UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
DEPARTAMENTO DE CIÊNCIAS AMBIENTAIS E TECNOLÓGICAS
ENGENHARIA DE ENERGIA

VINICIUS DE PAIVA COSTA

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL DE BAIXO CUSTO

MOSSORÓ-RN
2013

VINICIUS DE PAIVA COSTA

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL DE BAIXO CUSTO

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Energia da Universidade Federal Rural do Semi-Árido – UFERSA, campus Mossoró para obtenção do título de Engenheiro de Energia.

Orientador: Prof. Dr. Francisco José Targino Vidal – UFERSA

MOSSORÓ-RN

2013

Ficha catalográfica preparada pelo setor de classificação e catalogação da Biblioteca "Orlando Teixeira" da UFERSA

C837d Costa, Vinicius de Paiva.
Desenvolvimento de uma interface gráfica para aplicação em robótica educacional de baixo custo / Vinicius de Paiva Costa. – Mossoró, RN : 2013.
30f. : il.

Orientador: Profº. Dr. Francisco José Targino Vidal.
Monografia (Graduação) – Universidade Federal Rural do Semi-Árido, Graduação em Engenharia de Energia, 2013.

1. Robótica educacional. 2. Interface gráfica. 3. Controle eletrônico. 4. Desenvolvimento de software. I. Título.

CDD: 005.1

Bibliotecária: Marilene Santos de Araújo
CRB-5/1033

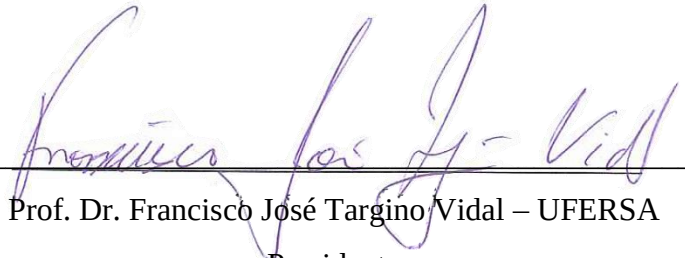
VINICIUS DE PAIVA COSTA

DESENVOLVIMENTO DE UMA INTERFACE GRÁFICA PARA APLICAÇÃO EM ROBÓTICA EDUCACIONAL DE BAIXO CUSTO

Trabalho de Conclusão de Curso apresentado ao curso de Engenharia de Energia da Universidade Federal Rural do Semi-Árido – UFERSA, campus Mossoró para obtenção do título de Engenheiro de Energia.

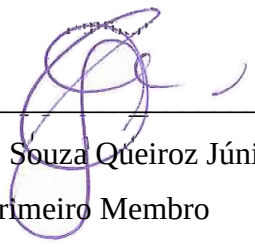
APROVADO EM: 11 / 09 / 2013

BANCA EXAMINADORA



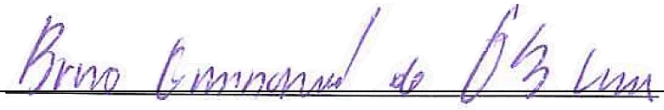
Prof. Dr. Francisco José Targino Vidal – UFERSA

Presidente



Prof. Dr. Idalmir de Souza Queiroz Júnior – UFERSA

Primeiro Membro



Prof. M.Sc. Bruno Emmanuel de Oliveira Barros Luna – UFERSA

Segundo Membro

À minha família e aos meus amigos.

AGRADECIMENTOS

A Deus, por absolutamente tudo;

Aos meus pais, Maria e Tarcísio, por me darem a vida, as oportunidades, a educação e os exemplos que moldaram meu caráter e me definiram como sou; aos meus irmãos, Marcus e Matheus, por me acompanharem nesse mesmo processo, tornando tudo mais interessante e divertido;

A Ana Clara e Ravi, não é necessário dizer o porquê;

Ao companheiro de desespero, Marcos Vinicius, parceiro de (vários) trabalhos e (várias) noites em claro tentando aprender alguma coisa, pelos ótimos anos de curso;

Aos amigos Dênis e Edmilson (Edii) por me ensinarem a transformar os dragões da programação em moinhos de vento;

A Eriana, por me aguentar durante todo esse processo de TCC, por me apoiar e incentivar e por se esforçar para entender explicações sobre ponteiros, motores, controle de rotação, etc;

A todos os meus outros amigos, por serem exatamente quem são e exatamente como são;

“Não há conhecimento que não seja poder.”

(Ralph Waldo Emerson)

RESUMO

Com a crescente popularização de computadores nas mais diversas classes brasileiras, a robótica educacional torna-se cada vez mais viável como instrumento de construção de raciocínio lógico e sequencial, além ser capaz de apresentar às pessoas de qualquer idade as possibilidades da automação de uma forma descomplicada, intuitiva e prazerosa. Entretanto, embora o acesso a computadores esteja cada vez mais fácil, muitos kits de robótica educacional ainda têm um custo elevado, fator desencorajador para o engajamento na área. Surge então a oportunidade de desenvolver um software de interface gráfica que faça essa ponte entre usuário e kit de robótica de baixo custo. O software de controle, batizado de "Barrichello", foi desenvolvido no ambiente Microsoft Visual Studio, usando a linguagem C++/CLI, com o intuito de oferecer um primeiro contato descomplicado para o usuário que nunca se envolveu com programação ou robótica. O software permite realizar o controle direto e pré-programado de um robô, oferecendo controles de movimentação básica e fazendo uso de recurso visuais simples e intuitivos, com imagens que chamam a atenção e são autoexplicativas.

Palavras-Chave: Robótica Educacional. Programação. Controle eletrônico. Desenvolvimento de software.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ANSI	American National Standards Institute
CSS	Cascading Style Sheets
GE	General Motors
HTML	HyperText Markup Language
IA	Inteligência Artificial
IAT	Índice de Avanço Tecnológico
IDE	Integrated Development Environment
ISO	International Organization for Standardization
LED	Light-Emitting Diode
MIT	Massachusetts Institute of Technology
MSDN	Microsoft Developer Network
ONU	Organização das Nações Unidas
PC	Personal Computer
STL	Standard Template Library
UFERSA	Universidade Federal Rural do Semi-Árido
WPF	Windows Presentation Foundation
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
XSLT	eXtensible Stylesheet Language for Transformation

SUMÁRIO

1 INTRODUÇÃO

A evolução da tecnologia sempre provoca alterações no comportamento da sociedade. Assim como a televisão, o rádio e o telefone (fixo e celular), o computador também afeta a vida de muitas pessoas. Entretanto, no caso do computador esse impacto é muito maior do que os outros, uma vez que o acesso à tecnologia e a usabilidade dessa ferramenta são maiores e mais abrangentes.

Comunicação, acesso às mais variadas mídias, evoluções nos equipamentos projetados para melhorar a qualidade de vida, são apenas algumas áreas que são beneficiadas diretamente pelos computadores. Todavia, essa é uma realidade que ainda se encontra distante de grande parte da população (principalmente brasileira), em virtude dos altos custos, falta de profissionais capacitados para ensinar, infraestrutura precária e ausência de uma política bem definida para inclusão digital.

Para melhor visualizar a questão, a ONU estabeleceu um novo indicador para o Desenvolvimento Humano: o Índice de Avanço Tecnológico (IAT), criado para avaliar a produção e disseminação das novas tecnologias e, acima disso, seu aproveitamento pela população. Foram analisados 72 países onde houve acesso a dados confiáveis. O Brasil ficou em 43o lugar. O índice leva em conta a criação e capacidade de inovação em novas tecnologias, difusão das mais recentes conquistas assim como das tecnologias mais antigas (eletricidade e telefonia, por exemplo) e habilidade intelectual, ou seja, a taxa de escolaridade. A delicada posição do Brasil no ranque, atrás de países como Panamá, Trinidad e Tobago e Romênia, mostra que precisamos investir ainda muito mais tempo e recursos. (BARROS, 2008)

É nesse contexto que o uso da robótica educacional vai servir como ferramenta de inclusão digital para alunos de escolas públicas. O maior ponto positivo é que essa abordagem atrai a curiosidade dos alunos por se tratar de algo que eles veem como um simples brinquedo, porém um brinquedo que pode ser “ensinado” a fazer o que eles quiserem. E para ensinar o robô a realizar as mais diferentes tarefas os alunos aprendem noções básicas de programação, lógica e desenvolvem raciocínio algorítmico.

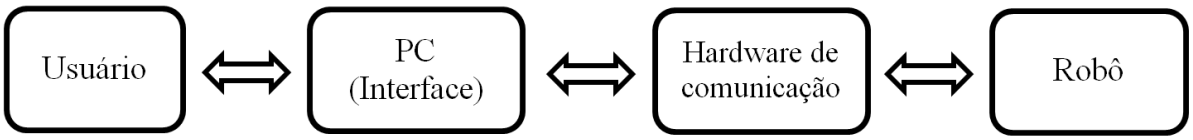
No entanto, as mais completas e amigáveis ferramentas de ensino (como o LEGO MindStorms) ainda são relativamente caras para a realidade de muitas escolas públicas e o seu valor elevado desestimula a introdução da robótica educacional para inclusão digital. Faz-se, então, necessário o desenvolvimento de kits de robótica de baixo custo assim como interfaces

computacionais compatíveis e ao mesmo tempo simples de serem usadas num contexto educacional.

2 OBJETIVOS

O objetivo do presente trabalho foi desenvolver um software de interface gráfica que possa ser utilizado para estabelecer uma comunicação computador-robô e comandar o mesmo, desde instruções de movimentação básica até sequencias e passos de rotinas programadas pelo usuário, tudo dentro do contexto de robótica educacional de baixo custo. O esquema de comunicação pode ser visto na Figura 1.

Figura 1 – Diagrama de uso da interface desenvolvida



Fonte: autoria própria

3 REVISÃO LITERÁRIA

A robótica é uma área responsável pelo desenvolvimento de dispositivos capazes de realizar tarefas com eficiência e exatidão, incluindo as que são impossíveis de serem executadas pelo homem sem risco de vida. Ela busca o desenvolvimento e a integração de técnicas e algoritmos para a criação de robôs. (FILHO, 2009)

3.1 A ROBÓTICA

3.1.1 A Robótica na Mitologia Grega

A primeira narrativa mitológica com ideias e conceitos comuns à robótica de que se tem notícia é a história de Pigmalião. Pigmalião, de acordo com a mitologia, era o rei da ilha de Chipre, além de ser escultor. Em sua busca pela mulher perfeita que atendesse a todas as suas expectativas e realizar todas as suas vontades, começou a esculpir uma estátua, pela qual acabou se apaixonando durante o processo. A deusa do amor, Afrodite, vendo então a paixão

de Pigmalião pela sua obra, resolve dar vida à estátua, chamando-a de Galatéia, que posteriormente veio a se tornar a esposa do rei.

Os gregos conheciam o mito de Hefáistos (ou Vulcano), que narra a história de um deus renegado que dominava a arte da forja, forjando joias, raios e instrumentos metálicos e mecânicos de um modo geral em sua oficina situada na boca do vulcão da ilha de Lemnos. De acordo com o mito, Hefáistos construiu para si servos de metal, os ciclopes, que tinham como função ajudar-lhe em sua oficina.

A vontade de criar seres que possam realizar tarefas para seus criadores também pode ser vista na história do Golem, de origem judaica. Segundo a lenda, o Golem teria sido feito a partir da argila do rio Moldava, em Praga, por um rabino que seguiu rituais específicos e lhe conferiu "vida" recitando um encanto em hebraico. O Golem tinha por obrigação obedecer ao rabino, ajudar e proteger o gueto judaico.

3.1.2 História da Robótica

Historicamente, acredita-se que as primeiras invenções mais próximas do conceito de robôs tenham sido construídas pelos gregos. Um matemático e engenheiro grego chamado Ctesibius, que viveu entre 285 e 222 a.C. em Alexandria, arquitetou vários aparelhos robóticos, sendo a clepsidra (ou relógio de água) o mais famoso entre eles, sendo um dos primeiros sistemas criados pelo homem para realizar medição de tempo. Acredita-se ainda que Heron de Alexandria, também grego, tenha construído diversas invenções na área de automação, incluindo a primeira máquina de vender bebidas da história, onde alguém inseria na mesma uma moeda e recebia um jato de água; também são creditadas a ele a invenção de um autômato capaz de andar para frente e para traz, movido por um sistema alimentado pela energia cinética de grãos de trigo caindo do seu topo, além do primeiro motor a vapor documentado na história.

Outro destaque do mundo da robótica é Jacques de Vaucanson, inventor e artista francês, que criou o primeiro robô funcional em 1738: um androide capaz de tocar flauta, além de um pato mecânico que se alimentava.

Os primeiros robôs produzidos para a industrialização, entretanto, só começaram a surgir na década de 1950. O primeiro robô, chamado *Unimate*, foi construído por Joseph F. Engelberger, considerado o "pai da robótica". O *Unimate* foi vendido para a *General Motors* e passou a trabalhar na linha de montagem em Nova Jersey, em 1961. Alguns autores também

atribuem o crédito do *Unimate* ao inventor George DeVol, afirmando que o mesmo trabalhou em equipe com Engelberger na sua estruturação.

A partir do *Unimate*, viu-se na robótica industrial a possibilidade de aumentar a produtividade e a qualidade dos produtos para as indústrias, além da redução de custos. Tal vertente, entretanto, trazia o revés social de fomentar o desemprego, uma vez que os robôs iriam substituir o trabalho humano.

Contudo, há áreas da robótica que geram um impacto social positivo, com situações em que robôs são utilizados para substituir seres humanos em trabalhos que ofereçam riscos, como robôs bombeiros, antibombas, submarinos, entre outros.

Nesse contexto fica claro que há vários tipos de robôs com as mais variadas aplicações e utilidades. Classificam-se, então, de acordo com as suas gerações.

3.1.3 Primeira geração da Robótica

De uma forma sucinta, são os braços robóticos industriais (tal como o *Unimate*), ou os braços para coleta de amostras submarinas. Possuem uma movimentação previamente programada e são capazes apenas de repetir a mesma sequência fixa de passos. Esses robôs têm sensores que coletam informações do seu estado interno. Requerem um ambiente bem estruturado com objetos bem posicionados para que possam efetuar a tarefa que lhes foi programada.

3.1.4 Segunda geração da Robótica

Diferente dos robôs da primeira geração, estes possuem sensores externos, além dos internos. A existência dos sensores externos permite que os robôs interajam com o meio externo e se adequem ao mesmo. Entre os sensores externos, destacam-se as câmeras de captura de imagens que podem ser comparadas com um banco de dados, sensores de toque, de peso, de luminosidade, de cores, de fim de percurso, entre outros. Os robôs montados com os kits mais comuns de robótica educacional pertencem a esta geração.

3.1.5 Terceira geração da Robótica

Os robôs da terceira geração possuem IA (Inteligência Artificial) e utilizam mecanismos complexos como visão computacional, síntese e reconhecimento de voz,

atualização de posicionamento, algoritmos de rotas, heurísticas, e simulação de comportamento humano ou animal – entre outras características. Além disso, também podem existir em ambientes físicos e/ou virtuais. (AZEVEDO, et al, 2009)

3.2 A ROBÓTICA EDUCACIONAL

A robótica educacional, enquanto matéria de ensino, procura preparar jovens e adultos para montar mecanismos robotizados simples. Para isso, utilizam-se “kits de montagem”, que fornecem noções de robótica e possibilitam o desenvolvimento de habilidades em montagem e programação de robôs. Ela incentiva a criação, o desenho, o desenvolvimento, a programação e a utilização de um robô, que está intimamente interligada com a solução de problemas do “mundo real”, podendo dar a cada um deles um embasamento sólido para o desenvolvimento de seus próprios projetos. (LIEBERKNECHT, 2009)

De acordo com Santos (2012):

> A robótica educacional ou robótica pedagógica são termos utilizados para caracterizar ambientes de aprendizagem que reúnem materiais de sucata ou kits de montagem compostos por peças diversas, motores e sensores controláveis por computador e softwares que permitam programar de alguma forma o funcionamento dos modelos montados. Aumentando o interesse e a criatividade dos alunos e integrando diversas disciplinas, a robótica educacional, ou robótica pedagógica, tem despertado a atenção de professores e alunos. Nesse tipo de atividade, o aluno vivencia na prática através da construção de maquetes e robôs controlados por computador, conceitos estudados em sala de aula.

Por se tratar de uma atividade lúdica e desafiadora, ela valoriza o trabalho em grupo, a cooperação, planejamento, pesquisa, tomada de decisões, definição de ações, promove o diálogo e o respeito a diferentes opiniões.

3.3 LINGUAGENS DE PROGRAMAÇÃO

3.3.1 As linguagens C e C++

A Linguagem *C++* começou como uma versão expandida de *C*. As extensões *C++* foram inventadas primeiramente por Bjarne Stroustrup, em 1979, no Bell Laboratories em Murray Hill, New Jersey. Ele inicialmente chamou a nova linguagem de “C com classes”. Contudo, em 1983 o nome foi mudado para *C++*. (SILVA, 2004)

Mesmo sendo *C* uma das linguagens mais utilizadas em todo o mundo, a invenção da Linguagem *C++* se fez necessária devido a um fator de programação: o aumento da complexidade. No decorrer dos anos, programas de computador se tornaram maiores e mais complexos. Mesmo sendo uma excelente linguagem de programação, *C* tem seus limites e uma vez que um programa atinja a marca de 25.000 à 100.000 linhas de códigos, este se torna tão complexo que é difícil analisá-lo como um todo. A manutenção de seu código se torna cansativa. O propósito do *C++* é quebrar esta barreira. A essência do *C++* é permitir que programadores compreendam e gerenciem programas cada vez mais complexos. (STROUSTRUP, 1994)

A maioria das adições feitas ao *C* por Stroustrup suportam a programação orientada a objetos. Stroustrup afirmou que algumas das características da orientação a objetos de *C++* foram inspiradas em uma linguagem chamada *Simula67*. Assim, *C++* representa a combinação de dois métodos poderosos de programação.

Desde que a linguagem *C++* foi inventada, ela já passou por três revisões importantes, com adições e modificações da linguagem. A primeira revisão ocorreu em 1985 e a segunda em 1990. A terceira ocorreu durante a padronização do *C++*.

O primeiro documento contendo o padrão proposto foi criado em 25 de janeiro de 1994. Neste documento, o comitê *ANSI/ISO C++* manteve as características definidas por Stroustrup e acrescentou algumas outras. Mas, no geral, este documento inicial refletiu o estado de *C++* na época. (SILVA, 2004)

Logo após a finalização do primeiro documento de padronização *C++*, um evento ocorreu e com ele a linguagem foi amplamente expandida: a criação da *Standard Template Library* (*STL*) por Alexander Stepanov. A *STL* é um conjunto de rotinas genéricas que podem ser utilizadas para manipular dados. A *STL* não é somente poderosa e elegante, mas também muito extensa.

3.3.2 A linguagem C++/CLI

C++/CLI (do inglês "*Common Language Infrastructure*" – Infraestrutura de Linguagem Comum, em tradução livre) é uma especificação de linguagem criada pela *Microsoft* com o objetivo de substituir o *Managed C++* – outro conjunto de variantes de *C++* da *Microsoft*. O *C++/CLI* é uma revisão completa que tem como objetivo simplificar a antiga sintaxe *Managed C++*, que caiu em desuso. Esta linguagem se encontra disponível nos programas *Visual Studio 2005*, *2008*, *2010* e *2012*, incluindo as *Express editions*. Embora seja

baseada no *C++*, há diferenças de sintaxe entre ambas, como o uso de ponteiros diferentes em aplicações de código padrão misturado com código gerenciado, por exemplo.

4 MATERIAIS E MÉTODOS

Uma vez que o foco do trabalho é a implementação de um software de controle para utilização em robótica de baixo custo, a construção do robô em si não será abordada, e sim apenas o desenvolvimento do software responsável pelo controle e comunicação.

4.1 KIT DE ROBÓTICA UTILIZADO

O kit de robótica utilizado para se comunicar com o software implementado foi desenvolvido por Marcos Vinicius Alves Costa, aluno do curso de Engenharia de Energia da Universidade Federal Rural do Semi-Árido – UFERSA, campus Mossoró. O robô básico é controlado por um microcontrolador PIC16F628A, a movimentação se dá por meio de dois motores de antena parabólica modificados. A comunicação com o computador ocorre por intermédio de um módulo *bluetooth* JY-MCU HC-06, que emula uma conexão RS-232 (padrão serial) com o computador conectado.

4.2 SOFTWARES EDUCACIONAIS

4.2.1 Scratch

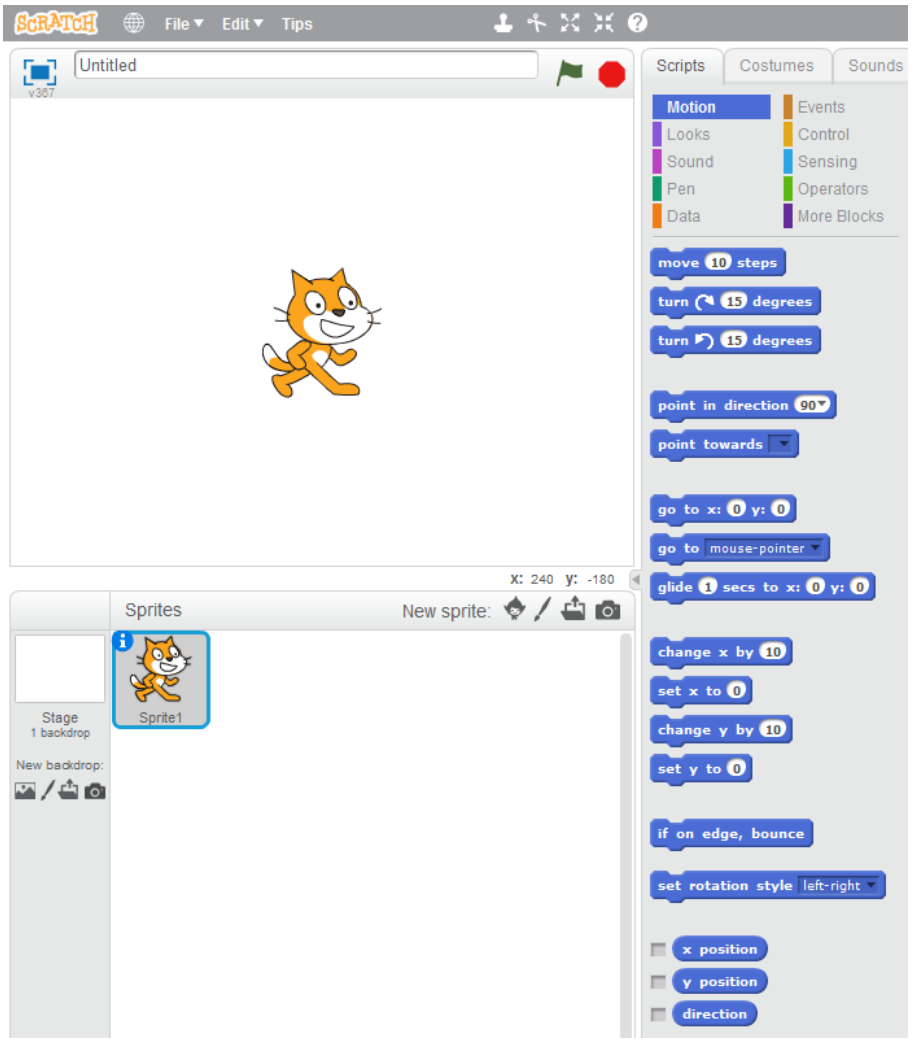
Scratch é uma linguagem de programação educacional e ferramenta de autoria multimídia que pode ser utilizada por alunos, professores e pais em uma vasta gama de projetos de matemática e ciências, incluindo simulações e visualizações de experimentos, apresentações animadas, estórias animadas e música e arte interativas. Além disso, a ferramenta pode ser usada para criar jogos simples.

A sua primeira versão foi desenvolvida em 2003 pelo *Lifelong Kindergarten group*, liderado por *Mitchel Resnick*, no MIT Media Lab. (MIT, 2013)

Apesar de não estar ligado diretamente a nenhum kit de robótica educacional (físico), o *Scratch* é uma ferramenta renomada no ensino de lógica algorítmica e pensamento sequencial. Altamente intuitivo, possui diversos "blocos", cada um associado a uma ação, condição, laço ou evento, que podem ser combinados para formar a sequência que se deseja

executar, como pode ser visto na Figura 2. A facilidade de uso serviu como influência para a interface gráfica desenvolvida neste trabalho.

Figura 2 – Interface do programa *Scratch*



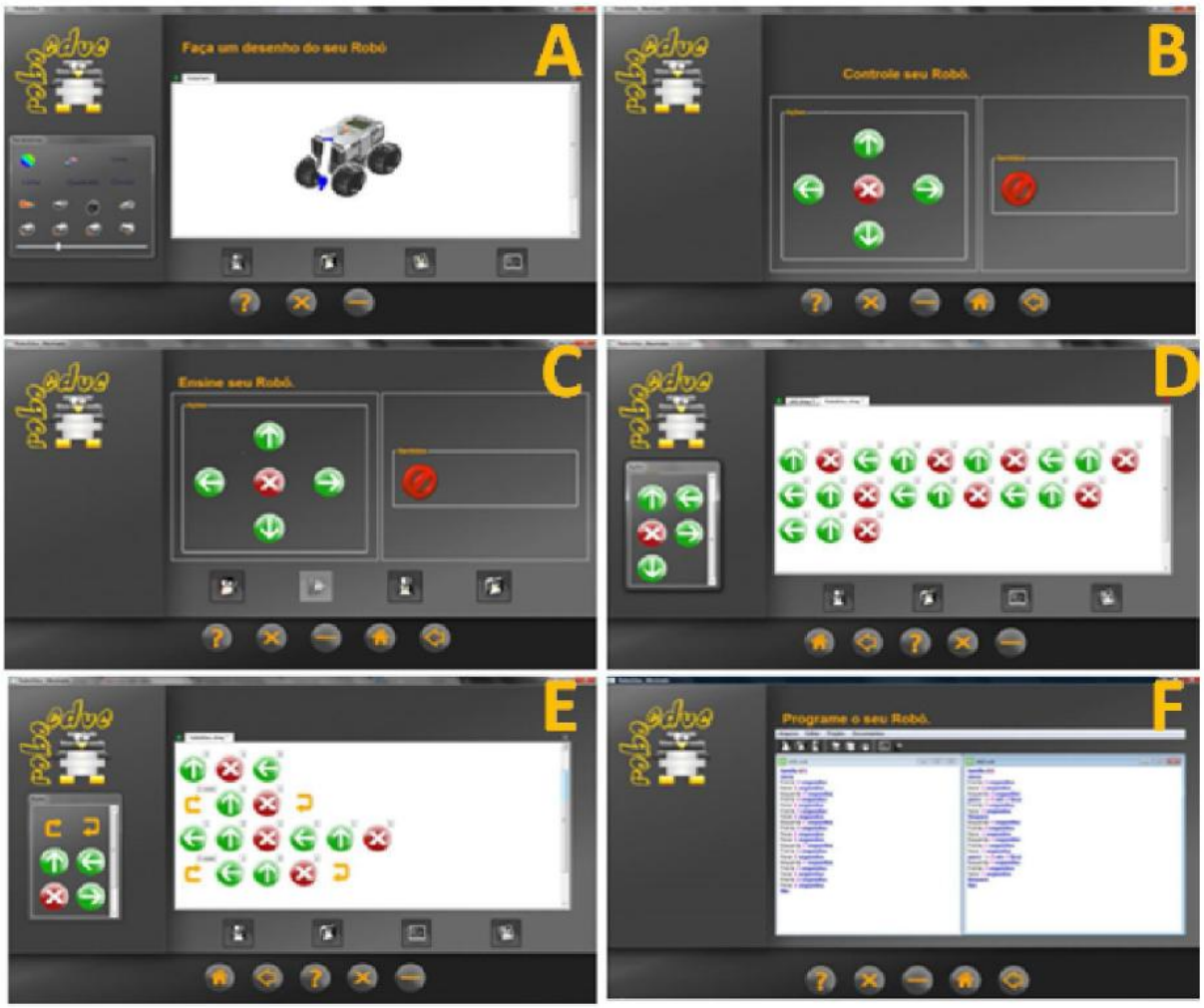
Fonte: Scratch Project Editor (2013)

4.2.2 RoboEduc

O *RoboEduc* surgiu a partir de um projeto de inclusão digital usando robôs para alunos de uma escola pública da periferia de Natal/RN. Em 2005, após as primeiras experiências, percebeu-se que era necessária uma pesquisa de novas ferramentas e metodologias para aplicação de robótica pedagógica. Através de trabalhos de alunos de graduação, mestrado e doutorado nas áreas de Educação e Computação, surgiram uma metodologia de robótica pedagógica e um sistema (software) de autoria para projeto, controle e programação de robôs. (AZEVEDO, et al, 2009)

A interface utilizada no *RoboEduc* foi concebida para um público amplo, pois é orientada para alunos desde a educação infantil a estudantes universitários. O propósito do sistema desenvolvido é ensinar robótica, linguagem de programação, lógica de algoritmos e informática, além de poder ser usada em outros conteúdos transversais. É possível, a partir deste software, que os alunos possam projetar o robô, através de uma interface gráfica de desenho usando imagens de peças do kit robótico usado em aula. Após a montagem, o usuário pode controlar, ensinar ou programar o robô de acordo com conjuntos de instruções e opções divididos em níveis de dificuldade de atividade, correspondente a diferentes graus de domínio da robótica e de linguagens de programação, como visto na Figura 3.

Figura 3 – Diferentes níveis de controle do RoboEduc (versão do ano de 2009)



Fonte: AZEVEDO, et al (2009)

4.3 AMBIENTE DE DESENVOLVIMENTO

O ambiente escolhido foi o *Microsoft Visual Studio*, versão 2008. Embora seja uma ferramenta normalmente paga, muitas universidades – incluindo a UFERSA – têm um convênio com a *Microsoft* chamado *DreamSpark*, que permite aos alunos da instituição fazer *download* gratuito das ferramentas disponibilizadas, sendo o *Visual Studio* uma delas.

Microsoft Visual Studio é um ambiente de desenvolvimento integrado (do inglês IDE - *Integrated Development Environment*) da empresa *Microsoft*. Ele é usado para desenvolver aplicações de interface gráfica junto de *Windows Forms* ou aplicações *WPF*, websites, aplicações web, e serviços web tanto em código nativo quanto com código gerenciado para todas as plataformas compatíveis com *Microsoft Windows*, *Windows Mobile*, *Windows CE*, *.NET Framework*, *.NET Compact Framework* e *Microsoft Silverlight*. (MSDN, 2013)

O *Visual Studio* é compatível com diferentes linguagens de programação, o que permite ao desenvolvedor utilizar praticamente qualquer linguagem de programação existente. As linguagens nativas incluem *C/C++*, *VB.NET*, *C#* e *F#*. Outras como *M*, *Python* e *Ruby* também são compatíveis, embora tenham que ser instaladas de forma separada. O *Visual Studio* também é compatível com *XML/XSLT*, *HTML/XHTML*, *JavaScript* e *CSS*.

4.4 O PROGRAMA DE CONTROLE DESENVOLVIDO: BARRICHELLO

4.4.1 Barrichello

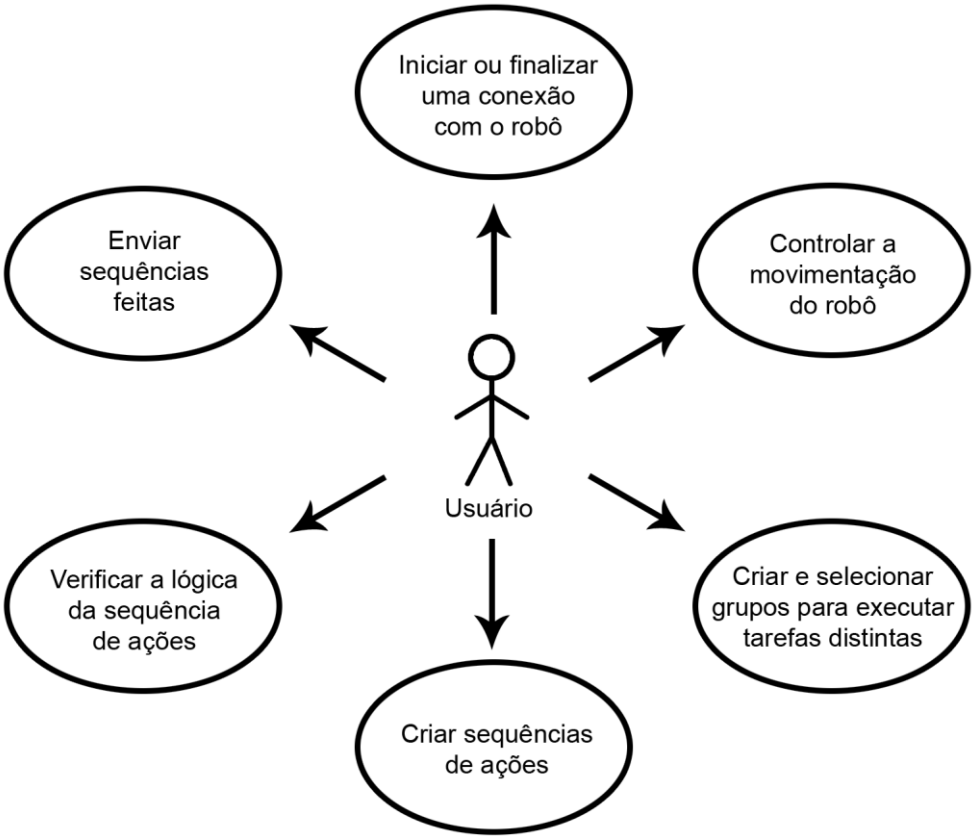
O software de controle desenvolvido foi nomeado “Barrichello”, como uma forma de homenagear o piloto brasileiro de Fórmula 1, Rubens Barrichello. Ele foi idealizado e implementado com o objetivo de ser o mais intuitivo possível, fazendo uso de imagens com diferentes cores para diferentes tarefas, botões com setas e ícones distintos para tarefas distintas. Entretanto, procurou-se não limitar muito o sistema, de modo que o usuário ainda tem acesso a informações e status mais relevantes e específicos quando necessário, sem que isso afete a facilidade de uso de forma negativa.

O sistema foi implementado na ferramenta *Visual Studio 2008*. Tal escolha se deu pelos recursos fornecidos pela mesma para a criação de interfaces gráficas. Dentre as opções de linguagem de programação oferecidas pelo *Visual Studio 2008*, foi escolhida a linguagem *C++/CLI* por ser uma variante do *C++*, linguagem altamente difundida que é bastante utilizada por grande parte do mundo, onde a variação *C++/CLI* ainda oferece suporte

facilitado tanto a códigos gerenciados quanto não gerenciados, simplificando então o acesso às bibliotecas já existentes no *Visual Studio 2008* – além de preferência pessoal do graduando.

Em termos de ensino, o software oferece duas opções de controle, sendo uma básica, e uma intermediária; tais opções surgem na forma das abas *Controle Remoto* e *Programar Sequência de Ações*. Um diagrama dos casos de uso pode ser visto na Figura 4.

Figura 4 – Diagrama dos casos de uso do software Barrichello



Fonte: autoria própria

4.4.1.1 Iniciar ou finalizar uma conexão com o robô

Uma vez que a transmissão de dados entre o programa e o robô se dá através do protocolo *bluetooth*, é necessário que em algum momento seja estabelecida a comunicação entre o computador e o robô. Por isso, os controles de conexão básicos são encontrados tanto na aba de controle remoto quanto na aba de programar sequência de ações. O módulo *bluetooth* utilizado no robô, uma vez conectado ao computador, emula uma porta RS-232 (serial) no mesmo. Devido ao fato dessa porta ser criada com um nome arbitrário escolhido pelo sistema operacional na hora da conexão (COM1, COM2, COM3, COM4...), seria

impossível para o sistema criado usar sempre uma porta específica e imutável. Dessa forma, cabe ao usuário identificar em qual porta está sendo feita a conexão e selecioná-la como porta a ser utilizada. Isso é feito ao clicar no botão “Procurar Portas”, que cria uma lista com todas as portas seriais existentes no computador.

Selecionada a porta a ser usada, o usuário deve clicar no botão “Conectar” para que a comunicação seja estabelecida. O evento associado a esse botão pode ser visto na Figura 5.

Figura 5 – Excerto de código: evento de conectar

```
private: System::Void conectar_Click(System::Object^  sender, System::EventArgs^
e) {
   if (((exptoggle == false) && (this->porta->Text == L"")) || ((exptoggle ==
true) && (this->porta2->Text == L""))){// caso a porta esteja em branco
       MessageBox::Show("Selecione uma porta para estabelecer a
conexão.","Notificação", MessageBoxButtons::OK, MessageBoxIcon::Asterisk);
   }
   else {
       if (exptoggle == false)
       this->serial->PortName = this->porta->Text; // define o nome da porta
       else if (exptoggle == true)
       this->serial->PortName = this->porta2->Text;// define o nome da porta
       this->serial->Open();                    // abre a porta
       this->serial->ReadTimeout = 3000;        // define os tempos de timeout
       this->serial->WriteTimeout = 3000;
       if (exptoggle == false){   // ativação/desativação dos botões
          this->conectar->Enabled = false;
          this->desconectar->Enabled = true;
          this->porta->Enabled = false;
          this->confcon->Enabled = false;
       }
       else if (exptoggle == true){ // ativação/desativação dos botões
          this->conectar2->Enabled = false;
          this->desconectar2->Enabled = true;
          this->porta2->Enabled = false;
          this->confcon2->Enabled = false;
          if (algok == true)      // se a sequência já tiver sido verificada
              this->enviar->Enabled = true;          // o botão de iniciar
                                                     sequência é ativado
          }
          this->stdesc->Visible = false;             // atribuição dos status
          this->stcon->Visible = true;
          this->stport->Text = this->serial->PortName;
          this->stcom1->Text = this->stcom2->Text; // registrar no histórico
          this->stcom2->Text = this->stcom3->Text;
          this->stcom3->Text = this->stcom4->Text;
          this->stcom4->Text = this->stcom5->Text;
          this->stcom5->Text = this->stcom6->Text;
          this->stcom6->Text = "Conectar";
          if (exptoggle == false)
              this->desconectar->Focus(); // leva o foco para o próximo botão
          else if (exptoggle == true)
              this->desconectar2->Focus();// leva o foco para o próximo botão
       }
}
```

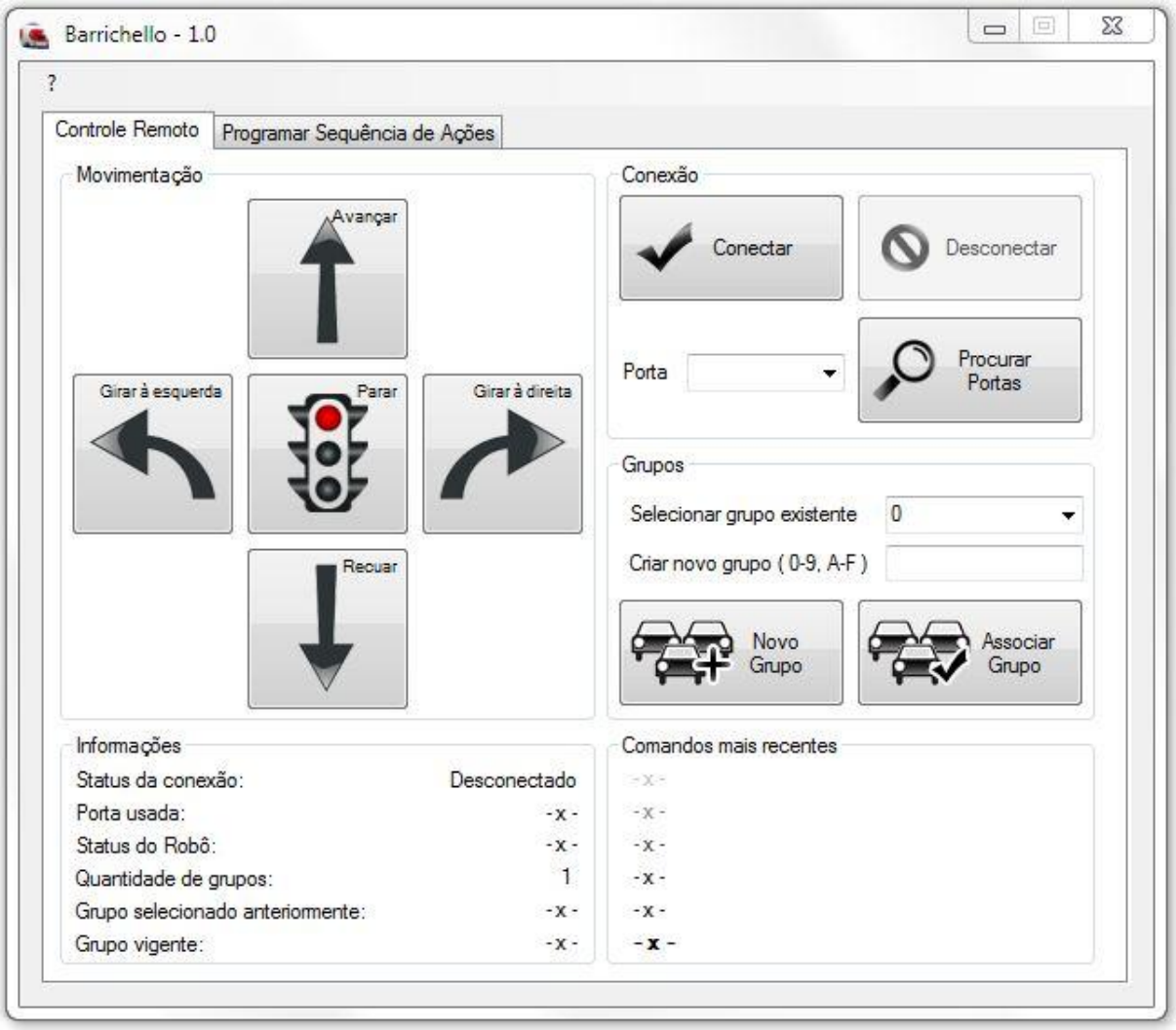
Fonte: autoria própria

A qualquer momento, o usuário pode interromper a comunicação clicando no botão “Desconectar”.

4.4.1.2 Controlar a movimentação do robô

O controle direto da movimentação do robô se dá através de setas presentes na aba “Controle Remoto”, como se vê na Figura 6. São cinco: avançar, recuar, girar à esquerda, girar à direita e parar. De uma forma intuitiva, ao pressionar o botão “Avançar”, o robô – uma vez que exista conexão – irá avançar indefinidamente, até que outro botão de movimentação seja pressionado.

Figura 6 – Tela inicial do software “Barrichello” – aba “Controle Remoto”



Fonte: autoria própria

Na programação do software, cada botão inicia um evento responsável por enviar para o robô o comando a ser efetuado. Como exemplo, tem-se para o botão avançar o código presente na Figura 7.

Figura 7 – Excerto de código: evento de avançar

```
private: System::Void bava_Click(System::Object^  sender,
System::EventArgs^  e) {
   if (this->serial->IsOpen == true){
      check = ' ';                  // zerar a confirmação de recebimento
      done = false;                 // zerar a confirmação de execução
      System::String^ grupo;
      grupo = this->grpset->Text;
      this->serial->Write(grupo);   // grupo
      System::Threading::Thread::Sleep(50);
      this->serial->Write("I");           // comando
      System::Threading::Thread::Sleep(50);
      this->serial->Write("000");   // giros (nulo)
      System::Threading::Thread::Sleep(50);
      this->strob->Text = L"Executando tarefa"; // altera o status do robô
      this->stgrp->Text = grupo;                // define o grupo vigente
      try {
         check = this->serial->ReadChar();      // espera confirmação
         if (check == 'D')          // se a confirmação for positiva
         done = true;               // a tarefa é dada como executada
      }
      catch (TimeoutException ^) {        // se não houver confirmação
         MessageBox::Show("O robô não respondeu ao comando.\n\nVerifique
a conexão e/ou a seleção do grupo e tente novamente.","Notificação",
         MessageBoxButtons::OK, MessageBoxIcon::Hand);
      }
      this->stcom1->Text = this->stcom2->Text;  // registrar no histórico
      this->stcom2->Text = this->stcom3->Text;
      this->stcom3->Text = this->stcom4->Text;
      this->stcom4->Text = this->stcom5->Text;
      this->stcom5->Text = this->stcom6->Text;
      if (done == true){
         this->stcom6->Text = L"Avançar";       // em caso de sucesso
         this->strob->Text = L"Avançando";      // Muda o status do robô
      }
      else{
         this->stcom6->Text = L"Erro de comunicação"; // em caso de falha
         this->strob->Text = L"Desconhecido";         // retorna o status
                                                      do robô para
                                                      desconhecido
      }
   }
}
```

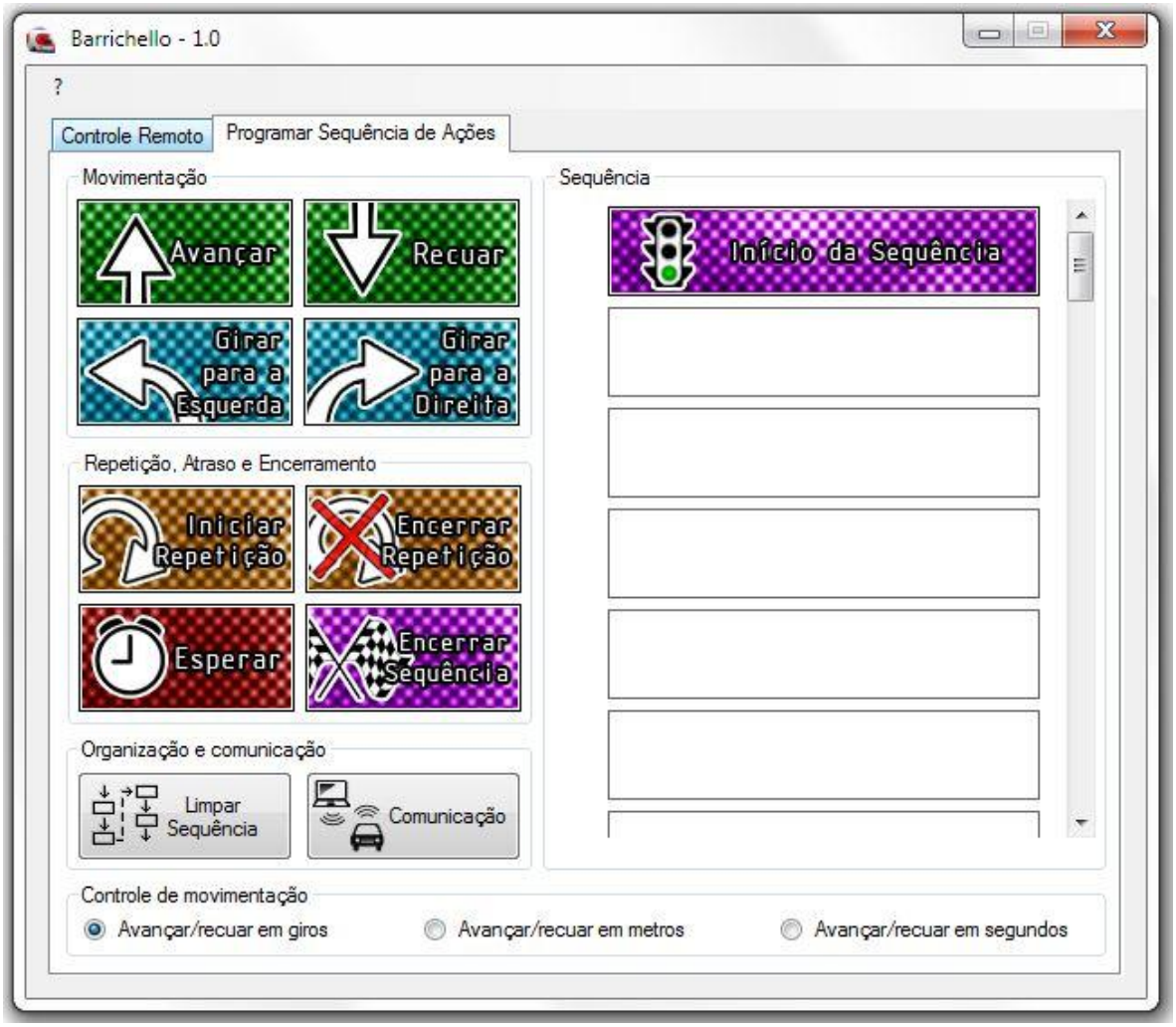
Fonte: autoria própria

A forma como a comunicação ocorre é explanada na seção de protocolo de comunicação.

4.4.1.3 Criar sequências de ações

Embora não seja tão avançado, esse modo de controle – visto na Figura 8 – já permite uma maior possibilidade de arranjos de sequências de ações. Nele, o usuário tem "lacunas" à direita que serão preenchidas com diferentes ações que podem ser encontradas à esquerda. As ações disponíveis se dividem em ações de movimentação: avançar, recuar, girar para a direita, girar para a esquerda; ações de repetição, atraso e encerramento: iniciar (e encerrar) repetição, esperar e encerrar a sequência. Além disso, também existe a opção de limpar a sequência - para que se possa recomeçar do zero.

Figura 8 – Aba "Programar Sequência de Ações"



Fonte: autoria própria

Para "montar" a sequência de ações - ou o algoritmo a ser seguido pelo robô - o usuário só precisa executar o movimento de clicar e segurar a ação desejada e arrastá-la até a

lacuna referente à posição da ordem em que ele deseja que a mesma seja executada. Após uma lacuna ser preenchida com uma ação, surge também um contador numérico que receberá o valor desejado associado àquela ação - como o número de giros que o robô deve avançar, ou a quantidade de vezes que ele deve repetir determinado grupo de tarefas.

Para as ações de avançar e recuar, ainda é possível escolher se o robô irá avançar ou recuar em metros, em giros de roda ou em segundos. Uma vez que a sequência está concluída, o usuário adiciona o indicador de fim de sequência, para que a mesma possa ser, então, enviada ao robô.

Embora o software trabalhe com a "montagem" do código de forma modular e visual, a estrutura da sequência procura simular uma estrutura de código textual, necessitando de um indicador de início e fim de programa, e de indicadores de abertura e encerramento de laços de repetição. Dessa forma, mesmo não tendo noções de programação, o usuário pode criar estruturas como ciclos de repetição, atrasos, repetições dentro de repetições, tal qual estivesse programando em alguma linguagem textual. A própria interface procura ajudar o usuário ao máximo, indicando onde existem repetições iniciadas que não foram encerradas, impedindo o usuário de inserir um encerramento de repetição antes que uma repetição seja aberta, e representando visualmente quais comandos cada laço de repetição. Um exemplo pode ser visto na Figura 9.

Figura 9 – Exemplo de representação de laço de repetição dentro de laço de repetição

Fonte: autoria própria

4.4.1.4 Criar e selecionar grupos para executar tarefas distintas

Presente na aba “Controle Remoto”, esta funcionalidade permite a criação, seleção e associação de grupos de robôs, caso se deseje controlar vários robôs no mesmo programa. É possível criar grupos nomeados por um caractere, variando de A a F e de 0 a 9, num máximo de 16 grupos. Após criar o grupo, o usuário pode associar o grupo escolhido aos robôs que desejar, além de escolher para qual grupo irá mandar os comandos de movimentação.

4.4.1.5 Verificar a lógica da sequência de ações

Após a sequência ser criada, cabe ao programa verificar se a mesma é coerente e se pode ser compreendida e enviada para o robô. Tal verificação é importante por checar se todos os lações de repetição estão devidamente encerradas, assim como os eventuais laços de repetição dentro de outros laços de repetição. Devido à possibilidade do usuário poder sobrescrever uma lacuna já preenchida com outro comando, é possível que haja uma quebra de alguma estrutura de repetição previamente bem definida. Para evitar quaisquer problemas, a verificação do código é obrigatória e se dá por meio do botão “Verificar Erros”, disponível na área de controle, que surge após pressionar o botão “Comunicação”. Tal área de controle pode ser vista na Figura 10.

Figura 10 – Opções de conexão e controle dentro da aba “Programar Sequência de Ações”

Fonte: autoria própria

Uma vez que o código é verificado, a sequência pode então ser enviada ao robô. Caso o usuário sobrescreva alguma ação na sequência já verificada, é necessário que a mesma seja verificada novamente. Caso apenas sejam feitos ajustes de contadores – como mudança na quantidade de giros de determinada ação – não se faz necessária uma nova verificação.

4.4.1.6 Enviar sequências feitas

Uma vez que a sequência foi construída e verificada, a mesma pode ser enviada para o robô. As opções de envio são os botões: iniciar sequência, pausar/retomar sequência, e parar sequência. Tais botões também se encontram nas opções de conexão e controle mostradas na Figura 10.

Devido ao fato do kit de robótica educacional utilizado não poder receber dados de programação de PIC de forma sem fio, a sequência criada pelo usuário é enviada para o robô na forma de blocos de comandos que são enviados gradativamente, de forma que o software "emula" – de forma automatizada – um usuário clicando nas mesmas setas de movimentação encontradas na aba de controle remoto. A interpretação lógica de laços de repetição é feita no próprio software, de modo que o robô em si só recebe instruções simples. Para facilitar o acompanhamento da execução da sequência, o software destaca a lacuna cujo bloco de comando está sendo enviado e/ou executado pelo robô ao longo da execução do algoritmo.

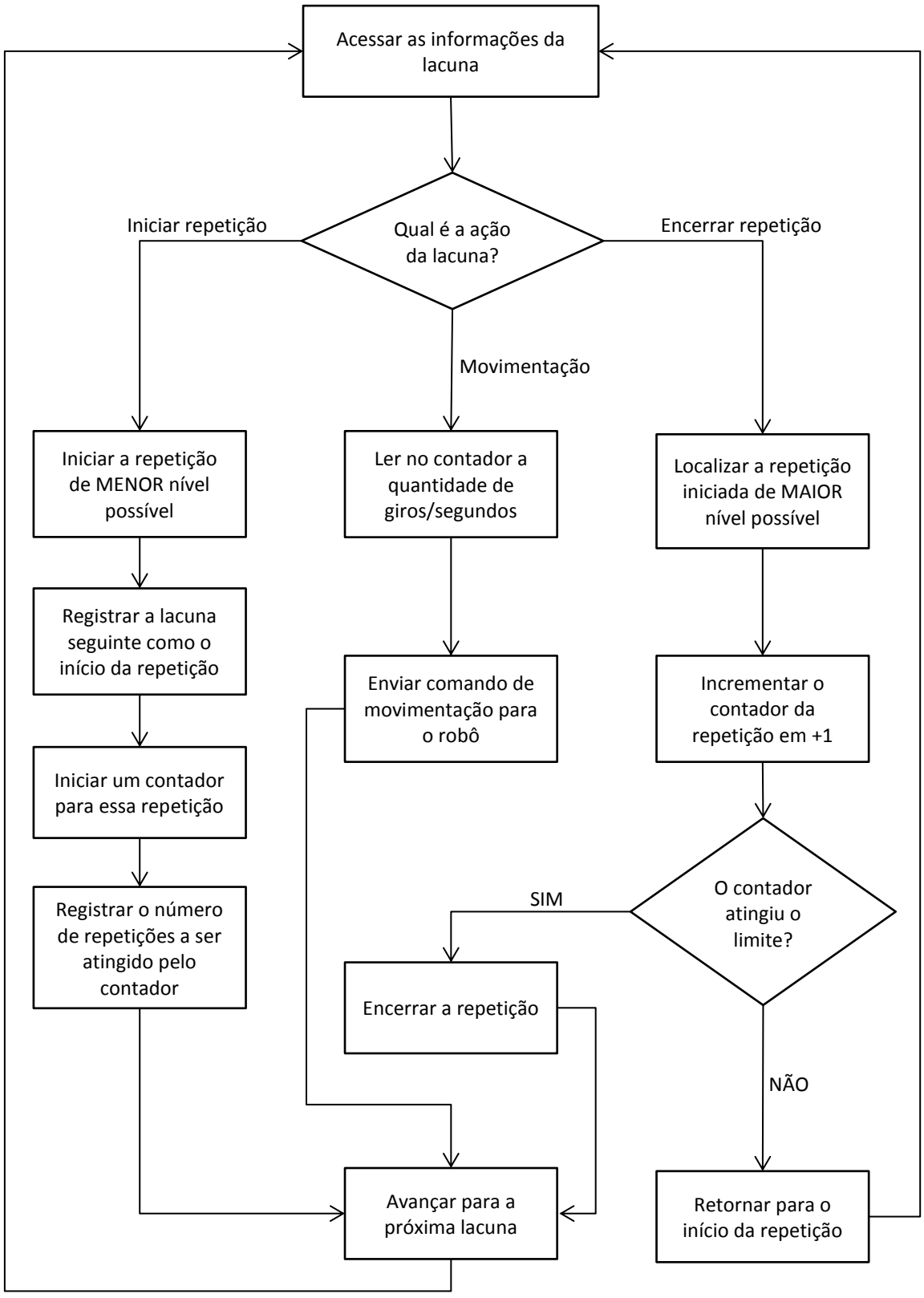
A qualquer momento da execução o usuário pode pausar ou interromper o envio de comandos. Caso o botão "Pausar Sequência" seja pressionado, o robô irá parar logo após executar a última instrução enviada pelo programa, ficando ocioso até o botão "Retomar Sequência" ser pressionado. Caso o botão "Parar Sequência" seja pressionado, o envio da sequência é abortado e o robô irá parar logo após executar a última instrução enviada pelo programa.

4.4.2 Funcionamento interno do programa Barrichello

4.4.2.1 Interpretação das lacunas da sequência de ação

Para saber qual instrução enviar para o robô, o programa analisa informações associadas às lacunas quando cada uma é preenchida com uma ação de forma linear, começando da primeira lacuna e seguindo de forma crescente até a última, eventualmente voltando para lacunas anteriores em caso de laços de repetição. Um fluxograma do funcionamento desta análise pode ser visto na Figura 11.

Figura 11 – Fluxograma de interpretação de lacunas



Fonte: autoria própria

É importante ressaltar que o fluxograma exibido só é aplicável enquanto houverem lacunas. Após chegar à última lacuna – Encerrar Sequência, obrigatoriamente – o software finaliza a sequência. Além disso, durante as ações de “Esperar”, o programa simplesmente aguarda o tempo indicado, não enviando nenhum comando para o robô.

4.4.2.2 Protocolo de comunicação utilizado

Devido ao fato do software Barrichello ter sido desenvolvido para ser usado em conjunto com a plataforma criada pelo aluno Marcos Vinicius Alves Costa, ambos se comunicam de uma forma diferente dos outros protocolos de comunicação utilizados por outros kits de robótica educacional disponíveis no mercado. O protocolo de comunicação usado pelo Barrichello para se comunicar com o robô foi concebido quando da idealização do software.

O software e o robô se comunicam através do envio e recebimento de caracteres na forma de blocos de instrução. Um bloco de instrução completo, quando enviado pelo software, contem cinco caracteres que são enviados no padrão RS-232. O primeiro caractere desse bloco indicará o grupo ao qual esse bloco de comando é destinado, ou conterá um caractere que indica associação de grupo. O segundo caractere indica o tipo da ação – avançar, recuar, girar à direita, girar à esquerda ou parar – ou o valor do novo grupo a ser associado. Os três caracteres seguintes, necessariamente números, estão associados ao tipo da ação, e informam ao robô quantos giros serão dados ou quantos graus serão girados.

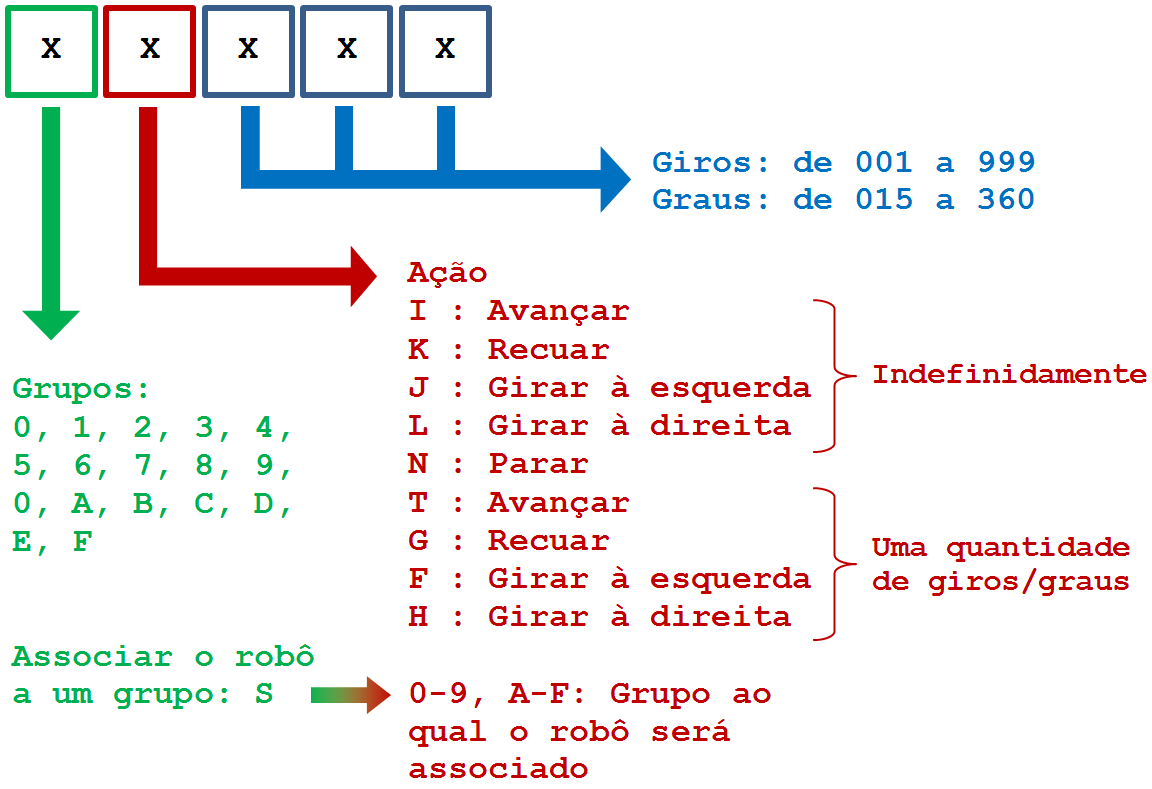
Os caracteres enviados pelo software são registrados pelo microcontrolador do robô em um vetor. Nenhuma ação é realizada até o vetor conter cinco caracteres. Uma vez que o vetor atinja os cinco caracteres, o robô analisa o bloco de instrução e realiza a ação correspondente ao mesmo. Os possíveis caracteres de um bloco são mostrados na Figura 12.

Perceba-se que caso o segundo caractere seja um comando de parar ou realizar ação indefinidamente – N, I, K, J ou L –, os três caracteres seguintes, que indicam a quantidade de giros ou graus, tornam-se irrelevantes. Todavia, esses três últimos caracteres ainda devem ser enviados para que se possa completar o bloco de instruções.

Após enviar o bloco de instruções para o robô, o software aguarda uma confirmação de recebimento e execução de tarefa do mesmo. Após receber, validar e executar a tarefa, o robô envia ao software um caractere “D”. Para comandos que envolvem giros ou graus, o robô também envia um caractere “W” para cada giro que executa, antes de enviar o caractere “D” de conclusão de tarefa. Caso não receba esses caracteres, o software Barrichello informa

ao usuário que ocorreu algum erro de comunicação ou de execução. Dessa forma, após enviar, por exemplo, um bloco de instrução “AT005” – grupo A, avançar, 5 giros –, o programa esperará receber a resposta “WWWWWD” do robô. Para um comando de realizar ação indefinidamente, por exemplo “3L000” – grupo 3, girar à direita indefinidamente –, o programa esperará receber apenas a resposta “D”, dado que esse bloco de instrução não requer nenhum controle de giros ou graus.

Figura 12 – Mapa de caracteres do bloco de instruções



Fonte: autoria própria

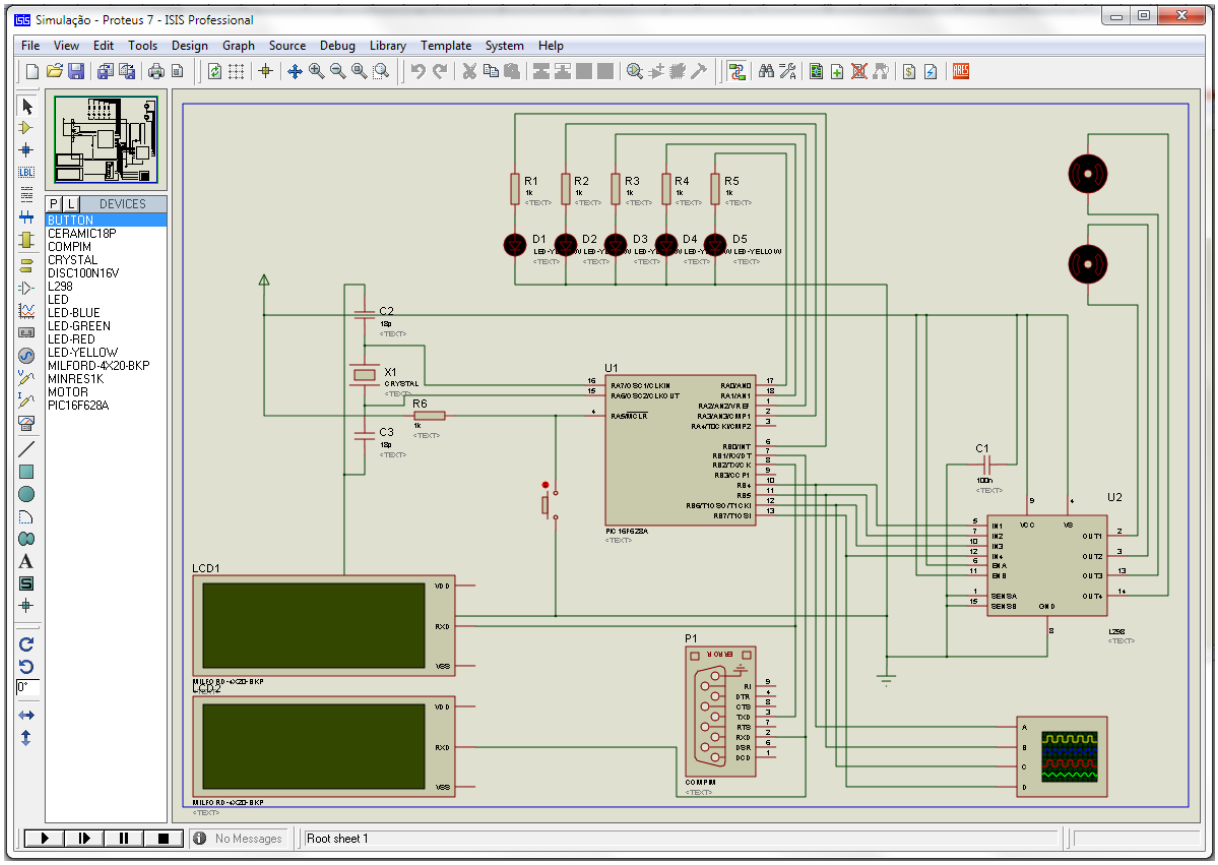
5 RESULTADOS E DISCUSSÕES

5.1 RESULTADOS EM SIMULAÇÕES

Enquanto estava em seu processo de desenvolvimento e implementação, o software de controle Barrichello era constantemente testado no ambiente de simulação *ISIS*, parte do programa *Proteus Design Suite 7*, da empresa *Labcenter Electronics*. As simulações envolviam o PIC16F628A, os motores, os LEDs de acionamento e a ponte-H responsável pelo acionamento. Para a parte de comunicação entre o Barrichello e a simulação, foi

utilizado o software gratuito *Virtual Serial Ports Emulator*, da empresa *Eterlogic.com*, que emulava uma porta serial no computador, estabelecendo a conexão entre o programa e o simulador. A Figura 13 mostra a tela da simulação no ISIS.

Figura 13 – Ambiente de simulação ISIS



Fonte: autoria própria

5.2 RESULTADOS EXPERIMENTAIS

O software Barrichello se comunicou com a plataforma utilizada da forma esperada. A conexão apresentou um tempo de resposta rápido, condizente com as taxas de transferência utilizadas. Entretanto, em alguns raros momentos ocorriam oscilações de tensão na alimentação do microcontrolador do robô que faziam com que o mesmo reiniciasse ou não registrasse algum caractere recebido de forma correta. Quando isso acontecia, embora a conexão continuasse sólida, a comunicação não ocorria como deveria, uma vez que o software sempre enviava os blocos de instrução em grupos de cinco caracteres mas as primeiras posições no vetor de caracteres do microcontrolador ainda estavam sendo utilizadas por caracteres pertencentes a um bloco antigo, fazendo assim com que os blocos de instrução

fossem interpretados como inválidos ou interpretados como instruções diferentes das que haviam sido enviadas.

5.3 CONSIDERAÇÕES SOBRE O SOFTWARE BARRICHELLO

Todo o processo de construção do Barrichello foi muito enriquecedor e gratificante. Desde as etapas de idealização inicial e concepção de funções às últimas etapas de programação, implementação e tratamento de exceções, a quantidade de conhecimento adquirido e empregado foi significativamente proveitosa.

Por ser um projeto novo, o Barrichello ainda não conta com certas funcionalidades mais avançadas encontradas em vários softwares utilizados em conjunto com kits de robótica educacional disponíveis comercialmente, entretanto, no ponto em que está, o programa já mostra que pode ser utilizado na robótica educacional em conjunto com a plataforma para o qual foi desenvolvido. Além disso, espera-se que tais funcionalidades sejam implementadas no futuro por quem quer que deseje usar o software Barrichello como base para algum trabalho acadêmico, tornando-o assim, o mais completo e didático possível.

REFERÊNCIAS

DEITEL, Harvey M.; DEITEL, Paul J. **C++: como programar**. Tradução de Carlos Arthur L. Lisbôa e Maria Lúcia L. Lisbôa. 3. ed. Porto Alegre : Bookman, 2001. 1098p.

JACOBSON, Ivar et al. **Object-Oriented Software Engineering: A Use Case Driven Approach**. 1. ed. Addison-Wesley, 1992. 528p.

BARROS, Renata Pitta. **RoboEduc: Uma ferramenta para programação de Robôs LEGO**. 2008. 51f. Monografia (Graduação em Engenharia de Computação) – Universidade Federal do Rio Grande do Norte, Natal, 2008.

SANTOS, Marcela Gonçalves dos. **RoboEduc: Um Software para Ensino da Robótica para Crianças Digitalmente Excluídas Utilizando Protótipos LEGO**. 2006. 45f. Monografia (Graduação em Engenharia da Computação) – Universidade Federal do Rio Grande do Norte, Natal, 2006.

FILHO, Cristóvão M. de O. L.; FREIRE, Raimundo C. S.; **Desenvolvimento de uma Plataforma Didática de Baixo Custo para o Ensino de Robótica**. UFCG, 2009. Disponível em: <http://limcserver.dee.ufcg.edu.br/semetro/www/pdf/52379_1.pdf >. Acesso em: 29 maio 2013.

AZEVEDO, Samuel et al. **Minicurso: Introdução a Robótica Educacional**. 2009. Disponível em: <http://www.sbpcnet.org.br/livro/62ra/minicursos/MC%20Samuel%20Azevedo.pdf>. Acesso em: 29 maio 2013.

STROUSTRUP, Bjarne. **The Design and Evolution of C++**. Addison-Wesley, 1994. 480p.

AGARWAL, Suprotim. **Drag and Drop Images Into a PictureBox**. 2008. Disponível em: <http://www.dotnetcurry.com/ShowArticle.aspx?ID=179>. Acesso em: 25 jun. 2013.

.NET - Pointer to picturebox - Stack Overflow. 2012. Disponível em: <http://stackoverflow.com/questions/13942520/pointer-to-picturebox>. Acesso em: 24 jul. 2013.

C++ - How to declare array of handel? - Stack Overflow. 2012. Disponível em: <http://stackoverflow.com/questions/9948325/how-to-declare-array-of-handel>. Acesso em: 25 jul. 2013.

Arrays of strings in Managed C++ - Stack Overflow. 2009. Disponível em: <http://stackoverflow.com/questions/995434/arrays-of-strings-in-managed-c>. Acesso em: 25 jul. 2013.

CCS :: View topic - pic16f628a/pwm. 2008. Disponível em: <http://www.ccsinfo.com/forum/viewtopic.php?p=92663>. Acesso em: 28 jul. 2013.

[Part] HC-06 Bluetooth to serial adapter - Tiny Labs - Electronic Projects. Disponível em: <http://www.tiny-labs.com/projects/electronics-projects/-part-hc-06-bluetooth-to-serial-adapter>. Acesso em: 31 jul. 2013

SILVA, Osmar J. **Origem da Linguagem C++**. 2004. Disponível em: <http://www.arquivodecodigos.net/dicas/c-origem-da-linguagem-c-1176.html>. Acesso em: 06 ago. 2013.

LIEBERKNECHT, Eduardo Augusto. **Robótica Educacional**. 2009. Disponível em: <http://www.portalrobotica.com.br/site/index.php?option=com_content&view=article&id=4&Itemid=2>. Acesso em: 06 ago. 2013.

SANTOS, Isaias. **O que é Robótica Educacional?** Disponível em: <http://www.roboticanaescola.com.br/>. Acesso em: 07 ago. 2013.

Um pouco de história sobre C/C++. Disponível em: <http://www.algoritmando.com.br/logica-de-programacao-teoria/um-pouco-de-historia-sobre-cc-6.html>. Acesso em: 07 ago. 2013.

MIT. **About Scratch**. Disponível em: <http://scratch.mit.edu/about/>. Acesso em: 08 ago. 2013.

MIT. **Scratch Project Editor - Imagine, Program, Share**. Disponível em: <http://scratch.mit.edu/projects/editor/?tip_bar=getStarted>. Acesso em: 08 ago. 2013.

MSDN. **Visual Studio Resources | MSDN**. 2013. Disponível em: <http://msdn.microsoft.com/en-us/vstudio/cc136611.aspx>. Acesso em: 09 ago. 2013.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **NBR 6023-2002: informação e documentação – referências – elaboração**. Rio de Janeiro: ABNT, 2002.

_____. **NBR 6024-2012: informação e documentação – numeração progressiva das seções de um documento escrito – apresentação**. Rio de Janeiro: ABNT, 2012.

_____. **NBR 6027-2007: informação e documentação – sumário – apresentação**. Rio de Janeiro: ABNT, 2003.

_____. **NBR 6028-2003: informação e documentação – resumo – apresentação**. Rio de Janeiro: ABNT, 2003.

_____. **NBR 10520-2002: informação e documentação – citações em documentos – apresentação**. Rio de Janeiro: ABNT, 2002.

_____. **NBR 14724-2011: informação e documentação – trabalhos acadêmicos – apresentação**. Rio de Janeiro: ABNT, 2011.