UNIVERSIDADE DE CAXIAS DO SUL

EDERSON LUIS BOFF

INTEGRAÇÃO DA ROBÓTICA LEGO® RCX COM O ARDUINO

CAXIAS DO SUL

2016

EDERSON LUIS BOFF

INTEGRAÇÃO DA ROBÓTICA LEGO® RCX COM O ARDUINO

Trabalho de conclusão de curso apresentado ao curso de Engenharia de Controle e Automação da Universidade de Caxias do Sul – UCS, como requisito para a obtenção do título de bacharel em Engenharia de Controle e Automação.

Orientadora Prof. Ma. Andréa Cantarelli Morales

CAXIAS DO SUL

2016

EDERSON LUIS BOFF

INTEGRAÇÃO DA ROBÓTICA LEGO® RCX COM O ARDUINO

Trabalho de conclusão de curso apresentado ao curso de Engenharia de Controle e Automação da Universidade de Caxias do Sul – UCS, como requisito parcial para a obtenção do título de bacharel em Engenharia de Controle e Automação.

Aprovado em 08/12/2016

Banca Examinadora

_____

Prof. (a) Ma. Andréa Cantarelli Morales
Universidade de Caxias do Sul – UCS

_____

Prof. Ms. Daniel Faccin
Universidade de Caxias do Sul – UCS

_____

Prof. (a) Ma. Patrícia Giacomelli
Universidade de Caxias do Sul – UCS

AGRADECIMENTOS

Agradeço a Deus, por iluminar o meu caminho durante esta jornada, pelo fim de mais essa etapa, pelos sonhos que se concretizam e as possibilidades que se renovam.

Aos meus pais Olinto e Denize, pela capacidade de acreditar e investir em mim, pelo cuidado e dedicação que, em determinados momentos, me deram a esperança para seguir.

A minha orientadora professora Ma. Andréa Cantarelli Morales, por seus ensinamentos, pela confiança no desenvolvimento deste trabalho, pela compreensão e amizade.

Aos amigos e colegas, pelas palavras de apoio, pelo auxílio durante esses anos de graduação e pela compreensão de minhas escolhas e ideias.

RESUMO

A robótica educacional nasceu do princípio da construção do conhecimento através da interação do ser humano com o ambiente e a sua compreensão do mesmo. Aliado a esta ideologia, o computador representa uma ferramenta muito importante nesse processo educacional, criando um ambiente de aprendizagem inovador. Surgiu então a robótica educacional, visando o aperfeiçoamento e a ampliação do conhecimento de estudantes através da montagem e programação de robôs autônomos utilizando *kit's* de robótica, além do desenvolvimento de habilidades e competências adjuntos aos avanços tecnológicos. Com base nesses aspectos, surgiu a motivação de realizar a integração da robótica LEGO® com a plataforma Arduino, unificando a tecnologia de programação com a atualidade. Para tal fim, efetuou-se a análise de *hardware* do bloco programável RCX, dos sensores e atuadores LEGO®. Com a finalidade de controlar esses atuadores com o Arduino UNO, se fez necessário o desenvolvimento de uma placa auxiliar, acoplada ao mesmo, bem como a expansão do número de conexões para a ligação de sensores e atuadores, aumentando a flexibilidade, diversidade e interação na montagem de projetos robóticos. Também foram confeccionados cabos de energia e trocados os sensores de luz. Para auxiliar na programação de *software* do Arduino, foi elaborado um manual didático e também foram demonstrados os resultados obtidos com o projeto proposto, em comparação ao *kit* de robótica educacional LEGO® *Mindstorms* RCX presente na Instituição de Ensino Superior onde foi desenvolvido o projeto.

Palavras-chave: LEGO®, Robótica, Arduino, Programação.

LISTA DE FIGURAS

Figura 1- Kit LEGO® Mindstorms RCX 16
Figura 2- Bloco Programável RCX 1.0 16
Figura 3- Especificações Técnicas RCX 1.0 17
Figura 4- Portas de Entrada 18
Figura 5- Portas de Saída 18
Figura 6- Botões de Controle e Display 18
Figura 7- Transmissor Infravermelho 19
Figura 8- Sensor de Toque 20
Figura 9- Sensor de Luz 20
Figura 10- Vista explodida de um motor CC 21
Figura 11- Motor 21
Figura 12- Lâmpada 22
Figura 13- Placa Auxiliar 22
Figura 14- Conexões Placa Auxiliar 23
Figura 15- Arduino UNO 24
Figura 16- Especificações técnicas Arduino UNO 25
Figura 17- Especificações dos pinos com configurações especiais do Arduino UNO 26
Figura 18- Especificações dos pinos de alimentação do Arduino UNO 26
Figura 19- Ponte H 27
Figura 20- Acionamento das Chaves 27
Figura 21- Circuito Integrado L293D 28
Figura 22 - Simbologia e Tabela-verdade 29
Figura 23- Circuito Integrado SN7404 30
Figura 24- Esquema Elétrico do Hardware 31
Figura 25- Layout Placa Auxiliar 31
Figura 26- Especificações das Conexões da Placa Auxiliar 32
Figura 27- Classificador de Peças 33
Figura 28- Conectores de Energia 34
Figura 29- Esquema Elétrico TCRT 5000 34
Figura 30- Sensor Óptico Reflexivo TCRT 5000 35
Figura 31- Site Arduino para realizar download do software 40
Figura 32- Interface do Arduino IDE 41
Figura 33- Selecionando a Plataforma Arduino 42

Figura 34- Escolha da Porta de Comunicação 42
Figura 35- Comandos de Verificação e Transferência 43
Figura 36- Estrutura do Sketch 44
Figura 37- Tipos de Dados e Operadores 45
Figura 38- Comando Serial.print( ) 48
Figura 39- Tela de Monitoramento Serial 49
Figura 40- Condicionais If e else 50
Figura 41- Condicional Switch-Case 51
Figura 42- Laço Do-while 52
Figura 43- Laço for 53

LISTA DE TABELAS

Tabela 1- Especificações L293D 54
Tabela 2- Especificações SN74LS04 55
Tabela 3- Especificações TCRT 5000 56

LISTA DE SIGLAS

CC *Direct Current* (Corrente contínua)

CI Circuito Integrado

CMOS *Complementary Metal-Oxide Semiconductor* (Semicondutor de metal-óxido complementar)

ICSP *In Circuit Serial Programming* (Circuito de Programação Serial)

IDE *Integrated Development Environment* (Ambiente de Desenvolvimento Integrado)

LCD *Liquid Crystal Display* (Display de Cristal Líquido)

MIT *Massachusetts Institute of Technology* (Instituto de tecnologia de Massachusetts)

PC *Personal Computer* (Computador Pessoal)

PWM *Pulse Width Modulation* (Modulação de Largura de Pulso)

RCX *Robotic Command Explorer* (Explorador de Comando Robótico)

RIA *Robotics Institute of American* (Instituto de Robótica Americano)

RIS *Robotics Invention System* (Sistema de Invenção Robótica)

RUR *Rossum's Universal Robots* (Robôs Universais de Rossum)

TTL *Transistor-Transistor Logic* (Lógica Transistor-Transistor)

UCS Universidade de Caxias do Sul

USB *Universal Serial Bus* (Barramento Serial Universal)

SUMÁRIO

1. INTRODUÇÃO 11

1.1 JUSTIFICATIVA DO TRABALHO 12

1.2 OBJETIVOS 13

1.2.1 Objetivo geral 13

1.2.2 Objetivos específicos 13

1.3 LIMITES DO TRABALHO 13

2. REFERENCIAL TEÓRICO 14

2.1 ROBÓTICA 14

2.2 LEGO® *MINDSTORMS* 15

2.2.1 Rcx 16

2.2.2 Transmissor de dados 19

2.2.3 Sensor de toque 19

2.2.4 Sensor de luz 20

2.2.5 Motor 21

2.2.6 Lâmpada 22

2.3 PROJETO DE INTEGRAÇÃO ENTRE PLATAFORMAS 22

2.3.1 Controlador 23

2.3.2 Ponte H 26

2.3.3 Circuito lógico digital 28

2.4 FUNCIONAMENTO DA PLACA AUXILIAR 30

2.5 RESULTADOS 32

CONSIDERAÇÕES FINAIS 36

REFERÊNCIAS 37

APÊNDICE A – MANUAL DIDÁTICO ARDUINO UNO 40

ANEXO A – DADOS TÉCNICOS L293D 54

ANEXO B – DADOS TÉCNICOS SN74LS04 55

ANEXO C – DADOS TÉCNICOS TCRT 5000 56

1. INTRODUÇÃO

Com os avanços tecnológicos, o ser humano tem buscado adaptar e inovar os métodos de aprendizagem, afim de tornar mais dinâmico o processo de construção do conhecimento (ZILLI, 2004). Isso também acontece nas instituições de ensino, que estão cada vez mais preocupadas com o futuro dos jovens inseridos nesse mundo de tecnologia, nas quais os métodos de transmissão de conhecimento para os estudantes devem ser diversificados para manter o interesse e o foco nas atividades de aprendizagem. Além da utilização de dispositivos como projetor de imagens, *tablets*, *notebooks* e *smartphones*, tem se implementado uma ferramenta de cunho prático, tanto para estudantes do ensino médio como estudantes de graduação. Esta nova ferramenta visa um ambiente de construção de conceitos, bem como, de acordo com Zilli (2004), o desenvolvimento de atividades relacionadas com áreas como a física, a matemática, a mecânica, a informática e a eletrônica.

Para essa integração de ambientes de aprendizagem surgiu a robótica educacional, a qual segundo Silva (2009) visa a ampliação do desenvolvimento cognitivo, a autonomia, a capacidade de trabalhar em grupo, o desenvolvimento de habilidades e competências adjuntos à lógica, a interdisciplinaridade, a motivação, o entusiasmo, a imaginação e a criatividade. Através da construção e controle de dispositivos robóticos, utilizando *kit's* programáveis constituídos por blocos de montar com variadas cores e tamanhos, motores, eixos, engrenagens, polias, correntes, rodas, cremalheiras, sensores, lâmpadas e um controlador programável (CABRAL, 2010).

Um dos *kit's* de robótica mais populares é o LEGO® *Mindstorms*, no qual apresenta um microcontrolador inserido em um bloco de montagem maior, o RCX (*Robotic Command Explorer*). De acordo com Cabral (2010 apud RESNICK et al, 1996), o bloco RCX é o cérebro de qualquer criação robótica e é responsável pela comunicação entre o projeto mecânico e o digital. Para sua programação utiliza-se um *software* para computador com linguagem gráfica (RoboLAb®), constituído de ícones para selecionar as instruções e comandos a serem executados pelos robôs criados, de forma simples e flexível. O *kit* LEGO® já está em sua terceira geração desde seu lançamento em 1998, tendo início com o modelo RCX, posteriormente em 2006, com o modelo NTX e o modelo mais atual, EV3 lançado em 2013. Cada modelo foi desenvolvido afim de acompanhar a evolução das tecnologias de computação e

robótica, apresentando características específicas de *software* e *hardware* para cada modelo lançado.

Pretende-se, com este trabalho, realizar a integração do *kit* LEGO® RCX 1.0, a primeira geração da linha LEGO® *Mindstorms*, disponível na UCS (Universidade de Caxias do Sul) para cursos de extensão, com o Arduino, plataforma de prototipagem eletrônica com código aberto, muito utilizada por estudantes para a realização de projetos. Com isso busca-se disponibilizar aos estudantes todo o material de robótica LEGO® já presente na instituição, mas com uma ferramenta de *software* utilizada para criar um ambiente atual de programação, para as diferentes aplicações robóticas a serem desenvolvidas.

1.1 JUSTIFICATIVA DO TRABALHO

Com o aumento de estudantes de ensino fundamental e médio interessados no curso de robótica educacional criativa disponível pela UCS, e também visando uma possível extensão desse curso para estudantes de graduação, surgiu a necessidade da utilização de outro controlador, bem como a programação de seu *software*, visando a utilização das peças disponíveis nos *kit's* e realizando a integração com outra plataforma, não só pela versatilidade nas aplicações práticas, mas também a disponibilidade maior de ligações para sensores e atuadores.

A migração para outro controlador justifica-se, devido a alguns fatores, como: o *software* de programação RoboLAb® ser exclusivo e necessitar de licença para instalação; A transmissão de dados entre o PC e o RCX ser realizada através de uma torre com transmissor de infravermelho, na qual utiliza comunicação através de porta serial, praticamente extinta nos PC's atuais; A perda de programação se o RCX ficar sem alimentação; A necessidade do mesmo em utilizar seis pilhas ligadas em série para seu funcionamento; Entradas e saídas limitadas em três cada, tanto para sensores como atuadores.

Devido a estas características do RCX, optou-se em utilizar a plataforma Arduino, principalmente por possuir *software* aberto, sendo possível de ser instalado em qualquer PC. Também possui muitas páginas na internet com exemplos de aplicações e pseudocódigos, além de uma vasta gama de sensores compatíveis com sua estrutura. É um controlador muito utilizado na atualidade para a elaboração de

projetos e, aproveitando esse contexto os estudantes terão o contato com essa plataforma e aprenderão noções de programação.

1.2 OBJETIVOS

1.2.1 Objetivo geral

Realizar a integração dos sensores e atuadores da plataforma educacional LEGO® *Mindstorm* RCX 1.0 com a plataforma Arduino, visando a flexibilidade nas aplicações, bem como, a interação na montagem de projetos robóticos tanto nas áreas de ensino fundamental e médio como de engenharia e pesquisa, buscando também, a montagem de novos *kit's* didáticos com um custo reduzido.

1.2.2 Objetivos específicos

a. Efetuar a análise de hardware do bloco programável RCX e também de seus sensores e atuadores;

b. Realizar a integração dos elementos LEGO® com a plataforma Arduino, buscando uma forma atual e alternativa para a programação do *software*;

c. Aumentar o número de conexões disponíveis para a ligação de sensores e atuadores a serem aplicados nos projetos;

d. Elaborar um manual didático, auxiliando na programação do *software* responsável pela comunicação entre o projeto digital e o mecânico.

1.3 LIMITES DO TRABALHO

Para o processo de integração entre as duas tecnologias e seus atuadores, surgiu a necessidade da construção de uma placa de circuito impresso auxiliar para o controle de velocidade e rotação dos motores. O *layout* do circuito será desenvolvido pelo autor, a placa será produzida por uma empresa terceirizada, visando a obtenção de um produto com qualidade profissional e, por fim, a montagem dos componentes dessa placa será efetuada pelo autor. O sensor de luz do LEGO® será substituído pelo TCRT 5000, já que o sensor do LEGO® é do tipo passivo, não sendo possível a integração com o Arduino.

2. REFERENCIAL TEÓRICO

2.1 ROBÓTICA

O termo robô surgiu no século XX, com a palavra *robota*, de origem tcheca e significado de atividade forçada (escravo), a qual foi concebida por Karel Capek em uma peça teatral intitulada RUR (*Rossum's Universal Robots*), encenada em Praga (República Tcheca), em 1921. Nessa peça, os robôs eram pessoas eficientes, fabricadas artificialmente, mas não dispunham de emoção (POLONSKII, 1996). Entretanto a ideia de criar mecanismos servos e seres artificiais, com o intuito de atenderem os desejos de seus criadores, tem acompanhado a humanidade desde a mitologia antiga, com os egípcios, os gregos e judeus (SILVA, 2009).

A construção de robôs, compostos por partes mecânicas, elétricas e computacionais, foi direcionada inicialmente para aplicações industriais, sendo necessário um aumento da produtividade e melhoria da qualidade dos produtos nas fábricas (ZILLI, 2004). Ainda, segundo Melo (2009, apud RIA (Robotics Institute Of American)), um robô é considerado "um manipulador, reprogramável e multifuncional, projetado para mover materiais, objetos, ferramentas ou dispositivos específicos, através de movimentos variados, podendo ser programado com vista à desempenhar variadas tarefas".

Em 1941 surgiu o conceito de robótica, com o estudo e uso dos robôs, inspirada na literatura de ficção científica do escritor russo-americano Isaac Asimov, intitulada *Runaround*. Posteriormente, Asimov criou as três leis fundamentais da robótica, as quais são utilizadas atualmente para evolução de sistemas inteligentes. Contudo, a robótica não é apenas uma obra de ficção. É a ciência que estuda a tecnologia associada a teoria, ao projeto, a fabricação e aplicação dos robôs, envolvendo inúmeras áreas de conhecimento, como a microeletrônica, a engenharia mecânica, a inteligência artificial, a física, a neurociência, entre outras áreas científicas (SILVA, 2009).

Ainda, segundo o mesmo autor, a robótica sempre seguiu uma linha de atuação em meios industriais, porém atualmente com sua expansão e com os avanços tecnológicos, o campo de aplicações dos robôs tem se diversificado, no qual podem ser inseridos em diferentes contextos, como nas áreas de entretenimento, de medicina, doméstica, militar, de segurança, de veículos autônomos inteligentes,

busca, salvamento e educacional. Graças a esta ferramenta interdisciplinar que é a robótica, foram criados *kit's* educacionais programáveis, visando o aperfeiçoamento na construção do conhecimento e uma vasta gama de experiências de aprendizagem.

2.2 LEGO® *MINDSTORMS*

Na década de 60, o professor e pesquisador do MIT (*Massachussets Institute of Technology*), Seymour Papert acreditava no computador como um ambiente de aprendizagem para o aperfeiçoamento do conhecimento dos estudantes. E baseado nos princípios educacionais do desenvolvimento cognitivo de Jean Piaget, que propunha a construção do conhecimento através da interação do ser humano com o ambiente e a sua compreensão do mesmo, Papert desenvolveu uma linguagem de programação chamada LOGO (SANTOS, 2005). Essa linguagem era composta por um robô móvel, na forma de uma tartaruga, na qual era possível realizar a programação de seus movimentos e visualizá-los no monitor do computador (SILVA, 2009).

Segundo Conchinha (2015, apud RICCA, LULIS, & BADE, 2006), no ano de 1985 ocorreu a junção do LOGO com os brinquedos da LEGO®, parceria concebida entre o MIT e o grupo LEGO, dando origem ao *Robotics Invention System* (RIS), sistema de robótica educacional no qual era possível construir e programar os próprios robôs, porém os mesmos ficavam atrelados ao computador por meio de cabos.

Já na década de 90, conforme Cabral (2010), Mitchel Resnick aprimorou a linguagem LOGO desenvolvida por Papert, utilizando uma linguagem gráfica baseada no *software* LabView®, intitulado como RoboLAb®, a qual era transferida para o bloco programável RCX, podendo ou não ser montado junto a estrutura do robô e possibilitando ao mesmo a execução das tarefas programadas, sem ficar "preso" por cabos ao computador. E em 1998, esse projeto é lançado então, como kit de Robótica Educacional LEGO® Mindstorms RCX (conforme ilustrado na Figura 1), nome inspirado no livro Mindstorms: Children, Computers and Powerfull Ideas (Tempestade Cerebral: Crianças, Computadores e Ideias Poderosas) de Papert (CONCHINHA, 2015).

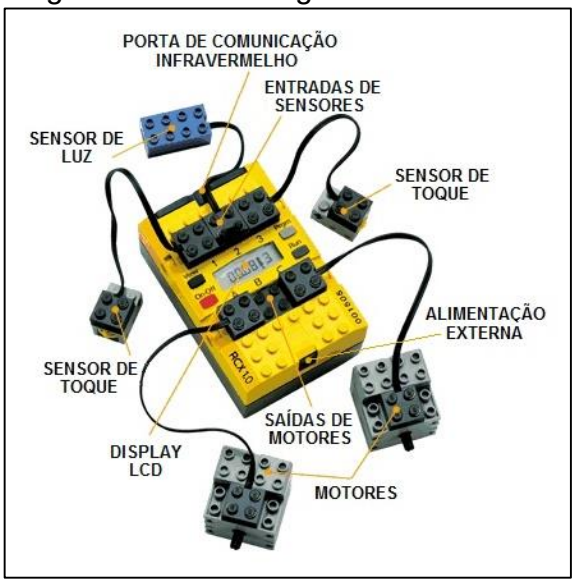
Figura 1- *Kit* LEGO® Mindstorms RCX

Fonte: Silva (2009)

2.2.1 Rcx

É uma unidade de controle constituída por um microcontrolador autônomo inserido em um bloco de montagem LEGO®, o qual pode ser programado por computador. O RCX é o componente essencial para qualquer desenvolvimento robótico, pois é responsável pelo processamento do programa criado, a leitura do ambiente através de sensores e a execução de comandos dos motores e lâmpadas para controle do robô (CABRAL, 2010). Na Figura 2 é apresentado o bloco programável RCX 1.0.

Figura 2- Bloco Programável RCX 1.0



Fonte: Adaptado de Fulber (2008)

De acordo com López (2006), o microcontrolador utilizado no núcleo da estrutura do RCX é um Hitachi H8/3292, com uma velocidade de *clock* de 16 MHz e tensão de alimentação de 5 V. Possui memória ROM de 16 KB, na qual encontra-se o *firmware*[1] básico, incorporado pelo fabricante com cinco programas básicos.

Dispõe de 512 Bytes de memória RAM interna e 32 KB externos, disponíveis para carregar o *firmware* e programas desenvolvidos que servirão de complemento às instruções gravadas na memória ROM. Se a alimentação for removida os dados da memória RAM serão perdidos, sendo necessário transferir novamente o programa para o controlador quando o mesmo for religado (SILVA, 2009).

O RCX possui dois temporizadores de 8 bits e um de 16 bits, um conversor analógico-digital de 8 bits e um mini alto-falante, capaz de emitir sons simples. As especificações técnicas do RCX 1.0 são apresentadas na Figura 3.

Figura 3- Especificações Técnicas RCX 1.0

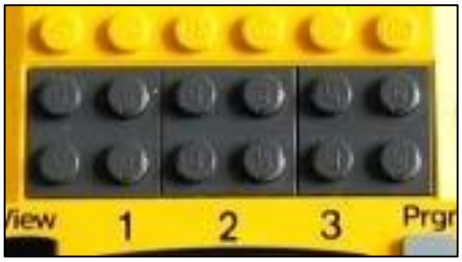
Microcontrolador	Hitachi H8/3292
Tensão de entrada (6 pilhas AA)	9 V
Tensão de entrada fonte externa (AC)	9 - 12 V
Tensão de Funcionamento	5 V
Corrente máxima suportada	500 mA
Pinos de Entradas Analógicas	3
Pinos de Saídas Analógicas (PWM)	3
Memória ROM	16 Kb
Memória RAM	512 Bytes
Velocidade de Clock	16 MHz
Comprimento	95 mm
Largura	63 mm
Altura	40 mm

Fonte: O autor (2016)

O bloco programável dispõe de três portas de entrada (1,2,3) para conexão de sensores, conforme a Figura 4. O *kit* básico é composto por dois sensores de toque e um sensor de luz, porém diversos outros podem ser adquiridos como, sensor de temperatura, sensor de som, medidor de pH, medidor de umidade relativa, medidor de tensão, e outros (FULBER, 2008).

[1] Conjunto de instruções operacionais que regem o funcionamento do sistema, programadas diretamente no *hardware* do equipamento eletrônico (OLIVEIRA, 2010).

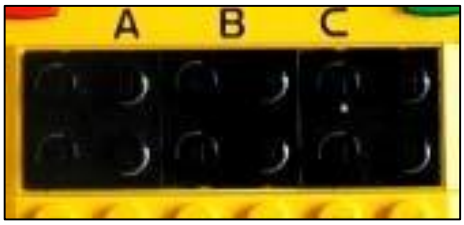
Figura 4- Portas de Entrada



Fonte: López (2006)

Possui também três portas de saída (A,B,C) para conexão de atuadores, mostrado na Figura 5. De modo geral, acompanham o *kit* dois motores e uma lâmpada. Os motores podem ainda, serem programados para girar em sentido horário ou anti-horário e utilizam o conceito de PWM (modulação por largura de pulso) para variar as velocidades dos mesmos em cinco níveis de intensidade (LÓPEZ, 2006).

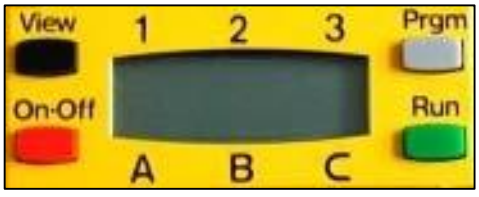
Figura 5- Portas de Saída



Fonte: López (2006)

O RCX tem quatro botões, definidos como “On-Off”, responsável por ligar e desligar o bloco, “Prgm”, o qual realiza a seleção do programa a ser executado, “Run”, coloca o programa selecionado em execução e “View”, utilizado para alterar o monitoramento das entradas ou saídas. Uma tela de LCD (display de cristal líquido) é responsável por exibir tais informações, e ainda informar o tempo que o RCX permanece ligado e o nível de bateria disponível (CAVALCA, 2006). Na Figura 6 são apresentados os botões de controle e o display.

Figura 6- Botões de Controle e Display



Fonte: López (2006)

2.2.2 Transmissor de dados

Para que ocorra a transferência do *firmware* e de programas realizados no RoboLab® para o RCX, se faz necessária a utilização de uma torre de comunicação infravermelha, conforme Figura 7, a qual é alimentada por uma bateria de 9 V e conectada à porta serial do PC (Computador Pessoal) (CABRAL, 2010).

Figura 7- Transmissor Infravermelho

Fonte: O autor (2016)

Segundo Silva (2006), para estabelecer a conexão entre o transmissor e o controlador, os mesmos devem estar alinhados e o campo de visão entre eles desobstruído. Como a comunicação é sem fio, por meio de luz infravermelha, pode ocorrer interferência na transmissão devido a incidência excessiva de luz sobre o conjunto, sendo ela realizada por lâmpadas ou mesmo pela luz do sol. Para evitar a ocorrência desse problema, deve-se tentar minimizar a exposição do conjunto a uma iluminação intensa, ou até mesmo cobrir a torre e o controlador.

O RCX, por sua vez, possui uma porta de comunicação infravermelha, que além de receber as informações da torre, permite a comunicação entre dois controladores, e também pode ser usada para o controle e monitoramento do robô construído, em tempo real (NETO, 2008).

2.2.3 Sensor de toque

Dispositivo que funciona como um interruptor normalmente aberto, e quando sua haste externa é pressionada, fecha os contatos elétricos internos ao bloco permitindo a circulação de corrente elétrica e variando a tensão de 0 a 5V (LÓPEZ, 2006).

É conectado às portas de entrada do RCX podendo ser utilizado como chave liga-desliga, botões para *joystick*, detecção de fim de deslocamento de peças e para indicar quando o robô toca ou colide em algum objeto, a fim de tomar a decisão de parar, voltar ou ainda mudar de direção. A Figura 8 ilustra o sensor de toque.

Figura 8- Sensor de Toque

Fonte: Neto (2008)

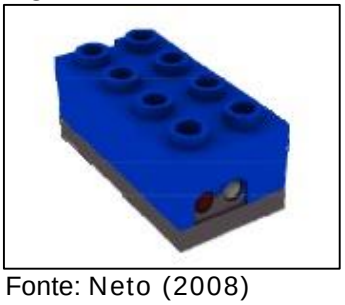
2.2.4 Sensor de luz

O sensor é composto por um LED vermelho, responsável por emitir uma luz que é refletida no material e captada por um fototransístor. Detecta apenas a existência de branco, preto e tons de cinza que condiz às porcentagens intermediárias de luz presentes no meio (NETO, 2008).

É um sensor do tipo passivo, no qual não necessita de alimentação para seu funcionamento, o mesmo realiza a leitura do meio e converte o sinal em uma variação de corrente elétrica. Essa variação é interpretada pelo RCX e corresponde à intensidade de reflexão de luz, a qual é expressa como uma porcentagem aproximada de brilho que varia de 0 a 100.

Conectado nas entradas do RCX, pode ser aplicado em projetos de detecção de proximidade, faixas de solo, limitação de área de atuação do robô e detecção de presença. A Figura 9 mostra o sensor de luz.

Figura 9- Sensor de Luz

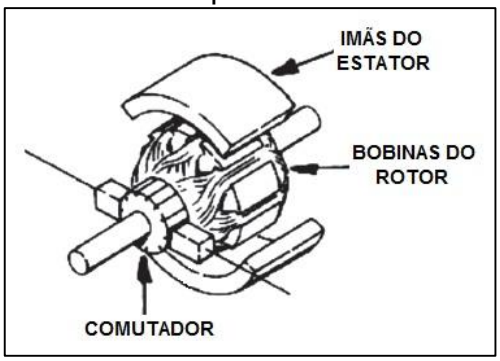


Fonte: Neto (2008)

2.2.5 Motor

O motor CC (Corrente contínua) é um dispositivo que converte energia elétrica em energia mecânica, possui ímãs circulares permanentes, denominado estator e bobinas montadas entre os polos destes ímãs, denominado rotor. A Figura 10 mostra uma vista explodida de um motor CC.

Figura 10- Vista explodida de um motor CC



Fonte: Adaptado de Mohan (2015)

Essas bobinas são alimentadas com uma tensão elétrica através do comutador, fazendo com que circule uma corrente elétrica nas espiras, produzindo um campo eletromagnético originando assim, uma força de repulsão com o campo magnético fixo do estator. Esta força faz o rotor mudar de posição toda vez que os polos se repelirem, tendo então o movimento circular e formando a força magnetomotriz (NASCIMENTO JR, 2014).

É possível estipular diferentes velocidades para o motor, conforme cada aplicação. A mesma é determinada via programação, na qual está disponível cinco níveis de intensidade.

É conectado às portas de saída do RCX e é aplicado em projetos que necessitam de movimento. A Figura 11 ilustra o motor.

Figura 11- Motor

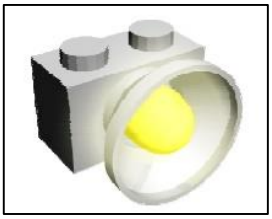
Fonte: Neto (2008)

2.2.6 Lâmpada

Dispositivo que transforma a energia elétrica em energia luminosa, no qual circula uma corrente elétrica através de um filamento de tungsténio, tornando-o incandescente, produzindo assim a luz, podendo variar a intensidade.

É conectada nas saídas do RCX e é utilizada em aplicações que necessitem de sinais luminosos ou alertas. A lâmpada é ilustrada na Figura 12.

Figura 12- Lâmpada



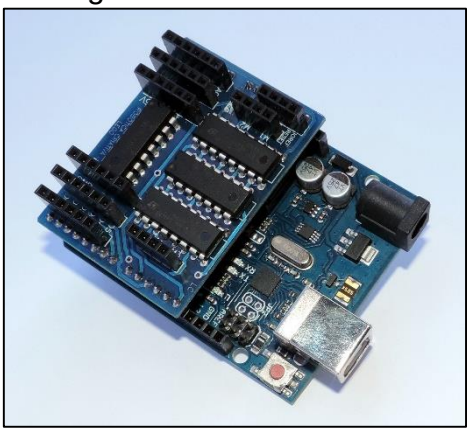
Fonte: Neto (2008)

2.3 PROJETO DE INTEGRAÇÃO ENTRE PLATAFORMAS

O projeto consiste na utilização do Arduino Uno para efetuar o controle dos sensores e atuadores do LEGO® *Mindstorms* RCX 1.0, além da expansão do número de conexões para a ligação de dispositivos, aumentando a flexibilidade, diversidade e interação na montagem de projetos robóticos.

Para ser possível essa implementação, se faz necessário o desenvolvimento de uma placa de circuito impresso auxiliar para expansão de *hardware*, a fim de controlar a velocidade e sentido de rotação dos motores. Essa placa será acoplada junto a placa principal do Arduino UNO, ilustrada na Figura 13.

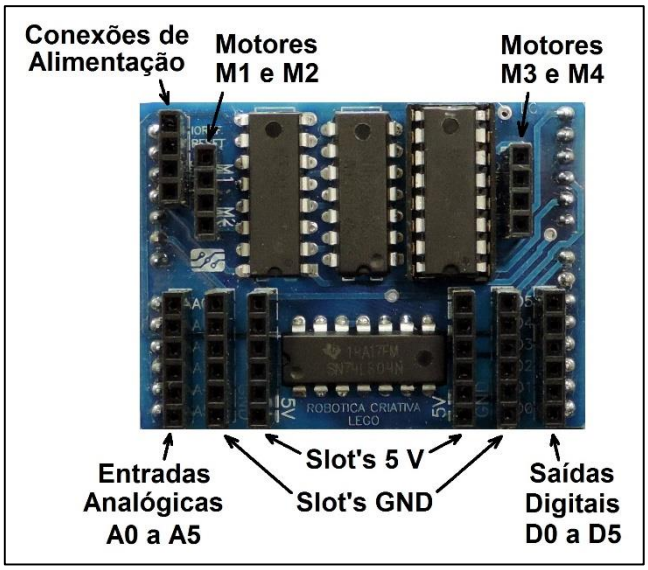
Figura 13- Placa Auxiliar



Fonte: O autor (2016)

A placa auxiliar terá a seguinte configuração: utilizará oito saídas digitais do Arduino para a execução de suas funções. As seis entradas/saídas digitais restantes, as seis entradas/saídas analógicas, bem como os pinos de alimentação (IOREF, Reset, Vin e 3,3 V) serão reproduzidos na placa auxiliar e também serão criados *slot's* (conectores de expansão) de alimentação (5 V) e aterramento (GND) para facilitar na conexão dos dispositivos. A Figura 14 mostra de forma detalhada as conexões da placa desenvolvida.

Figura 14- Conexões Placa Auxiliar



Fonte: O autor (2016)

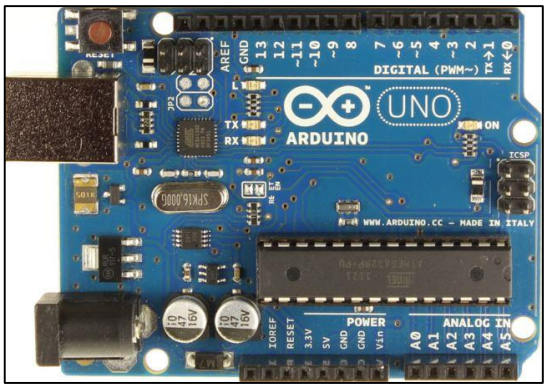
O circuito a ser utilizado para a elaboração do projeto é composto por uma bateria (9 V) para alimentação, a qual alimentará também o controlador, um Arduino UNO, responsável pela aquisição, análise de dados e execução de tarefas. Contará com dois CI's L293D (ponte H) para controle dos motores, dois CI's SN7404 (lógica NOT) para determinar as lógicas das saídas de sinal e conectores de barras de pinos fêmea 180 graus que permitem a ligação de periféricos. Ainda serão confeccionados cabos de energia para ligar os blocos LEGO® com o Arduino e também, a troca dos sensores de luz, itens que serão explicados no decorrer do trabalho.

2.3.1 Controlador

Na elaboração da integração será utilizado, como controlador, o Arduino UNO. Arduino é uma plataforma de prototipagem eletrônica com código aberto, composta

por uma única placa, com um ambiente de desenvolvimento integrado, contendo um microcontrolador Atmel AVR, algumas linhas de entrada e saída, além de uma interface USB (barramento serial universal). Utiliza para programação uma linguagem padrão baseada em C/C++ (OLIVEIRA, 2015). O objetivo da plataforma é oferecer ferramentas acessíveis e de baixo custo, muito flexíveis e fáceis de usar por programadores amadores ou profissionais. A Figura 15 ilustra o Arduino UNO.

Figura 15- Arduino UNO



Fonte: Arduino.cc (2016)

O Arduino Uno se mostrou como melhor opção para a aplicação desenvolvida em comparação ao microcontrolador Hitachi H8/3292, inserido no núcleo do RCX, embora ambos apresentem configurações semelhantes. O bloco RCX necessita de um *software* proprietário para a elaboração do programa e uma torre de transmissão infravermelha para realizar a transferência dos dados, necessitando ainda a conexão com o PC via porta serial, tecnologia esta que não se encontra mais nos PC's atuais. Possui ainda, um número limitado de entradas e saídas para conexão de sensores e atuadores e quando as pilhas são retiradas do controlador, o mesmo perde o programa transferido pelo usuário.

Entre outras razões para a escolha desta plataforma estão a questão da praticidade na comunicação e programação com o computador, o aumento do número de dispositivos que podem ser ligados no controlador, podendo com isso desenvolver montagens mais elaboradas e interativas e também por apresentar grande variedade de sensores desenvolvidos para a plataforma Arduino e sua vasta disponibilidade nas lojas de peças eletrônicas. Na Figura 16 são ilustradas as especificações técnicas da plataforma do Arduino UNO.

Figura 16- Especificações técnicas Arduino UNO

Microcontrolador	ATmega328P
Tensão de entrada (recomendada)	7 - 12 V
Tensão de entrada (limite)	6 -20 V
Tensão de Funcionamento	5 V
Pinos I/O Digitais	14
Pinos I/O Digitais de PWM	6
Pinos de Entradas Analógicas	6
Corrente DC por pino I/O	20 mA
Corrente DC por pino com 3.3 V	50 mA
Memória Flash	32 KB
Memória SRAM	2 KB
Memória EEPROM	1 KB
Velocidade de Clock	16 MHz
Comprimento	68.6 mm
Largura	53.4 mm
Peso	25 g

Fonte: Adaptado de Arduino.cc (2016)

Conforme Stevan Jr (2015), o microcontrolador presente na placa Arduino UNO é o Atmel ATMEGA328, um dispositivo de 8 bits, com arquitetura RISC[2] avançada e com encapsulamento DIP28[3]. Ele dispõe de 32 KB de memória flash (sendo 512 Bytes utilizados para executar um programa padrão de inicialização), 2 KB de RAM e 1 KB de EEPROM, um cristal de 16 MHz, um comparador analógico interno e diversos *timer's*, além de 6 PWM's.

O programa a ser desenvolvido para o Arduino UNO é escrito em um ambiente de desenvolvimento integrado (IDE), que já possui um compilador integrado, responsável pela conversão do código em linguagem de máquina. A transferência do programa do computador para o Arduino é realizada pelo próprio microcontrolador auxiliar, que realiza a comunicação com o computador através do USB. Porém, a comunicação também pode ser realizada através do conector ICSP (Circuito de Programação Serial), sendo necessário um compilador externo específico para a família ATMEL (STEVAN, 2015).

[2] Pequeno conjunto de instruções simples que são executadas diretamente pelo *hardware* (SICA, 1999).
[3] Invólucro protetor de um circuito integrado, o número 28 representa o número de pinos do CI (SIARKOWSKI, 2009).

A placa Arduino UNO contém ainda 14 pinos que podem ser configurados como entrada ou saída digitais, especificados para trabalhar com 5 V, cada um fornecendo ou recebendo uma corrente de 20 mA. A fim de evitar danos ao microcontrolador não deve ser excedido os 40 mA em qualquer entrada ou saída. Alguns desses pinos possuem funções especiais, como mostrado na Figura 17.

Figura 17- Especificações dos pinos com configurações especiais do Arduino UNO

Função	Pinagem Arduino UNO	Descrição
PWM	3,5,6,9,10 e 11	Podem ser usados como saídas PWM de 8 bits.
Comunicação serial	0 e 1	podem ser utilizados para comunicação serial.
Interrupção externa	2 e 3	Podem ser configurados para gerar uma interrupção externa.

Fonte: Adaptado Arduino.cc (2016)

Para interface analógica, a placa Arduino UNO apresenta 6 canais, os quais operam com um conversor analógico-digital com resolução de 10 bits. Assim, quando a entrada estiver com 5 V o valor da conversão analógica-digital será 1023.

Possui ainda três pinos com diferentes valores de tensão de alimentação, os quais podem ser utilizados para ligar módulos externos que necessitam um padrão distinto de alimentação, e também possui uma entrada para *reset* externo. Os pinos de alimentação são especificados na Figura 18.

Figura 18- Especificações dos pinos de alimentação do Arduino UNO

Função	Descrição
IOREF	Fornece uma tensão de referência para módulos externos.
RESET	Utilizado para um *reset* externo da placa Arduino.
3.3 V	Fornece uma tensão de 3,3 V para alimentação de módulos externos.
5 V	Fornece uma tensão de 5 V para alimentação de circuitos externos.
GND	Pinos de referência, terra.
VIN	Alimenta a placa através de bateria externa. Quando é alimentada através do conector Jack, esse pino terá a tensão da fonte.

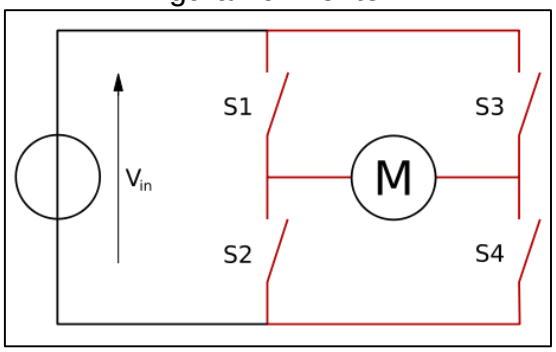
Fonte: Adaptado Arduino.cc (2016)

2.3.2 Ponte H

É um sistema inversor baseado em eletrônica de potência, no qual se obtém o controle de motores ou geradores CC pela tensão do terminal de armadura. Funciona com o chaveamento de componentes eletrônicos, geralmente utilizando o

método PWM para determinar o módulo da tensão no circuito, a polaridade, além do sentido da corrente. Recebe o nome de Ponte H devido a representação gráfica do circuito (UMANS, 2014), conforme é demonstrada na Figura 19.

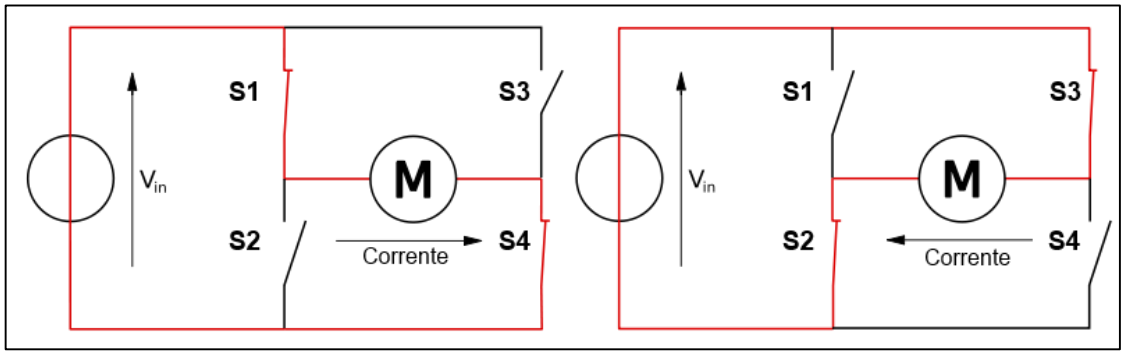
Figura 19- Ponte H



Fonte: Arduinoecia.com.br (2016)

Segundo a Figura 20, observa-se que ao acionar as chaves S1 e S4, a corrente começa a fluir no circuito, da esquerda para a direita (sentido convencional), acionando o motor. Ao desligar as chaves S1 e S4 e ligar S2 e S3, o sentido da corrente passa a ser da direita para a esquerda, invertendo o sentido de rotação do motor. Entretanto S1 e S2 não podem ser comutadas ao mesmo tempo, nem S3 e S4. Neste caso a fonte de alimentação entraria em curto circuito (HART, 2012).

Figura 20- Acionamento das Chaves

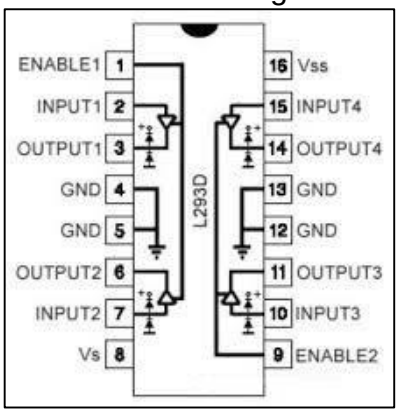


Fonte: Arduinoecia.com.br (2016)

De maneira análoga, ao acionar as chaves S1 e S3 ou S2 e S4, ocasiona-se um curto circuito nos terminais do motor, que passa a se comportar como um gerador, quando apresenta movimento em seu eixo. Com a falta de corrente o torque necessário para manter o eixo girando aumenta, gerando assim uma resistência ao movimento. Esse efeito é desejado quando se faz necessário frear o motor.

No projeto será utilizado o circuito integrado L293D, demonstrado na Figura 21, modelo escolhido por possuir *drive's* de corrente bidirecional, que funcionam como ponte H. Apresenta diodos de alta velocidade ligados na saída, incorporados internamente no CI, responsáveis por proteger o componente, extinguindo qualquer corrente induzida pelo motor. Possui ainda transistores para fazer o acionamento das chaves descritas anteriormente. O circuito da Ponte H ainda poderia ser construído com os componentes citados. Este CI apresenta um tamanho compacto, representando uma grande vantagem na montagem do circuito para controle do motor. Além disso, ele é composto por duas Pontes H, onde é possível controlar dois motores com apenas um circuito integrado.

Figura 21- Circuito Integrado L293D



Fonte: Arduinoecia.com.br (2016)

Conforme a ficha de dados técnicos (ver Anexo A), o L293D admite tensões de alimentação de 4,5 V à 36 V, permitindo controlar variados tipos de motores. O CI suporta uma corrente de saída de 600 mA por canal, ou seja, é possível ligar até dois motores de 600 mA cada. Entretanto, por questão de segurança é recomendado utilizar motores com correntes menores, mesmo o CI sendo capaz de superar picos de corrente de 1,2 A.

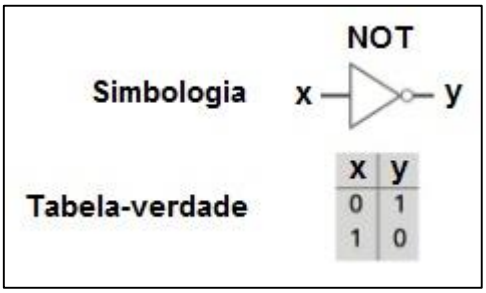
2.3.3 Circuito lógico digital

Em 1854, o matemático Georg Boole desenvolveu uma estrutura para utilizar os métodos algébricos na normatização da lógica e raciocínio humano, publicado no trabalho *Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities* (Investigação das Leis do

Pensamento, em que são fundadas as Teorias Matemáticas de Lógica e Probabilidades) (FLOYD, 2007). Essa estrutura é chamada de “Álgebra Booleana”, a qual foi aplicada primeiramente por Claude Shannon, em 1938, visando a análise e projeto de circuitos lógicos digitais expressos através de valores binários (0 e 1), chamados de portas lógicas.

A porta lógica NOT tem uma entrada “x” e uma saída “y”, no qual o nível lógico de “y” será sempre o oposto do nível lógico de “x”, por esta característica também é chamada de inversor. O comportamento da porta NOT, elencando a saída para cada entrada possível, é denominada de tabela-verdade (VAHID, 2008). Sua representação e a simbologia da porta são ilustradas na Figura 22.

Figura 22 - Simbologia e Tabela-verdade



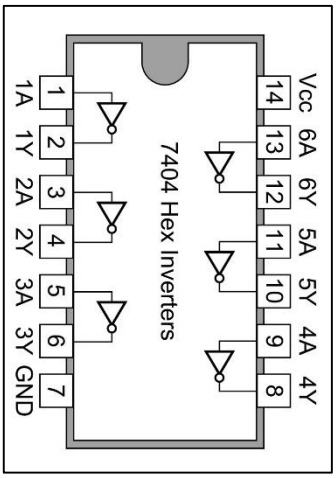
Fonte: Adaptado de Vahid (2008)

Segundo Lima (2016), os circuitos lógicos podem ser implementados utilizando tecnologias diferentes, de acordo com o componente utilizado na fabricação do seu circuito integrado, o qual possibilita a colocação de diversos componentes interligados em um único invólucro, surgindo então as famílias lógicas. As mais utilizadas em equipamentos digitais são as famílias TTL (Lógica Transistor-Transistor) e CMOS (Semicondutor de metal-óxido complementar).

A família TTL foi a primeira a surgir, desenvolvida pela Texas Instruments, na qual utiliza transistores bipolares como elemento principal do circuito. Já a família CMOS utiliza transistores unipolares construídos através de semicondutores, gerando vantagens sobre a tecnologia TTL como, fabricação simplificada, menor custo de produção, *chip’s* menores, alta densidade de integração dos CI’s, baixo consumo de energia, maior imunidade a ruídos, mas também possui desvantagens como, velocidade de operação parcialmente baixa e propenso a danos causados pela eletricidade estática (LIMA, 2016).

No projeto será utilizado o circuito integrado SN7404, da família TTL, demonstrado na Figura 23, o qual possui internamente transistores e resistores responsáveis pela inversão do sinal. O CI apresenta um tamanho compacto, ideal na montagem do circuito, além de ser composto por seis portas lógicas no mesmo invólucro. Para consulta de dados técnicos verificar Anexo B.

Figura 23- Circuito Integrado SN7404



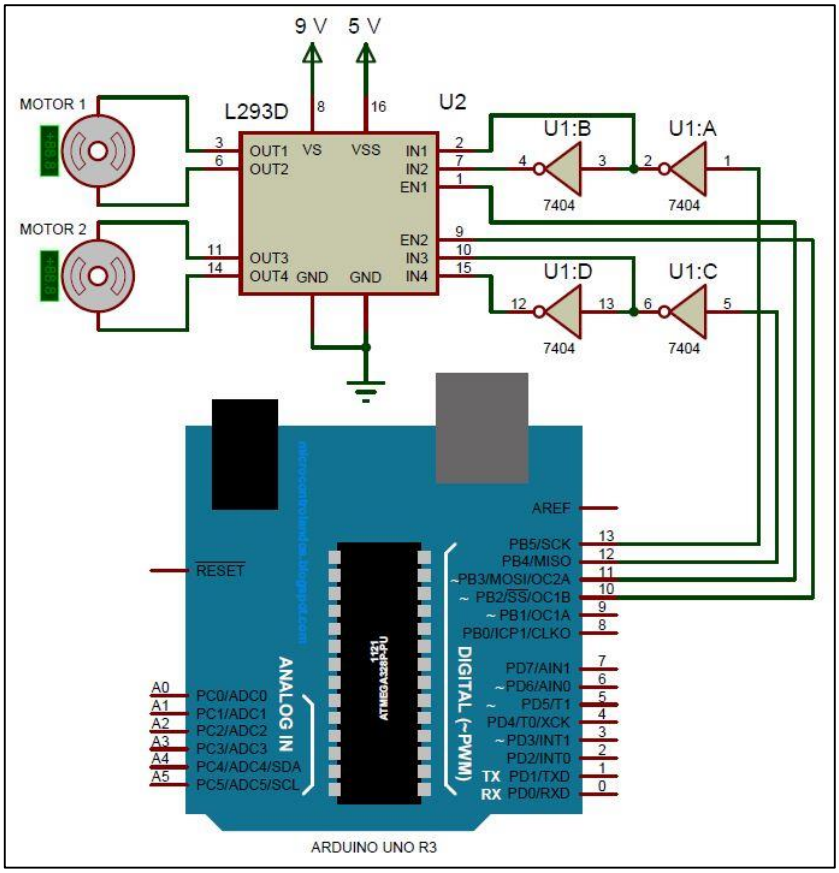
Fonte: Datasheet Texas Instruments (2004)

2.4 FUNCIONAMENTO DA PLACA AUXILIAR

Com o programa já gravado no Arduino Uno, o mesmo será responsável em acionar um pino de saída digital, já definido para a placa auxiliar, com um estado lógico *LOW* ou *HIGH*. Esse pino será ligado na entrada de uma porta lógica NOT, no CI SN74LS04, conectada em série com outra do mesmo tipo, afim de produzir dois sinais lógicos na saída, que serão conectadas nos pinos “INPUT” do CI L293D, a ponte H. Esses pinos serão responsáveis por controlar o sentido de giro do motor. Com o intuito de controlar também a velocidade, uma saída PWM do Arduino é ligada no pino "ENABLE” da ponte H, onde serão aplicadas diferentes larguras de pulso de uma onda quadrada para determinar a velocidade do motor.

O esquema elétrico do *hardware* descrito anteriormente, é ilustrado na Figura 24, no qual é demonstrado apenas uma parte da ligação para melhor visualização. As portas lógicas e ponte H, responsáveis pelo acionamento de dois motores, são duplicadas e conectadas nas demais saídas do Arduino, possibilitando assim a conexão de até quatro motores (M1, M2, M3 e M4).

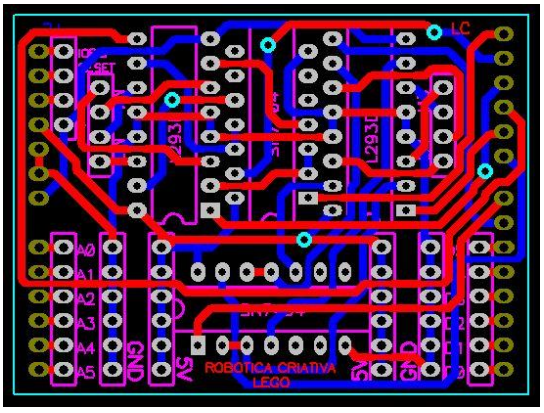
Figura 24- Esquema Elétrico do *Hardware*



Fonte: O autor (2016)

Após a concepção do esquema elétrico foi desenvolvido, o layout da placa de circuito impresso auxiliar conforme mostrado na Figura 25, utilizando o *software* PCAD2006. A confecção da placa auxiliar foi realizada por uma empresa terceirizada. Foi especificado o material, as cores de tinta isolante e as cores dos textos dos componentes iguais as da placa do Arduino, afim de manter as mesmas características.

Figura 25- *Layout* Placa Auxiliar



Fonte: O autor (2016)

As conexões disponíveis para a placa auxiliar, bem como suas descrições são mostradas na Figura 26. A placa possui ainda *slot's* para alimentação (5 V) e aterramento (GND), visando auxiliar na ligação dos dispositivos.

Figura 26- Especificações das Conexões da Placa Auxiliar

Função	Descrição
M1, M2, M3 e M4	Ligação dos Motores
IOREF, 3.3V e Vin	Pinos de Alimentação
A0, A1, A2, A3, A4 e A5	Entradas Analógicas
D0, D1, D2, D3, D4 e D5	Entradas ou Saídas Digitais
D0 e D1	Comunicação Serial
D2 e D3	Interrupção Externa
D3 e D5	Saída PWM
RESET	Reset Externo

Fonte: O autor (2016)

2.5 RESULTADOS

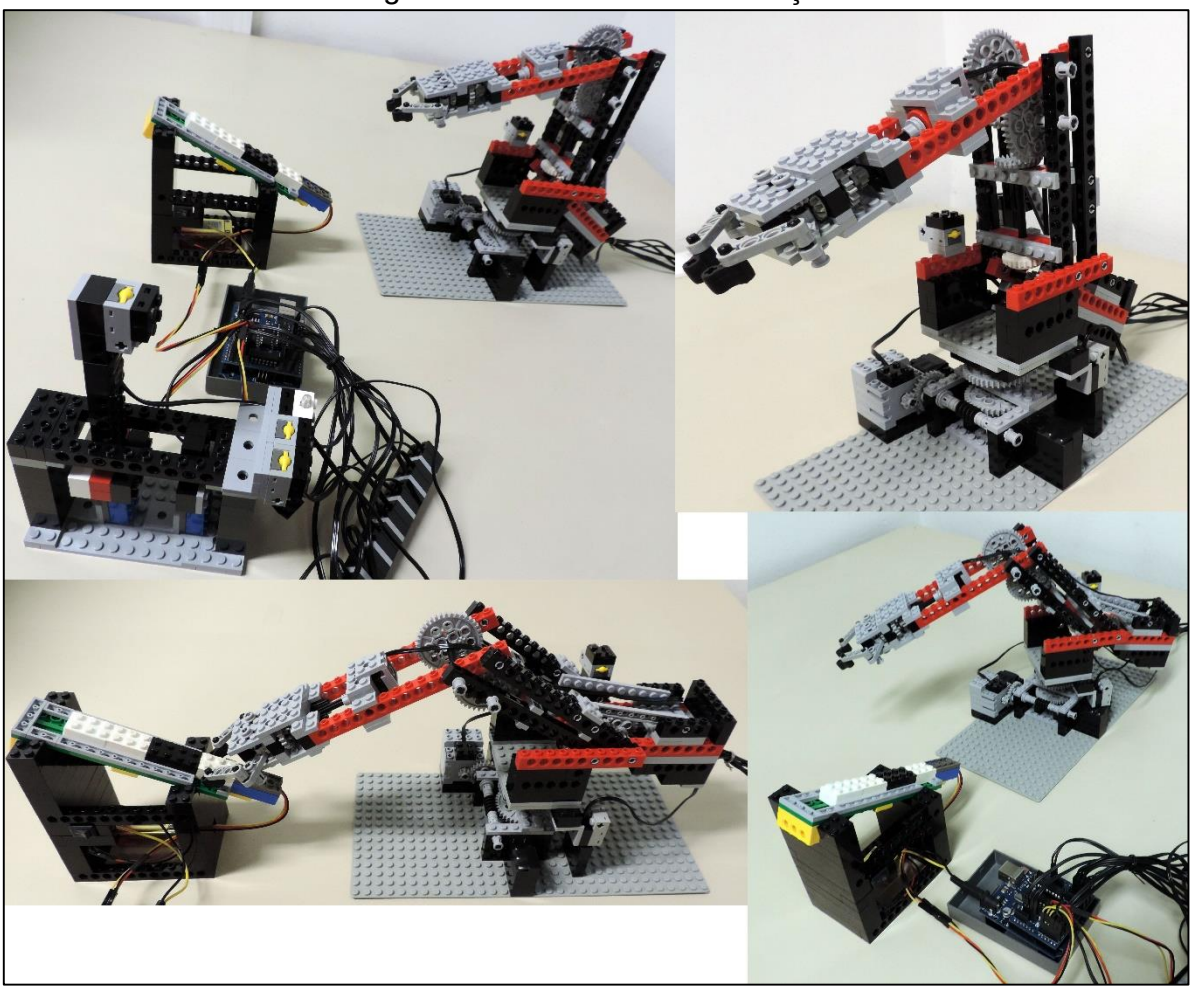
O desenvolvimento e implementação do projeto proposto ocorreu com êxito, na análise de *hardware* dos componentes do *kit* LEGO® *Mindstorms* RCX 1.0, na integração das plataformas, na criação de um manual didático para auxiliar na programação do Arduino UNO, mas principalmente no acréscimo considerável de conexões disponíveis para ligação de sensores e atuadores por intermédio da placa auxiliar.

No *kit* LEGO® é possível conectar ao controlador RCX três sensores e três motores, em conexões já estabelecidas pelo fabricante. Já na placa auxiliar, acoplada ao Arduino é possível ligar quatro motores e seis sensores analógicos em conexões já estabelecidas. Também possui seis conexões digitais que podem ser configuradas via *software* como entradas ou saídas, tendo duas dessas conexões que podem ser configuradas como saídas PWM, outras duas como entradas para interrupções externas e ainda, mais duas que podem ser utilizadas para comunicação serial.

Após o projeto concluído, foi realizada a montagem de um classificador de peças automático, ilustrado na Figura 27, no qual identifica se as peças são de cor branca ou preta utilizando um sensor infravermelho. Com o auxílio de um braço robótico, composto por três motores, que lhe conferem três graus de liberdade, pega cada peça e de acordo com a cor, destina a mesma à direita do braço (peça preta) ou

à esquerda (peça branca). O acionamento dos motores é controlado por sensores de fim de curso. O sistema possui ainda um modo de operação manual, que é alternado com o modo automático através de dois botões e sinalizado com uma lâmpada. No modo manual, o braço robótico é controlado através de um *joystick*, que possui dois sensores infravermelho para realizar a movimentação do braço e um botão para abrir e fechar a garra. Essa montagem tem por objetivo demonstrar uma aplicação que demande a utilização de diversos dispositivos, a qual não seria possível utilizando o controlador RCX.

Figura 27- Classificador de Peças

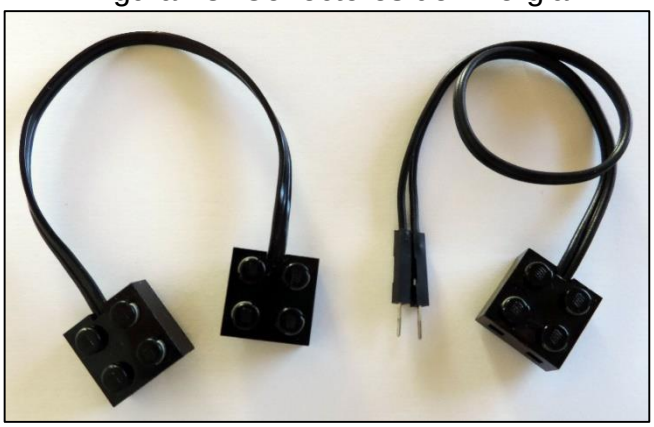


Fonte: O autor (2016)

Devido aos diferentes padrões de conectores presentes em cada plataforma, os cabos de alimentação e sensoriamento ligados entre a placa auxiliar do Arduino e os periféricos, passaram por alterações. No padrão LEGO®, os cabos possuem em suas extremidades conectores de encaixe, utilizados para ligar o RCX nos blocos dos periféricos. Já no projeto implementado, se manteve um conector de encaixe em uma

extremidade para ligação de periféricos LEGO® e na outra foi adicionado conectores do tipo “macho” para permitir a ligação com a placa auxiliar. A Figura 28 demonstra a alteração dos conectores.

Figura 28- Conectores de Energia

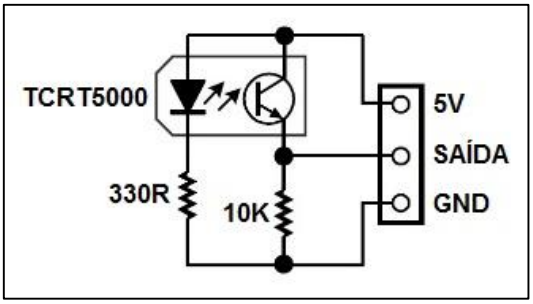


Fonte: O autor (2016)

Outra alteração realizada foi no sensor de luz do LEGO®, que funciona através da variação de corrente elétrica (sensor passivo). As entradas do Arduino porém, funcionam com a variação de tensão portanto, não foi possível realizar a integração do mesmo com a plataforma Arduino. Em sua substituição, foi utilizado o sensor óptico reflexivo TCRT 5000, que consiste de um diodo emissor de infravermelho e um fototransistor que irá receber o sinal quando ocorrer uma reflexão (OLIVEIRA, 2015), e por ser um sensor do tipo ativo, terá três fios, um para alimentação, um para o GND e outro para saída de sinal.

O circuito elétrico para funcionamento do TCRT 5000 é simples, necessitando apenas de dois resistores para limitar a corrente no diodo e no fototransistor, e os fios de alimentação e saída de sinal do sensor. Para acessar os dados técnicos verificar o Anexo C. A Figura 29 apresenta o esquema elétrico para ligação do sensor.

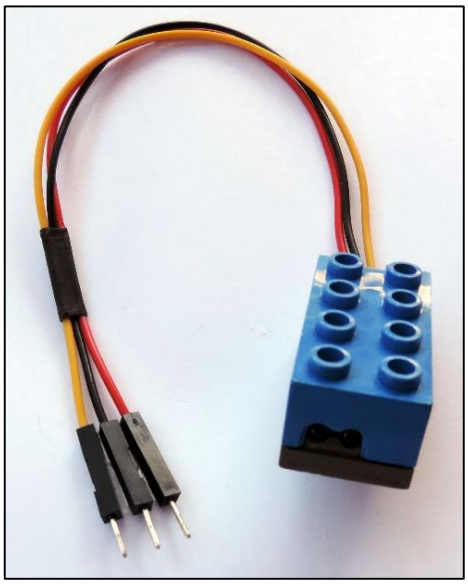
Figura 29- Esquema Elétrico TCRT 5000



Fonte: O autor (2016)

Visando o aproveitando da mesma estrutura LEGO® do sensor de luz para inserir o novo circuito do TCRT 5000, foi desmontado o bloco e substituído o sensor conforme visto na Figura 30. Também foi alterado o cabo de alimentação e saída do sensor, deixando os conectores de acordo com o padrão da placa auxiliar.

Figura 30- Sensor Óptico Reflexivo TCRT 5000



Fonte: O autor (2016)

CONSIDERAÇÕES FINAIS

A robótica educacional mostra ter cada vez mais potencial como ferramenta pedagógica no processo de aprendizagem, por ser uma atividade dinâmica que possibilita o desenvolvimento de competências e habilidades. Com os estudantes cada vez mais conectados em ambientes computacionais, a robótica é uma forma de aliar a tecnologia com a construção do conhecimento, aproximando aplicações reais ao contexto didático, potencializando a interdisciplinaridade, as tomadas de decisão e resolução de problemas.

Com o desenvolvimento desse trabalho, o qual contempla um conceito atual de programação, familiar a grande parte dos estudantes, se buscará uma ligação mais forte entre a parte teórica e prática. Elucidando os conteúdos, assimilando ou reforçando a aprendizagem em programação, através de inúmeras montagens e aplicações robóticas utilizando a plataforma Arduino UNO.

Os objetivos propostos foram alcançados de forma satisfatória, mesmo o *kit* LEGO® apresentando uma tecnologia praticamente extinta, foi possível realizar a integração com a plataforma do Arduino UNO. Apesar de não ter sido possível a integração com o sensor de luz, o problema foi resolvido com a substituição do mesmo por outro sensor compatível com o Arduino, com leitura mais precisa e implementado de forma simplificada.

Foram produzidas uma dezena de placas auxiliares, que ficarão disponíveis para montagem e uso, em futuros cursos de robótica disponibilizados pela instituição de ensino UCS, bem como o manual didático para auxiliar na programação do Arduino que, aliado ao *kit* LEGO®, será uma ferramenta de estímulo para criações robóticas cada vez mais complexas e criativas.

Como trabalhos futuros sugere-se: a utilização de sensores disponíveis para a plataforma Arduino, afim de desenvolver novas possibilidades de montagens robóticas e com maior interação com o meio; a inserção desses sensores em blocos LEGO® a serem confeccionados por impressão 3D, com o intuito de seguir o mesmo padrão de montagem das demais peças e também o desenvolvimento de uma linguagem gráfica de programação para o Arduino.

REFERÊNCIAS

ARDUINO. **Produtos.** Disponível em: <https://www.arduino.cc/en/main/arduinoBoard Uno#>. Acesso em 17 out. 2016.

ARDUINOECIA. **Controle de motor CC com o L293D - Ponte H**. Disponível em: <http://www.arduinoecia.com.br/2014/04/controle-de-motor-cc-com-o-l293d-ponte-h.html>. Acesso em: 27 out. 2016.

CABRAL, Cristiane Pelisolli. **Robótica Educacional e Resolução de Problemas:** uma abordagem microgenética da construção do conhecimento. Dissertação de mestrado, Programa de Pós-graduação em Educação, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2010.

CAVALCA, Eduardo; BASTOS, Guilherme. **Utilizando a Plataforma Mindstorms para o Desenvolvimento de um Agente Autônomo Inteligente**. Universidade Federal de Itajubá, Minas Gerais, 2006.

CIRCUITAR. **Programação para Arduino:** Primeiros Passos. Disponível em: <https://www.circuitar.com.br/tutoriais/programacao-para-arduino-primeiros-passos/ #funo>. Acesso em: 12 nov. 2016

CONCHINHA, Cristina; FREITAS, João Correia de. **Robots & Necessidades Educativas Especiais:** A Robótica Educativa Aplicada a Alunos Autistas. In: Atas da IX Conferência Internacional de TIC na Educação, 2015, Portugal. **Anais** Challenges 2015: Meio Século de TIC na Educação. Universidade do Minho, Centro de Competência TIC do Instituto de Educação, Braga, Portugal, 2015. p. 23-26.

EMBARCADOS. **Arduino UNO.** Disponível em: <http://www.embarcados.com.br/ arduino-uno/>. Acesso em 18 out. 2016.

ESCOLADEROBOTICA. **Tutorial de Programação Arduino**. Disponível em: <http://escoladerobotica.ipcb.pt/?page_id=297>. Acesso em: 12 nov. 2016

FLOYD, Thomas. **Sistemas Digitais:** Fundamentos e Aplicações. 9. ed. Porto Alegre: Bookman, 2007.

FULBER, Cassiana Chassot; SCHOLEM, Tyron Wittée Neetzow. **Introdução ao LEGO RCX e ao Programa RoboLab.** Universidade Federal do Rio Grande do Sul, Porto Alegre, 2008.

HART, Daniel W. **Eletrônica de Potência:** Análise e Projetos de Circuitos. Porto Alegre: AMGH, 2012.

LIMA, José Antônio Gomes de. **Circuitos Integrados e Famílias Lógicas**. Universidade Federal da Paraíba. Disponível em: < http://www.di.ufpb.br/jose/ familias.pdf>. Acesso em: 01 nov. 2016.

LÓPEZ, Estela Díaz; GÓMEZ, David Gualda; RODRIGO, Luis de Santiago et al. **Introducción al diseño de microrobots móviles.** Univerdidad de Alcalá, Madrid, 2006.

MELO, Mário Marcelino Luís de. **Robótica e Resolução de Problemas:** Uma Experiência com o Sistema Lego *Mindstorms* no 12° ano. Dissertação de mestrado, Área de Especialização em Tecnologias Educativas, Universidade de Lisboa, Lisboa, 2009.

MOHAN, Ned. **Máquinas Elétricas e Acionamentos** - Curso Introdutório. 1. ed. Rio de Janeiro: LTC, 2015.

MONK, Simon. **Programação com Arduino:** Começando com Sketches. 1. ed. Porto Alegre: Bookman, 2013.

NASCIMENTO JR, Geraldo Carvalho do. **Máquinas Elétricas**. 1. ed. São Paulo: Érica, 2014.

NETO, Amadeu Zanon. **Robótica:** Estudo de Peças e Componentes. Araçatuba, São Paulo, 2008. (Apostila).

OLIVEIRA, André Schneider de, ANDRADE, Fernando Souza de. **Sistemas Embarcados:** Hardware e Firmware na Prática. 2. ed. São Paulo: Érica, 2010.

OLIVEIRA, Cláudio L. Vieira, ZANETTI, Humberto A. Piovesana. **Arduino Descomplicado:** Como Elaborar Projetos de Eletrônica. 1. ed. São Paulo: Érica, 2015.

POLONSKII, Mikhail M. **Introdução à Robótica e Mecatrônica**. 2. ed. Caxias do Sul: EDUCS, 1996.

SANTOS, Carmen Faria; MENEZES, Crediné Silva de. **A Aprendizagem da Física no Ensino Fundamental em um Ambiente de Robótica Educacional**. In: XXV Congresso da Sociedade Brasileira de Computação, 2005, São Leopoldo. **Resumos** A Universidade da Computação: Um Agente de Inovação e Conhecimento. São Leopoldo: UNISINOS, 2005. p. 2-4.

SGS-THOMSON. **Push-Pull Four Channel Driver With Diodes**. Disponível em: <http://pdf1.alldatasheet.com/datasheet-pdf/view/22432/STMICROELECTRONICS/L293D.html>. Acesso em: 03 nov. 2016.

SIARKOWSKI, Acácio Luiz. **Fundamentos de Fabricação de Circuitos Integrados**. Centro Universitário Sant'Anna, São Paulo, 2009. Disponível em: <http://www.lsi.usp.br/~acacio/fpci02_Encapsulamentos.pdf>. Acesso em: 08 out. 2016.

SICA. Carlos. **Estudo Sobre a Arquitetura RISC**. Universidade Estadual de Maringá, Paraná, 1999.

SILVA, Alzira Ferreira da. **RoboEduc:** Uma Metodologia de Aprendizado com Robótica Educacional. Tese de Doutorado, Programa de Pós-graduação em Engenharia Elétrica, Universidade Federal do Rio Grande do Norte, Natal, 2009.

STEVAN JR., Sergio Luiz; SILVA, Rodrigo Adamshuk. **Automação e Instrumentação Industrial com Arduino:** Teoria e Projetos. 1. ed. São Paulo: Érica, 2015.

TEXAS INSTRUMENTS. **SN74LS04 HEX Inverters**. Disponível em: <https://www.ti.com/lit/ds/symlink/sn54ls04-sp.pdf>. Acesso em: 03 nov. 2016.

UMANS, Stephen D. **Máquinas Elétricas de Fitzgerald e Kingsley**. 7. ed. Porto Alegre: AMGH, 2014.

VAHID, Frank. **Sistemas Digitais:** Projeto, Otimização e HDLs. Porto Alegre: Bookman, 2008.

VISHAY SEMICONDUCTORS. **Reflective Optical Sensor with Transistor Output**. Disponível em: < http://pdf1.alldatasheet.com/datasheet-pdf/view/442894/VISHAY/TCRT5000.html>. Acesso em: 10 nov. 2016.

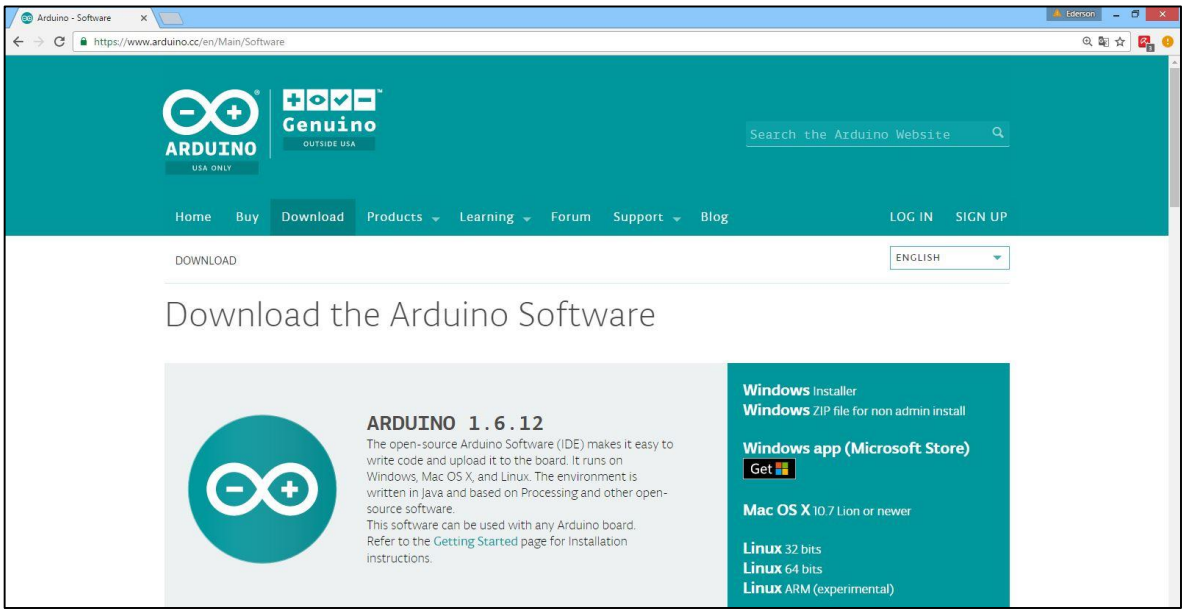
ZILLI, Silvana do Rocio. **A robótica educacional no ensino fundamental:** Perspectivas e práticas. 2004. Dissertação de mestrado, Programa de Pós-graduação em Engenharia de Produção, Universidade Federal de Santa Catarina, Florianópolis, 2004.

APÊNDICE A – MANUAL DIDÁTICO ARDUINO UNO

O *software* Arduino IDE, é um ambiente de programação que permite ao usuário criar códigos e transferi-los para uma placa Arduino visando a automatização de tarefas, com a leitura do meio através de sensores e chaves, e o acionamento de motores, sinais sonoros ou luminosos, neste caso, aplicado a estruturas desenvolvidas em LEGO (STEVAN 2015). A seguir será apresentado o ambiente de desenvolvimento de projetos para a plataforma Arduino.

A ferramenta de programação a ser instalada é gratuita e encontra-se disponível no site do Arduino (<http://arduino.cc/en/Main/Software>). A Figura 31 apresenta a página na qual pode ser realizado o download do *software*.

Figura 31- Site Arduino para realizar download do *software*



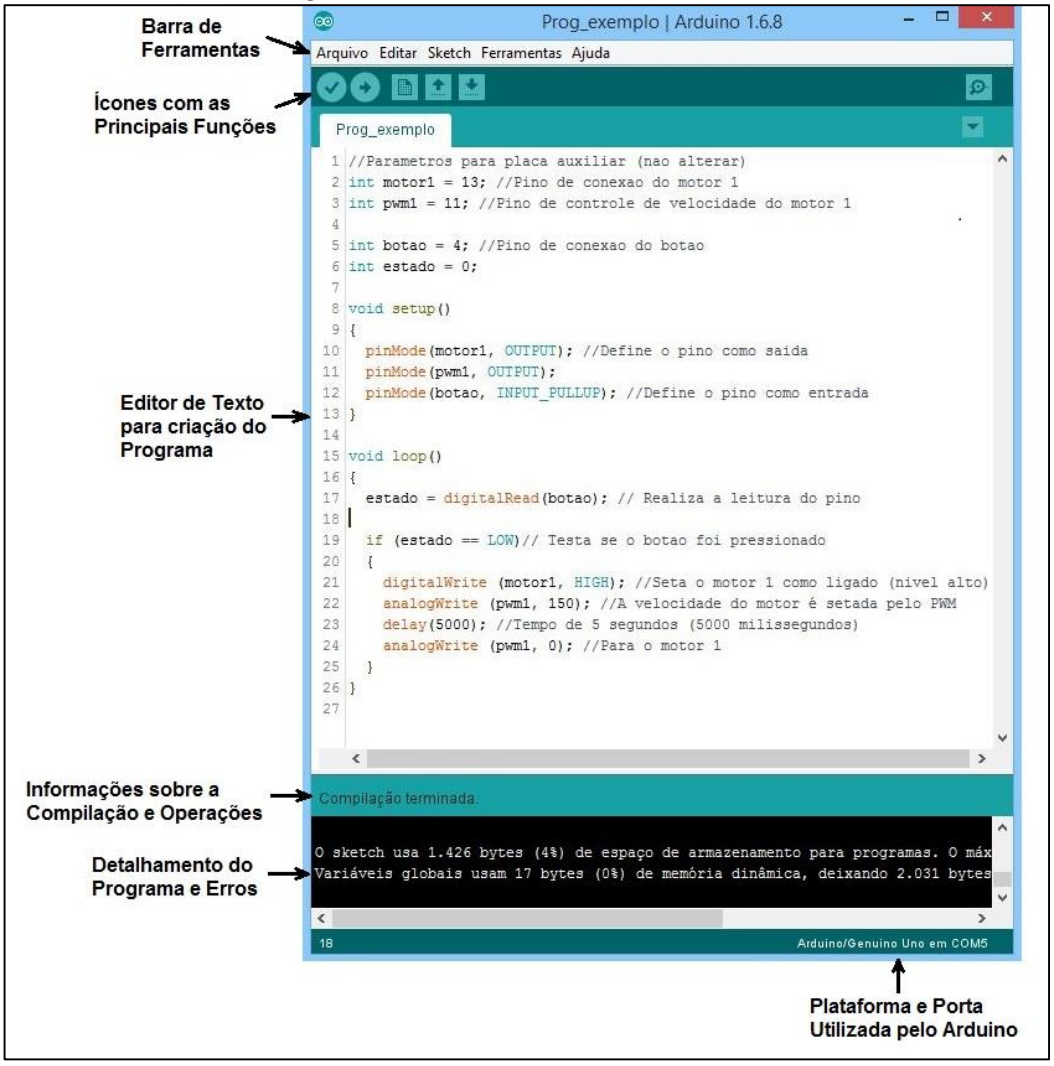
Fonte: O autor (2016)

Conecte a placa do Arduino ao computador utilizando um cabo USB. Ao realizar essa ligação pela primeira vez, o computador detectará o novo dispositivo e realizará a instalação automática dos *drive's* de comunicação. Com a conexão estabelecida, um LED (demarcado por ON) deve acender na placa do Arduino, indicando que há alimentação.

Com o programa devidamente instalado, dirija-se até a pasta que está salvo o *software* Arduino IDE e dê um duplo clique no ícone do Arduino para abrir a interface

de programação, ilustrada na Figura 32. A mesma demonstra ainda, as funções presentes na tela do ambiente.

Figura 32- Interface do Arduino IDE

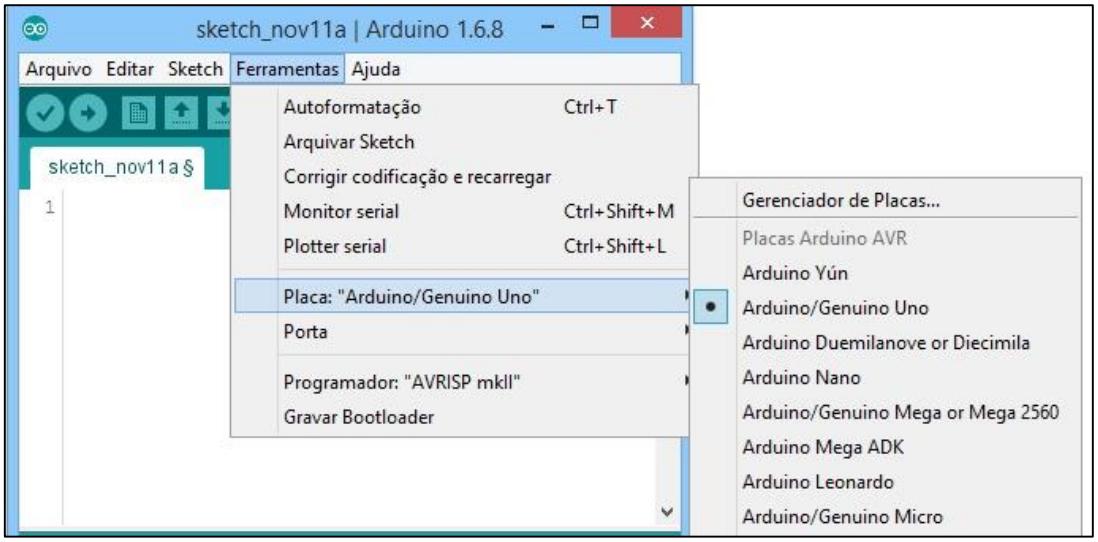


Fonte: O autor (2016)

Conforme observado na figura anterior, a interface de programação é simples, apresentando uma barra de ferramentas e atalhos para acessar diferentes funções. Um editor de texto, no qual é utilizado para criar um projeto composto por comandos, projetos esses chamados de "*sketch*". E uma aba na parte inferior da janela responsável por exibir informações sobre as operações, transferência do programa, erros e qual o modelo e porta de comunicação está sendo utilizada pelo *software*.

Agora deve-se indicar para o *software* qual é a plataforma Arduino que será utilizada. Para isso, clique na aba "Ferramentas", selecione a opção "Placa" para abrir um menu no qual deve-se escolher o modelo de placa correspondente ao seu Arduino. Esse processo é exemplificado na Figura 33.

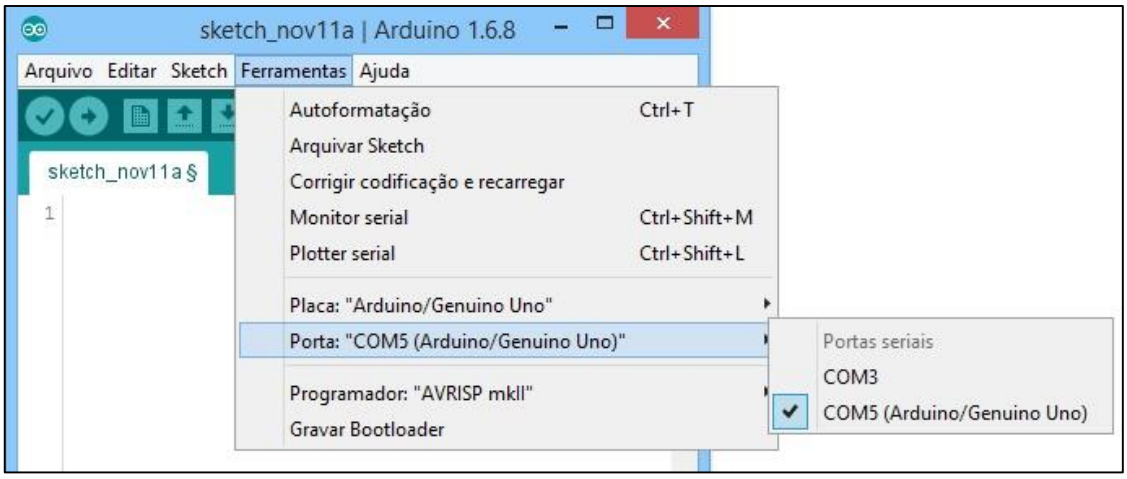
Figura 33- Selecionando a Plataforma Arduino



Fonte: O autor (2016)

O próximo passo é selecionar a porta de comunicação serial a ser utilizada para estabelecer a conexão do *software* com a placa do Arduino. Acesse novamente a aba “Ferramentas”, selecione a opção “Porta”, é provável que a porta utilizada seja COM4 ou superior. Para confirmar, desconecte o Arduino e reabra o menu novamente, a opção de porta que desaparecer é a da placa do Arduino. Reconecte a placa e selecione a porta serial. A Figura 34 demonstra a execução desse passo.

Figura 34- Escolha da Porta de Comunicação



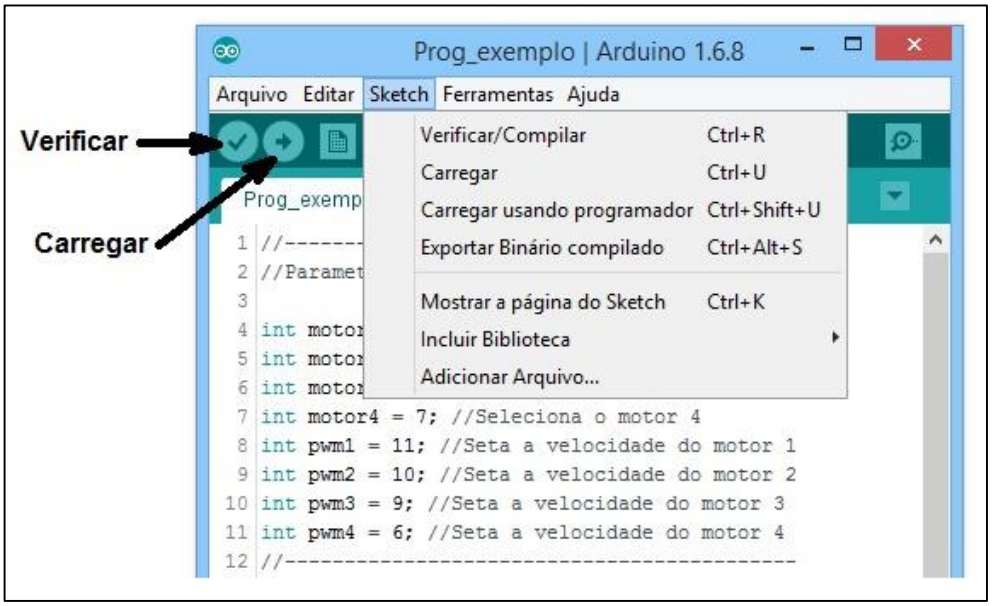
Fonte: O autor (2016)

Após o programa ser finalizado, o mesmo deve ser verificado pelo *software*, afim de analisar possíveis erros de compilação. Para isso deve-se acessar a aba “*Sketch*”, e selecionar a opção “Verificar/Compilar”, ou clicando diretamente na tela, em um ícone que possui o sinal de “correto”. Se o programa estiver certo, na parte

inferior da tela de programação irá aparecer a mensagem “Compilação terminada”, caso contrário, o programa exibirá uma mensagem de erro e indicará a linha que apresenta problema, que deve ser corrigida.

Em seguida o programa deve ser transferido para o Arduino, podendo ser acessado pela aba “*Sketch*”, e selecionando a opção “Carregar”, ou clicando diretamente na tela, em um ícone de uma seta voltada para direita. Se o *software* realizar a transferência corretamente, na parte inferior da tela de programação irá aparecer a mensagem “Carregado”, caso contrário, exibirá a mensagem “Problema ao carregar para a placa”, então deve-se verificar se o Arduino está conectado no PC, se a configuração da plataforma está correta e tentar transferir o programa novamente. A Figura 35 ilustra os comandos de verificação e transferência do *sketch*.

Figura 35- Comandos de Verificação e Transferência



Fonte: O autor (2016)

- CRIAÇÃO DO *SKETCH*

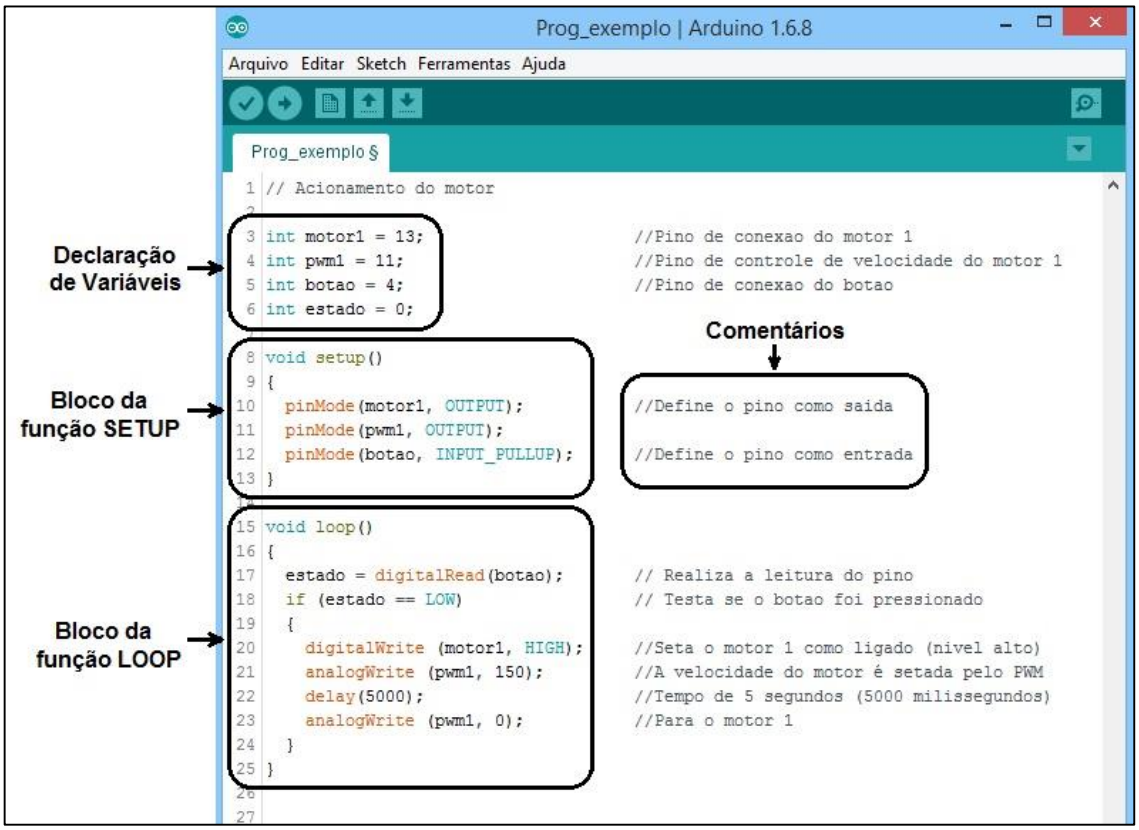
Com o ambiente configurado podemos iniciar a criação do *sketch*. A estrutura do mesmo é formada basicamente por três blocos, o bloco de declaração das variáveis, o bloco da função “*setup*( )”, com declarações, inicializações e funções internas, as quais serão executadas uma única vez e o bloco da função “*loop*( )”, com funções que se repetirão infinitas vezes (STEVAN, 2015).

Conforme Oliveira (2015), na construção dos blocos é necessário utilizar alguns símbolos para permitir o correto funcionamento do código, como o ponto e

vírgula “;”, usado para finalizar cada argumento, as chaves “{ }”, que limitam os argumentos escritos em uma função e o uso de duas barras “//” para atribuir um comentário em alguma linha. Além disso, o programa também difere letras maiúsculas de minúsculas.

O *sketch* demonstrado na Figura 36 é um exemplo de aplicação para o acionamento de um motor, no qual o controlador realiza a leitura de um botão, que ao ser pressionado coloca o motor em funcionamento por cinco segundos e depois o desliga, aguardando novamente o botão ser pressionado para repetir a ação. Na figura são destacados ainda os três blocos principais e pode-se observar a aplicação dos símbolos inseridos no código.

Figura 36- Estrutura do *Sketch*



Fonte: O autor (2016)

Agora vamos analisar o *sketch* que foi desenvolvido para entender sua estruturação no programa. Primeiramente foram declaradas as variáveis, recurso utilizado para determinar quais pinos do controlador serão utilizados, armazenar dados de sensores, ou ainda valores para cálculos.

Na linha 3 temos a declaração de uma variável “motor1”, do tipo inteira, e com valor inicial 13. No bloco “*setup* ( )”, temos uma configuração inicial “pinMode(motor1,

OUTPUT)", a qual indica que a variável do pino 13 está configurada como saída. No bloco "*loop* ( )", tem-se um procedimento de leitura de um pino "estado = digitalRead(botao)", designado para o botão. Já na linha 18 é realizado um teste condicional para saber se o botão foi pressionado, "if (estado == LOW)", se a condição for verdadeira tem-se a atribuição de um valor lógico (*HIGH*) para o motor "digitalWrite (motor1, HIGH)", ao mesmo tempo é definida a velocidade para o mesmo através da saída PWM analógica "analogWrite (pwm1, 150)". Temos ainda um procedimento de temporização "delay(5000)", estipulado em 5000 ms (5s).

- TIPOS DE DADOS E OPERADORES

A linguagem de programação contém um grupo padrão de dados que determinam o gênero do valor a ser trabalhado. Com esses valores armazenados nas variáveis pode-se realizar operações aritméticas e também usá-las em sentenças por meio dos operadores lógicos (STEVAN, 2015). Na Figura 37 serão demonstrados os tipos de dados, os operadores lógicos e aritméticos mais utilizados na programação.

Figura 37- Tipos de Dados e Operadores

Tipos de Dados	
boolean	valor verdadeiro (true) ou falso (false);
char	um caractere;
byte	sequência de 8 bits;
int	número inteiro de 16 bits com sinal (-32.768 a 32.767);
unsigned int	número inteiro de 16 bits sem sinal (0 a 65.535);
long	número inteiro de 32 bits com sinal (-2.147.483.648 a 2.147.483.647);
unsigned long	número inteiro de 32 bits sem sinal (0 a 4.294.967.295);
float	número real de precisão simples (ponto flutuante);
double	número real de precisão dupla (ponto flutuante);
string	sequência de caracteres;
void	tipo vazio (não tem tipo).

Operadores Aritiméticos		Operadores de Comparação	
=	atribuição	==	igual que
+	soma	!=	diferente que
-	subtração	<	menor que
*	multiplicação	>	maior que
/	divisão	<=	menor ou igual que
%	módulo	>=	maior ou igual que

Operadores Compostos		Operadores booleanos	
++	incremento	&&	and
--	decremento	||	or
		!	not

Fonte: Baseado em Stevan (2015)

- FUNÇÕES

Uma função é uma sequência de comandos usadas para criar pequenos pedaços de códigos separados do programa principal (fora da função “*loop*( )”). Pode ser reutilizada inúmeras vezes ao longo de um programa, poupando tempo de programação e linhas de código, além de ajudar a fragmentar o código em partes menores (ESCOLADEROBOTICA.IPCB.PT).

pinMode ( )

A função é utilizada para especificar se o pino se comportará como uma entrada ou uma saída digital. É preciso informar o número do pino ou o nome de sua atribuição, ao qual se deseja configurar e em seguida, determinar se será uma entrada ou uma saída, conforme o exemplo “pinMode(pwm1, OUTPUT);”.

Na utilização de botões se faz necessário habilitar, via programação, um resistor de “*pull-up*” interno do Arduino, pois quando o botão não estiver pressionado, o pino de entrada fica flutuando, podendo produzir uma leitura falsa. Ao habilitar esse resistor a entrada passará a ter 5 V e quando o botão for pressionado, ela se sobrepõe ao resistor e forçará a entrada para 0 V. Para isso, usa-se a declaração conforme o exemplo “pinMode(botao, INPUT_PULLUP);”.

digitalWrite ( )

Utilizada para escrever um valor *LOW* (0 V) ou *HIGH* (5 V) em um determinado pino. É necessário informar o número do pino ou o nome de sua atribuição, que será configurado e, após determinar seu nível lógico, de acordo com o exemplo “digitalWrite (motor1, HIGH);”.

digitalRead( )

Efetua a leitura de um pino digital específico, que retorna um valor *LOW* ou *HIGH*, que é informado o pino que efetuará a leitura ou o nome lhe dado, e o seu valor será atribuído a uma variável já declarada no início do *sketch*, podendo ser escrita segundo o exemplo “valor=digitalRead(botão);”.

analogWrite ( )

Essa função realiza a escrita analógica, utilizando as portas de saída PWM para gerar uma onda quadrada, na qual é possível configurar o tempo que a mesma permanece em nível lógico alto (*HIGH*), com valores entre 0 e 255. Para usá-la é necessário informar o número do pino ou o nome de sua atribuição que será configurado e, em seguida, determinar o valor analógico, conforme o exemplo “analogWrite (velocidade, 150);”.

analogRead ( )

Função capaz de ler o valor de um pino analógico, através do uso de um conversor analógico-digital de 10 bits, permitindo mapear voltagens de entrada de 0 V a 5 V para valores inteiros entre 0 e 1023. Para ser expressa, informa-se o pino que efetuará a leitura ou o nome do mesmo, e o seu valor será atribuído a uma variável já declarada no início do *sketch*, de acordo com o exemplo “valor=analogRead(3);”.

delay( )

Função de tempo responsável por gerar um atraso na sequência de execução do programa por um tempo específico, em milissegundos, o qual é informado conforme o exemplo “delay(5000);”, gerando um tempo de parada de cinco segundos.

millis( )

Retorna o tempo em milissegundos a partir do momento em que o Arduino começou a execução do programa. Essa função é melhor que a descrita anteriormente, pois não para o programa ao efetuar a contagem do tempo. Para sua utilização necessita da declaração, no início do *sketch*, de uma variável do tipo real e sem sinal (unsigned long tempo=0;), e usada segundo o exemplo “tempo = millis( );”.

Serial.begin( )

Função que habilita a comunicação serial entre o Arduino e o PC, possibilitando o monitoramento em tempo real de variáveis específicas. Sua

declaração deve ser feita dentro da função “*setup*( )” e informada a taxa de transferência em bits por segundo para realizar a transmissão de dados, a qual possui valor usual conforme exemplo “Serial.begin(9600);”.

Serial.print ( ) e Serial.println ( )

Complementa a função anterior, escrevendo na serial um texto em formato ASCII, de acordo com o exemplo “Serial.print ("VALOR: ");”. Pode-se também escrever o valor de variáveis, informando qual delas se deseja supervisionar, segundo o exemplo “Serial.println(leitura_temp);”, a qual também acrescenta uma quebra de linha ao dado escrito. A Figura 38 ilustra uma aplicação para este comando.

Figura 38- Comando Serial.print( )

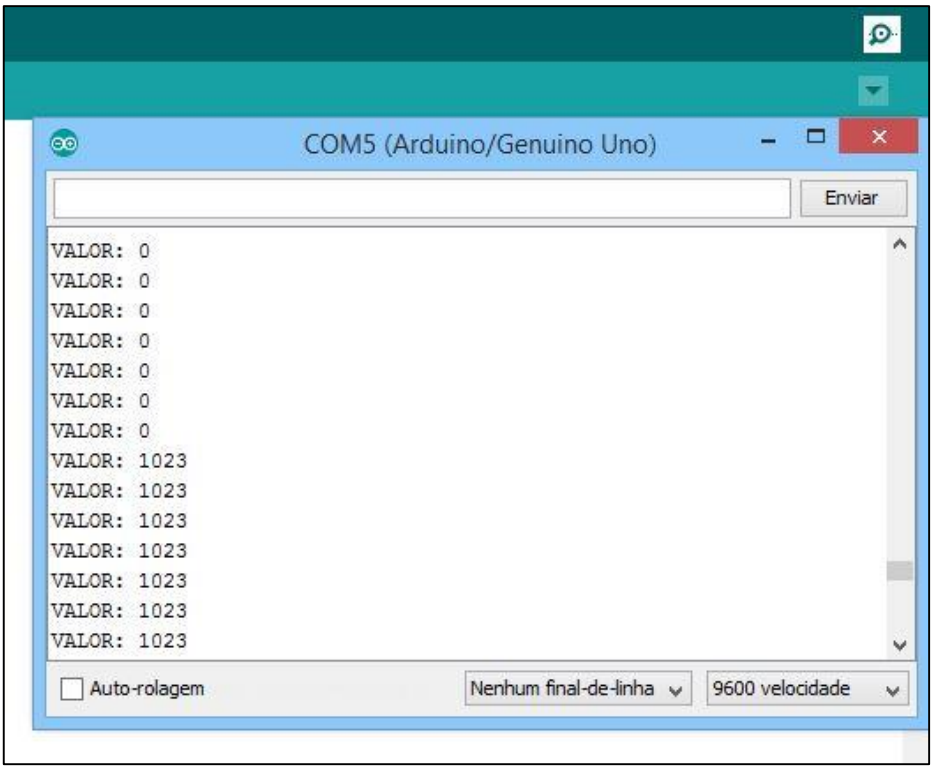
exemplo_serial

```
1  int sensor = A0; // Pino analógico "0" do Arduino onde será conectado o sensor
2
3  int leitura_sensor = 0; // Declara variável e a inicializa com valor igual a "0"
4
5  void setup() {
6    pinMode(sensor, INPUT); // Define pino como entrada
7
8    Serial.begin(9600);    // Estabelece a comunicação serial
9  }
10
11 void loop() {
12
13   leitura_sensor = analogRead(sensor); // Realiza a leitura do sensor
14
15   Serial.print ("VALOR: ");        // Escreve o conteudo adicionado entre aspas "VALOR: "
16   Serial.println(leitura_sensor); // Escreve o valor lido pelo sensor
17   delay(100);                     // Repete a escrita a cada 100 milissegundos
18 }
19
```

Fonte: O autor (2016)

Para visualizar os textos e os valores das variáveis que foram escritos, deve-se assegurar que a conexão entre o PC e o Arduino mantenha-se estabelecida. A tela a qual foi realizada a programação esteja aberta, posteriormente é necessário clicar na aba “Ferramentas”, após selecionar a opção “Monitor serial”, ou clicar diretamente na tela, em um ícone de uma lupa, que abrirá uma tela de monitoramento, conforme demonstrado na Figura 39.

Figura 39- Tela de Monitoramento Serial



Fonte: O autor (2016)

- ESTRUTURAS DE CONTROLE CONDICIONAIS E DE REPETIÇÃO

Blocos de comandos que alteram a sequência de execução do *sketch*. Com o uso dessas estruturas é possível executar instruções diferentes, conforme uma condição ou, repetir diversas vezes um conjunto de instruções (CIRCUITAR.COM.BR).

Condicionais IF e IF...ELSE

As instruções condicionais fornecem a tomada de decisões em um *sketch*. A expressão condicional é escrita entre parênteses, precedida do termo "*if*", de acordo com o exemplo "if (i == 2)", a qual executará um conjunto de comandos escrito após a expressão, dentro de chaves, somente se a condição for verdadeira. Pode ser realizado o teste condicional com duas ou mais expressões, relacionando as mesmas através de operadores lógicos. A Figura 40 ilustra uma aplicação condicional.

Ainda pode-se escrever um conjunto de instruções para o caso do teste condicional resultar em falso, utilizando a instrução "*else*".

Figura 40- Condicionais If e else

exemplo_if §

```
// Variável para contar o número de vezes que o LED piscou

int led = 2; // Pino "2" do Arduino onde será conectado o lED

int i = 0; // Declara variável e a inicializa com valor igual a "0"

void setup()
{
  pinMode (led, OUTPUT); //Define o pino como saida
}

void loop()
{
  // Pisca o LED três vezes
  for (i = 0; i < 3; i++)
  {
    if (i == 2)                 // Sintaxe: if(condição) {
    {
      digitalWrite(led, HIGH);  // Atribui nível lógico alto ao pino do LED, acendendo-o
      delay(1000);              // Espera 1000 milissegundos (um segundo)
      digitalWrite(led, LOW);   // Atribui nível lógico baixo ao pino do LED, apagando-o
      delay(1000);              // Espera 1000 milissegundos (um segundo)
    }
    else
    {
      digitalWrite(led, HIGH);
      delay(1000);
      digitalWrite(led, LOW);
      delay(1000);
    }
  }
  delay(5000);              // Espera 5 segundos para piscar o LED de novo
}

```

Fonte: O autor (2016)

Condicional SWITCH/CASE

Realiza a avaliação do valor de entrada e comparações sequenciais de igualdade. Executa um conjunto de instruções quando uma opção for verdadeira, e sai da mesma ao encontrar uma instrução de parada. É inserido na função “*loop*( )”, como “switch(opção){“, dentro das chaves estão inseridas as opções, por exemplo “case 1:” se o sensor efetuar a leitura de uma peça branca ou, “case 2:” para leitura de uma peça preta, no qual possui internamente a descrição de que ação será tomada para cada opção e, após é finalizado com “break;”. A sintaxe completa desse comando pode ser observada na Figura 41.

Figura 41- Condicional Switch-Case

exemplo_case

```
// Variável para contar o número de vezes que o LED piscou

int led_verde = 2; // Pino "2" do Arduino onde será conectado o lED verde
int led_vermelho = 3; // Pino "3" do Arduino onde será conectado o lED vermelho

int i = 0; // Declara variável e a inicializa com valor igual a "0"
int opcao = 0;

void setup()
{
  pinMode (led_verde, OUTPUT); //Define o pino como saida
  pinMode (led_vermelho, OUTPUT);
}

void loop()
{
  // Pisca três vezes cada LED
  if (i < 2)
  {
   opcao = 1;                       // Direciona a opcao para o "case 1"
  }
  if  ((i > 2) && (i < 5))
  {
   opcao = 2;                       // Direciona a opcao para o "case 1"
  }
  if ( i == 6)
  {
    i = 0;                          // Reinicia o contador de número de "piscadas"
    opcao = 0;
  }

  switch (opcao)
  {
    case 1:
    digitalWrite(led_verde, HIGH);  // Atribui nível lógico alto ao pino do LED, acendendo-o
    delay(1000);                    // Espera 1000 milissegundos (um segundo)
    digitalWrite(led_verde, LOW);   // Atribui nível lógico baixo ao pino do LED, apagando-o
    delay(1000);                    // Espera 1000 milissegundos (um segundo)
    i++;
    break;

    case 2:
    digitalWrite(led_vermelho, HIGH);  // Atribui nível lógico alto ao pino do LED, acendendo-o
    delay(1000);                       // Espera 1000 milissegundos (um segundo)
    digitalWrite(led_vermelho, LOW);   // Atribui nível lógico baixo ao pino do LED, apagando-o
    delay(1000);                       // Espera 1000 milissegundos (um segundo)
    i++;
    break;
  }
}
```

Fonte: O autor (2016)

Laço DO...While

Esse laço executa um conjunto de instruções repetidamente, enquanto uma condição atribuída ao “while” for verdadeira. No caso dessa condição ser

implementada por um contador, é obrigatório a inicialização do mesmo antes do início do *loop*. Após é inserido no *sketch* como “do {“, dentro das chaves são escritas as linhas de código, e após é finalizado com “} while (i < 3);”, nesse caso o programa fica “preso” nessa estrutura, sendo que o laço se repetirá enquanto o valor do contador for menor que 3. A explanação do laço é mostrado na Figura 42.

Figura 42- Laço Do-while

exemplo_do

```
// Variável para contar o número de vezes que o LED piscou

int led = 2; // Pino "2" do Arduino onde será conectado o lED

int i = 0; // Declara variável e a inicializa com valor igual a "0"

void setup()
{
  pinMode (led, OUTPUT); //Define o pino como saida
}

void loop()
{
  // Pisca o LED três vezes
  do
  {
    digitalWrite(led, HIGH);  // Atribui nível lógico alto ao pino do LED, acendendo-o
    delay(1000);              // Espera 1000 milissegundos (um segundo)
    digitalWrite(led, LOW);   // Atribui nível lógico baixo ao pino do LED, apagando-o
    delay(1000);              // Espera 1000 milissegundos (um segundo)
    i++;                      // Incrementa o número de "piscadas"
  }
  while (i < 3);              // Sintaxe: while(condição);

  i = 0;                      // Reinicia o contador de número de "piscadas"
  delay(5000);                // Espera 5 segundos para piscar o LED de novo
}

```

Fonte: O autor (2016)

Laço FOR

Executa um conjunto de comandos enquanto realiza a contagem entre um valor inicial até um valor final. A cada passagem pelo laço de “for” o contador é incrementado. Sua sintaxe é mostrada no exemplo “for (i = 0; i < 3; i++)”, no qual o primeiro termo informado é um valor de inicialização, o segundo, uma condição e o último, é um valor de finalização. O conjunto de comandos é escrito após a expressão, dentro de chaves, conforme é demonstrado na Figura 43.

Figura 43- Laço for

Exemplo_for §

```
// Variável para contar o número de vezes que o LED piscou

int led = 2; // Pino "2" do Arduino onde será conectado o lED

int i; // Declara variável

void setup()
{
  pinMode (led, OUTPUT); //Define o pino como saida
}

void loop()
{
  // Pisca o LED três vezes
  for (i = 0; i < 3; i++) {  // Sintaxe: for(inicialização; condição; finalização) {
    digitalWrite(led, HIGH); // Atribui nível lógico alto ao pino do LED, acendendo-o
    delay(1000);             // Espera 1000 milissegundos (um segundo)
    digitalWrite(led, LOW);  // Atribui nível lógico baixo ao pino do LED, apagando-o
    delay(1000);             // Espera 1000 milissegundos (um segundo)
  }
  delay(5000);               // Espera 5 segundos para piscar o LED de novo
}

```

Fonte: O autor (2016)

ANEXO A – DADOS TÉCNICOS L293D

Tabela 1- Especificações L293D

ELECTRICAL CHARACTERISTICS (for each channel, V_S = 24 V, V_{SS} = 5 V, T_{amb} = 25 °C, unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit
V_S	Supply Voltage (pin 10)		V_{SS}		36	V
V_{SS}	Logic Supply Voltage (pin 20)		4.5		36	V
I_S	Total Quiescent Supply Current (pin 10)	V_i = L ; I_O = 0 ; V_{en} = H		2	6	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	24	mA
		V_{en} = L			4	mA
I_{SS}	Total Quiescent Logic Supply Current (pin 20)	V_i = L ; I_O = 0 ; V_{en} = H		44	60	mA
		V_i = H ; I_O = 0 ; V_{en} = H		16	22	mA
		V_{en} = L		16	24	mA
V_{IL}	Input Low Voltage (pin 2, 9, 12, 19)		– 0.3		1.5	V
V_{IH}	Input High Voltage (pin 2, 9, 12, 19)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
I_{IL}	Low Voltage Input Current (pin 2, 9, 12, 19)	V_{IL} = 1.5 V			– 10	µA
I_{IH}	High Voltage Input Current (pin 2, 9, 12, 19)	2.3 V ≤ V_{IH} ≤ V_{SS} – 0.6 V		30	100	µA
$V_{en\,L}$	Enable Low Voltage (pin 1, 11)		– 0.3		1.5	V
$V_{en\,H}$	Enable High Voltage (pin 1, 11)	$V_{SS} \leq 7$ V	2.3		V_{SS}	V
		$V_{SS} > 7$ V	2.3		7	V
$I_{en\,L}$	Low Voltage Enable Current (pin 1, 11)	$V_{en\,L}$ = 1.5 V		– 30	– 100	µA
$I_{en\,H}$	High Voltage Enable Current (pin 1, 11)	2.3 V ≤ $V_{en\,H}$ ≤ V_{SS} – 0.6 V			± 10	µA
$V_{CE(sat)H}$	Source Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = – 0.6 A		1.4	1.8	V
$V_{CE(sat)L}$	Sink Output Saturation Voltage (pins 3, 8, 13, 18)	I_O = + 0.6 A		1.2	1.8	V
V_F	Clamp Diode Forward Voltage	I_O = 600nA		1.3		V
t_r	Rise Time (*)	0.1 to 0.9 V_O		250		ns
t_f	Fall Time (*)	0.9 to 0.1 V_O		250		ns
t_{on}	Turn-on Delay (*)	0.5 V_i to 0.5 V_O		750		ns
t_{off}	Turn-off Delay (*)	0.5 V_i to 0.5 V_O		200		ns

Fonte: Datasheet SGS-Thomson (1996)

ANEXO B – DADOS TÉCNICOS SN74LS04

Tabela 2- Especificações SN74LS04

SN5404, SN54LS04, SN54S04,
SN7404, SN74LS04, SN74S04
HEX INVERTERS

SDLS029C – DECEMBER 1983 – REVISED JANUARY 2004

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Figure 1)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	SN5404 SN7404 MIN	SN5404 SN7404 TYP	SN5404 SN7404 MAX	UNIT
t_{PLH}	A	Y	R_L = 400 Ω, C_L = 15 pF		12	22	ns
t_{PHL}					8	15	

recommended operating conditions (see Note 3)

		SN54LS04 MIN	SN54LS04 NOM	SN54LS04 MAX	SN74LS04 MIN	SN74LS04 NOM	SN74LS04 MAX	UNIT
V_{CC}	Supply voltage	4.5	5	5.5	4.75	5	5.25	V
V_{IH}	High-level input voltage	2			2			V
V_{IL}	Low-level input voltage			0.7			0.8	V
I_{OH}	High-level output current			−0.4			−0.4	mA
I_{OL}	Low-level output current			4			8	mA
T_A	Operating free-air temperature	−55		125	0		70	°C

NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	TEST CONDITIONS†			SN54LS04 MIN	SN54LS04 TYP‡	SN54LS04 MAX	SN74LS04 MIN	SN74LS04 TYP‡	SN74LS04 MAX	UNIT
V_{IK}	V_{CC} = MIN,	I_I = −18 mA				−1.5			−1.5	V
V_{OH}	V_{CC} = MIN,	V_{IL} = MAX,	I_{OH} = −0.4 mA	2.5	3.4		2.7	3.4		V
V_{OL}	V_{CC} = MIN,	V_{IH} = 2 V	I_{OL} = 4 mA		0.25	0.4			0.4	V
			I_{OL} = 8 mA					0.25	0.5	
I_I	V_{CC} = MAX,	V_I = 7 V				0.1			0.1	mA
I_{IH}	V_{CC} = MAX,	V_I = 2.7 V				20			20	µA
I_{IL}	V_{CC} = MAX,	V_I = 0.4 V				−0.4			−0.4	mA
I_{OS}§	V_{CC} = MAX			−20		−100	−20		−100	mA
I_{CCH}	V_{CC} = MAX,	V_I = 0 V			1.2	2.4		1.2	2.4	mA
I_{CCL}	V_{CC} = MAX,	V_I = 4.5 V			3.6	6.6		3.6	6.6	mA

† For conditions shown as MIN or MAX, use the appropriate value specified under recommended operating conditions.
‡ All typical values are at V_{CC} = 5 V, T_A = 25°C.
§ Not more than one output should be shorted at a time, and the duration of the short-circuit should not exceed one second.

switching characteristics, V_{CC} = 5 V, T_A = 25°C (see Figure 2)

PARAMETER	FROM (INPUT)	TO (OUTPUT)	TEST CONDITIONS	SN54LS04 SN74LS04 MIN	SN54LS04 SN74LS04 TYP	SN54LS04 SN74LS04 MAX	UNIT
t_{PLH}	A	Y	R_L = 2 kΩ, C_L = 15 pF		9	15	ns
t_{PHL}					10	15	

Fonte: Datasheet Texas Instruments (2004)

ANEXO C – DADOS TÉCNICOS TCRT 5000

Tabela 3- Especificações TCRT 5000

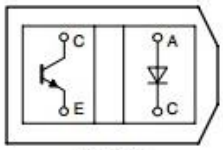
VISHAY

TCRT5000, TCRT5000L

Vishay Semiconductors

Reflective Optical Sensor with Transistor Output

19156_2



Top view 19156_1

FEATURES

- Package type: leaded
- Detector type: phototransistor
- Dimensions (L x W x H in mm): 10.2 x 5.8 x 7
- Peak operating distance: 2.5 mm
- Operating range within > 20 % relative collector current: 0.2 mm to 15 mm
- Typical output current under test: I_C = 1 mA
- Daylight blocking filter
- Emitter wavelength: 950 nm
- Lead (Pb)-free soldering released
- Compliant to RoHS directive 2002/95/EC and in accordance to WEEE 2002/96/EC

Pb-free

e2

RoHS COMPLIANT

DESCRIPTION

The TCRT5000 and TCRT5000L are reflective sensors which include an infrared emitter and phototransistor in a leaded package which blocks visible light. The package includes two mounting clips. TCRT5000L is the long lead version.

APPLICATIONS

- Position sensor for shaft encoder
- Detection of reflective material such as paper, IBM cards, magnetic tapes etc.
- Limit switch for mechanical motions in VCR
- General purpose - wherever the space is limited

BASIC CHARACTERISTICS (1)

PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
INPUT (EMITTER)						
Forward voltage	I_F = 60 mA	V_F		1.25	1.5	V
Junction capacitance	V_R = 0 V, f = 1 MHz	C_j		17		pF
Radiant intensity	I_F = 60 mA, t_p = 20 ms	I_e			21	mW/sr
Peak wavelength	I_F = 100 mA	λ_P	940			nm
Virtual source diameter	Method: 63 % encircled energy	d		2.1		mm
OUTPUT (DETECTOR)						
Collector emitter voltage	I_C = 1 mA	V_{CEO}	70			V
Emitter collector voltage	I_e = 100 µA	V_{ECO}	7			V
Collector dark current	V_{CE} = 20 V, I_F = 0 A, E = 0 lx	I_{CEO}		10	200	nA
SENSOR						
Collector current	V_{CE} = 5 V, I_F = 10 mA, D = 12 mm	I_C (2) (3)	0.5	1	2.1	mA
Collector emitter saturation voltage	I_F = 10 mA, I_C = 0.1 mA, D = 12 mm	V_{CEsat} (2) (3)			0.4	V

Note

(1) T_{amb} = 25 °C, unless otherwise specified

(2) See figure 3

(3) Test surface: mirror (Mfr. Spindler a. Hoyer, Part No. 340005)

Fonte: Datasheet Vishay Semiconductors (2009)