. M. D. L. e. G. P. From reactive to cognitive agents: Extending reinforcement learning to generate symbolic knowledge bases. *SBAI - Simpósio Brasileiro de Automação Inteligente*, 2013. 63

109 BITTENCOURT, G. In the quest of the missing link. *IJCAI 15 International Join Conference on Artificial Inteligence, Nagoia, Japan*, p. 310–315, 1997. 63

110 COSTA, A. L. e. a. Embarcando o agente autonomo concorrente no robo movel omnidirecional AxeBot: nivel reativo. X SBAI - Simpósio Brasileiro de Automação Inteligente, p. 635–640, 2011. 63, 73

111 TSOUKALAS L. H.; UHRIG, R. E. *Fuzzy and Neural Approaches in Engineering.* New York, USA: John Wiley and Sons, Inc., 1996. 65

112 SIMÕES MARCELO G.; SHAW, I. S. *Controle e Modelagem Fuzzy.* São Paulo: Editora Edgard Blucher Ltda., 2007. 65, 66

113 TSOUKALAS L. H.; UHRIG, R. E. *Lógica Nebulosa.* São José dos Campos - SP: V Escola de Redes Neurais: Conselho Nacional de Redes Neurais, ITA, 1999. 65

114 TSOUKALAS L. H.; UHRIG, R. E. *Fuzzy Logic: A Practical Approach.* [S.l.]: Morgan Kauf-mann, 1994. 65

115 VILJAMAA, P. *Fuzzy gain scheduling and tuning of multivariable fuzzy control-methods of fuzzy computing in control systems.* Dissertação (Mestrado) — Automatic Control Institute, Tampere University of Technology, 2012. 66

116 AWARD O. A., M. R. A. Full parameterization process for singleton fuzzy logic controllers: A computing algorithm. *International Journal of Engineering Research and Technology (IJERT)*, p. 1057–1063, 2014. 66

117 WANG, L. X. A supervisory controller for "fuzzy control systems that guarantee stability". *IEEE Transactions on Automatic Control, vol.39*, p. 1845–1847, 1994. 67, 68

118 ALMEIDA P. E. M., E. A. G. *Sistemas Inteligentes: Fundamentos e Aplicações, cap. Sistemas Fuzzy.* [S.l.]: Manole, Barueri, 2005. 67

119 ZHANG, P. Advanced industrial control technology. In: . [S.l.: s.n.]. 69, 70, 73

120 PAZUL, K. Controller area network (can) basics. In: . [S.l.: s.n.], 1999. 69

121 RICHARDS, P. A can physical layer discussion. In: . [S.l.: s.n.], 2002. 70

122 INCORPORATED, T. I. Sn65hvd23x 3.3-v can bus transceivers. In: . [S.l.: s.n.], 2001. 70, 72

123 RANJITH, M. Application note 52701: Getting started with controller area network (can). In: . [S.l.: s.n.], 2013. 71, 72, 73

124 CORRIGAN, S. Introduction to the controller area network (can). In: . [S.l.: s.n.], 2002. 71

125 FORTE, C. H. V. *Processadores ARM: Visão Geral e Aplicações.* [S.l.]: DCCE - Universidade Estadual Paulista, 2015. 1–16 p. 77

126 WRTNODE. *OpenJumper ElecSpark*. 2020. <http://wrtnode.cc/html/hardware_1.html>. Acesso em: 28 jul. 2020. 78

127 FORUM, F. *ESP32S-HiLetgo Dev Boad with Pinout Template*. 2020. <https://forum.fritzing.org/t/esp32s-hiletgo-dev-boad-with-pinout-template/5357>. Acesso em: 28 jul. 2020. 80

128 TUTORIAIS, T. e. T. F. K. *ESP32: Diagrama em blocos*. 2020. <https://www.fernandok.com/2018/03/esp32-detalhes-internos-e-pinagem.html>. Acesso em: 28 jul. 2020. 81

129 INC., G. *GitHub Help*. 2020. <https://github.com/>. Acesso em: 26 ago. 2020. 81

130 WU, W. Dcmotor parameter identification using speed step responses. *Hindawi Publishing Corporation Modelling and Simulation in Engineering*, p. 1–6, 2012. 94

131 UNIVERSITY, M. U. M. U. M. *Control Tutorials for Matlab and Simulink*. 2020. <http://ctms.engin.umich.edu/CTMS/index.php?aux=ActivitiesDCmotorB>. Acesso em: 29 jul. 2020. 94

132 PETTEY, B. *Servocity - We have the parts for your ideas*. 1994. <https://www.servocity.com/624-rpm-premium-planetary-gear-motor-w-encoder>. Acesso em: 29 jul. 2020. 96

133 LJIUNG, L. *System Identification: Theory for the User*. [S.l.]: Prentice Hall, 2nd edition, 1999. 96

134 UNBEHAUEN, H.; RAO, G. P. *A review of identification in continuous-time systems*. [S.l.]: Annual Reviews in Control, vol. 22, 1998. 96

135 FRANKLIN, J. D. P. G. F.; WORKMAN, M. L. *Digital Control of Dynamic Systems*. [S.l.]: Addison Wesley, 2nd edition, 1990. 96

136 BASILIO, J. C.; V.MOREIRA, M. "state-space parameter identification in a second control laboratory". *IEEE Transactions on Education, vol. 47, no. 2*, p. 204–210, 2004. 96

137 MAMANI, J. B. G.; FELIU-BATLLE, V. "on-line fast algebraic parameter and state estimation for a dc motor applied to adaptive 16 control". In: . [S.l.: s.n.], 2008. 96

138 RUDERMAN J. KRETTEK, F. H. M.; BETRAN, T. Optimal state space control of dc motor. In: . [S.l.: s.n.]. p. 5796–5801. 96

139 ROBOTICS, P.; ELECTRONICS. *Pololu G2 High-Power Motor Driver 18v17*. 2020. <www.pololu.com/product/2991>. Acesso em: 28 jul. 2020. 97, 100, 104

140 ALPHA; SEMICONDUCTOR, O. *30V N-Channel AlphaMOS*. 2020. <http://www.aosmd.com/res/data_sheets/AON7418.pdf>. Acesso em: 28 jul. 2020. 97

141 INSTRUMENTS, T. *DRV8701 Brushed DC Motor Full-Bridge Gate Driver - Datasheet SLVSCX5B*. 2015. <https://www.ti.com/product/DRV8701>. Acesso em: 29 jul. 2020. 98, 99

142 DORF RICHARD, B. R. *Sistemas de Controle Moderno. 8ª edição. Rio de Janeiro-RJ*. [S.l.]: LTC Editora, 2001. 107

143 S., T.; STOICA, P. *System Identification*. [S.l.]: Prentice-Hall, London, United Kingdom - New York, 1989. 109

144 ROJAS, R.; FORSTER, A. G. Holonomic control of a robot with an omnidirectional drive. *To appear in KI - Kunstliche Intelligenz, BottcherIT Verlag*, p. 1–7, 2006. 118

145 TSAI CHING-CHIH, L.-B. J. T.-Y. W. e. T.-S. W. Kinematics control of an omnidirectional mobile robot. In: . [S.l.: s.n.], 2005. 120

146 AMAZON. *CAN Bus Module Transceiver TJA1050*. 2020. <https://www.amazon.com/Comimark-Transceiver-TJA1050-Controller-Schnittstelle/dp/B07W4VZ2F2>. Acesso em: 29 jul. 2020. 121

147 NXP. *Transceiver CAN TJA1050*. 2020. <https://www.nxp.com/products/interfaces/can-transceivers/legacy-can/high-speed-can-transceiver:TJA1050>. Acesso em: 29 jul 2020. 121, 122

148 CORP., C. S. *PSoC 5LP Prototyping Kit Guide*. 2020. <https://www.cypress.com/file/157971/download>. Acesso em: 29 jul. 2020. 124, 125

149 CHEN, L. Arm cortex-m - the architecture for the digital world. In: . [S.l.: s.n.], 2010. 126

150 SHAKIBAFAR, B. *Early thought on MBED-OS*. 2019. <https://medium.com/@b.shakibafar/early-thought-on-mbed-os-72a04728c0d4>. Acesso em: 30 jul. 2020. 127

151 SOLUTIONS, L. Y. challenges our. *L-Tek FF-LPC546xx*. 2019. <https://www.l-tek.com/l-tek-products/l-tek-ff-lpc546xx/>. Acesso em: 30 jul. 2020. 128

152 RENESAS. *GR-PEACH*. 2010. <https://www.renesas.com/br/en/products/gadget-renesas/boards/gr-peach.html>. Acesso em: 30 jul. 2020. 130

153 MBED, A. *GR-PEACH*. 2010. <https://os.mbed.com/platforms/Renesas-GR-PEACH/>. Acesso em: 30 jul. 2020. 131

154 GROUP, C. *GR-PEACH - Quadro de referência*. 2010. <https://www.core.co.jp/service/iot/gr-peach>. Acesso em: 30 jul. 2020. 132

155 (UESPI-TERESINA) *Robotic Research Group* (RRG) Universidade Estadual do P. *eFLL - A Fuzzy Library for Arduino and Embedded Systems*. 2020. <https://blog.zerokol.com/2012/09/arduinofuzzy-uma-biblioteca-fuzzy-para.html>. Acesso em: 18 fev. 2020. 140

Apêndices

APÊNDICE A – Trabalhos Publicados

Seguem abaixo os resumos dos trabalhos científicos publicados com os resultados encontrados da pesquisa, que fundamentou esta dissertação.

A.1 Artigos em Congressos

1. *Marcio Figueiredo Garcia, Augusto Cesar Pinto Loureiro da Costa, Diego Stéfano Fonseca Ferreira.* **AN EMBEDDED NETWORK BASED PLATFORM FOR COGNITIVE ROBOTICS**. *8th Brazilian Conference on Intelligent Systems (BRACIS).* Salvador - BA, Outubro de 2019.

Resumo:

O agente autônomo concorrente AAC é uma arquitetura cognitiva composta por três níveis que são executados simultaneamente: o nível reativo, que executa o ciclo de percepção-ação usando comportamentos reativos; o nível instintivo, que coordena a execução do plano; e o nível cognitivo, responsável pelo planejamento. Neste trabalho, é apresentada uma arquitetura de hardware dedicada à execução da arquitetura de agentes cognitivos em um robô omnidirecional autônomo.

2. *Marcio Figueiredo Garcia, Augusto Cesar Pinto Loureiro da Costa, Diego Stéfano Fonseca Ferreira.* **PROJETO E CONCEPÇÃO DE UMA PLATAFORMA PARA ROBÓTICA COGNITIVA EMBARCADA**. *14º SIMPÓSIO BRASILEIRO DE AUTOMAÇÃO INTELIGENTE - SBAI.* Universidade Federal de Ouro Preto-MG (UFOP), Outubro de 2019.

Resumo:

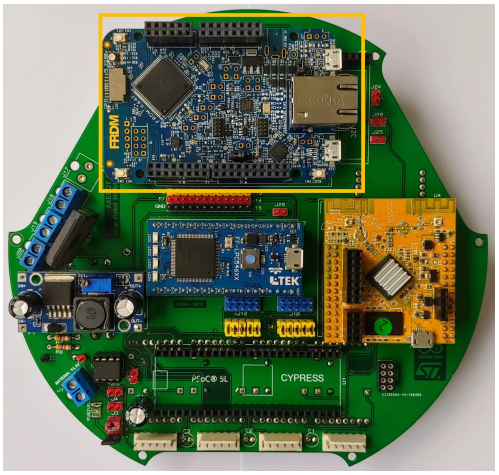
O Agente Autônomo Concorrente (AAC) é uma arquitetura cognitiva composta por três níveis que executam concorrentemente: o nível reativo, que executa o ciclo percepção-ação do agente utilizando comportamentos reativos; o nível instintivo, que coordena a execução de um plano; e o nível cognitivo, responsável pelo planejamento. Neste trabalho, configurou-se uma arquitetura de hardware para a execução do AAC no robô omnidirecional *AxeBot II*. O trabalho corrige erros de trabalhos anteriores e concebe uma plataforma embarcada completa para robótica cognitiva. Os resultados mostram o desenvolvimento do hardware e apresentam um experimento que valida o funcionamento da plataforma.

APÊNDICE B – Dados e referências de plataformas de desenvolvimento

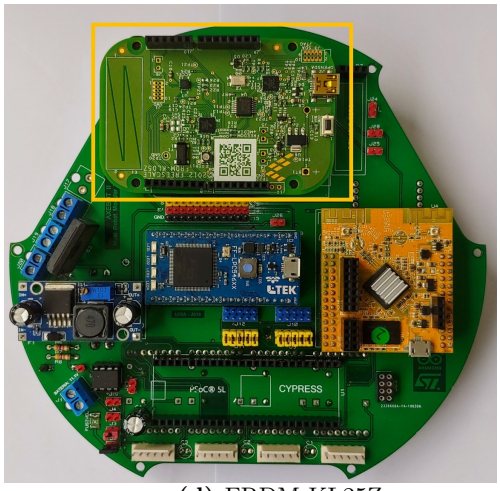
Projetos podem ser criados baseado-se no sistema operacional *mbed*, um ambiente de programação onde se utiliza plataformas de desenvolvimento baseadas em microcontroladores ARM® Cortex®. Esse ambiente, possui uma ampla gama de opções de conectividade com suporte para bibliotecas de software, hardware, tutoriais e exemplos de códigos para diversas aplicações. O ambiente de desenvolvimento *mbed* OS (*Operational System*) suporta as principais famílias de microcontroladores, como os de alguns fabricantes desses dispositivos, como: STM32, Kinetis, FRDM (NXP), Nordic, Cypress, nRF52, dentre outros. A Figura 116 mostra 4 exemplos utilizando plataformas FRDM da *NXP* ARM® Cortex®.

Figura 116 – Plataformas da *Freescale*.

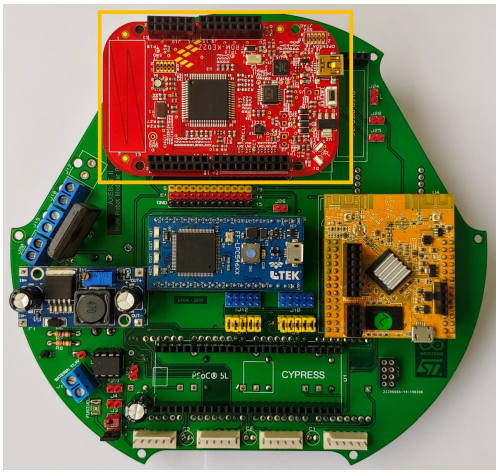
(a) FRDM-K64F



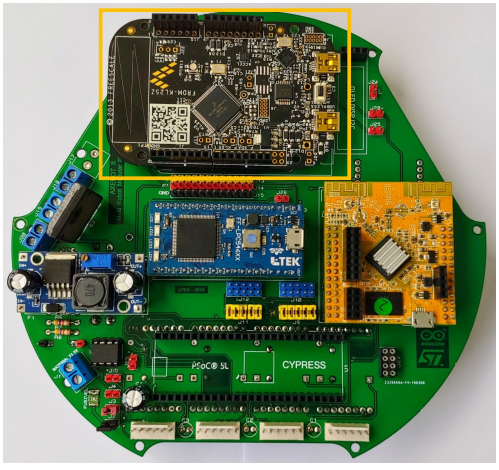
(b) FRDM-KL05Z



(c) FRDM-KE02Z



(d) FRDM-KL25Z



Fonte: Do autor.

A Tabela 20 a seguir lista algumas dessas plataformas disponíveis comercialmente na qual possuem medidas padrão Arduino compatíveis com a PCB desenvolvida.

Tabela 20 – Plataformas compatíveis com a PCB.

Plataforma	Fabricante	Característica da CPU
HANI-IoT	Arrow Electronics	ARM® Cortex®-M33
ThunderSoft TT_M4G9	TOSHIBA	ARM® Cortex®-M4
ThunderSoft TT_M3HQ	TOSHIBA	ARM® Cortex®-M3
NUCLEO-WB55RG	ST Eletronics	ARM® Cortex®-M4
Sequana PSoC63	FUTURE Eletronics	ARM® Cortex®-M0+/M4
NUCLEO-R433RC-P	ST Eletronics	ARM® Cortex®-M4
GR-PEACH	RENESAS	ARM® Cortex®-A9
FRDM-KL82Z	NXP Semiconductors	ARM® Cortex®-M0+
FRDM-K82F	NXP Semiconductors	ARM® Cortex®-M4
FRDM-K66F	NXP Semiconductors	ARM® Cortex®-M4
FRDM-KL43Z	NXP Semiconductors	ARM® Cortex®-M0+
FRDM-KL27Z	NXP Semiconductors	ARM® Cortex®-M0+
FRDM-K22F	NXP Semiconductors	ARM® Cortex®-M4
FRDM-KL46Z	NXP Semiconductors	ARM® Cortex®-M0+
FRDM-KL25Z	NXP Semiconductors	ARM® Cortex®-M0+
FRDM-K64F	NXP Semiconductors	ARM® Cortex®-M4
FRDM-KL05Z	NXP Semiconductors	ARM® Cortex®-M0+
FRDM-KW41Z	NXP Semiconductors	ARM® Cortex®-M0+
WIZwiki-W7500P	WIZnet	ARM® Cortex®-M0
B96B-F446VE	ST Eletronics	ARM® Cortex®-M4
Seeed Arch Link	Seeed Studio	ARM® Cortex®-M0
NUCLEO-F401RB	ST Eletronics	ARM® Cortex®-M4
NUCLEO-F446RE	ST Eletronics	ARM® Cortex®-M4
NUCLEO-F410RB	ST Eletronics	ARM® Cortex®-M4
FRDM-KW24D512	NXP Semiconductors	ARM® Cortex®-M4
NUCLEO-L476RG	ST Eletronics	ARM® Cortex®-M4
NUCLEO-L073RZ	ST Eletronics	ARM® Cortex®-M0+
NUCLEO-F070RB	ST Eletronics	ARM® Cortex®-M0
NUCLEO-F091RC	ST Eletronics	ARM® Cortex®-M0
Seeed Arch BLE	Seeed Studio	ARM® Cortex®-M0
NUCLEO-F411RE	ST Eletronics	ARM® Cortex®-M4
NUCLEO-F303RE	ST Eletronics	ARM® Cortex®-M4
LPCXpresso824-MAX	NXP Semiconductors	ARM® Cortex®-M0+
FRDM-K20D50M	NXP Semiconductors	ARM® Cortex®-M4
RedBearLab nRF51822	NORDIC Semiconductors	ARM® Cortex®-M0
LPCXpresso11u68	NXP Semiconductors	ARM® Cortex®-M0+
LPCXpresso1549	NXP Semiconductors	ARM® Cortex®-M3
NUCLEO-F334R8	ST Eletronics	ARM® Cortex®-M4
NUCLEO-F072RB	ST Eletronics	ARM® Cortex®-M0
NUCLEO-F030R8	ST Eletronics	ARM® Cortex®-M0
NUCLEO-F401RE	ST Eletronics	ARM® Cortex®-M4

Plataforma	Fabricante CPU	Característica
NUCLEO-L053R8	ST Eletronics	ARM® Cortex®-M0+
NUCLEO-R152RE	ST Eletronics	ARM® Cortex®-M3
NUCLEO-F302R8	ST Eletronics	ARM® Cortex®-M4
Seeeduino-Arch-Pro	Seeed Studio	ARM® Cortex®-M3
Seeeduino-Arch	Seeed Studio	ARM® Cortex®-M0
NUCLEO-F103RB	ST Eletronics	ARM® Cortex®-M3
LPC800-MAX	NXP Eletronics	ARM® Cortex®-M0+
Genuino 101	Intel	x86(Quark) + ARM® (ARC)
Arduino YUN Rev 2	ARDUINO	ATmega32U4
Arduino UNO WIFI REV2	ARDUINO	ATmega4809
Arduino Leonardo	ARDUINO	ATmega32u4
Arduino ZERO	ARDUINO	ARM® Cortex®-M0+
DFRduino UNO R3	DFROBOT	AVR 8 *bits*
GR-LYCHEE	RENESAS	ARM® Cortex®-A9
GR-SAKURA	RENESAS	RX63N MCU
Spresense	SONY	CXD5602
XLR8	Alorium Technology	AVR
WiFi UNO R3	NodeMCU Wemos	Atmega328P + ESP8266
A-Star SV	Pololu	ATmega32U4 Prime microSD
MAX32600MBED	Maxim Integrated	ARM® Cortex®-M3
EVK-ODIN-W2	u-blox	Cortex®-M4 with FPU
Seeed Arch Link	Seeed Studio	ARM® Cortex®-M0
GD32-E103VB	GigaDevice	ARM® Cortex®-M4
RTL8195AM	Realtek	ARM® Cortex®-M3

Os aplicativos para as plataformas com suporte *mbed* podem ser desenvolvidos utilizando a IDE (do inglês, *Integrated Development Environment*) *mbed*, um editor e compilador de código on-line. Apenas um navegador web precisa ser instalado no PC, uma vez que um projeto é compilado em nuvem, ou seja, em um servidor remoto, usando o compilador ARMCC C/C++. O *mbed* OS fornece espaços de trabalho privados com funções para importar, exportar e compartilhar código e também pode ser usado para geração de documentação desses códigos. Os aplicativos também podem ser desenvolvidos com outros ambientes de desenvolvimento, como *Keil uVision*, IAR *Embedded Workbench* e *Eclipse* com ferramentas embarcadas GCC ARM®. Já para as plataformas de 8*bits* da família AVR, um compilador GCC desenvolvido em linguagem C e $C++$, que é baseado em *Wiring* utiliza uma interface gráfica em Java no projeto *Processing* lograda a uma IDE simples e intuitiva do próprio Arduino.

As plataformas citadas são apenas algumas das opções encontradas em pesquisa, na qual possuem outras plataformas que possuem *Arduino pin headers*, porém com dimensões incompatíveis com as dimensões da PCB desenvolvida neste trabalho. Devido a isso, atualizações futuras da PCB, com o projeto de uma versão atualizada das dimensões físicas, ampliarão suas funcionalidades.