UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE AUTOMAÇÃO E SISTEMAS

Volnei Fontana Júnior

Desenvolvimento e Cooperação de Robôs Através da Plataforma Arduino

Florianópolis
2018

Volnei Fontana Júnior

Desenvolvimento e Cooperação de Robôs Através da Plataforma Arduino

Relatório submetido à Universidade Federal de Santa Catarina como requisito para a aprovação na disciplina **DAS 5511: Projeto de Fim de Curso,** do curso de Graduação em Engenharia de Controle e Automação. Orientador: Prof. Dr. Ubirajara Franco Moreno

Florianópolis
2018

Volnei Fontana Júnior

Desenvolvimento e Cooperação de Robôs Através da Plataforma Arduino

Esta monografia foi julgada no contexto da disciplina DAS5511: Projeto de Fim de Curso e aprovada na sua forma final pelo Curso de Engenharia de Controle e Automação.

Florianópolis, 14 de dezembro de 2018.

Banca Examinadora:

Prof. Dr. Ubirajara Franco Moreno
Orientador na Empresa
Laboratório de Projetos – Departamento de Automação e Sistemas

Prof. Dr. Ubirajara Franco Moreno
Orientador no Curso
Universidade Federal de Santa Catarina

Prof. Dr. Carlos Barros Montez
Avaliador
Universidade Federal de Santa Catarina

Bruno Rodrigues Battistotti
Debatedor
Universidade Federal de Santa Catarina

Nathaniel Salvador de Oliveira
Debatedor
Universidade Federal de Santa Catarina

Dedico este trabalho à minha família, amigos e professores.

AGRADECIMENTOS

Em primeiro lugar, à minha amada e querida mãe Maria José Isoppo Silva e ao meu pai, Volnei Fontana, por todo o carinho e por despertarem em mim, o amor pela Ciência e Tecnologia. Aos meus irmãos Sarah Fontana e Gustavo Fontana, por todo o carinho e apoio.

Aos meus avós, Maria Cleci Isoppo, Emília Possamai Fontana.

À memória de meus bisavós, João Isoppo e Helena Peretto Isoppo e aos meus avôs, Domingos Bússolo Silva e Antônio Afonso Fontana.

À minha tia, Taia Mara Isoppo Silva, quem ajudou a cuidar de mim, por vários anos, junto aos meus pais. Tenho um carinho especial, foi para mim minha segunda mãe.

Aos meus tios, Domingos Renato Isoppo Silva, Charles Isoppo e Marcos Ivanoé Isoppo, pelas conversas e conselhos, enriquecedores. Tal como meus pais, contribuíram em minha inclinação pelas Ciências Exatas.

Agradeço ao meu tio, Carlos Schmidt e tias, Maura Isoppo Schmidt, Kátia Vivian Isoppo e Josimari Isoppo.

À Minha tia, Sílvia Clenildes Isoppo, sou grato pelo carinho e apoio concedidos, sobretudo no decorrer de toda minha graduação.

À amiga Ane Abreu, pela solidez e amizade de longa data. Sempre inspiradora a todos à sua volta. Sou grato às suas, sempre nobres, palavras.

Aos meus primos, Jaqueline Isoppo da Cunha, João Carlos Isoppo, José Isoppo, Helena Carla Isoppo Schmidt, Keity Isoppo, Maycon Fontana, Vinícius Isoppo, Higor Isoppo e Larissa Isoppo.

Ao meu Professor e orientador, Dr. Ubirajara Franco Moreno, o qual contribuiu com seus ensinamentos e auxílios para a realização deste trabalho, bem como teve importante papel em minha trajetória como graduando.

Aos pós-graduandos: Feres, Renan, Sidney, Miguel, Gabriel Manoel e Rafael, por todo o apoio dado à minha atividade de PFC.

Agradeço a todos os amigos, que fazem, ou fizeram parte da minha vida, entre eles: Maria Júlia de Abreu, Maria Zaida de Oliveira Guedes, Diego Cidade, Diego Neves, Daniele Pacheco, Andy, Leandro Guedes, Júlio César Mosena, Marcela Caetano Alves, Thiago de Souza Clézar, Edevar Boteon, Graziele Raupp, Renata Coelho, Paula Romano, Bruno Ricci Piacenti do Santos, Thiado Harry, Anna Paula Vitali, Merlin Lundi, Laísa Santos, Mirian Carvalho, Natália Carla Miranda, Fabrícia Dalmas, Ramon Costamilan, Jair Júnior, José Henrique de Oliveira, Felipe Pardo, Felipe Ramos, Felipe Isoppo, Juliana Tiskoski, Jéssica Villa Real, Thaísa Reis Vilanova e Vinícius Cardozo Macagnan.

Concluo meus agradecimentos, com a palavra "*ubuntu*[1]*".*

[1] Palavra de origem bantu, a qual significa "Sou o que sou pelo que todos nós somos".

“A verdade é filha do tempo, e não da autoridade”.

Galileu Galilei

RESUMO

A Robótica, apesar de já consolidada, seja na indústria, ou em outras áreas, possui significativa potencialidade para pesquisa. Estudos e projetos variados vêm sendo desenvolvidos, para a solução de diversos problemas, constantemente crescentes. Um dos problemas está relacionado ao fato de que, muitas vezes, um robô é incapaz de realizar uma tarefa de forma individual. Pode-se compensar esse problema utilizando uma maior quantidade de robôs, de modo que possa haver cooperação entre eles, a fim de cumprirem a tarefa. A Robótica Cooperativa[2], pertinente ao campo de Sistemas Multi-Robôs[3], faz-se cada vez mais necessária (SATO; SANTOS; ALMEIDA, 2015). Explorando essa necessidade, se deu o desenvolvimento deste Projeto de Fim de Curso. Portanto, são apresentadas nesta monografia, as atividades pertinentes ao Projeto de Fim de Curso, desenvolvidas ao longo de todo o ano letivo de 2018, realizadas no Laboratório de Projetos (LPR) e Laboratório de Montagem Mecatrônica (LMM), pertencentes ao Departamento de Automação e Sistemas. O projeto consistiu em desenvolver protótipos em Robótica Educacional (robôs móveis), utilizando a plataforma Arduino, como alternativa aos robôs LEGO utilizados até então. A partir dos protótipos desenvolvidos, foram criadas estratégias de cooperação entre os mesmos. Ao longo da monografia, serão abordados os seguintes pontos: as motivações do desenvolvimento dos protótipos, a visão arquitetural, o projeto, implementação dos protótipos, as estratégias de cooperação e testes dos protótipos.

Palavras-chave: Robôs móveis. Arduino. Cooperação.

[2] Área da Robótica pertinente à Robótica Coletiva, isto é, lidar com sistemas onde há mais de um agente. A Robótica Cooperativa estuda a associação com outros agentes, de forma a haver benefício mútuo entre si (CAO; FUKUNAGA; KAHNG, 1997).

[3] Do ponto de vista geral, pode ser caracterizado como um conjunto de robôs operando no mesmo ambiente. No entanto, sistemas robóticos podem ser desde simples sensores às máquinas semelhantes ao comportamento humano, capazes de interagir com o ambiente de formas relativamente complexas (FARINELLI; IOCCHI; NARDI, 2004).

ABSTRACT

Robotics, although already consolidated, whether in industry or in other areas, has significant potential for research. Various studies and projects have been developed to solve a number of constantly increasing problems. One of the problems is related to the fact that, often, a robot is unable to perform a task individually. You can compensate for this problem by using a larger number of robots, so that there can be cooperation between them in order to accomplish the task. Cooperative Robotics[4], pertinent to the field of Multi-Robot Systems[5], is becoming increasingly necessary (SATO; SANTOS; ALMEIDA, 2015). Exploring this need, the development of this End-of-Course Project was developed. Therefore, we present in this monograph the activities related to the End-of-Course Project, developed throughout the academic year of 2018, held in the Laboratory of Projects (LPR) and Mechatronics Assembly Laboratory (LMM), belonging to the Department of Automation and Systems. The project consisted in developing prototypes in Educational Robotics (mobile robots), using the Arduino platform, as an alternative to the LEGO robots used until then. From the developed prototypes, cooperation strategies were created between them. Throughout the monograph, the following points will be addressed: prototype development motivations, architectural vision, design, prototype implementation, cooperation strategies and prototype testing.

Key-words: Mobile robots. Arduino. Cooperation.

[4] Robotics area pertinent to Collective Robotics, that is, to deal with systems where there is more than one agent. Cooperative Robotics studies the association with other agents, in order to be mutually beneficial (CAO; FUKUNAGA; KAHNG, 1997).

[5] From a general point of view, it can be characterized as a set of robots operating in the same environment. However, robotic systems can be from simple sensors to machines similar to human behavior, capable of interacting with the environment in relatively complex ways (FARINELLI; IOCCHI; NARDI, 2004).

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

LMM – Laboratório de Montagem Mecatrônica.

LPR – Laboratório de Projetos.

CPU – *Central Processing Unit* (unidade central de processamento).

MOSFET - Transistor de efeito de campo metal-óxido semicondutor.

PIC - *Peripheral interface controller* – Controlador de interface periférica.

RF – Rádio-Frequência.

PCB – Printed Circuit Board – Placa de circuito impresso.

CI – Circuito Integrado.

AMPOP – Amplificador Operacional.

ADC – *Analog-to-digital Converter* – Conversor analógico-digital.

EEPROM – *Electrically erasable programmable read-only-memory* – memória somente-para-leitura programável eletricamente apagável.

LISTA DE SÍMBOLOS

A – Ampère (unidade de medida de corrente elétrica).

V – Volt (unidade de medida de tensão elétrica).

Ω - Ohm (unidade de medida de resistência elétrica).

F – Farad (unidade de capacitância).

m – metro (unidade de comprimento).

f – frequência (Hz).

λ – comprimento de onda (m).

v – velocidade (m/s).

MHz – MegaHertz (unidade de frequência).

mAh – miliAmpère-hora (unidade de carga, geralmente utilizada em baterias).

mils – milésimos de polegada (unidade para os desenvolvimento das PCB's).

CC – Corrente Contínua.

X – Distância percorrida (cm).

Θ – Ângulo (para o trabalho desenvolvido, sua unidade é graus).

V_{som}– Velocidade do som no ar (m/s).

t_{echo} – Tempo em que o sinal ECHO permanece em nível lógico 1 (s).

SUMÁRIO

1 INTRODUÇÃO

A grande motivação em desenvolver o Projeto de Fim de Curso (PFC), foi à criação de robôs, a partir de peças oriundas dos kits LEGO Mindstorms, disponíveis no LPR. Esses robôs têm por objetivo, tanto servir como um primeiro contato com a Robótica, por parte dos estudantes calouros, quanto aos alunos de pós-graduação, do curso de Engenharia de Controle e Automação da UFSC.

A Robótica, de um ponto vista geral, é uma área multidisciplinar, envolvendo áreas como: Ciência da Computação, Engenharia Elétrica, Engenharia Mecânica, Biologia, Psicologia, Filosofia e Sociologia (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011). É uma área que, apesar de bastante consolidada (principalmente no chão de fábrica e sob uma crescente demanda, tanto em relação aos robôs móveis quanto manipuladores, com seu espectro indo além da indústria), possui um vasto campo de pesquisa. Diante de novos problemas pertinentes à Robótica, muitos estudos têm surgido a fim de solucioná-los, tais como: flexibilidade e autonomia (SATO; SANTOS; ALMEIDA, 2015).

Com base nessa multidisciplinaridade, é uma ferramenta poderosa inclusive no campo educacional, onde os estudantes podem fixar conceitos a partir de experiências práticas. O conhecimento é construído a partir da experiência do estudante, mediante a realização de uma dada atividade (no caso, montagem e programação de robôs LEGO). Tal ideia é denominada “Construtivismo”, desenvolvida por Jean Piaget[6] (TAVARES; AGOSTINI; RIGO, 2018).

A Robótica, como ferramenta de aprendizagem, é capaz de ampliar as atividades que podem ser desenvolvidas e fazer com que diferentes áreas do conhecimento sejam integradas entre si. A Robótica tem um elevado potencial, condensando os mais diversos saberes de disciplinas, distintas entre si, permitindo que os estudantes possuam experiência, na prática, do método científico, por meio do desenvolvimento de protótipos (PIO; CASTRO; CASTRO JÚNIOR, 2006).

Os robôs móveis possuem capacidade de locomoção, diferentemente dos robôs manipuladores. (ALBUQUERQUE; ÁLVARES, 2011). No entanto, robôs

[6] Epistemólogo suíço. Dedicou-se a investigar os processos e etapas envolvidos à construção do conhecimento humano, a partir da interação sujeito/objeto (ABREU et. al., 2010).

móveis possuem complexidade mais elevada no âmbito de percepção, localização e navegação. (SIEGWART; NOURBAKHSH; SCARAMUZZA, 2011).

Os robôs foram projetados para a pista disponível no LPR, ou seja, submetidos a um ambiente determinístico, estático e parcialmente observável. Um ambiente típico do mundo real é probabilístico (estocástico), muitas vezes dinâmico e também parcialmente observável (RUSSELL; NORVIG, 2004). Mesmo que haja essa discrepância entre a pista do LPR e o mundo físico real, podem-se criar estratégias de cooperação a partir de um ambiente mais simplificado e em escala reduzida. Pois muitos problemas podem ser solucionados considerando-se o ambiente estático e estocástico, quer dizer, os efeitos probabilísticos e dinâmicos nestes ambientes são pouco relevantes à solução do problema.

Outra grande motivação para o desenvolvimento dos protótipos (robôs móveis) está relacionada à importância, cada vez maior para a indústria. Do ponto de vista histórico, nos anos de 1960, os robôs móveis já eram utilizados no chão de fábrica, no entanto, sua utilização era bastante restrita em relação aos robôs manipuladores, devido a fatores, como por exemplo, o custo de aquisição, manutenção, hardware e software com desempenho insatisfatório. Tal situação sofreu alterações nos anos posteriores, fazendo com que um robô móvel se tornasse uma ferramenta de trabalho sofisticada nas operações de manufatura, devido ao aumento do poder computacional, barateamento do hardware (CPU, sensores, atuadores). A manufatura industrial flexível requer adaptação a mudanças constantes nos requisitos de produção e redução de custos. A Robótica é uma área crucial para tornar qualquer indústria competitiva, por meio da redução de custos, redução dos tempos de manufatura e entrega, aumento da qualidade do produto final e otimização dos recursos (ALBUQUERQUE; ÁLVARES, 2011).

Além dos protótipos desenvolvidos, foram elaboradas estratégias de cooperação. A Robótica Cooperativa é uma área pertinente aos denominados “Sistemas Multi-Robôs”, cujos estudos iniciaram-se no final dos anos 80 (PARKER, 2003). A Robótica Cooperativa (bem como “Sistemas Multi-Robôs”) é uma área com grande potencial para pesquisa, bem como aplicações práticas, indo além da indústria.

Quanto à cooperação entre os robôs desenvolvidos, a comunicação é fator crucial, uma vez que a capacidade sensora dos mesmos é expandida através de

troca de informação com outros robôs. A comunicação pode ser tanto direta quanto indireta. No caso de comunicação direta, a mesma ocorre por troca de mensagens. Na comunicação indireta, um sinal é difundido no canal de comunicação (FERBER, 1999). Para os robôs desenvolvidos, a comunicação é do tipo direta.

1.1 Justificativa

O que levou ao desenvolvimento de estratégias de cooperação dos robôs desenvolvidos é o fato de que muitos problemas não podem ser resolvidos por um único robô, pelas seguintes limitações:

- Recursos limitados ou inexistentes;
- Pouca Habilidade;
- Conhecimento limitado;
- Necessidade de robustez;
- Alta complexidade da tarefa para ser realizada por um único robô;
- O custo de somente um robô (de grande dimensão, alta complexidade e hardware de alto desempenho) pode ser proibitivo (FERBER, 1999).

Quanto à atividade deste PFC (Projeto de Fim de Curso), foram desenvolvidos oito protótipos no LMM, para uso no LPR, utilizando-se como estrutura mecânica, peças de LEGO oriundas dos kits LEGO Mindstorms, porém com estrutura eletrônica (circuitos e sensores) alternativa às originais LEGO.

Os robôs desenvolvidos no LMM são do tipo móvel, ou seja, não possuem estruturas de manipulação de objetos, como por exemplo, garras ou braços manipuladores. Por outro lado, o espaço de trabalho de um robô manipulador é fixo, enquanto o espaço de trabalho de um robô móvel (de acordo com o ponto de vista teórico) pode ser todo o plano (x,y) ou o espaço (x,y,z). O espaço de trabalho de um robô móvel, não necessariamente será bem definido. Essa situação torna-se evidente em situações de exploração, em que o robô móvel deve percorrer caminhos sem possuir uma informação prévia do mesmo. Quanto à finitude do espaço de trabalho de um robô móvel, na prática, o fator limitante é o próprio ambiente em que o mesmo está inserido e por sua vez, pode mover-se (a presença

de obstáculos restringiria ainda mais o espaço de trabalho, inclusive a própria manobrabilidade do robô).

Os componentes utilizados são de fácil acesso e baixo custo, cujo projeto é simples de ser replicado e até mesmo aprimorado. Basicamente, possuindo-se um conhecimento básico em Eletrônica, projeto e montagem de PCB's (placa de circuito impresso), será possível replicá-los sem grandes dificuldades.

Apesar de terem sido utilizadas peças de LEGO na estrutura mecânica dos robôs, o projetista é livre para utilizar outros componentes mecânicos em substituição ao LEGO.

Os conceitos, bem como o desenvolvimento dos protótipos, são pertinentes às seguintes disciplinas do curso de Engenharia de Controle e Automação (na graduação, são assuntos pertinentes da 1ª à 6º fase).

- Introdução à Engenharia de Controle e Automação;
- Arquitetura e Programação de Sistemas Microcontrolados;
- Circuitos Elétricos;
- Eletrônica Básica;
- Instrumentação em Controle;
- Redes de Computadores;
- Modelagem e Controle de Sistemas a Eventos Discretos;
- Sistemas Distribuídos;
- Robótica Móvel (a nível de mestrado).

Quanto aos objetivos, são eles:

1.1.1 Objetivos Gerais

- Desenvolver protótipos alternativos aos kits LEGO;
- A partir dos protótipos desenvolvidos, desenvolver estratégias de cooperação entre os mesmos;
- Permitir aos estudantes iniciantes do curso, um primeiro contato com a Robótica, e aos pós-graduandos, uma plataforma que lhes possibilite pôr prática conceitos e modelos teóricos;

- Protótipos de baixo custo.

1.1.2 Objetivos Específicos

- Criação de protótipos com pequena curva de aprendizado;
- Protótipos com arquitetura modular e escalável;
- Protótipos com elevado grau de autonomia;
- Interface de uso amigável.

1.2 Visão Geral do Trabalho Desenvolvido

Quanto à apresentação da monografia, no capítulo 2 serão abordados o funcionamento dos robôs LEGO, seus componentes, características e limitações.

No capítulo 3, são dadas as descrições dos componentes utilizados nos protótipos, bem como suas justificativas.

O capítulo 4 aborda o projeto dos protótipos.

A cooperação será discutida no capítulo 5.

Por fim, os testes com os protótipos e a considerações finais, nos capítulos 6 e 7, respectivamente.

2 CARACTERÍSTICAS E ARQUITETURA DOS ROBÔS LEGO

2.1 Introdução

No presente capítulo serão apresentados a arquitetura dos kit's LEGO, seus componentes e funcionalidades dos robôs LEGO. Por fim, uma visão geral sobre a arquitetura dos mesmos. Nas figuras 1 e 2, encontram-se ilustrados os robôs utilizados no LPR.

Figura 1 – Vista lateral direita dos robôs LEGO utilizados no LPR.

Fonte: Autor (2018).

Figura 2 – Vista frontal dos robôs LEGO utilizados no LPR.

Fonte: Autor (2018).

2.2 LEGO Mindstorms

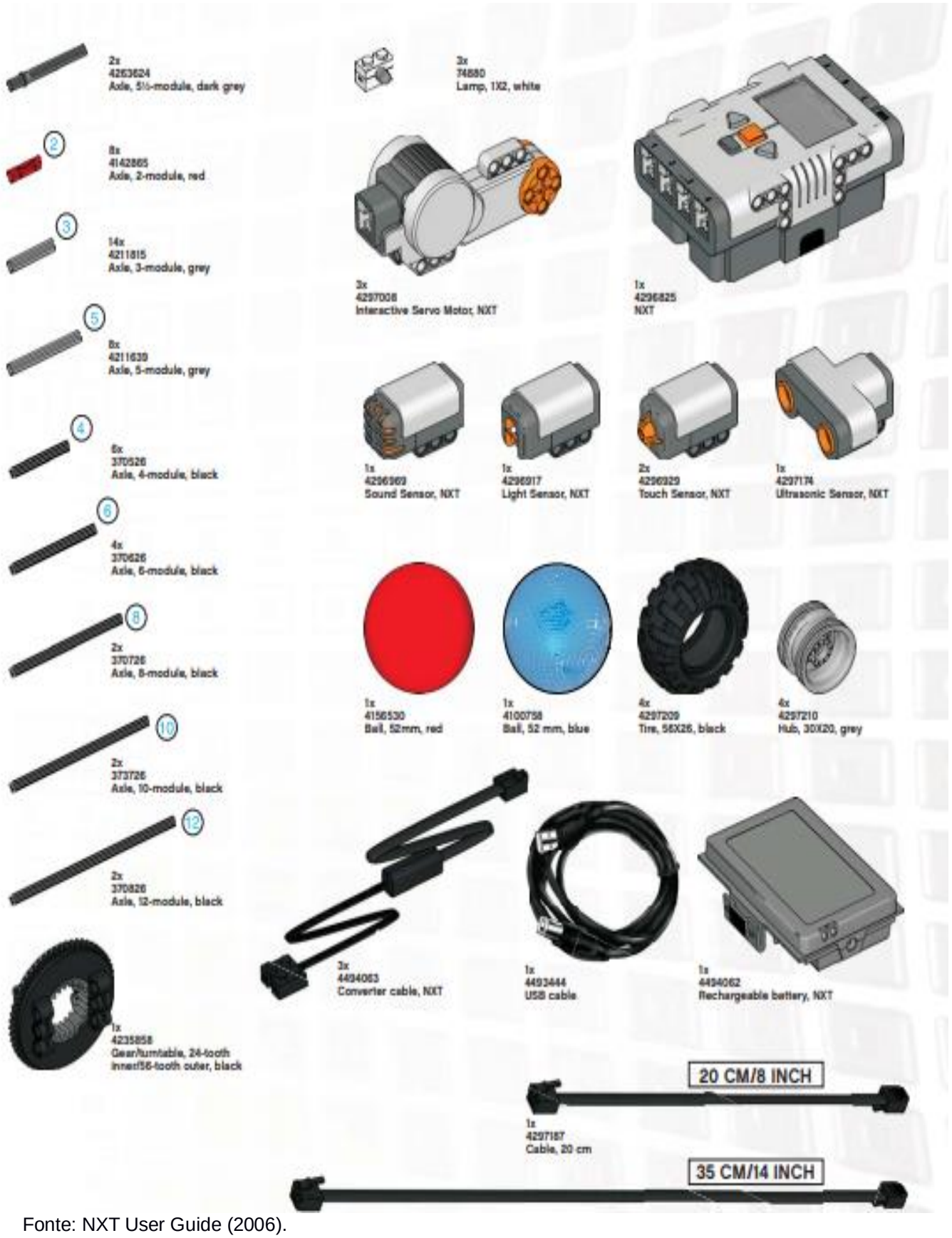
O LEGO “*Mindstorms Education*”, é a segunda geração da LEGO em Robótica educacional. Possuindo como CPU, o NXT “Brick”, de 32-bits, peças, sensores e motores.

O Kit é composto pelos seguintes componentes (além dos blocos de montar):

- 3 motores;
- 2 sensores táteis (botões);
- 1 sensor ultrassônico;
- 1 sensor óptico;
- 1 microfone (sensor sonoro);
- 1 bateria (com carregador externo);
- 1 NXT “Brick” (CPU);
- Cabos de conexão (sensores e motores);
- 2 esferas (uma azul e uma vermelha);
- 4 rodas com pneu.

A figura 3 ilustra as peças que acompanham o kit LEGO Mindstorms.

Figura 3 – Lista de peças do kit LEGO Mindstorms.



Fonte: NXT User Guide (2006).

Os robôs utilizados no LPR são oriundos do LEGO NXT, possuindo como componentes:

- 2 sensores ópticos (frontais);
- 1 sensor ultrassônico (frontal);
- 2 motores (com odometria);
- 1 bateria (com carregador externo);
- NXT Brick (CPU).

2.3 Componentes

Serão abordados os principais componentes do LEGO Mindstorms utilizados no LPR (para os robôs), tais como sensores e atuadores (motor).

2.3.1 NXT “Brick”

É a CPU utilizada nos robôs. Possui internamente um microcontrolador AtmelARM7TDMI-core, de 32-bits, um microcontrolador AT91SAM7S256, de 256 Kbytes de memória do tipo FLASH (EEPROM) e 64 Kbytes de RAM e um microcontrolador de 8-bits Atmel AVR ATmega48. O NXT “Brick” possui um display LCD monocromático de 100x60 pixels e suporte a Bluetooth.

A alimentação do NXT “Brick” pode ser realizada tanto por bateria (acompanha o kit), quanto por pilhas AA (6 pilhas de 1,5 V cada, ou 6 pilhas recarregáveis de 1,2 V cada).

O mesmo também é dotado de alto-falante e ADC (conversor analógico-digital) e DAC (conversor digital-analógico, para o controle de velocidade dos motores).

Quanto à conexão com sensores e atuadores, são realizadas por meio de fios, com conexão baseada no padrão RJ12 (levemente modificado pela posição onde se situa a trava). O NXT “Brick” é ilustrado nas figuras 4 e 5.

Figura 4 - LEGO NXT “Brick”, com a vista das portas dos sensores (entradas). Da esquerda para direita, as portas são enumeradas de 1 a 4.

Fonte: Autor (2018).

Figura 5 – Vista das portas de saída, para os atuadores (motores e LED’s que acompanham os kits LEGO). Há 3 portas de saída, cada uma nomeada de A à C (na foto, da direita para esquerda). Ao canto esquerdo, pode-se ver a conexão USB (semelhante à utilizada por impressoras).

Fonte: Autor (2018).

2.3.2 Sensor Óptico

Cada Kit LEGO Mindstorms acompanha somente 1 sensor óptico (figura 6), o qual são compostos por 1 LED de luz monocromática (vermelho) e um foto receptor.

Utilizados para realizar identificação entre superfícies claras e escuras, por meio do grau de reflexibilidade da superfície (luz refletida pela mesma).

Figura 6 – Sensor óptico.

Fonte: Autor (2018).

2.3.3 Sensor Ultrassônico

O sensor ultrassônico utilizado (figura 7) é composto por um emissor e receptor. Utilizado para medição de distância com base no tempo de ida e volta do sinal emitido, captado pelo receptor.

Figura 7 – Sensor ultrassônico.

Fonte: Autor (2018).

2.3.4 Motor LEGO NXT

Os motores LEGO NXT (figura 8) possuem tensão nominal de alimentação (máxima) de 9 V, redução interna e encoders (odometria).

Figura 8 – Motor LEGO NXT.

Fonte: Autor (2018).

2.3.5 Bateria

Junto ao kit LEGO Mindstorms, uma bateria acompanha o mesmo (figura 9). Possui 2 células de lítio, em série, de 3,7 V cada, totalizando 7,4 V e uma carga de 1400 mAh. Além da bateria, há um carregador externo, bivolt (110 V/220 V), com tensão de saída de 12 Volts.

Figura 9 – Bateria dos robôs LEGO.

Fonte: Autor (2018).

2.4 Funcionalidades

Quanto aos robôs LEGO, estes têm como capacidade:

- Seguimento de linha;

- Desvio/parada, em caso de obstáculos;

- Capacidade de estimar posição (odometria), mediante encoders;

- Controle de velocidade de motores;

- Comunicação com outros robôs ou PC (BLUETOOTH);

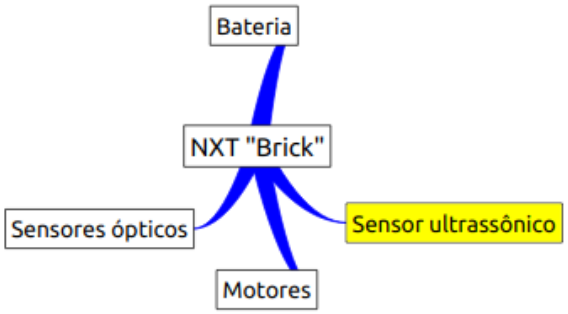
- Comunicação com o PC, por meio de cabo USB;

- Identificação de cores de superfícies (de acordo com a tonalidade/reflexibilidade).

2.5 Arquitetura dos Robôs LEGO

Visto a forma como os sensores foram montados, bem como os motores, ao NXT “Brick”, a arquitetura dos robôs foi definida sob a seguinte estrutura (figura 10).

Figura 10 – Arquitetura dos robôs LEGO

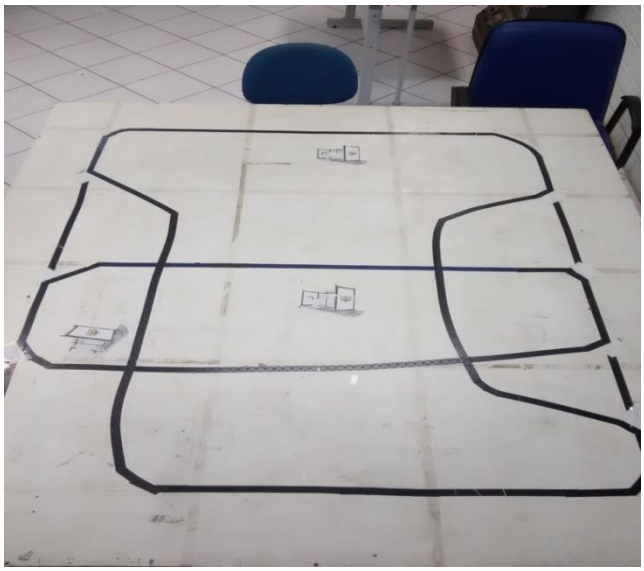


Fonte: Autor (2018).

2.6 Espaço de Trabalho dos Robôs LEGO (Workspace)

Os robôs LEGO operam na pista do LPR (seu ambiente de trabalho). As dimensões da pista são: 1.83 m X 1.55 m. A pista de teste dos robôs LEGO NXT é mostrada na figura 11.

Figura 11 – Pista dos robôs, no LPR.



Fonte: Autor (2018).

2.7 LIMITAÇÕES DOS ROBÔS LEGO

Como limitações em relação aos robôs LEGO utilizados, são as principais:

- Poucas portas de E/S, somente 4 portas de entrada (sensores) e 3 de saída (atuadores);
- Apesar da modularidade do LEGO, as conexões para sensores e atuadores são incompatíveis com componentes eletrônicos típicos (porém, estes últimos são quase sempre compatíveis com protoboards, Arduino, PIC, etc);
- Dificuldade de reposição de peças (principalmente o NXT “Brick”, motores e sensores);
- Alto custo de aquisição (tanto ao kit, quanto componentes avulsos);

- A comunicação entre os NXT “Brick” permite uma rede de no máximo 3 estações conectadas (denominada “grupo”). Outra limitação, os NXT não trocam mensagens entre si, somente arquivo de programas. Tal questão é um problema no quesito cooperação entre robôs. Uma estratégia de cooperação entre robôs LEGO foi desenvolvida, denominada “Orientação Cooperativa”, mas sem comunicação direta (mensagens), somente baseada em interação (MARZAT; PIET-LAHANIER; KAHN, 2014).

2.8 Solução Alternativa aos Robôs LEGO

Foi proposta uma alternativa, com objetivo de preservar as funcionalidades originais do LEGO, mas com uma funcionalidade adicional, permitir a cooperação entre os robôs (via troca de mensagens). No entanto, utilizando-se para isso, componentes de baixo custo, facilmente disponíveis no mercado.

Quanto ao projeto dos robôs, foi assim definido:

- Estrutura mecânica (feita com peças LEGO, já disponíveis no LPR), onde os sensores e todas as placas são instaladas;

- Robôs dotados de 2 sensores ópticos frontais, para a detecção de superfícies claras e escuras, mediante o grau de reflexibilidade da mesma;

- Utilização de 1 sensor ultrassônico frontal, para o desvio ou até mesmo parada dos robôs, em caso de obstáculos;

- Uma CPU, utilizando-se para isso a plataforma Arduino Nano;

- Bateria, construída a partir de 2 células de lítio, provenientes de bateria de notebooks de sucata;

- Utilização de placa de monitoramento de bateria, a partir dessa, alimenta a CPU, esta última alimenta todos os demais componentes do robô;

- Robôs dotados de 2 motores LEGO NXT, controlados de forma independente;

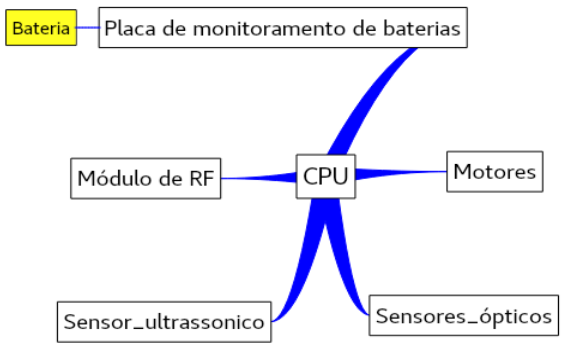
- Para a comunicação, foi utilizado um módulo de RF para Arduino, composto por um transmissor e um receptor, instalados na CPU do robô;

- Um carregador de bateria, do tipo tensão constante, externo aos robôs.

2.9 – Visão Arquitetural dos Protótipos

Quanto à estrutura de montagem (mecânica e elétrica) desenvolvida, sobre o chassi encontram-se todas as placas e sensores dos robôs. A bateria encontra-se abaixo da CPU, esta alimenta a placa de monitoramento, a placa de monitoramento alimenta a CPU, esta última, por sua vez, alimenta os sensores, o módulo de RF (transmissor e receptor) e realiza o acionamento dos motores. A figura 12 ilustra a arquitetura dos protótipos desenvolvidos.

Figura 12 – Arquitetura dos robôs desenvolvidos.



Fonte: Autor (2018).

3 CONCEITOS E COMPONENTES UTILIZADOS NOS PROTÓTIPOS

Neste capítulo será apresentada uma abordagem conceitual sobre os robôs desenvolvidos, bem como uma descrição de funcionamento e justificativa dos componentes escolhidos para a solução do problema anteriormente apresentado.

3.1 – Robôs Desenvolvidos

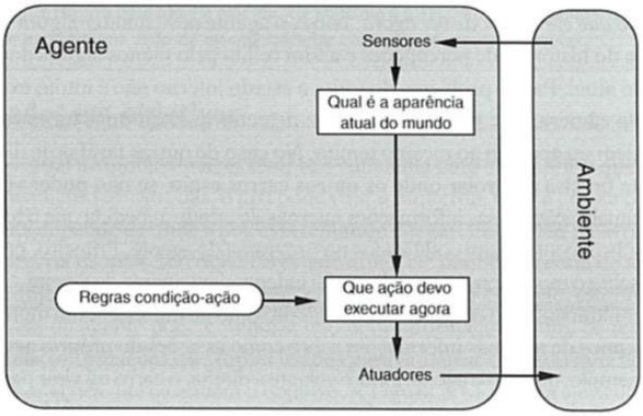
Dado a arquitetura dos robôs desenvolvidos, bem como a posição dos sensores, eles podem ser considerados veículos de Braintenberg (quando utilizados para seguimento de linha, por exemplo), possuindo tanto comportamento explorador, quanto comportamento temeroso (ALCOBA; CALLEJAS; PAZ, 2016), vistos nas figuras 13 e 14, respectivamente. O tipo de comportamento adquirido é implementado em software. Para a aplicação de seguimento de linha, no LPR, eles foram configurados como exploradores.

Figura 13 – Arquitetura de veículos de Braintenberg (à esquerda, temeroso e à direita, explorador)

Fonte: ALCOBA; CALLEJAS; PAZ. p. 161. (2016).

Os robôs desenvolvidos, quando utilizados isoladamente, possuem comportamento reativo (sua arquitetura pode ser visualizada na figura 14), ou seja, operam sobre a percepção atual (não armazenam percepções passadas), ou seja, desprovidos de memória de trabalho. Baseiam-se na premissa condição-ação (se-então) (RUSSELL; NORVIG, 2004). A figura 15 apresenta o algoritmo de um agente reativo.

Figura 14 – Arquitetura de um agente reativo simples.



Fonte: RUSSELL, NORVIG (2004).

Figura 15 – Algoritmo de um agente reativo simples.

```
função AGENTE-REATIVO-SIMPLES(percepção) retorna uma ação
    variáveis estáticas: regra, um conjunto de regras condição-ação
        estado ← INTERPRETAR-ENTRADA (percepção)
        regra ← REGRA-CORRESPONDENTE (estado, regras)
        ação ← AÇÃO-DA-REGRA[regra]
        retornar ação
```

Fonte: RUSSELL, NORVIG (2004).

Quanto ao grau de mobilidade dos robôs, os robôs desenvolvidos são considerados não-holonômicos, ou seja, o grau de mobilidade dos robôs desenvolvidos é menor que a dimensionalidade do espaço de trabalho (dimensão do espaço de trabalho = 3, enquanto que o grau de mobilidade dos robôs = 2) (SIEGWART; NOUBARKHSH; SCARAMUZZA, 2011).

3.2 – Componentes Constituintes dos Robôs Desenvolvidos (Arduino)

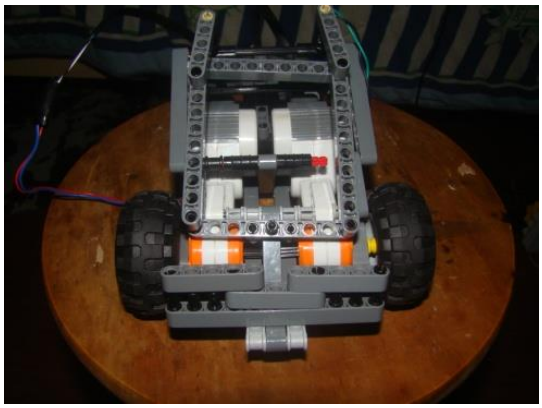
Nesta seção, serão abordados todos os componente utilizados nos protótipos.

3.2.1 – Chassi

O chassi (figuras 16 e 17) é composto por somente por peças LEGO e 2 motores LEGO NXT. Cada motor é acionado de forma independente. Os 2 motores são

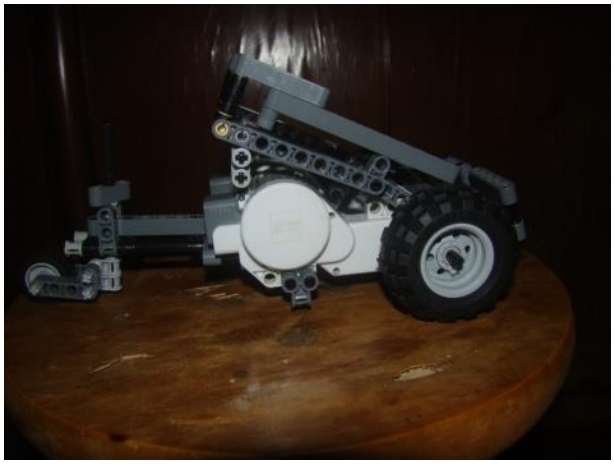
montados na parte frontal, havendo uma terceira roda na traseira do chassi, capaz de mover-se livremente (roda castor).

Figura 16 – Vista frontal do chassi.



Fonte: Autor (2018).

Figura 17 – Vista lateral direita do chassi.



Fonte: Autor (2018).

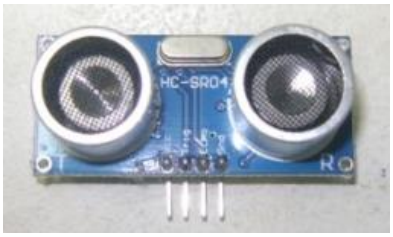
3.2.2 Sensores Ópticos

Existem sensores ópticos disponíveis no mercado, na forma de módulos, no entanto, preferiu-se desenvolver os módulos no próprio LMM. Os módulos utilizam um LED IR como emissor e um fotorreceptor (fotodiodos e fototransistores).

3.2.3 Sensor Ultrassônico

Para tal, foi utilizado o módulo HC-SR04 (figura 18). É um sensor simples de ser utilizado e compatível com a plataforma Arduino. Não requer componentes adicionais, a conexão entre o sensor e o Arduino é feita diretamente. Possui baixo custo e baixa potência de consumo.

Figura 18 – Sensor ultrassônico HC-SR04.



Fonte: Autor (2018).

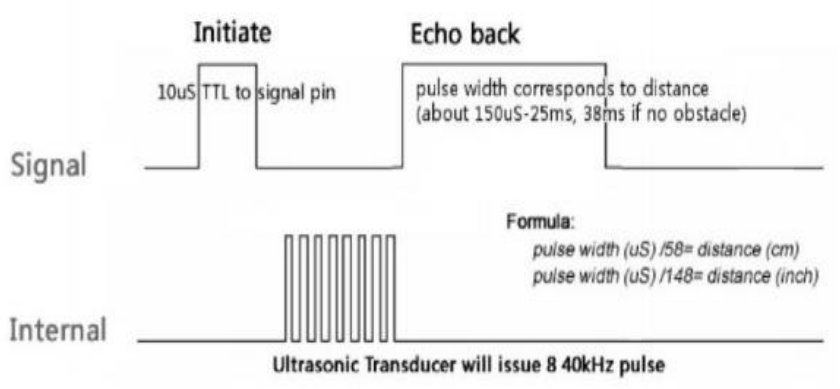
Seu funcionamento é simples. Existem 4 terminais no módulo (VCC, TRIGGER, ECHO, GND). Há um emissor (TRIGGER) e um receptor (ECHO) de sinal acústico. O emissor envia um trem de pulsos (8 pulsos, de tempo total 10 us), em que esses sinais são captados pelo receptor. A partir do momento em que o emissor envia o primeiro pulso, o terminal correspondente ao receptor (ECHO) vai a nível lógico 1, passando para nível lógico 0 quando o último (oitavo) pulso for captado pelo receptor.

A distância (1), está relacionada ao tempo (em segundos) em que o sinal ECHO permanece em nível lógico 1 e a velocidade do som no ar (340 m/s).

$$\text{Distância (m)} = (t_{echo} * v_{som}) / 2 \quad (1)$$

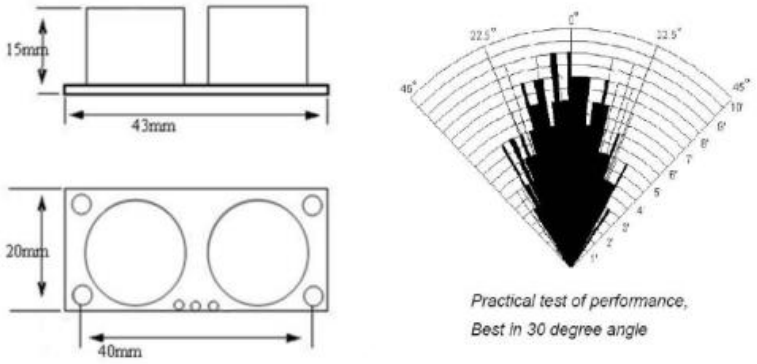
O funcionamento do sensor ultrassônico, bem como o ângulo de abertura do sensor, é mostrado nas figuras 19 e 20, respectivamente.

Figura 19 – Funcionamento do módulo ultrassônico.



Disponível em: <https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf> Acesso em: 15 nov. 2018.

Figura 20 – ângulo de abertura do sensor.



Disponível em: <https://www.mouser.com/ds/2/813/HCSR04-1022824.pdf> Acesso em: 15 nov. 2018.

Para utilizá-lo, basta configurar o pino do Arduino respectivos ao ECHO como entrada e o pino relacionado ao TRIGGER, como saída.

3.2.4 – CPU

A plataforma Arduino Nano (figura 21) foi utilizada, devido ao custo razoavelmente baixo, por possuir pinos compatíveis à montagem em PCB e pela grande quantidade de pinos de E/S (acionamento dos motores, leitura dos sensores e envio e recebimento de sinais ao módulo de RF) comparados à quantidade de entradas e saída do LEGO NXT (4 entradas para sensores, 3 saídas para motores).

O Arduino é open-source e o software Arduino IDE, onde é feita sua programação, é compatível com Windows, LINUX e MAC.

Dentre algumas das características do Arduino Nano (v 3.0):

- Utiliza como microcontrolador, o Atmega328, de 8 bits;

- 32 Kbytes de memória de programa (2 Kbytes para o *bootloader*);

- 14 pinos de E/S digitais;

- 8 pinos de entrada analógica;

- 2 Kbytes de memória RAM (estática);

- 1 Kbyte de EEPROM;

- Frequência de operação de 16 MHz;

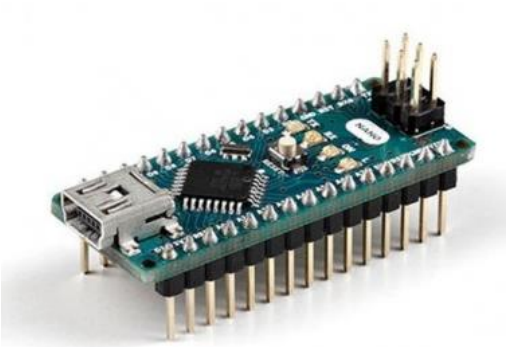
- ADC de 10 bits;

- Suporte à comunicação com o PC, via terminal, através de comunicação serial;

-Programação em linguagem C/C++ (Arduino IDE).

O Arduino Nano pode ser alimentado tanto através da porta USB, como por alimentação externa (bateria, no caso dos protótipos desenvolvidos).

Figura 21 – Arduino Nano v 3.0



Disponível em: <http://roboromania.ro/datasheet/Arduino-Nano-roboromania.pdf> Acesso em: 15 nov. 2018.

3.2.5 - Bateria

A bateria foi desenvolvida a partir de 2 células de lítio, oriundas de baterias de notebook de sucata, disponíveis no LPR. As baterias não foram desenvolvidas com células idênticas. Ao todo, 3 modelos de células foram utilizadas, são elas: CGR17670A (figura 22), INR18650 (figura 23) E UR18650A (figura 24).

Figura 22 – Célula de lítio CGR17670A

Fonte: Autor (2018).

Figura 23 – Célula de lítio INR18650.

Fonte: Autor (2018).

Figura 24 – Células de lítio UR18650A.

Fonte: Autor (2018).

As 3 células de lítio possuem tensão típica de operação entre 3,7 V (podem ocorrer variações, por exemplo, até 4 V). A primeira célula (CGR17670A) possui carga de 1500 mAh, já a célula de lítio INR18650 e UR18650A possuem carga nominal típica de 2500 mAh e 2250 mAh respectivamente.

As células de lítio não podem operar abaixo de um limiar, para as 3 células utilizadas nos robôs, seu valor é de 2,5 V, abaixo desse valor, as células são danificadas. Outra questão importante é a temperatura. Não devem ficar acima de 60 ºC, sob risco de sofrerem rápida descarga e redução de vida útil.

3.2.6 – Monitor de Bateria

Foi desenvolvido um circuito para os robôs, para o monitoramento da bateria e com sistema de proteção, no caso de superaquecimento das células de lítio, ou caso as células descarreguem até o limiar de 3 V. Caso essas 2 situações ocorram, A alimentação da CPU do robô, bem como os demais componentes alimentados à ela, é interrompida. A bateria alimenta o circuito de monitoramento, ao mesmo tempo em que é monitorada e a partir do circuito de monitoramento, o mesmo alimenta a CPU do robô.

Primeiramente, cogitou-se a ideia de utilizar o monitoramento de temperatura e tensão no próprio Arduino e fazer a leitura dos valores de tensão diretamente através dos pinos de entrada analógica. No entanto, os recursos do Arduino Nano são escassos, e, inclusive para reduzir o *overhead* do processador, um circuito de monitoramento dedicado às células foi projetado.

Para o monitoramento das células de lítio, foi utilizado um microcontrolador PIC16F716 (fabricado pela Microchip), de baixo custo, com as seguintes características:

- Microcontrolador de 8 bits;

- ADC de 8 bits (4 pinos/canais multiplexados);

- 2048 KWords de memória de programa;

- 128 Kbytes de EEPROM.

A frequência de operação do microcontrolador PIC16F716 (figura 25) é dependente do cristal oscilador conectado a ele (gerador de clock externo). Para o circuito em questão, foi utilizado um cristal de 4 MHz, fazendo com que a frequência interna do mesmo seja de 1 MHz (razão frequência_externa/frequência_interna de 4:1).

Quanto ao dispositivo de proteção, como ideia inicial, cogitou-se utilizar um relê, no entanto, por se tratar de uma chave eletromecânica, esta por sua vez produz ruído cada vez que ocorre comutação (liga-desliga), tanto pelo interruptor do relê quanto pelo acionamento da bobina (carga indutiva). A ideia do relê foi substituída por um MOSFET, por ser dispositivo semicondutor, o problema de ruído é consideravelmente atenuado.

Figura 25 - Microcontrolador PIC16F716

Disponível em: https://www.microchip.com/wwwproducts/en/PIC16F716. Acesso em: 16 out. 2018.

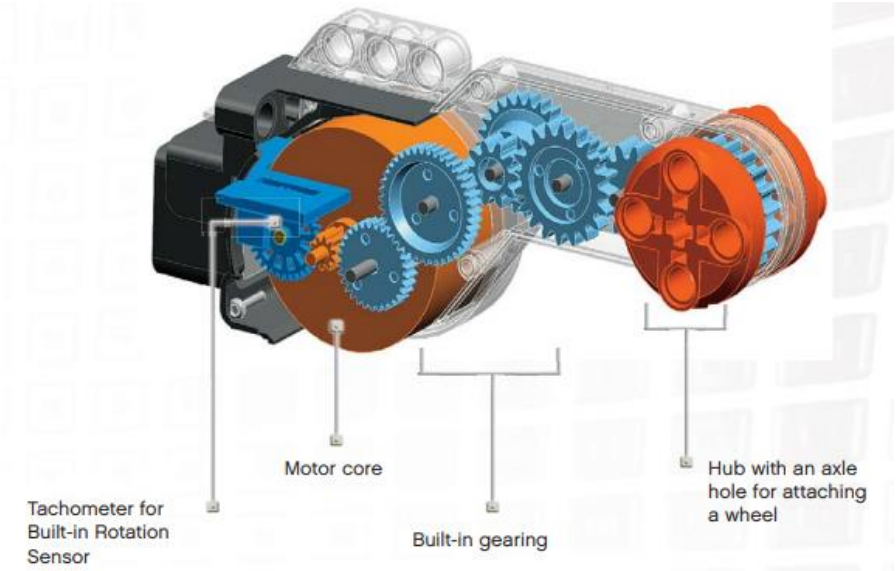
3.2.7 – Motores LEGO NXT

Para os protótipos desenvolvidos, foram utilizados os motores LEGO NXT (figura 26), dos próprios kits LEGO Mindstorms.

Os motores LEGO possuem internamente um encoder (com resolução de 1° por passo do motor) e redução, tornando desnecessária a utilização de engrenagens externas ao motor, durante a montagem dos protótipos.

Os motores operam com tensão elétrica típica de 9 V, para os protótipos originais LEGO. Nos protótipos desenvolvidos, a tensão elétrica típica é de 7,2 V a 8 V, por conta da tensão das células de lítio.

Figura 26 - Estrutura interna motor LEGO NXT



Fonte: NXT User Guide (2006).

3.2.8 – Módulos de RF

Para comunicação e cooperação dos robôs, foram utilizados módulos de RF, compostos por um transmissor e um receptor (figura 27).

Os módulos de RF operam sob modulação ASK (amplitude shift keying, ou modulação em amplitude por chaveamento), uma técnica de modulação de sinais digitais. No caso desses módulos, o sinal digital (modulante) somente admite 2 níveis de tensão (nível de tensão alto – 1, nível de tensão baixo - 0).

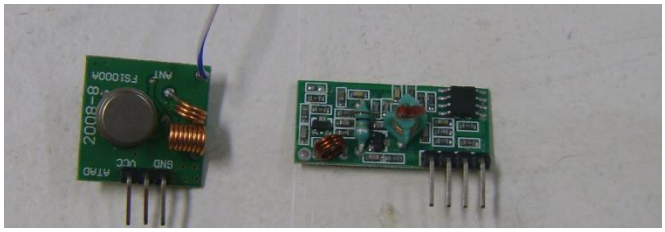
Foi instalada ao transmissor (de cada um dos módulos), uma antena, a fim de elevar o alcance de transmissão. As antenas são simples, feitas a partir de fios comuns (cabo de rede Ethernet), de ¼ de onda.

Para obter o comprimento de onda do sinal (2), ou seja, faz-se a divisão entre a velocidade de uma onda eletromagnética ($3*10^8$ m/s) pela freqüência da mesma (portadora), para os módulos de RF empregados, o valor da frequência foi de 433 MHz ($4,33*10^8$ m/s).

$$\lambda = v/f \quad (2)$$

O valor λ obtido foi 0,69 m, ou 69 cm. Como as antenas são do tipo ¼ de onda, o comprimento da antena deve ser igual a ¼ o comprimento de onda da antena, portanto, o comprimento da antena obtido, foi igual a 17,3 cm.

Figura 27 – Módulo de RF (à esquerda, transmissor, à direita, receptor).



Fonte: Autor (2018).

3.2.9 – Carregador de Bateria

Quanto ao carregador de bateria, foram avaliadas as seguintes alternativas:

- Desenvolver um circuito dedicado à carga das células de notebook;
- Comprar um carregador de células para os protótipos;

Pelo fato de que cada bateria utiliza 2 células de lítio, um circuito de recarga de bateria típico requer 3 ou 4 células em série, não sendo possível utilizar um carregador para 2 células.

Com base nisso foi desenvolvido um carregador próprio para as baterias (há um carregador para cada bateria desenvolvida).

A recarga de bateria de lítio é complexa, comparada aos outros tipos de baterias (como por exemplo, chumbo). Para tal, seria necessária a utilização de um CI dedicado para recarga de células de lítio (um CI para 2 células). No entanto, este CI não é comercializado no Brasil. Como solução foi desenvolvido um circuito próprio, com componentes de fácil obtenção.

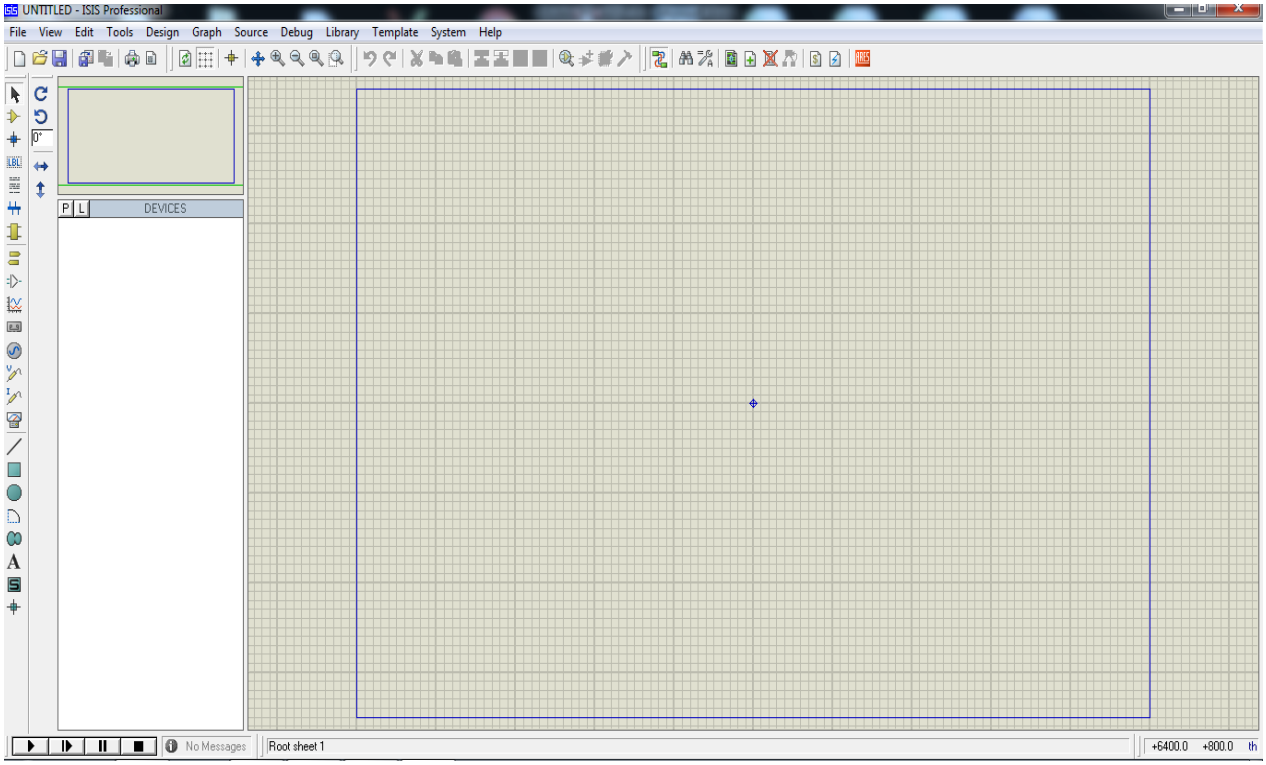
3.3 Softwares Empregados No Projeto dos Protótipos

Para o desenvolvimento dos circuitos dos protótipos, foi utilizado o software Proteus 7.8 (ARES/ISIS), produzido pela empresa Labcenter Electronics. ISIS (figura 28) é voltado ao projeto de diagramas de circuitos elétricos e eletrônicos, bem como sua simulação. ARES (figura 28), por sua vez, voltado ao projeto de PBC's.

Quanto ao desenvolvimento do programa do microcontrolador PIC16F716, existente na placa de monitoramento de bateria, foi utilizado o software MPLAB IDE (figura 30) fornecido pela empresa Microchip. MPLAB também suporta a simulação de microcontroladores, bem como o monitoramento de pinos de E/S e valores de variáveis.

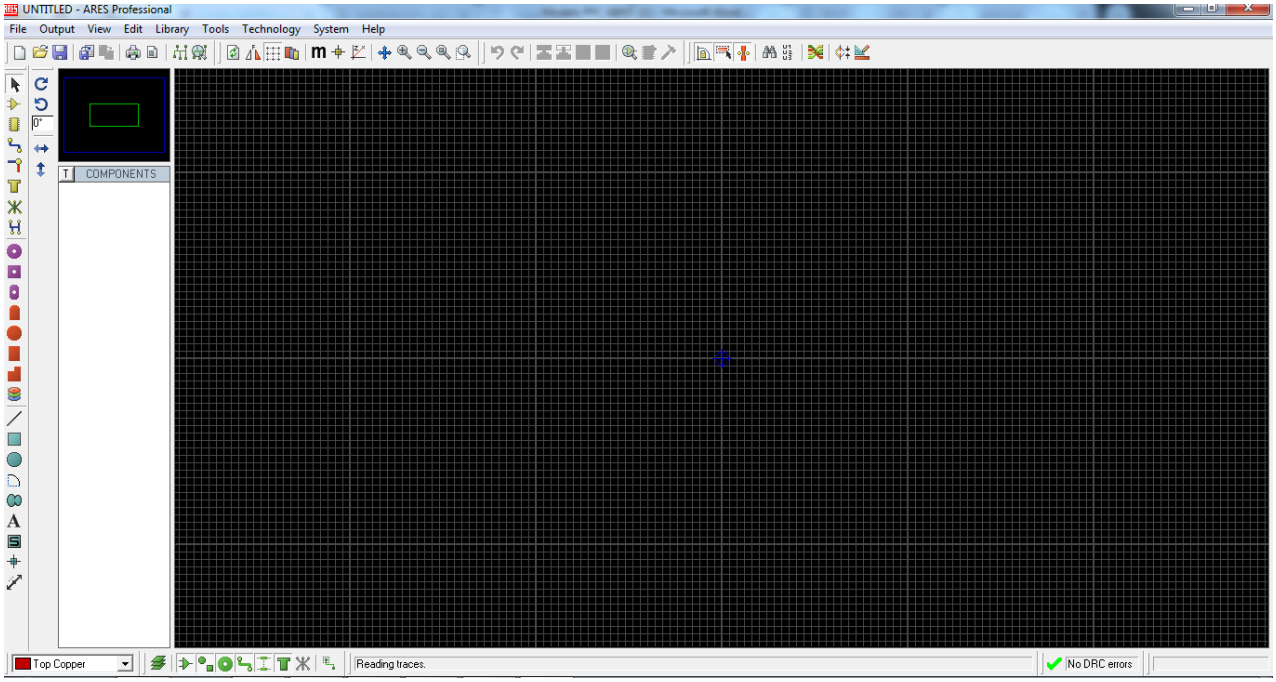
Para o projeto da bateria (case), foi utilizado o software SolidWorks 2010 (figura 31), fornecido pela empresa Dassault Systèmes.

Figura 28 – Interface do software Proteus (ISIS).



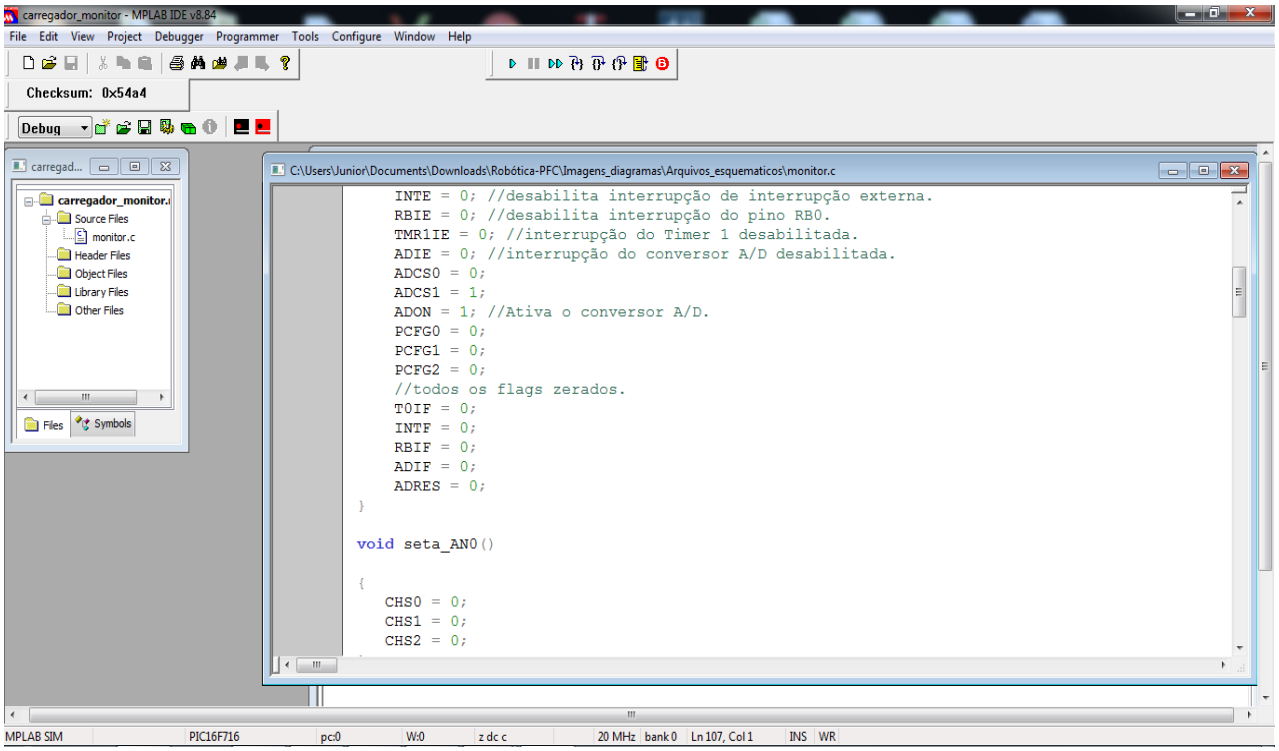
Fonte: Autor (2018).

Figura 29 – Interface do software Proteus (ARES).



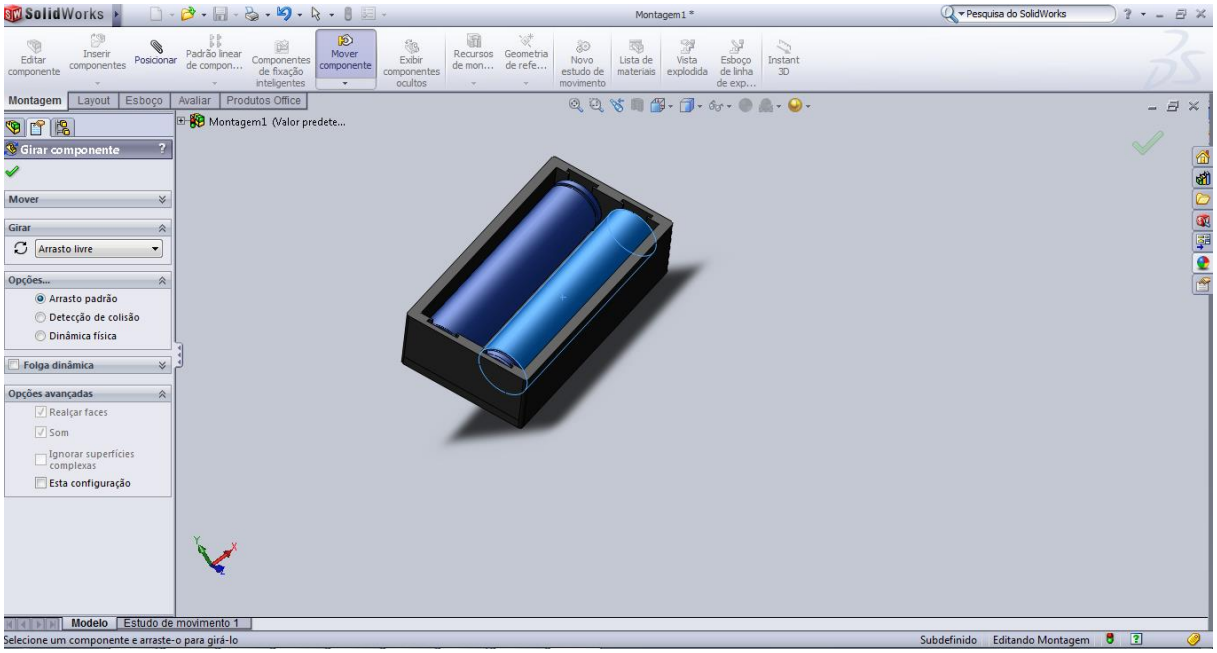
Fonte: Autor (2018).

Figura 30 – Interface do Software MPALB IDE



Fonte: Autor (2018).

Figura 31 – Interface do software SolidWorks 2010



Fonte: Autor (2018).

4 PROJETO E MONTAGEM DOS PROTÓTIPOS

Neste capítulo, serão abordados todos os projetos desenvolvidos dos protótipos, desde os diagramas de circuitos desenvolvidos, desenhos dos cases das baterias e cases dos sensores ópticos. Por fim, serão apresentados a montagem das PCB's, bem como a montagem final dos protótipos.

4.1 – Projeto dos Protótipos

4.1.1 Sensores Ópticos

Foram desenvolvidos, para cada um dos robôs, 2 sensores ópticos (representados nas figuras 32, 33 e 34 respectivamente) localizados à parte frontal dos mesmos. Possuem como objetivo a identificação de superfícies claras e escuras, em função de seu grau de reflexibilidade das mesmas.

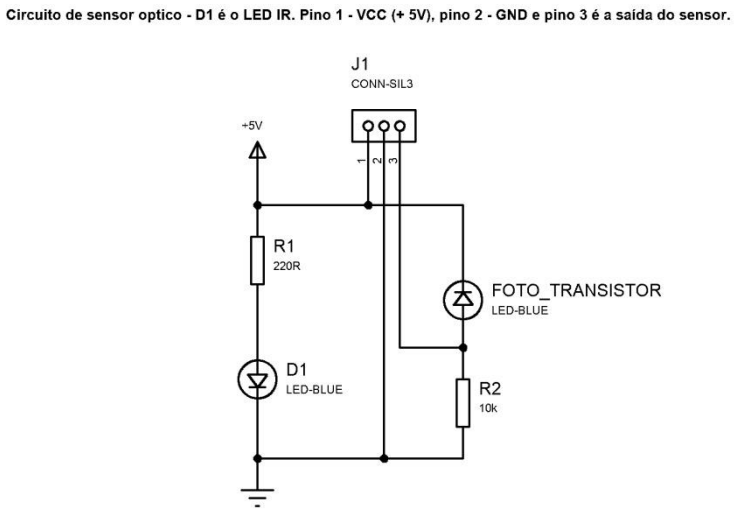
Cada sensor é composto por um LED IR (emissor) e um fotorreceptor.

A tensão de saída do sensor é diretamente proporcional à intensidade de luz refletida (emitida pelo emissor) captada no receptor.

No circuito abaixo, existe um conector de 3 pinos (J1), sua pinagem é:

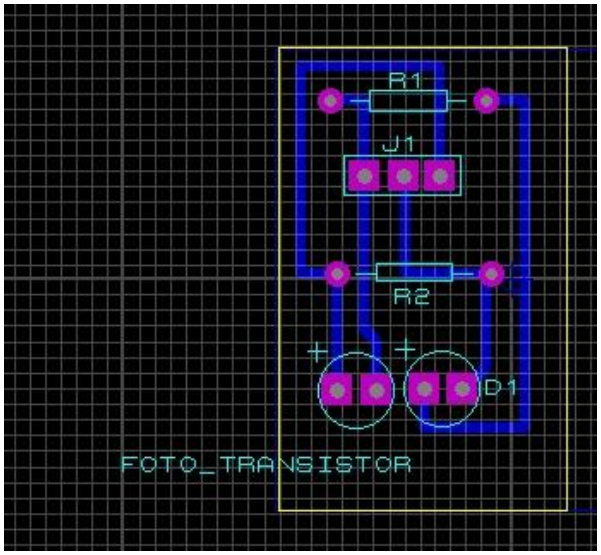
- Pino 1 -> +VCC (5 V);
- Pino 2 -> GND (0 V);
- Pino 3 -> Saída (sinal analógico, que posteriormente será lido pelo ADC do Arduino Nano).

Figura 32 – Diagrama de Circuito do sensor óptico (ISIS).



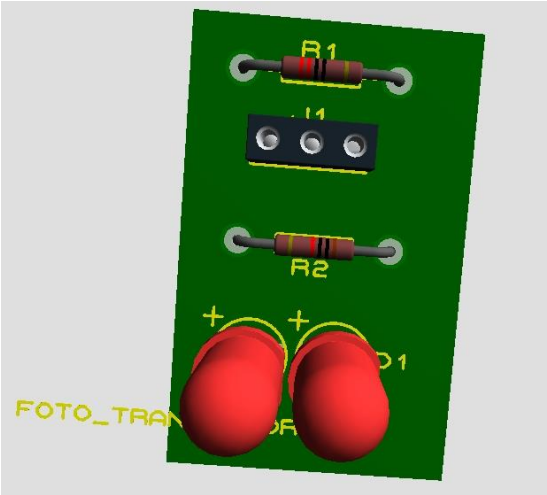
Fonte – Autor (2018).

Figura 33 – PCB do sensor óptico (ARES).



Fonte: Autor (2018).

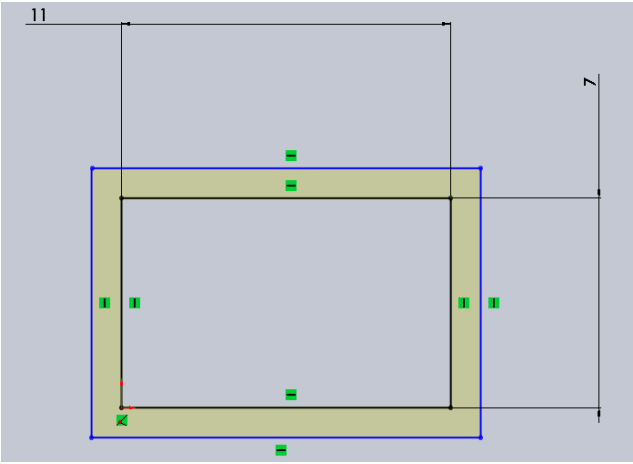
Figura 34 – Visão do sensor óptico renderizado (ARES).



Fonte: Autor (2018).

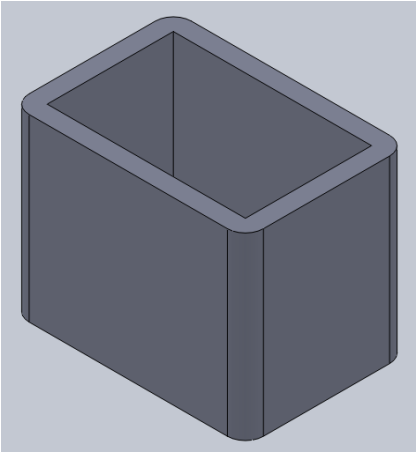
Para os sensores ópticos, também foram desenvolvidos um case (figuras 35 e 36) para cada um dos sensores, com a finalidade de atenuar a perturbação de medição do receptor IR devido à iluminação ambiente.

Figura 35 – Projeto do case do sensor óptico.



Fonte: Autor (2018).

Figura 36 – Geração do sólido 3D do case do sensor óptico (perspectiva isométrica).



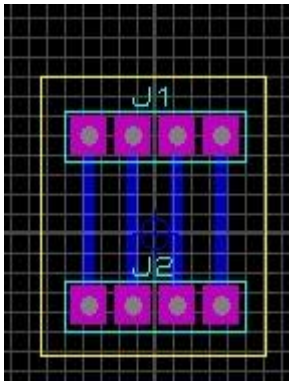
Fonte: Autor (2018).

4.1.2 – Sensor Utrassônico

Para o sensor ultrassônico, foi desenvolvida uma simples placa de suporte ao sensor. Foi projetado diretamente o circuito em PCB, por tratar-se de um circuito simples.

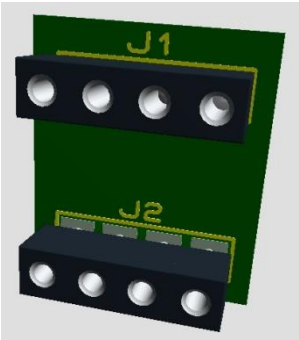
Sua função é servir de suporte, tanto para o sensor ultrassônico (conectores fêmea) quanto para a conexão de fios (conectores macho) que irão para a CPU. A figura 37 representa o circuito da PCB do suporte ao sensor e na figura 38, a PCB do suporte renderizada.

Figura 37 – PCB do suporte do sensor ultrassônico (ARES).



Fonte: Autor (2018).

Figura 38 – PCB do suporte renderizado do sensor ultrassônico (ARES).



Fonte: Autor (2018).

4.1.3 – CPU

A CPU dos robôs tem como funcionalidades:

- A leitura dos sensores ópticos, ultrassônico e dos encoders dos motores;
- Envio e recebimento de mensagens por meio dos módulos de RF;
- Acionamento dos motores LEGO.

A CPU possui como principais componentes:

- 1 plataforma Arduino Nano;

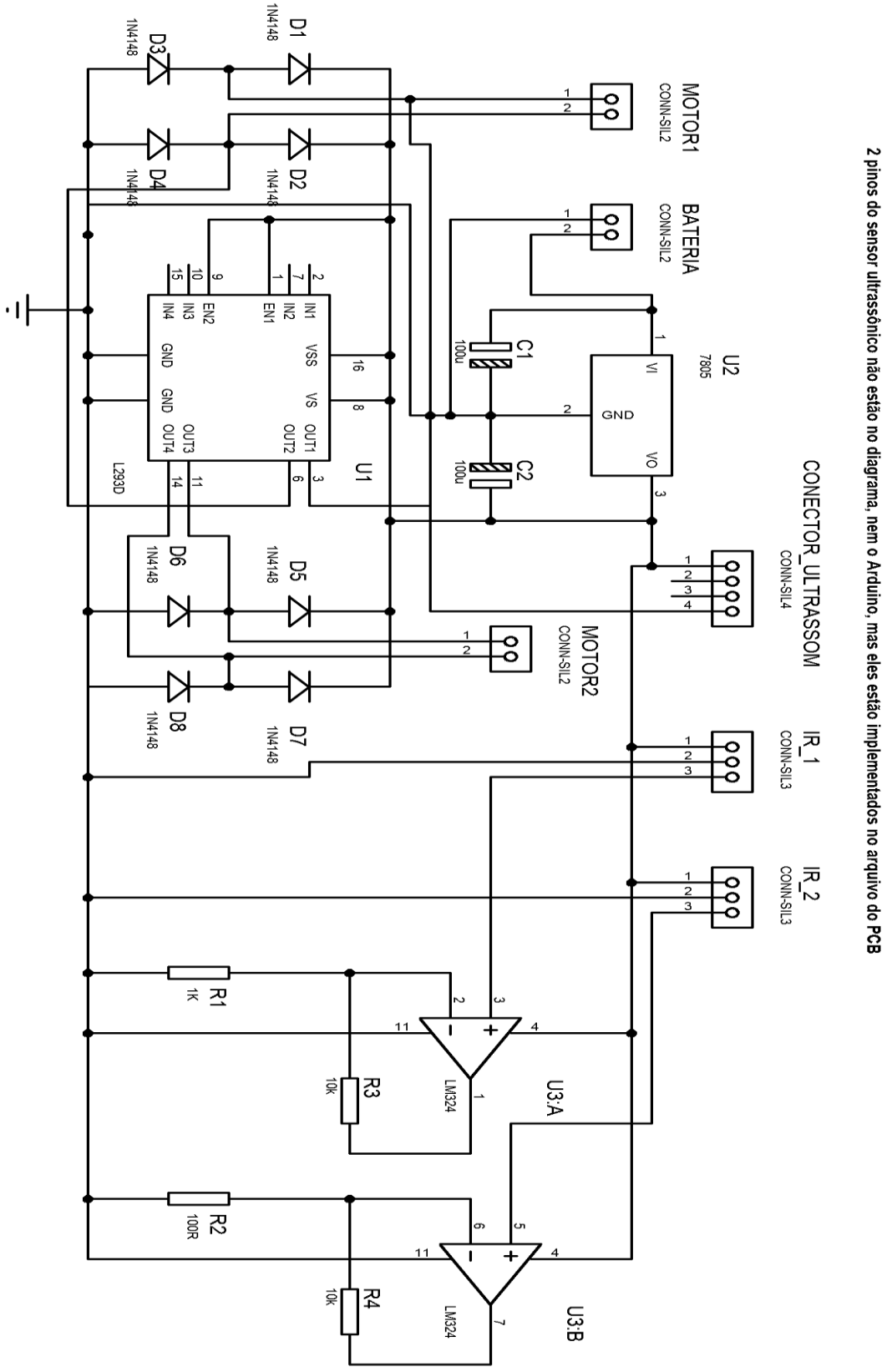
- 1 CI LM324 (AMPOP);

- 1 CI 7805 (Regulador de tensão linear);

- 1 CI L293B (ponte-H).

A CPU recebe a tensão de 8 V (típica), a partir da placa de monitoramento de bateria, esta por sua vez alimenta os demais componentes da placa em uma tensão regulada em 5 V, por meio do CI 7805. Na figura 39, tem-se o diagrama de circuito da CPU dos protótipos alternativos ao LEGO NXT.

Figura 39 – Diagrama de circuito da CPU (ISIS).



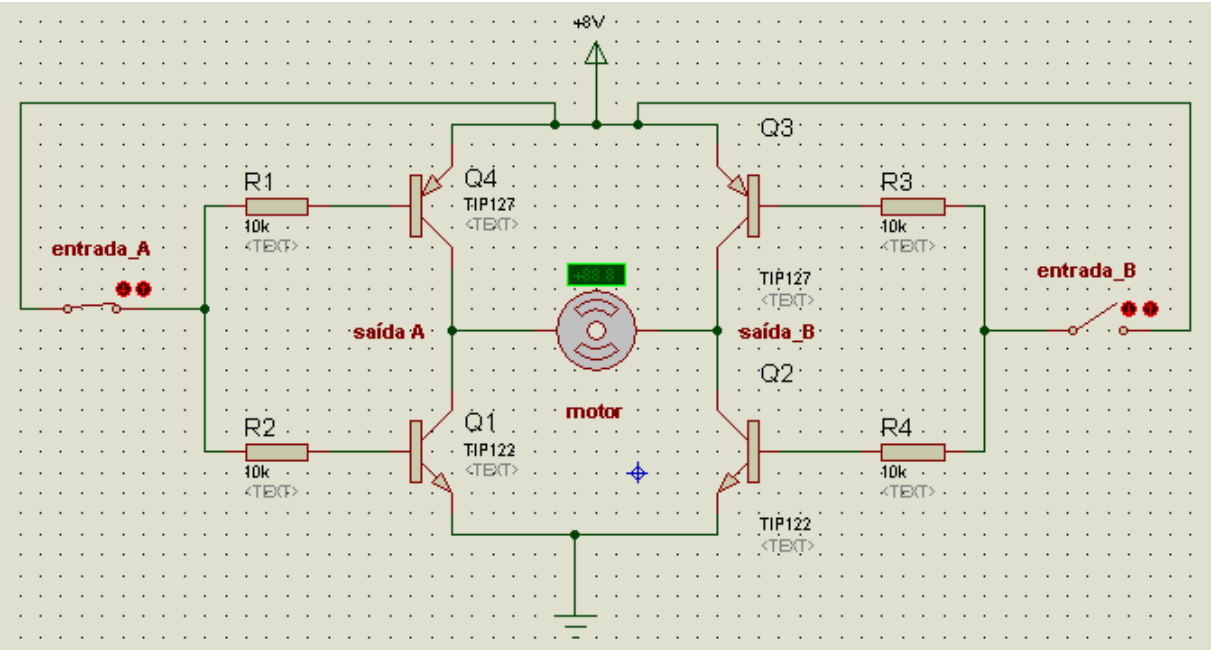
Fonte: Autor (2018).

O CI LM324 (U3) serve para amplificar o sinal elétrico proveniente dos sensores ópticos, localizados na parte frontal dos robôs, sendo posteriormente lidos pelo Arduino Nano.

O CI L293B é um driver (2 canais, 1 canal por motor), para acionamento de motores com corrente nominal de até 1 A (corrente de pico de 2 A) por canal. Este CI é responsável por amplificar os sinais de comando do Arduino (este último possui corrente da ordem de 20 mA por pino, insuficiente para os motores), gerando um sinal de corrente maior em sua saída (do CI L293B), suficiente para o acionamento dos motores LEGO.

Internamente, o CI L293B é uma ponte-H (figura 40) de 2 canais, 1 canal por motor CC, ou seja, um circuito de acionamento de cargas (geralmente indutivas, como motores CC). É um circuito lógico, seu funcionamento se dá pelo seguinte exemplo, se a chave à esquerda (entrada A) estiver fechada (nível lógico 1), o transistor Q4 (PNP) estará aberto e Q1 (NPN) estará em condução, a saída A para o motor (à esquerda), estará em nível lógico 0. Se a chave à direita (entrada B) estiver em nível lógico 0, o transistor Q3 (PNP) estará em condução e o transistor Q2 (NPN) estará aberto, fazendo com que a saída à direita fique em nível lógico 1. A tabela 1 ilustra todas as possíveis combinações de entradas e suas respectivas saídas.

Figura 40 – Diagrama de circuito de uma ponte-H.



Fonte: Autor (2018).

Tabela 1- Saída do motor, conforme as entradas aplicadas à ponte-H.

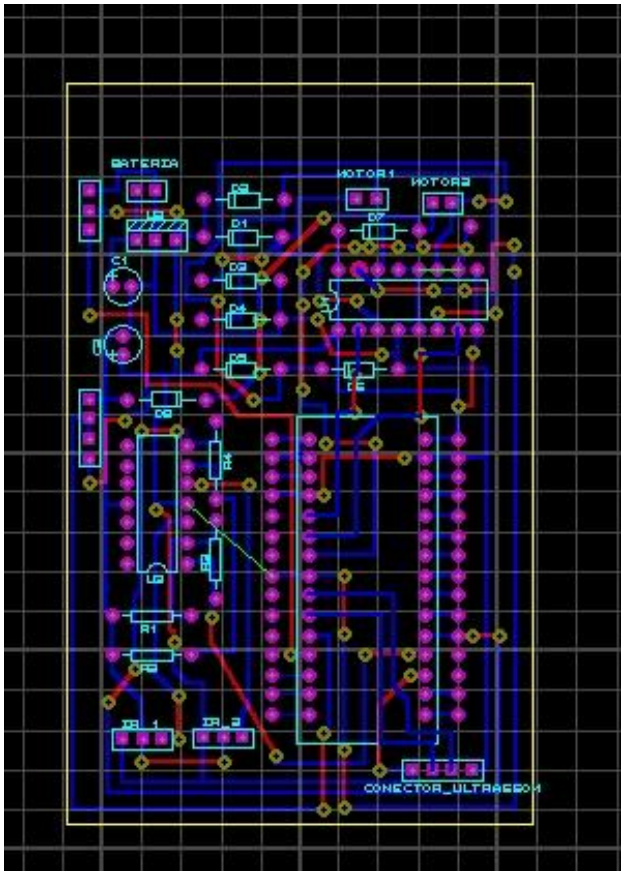
Entrada A	Entrada B	Saída_A	Saída_B	V_AB (A-B)
0 (0 V)	0 (0 V)	1 (8 V)	1 (8 V)	0 V
0 (0 V)	1 (8 V)	1 (8 V)	0 (0 V)	8 V
1 (8 V)	0 (0 V)	0 (0 V)	1 (8 V)	- 8 V
1 (8 V)	1 (0 V)	0 (0 V)	0 (0 V)	0 V

Fonte: Autor (2018).

Quando os valores de entrada forem iguais, a tensão relativa entre a saída_A e saída_B é de 0 V.

Para o circuito de CPU dos protótipos, o mesmo possui 8 diodos 1N4148 (D1 a D8), de forma a proteger o CI L293B da tensão reversa durante o chaveamento dos motores (liga-desliga e mudança do sentido de rotação dos mesmos). Na figura 41, tem-se a PCB da CPU, obtida através do software Proteus.

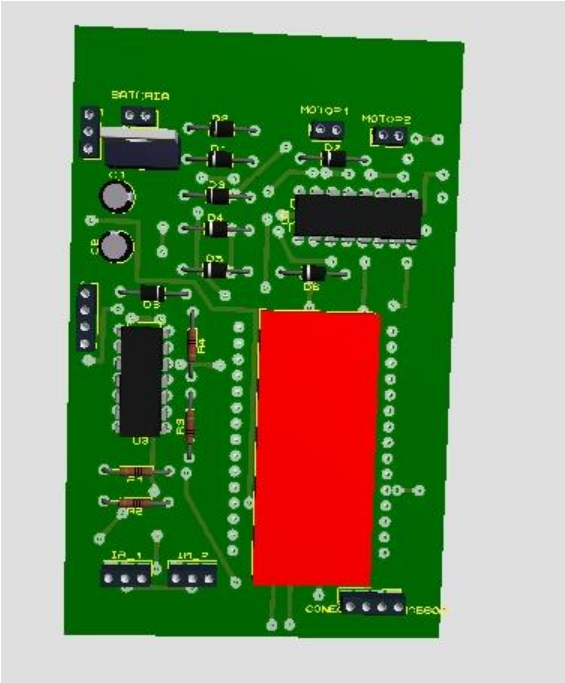
Figura 41 – PCB da CPU (ARES).



Fonte: Autor (2018).

No diagrama de circuito da CPU, o Arduino não está presente devido ao fato do software Proteus (Ares/ISIS) não possuir o mesmo em sua biblioteca. A fim de contornar essa situação, foi adicionado um encapsulamento (package), por parte do autor, com as mesmas dimensões do Arduino (em mils), durante o projeto da PCB. A figura 42 ilustra a renderização obtida com o ARES.

Figura 42 – PCB renderizada da CPU (ARES).

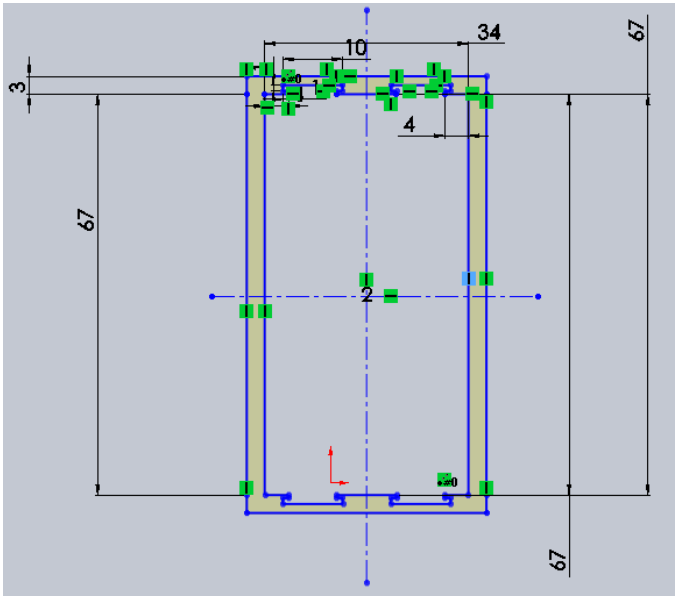


Fonte: Autor (2018).

4.1.4 – Bateria

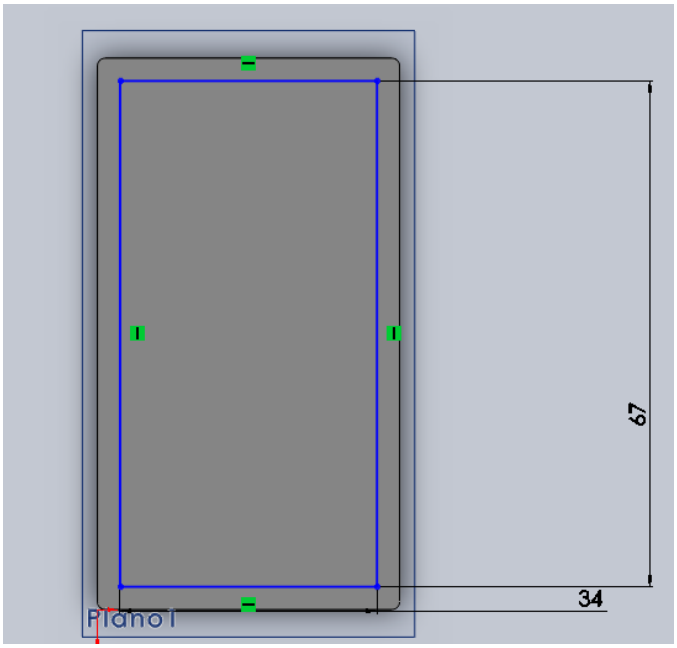
A bateria foi desenvolvida para 2 células de lítio, de forma que as células fossem associadas em série. Foi desenvolvido um case (figura 43 e 45) para 2 células, com tampa (figuras 44 e 46). Internamente à bateria, há um CI LM 35, sendo este um sensor linear de temperatura (com resolução de saída de 10 mV / ºC). O case foi desenvolvido mediante impressão 3D, no próprio LMM. Há 2 tipos de cases, de dimensões distintas, um utilizando as células CGR17670A e o outro, utilizando células INR18650 e células UR18650A.

Figura 43 – Dimensões do case para as célula CGR17670A.



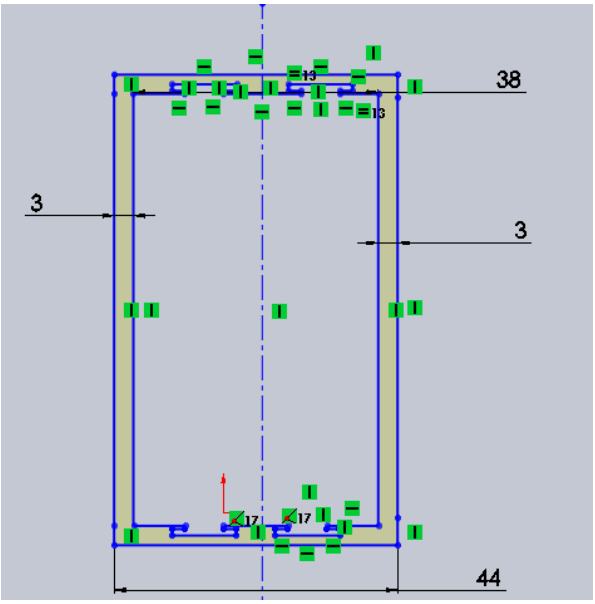
Fonte: Autor (2018).

Figura 44 – Dimensões da tampa para as células CGR17670A.



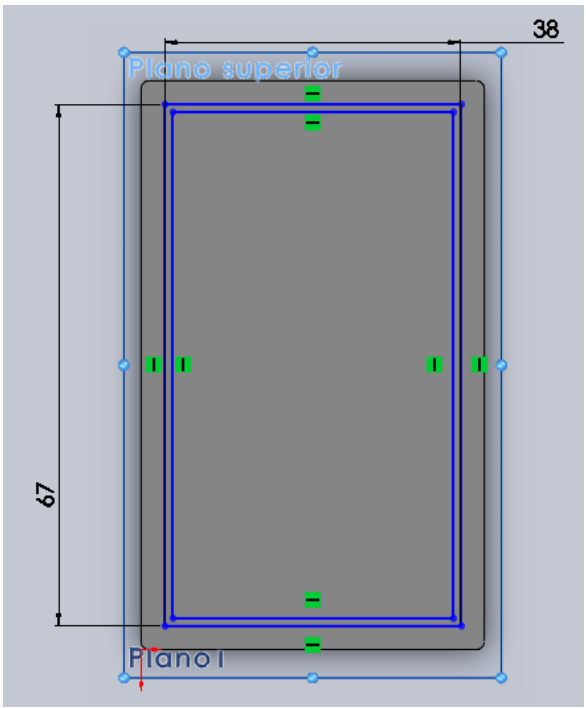
Fonte: Autor (2018).

Figura 45 – Dimensões do case para as células INR18650 e UR18650A.



Fonte: Autor (2018).

Figura 46 – Dimensões da tampa para as células INR18650 e UR18650A.



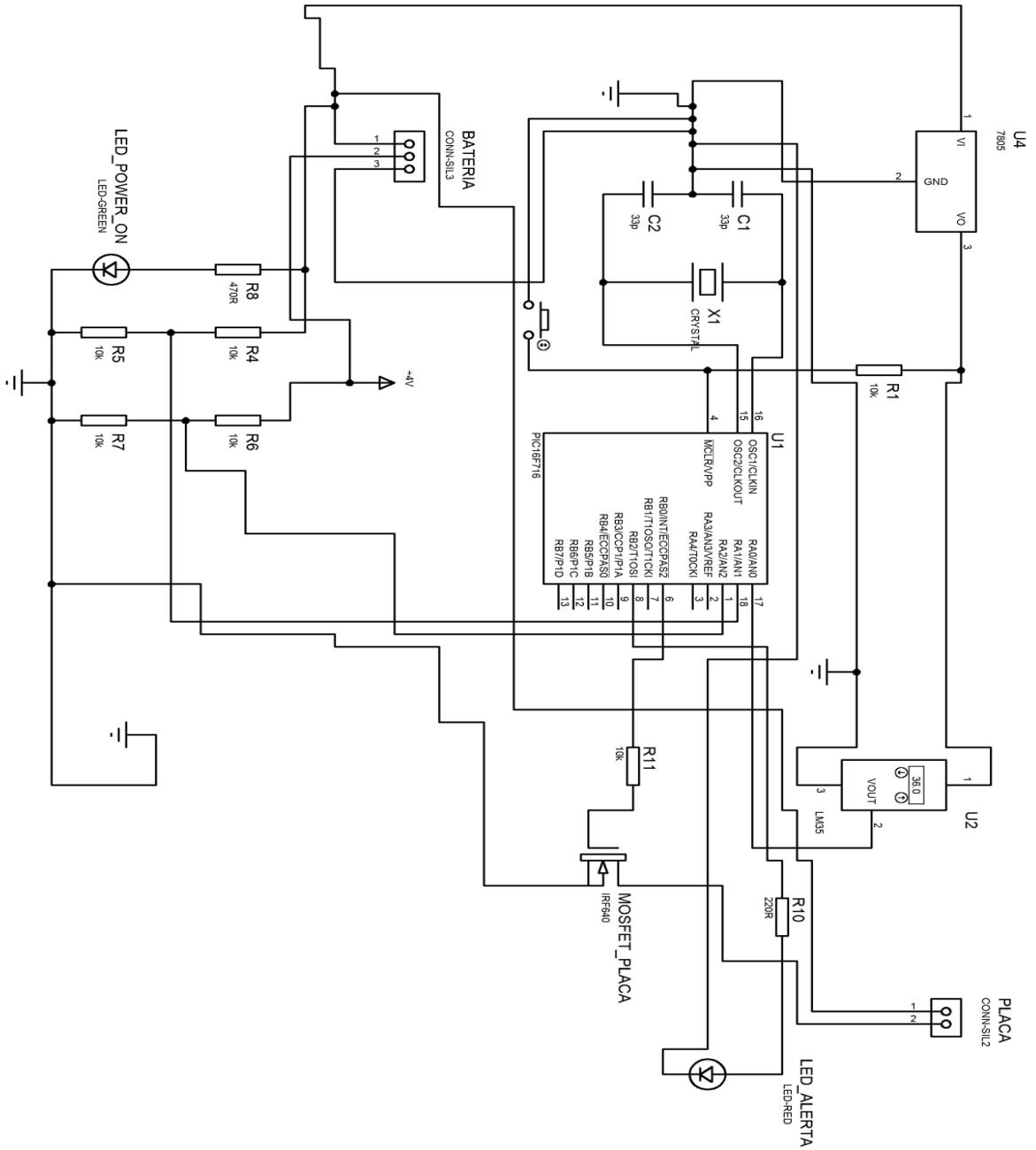
Fonte: Autor (2018).

4.1.5 – Monitor de Bateria

Para as baterias, foi desenvolvido um circuito de monitoramento para cada uma das células que compõe a bateria dos protótipos. O circuito utiliza o microcontrolador PIC16F716 (Microchip). Foi utilizado este microcontrolador devido à presença de pinos capazes de serem configurados como entrada analógica (disponibilidade de 4 pinos) e por possuir um ADC de 8 bits de resolução.

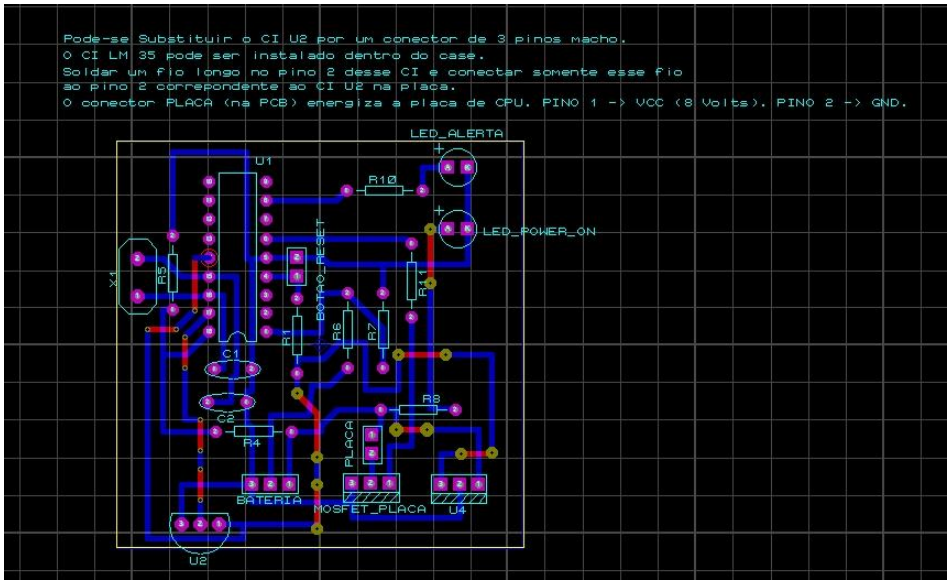
Este circuito (figura 47) realiza o monitoramento de tensão de cada célula individualmente, bem como o monitoramento de temperatura das células. O monitor de bateria alimenta a CPU dos robôs. Caso ocorra uma diminuição da tensão elétrica de pelo menos uma das células (tensão monitorada for igual ou menor que 3 V por célula), bem como superaquecimento (temperatura igual ou maior que 60 ºC), o monitor de bateria realiza o desligamento da CPU, por meio de um MOSFET de proteção (IRF640). Nas figuras 48 e 49, tem-se a PCB e a PCB renderizada, respectivamente.

Figura 47 – Diagrama de circuito do monitor de bateria.



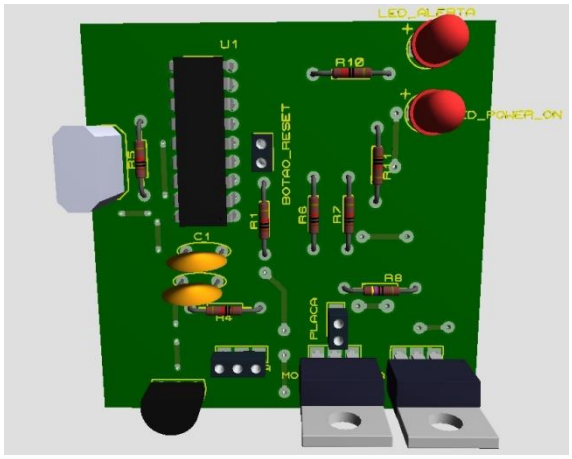
Fonte: Autor (2018).

Figura 48 – PCB do monitor de bateria.



Fonte: Autor (2018).

Figura 49 – PCB renderizada do monitor de bateria.



Fonte: Autor (2018).

Quando o robô é ligado e a bateria está em níveis “normais” de temperatura e tensão, somente o LED “POWER_ON” (verde) permanece aceso. Caso a bateria possua pelo menos uma das células com tensão abaixo de 3 V, o LED de alerta (vermelho) permanece aceso, juntamente com o LED “POWER_ON”, indicando que a bateria do robô deve ser submetida à operação de recarga. Caso haja superaquecimento, o LED de alerta (vermelho) permanecerá piscando, fazendo com que a CPU do robô seja religada quando a temperatura cair para 45 ºC.

4.1.6 – Carregador de Bateria

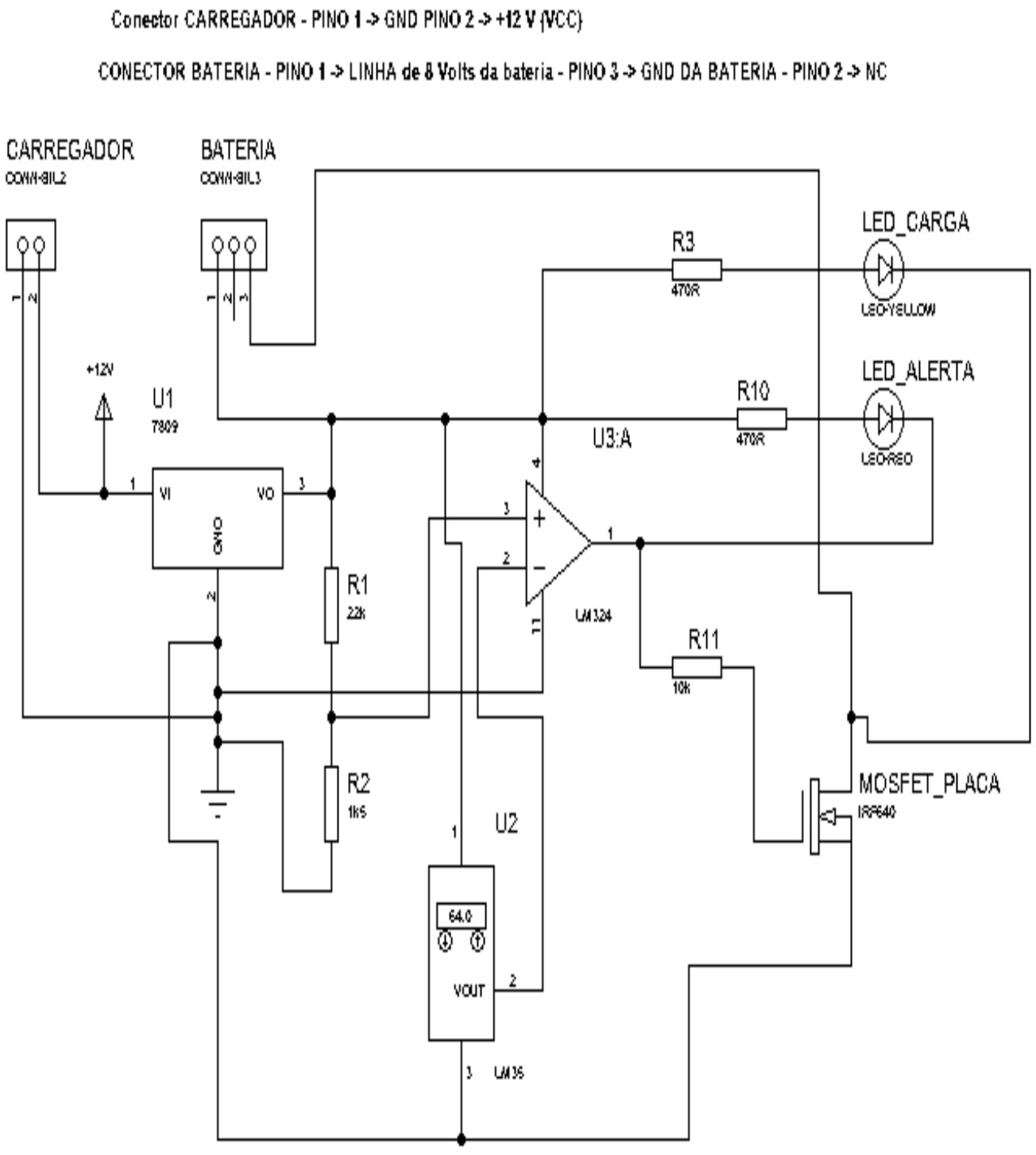
Para realizar o processo de recarga de bateria, foi desenvolvido um circuito de recarga (figura 50) operando como uma fonte de tensão constante.

O elemento responsável pela operação de recarga é o CI 7808, um regulador de tensão linear com tensão de saída de 8 V, utilizada para alimentar as baterias no processo de recarga. O CI é dotado de dissipador de calor, devido à dissipação térmica oriunda da elevada corrente de recarga inicial.

O Circuito possui um MOSFET (IRF640) de proteção para o caso de superaquecimento das células que compõe a bateria.

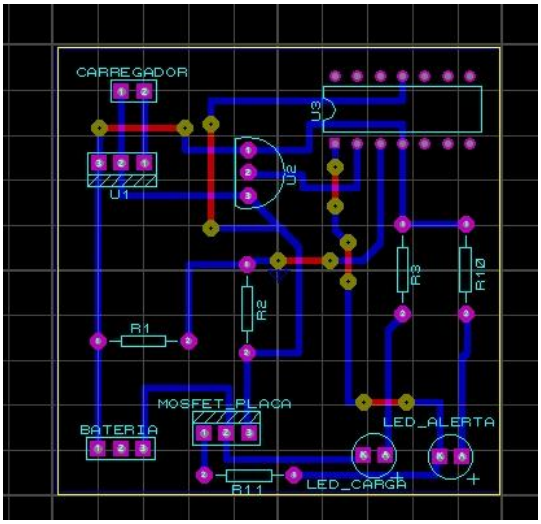
Foi utilizado um AMPOP (LM324) na configuração de um comparador de tensão, possuindo uma tensão pré-fixada de referência e a compara com a tensão proveniente do sensor de bateria. Se a tensão do sensor de temperatura (LM35) for maior que este limiar, a saída do AMPOP vai a nível lógico 0, fazendo com o que o MOSFET deixe de conduzir, interrompendo o processo de carga da bateria. Foi desenvolvido um carregador de bateria por robô. Nas figuras 51 e 52, tem-se a PCB e a PCB renderizada, respectivamente.

Figura 50 – Diagrama do carregador de bateria (ISIS).



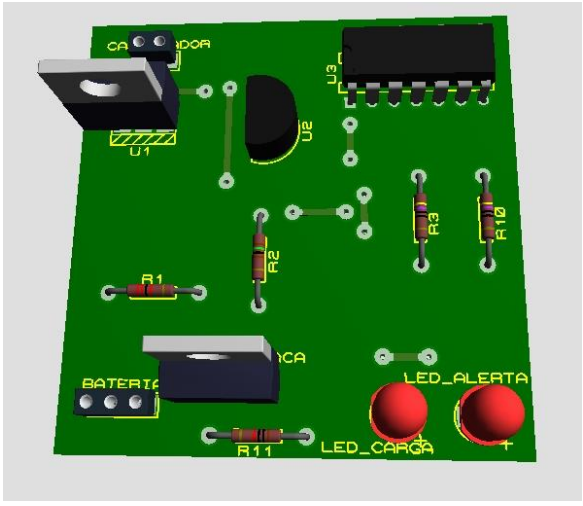
Fonte: Autor (2018).

Figura 51 – PCB do carregador de bateria (ARES).



Fonte: Autor (2018).

Figura 52 – PCB renderizada do carregador de bateria (ARES).



Fonte: Autor (2018).

4.2 – Montagem dos Protótipos

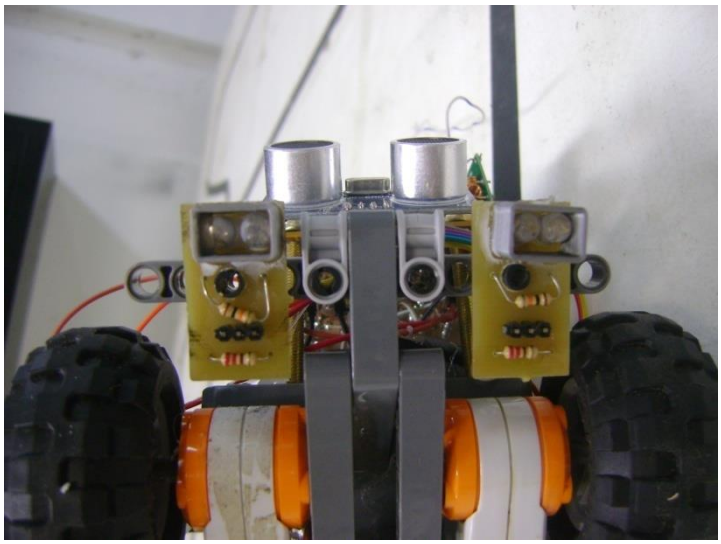
Nesta seção, serão apresentadas algumas figuras pertinentes à montagem dos protótipos, dos sensores ópticos (figuras 53 e 64), CPU (figura 55), bateria (figuras 56 e 57), o monitor de bateria (figura 58) e o carregador da mesma (figuras 61 e 62). Também serão apresentados o gravador do PIC16F716 e os 8 protótipos finalizados.

4.2.1 – Sensores ópticos

Figura 53 - Sensor óptico desenvolvido

Fonte: Autor (2018).

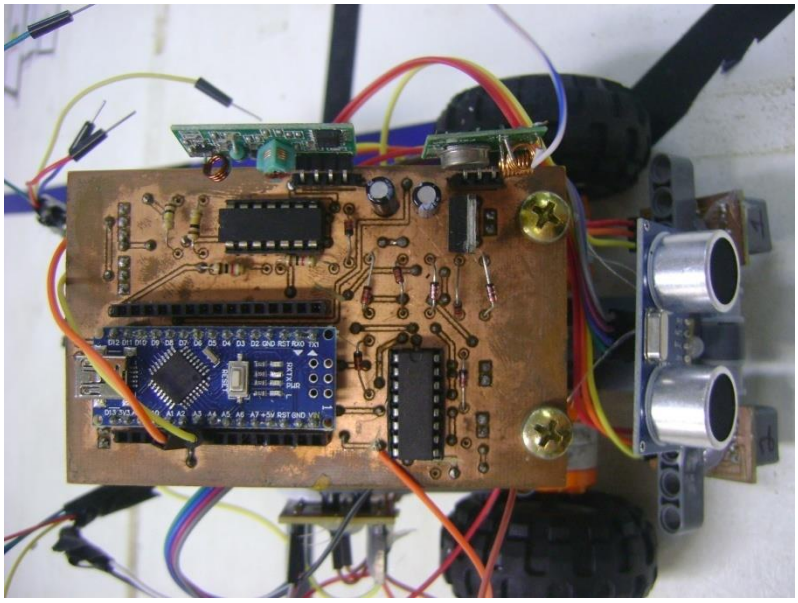
Figura 54 – Sensor óptico, montados à parte frontal do protótipo.



Fonte: Autor (2018).

4.2.2 – CPU

Figura 55 – Vista da CPU do protótipo.



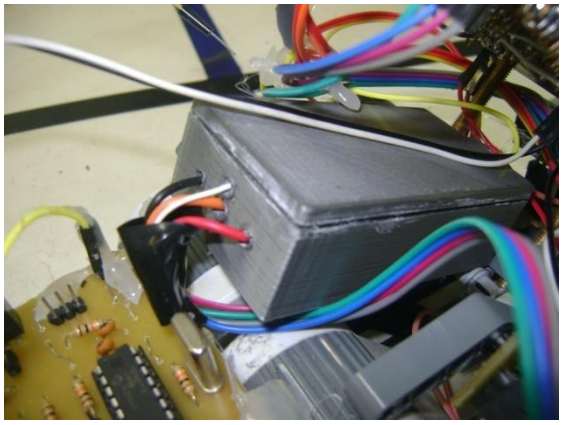
Fonte: Autor (2018).

4.2.3 – Baterias

Figura 56 – Case e tampa da bateria para as células CGR17670A.

Fonte: Autor (2018).

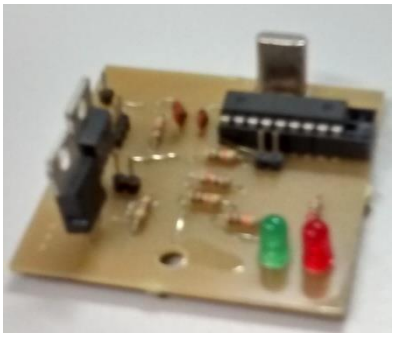
Figura 57 – Case e tampa da bateria para as células INR18650 e UR18650A.



Fonte: Autor (2018).

4.2.4 – Monitor de Bateria

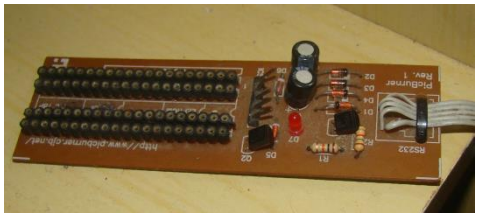
Figura 58 – Monitor de bateria.



Fonte: Autor (2018).

Para o circuito de monitoramento de bateria, foi utilizado para a programação do PIC16F716, o gravador PICBURNER (figura 59), de posse do próprio autor.

Figura 59 – gravador PICBURNER



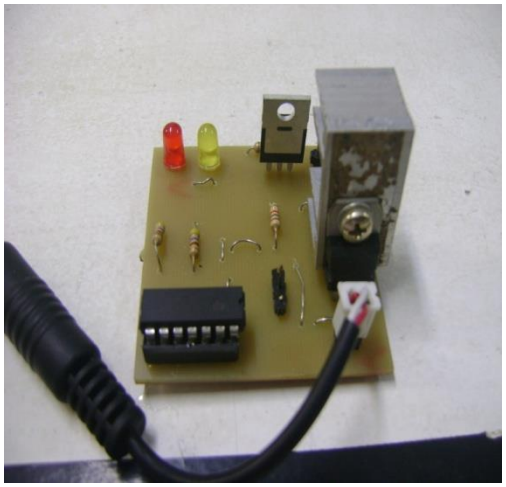
Fonte: Autor (2018).

A programação do PIC16F716 foi realizada no sistema operacional Ubuntu 18.04 (LINUX), através do software PICPgm. Ele não possui interface gráfica, todo o processo de programação/limpeza de dados do PIC é realizado através de comando de texto.

O próprio software tem a vantagem de fazer a identificação do PIC automaticamente, uma vez conectado ao gravador.

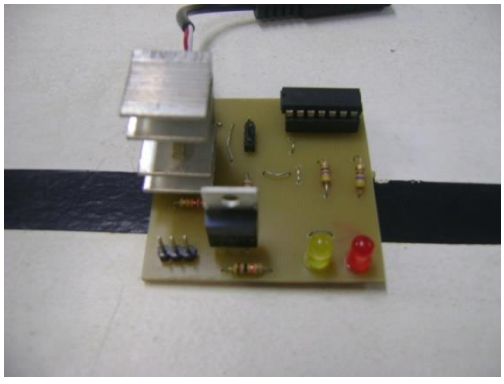
4.2.5 – Carregador de Bateria

Figura 60 – Carregador de bateria (regulador de tensão 7808 com dissipador de calor).



Fonte: Autor (2018).

Figura 61 – Carregador de bateria (vista do MOSFET IRF640, está sem dissipador de calor).

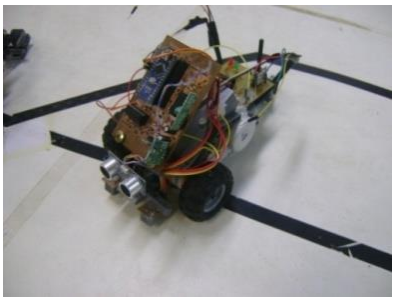


Fonte: Autor (2018).

4.2.6 – Protótipos Desenvolvidos no LMM

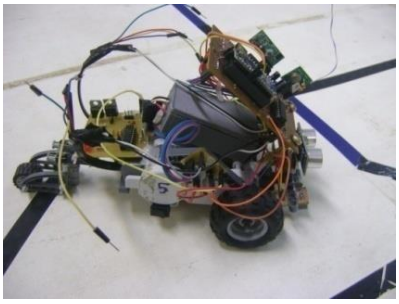
Os protótipos são iguais entre si (homogêneos). Cada um deles recebeu um número de identificação, de 1 a 8. Os protótipos são apresentados nas figuras 62, 63 e 64.

Figura 62 – Protótipo desenvolvido (robô 5), pronto para ser utilizado, no LPR.



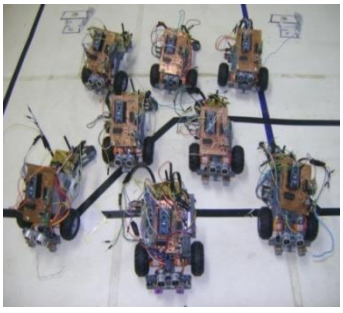
Fonte: Autor (2018).

Figura 63 – Vista lateral do protótipo (robô 5).



Fonte: Autor (2018).

Figura 64 – Os 8 robôs desenvolvidos.



Fonte: Autor (2018).

5 COOPERAÇÃO DESENVOLVIDA ENTRE OS PROTÓTIPOS

5.1 Introdução

Nesta seção, será dada uma visão conceitual sobre cooperação. Feito isso, a estratégia de cooperação desenvolvida para os protótipos será abordada.

A Robótica Cooperativa é uma área pertinente ao chamado “Sistemas Multi-Robôs”, possuindo grande interesse por parte dos pesquisadores. Em um sistema Multi-Robôs, robôs cooperam uns com os outros de forma a cumprir certas tarefas. Através de cooperação, robôs móveis são capazes de lidar com tarefas em que um único robô seria incapaz de realizá-las. Através de cooperação, pode-se criar um sistema redundante, onde a falha de um robô seria compensada por outros (LOH; TRAECHTLER, 2012).

Outra grande característica da robótica cooperativa, está relacionada à menor necessidade de robôs de grandes dimensões, complexos e eventualmente de alto valor agregado, e sim, a possibilidade de se utilizar robôs menores, mais simples e baratos (CARVALHO, 2015).

Para que a cooperação de fato seja viabilizada, a comunicação é fator crucial. Através da comunicação, as capacidades de percepção dos robôs (agentes) são expandidas, permitindo aos mesmos, beneficiarem-se da informação de conhecimento de outros robôs. Em sistemas cognitivos, a mesma se dá por troca de mensagens, em sistemas reativos, dá-se por meio da difusão de um sinal no ambiente (FERBER, 1999).

Do ponto de vista conceitual, as algumas definições de cooperação são:

- Comportamento colaborativo, dirigido rumo a uma meta, no qual existe um interesse comum ou recompensa (BARNES; GRAY, 1991);
- Uma forma de interação, geralmente baseada em comunicação (MATARIC, 1994);
- Unir-se para a realização de algo que crie um resultado progressivo, como aumento de desempenho ou economia de tempo (PREMVUTI; YUTA, 1990).

5.2 – Projetos pertinentes à Robótica Cooperativa

Existem muitos projetos relacionados à área de cooperação entre robôs, dado o seu grande potencial nas diversas aplicações. Nesta seção, serão abordados alguns projetos.

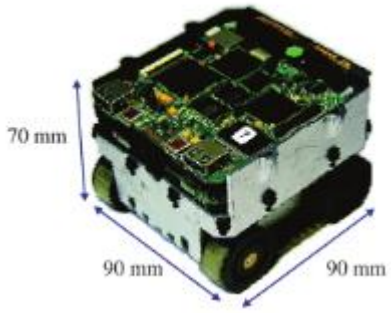
5.2.1 – Bebot

Robôs desenvolvidos na Universidade de Paderborn, na Alemanha.
O projeto consistiu em utilizar 3 robôs não-holônomicos para transporte de carga.

Não há comunicação explícita entre os robôs, a cooperação é realizada mediante interações físicas (como forma de comunicação indireta) entre eles.

A estratégia de cooperação foi baseada em seguimento de líder (1 robô como líder e outros 2 como seguidores de líder). Pode-se visualizar o Bebot na figura 65.

Figura 65 - Bebot



Disponível em: <https://cyberleninka.org/article/n/1510465/viewer >. Acesso em: 17 nov. 2018.

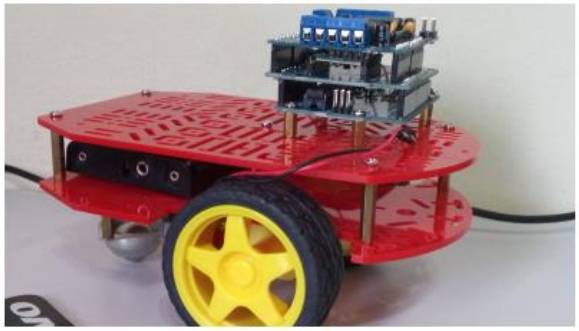
5.2.2 – Hivebots

Projeto desenvolvido na Universidade Tecnológica Federal do Paraná - UTFPR. Foram utilizados 3 robôs (utilizando Arduino Uno) e um PC (estação base). A comunicação utilizada (tanto no PC quanto nos robôs) foi baseada em Xbee.

O projeto consistiu em navegação e transporte de carga cooperativa, tendo o PC como mestre (envia os comandos para os robôs, as ações que os mesmos

devem tomar), bem como recebe dos robôs, suas coordenadas e calcula o caminho ótimo para os robôs (SATO; SANTOS; ALMEIDA, 2015). Na figura 66 tem-se a imagem do robô utilizado.

Figura 66 – Hivebots.



Disponível em: <http://paginapessoal.utfpr.edu.br/gustavobborba/if66j-s71-projetos/files/IF66J-15b_RT_Hivebots.pdf>. Acesso em: 17 nov. 2018.

5.2.3 – Kilobots

Os Kilobots (figura 67) fazem parte da “Swarm Robotics”, ou seja, Robótica Cooperativa bioinspirada, baseada em colônias de abelhas e formigas. Foram desenvolvidos para permitirem aos pesquisadores uma maior compreensão sobre comportamento coletivo e a partir dessa compreensão, desenvolverem soluções para os mais diversos problemas.

Utilizam como microcontrolador, o ATmega328P (mesmo microcontrolador da plataforma Arduino Nano utilizado nos protótipos) e a comunicação com outros robôs se dá por meio de LED’s IR (K-TEAM, [s. d.]).

Figura 67 – Kilobot.

Disponível em: <https://www.k-team.com/mobile-robotics-products/kilobot#manual>. Acesso em: 17 de nov. 2018.

5.2.4 – SWARM-BOTS

Projeto bioinspirado. Consiste em robôs com comportamento coletivo, baseado em colônias de insetos sociais e outros animais.

O projeto é focado no estudo dos mecanismos que governam processos de auto-organização e auto-agrupamento de robôs.

O comportamento individual e coletivo se dá por meio da interação entre os SWARM-BOTS. Os SWARM-BOTS (figura 68) possuem atuadores (braços e garras), os quais permitem acoplamento físico entre os robôs. (DORIGO et. al., 2005).

Figura 68 – SWARM-BOTS.

Disponível em: <https://www.researchgate.net/publication/221116556_The_SWARM-BOTS_project>. Acesso em: 17 nov. 2018.

5.3 Estratégia Desenvolvida no Laboratório de Projetos (LPR)

A estratégia desenvolvida, por parte do autor, no Laboratório de projetos (LPR), foi baseada em controle de formação. Consistiu em utilizar 3 robôs, 1 como líder, os outros 2 como seguidores de líder (receptores). A comunicação entre os robôs móveis é do tipo unidirecional (simplex), através de broadcast, ou seja, as mensagens captadas pelos 2 robôs seguidores de líder são idênticas.

Para a cooperação, foram utilizados os seguintes elementos dos robôs: odometria, sensores ultrassônicos e os módulos de RF.

5.3.1 Funcionamento

Os robôs devem se deslocar em linha reta, em direção ao obstáculo (2 caixas de papelão), os 3 robôs devem cessar o movimento quando a distância ao obstáculo for menor ou igual a 6 cm.

Cada um dos robôs mede continuamente sua distância em relação ao obstáculo. O robô líder envia a cada 200 ms uma mensagem sobre sua atual distância. Cada um dos seguidores compara a distância recebida, por parte do líder, e compara com a sua própria.

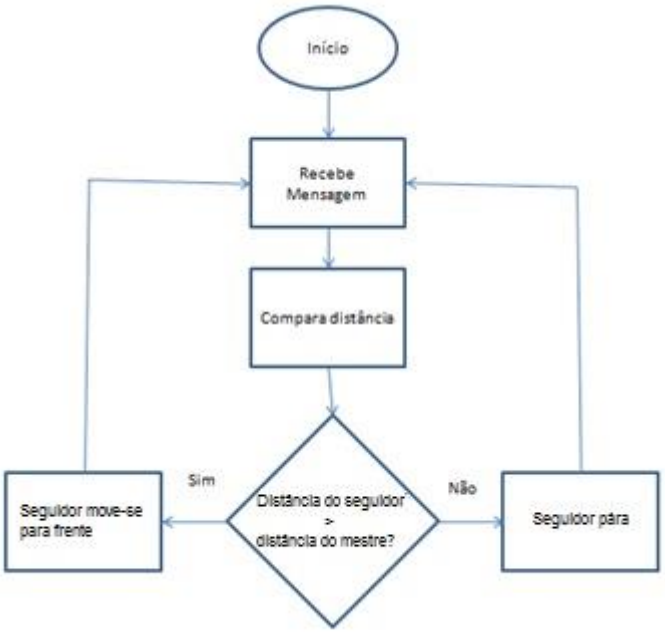
Se a distância de qualquer um dos seguidores de líder for menor que a distância do líder, os mesmos devem interromper o movimento (significa que eles estão mais adiantados do que o líder). Se a distância de qualquer um dos seguidores for maior que a distância do líder, eles movem-se em direção ao obstáculo (estão mais afastados do que o líder).

A estratégia desenvolvida tem por objetivo manter o grupo de robôs alinhados, ao longo do trajeto. Nas figuras 69, tem-se o fluxograma da estratégia desenvolvida (do ponto de vista dos seguidores) e na figura 70, a modelagem da estratégia utilizando Redes de Petri, respectivamente.

Como principais objetivos do controle de formação, são eles (GOUVÊA, 2011):

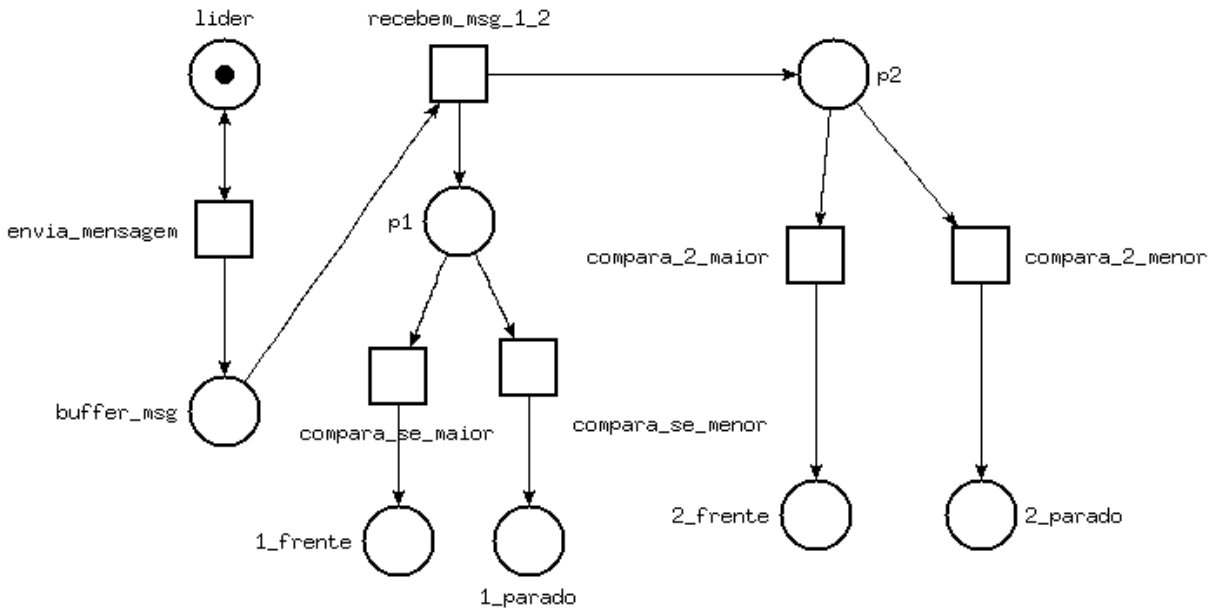
- Manter o grupo coeso;
- O grupo deve atingir um objetivo comum.

Figura 69– Fluxograma da estratégia desenvolvida (criado a Partir do Microsoft Office Powerpoint).



Fonte: Autor (2018).

Figura 70 – Estrutura de comunicação do controle de formação (realizado no software Tina).



.

Fonte: Autor (2018).

5.3.2 Vantagens da estratégia proposta

Como vantagens da estratégia proposta, são:

- Simplicidade do algoritmo;
- Facilidade de manutenção do código;
- Escalabilidade (do ponto de vista dos seguidores);
- Pouca ou nenhuma alteração do código a ser replicado nos seguidores, no máximo, a configuração de pinos pertinentes aos sensores e motores, no próprio código do Arduino.

5.3.3 Desvantagens da estratégia proposta

Como limitações decorrentes da estratégia implementada, foram:

- Um único ponto de falha, devido à comunicação ser centralizada;
- Comunicação unidirecional, não há garantia de entrega de pacotes, nem retransmissão dos mesmos;
- A coesão é mantida em relação ao líder, não aos demais.

5.4 Aplicações da Robótica Cooperativa

Entre as aplicações, são algumas:

- Geração de mapas (KHAMIS; ELGINDY, 2012);
- Patrulhamento (CARVALHO, 2015);
- Resgate de vítimas em escombros (CARVALHO, 2015);
- Desarme de minas terrestres (KHAMIS; ELGINDY, 2012);
- Transporte de cargas (LOH; TRAECHTLER, 2012);
- Operações de montagem (MAKRIS et. al., 2012);
- Agricultura (LIEKNA; GRUNDSPENKS, 2014).

6 TESTES E COMPARATIVOS DOS PROTÓTIPOS

Serão apresentados os testes realizados aos protótipos (sensores ópticos, sensor ultrassônico, motores, odometria, comunicação e comparação com os robôs LEGO).

Todos os testes foram realizados no LPR, utilizando multímetro (figuras 72 e 73) e fonte de alimentação (tensão regulável de 0 V a 32 V, corrente regulável de 0 A à 3 A, apresentada na figura 71).

Os gráficos foram obtidos através do software MATLAB.

Figura 71 – Fonte de alimentação utilizada no LPR (MINIPA MPS-300S).

Fonte: Autor (2018).

Figura 72 – Multímetro MAS838 (disponível no LMM)

Fonte: Autor (2018).

Figura 73 – Multímetro digital DT830B (de posse do autor)

Fonte: Autor (2018).

6.1 Testes dos Protótipos

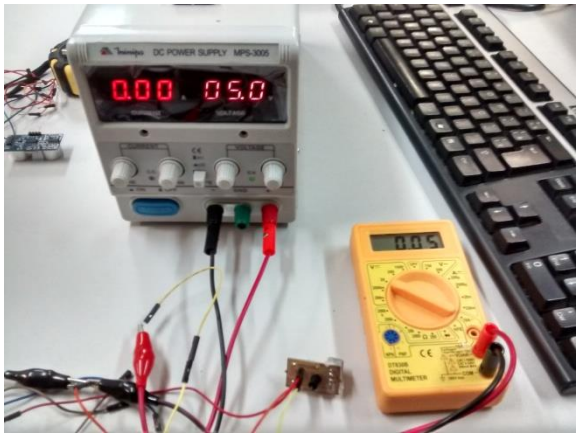
6.1.1 Sensores ópticos

Os testes dos sensores ópticos foram realizados com fonte de alimentação, ou seja, desconectados dos robôs, testados isoladamente. Os sensores foram alimentados com tensão de 5 V.

Os testes foram realizados em superfície clara e superfícies escuras (para isso, utilizou-se uma fita isolante preta), medindo-se a tensão de saída dos sensores em cada umas dessas superfícies. Além da comparação entre os valores claros e escuros, o teste realizado levou em consideração à altura do sensor à superfície, iniciando-se com o sensor na superfície (altura de 0 cm), até a altura de 10 cm, em relação à mesma, com passo incremental de 1 cm por medição realizada.

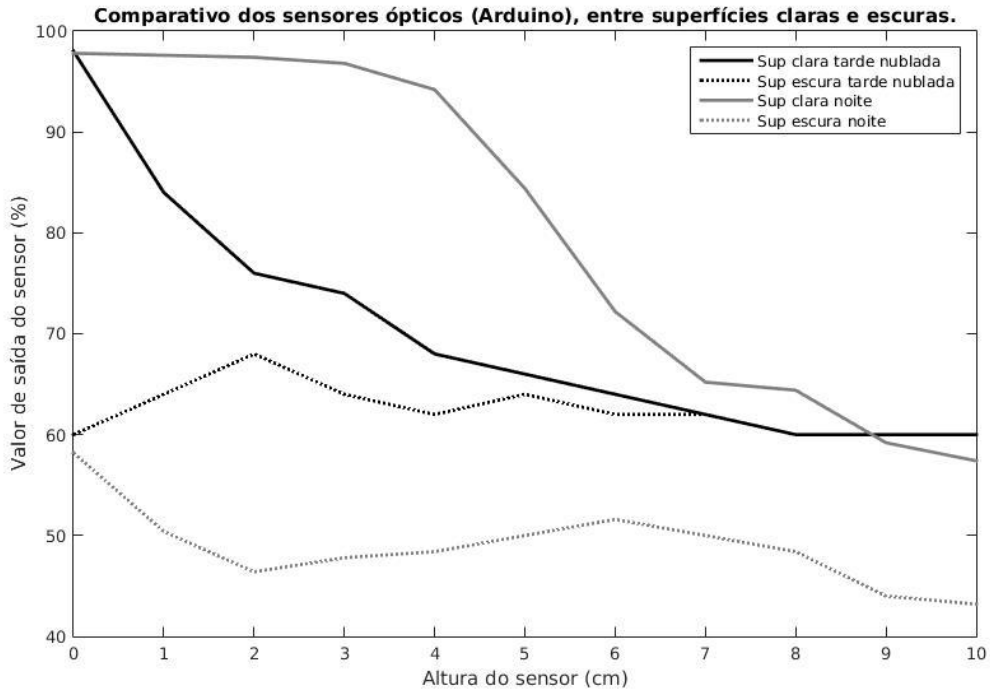
Os valores de saída dos sensores (dos protótipos desenvolvidos) não estão em Volts, e sim em percentual. O LEGO NXT mede em percentual, portanto, foi mais conveniente ilustrar os valores dos sensores (dos protótipos desenvolvidos) na mesma unidade do LEGO NXT. A figura 74 ilustra a alimentação do sensor óptico e a figura 75, os resultados da medição dos sensores.

Figura 74 – Alimentação do sensor óptico



Fonte: Autor (2018).

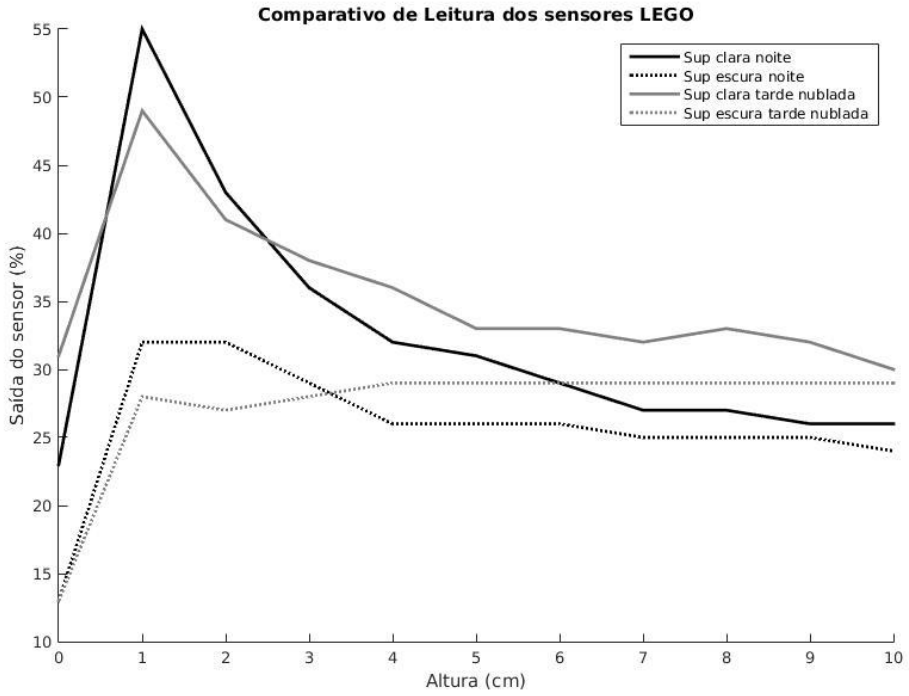
Figura 75 – Teste de leitura dos sensores ópticos dos protótipos desenvolvidos.



Fonte: Autor (2018).

Além do teste com o sensor óptico, dos protótipos desenvolvidos com base na plataforma Arduino, foi realizado o teste do sensor óptico do LEGO NXT, como se pode visualizar na figura 76.

Figura 76 – Teste de leitura dos sensores ópticos dos robôs LEGO NXT.



Fonte: Autor (2018).

6.1.2 Sensor Ultrassônico

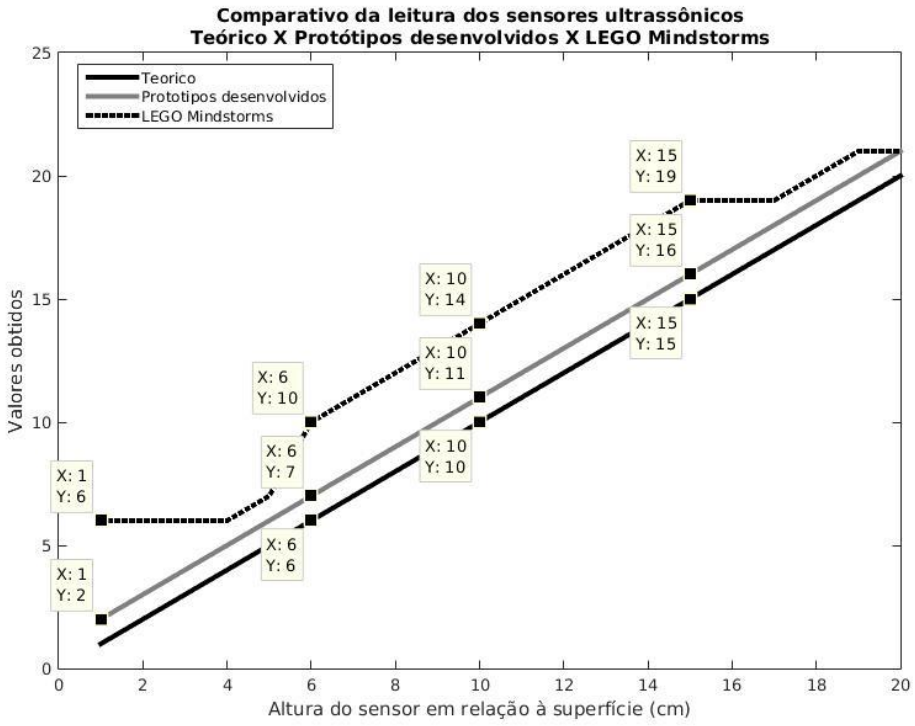
No teste dos sensores ultrassônicos, foi realizada a medição utilizando-se o sensor do Arduino (HC-SR04) e utilizando-se também, o sensor do LEGO NXT.

Os testes foram realizados sobre uma superfície plana (mesa do LPR), partindo-se de uma altura relativa de 1 cm, entre o sensor e a superfície, até a altura de 20 cm, com passo incremental de 1 cm por medição. Na figura 77, são apresentados os sensores LEGO NXT e o sensor HC-SR04 (dos protótipos) e na figura 78, são apresentados os resultados das medições realizadas com os sensores LEGO e HC-SR04.

Figura 77 – Sensor ultrassônico LEGO (esquerda) e sensor HC-SR04 (direita), utilizado nos protótipos.

Fonte: Autor (2018).

Figura 78 – Teste dos sensores ultrassônicos.

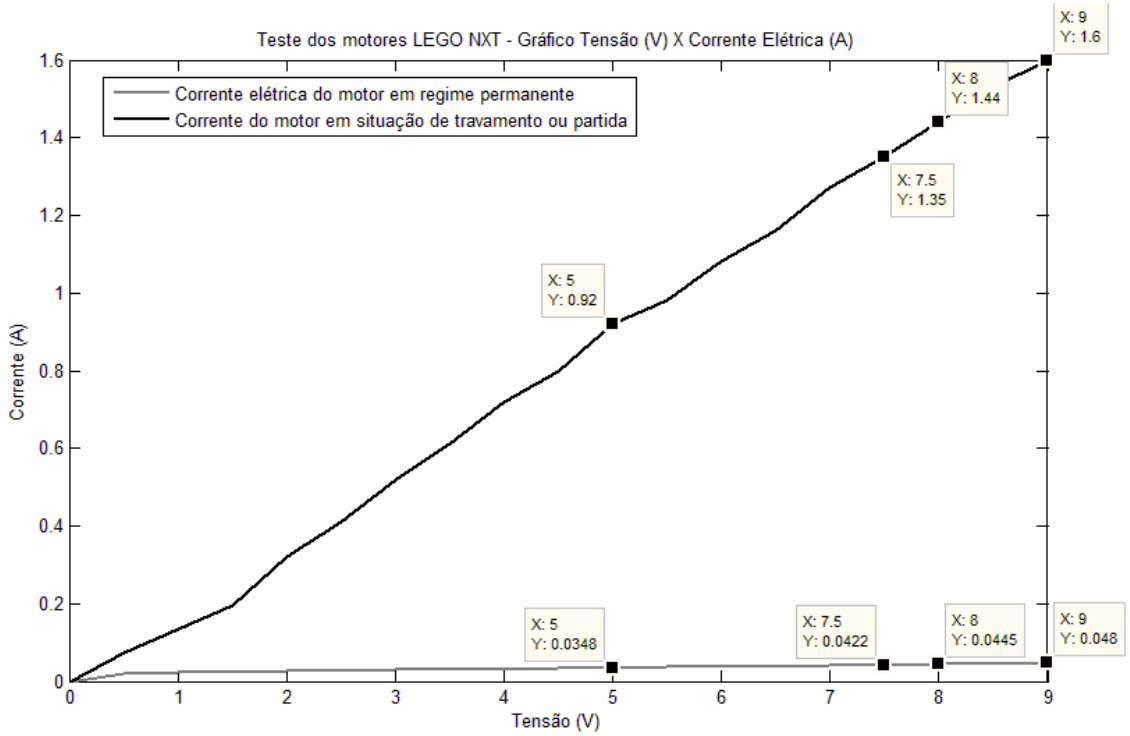


Fonte: Autor (2018).

6.1.3 Motores

Os motores LEGO NXT possuem tensão de alimentação típica de 9 V. Foi realizado o teste utilizando somente um motor. O objetivo do teste foi analisar a variação de corrente elétrica do motor LEGO NXT em função da variação da tensão elétrica aplicada ao mesmo, por meio da fonte de alimentação regulável. A tensão elétrica aplicada ao motor foi iniciada em 0 V, com passos incrementais de 0,5 V até o valor nominal de 9 V. A figura 79 ilustra o resultado obtido com os testes.

Figura 79 – Gráfico do teste do motor LEGO NXT.



Fonte: Autor (2018).

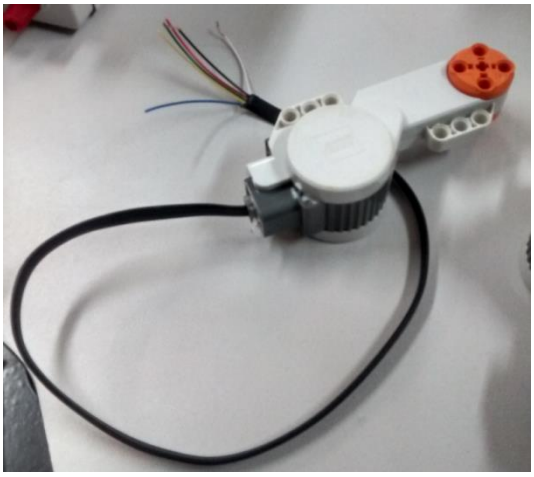
6.1.4 Odometria

Quanto aos robôs LEGO, os encoders operam no modo quadratura (além de medir o passo do motor, identificam o sentido de rotação), para tal, 2 fios do encoder do motor LEGO NXT são utilizados.

Para os protótipos desenvolvidos, eles utilizam somente 1 fio do encoder (incremental), devido à facilidade de uso. Os 2 fios podem utilizados, inclusive a fim de duplicar a resolução do mesmo (enquanto os protótipos tem resolução de 2º/pulso, os robôs LEGO tem resolução de 1º/pulso). Na figura 80, é ilustrado o motor LEGO NXT com o cabo conectado ao mesmo.

Na tabela 2, tem-se a funcionalidade de cada fio do cabo NXT e na figura 81, o sinal gerado pelo encoder de quadradura dos robôs LEGO.

Figura 80 – Motor LEGO NXT com cabo conectado (alimentação do motor e do circuito de odometria).



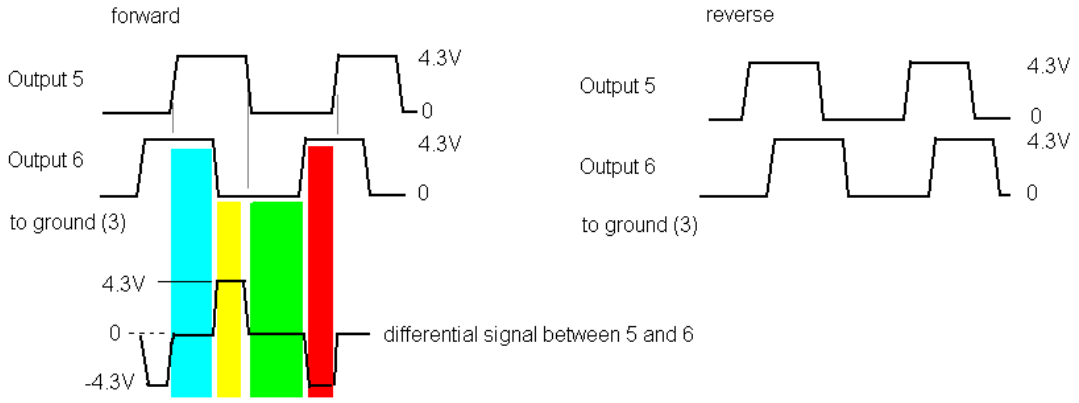
Fonte: Autor (2018).

Tabela 2 – Função de cada fio do cabo NXT em função de sua cor.

Pin number of original nxt-cable	cable color	function	signal / polarity	signal form
1	white	motor power supply	DC 9V + or -	DC
2	black	motor power supply	DC 9V - or +	DC
3	red	rotation detector supply	ground (0V)	DC
4	green	rotation detector supply	+ 4.3 (to 5.0 V)	DC
5	yellow	rotation detector output 1	0 / 4.3 V against 3, -4.3V / 0V against 4	rectangle
6	blue	rotation detector output 2	0 / 4.3 V against 3, -4.3V / 0V against 4	rectangle

Disponível em: http://trivox.tripod.com/lego-nxt-motor-input-output.html. Acesso em: 16 out. 2018.

Figura 81 – Sinal do encoder do motor LEGO NXT.



Disponível em: http://trivox.tripod.com/lego-nxt-motor-input-output.html. Acesso em: 16 out. 2018.

Para a odometria dos robôs, chegou-se à equação que correlaciona o número de pulsos do encoder à distância percorrida, isto é, sua translação (3). A variável **X** está relacionada à distância percorrida (em cm), e $\mathbf{D_{roda}}$, o diâmetro da roda dos robôs, cujo valor é igual a 5,6 cm.

$$\textbf{Nº de Pulsos (Translação)} = X*180/(\pi*D_{roda}) \quad (3)$$

A equação (4) correlaciona o número de pulsos do encoder ao ângulo de rotação dos robôs. A variável **θ** é o ângulo de rotação do robô, dado em graus e a variável **r** é o raio dos robôs (metade da distância entre uma roda e outra). O valor de **r** é igual a 6,5 cm.

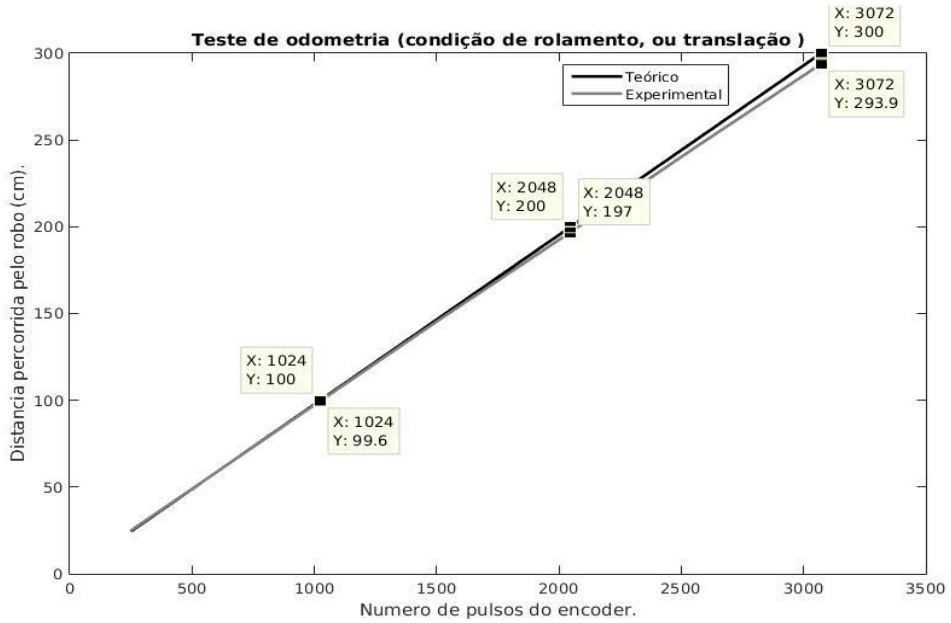
$$\textbf{Nº de Pulsos (rotação)} = r*\theta*180/(\pi*D\_roda) \quad (4)$$

Para a odometria, foram realizados 3 testes. Os testes têm por objetivo, analisar a margem de erro de estimação dos encoders. O primeiro teste (figura 82) foi realizado utilizando-se somente um robô (robô 7), de forma a percorrer as seguintes distâncias:

- 100 cm;
- 200 cm;
- 300 cm;

As distâncias (em cm) foram percorridas em função do número de pulsos do encoder. Para cada 1 m (100 cm), são necessários 1024 pulsos do encoder.

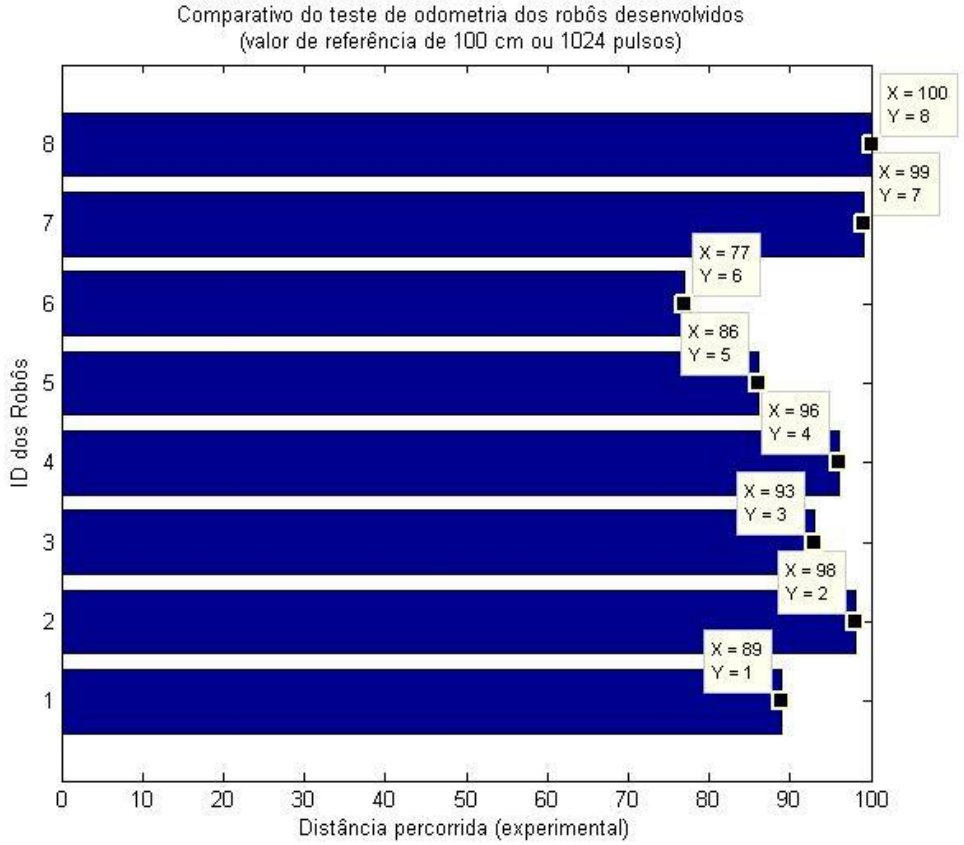
Figura 82 – Comparativo da distância teórica X experimental do encoder do robô 7, em função do Nº de pulsos do encoder (X – nº de pulsos, Y – distância percorrida em cm).



Fonte: Autor (2018).

O segundo teste com o encoder foi realizado entre todos os robôs desenvolvidos. A distância de referência foi de 1 m (100 cm). A figura 83 ilustra o teste realizado.

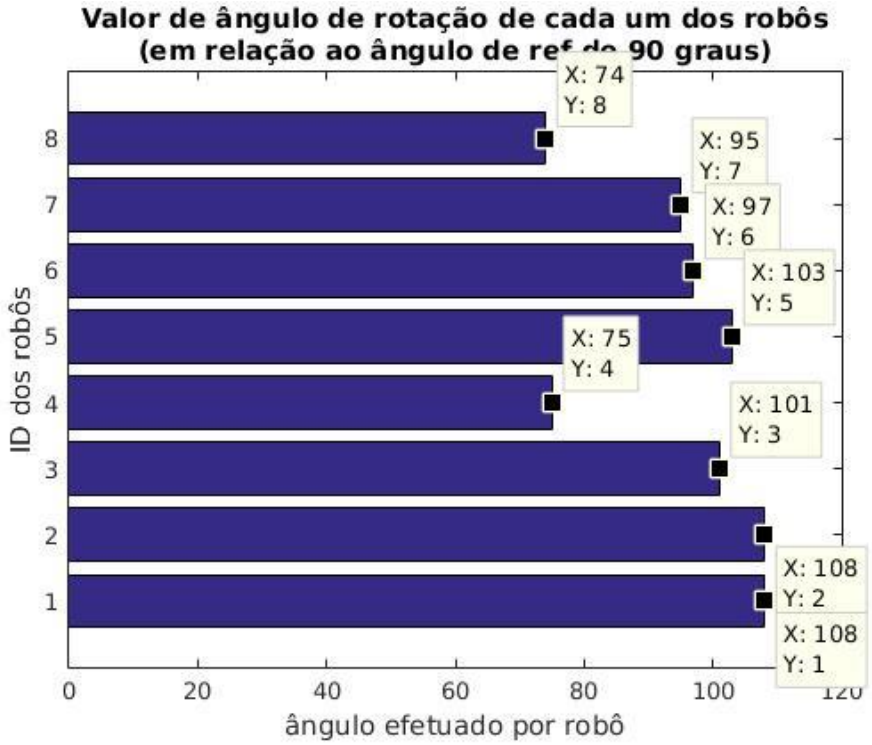
Figura 83 – Gráfico comparativo da distância percorrida (em cm) para 1024 pulsos do encoder.



Fonte: Autor (2018).

Além do teste dos encoders para o deslocamento frontal (translação), foi realizado o terceiro teste, para a rotação, como representado na figura 84.

Figura 84 – teste dos encoders realizados para um ângulo de referência de 90º.



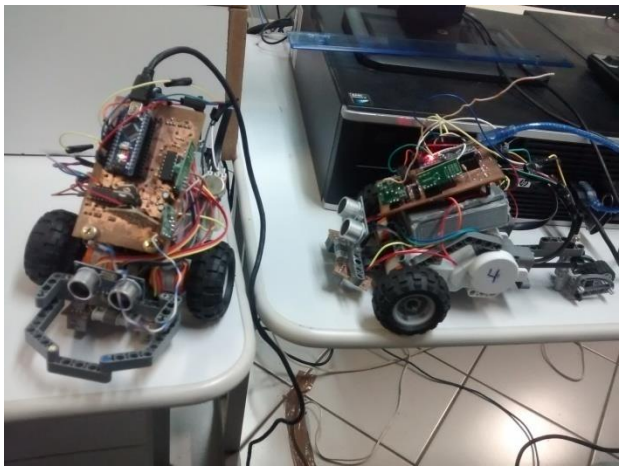
Fonte: Autor (2018).

6.1.5 Comunicação

Para a comunicação dos robôs, por serem do tipo unidirecional (simplex), foi realizado o teste de perda de pacotes. Os testes foram realizados utilizando 2 robôs, um como transmissor e outro como receptor.

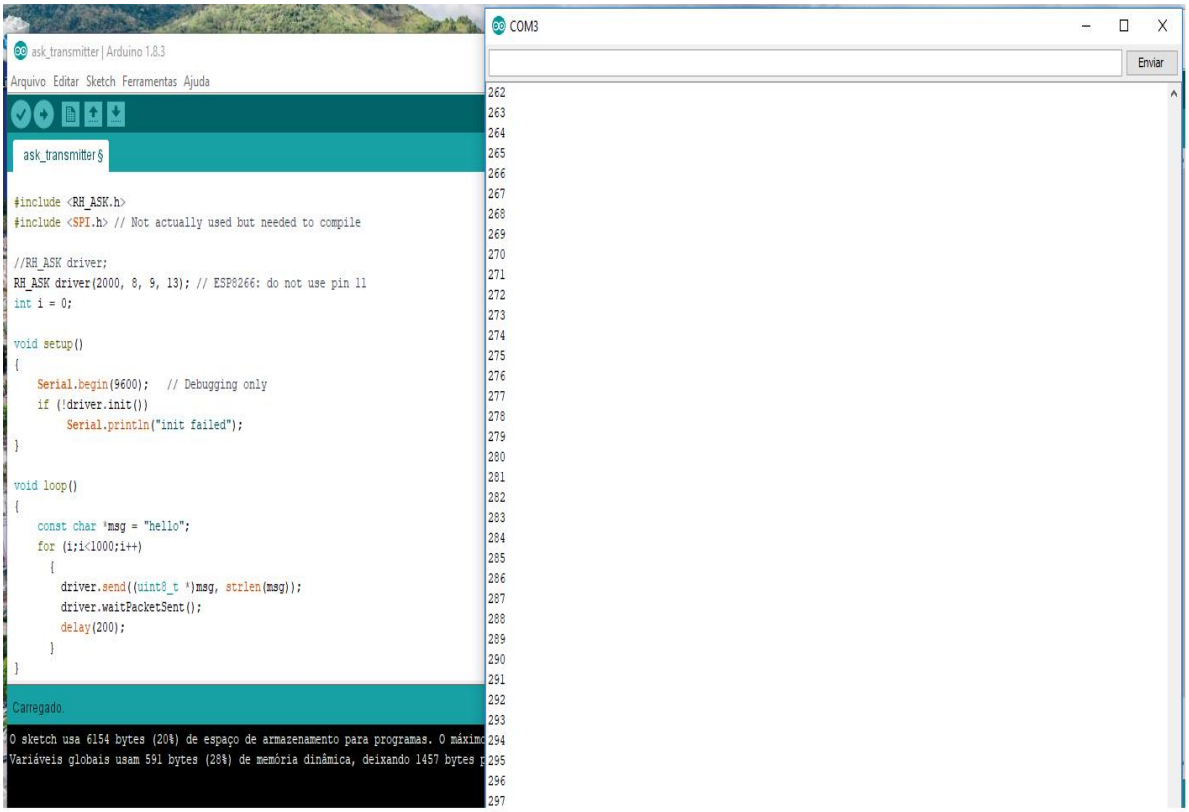
Os testes foram realizados para e seguintes quantidade de pacotes, na seguinte ordem: 100, 200, 500 e 1000 pacotes. Foi implementado no programa do receptor um contador, cujo valor é incrementado para cada pacote recebido. Ao fim do teste, para cada uma da quantidade de pacotes enviados, são comparados quantos os pacotes foram captados pelo receptor. Os testes foram realizados com uma distância de 20 cm entre emissor e receptor, com período de transmissão de pacotes a cada 200 ms (milissegundos), na figura 85. Na figura 86, tem-se o print do Arduino IDE para o teste do receptor.

Figura 85 – Robôs utilizados para o teste de comunicação (à esquerda. o receptor, à direita, o transmissor).



Fonte: Autor (2018).

Figura 86 – Print do software Arduino Nano, ilustrando o monitoramento de pacotes recebidos.



Fonte: Autor (2018).

Tabela 3 – Comparativo entre os pacotes transmitidos e recebidos (comunicação unidirecional).

Pacotes transmitidos	Pacotes recebidos
100	92
200	188
500	470
1000	957

Fonte: Autor (2018).

Quanto à perda de pacotes, sob as condições de teste realizadas, chegam a média de 6%.

6.1.6 Teste de Consumo dos Protótipos Desenvolvidos

Foi realizada a medição de corrente de consumo dos protótipos. Foram realizados 2 testes, o primeiro com os motores desligados, o segundo teste foi realizado com os motores acionados. O teste leva em consideração somente em consideração a corrente em regime permanente, desconsiderando a corrente de partida dos motores, como pode ser visualizado na tabela 4.

Tabela 4 – Testes realizados para uma tensão de alimentação de 8 V.

Condição de teste	Corrente (mA)
Motores desligados	130
Motores acionados	240

Fonte: Autor (2018).

Na tabela 5 são ilustrados o tempo de autonomia dos protótipos, conforme o modelo de célula de lítio utilizada nas baterias. O tempo de autonomia está intimamente relacionado à carga das células.

Tabela 5 – Tempo de autonomia dos protótipos em função da célula de lítio utilizada

Modelo Célula	Carga (mAh)	Tempo de autonomia (horas)
CGR17670A	1500	6 horas e 15 minutos
INR18650	2500	10 horas e 25 minutos
UR18650A	2250	9 horas e 22 minutos

Fonte: Autor (2018).

6.2 Custos dos Protótipos

A tabela 6 ilustra os custos do desenvolvimento dos protótipos:

Tabela 6 – Custo dos protótipos (Valor típico por protótipo).

Componente Robôs	Custo (R$)
1 Arduino Nano	54,15
1 PIC 16F716	6,51
1 Fotorreceptor	0,89
1 LED IR	0,89
2 motores LEGO	200,00
1 CI LM 35	8,90
2 CI's 7805	2,22
1 CI 7808	1,11
2 CI LM324	1,88
1 CI L293B	17,82
2 MOSFET's IRF640	7,18
1 Cristal oscilador	1,01
2 Capacitores 33 pF	0,10
2 Capacitores 100 uF	0,42

8 diodos 1N4148	0,40
1 sensor ultrassônico	14,28
1 Módulo RF (transmissor e receptor)	15,00
1 Conector P4 fêmea	1,90
Outros (fios e conectores)	50,00
Custo Total	384,66

Fonte: Autor (2018).

6.3 Comparativo Quanto aos Robôs LEGO NXT.

Na tabela 7, é possível visualizar as funcionalidades dos protótipos desenvolvidos, bem como as funcionalidades dos robôs LEGO NXT, já previamente utilizados no LPR.

Tabela 7 – Comparativo quanto às funcionalidades entre os robôs LEGO Mindstorms e os protótipos desenvolvidos utilizando Arduino Nano.

	LEGO	Arduino Nano
Sensores ópticos	2	2
Motores	2	2
Sensor ultrassônico	1	1
CPU	32-bit	8-bit
Frequência de Comunicação (MHz)	2400	433
Nº de robôs capazes de se conectarem à rede	3 (grupo)	Ilimitado* (raio de alcance)
Tecnologia de comunicação	Bluetooth	Modulação ASK *(amplitude shift keying)*

Permite troca de mensagem entre os robôs	Não*	Sim
Controle de velocidade dos motores	Sim	Não*
Odometria	Sim	Sim
Compatibilidade com componentes Não-LEGO	Não	Sim
Modularidade	Sim	Sim
Compatibilidade com Windows	Sim	Sim
Compatibilidade com Mac	Sim	Sim
Compatibilidade com Linux	Não	Sim
CPU expansível	Não	Sim
Resolução do encoder	1º por pulso	2º por pulso
Pinos de E/S	4 entradas 3 saídas	8 entradas analógicas 14 portas digitais (E/S)
Número de portas disponíveis para expansão	-	4 entradas analógicas 6 portas digitais

Fonte: Autor (2018).

6.4 Problemas Com os Protótipos Desenvolvidos

Serão abordados alguns problemas referentes à montagem dos protótipos:

6.4.1 Sensores Ópticos

Foram montados 16 sensores, cada protótipo utiliza 2 sensores. Para o total de 8 protótipos (Arduino) desenvolvidos, os seguintes apresentaram problemas com os sensores ópticos:

Protótipo 3 -> Sensor direito não funciona;
Protótipo 4 -> Sensor direito com pouca sensibilidade;
Protótipo 7 -> Sensor esquerdo com baixo valor de leitura.

OBS: Nem todos os sensores precisaram ter os sinais de saída amplificados pelos AMPOP's. Alguns possuem tensão alta o suficiente para serem ligados diretamente à entrada analógica do Arduino.

6.4.2 CPU

A PCB da CPU, bem como todas as PCB's dos demais circuitos dos robôs, foi desenvolvida no próprio LMM, utilizando-se a fresa disponível no mesmo.

O problema comum a todas as PCB's das CPU's foi o fato de haver curto-circuito em algumas trilhas das mesmas. Foi solucionado mediante retrabalho manual e alteração de conexões em pinos referentes aos sensores (ópticos, ultrassônico e circuito de odometria) e aos motores. A tabela 8 ilustra o esquema de configuração de pinos dos protótipos.

Tabela 8 – Esquema de conexão de pinos do Arduino aos sensores, motores e módulo de RF.

Protótipo 1	PINO	**Protótipo 2**	PINO	**Protótipo 3**	PINO	**Protótipo 4**	PINO
Sensor óptico1 (esquerdo)	A0		A0		A2		A2
Sensor óptico 2 (direito)	A1		A1		A3		A3
Sensor ultrassônico (ECHO)	D6		D6		D12		D6
Sensor ultrassônico (TRIGGER)	D7		D12		D7		D7
Motor 1_A (esquerdo)	D0		D0		D0		D0
Motor 1_B (esquerdo)	D1		D1		D1		D1
Motor 2_A (direito)	D2		D2		D2		D2
Motor 2_B (direito)	D3		D3		D3		D3
Receptor RF	D8		D8		D8		D8
Transmissor RF	D9		D9		D9		D9

Odometria motor esquerdo	A4		A5		A5		A5
Odometria motor direito	A6		A7		A7		A7
Protótipo 5	PINO	**Protótipo 6**	PINO	**Protótipo 7**	PINO	**Protótipo 8**	PINO
Sensor óptico1	A0		A2		A2		A2
Sensor óptico2	A1		A3		A3		A3
Sensor ultrassônico (ECHO)	D6		D6		D13		D12
Sensor ultrassônico (TRIGGER)	D7		D7		D12		D7
Motor 1_A (esquerdo)	D0		D0		D0		D0
Motor 1_B (esquerdo)	D1		D1		D1		D1
Motor 2_A (direito)	D2		D10		D10		D2
Motor 2_B (direito)	D3		D11		D11		D3
Receptor RF	D8		D8		D8		D8

Transmissor RF	D9		D9		D9		D9
Odometria motor esquerdo	A5		A5		A5		A5
Odometria motor direito	A7		A7		A7		A7

Fonte: Autor (2018).

Em relação à tabela 7, os pinos nomeados ‘A’ são configurados como entrada analógica e os pinos ‘D’ (à exceção dos pinos dos protótipos referentes ao ECHO do sensor ultrassônico) são todos configurados como saída digital.

6.4.3 Odometria

A odometria funcionou para todos os protótipos testados. O único inconveniente foi ao protótipo 3, em que a odometria somente funciona com o cabo USB do PC conectado a ele.

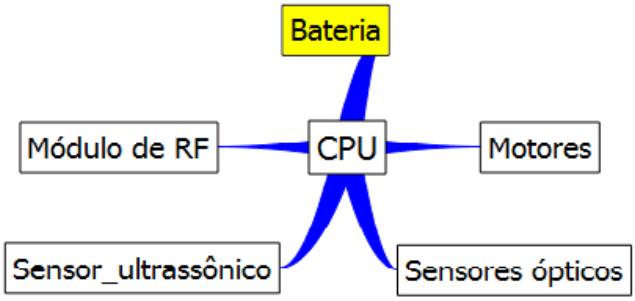
6.4.4 Monitor de bateria

De todos os problemas com os protótipos, o mais considerável foi o circuito de monitoramento de bateria. A priori, o circuito monitora as células de lítio de forma satisfatória, bem como alimenta os protótipos.

O problema do monitor de bateria está relacionado ao acionamento dos motores LEGO NXT, os protótipos frequentemente sofriam desligamento (CPU) ou o Arduino da CPU ficava em condição de oscilação (liga-desliga).

Para contornar esse problema, a arquitetura dos protótipos foi refeita por meio da remoção do monitor de bateria. A alimentação da CPU faz-se diretamente a partir da bateria, como se pode ver na figura 87.

Figura 87 – Nova arquitetura dos protótipos.



Fonte: Autor (2018).

6.4.5 Comunicação entre os protótipos.

Nos testes de comunicação, alguns protótipos (3, 4 e 6) não recebiam mais pacotes quando os motores eram acionados (alta corrente de partida). Para contornar esse problema, foi realizada a modificação no código dos transmissores e receptores, de modo que o tempo de transmissão de mensagem seja maior do que o tempo de uma operação antes da parada do receptor, a fim de receber a próxima mensagem.

6.4.6 Bateria

Algumas baterias estiveram com problema de mau contato, fazendo com que alguns protótipos sofressem desligamento.

O problema foi solucionando mediante retrabalho na soldagem dos fios conectados às células de lítio e limpeza das mesmas.

6.5 Análise dos resultados

Os testes realizados com os sensores ópticos provaram que os mesmos são mais sensíveis à iluminação ambiente (perturbação) que os sensores ópticos LEGO, ou seja, a calibração dos sensores deve ser refeita, sempre que houver alteração da luminosidade do ambiente.

Quanto aos sensores ultrassônicos, os protótipos demonstraram um erro sistemático de 1 cm acima do valor teórico (referência), o erro pode ser facilmente compensado, através da programação do Arduino. No geral, o sensor óptico HC-SR04 provou-se ter maior sensibilidade, aliada a um menor erro de medição, comparado ao sensor LEGO.

Em relação à odometria, nem todos os protótipos possuem mesma exatidão em relação ao número de pulsos do encoder, por exemplo, o robô 1 percorreu 88 cm, contra 100 cm percorridos pelo robô 8 (dado 100 cm como distância de referência). O mesmo ocorreu para a rotação, para uma referência de 90 graus, o robô 1 rotacionou 108 graus e o robô 8, 74 graus (erros de estimação de 20% e 17,7%, respectivamente).

Um grande inconveniente da odometria está no fato de que o erro de estimação aumenta conforme o aumento da distância a ser percorrida, o erro é cumulativo, o comportamento dos sensores de odometria assemelha-se a integradores. Em ambientes nos quais há um coeficiente de atrito razoavelmente baixo entre a superfície de contato e as rodas dos robôs, em que as mesmas podem “patinar”, o desempenho em estimação de distância percorrida fica ainda pior, uma vez que a odometria estima a distância percorrida (deslocamento), somente em função da rotação do motor.

Nos testes de comunicação, a perda média de pacotes foi de 6%. Em situações extremas, podem chegar a 20% (quando há obstáculos e fontes de interferência). A tecnologia empregada pelos módulos, modulação ASK (modulação em amplitude por chaveamento), por se tratar de uma tecnologia antiga (a primeira técnica desenvolvida para transmissão de sinais digitais), é pouco imune a ruídos. No entanto, foram utilizados os módulos de RF devido ao baixo custo e simples utilização. Devido à arquitetura modular dos protótipos, e por possuírem conectores de expansão, pode-se trocar por módulos Zigbee/Xbee, por exemplo.

Quanto aos custos dos protótipos, o custo médio dos mesmos foi de R$ 384,66 (por protótipo), levando-se em conta somente os custos dos componentes eletrônicos utilizados e dos 2 motores LEGO NXT (considerando-se a compra dos motores separadamente), uma vez que a estrutura mecânica foi feita a partir de peças de LEGO já disponíveis no LPR.

7 CONSIDERAÇÕES FINAIS E PERSPECTIVAS

Quanto ao trabalho desenvolvido, procurou-se uma alternativa de baixo custo, utilizando componentes facilmente acessíveis no mercado. Não foram utilizados componentes SMD (dispositivos montados em superfície), de forma que a soldagem possa ser feita somente com um ferro de solda comum.

Quanto às funcionalidades dos protótipos alternativos aos kits LEGO “Mindstorms”, existem apenas duas diferenças significativas. A primeira é o fato de que os protótipos desenvolvidos dão suporte à cooperação. A segunda, a questão do controle de velocidade dos motores. Enquanto o LEGO NXT permite o controle (suponha-se por PWM), nos protótipos desenvolvidos com o Arduino, essa funcionalidade não está explícita (devido aos pinos utilizados no acionamento dos motores), no entanto, pode-se implementar via software um controle de velocidade (um PWM emulado via software, nenhum teste prático foi realizado, mas é possível de ser implementado).

Quanto à utilização dos protótipos alternativos, seu desempenho ficou satisfatório. Talvez os 2 maiores inconvenientes tenham sido problemas provenientes de mau-contato em fios, conectores e quanto ao carregamento das baterias, não se tornou muito prático.

Ainda, em relação ao carregador de bateria, por ser do tipo tensão constante, não é recomendado para carregamento de células de lítio. Quando os projeto dos protótipos começaram a ser desenvolvidos, o autor não havia encontrado carregadores à venda para 2 células de lítio (para notebooks). No entanto, já existe à venda tal carregador, como por exemplo, no site Mercado Livre.

Houve problemas quanto ao fresamento das PCB’s, principalmente nas placas que compõe a CPU (do tipo dupla camada), algumas placas ficaram com problemas de curto-circuito, outras em que a fresa não completava o corte na superfície de cobre, necessitando nos dois casos anteriormente citados, retrabalho manual.

Toda a arquitetura dos robôs é modular, permitindo que placas sejam substituídas em caso de quaisquer problemas, mediante desconexões e re-conexões nos fios.

Ainda, quando a parte se sensoriamento, os protótipos não utilizam INS (sistemas de navegação inercial, tal como os acelerômetros e giroscópios), mas podem ser instalados aos mesmos.

Todas as atividades do PFC desenvolvidas até aqui possuem enorme potencial de estudo, tanto em análise de robôs móveis quanto em novas estratégias de cooperação.

Quanto ao projeto dos protótipos, todos os arquivos relacionados ao projeto das PCB's, os diagramas de circuito, os cases de bateria e dos sensores ópticos, foram encaminhados ao LMM (Laboratório de Montagem Mecatrônica), para que qualquer estudante do DAS (Departamento de Automação e Sistemas da Universidade Federal de Santa Catarina) possa replicá-los e até mesmo aperfeiçoá-los.

Quanto à estratégia de cooperação, seria abordada uma segunda estratégia, utilizando o algoritmo de consenso para eleição de líder. Não foi implementada devido a problemas em relação aos protótipos. Por questões de cronograma, somente uma pôde ser implementada (controle de formação).

Para os protótipos desenvolvidos, um manual de usuário para cada um dos protótipos está em andamento, sendo disponibilizado ao LPR. Nele, estarão incluídos as bibliotecas que devem ser importadas ao Arduino IDE, as instruções de recarga de baterias, instalação de placas, motores e sensores, bem como configurar os pinos de E/S e os módulos de RF.

Pode-se dizer que todos os objetivos (gerais e específicos) foram cumpridos, uma vez que:

- Os robôs estão em condições de utilização;

- Os protótipos são de baixo custo;

- A curva de aprendizado é pequena, pois os protótipos são baseados na mesma plataforma utilizada pelos estudantes no LPR, ou seja, Arduino. Os estudantes têm um relativo grau de familiarização com o mesmo;

- A arquitetura dos protótipos, tal como os robôs LEGO NXT, é modular e escalável;

- A interface de uso é amigável, pois o software utilizado no LPR (Arduino IDE) é baseado em linguagem C/C++, a mesma linguagem utilizada como um primeiro contato à programação, no curso de Engenharia de Controle e Automação da UFSC;

- A estratégia de cooperação foi implementada (controle de formação). Apesar da comunicação entre os protótipos ter sido realizada de forma unidirecional, por problemas de comunicação (coordenação e retransmissão de mensagens), a cooperação foi satisfatória, uma vez que o controle de formação fez com que o grupo de robôs convergisse para a posição desejada, mantendo-se a coesão do grupo.

É um grande desejo, para não dizer alegria, por parte do autor, que esse projeto não seja um fim em si mesmo, mas um novo começo.

REFERÊNCIAS

ABREU, Luiz Carlos de et. al. **A epistemologia genética de Piaget e o construtivismo.** Disponível em: <http://pepsic.bvsalud.org/scielo. php?script= sci_arttext&pid=S0104-12822010000200018>. Acesso em: 26 nov. 2018.

ALBUQUERQUE, L. A.; ÁLVARES, A. J. Uma Análise Sobre a Integração de Robôs Móveis Na Automação De Manufatura Industrial, e o Seu Papel Em Sistemas de Manufatura Flexível. In: Congresso Brasileiro de Engenharia de Fabricação, 6., 2011, Caxias do Sul. **Artigo**... Brasília: UNB, 2011. p. 455-468. Disponível em: <http://alvarestech.com/temp/cobef2011/grima.ufsc.br/cobef2011/media/trabalhos/COF11-0346.pdf>. Acesso em: 03 nov. 2018.

ALCOBA, M. S.; CALLEJAS, M. G.; PAZ, L. E. Análisis de Comportamientos de Vehículos de Braitenberg para la búsqueda robótica usando El Robot LEGO EV3. **Fides Et Ratio**, La Paz, v. 12, n. 12, p. 155-166, 2016. Disponível em: <http://www.scielo.org.bo/pdf/rfer/v12n12/v12n12_a09.pdf> Acesso em: 04 nov. 2018.

BARNES, D.; Gray, J. Behaviour synthesis for cooperant mobile robot control. In: International Conference on Control, Edinburgh, UK. **Article**. University of Salford. pp. 1135–1140, 1991. Disponível em: <https://www.researchgate.net/publication/239021156_Behaviour_synthesis_for_cooperant_mobile_robot_control>. Acesso em: 22 nov. 2018.

BOYLESTAD, R.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos.** 3. ed. Rio de Janeiro: Prentice-Hall do Brasil, 1986. 704p.

CAO, Y. UNY.; FUKUNAGA, A. S.; KAHNG, A. B. **Cooperative Mobile Robotics: Antecedents and Directions.** Disponível em: <http://metahack.org/cooperative-robots-survey-journal.pdf>. Acesso em: 22 nov. 2018.

CARVALHO, Sidney Roberto Dias de. **Controle de topologia em redes de robôs móveis cooperativos utilizando consenso.** 2015. 124 p. Dissertação (Mestrado) - Universidade Federal de Santa Catarina, Centro Tecnológico, Programa de Pós-

Graduação em Engenharia de Automação e Sistemas, Florianópolis, 2015. Disponível em: <http://www.bu.ufsc.br/teses/PEAS0195-D.pdf>. Acesso em: 11 out. 2018.

DORIGO et. al. **The SWARM-BOTS Project**. Disponível em: <https://www.researchgate.net/publication/221116556_The_SWARM-BOTS_project>. Acesso em: 17 nov. 2018.

Farinelli, A.; Iocchi, L.; Nardi, D. **Multi-Robot Systems: A classification focused on coordination.** Disponível em: <http://profs.scienze.univr.it/~farinelli/pubs/farinelli-TSMC-04.pdf> Acesso em: 22 nov. 2018.

FERBER, J. **Multi-Agent Systems.** 1. ed. Harlow: Addison Wesley, 1999. 509 p.

GOUVÊA, J. A. **CONTROLE DE FORMAÇÂO DE ROBÔS NÃO-HOLONÔMICOS COM RESTRIÇÃO DE CURVATURA UTILIZANDO FUNÇÃO POTENCIAL.** Tese (Doutorado em Engenharia Elétrica). Universidade Federal do Rio de Janeiro, INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE), Programa de Pós-Graduação em Engenharia Elétrica, Rio de Janeiro. 2011. 124p. Disponível em:<http://www.pee.ufrj.br/index.php/pt/producao-academica/teses-de-doutorado/2011/2011093002-2011093002/file>. Acesso em: 27 nov. 2018.

Group, The LEGO. *NXT User Guide*, 2006. Disponível em: <https://www.generationrobots.com/media/Lego-Mindstorms-NXT-Education-Kit.pdf>. Acesso em: 2 nov. 2018.

KHAMIS, A.; ELGINDY, A. **Minefield Mapping Using Cooperative Multirobot Systems.** Disponível em: <https://www.hindawi.com/journals/jr/2012/698046/>. Acesso em: 19 nov. 2018.

KILOBOT. **k-team**, [s. d.]. Disponível em: <http://www.k-team.com/mobile-robotics-products/kilobot#manual>. Acesso em: 17 nov. 2018.

LATHI, B.P. **Sinais e Sistemas Lineares.** 2. ed. Porto Alegre: Bookman, 2007. vii, 856 p. ISBN 9788560031139.

LIEKNA, A.; GRUNDSPENKIS, A. **Towards Practical Applications of Swarm Robotics: Overview of Swarm Tasks.** Disponível em: <http://tf.llu.lv/conference/proceedings2014/Papers/46_Liekna_A.pdf >. Acesso em: 19 nov. 2018.

LOH, C. C.; TRAECHTLER, A. Cooperative transportation of a load using nonholonomic mobile robots. In: International Symposium on Robotics and Intelligent Sensorsc (IRIS), 41., 2012, Paderborn, Germany. **Proceedings...** Paderborn: University of Paderborn, Control Engineering and Mechatronics, Heinz Nixdorf Institute, 2013. Paper, p 860-866. Disponível em: <https://ac.els-cdn.com/S1877705812026550/1-s2.0-S1877705812026550-main.pdf?_tid=5a7d7b64-e2d8-4dd6b7db3957721e5933&acdnat=1540876142_a6cec5a8b9d69fe383220048dcdc046e>. Acesso em: 18 nov. 2018.

MAKRIS et. al. **Cooperative Robots for Reconfigurable Assembly Operations: Review and Challenges.** Disponível em: < https://www.sciencedirect.com/science/article/pii/S2212827112002326 >. Acesso em: 19 nov. 2018.

MARZAT J.; PIET-LAHANIER H.; KAHN A. Cooperative guidance of LEGO Mindstorms NXT mobile robots. INTERNATIONAL CONFERENCE ON INFORMATICS IN CONTROL, AUTOMATION AND ROBOTICS, 11,. 2014, Vienne, Austria. 2, **Paper**. pp. 605-610, 2014. Disponível em: <https://hal.archives-ouvertes.fr/hal-01064027/file/2014_-_ICINCO_-_Cooperative_guidance_of_LEGO_Mindstorms_NXT_mobile_robots.pdf> Acesso em: 07 nov. 2018.

M. Mataric. **Interaction and Intelligent Behavior**. Disponível em:<https://dspace.mit.edu/bitstream/handle/1721.1/7343/AITR1495.pdf?sequence=2>. Acesso em: 22 nov. 2018.

PARKER, L. E. Current research in multirobot systems. **Artificial Life and Robotics**, Knoxville, v. 7, n. 1, p. 1-5, 2003. Disponível em:

<https://www.researchgate.net/publication/225383137_Current_research_in_multirobot_systems>. Acesso em: 04 nov. 2018.

PIO, J. L. S.; CASTRO, T. H. C.; CASTRO JÚNIOR, A. N. A Robótica Móvel como Instrumento de Apoio à Aprendizagem de Computação. In: Simpósio Brasileiro de Informática e Educação (SBIE), 15., 2006, Brasília. **Anais…** Manaus: UFAM, 1996. p. 498-506.
Disponível em: <http://www.brie.org/pub/index.php/sbie/article/view/510/496>. Acesso em: 3 de nov. 2018.

PREMVUTI, S.; YUTA, S. Consideration on the cooperation
of multiple autonomous mobile robots. In: IEEE/RSJ IROS, Ibaraki, Japan. Paper. pp. 59–63, 1990. Tsukuba University, Japan.
Disponível em: <https://ieeexplore.ieee.org/document/262369>. Acesso em: 21 nov. 2018.

RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência artificial.** Rio de Janeiro: Elsevier, Campus, 2004. 1021 p. ISBN 8535211772.

SATO, Elton M.; SANTOS, Leandro A.; ALMEIDA, Nicholas. **Hivebots: interação entre robôs para execução de tarefas.** Curitiba: UTFPR, 2015. 15 p. Disponível em: <http://paginapessoal.utfpr.edu.br/gustavobborba/if66j-s71-projetos/files/IF66J-15b_RT_Hivebots.pf>. Acesso em: 3 nov. 2018.

SIEGWART, R.; NOURBAKHSH, I.R.; SCARAMLUZZA, D. **Introduction to Autonomous Mobile Robots.** 2. ed. Cambridge, Massachusetts: The MIT Press, 2011. 445 p.

TAVARES, M. R. N.; AGOSTINI. A. E. B.; RIGO. L. A. In: SEMINÁRIO NACIONAL DE INCLUSÃO DIGITAL, 5, 2018, Passo Fundo. **Anais...** Porto Alegre: Núcleo de Tecnologia Educacional de Porto Alegre – Departamento Pedagógico da Secretaria Estadual de Educação do Rio Grande do Sul (NTE/DP/SEDUC-RS), 2018, 10 p.

Disponível em: <https://www.upf.br/_uploads/Conteudo/senid/2018-artigos-completos/177831.pdf>. Acesso em: 12 nov. 2018.

APÊNDICE A – PROGRAMA PARA TESTE DA COMUNICAÇÃO VIA RF RECEPTOR (ARDUINO)

```
//Programa de teste do receptor de RF.
//Quando o receptor recebe um dado caractere (comando), a partir do transmissor,
//o mesmo realizará as seguintes tarefas:

// '0' -> receptor permanecerá parado.
// '1' -> irá para frente.
// '2' -> virará à esquerda.
// '3' -> virará à direita.

//Autor: Volnei Fontana Junior.
//Data: 18/05/2018.

#include <RH_ASK.h>
#include <SPI.h> // Not actualy used but needed to compile

//RH_ASK driver;
RH_ASK driver(1000, 8, 9, 13); // ESP8266: do not use pin 11

const int Motor1_A = 2;
const int Motor1_B = 3;
const int Motor2_A = 4;
const int Motor2_B = 5;

int flag_OK = 0;

//Mensagem "OK" que o robô "seguidor" enviará para o "mestre" para receber mensagens de
//comando.
//Para cada mensagem "OK" enviada pelo seguidor, o "mestre" uma mensagem de comando.
int tempo = 0;

void setup()
{
Serial.begin(9600);      // Debugging only
if (!driver.init())
Serial.println("init failed");
}
```

```
void loop()
{
uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
uint8_t buflen = sizeof(buf);

char *msg_OK = "OK";

if (!flag_OK) //se não enviou OK.
{
driver.send((uint8_t *)msg_OK, strlen(msg_OK));
driver.waitPacketSent();
flag_OK = 1; //Robô "seguidor" já enviou mensagem OK! Não enviará de novo.
buf[RH_ASK_MAX_MESSAGE_LEN] = "\0"; //buffer do "seguidor deve ser limpo".
}

if ((flag_OK) && (driver.recv(buf, &buflen)))  //Se já enviou mensagem de "OK" e recebeu mensagem
                        //do "mestre".
{
if ((buf[0] == '0')) //se valor for zero, robô fica parado!!!
{
parado();
flag_OK = 0;
Serial.print(buf[0]);
}

if ((buf[0] == '1')) //Se valor for 1, robô vai para frente!!!
{
frente();
flag_OK = 0;
Serial.print(buf[0]);
}

if ((buf[0] == '2')) //Se valor for 1, robô vai para frente!!!
{
esquerda();
flag_OK = 0;
Serial.print(buf[0]);
}
```

```
if ((buf[0] == '3')) //Se valor for 1, robô vai para frente!!!

{
direita();
flag_OK = 0;
Serial.print(buf[0]);
}
}
}

void frente()

{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, HIGH);
digitalWrite(Motor2_B, LOW);
}

void esquerda()
{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, HIGH);
digitalWrite(Motor2_A, HIGH);
digitalWrite(Motor2_B, LOW);
}

void direita()
{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, HIGH);
}

void parado()

{
digitalWrite(Motor1_A, LOW);
```

```
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, LOW);
}
```

APÊNDICE B – PROGRAMA PARA TESTE DA COMUNICAÇÃO VIA RF TRANSMISSOR (ARDUINO)

```
//Código teste para comunicação entre cada um robôs
//Código para o transmissor.

//O transmissor envia um caractere (mensagem) a cada 3 segundos.
//Cada caractere é um comando para o receptor.

//Comunicação do tipo simplex (unidirecional).

//Cada caractere captado pelo receptor, o mesmo realizará as

//seguintes operações:

// 0 -> Receptor vai para frente.
// 1 -> Receptor vira à esquerda.
// 2 -> Receptor vira à direita.
// 3 -> Receptor permanece parado.

//Autor: Volnei Fontana Junior
//Data: 18/05/18

#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile

RH_ASK driver(500,8,9,13);
// RH_ASK driver(2000, 2, 4, 5); // ESP8266: do not use pin 11

void setup()
{
Serial.begin(9600);
if (!driver.init())
Serial.println("init failed");
}

void loop()
{
char msg[2] = {'0'};
```

```
char msg2[2] = {'1'};
char msg3[2] = {'2'};
char msg4[2] = {'3'};

driver.send((uint8_t *)msg, strlen(msg));
driver.waitPacketSent();
delay(3000);

driver.send((uint8_t *)msg2, strlen(msg));
driver.waitPacketSent();
delay(3000);

driver.send((uint8_t *)msg3, strlen(msg));
driver.waitPacketSent();
delay(3000);

driver.send((uint8_t *)msg4, strlen(msg));
driver.waitPacketSent();
delay(3000);
}
```

APÊNDICE C – PROGRAMA PARA TESTE DOS ENCODERS (ARDUINO)

```
//Programa para teste dos encoders
//Autor: Volnei Fontana Junior
//Data: 03/10/2018

const int encoder1 = A4;  // Entrada analógica para o encoder do motor esquerdo
const int encoder2 = A6;  // Entrada analógica para o encoder do motor direito.

//RobÔ 1:
//Encoder1 = A3;
//Encoder2 = A5;

//Todos os demais utilizam A4 e A6!!!

const int Motor1_A = 2;
const int Motor1_B = 3;
const int Motor2_A = 10; //Robô  6 e 7 ->10 Para os denmais -> 4
const int Motor2_B = 11; //RobÔ 6 e 7 ->11  Para os demais -> 5

int sensorValue1;
int sensorValue2;

int pulsos1 = 0;
int pulsos2 = 0;
const float pi = 3.1416;
//const float r = 6.5;   //metade da distância entre uma roda e outra
        //(valor teórico, na prática,
        //teve de ser adaptado conforme o robô)!!!

//const float r = 8.0; //valor para o robô 5.
//const float r = 7;   //valor para o robô 1 e 3!!!
const float r = 6.0;   //metade da distância entre uma roda e outra,
        //todos os demais robôs!
float D_roda = 5.6; //Diâmetro da roda do LEGO (cm).
long int pulsos1INT = 0; //Variável pulsos1 e pulsos2 do tipo inteiro;
long int pulsos2INT = 0;
int conta1 = 0;    //Contadores dos pulsos dos encoders.
int conta2 = 0;
```

```
int flag1 = 0;
int flag2 = 0;
int sentido = 0;

void frente()

{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, HIGH);
digitalWrite(Motor2_B, LOW);
}

void parado()

{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, LOW);
}

void esquerda()

{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, HIGH);
digitalWrite(Motor2_A, HIGH); //Para o robô 4, fica desligado, para teste do encoder para rotação!!!
digitalWrite(Motor2_B, LOW);
}

void direita()

{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, HIGH);
}
```

```
void setup()
{
pinMode(Motor1_A, OUTPUT);
pinMode(Motor1_B, OUTPUT);
pinMode(Motor2_A, OUTPUT);
pinMode(Motor2_B, OUTPUT);
}

void giraGraus(int theta) //ângulo em graus theta

{
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if (theta >= 0)
{
sentido = 0; //Esquerda (anti-horário) -> ângulo positivo!
}
if (theta < 0)
{
theta = abs(theta);
sentido = 1;
}
pulsos1 = (float) r*theta /D_roda;
pulsos2 = (float) r*theta/D_roda;

pulsos1INT = (int) pulsos1;
pulsos2INT = (int) pulsos2;

while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT) && (sentido == 0))
{
esquerda();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
```

```
if (sensorValue1 <= 1000)
{
flag1 = 0;
}
if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
}
while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT) && (sentido == 1))
{
direita();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
if (sensorValue1 <= 1000)
{
flag1 = 0;
}

if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
}
```

```
pulsos1 = 0;
pulsos2 = 0;
conta1 = 0;
conta2 = 0;
parado();
}
void percorreXcm(int distancia) // funcção que recebe como parâmetro,
        // a distância que o robô deve seguir!!!
{
parado();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
pulsos1 = (float) distancia * 180 / (pi * D_roda);
pulsos2 = (float) distancia * 180  / (pi * D_roda);
pulsos1INT = (int) pulsos1;
pulsos2INT = (int) pulsos2;

while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT))
{
frente();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
if (sensorValue1 <= 1000)
{
flag1 = 0;
}
if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
```

```
}
}
pulsos1 = 0;
pulsos2 = 0;
conta1 = 0;
conta2 = 0;
parado();
}
void loop() {
percorreXcm(40); //Percorre 40 cm
giraGraus(90);   //gira 90 graus em sentido anti-horário.
}
```

APÊNDICE D – PROGRAMA PARA TESTE DOS MOTORES (ARDUINO)

```
//Programa simples, para testar os motores dos
//robôs desenvolvidos.
//Autor: Volnei Fontana Junior. Data 04/10/2018.

const int Motor1_A = 2;
const int Motor1_B = 3;
const int Motor2_A = 4; //Robô  6 e 7 ->10
const int Motor2_B = 5; //RobÔ 6 e 7 ->11

void frente() //Sinal enviado ao CI L293B é 1010

{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, HIGH);
digitalWrite(Motor2_B, LOW);
}

void parado() //Sinal enviado ao CI L293B é 0000

{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, LOW);
}

void esquerda() //Sinal enviado ao CI é 0110
        //Motor esquerdo dá a ré
        //Direito gira no sentido para a frente.
{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, HIGH);
digitalWrite(Motor2_A, HIGH);
digitalWrite(Motor2_B, LOW);
}
```

```
void direita()  //Sinal enviado ao CI é 1001
//Motor esquerdo para frente.
//Direito dá a ré.

{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, HIGH);
}

void re()     //Sinal enviado ao CI 0101.
              //Ambos os motores em situação de ré.
{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, HIGH);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, HIGH);
}

void setup() {
// put your setup code here, to run once:
pinMode(Motor1_A, OUTPUT);
pinMode(Motor1_B, OUTPUT);
pinMode(Motor2_A, OUTPUT);
pinMode(Motor2_B, OUTPUT);
}

void loop() {
frente();
delay(2000); //Aguarda 2 segundos.
esquerda();
delay(2000);
direita();
delay(2000);
re();
delay(2000);
parado();
delay(2000);}
```

APÊNDICE E – PROGRAMA PARA TESTE DOS SENSORES ÓPTICOS (ARDUINO)

```
//Programa para teste dos sensores ópticos.
//Realiza a leitura dos sensores, cuja saída de cada sensor está conectada
//à uma entrada analógica do Arduino.

//Feito isso, os valores de saída dos sensores são mostrados no monitor
//do próprio Arduino IDE, via comunicação serial.

//Autor: Volnei Fontana Junior.
//Data: 16/03/18.

const int sensor1 = A0; //Sensor esquerdo.
const int sensor2 = A1; //Sensor direito.

int sensorValue1 = 0;
int sensorValue2 = 0;

void setup() {
Serial.begin(9600); //Inicializa a comunicação serial a 9600 bps.
}

void loop() {
sensorValue1 = analogRead(sensor1);
sensorValue2 = analogRead(sensor2);
Serial.print("sensor1 = "); //Exibe os valores do sensor esquerdo e direito
Serial.print(sensorValue1); //através da comunicação serial.
Serial.print("sensor2 = ");
Serial.println(sensorValue2);
delay(200);                 //Delay de 200 ms.
}
```

APÊNDICE F – PROGRAMA PARA TESTE DO SENSOR ULTRASSÔNICO (ARDUINO)

```
//Programa para teste do sensor ultrassônico HC-SR04.
#define trigPin 7
#define echoPin 6
void setup()
{
Serial.begin (9600);
pinMode(trigPin, OUTPUT); //TRIGGER como saída digital.
pinMode(echoPin, INPUT);  //ECHO como entrada digital.
}
void loop()
{
long duration, distance;
digitalWrite(trigPin, LOW);         //Pino correspondente ao TRIGGER em nível lógico 0.
delayMicroseconds(2);
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;

if (distance >= 200 || distance <= 0)
{                                   //se distancia maior que 200 cm ou menor que 0 cm.
Serial.println("FORA DE ALCANCE");
}

else {
Serial.print(distance);
Serial.println(" cm");
}

delay(500);     //leitura feita a cada 0,5 segundos.
}
```

Apêndice G – CÓDIGO DO TRANSMISSOR DE CONTROLE DE FORMAÇÃO (ARDUINO)

```
/*Programa usando o robô 1 como transmissor
Comunicação centralizada, unidirecional, visando demonstrar a propriedade de coesão, em sistemas multi-Robôs.
RobÔ envia mensagem sobre a distância entre o mesmo e a meta (obstáculo).
Envio de mensagens por broadcast (1 para todos).
Mensagem recebida pelo receptor equivale a distância do emissor à meta.
Para haver coesão, deve-se ocorrer as 2 situações

Se distância do transmissor > distância do receptor
receptor parado!!!

Se distância do transmissor < distãncia do receptor
receptor frente!!!
*/

//Código do Robô 1!!!

//Autor: Volnei Fontana Junior.
//Data: 18/05/2018.

//Arduino Nano
//Pinos de saída analógica: 3,5,6,9 e 10.
#define trigPin 7
#define echoPin 6

//#define trigPin 12 //Setup robÔ 2
//#define echoPin 6 //robô 2


const int encoder1 = A4;  // Entrada analógica para o encoder do motor esquerdo
const int encoder2 = A6;  // Entrada analógica para o encoder do motor direito.

int sensorValue1;
int sensorValue2;

int pulsos1 = 0;
```

```
int pulsos2 = 0;
const float pi = 3.1416;
//const float r = 6.5;   //metade da distância entre uma roda e outra
//(valor teórico, na prática,
//teve de ser adaptado conforme o robô)!!!

//const float r = 8.0; //valor para o robô 5.
//const float r = 7;   //Valor para o robô 1 e 3!!!
const float r = 6.0;   //metade da distância entre uma roda e outra,
//todos os demais robôs!

float D_roda = 5.6; //Diâmetro da roda do LEGO (cm).

long int pulsos1INT = 0; //Variável pulsos1 e pulsos2 do tipo inteiro;
long int pulsos2INT = 0;
int conta1 = 0;    //Contadores dos pulsos dos encoders.
int conta2 = 0;
int flag1 = 0;
int flag2 = 0;
int sentido = 0;



#include <RH_ASK.h>
#include <SPI.h> // Not actualy used but needed to compile

//RH_ASK driver;
RH_ASK driver(500, 8, 9, 13); // ESP8266: do not use pin 11

const int Motor1_A = 2;
const int Motor1_B = 3;
const int Motor2_A = 4; //Robôs 6 e 7 -> 10 Demais -> 4
const int Motor2_B = 5; //Robôs 6 e 7 -> 11 Demais -> 5

int flag_msg = 0; //Mensagem "OK" que o robô "seguidor" enviará para o "mestre" para receber
mensagens de comando.
//Para cada mensagem "OK" enviada pelo seguidor, o "mestre" uma mensagem de "comando".

int tempo = 0;
```

```
void setup()
{

pinMode(Motor1_A, OUTPUT);          // sets the digital pin 13 as output
pinMode(Motor1_B, OUTPUT);
pinMode(Motor2_A, OUTPUT);
pinMode(Motor2_B, OUTPUT);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);  // Debugging only
if (!driver.init())
Serial.println("init failed");
}

void loop()
{
uint8_t buf[RH_ASK_MAX_MESSAGE_LEN];
uint8_t buflen = sizeof(buf);
uint8_t dist = 0;
char vRecebido[4]; // string dos seguidores para cada comando do mestre!!!
char msg1[4] = "\0";
char *msg; //ID ROBÔ 7!
msg = msg1;
buf[RH_ASK_MAX_MESSAGE_LEN];

/*Setup sensor ultrassÔNICO*/

long duration;
uint8_t distance;
digitalWrite(trigPin, LOW);  // Added this line
delayMicroseconds(2); // Added this line
digitalWrite(trigPin, HIGH);
//  delayMicroseconds(1000); - Removed this line
delayMicroseconds(10); // Added this line
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;

if (flag_msg == 0){
```

```
if (distance >= 6 ) {  // This is where the LED On/Off happens
/*   digitalWrite(led,HIGH); // When the Red condition is met, the Green LED should turn off
digitalWrite(led2,LOW);*/
frente();
delay(200);
parado();
flag_msg = 1;
}
else{

parado();
flag_msg = 1;
}
}
if (flag_msg)

{
distance = (duration/2) / 29.1;
itoa(distance,msg,10); // converte o inteiro da distância em string.
driver.send((uint8_t *)msg, strlen(msg));
driver.waitPacketSent();
Serial.println(msg);
delay(200);
flag_msg = 0;
}
parado();
}

void frente()

{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, HIGH);
digitalWrite(Motor2_B, LOW);

}

void esquerda()
```

```
{
digitalWrite(Motor1_A, LOW);
analogWrite(Motor1_B, 200);
digitalWrite(Motor1_B, HIGH);
digitalWrite(Motor2_A, LOW);
}

void direita()

{
digitalWrite(Motor1_A, HIGH);
analogWrite(Motor1_B, 50);
analogWrite(Motor1_B, 200);
digitalWrite(Motor2_A, LOW);
}

void parado()

{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, LOW);
}

void giraGraus(int theta) //ângulo em graus theta

{
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if (theta >= 0)
{
sentido = 0; //Esquerda (anti-horário) -> ângulo positivo!
}
if (theta < 0)
{
theta = abs(theta);
sentido = 1;
```

```
}
pulsos1 = (float) r*theta /D_roda;

pulsos2 = (float) r*theta/D_roda;

pulsos1INT = (int) pulsos1;
pulsos2INT = (int) pulsos2;

while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT) && (sentido == 0))
{
esquerda();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
if (sensorValue1 <= 1000)
{
flag1 = 0;
}

if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
}
while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT) && (sentido == 1))
{
direita();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
```

```
{
conta1++;
flag1 = 1;
}
if (sensorValue1 <= 1000)
{
flag1 = 0;
}

if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
}
pulsos1 = 0;
pulsos2 = 0;
conta1 = 0;
conta2 = 0;
parado();
}

void percorreXcm(int distancia) // funcção que recebe como parâmetro,
// a distância que o robô deve seguir!!!

{
parado();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);

pulsos1 = (float) distancia * 180 / (pi * D_roda);

pulsos2 = (float) distancia * 180  / (pi * D_roda);

pulsos1INT = (int) pulsos1;
```

```
pulsos2INT = (int) pulsos2;

while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT))
{
frente();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
if (sensorValue1 <= 1000)
{
flag1 = 0;
}

if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
}
pulsos1 = 0;
pulsos2 = 0;
conta1 = 0;
conta2 = 0;
parado();
}
```

APÊNDICE H – PROGRAMA DOS SEGUIDORES DE CONTROLE DE FORMAÇÃO (ARDUINO)

```
/*Programa usando N robôs como receptor.
Comunicação centralizada, unidirecional, visando demonstrar a propriedade de coesão, em sistemas multi-Robôs. Robô envia mensagem sobre a distância entre o mesmo e a meta (obstáculo).
Envio de mensagens por broadcast (1 para todos).
Mensagem recebida pelo receptor equivale a distância do emissor à meta.
Para haver coesão, deve-se ocorrer as 2 situações

Se distância do transmissor > distância do receptor
receptor parado!!!

Se distância do transmissor < distãncia do receptor
receptor frente!!!
*/

//modificado para o robô 7

//modificar pro robô2

//Autor: Volnei Fontana Junior.
//Data: 18/05/2018.

//Arduino Nano
//Pinos de saída analógica: 3,5,6,9 e 10.
#define trigPin 12 //Setup robÔ 2
#define echoPin 6 7///robô 2

//#define trigPin 12 //12 para 4
//#define echoPin 5 //13 para 5

const int encoder1 = A4;  // Entrada analógica para o encoder do motor esquerdo
const int encoder2 = A6;  // Entrada analógica para o encoder do motor direito.

int sensorValue1;
int sensorValue2;

int pulsos1 = 0;
```

```
int pulsos2 = 0;
const float pi = 3.1416;
//const float r = 6.5;   //metade da distância entre uma roda e outra
        //(valor teórico, na prática,
//teve de ser adaptado conforme o robô)!!!

//const float r = 8.0; //valor para o robô 5.
//const float r = 7;   //Valor para o robô 1 e 3!!!
const float r = 6.0;   //metade da distância entre uma roda e outra,
//todos os demais robôs!

float D_roda = 5.6; //Diâmetro da roda do LEGO (cm).

long int pulsos1INT = 0; //Variável pulsos1 e pulsos2 do tipo inteiro;
long int pulsos2INT = 0;
int conta1 = 0;    //Contadores dos pulsos dos encoders.
int conta2 = 0;
int flag1 = 0;
int flag2 = 0;

int sentido = 0;
int flag_msg = 0;

#include <RH_ASK.h>
#include <SPI.h> // Not actualy used but needed to compile

//RH_ASK driver;
RH_ASK driver(500, 8, 9, 13); // ESP8266: do not use pin 11

const int Motor1_A = 2;
const int Motor1_B = 3;
const int Motor2_A = 4; //Robôs 6 e 7 -> 10 Demais -> 4
const int Motor2_B = 5; //Robôs 6 e 7 -> 11 Demais -> 5

int tempo = 0;

int dist_recebida = 0;
long duration;
long distance;
```

```
void setup()
{

pinMode(Motor1_A, OUTPUT);          // sets the digital pin 13 as output
pinMode(Motor1_B, OUTPUT);
pinMode(Motor2_A, OUTPUT);
pinMode(Motor2_B, OUTPUT);
pinMode(trigPin, OUTPUT);
pinMode(echoPin, INPUT);
Serial.begin(9600);  // Debugging only
if (!driver.init())
Serial.println("init failed");
}

void loop()

{
uint8_t buf[4] = "\0";
uint8_t buflen = sizeof(buf);

if (driver.recv(buf, &buflen)) // Non-blocking
{
digitalWrite(trigPin, LOW);  // Added this line
delayMicroseconds(2); // Added this line
digitalWrite(trigPin, HIGH);
//  delayMicroseconds(1000); - Removed this line
delayMicroseconds(10); // Added this line
digitalWrite(trigPin, LOW);
duration = pulseIn(echoPin, HIGH);
distance = (duration/2) / 29.1;
Serial.println(distance);

dist_recebida = atoi((char*)buf);
Serial.println(dist_recebida);
if ((distance > dist_recebida) && (distance >= 6)) //Robô seguidor atrás do mestre!!!
{
percorreXcm(2);
// flag_msg = 0;
```

```
}
if ((distance <= dist_recebida) && (distance < 6)) // Robô seguidor à frente do mestre!!!
{
parado();
// flag_msg = 0;
}

}
}

void frente()

{
digitalWrite(Motor1_A, HIGH);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, HIGH);
digitalWrite(Motor2_B, LOW);

}

void esquerda()

{
digitalWrite(Motor1_A, LOW);
analogWrite(Motor1_B, 200);
digitalWrite(Motor1_B, HIGH);
digitalWrite(Motor2_A, LOW);
}

void direita()

{
digitalWrite(Motor1_A, HIGH);
analogWrite(Motor1_B, 50);
analogWrite(Motor1_B, 200);
digitalWrite(Motor2_A, LOW);
}

void parado()
```

```
{
digitalWrite(Motor1_A, LOW);
digitalWrite(Motor1_B, LOW);
digitalWrite(Motor2_A, LOW);
digitalWrite(Motor2_B, LOW);
}

void giraGraus(int theta) //ângulo em graus theta

{
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if (theta >= 0)
{
sentido = 0; //Esquerda (anti-horário) -> ângulo positivo!
}
if (theta < 0)
{
theta = abs(theta);
sentido = 1;
}
pulsos1 = (float) r*theta /D_roda;

pulsos2 = (float) r*theta/D_roda;

pulsos1INT = (int) pulsos1;
pulsos2INT = (int) pulsos2;

while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT) && (sentido == 0))
{
esquerda();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
```

```
if (sensorValue1 <= 1000)
{
flag1 = 0;
}

if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
}
while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT) && (sentido == 1))
{
direita();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
if (sensorValue1 <= 1000)
{
flag1 = 0;
}

if ((sensorValue2 > 1000) && (!flag2))
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
```

```
}
pulsos1 = 0;
pulsos2 = 0;
conta1 = 0;
conta2 = 0;
parado();
}

void percorreXcm(int distancia) // funcção que recebe como parâmetro,
// a distância que o robô deve seguir!!!

{
parado();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);

pulsos1 = (float) distancia * 180 / (pi * D_roda);

pulsos2 = (float) distancia * 180  / (pi * D_roda);

pulsos1INT = (int) pulsos1;
pulsos2INT = (int) pulsos2;

while ((conta1 < pulsos1INT) && (conta2 < pulsos2INT))
{
frente();
sensorValue1 = analogRead(encoder1);
sensorValue2 = analogRead(encoder2);
if ((sensorValue1 > 1000) && (!flag1))
{
conta1++;
flag1 = 1;
}
if (sensorValue1 <= 1000)
{
flag1 = 0;
}

if ((sensorValue2 > 1000) && (!flag2))
```

```
{
conta2++;
flag2 = 1;
}
if (sensorValue2 <= 1000)
{
flag2 = 0;
}
}
pulsos1 = 0;
pulsos2 = 0;
conta1 = 0;
conta2 = 0;
parado();
}
```

APÊNDICE I – PROGRAMA DO MONITOR DE BATERIA (PIC16F716)

```
#include <stdio.h>
#include <pic.h>
#include <htc.h>
#include <pic16f716.h>

#define TRUE 1
#define _XTAL_FREQ 4000000 //Cristal de 4 MHz.

#define PLACA RB0
#define LED_ALERTA RB2


void seta_PIC()

{
__CONFIG(0x3FB9);
TRISA = 0b00000111; //Pinos AN0, AN1 e AN2 como entradas.
TRISB = 0b00000000; //todos os pinos de PORTB como saídas.
PORTA = 0;
PORTB = 0;

GIE = 1; //habilita interrupção global.
PEIE = 1; //habilida interrupção dos periféricos.
T0IE = 0; //desabilita interrupção do timer 0.
INTE = 0; //desabilita interrupção de interrupção externa.
RBIE = 0; //desabilita interrupção do pino RB0.
TMR1IE = 0; //interrupção do Timer 1 desabilitada.
ADIE = 0; //interrupção do conversor A/D desabilitada.
ADCS0 = 0;
ADCS1 = 1;
ADON = 1; //Ativa o conversor A/D.
PCFG0 = 0;
PCFG1 = 0;
PCFG2 = 0;
//todos os flags zerados.
T0IF = 0;
INTF = 0;
```

```
RBIF = 0;
ADIF = 0;
ADRES = 0;
}

void seta_AN0()

{
CHS0 = 0;
CHS1 = 0;
CHS2 = 0;
}

void seta_AN1()

{
CHS0 = 1;
CHS1 = 0;
CHS2 = 0;
}

void seta_AN2()

{
CHS0 = 0;
CHS1 = 1;
CHS2 = 0;
}

int le_AN0()

{
int temp = 0;
GO = 1;
while(GO);
temp = ADRES;
return temp;
}
```

```
int le_AN1()

{
int v1 = 0;
GO = 1;
while(GO);
v1 = ADRES;
return v1;
}

int le_AN2()

{
int v2 = 0;
GO = 1;
while(GO);
v2 = ADRES;
return v2;
}

void piscaled()

{
LED_ALERTA = 0;
__delay_ms(250);
LED_ALERTA = 1;
__delay_ms(250);
}

int main()

{
int temp = 0, tensao_V1 = 0, tensao_V2 = 0;
seta_PIC();
PLACA = 1;
LED_ALERTA = 0;

while (TRUE)
{
```

```
seta_AN0();
__delay_ms(1);
temp = le_AN0();  //LÊ temperatura do sensor.
__delay_ms(5); //aguarda 10 milissegundos.
seta_AN1();
__delay_ms(1);
tensao_V1 = le_AN1(); //LÊ tensão da célula 1(2 células em série).
__delay_ms(5); //aguarda 10 milissegundos.
seta_AN2();
__delay_ms(1);
tensao_V2 = le_AN2(); //LÊ tensão da célula 2.
__delay_ms(5);

//Se tensão na célula 1 menor que 6 Volts ou tensão na célula 2 menor que 3 Volts, ou diferença de
//de tensão V1 - V2 maior ou igual a 3 Volts

if ((tensao_V1 <= 150) || (tensao_V2 <= 75) || ((tensao_V1 - tensao_V2) < 75))
{

if (temp >= 31)
{
PLACA = 0;  //Mosfet abre o cicuito da placa do robô.
piscaled();
}

if (temp <= 23)
{
PLACA = 0;  //Mosfet abre o cicuito da placa do robô.
LED_ALERTA = 1; //Sem tensão, o led só fica aceso.
}
}

//else
if ((tensao_V1 >= 175) && (tensao_V2 >= 100) && ((tensao_V1 - tensao_V2) >= 75))
{

if (temp <= 23) //Se temperatura menor ou igual a 45 °C.
{
PLACA = 1;    // Religa a placa e o carregador.
```

```
LED_ALERTA = 0;  //Sinal de OK.
}

if (temp >= 31)
{
PLACA = 0;    // Religa a placa e o carregador.
piscaled();
}

}
}

return 0;
}
```

ANEXO A – DADOS DO CI L293B

SGS-THOMSON MICROELECTRONICS

L293B
L293E

PUSH-PULL FOUR CHANNEL DRIVERS

- OUTPUT CURRENT 1A PER CHANNEL
- PEAK OUTPUT CURRENT 2A PER CHANNEL (non repetitive)
- INHIBIT FACILITY
- HIGH NOISE IMMUNITY
- SEPARATE LOGIC SUPPLY
- OVERTEMPERATURE PROTECTION

DIP16

ORDERING NUMBER : L293B

POWERDIP (16 + 2+ 2)

ORDERING NUMBER : L293E

DESCRIPTION

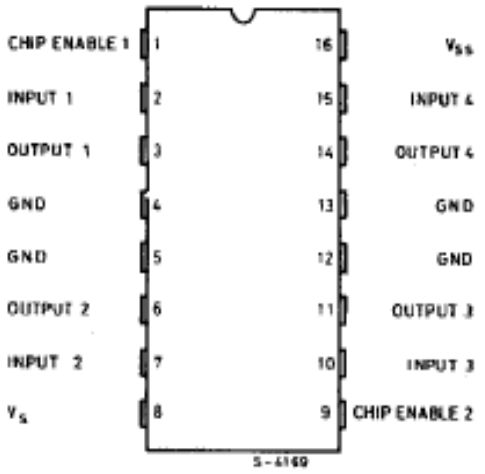
The L293B and L293E are quad push-pull drivers capable of delivering output currents to 1A per channel. Each channel is controlled by a TTL-compatible logic input and each pair of drivers (a full bridge) is equipped with an inhibit input which turns off all four transistors. A separate supply input is provided for the logic so that it may be run off a lower voltage to reduce dissipation.

Additionally, the L293E has external connection of sensing resistors, for switchmode control.

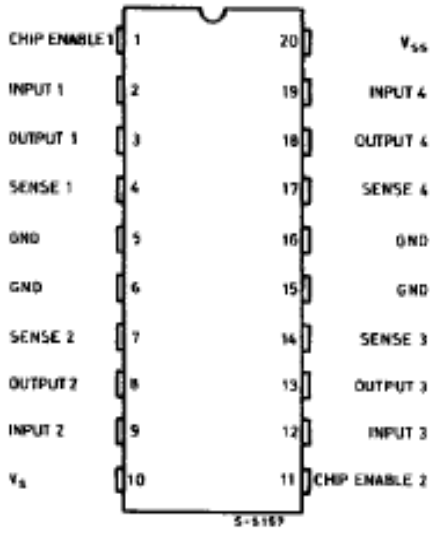
The L293B and L293E are package in 16 and 20-pin plastic DIPs respectively ; both use the four center pins to conduct heat to the printed circuit board.

PIN CONNECTIONS

DIP16 - L293B



POWERDIP (16+2+2) - L293E



ANEXO B – DADOS DO CI LM324N

Low power quad op amps

LM124/224/324/324A/ SA534/LM2902

DESCRIPTION

The LM124/SA534/LM2902 series consists of four independent, high-gain, internally frequency-compensated operational amplifiers designed specifically to operate from a single power supply over a wide range of voltages.

UNIQUE FEATURES

In the linear mode, the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage.

The unity gain crossover frequency and the input bias current are temperature-compensated.

FEATURES

- Internally frequency-compensated for unity gain
- Large DC voltage gain: 100dB
- Wide bandwidth (unity gain): 1MHz (temperature-compensated)
- Wide power supply range Single supply: $3V_{DC}$ to $30V_{DC}$ or dual supplies: $\pm1.5V_{DC}$ to $\pm15V_{DC}$
- Very low supply current drain: essentially independent of supply voltage (1mW/op amp at $+5V_{DC}$)
- Low input biasing current: $45nA_{DC}$ (temperature-compensated)
- Low input offset voltage: $2mV_{DC}$ and offset current: $5nA_{DC}$
- Differential input voltage range equal to the power supply voltage
- Large output voltage: $0V_{DC}$ to V_{CC}-$1.5V_{DC}$ swing

PIN CONFIGURATION

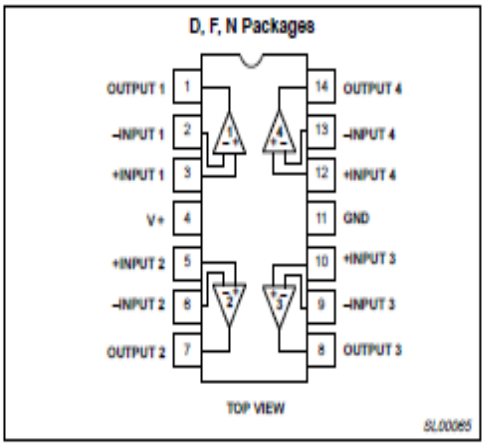


Figure 1. Pin Configuration

ORDERING INFORMATION

DESCRIPTION	TEMPERATURE RANGE	ORDER CODE	DWG #
14-Pin Plastic Dual In-Line Package (DIP)	-55°C to +125°C	LM124N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	-55°C to +125°C	LM124F	0581B
14-Pin Plastic Dual In-Line Package (DIP)	-25°C to +85°C	LM224N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	-25°C to +85°C	LM224F	0581B
14-Pin Plastic Small Outline (SO) Package	-25°C to +85°C	LM224D	SOT108-1
14-Pin Plastic Dual In-Line Package (DIP)	0°C to +70°C	LM324N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	0°C to +70°C	LM324F	0581B
14-Pin Plastic Small Outline (SO) Package	0°C to +70°C	LM324D	SOT108-1
14-Pin Plastic Dual In-Line Package (DIP)	0°C to +70°C	LM324AN	SOT27-1
14-Pin Plastic Small Outline (SO) Package	0°C to +70°C	LM324AD	SOT108-1
14-Pin Plastic Dual In-Line Package (DIP)	-40°C to +85°C	SA534N	SOT27-1
14-Pin Ceramic Dual In-Line Package (CERDIP)	-40°C to +85°C	SA534F	0581B
14-Pin Plastic Small Outline (SO) Package	-40°C to +85°C	SA534D	SOT108-1

ANEXO C – DADOS DO CI LM35

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

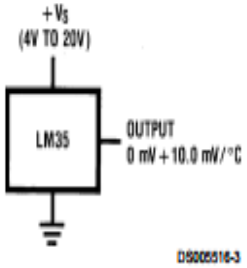
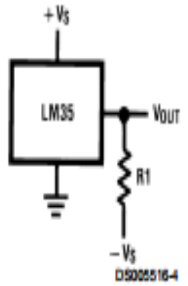


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)



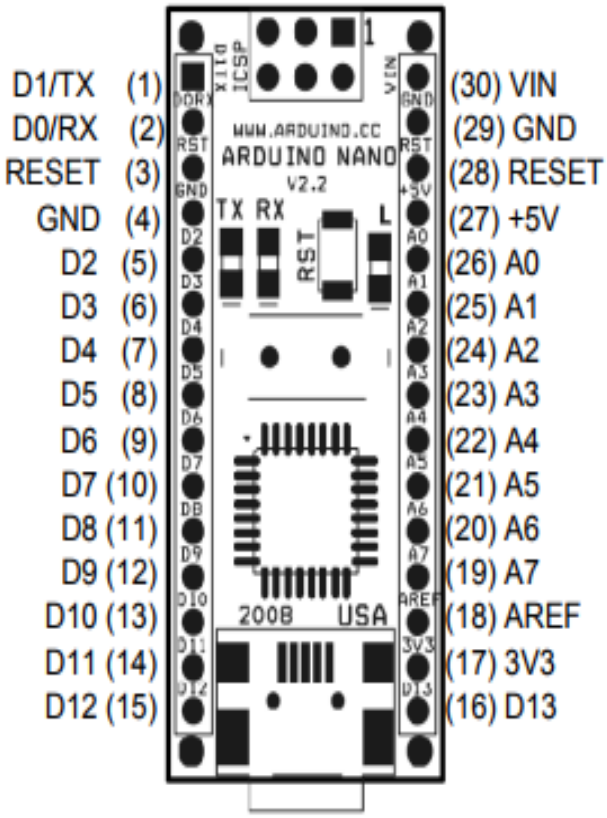
Choose $R_1 = -V_S/50\ \mu A$

V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

ANEXO D – DADOS DO ARDUINO NANO

Arduino Nano Pin Layout



Pin No.	Name	Type	Description
1-2, 5-16	D0-D13	I/O	Digital input/output port 0 to 13
3, 28	RESET	Input	Reset (active low)
4, 29	GND	PWR	Supply ground
17	3V3	Output	+3.3V output (from FTDI)
18	AREF	Input	ADC reference
19-26	A7-A0	Input	Analog input channel 0 to 7
27	+5V	Output or Input	+5V output (from on-board regulator) or +5V (input from external power supply)
30	VIN	PWR	Supply voltage

ANEXO E – DADOS DO SENSOR HC-SR04

HC-SR04 Specifications

- Working Voltage: DC 5V
- Working Current: 15mA
- Working Frequency: 40Hz
- Max Range: 4m
- Min Range: 2cm
- Measuring Angle: 15 degree
- Trigger Input Signal: 10µS TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion
- Dimension 45 * 20 * 15mm

ANEXO F – DADOS DOS CI'S 7805 E 7808

FAIRCHILD
SEMICONDUCTOR®

www.fairchildsemi.com

MC78XX/LM78XX/MC78XXA

3-Terminal 1A Positive Voltage Regulator

Features

- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The MC78XX/LM78XX/MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

TO-220

1

D-PAK

1

1. Input 2. GND 3. Output

ANEXO G – DADOS DO PIC16F716

PIC16F716

18-Pin Diagram

18-pin PDIP, SOIC

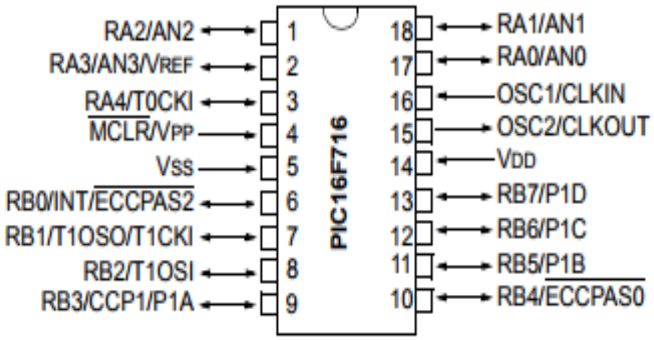


TABLE 1: 18-PIN PDIP, SOIC SUMMARY

I/O	Pin	Analog	ECCP	Timer	Interrupts	Pull-ups	Basic
RA0	17	AN0	—	—	—	—	—
RA1	18	AN1	—	—	—	—	—
RA2	1	AN2	—	—	—	—	—
RA3	2	AN3/VREF	—	—	—	—	—
RA4	3	—	—	T0CKI	—	—	—
RB0	6	—	$\overline{\text{ECCPAS2}}$	—	INT	Y	—
RB1	7	—	—	T1CKI	—	Y	—
RB2	8	—	—	T1OSI	—	Y	—
RB3	9	—	CCP1/P1A	—	—	Y	—
RB4	10	—	$\overline{\text{ECCPAS0}}$	—	IOC	Y	—
RB5	11	—	P1B	—	IOC	Y	—
RB6	12	—	P1C	—	IOC	Y	ICSPCLK
RB7	13	—	P1D	—	IOC	Y	ICSPDAT
—	14	—	—	—	—	—	VDD
—	5	—	—	—	—	—	VSS
—	4	—	—	—	—	—	$\overline{\text{MCLR}}$/VPP
—	16	—	—	—	—	—	OSC1/CLKIN
—	15	—	—	—	—	—	OSC2/CLKOUT

ANEXO H – DADOS DA CELULA DE LITIO CGR17670A

Fedco Batteries™

About Us | Quality | Where To Buy | Login | Contact Us ENERGY+®

Search: GO

What's New

Computer Batteries
Industrial Batteries
Medical Batteries
Military Batteries
Battery Cells

Custom Battery Assembly

1-800-542-9761

Panasonic Li-Ion Cell for Use With Approved Safety Circuit

Energy+ Part:	CGR17670A
Brand:	Panasonic
Type:	Li-Ion
Voltage:	3.7
Capacity:	1500 mAh
Color:	
Size:	17mm dia. x 67mm tall
Contacts:	Button Type
Warranty:	12 Months
Suggested Retail:	$ Please Call
Quantity Available:	900

Buy Online

Please Call to speak to a Sales Engineer.

Please Call to speak to a Sales Engineer.

Please Call to speak to a Sales Engineer.

Please Call to speak to a Sales Engineer.

Panasonic Li-Ion

We're sorry, this item is shown for reference purposes only and is not currently available for sale.

For safety reasons individual cells are not for sale.

We are an authorized Panasonic Li-Ion pack assembler.

Please contact our sales department at 1-800-542-9761 to discuss having us build you a custom battery pack.

ANEXO I – DADOS DA CELULA DE LITIO INR18650

Spec. No.	INR18650-25R	Version No.	1.0	In-Young Jang

1.0. Scope

This product specification has been prepared to specify the rechargeable lithium-ion cell ('cell') to be supplied to the customer by Samsung SDI Co., Ltd.

2.0. Description and model name

2.1 Description lithium-ion rechargeable cell

2.2 Model name INR18650-25R

3.0. Nominal specifications

Item	Specification
3.1 Nominal discharge capacity	2,500mAh Charge: 1.25A, 4.20V,CCCV 125mA cut-off, Discharge: 0.2C, 2.5V discharge cut-off
3.2 Nominal voltage	3.6V
3.3 Standard charge	CCCV, 1.25A, 4.20 ± 0.05 V, 125mA cut-off
3.4 Rapid charge	CCCV, 4A, 4.20 ± 0.05 V, 100mA cut-off
3.6 Charging time	Standard charge : 180min / 125mA cut-off Rapid charge: 60min (at 25℃) / 100mA cut-off
3.7 Max. continuous discharge (Continuous)	20A(at 25℃), 60% at 250 cycle
3.8 Discharge cut-off voltage End of discharge	2.5V
3.9 Cell weight	45.0g max
3.10 Cell dimension	Height : 64.85 ± 0.15mm Diameter : 18.33 ± 0.07mm
3.11 Operating temperature (surface temperature)	Charge : 0 to 50℃ (recommended recharge release < 45℃) Discharge: -20 to 75℃ (recommended re-discharge release < 60℃)
3.12 Storage temperature (Recovery 90% after storage)	1.5 year -30~25℃(1*) 3 months -30~45℃(1*) 1 month -30~60℃(1*)

ANEXO J – DADOS DA CELULA DE LITIO UR18650A

Panasonic

Lithium Ion

UR18650A

Features & Benefits

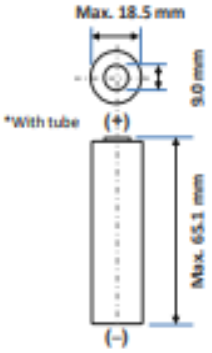
- General purpose model
- Superior cost performance
- Ideal for notebook PCs, backup applications, handheld devices, etc.

Specifications

Rated capacity[1]	Min. 2100mAh
Capacity[2]	Min. 2150mAh Typ. 2250mAh
Nominal voltage	3.6V
Charging	CC-CV, Std. 1510mA, 4.20V, 3.0 hrs
Weight (max.)	44.0 g
Temperature	Charge: 0 to +45°C Discharge: -20 to +60°C Storage: -20 to +50°C
Energy density[3]	Volumetric: 453 Wh/l Gravimetric: 176 Wh/kg

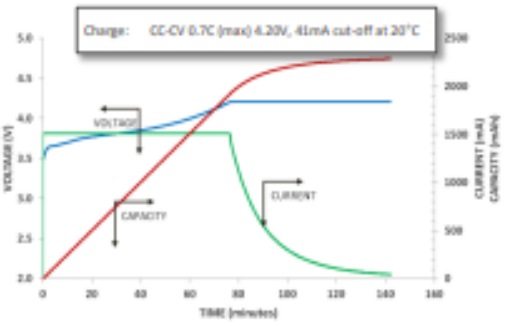
[1] At 20° C [2] At 25° C [3] Energy density based on bare cell dimensions

Dimensions

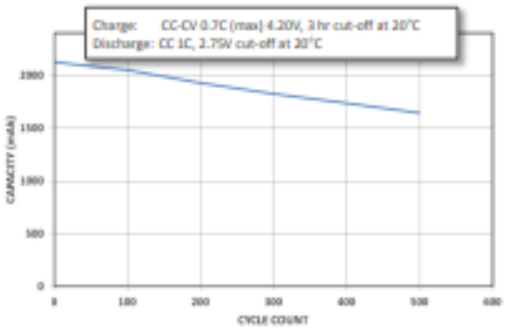


For Reference Only

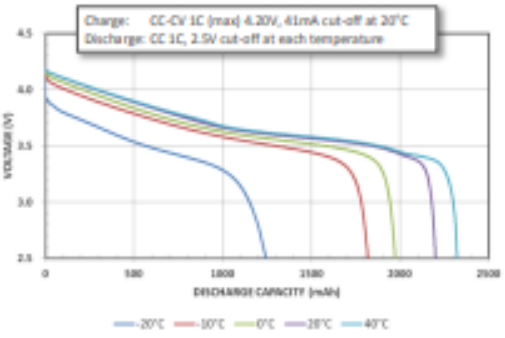
Charge Characteristics



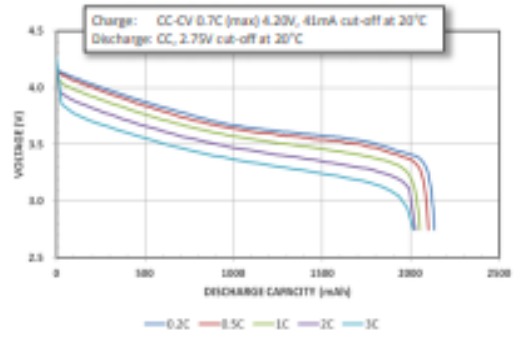
Cycle Life Characteristics



Discharge Characteristics (by temperature)



Discharge Characteristics (by rate of discharge)



The data in this document is for descriptive purposes only and is not intended to make or imply any guarantee or warranty.

ANEXO K – DADOS DO MÓDULO DE RF (TRANSMISSOR E RECEPTOR)

433Mhz RF Transmitter With Receiver Kit For Arduino ARM MCU Wireless

Description:

This is 433Mhz RF transmitter with receiver kit for Arduino ARM MCU wireless
Application environment:

Remote control switch, receiver module, motorcycles, automobile anti-theft products, home security products, electric doors, shutter doors, windows, remote control socket, remote control LED, remote audio remote control electric doors, garage door remote control, remote control retractable doors, remote volume gate, pan doors, remote control door opener, door closing device control system, remote control curtains, alarm host, alarm, remote control motorcycle remote control electric cars, remote control MP3.

Specification:

Receiver module:
Product Model: XD-RF-5V
Operating voltage: DC5V
Quiescent Current: 4MA
Receiving frequency: 433.92MHZ
Receiver sensitivity: -105DB
Size:30x14x7mm

Transmitter:

Product Model: XD-FST
Launch distance :20-200 meters (different voltage, different results)
Operating voltage :3.5-12V
Dimensions: 19 * 19mm
Operating mode: AM
Transfer rate: 4KB / S
Transmitting power: 10mW
Transmitting frequency: 433M
Pinout from left → right: (DATA; VCC; GND)

See for how to: http://electronics-diy.com/arduino-rf-link-using-433mhz-transmitter-receiver-modules.php

ANEXO L – DADOS DA FONTE DE ALIMENTACAO MINIPA MPS-3005

6) ESPECIFICAÇÕES

A. Especificações Gerais

- Alimentação 110V/220V ± 10% - 50/ 60Hz (selecionável).
- Uso interno.
- Altitude: 2000m (máximo).
- Grau de Poluição: 2.
- Consumo Aprox.: 175W (MPS-3003);
  260W (MPS-3005).
- Ambiente de Operação: 0°C a 40ºC, RH < 80%.
- Ambiente de Armazenamento: -10ºC a 70ºC, RH < 70%.
- Dimensões: 145 (A) x 128 (L) x 285 (P) mm (MPS-3003 e MPS-3005).
- Peso Aprox.: 4Kg (MPS-3003);
  5Kg (MPS-3005).

B. Especificações Elétricas

As especificações são influenciadas pelas resistências dos contatos e dos cabos. Portanto tente minimizá-las, assim como utilizar conexões externas auxiliares nos modos Série e Paralelo, mesmo que já exista uma comutação interna.

- ***Operação Tensão Constante:***

FONTE		**MPS-3003**	**MPS-3005**
Saída (Continuamente Ajustável):		0 ~ 30V	
Precisão		± (1x10^{-4} + 0,5mV)	
Regulação	Linha:	< 0,01%+3mV	
	Carga:	< 0,01%+3mV	< 0,01%+5mV
Ripple e Ruído (5Hz ~ 1MHz):		< 0,5mV rms	< 1mV rms

- ***Operação Corrente Constante:***

FONTE		**MPS-3003**	**MPS-3005**
Saída (Continuamente Ajustável):		0 ~ 3A	0 ~ 5A
Precisão		± (2x10^{-3} + 6mA)	
Regulação	Linha:	< 0,2%+3mA	
	Carga:	< 0,2%+3mA	
Ripple Ruído:		< 3mA rms	< 3mA rms

- ***Tempo de recuperação para troca de cargas: < 100µs (para variação de carga de 50% e corrente mínima da carga de 0,5A).***