UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

ROBOTOY: APLICAÇÃO PARA PROGRAMAÇÃO E SIMULAÇÃO DE ROBÔS

JOÃO PAULO MACHADO

BLUMENAU
2017

JOÃO PAULO MACHADO

ROBOTOY: APLICAÇÃO PARA PROGRAMAÇÃO E SIMULAÇÃO DE ROBÔS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Ciência da Computação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

Profa. Joyce Martins, Mestre - Orientadora

BLUMENAU
2017

ROBOTOY: APLICAÇÃO PARA PROGRAMAÇÃO E SIMULAÇÃO DE ROBÔS

Por

JOÃO PAULO MACHADO

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Profa. Joyce Martins, Mestre – Orientadora, FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB
Membro:	Prof. Matheus Luan Krueger, Mestre – FURB

Blumenau, 4 de julho de 2017.

AGRADECIMENTOS

Aos meus pais, Adilson Machado e Marili Fátima Machado, por terem me apoiado e guiado em todas as etapas da minha vida e, em especial, na árdua jornada da graduação.

Ao meu irmão Luís Otávio Machado, pelos conselhos e incentivos relacionados à conclusão deste trabalho. A minha irmã Tamara Regina Machado, por sempre estar ao meu lado em fases difíceis.

Aos meus amigos, que serviram como usuários e testadores do aplicativo e sempre apoiarem a ideia.

Ao professor Aurélio Faustino Hoppe e a Juliana Carolina Batista, por terem emprestados os kits Lego Mindstorms NXT e Arduino, tornando assim possível a elaboração do trabalho.

A minha orientadora, Joyce Martins, por ter apresentado o tema e sempre ter acreditado na conclusão do mesmo, sempre me conduzindo com profissionalismo e competência.

Nunca deixe de tentar.

Michael Jordan

RESUMO

Este trabalho apresenta o desenvolvimento da ferramenta Robotoy, para programação e simulação de robôs. Inicialmente, a ferramenta possibilitava somente a programação de robôs Lego Mindstorms NXT (TORRENS, 2014). Foi implementada uma extensão para possibilitar a programação de robôs Arduino (BATISTA, 2016), além da criação de um ambiente de simulação 2D (SILVA, 2016). Neste sentido, foi proposta e implementada a integração dessas três ferramentas em único ambiente, adicionando o recurso de comando por voz. São reconhecidos por voz apenas alguns comandos da linguagem Robotoy, mas ainda é possível construir textualmente programas mais elaborados usando variáveis e rotinas. Para realizar o reconhecimento de voz, utilizou-se inicialmente a *engine* Julius e, posteriormente, a Microsoft Speech Platform, através da elaboração de uma gramática em XML. Alguns experimentos foram efetuados comparando o grau de confiança das sentenças reconhecidas pelas duas bibliotecas. Os resultados obtidos demonstraram que, em um ambiente sem ruído, ambas bibliotecas têm a mesmo desempenho. No entanto, em um ambiente com ruído, por exemplo, em uma sala de aula de uma escola da educação básica, a Microsoft Speech Platform mostrou ser mais adequada. Também foi elaborada uma comparação entre o tempo necessário para envio dos programas para as plataformas Lego Mindstorms NXT e Arduino.

Palavras-chave: Robótica educacional. Reconhecimento por voz. Programação e simulação de robôs.

ABSTRACT

This paper presents the development of the tool “Robotoy”, for programming and simulation of robots. Initially the tool only allowed the programing of Lego Mindstorms NXT robots (TORRENS, 2014). An extension on that was implemented to allow Arduino robots programming (BATISTA, 2016), and also an 2D environment for simulation (SILVA, 2016). In this sense the integration of these three tools in the same environment was presented and implemented, adding the voice recognition feature. Only few voice commands of the Robotoy Language are recognized, but it is still possible to to build textually elaborate programs using variable and routines. For the voice recognition it was used initially the engine Julius and, afterwards the Microsoft Speech Platform, through the creation of a grammar using XML. Some experiments were done comparing the degree of confidence of the sentences recognized by the libraries. The results displayed that in an noiseless environment both libraries have the same performance. However in a noisy environment, for example, in a classroom of an elementary school, the Microsoft Speech Platform proved to be more appropriated. It was also elaborated a comparison between the upload time of the programs to the platform Lego Mindstorms NXT and Arduino.

Keywords: Educational Robotic. Voice Recognition. Programming and Robot Simulation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

BNF – Backus Naur Form

COM – Component Object Model

SRGS – Speech Recognition Grammar Specification

UDP – User Datagram Protocol

USB – Universal Serial Bus

XML – eXtensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Atualmente, as pessoas vivem em uma era onde são surpreendidas no dia-a-dia pelos avanços tecnológicos e pelas facilidades proporcionadas. Nessa abordagem, o ambiente escolar é importante para a busca de um novo conhecimento. Para Azevedo, Aglaé e Pitta (2010, p. 18), "a escola precisa estar em consonância com a sociedade emergente, não podendo permanecer com as mesmas funções através do tempo e do espaço".

A inclusão da robótica como ferramenta pedagógica e educacional possibilita o desenvolvimento do raciocínio lógico e criativo. A robótica educacional ou robótica pedagógica apresenta ao educando e ao educador um instrumento ideal, por ser desafiador e lúdico na solução de problemas propostos, bem como na inovação de hardware e software (MIRANDA; SAMPAIO; BORGES, 2010). Essa ideia surgiu por volta da década de 1960, pelo protagonismo do matemático americano Seymour Papert, que aplicava a teoria do construcionismo com base nos pensamentos do epistemólogo suíço Jean Piaget.

Papert (1986 apud GOMES et al., 2010) propôs o uso de computadores, robôs e programação como forma motivadora e que atraía as crianças para o âmbito educacional. Essa abordagem passa a considerar a interdisciplinaridade no currículo escolar, proporcionando o conhecimento em diversas áreas, tais como: linguagem, matemática, física, entre outras. Mas para a inserção da robótica no ambiente educacional, se faz necessário o uso de materiais alternativos como sucata ou de kits específicos como o Lego Mindstorms ou o Arduino.

Com o objetivo de simplificar a programação para Lego Mindstorms NXT, Torrens (2014) desenvolveu uma linguagem e uma ferramenta de programação acessíveis para alunos das séries iniciais da educação básica, denominada Robotoy. Contudo, os kits Lego Mindstorms possuem um alto custo para aquisição. Assim, Batista (2016) adaptou a ferramenta Robotoy para permitir o uso de robôs Arduinos. O destaque do Arduino sobre outras plataformas de desenvolvimento, segundo McRoberts (2011, p. 20), "é a facilidade de sua utilização; pessoas que não são da área técnica podem, rapidamente, aprender o básico e criar seus próprios projetos em um intervalo de tempo relativamente curto". Outra proposta para viabilizar o uso da linguagem Robotoy dispensando a aquisição e o uso de um kit de robótica e, portanto, alavancando a inclusão de mais usuários, foi desenvolvida por Silva (2016). Este implementou um simulador 2D, que permite criar e editar cenários para a resolução de exercícios. Assim, com base nos argumentos exposto sobre o uso da robótica em ambientes educacionais, propõe-se a integração destas três ferramentas em uma única

aplicação, adicionando a entrada de comando por voz, como melhoria no quesito de usabilidade, para programação e simulação dos robôs.

1.1 OBJETIVOS

O objetivo deste trabalho é disponibilizar um ambiente integrado para a programação e a simulação de robôs na linguagem de programação Robotoy.

Os objetivos específicos do trabalho são:

a) disponibilizar uma interface para a elaboração de programas na linguagem Robotoy;
b) disponibilizar uma interface para a elaboração dos cenários 2D;
c) possibilitar a simulação e a programação de robôs através de comandos por voz;
d) gerar código para o simulador, para robôs Lego Mindstorms NXT e para robôs Arduino a partir dos programas escritos em Robotoy;
e) executar os programas gerados nos cenários 2D criados, nos robôs Lego Mindstorms NXT e nos robôs Arduino.

1.2 ESTRUTURA

Este trabalho está dividido em quatro capítulos: introdução, fundamentação teórica, desenvolvimento e conclusão. No segundo capítulo, tem-se a fundamentação teórica, apresentando uma visão sobre a robótica educacional e, em seguida, fala-se sobre reconhecimento de voz, descrevendo a *engine* Julius e a Application Programming Interface (API) Microsoft Speech Platform. Também são apresentados três trabalhos correlatos, assim como as ferramentas Robotoy que foram integradas nesse trabalho. Na sequência mostra-se o desenvolvimento desse trabalho, que contempla a especificação, a implementação da aplicação e do reconhecimento por voz, bem como os resultados obtidos. Por fim, encerra-se com a conclusão e possíveis extensões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo está divido em quatro seções. A seção 2.1 aborda uma visão geral sobre a robótica educacional, enquanto a seção 2.2 apresenta a evolução e as dificuldades relacionadas à área de reconhecimento de voz, bem como descreve as características e funcionalidades da *engine* Julius e da Microsoft Speech Platform. A seção 2.3 apresenta três trabalhos correlatos a este e a seção 2.4 descreve as três ferramentas para programação e a simulação de robôs em Robotoy.

2.1 RÓBOTICA EDUCACIONAL

Define-se a robótica educacional como um ambiente de aprendizagem que utiliza a tecnologia como instrumento para que os participantes possam vivenciar experiências semelhantes ao cotidiano. Neste ambiente, segundo Maisonnette (2002), o aluno é levado a observar, abstrair e inventar, criando modelos a partir de peças de brinquedos ou eletrodomésticos danificados, de peças de kits Lego ou Arduino e circuitos eletrônicos.

De acordo com Quintanilha (2008), a utilização da robótica como ferramenta pedagógica é extremamente difundida em determinados países, “Holanda e a Alemanha já têm a robótica pedagógica em 100% das escolas públicas. Inglaterra, Itália, Espanha, Canadá e Estados Unidos caminham na mesma direção”. Ainda conforme afirma o autor, países da América Latina, como por exemplo, México e Peru, deviam chegar a 3 mil escolas públicas com aulas de robótica em 2008.

Neste contexto, é importante tornar o aluno um agente ativo de conhecimento, possibilitando-o construir todo o seu processo de aprendizagem. Defende-se a inclusão da robótica educacional como ferramenta desta transformação, já que o aluno tem a responsabilidade da busca do entendimento necessário para que seu projeto alcance o objetivo proposto (CASTILHO, 2009).

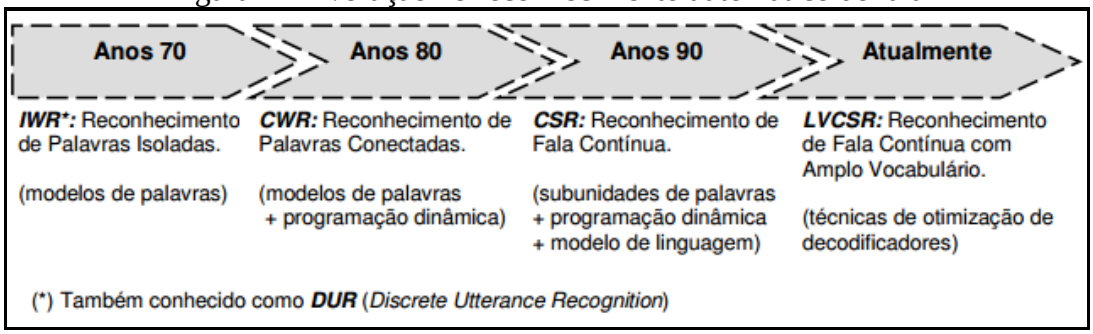
Contudo, pode-se identificar a robótica educacional como um método de aprendizagem que ultrapassa os limites da sala de aula, resolvendo assim problemas curriculares de aprendizagem contextualizada e duradoura. Além disso, a robótica educacional interfere de forma significativa na motivação dos alunos, uma vez que ensina de forma prática a aplicação da tecnologia em problemas corriqueiros (FORNAZA; WEBBER, 2014). O ensino de robótica educacional permite a elaboração de um currículo interdisciplinar, visto que para solucionar um problema o aluno deve ser capaz de relacionar diversos conhecimentos e competências, como por exemplo, ler, articular e interpretar símbolos ou código em diferentes linguagens. Diferentemente da metodologia tradicional de programação, o desenvolvimento

do raciocínio lógico torna-se dinâmico e desafiador proporcionando a criação de uma solução com o propósito de resolver problemas rotineiros. (SANTOS et al., 2010; SCHIVANI, 2014).

2.2 RECONHECIMENTO DE VOZ

A construção de sistemas para reconhecimento de voz por uma máquina tem o propósito da intepretação do sinal acústico, sendo o objeto de estudo de pesquisadores por mais de quatro décadas. A partir da década 70 do século passado, a área de reconhecimento automático de fala teve um significativo avanço, influenciado pelas áreas de processamento de sinais, algoritmos, arquitetura de software e hardware (TEVAH, 2006). A Figura 1 apresenta resumidamente os avanços das áreas nas últimas quatro décadas.

Figura 1 – Evolução no reconhecimento automático de fala



Fonte: Tevah (2006, p. 2).

Martins (1997) afirma que as pesquisas na área de reconhecimento automático de fala têm como propósito o desenvolvimento de uma máquina capaz de transcrever a fala, em um alto nível de precisão independente de ambiente e locutor, possibilitando a comunicação entre homem e máquina de forma similar ao ser humano. Este processo tem como entrada a forma acústica produzida pelo ser humano gerando dados na qual os processadores de dados possam interpretar. Segundo Bresolin (2003), um dos aspectos mais difíceis no desenvolvimento do reconhecimento da voz pela máquina é sua interdisciplinaridade natural com várias áreas do conhecimento. Contudo, existem atualmente sistemas para reconhecimento da fala com alto nível de precisão, atendendo a necessidade de compreender diversos idiomas. Porém estes softwares são demasiadamente caros e de código fechado. Como alternativa a essas ferramentas, tem-se a *engine* Julius, uma ferramenta de código aberto para reconhecimento da fala contínua, e a plataforma Microsoft Speech Platform, uma API para processamento e interpretação da voz humana.

A *engine* Julius, conforme descreve Lee (2009), é um software decodificador para reconhecimento da fala, com boa performance para grandes vocabulários. Desenvolvida na Kyoto University em 1998, está em aperfeiçoamento devido a desafios técnicos e as

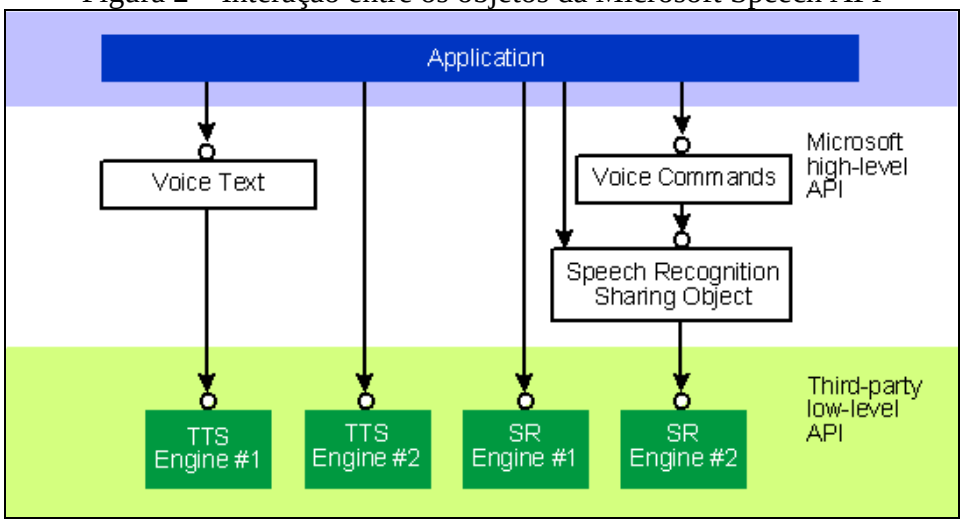
solicitações de melhorias dos usuários. Dispõe de versão tanto para Windows como para Linux, sendo que o reconhecimento de fala pode ocorrer tanto em tempo real através de um dispositivo como o microfone ou através de arquivos de áudio. O motor da aplicação foi escrito na linguagem C, podendo ser incorporado por outras aplicações através das manipulações dos eventos (LEE; KAWAHARA, 2009).

O funcionamento da *engine* Julius independe do idioma. Lee (2009) explica que, ao ser inicializada, a *engine* carrega o dicionário fonético, o modelo acústico e o modelo de linguagem, que são indicados através do arquivo de configuração. Após carregar esses arquivos, uma análise estatística é efetuada a partir das opções fornecidas no dicionário fonético, determinando qual a sequência de fonemas será disponibilizada para os usuários. A cada execução do decodificador são disponibilizados três resultados representando: o comando de voz em texto, o nível de confiança e o coeficiente de Vertebi, ou seja, a constante do algoritmo do cálculo da confiabilidade do reconhecimento de fonemas (PERICO; SHINOHARA; SARMENTO, 2014).

Já a Microsoft Speech Platform consiste em conjuntos de interfaces que permitem a inclusão do recurso de reconhecimento e síntese de fala em aplicações. Com suporte a vinte e seis idiomas, entre eles o português brasileiro, a API é especificada utilizando um conjunto de objetos Component Object Model (COM), possibilitando o acesso e controle através de qualquer aplicação desenvolvida sobre a plataforma .NET (MICROSOFT COMPORATION, 2016a). Para o reconhecimento de fala, deve-se definir quais sentenças serão reconhecidas na especificação de uma gramática usando a Speech Recognition Grammar Specification (SRGS), um arquivo em formato eXtensible Markup Language (XML). Esse arquivo pode ser compilado em formato binário para otimizar a aplicação (MICROSOFT COMPORATION, 2016b). A cada sentença processada pela API, um evento é disparado informando o grau de confiança e a sentença reconhecida.

Conforme Rozak (1996), os objetos da Microsoft Speech Platform podem ser divididos em dois grupos pelo seu nível de acesso: os objetos de alto-nível, que têm como objetivo facilitar o controle do mecanismo de voz, e os objetos de baixo nível, que realizam a comunicação diretamente com os mecanismos de fala e *drivers* da placa de som. Sendo assim, para o funcionamento da API, torna-se necessária a utilização de um motor de reconhecimento de fala compatível com as interfaces. Na Figura 2 apresenta-se a interação entre objetos do reconhecimento e síntese de fala.

Figura 2 – Interação entre os objetos da Microsoft Speech API



Fonte: Rozak (1996).

2.3 TRABALHOS CORRELATOS

São apresentados três trabalhos correlatos, que possuem características semelhantes à proposta deste trabalho: o RoboEduc (SILVA, 2009), software educacional para a programação de robôs desenvolvido na Universidade do Rio Grande do Norte; o Webots (CYBERBOTICS, 2016), um ambiente de desenvolvimento para modelagem, simulação e programação de robôs; o RoboMind FURB (BENITTI et al., 2008), um *plugin* para o RoboMind que permite o envio dos programas a robôs Lego MindStorm NXT.

2.3.1 Roboeduc

Silva (2009) descreve o RoboEduc como um software educacional que serve como mediador em atividades de robótica educativa. A utilização do RoboEduc inicia-se com a seleção do protótipo do robô, podendo ser um dos modelos previamente cadastrados, tais como robô carregador, robô segurador, robô lixeira ou robô batedor, ou através da construção de um novo modelo, a partir da escolha dos componentes de hardware, como atuadores, sensores e bases que deverão compor o protótipo.

Em seguida, o usuário pode determinar a forma de comandar o robô, entre controlar ou programar. Ao ser selecionada a opção para controlar o robô, inicia-se a comunicação de forma imediata entre o robô e o RoboEduc, sendo apresentados um conjunto de comandos para movimentação, em forma de ponteiros indicando as direções que o robô deve percorrer. Já a opção para programar o robô permite a elaboração de um conjunto de instruções a serem compiladas sem a necessidade de uma interação do usuário com o robô. As plataformas de

robôs suportadas pelo software são: Lego Mindstorms NXT, Lego RCX e H-EDUC, protótipo de plataforma desenvolvida pelos idealizadores do RoboEduc.

Conforme afirma Silva (2009), o ambiente possibilita a criação de programas através de componentes gráficos, que representam cada ação executada pelo robô, ou pela escrita de código nas linguagens de programação RoboEduc ou C. O software possui cinco níveis de programação, sendo que os níveis 1, 2 e 4 permitem a programação através de componentes gráficos por meio do mecanismo arrasta-e-solta. O nível 1 incorpora comandos idênticos à opção para controlar o robô, sendo que cada instrução é armazenada para posterior compilação e envio ao robô. O nível 2, apresentado na Figura 3, inclui as estruturas de controle de fluxo e o nível 4 funções semelhantes às do RoboLab, ambiente de programação vendido com os kits Lego Mindstorms. Já os níveis 3 e 5 englobam a programação textual, sendo que no nível 3 pode-se programar em RoboEduc e no nível 5 em C.

Figura 3 – Nível 2 para programação de robôs

Fonte: Silva (2009, p. 62).

A linguagem RoboEduc foi especificada com o intuito de diminuir a complexidade encontrada nas linguagens de programação já existente. Conforme Barros (2011, p. 21), "ao contrário de outras linguagens, esta [RoboEduc] permite a pessoa programar sem a necessidade de muitos conhecimentos prévios". Os programas desenvolvidos na linguagem RoboEduc são traduzidos para C. Posteriormente, o software envia o código C para o compilador BrickOS para que possa então ser enviado aos robôs. O Quadro 1 apresenta um programa desenvolvido na linguagem RoboEduc, onde a movimentação do robô é efetuada através da indicação do tempo que o robô deve executar determinado comando. Assim, `esquerda 1 segundos` comanda o robô para virar à esquerda durante um segundo.

Comandos de repetição e seleção podem ser adicionados, contudo o código para interação com sensores deve ser escrito em C.

Quadro 1 – Programa na linguagem RoboEduc

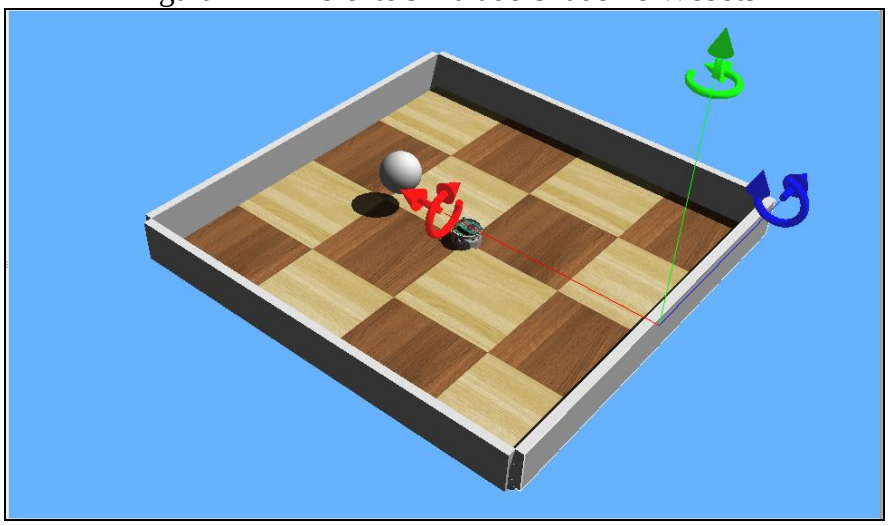
```
tarefa RoboMexer
inicio
   para i := 0 ate 2 faca
       frente 3 segundos
       esquerda 1 segundos
   fimpara
fim
```

Fonte: elaborado pelo autor.

2.3.2 Webots

O Webots é um ambiente de desenvolvimento tridimensional criado pela CyberBotics (2016) que possibilita modelagem, programação e simulação de robôs. Possui um ambiente virtual fisicamente realista, ou seja, o cenário simulado aplica propriedades da física, como por exemplo, distribuição de massa, coeficiente de atrito e inércia para torná-lo mais próximo da realidade. O ambiente de simulação permite a inclusão de objetos com diversas formas geométricas em uma estrutura de grafo de cena para interação com os robôs a serem desenvolvidos. Estes objetos podem ser classificados em: segmentos de entrada, semáforos e luzes, objetos da natureza, construções e seus elementos. Na Figura 4 pode-se observar um cenário contendo um objeto na forma de bola em uma estrutura de tabuleiro.

Figura 4 – Ambiente simulado criado no Webots



Fonte: CyberBotics (2016).

Denominados como controladores, os programas desenvolvidos no Webots devem ser escritos nas linguagens C, C++, Java, Python, URBI ou MATLAB$_{TM}$. Os controladores podem ser desenvolvidos no ambiente integrado ao software que, após a verificação de erros, libera a opção para envio imediato dos programas a diversas plataformas de robôs, tais como Khepera, Hemisson e Lego Mindstorms (WING, 2011). Diferente dos softwares para robótica

educacional, a complexidade na elaboração dos programas torna-se alta, pois pressupõe-se que o usuário possua conhecimento na linguagem de programação a ser utilizada. No Quadro 2 é apresentado um programa em C para apresentação de uma mensagem simples. O objetivo do código mostrado é o envio da mensagem "Olá Mundo utilizando o Webots" para o console do simulador, exibindo a mensagem a cada intervalo de um segundo.

Quadro 2 – Programa em C desenvolvido no Webots

```
#include <webots/robot.h>
#include <stdio.h>

int main(){
   wb_robot_init();
   while(1){
      printf("Olá Mundo utilizando o Webots!\n");
      wb_robot_step(1000);
   }
   return 0;
}
```

Fonte: elaborado pelo autor.

A modelagem de robôs no Webots permite a criação de inúmeros protótipos de robôs com diferentes tipos de sensores e atuadores. Pode-se citar como exemplo o uso de sensores de distância e toque, câmeras, entre outros.

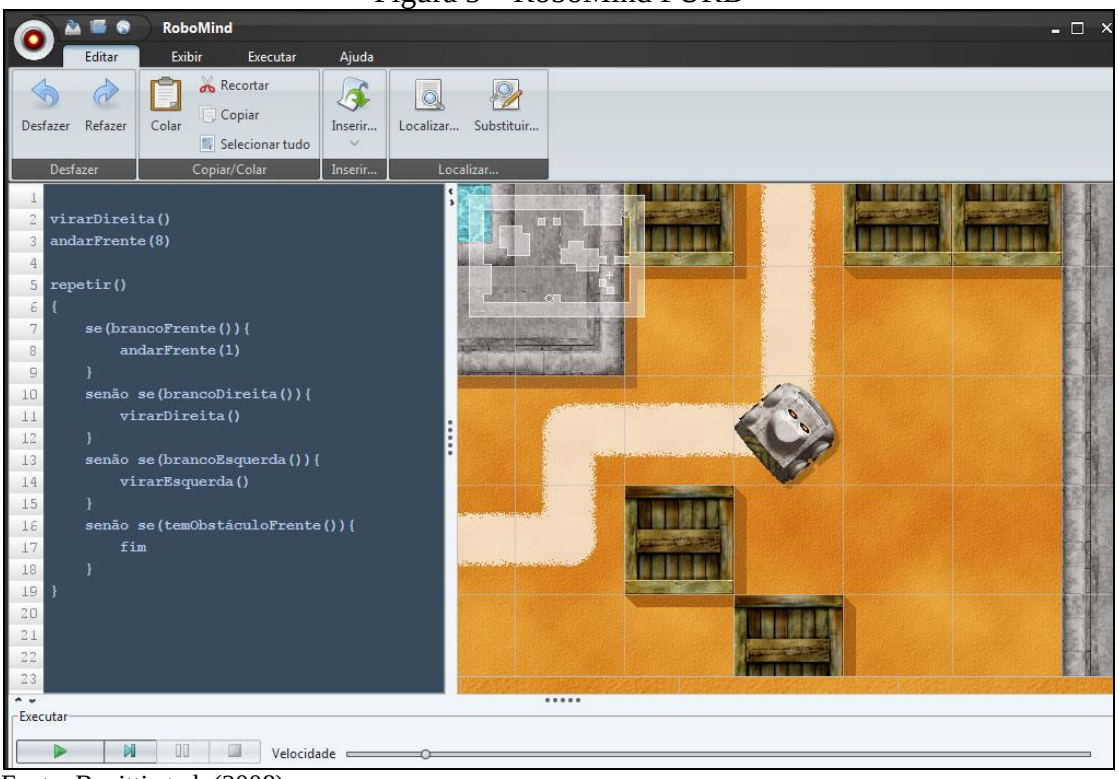
2.3.3 Robomind FURB

Elaborado por Arvid Hama na Universidade de Amsterdam (BENITTI et al., 2008), o RoboMind é um software educacional que possui uma linguagem de programação simples, denominada Robo, para a programação e o desenvolvimento de robôs em um ambiente bidimensional. Inspirado pela linguagem Logo, Robo foi projetada com o intuito de permitir a programação de forma imediata. No que se refere à modelagem dos robôs, no RoboMind é possível adicionar sensores, como o ultrassônico para identificação de obstáculos à frente, bem como atuadores, que permitem a manipulação de objetos no cenário (ROBOMIND ACADEMY, 2014).

Já o RoboMind FURB, conforme descreve Benitti et al. (2008), é uma extensão do software original. Desenvolvido no âmbito do Departamento de Sistemas e Computação (DSC) da Universidade Regional de Blumenau (FURB), a extensão elaborada promove o envio dos códigos aos robôs para execução da tarefa. Antes disso, deve ser feita a seleção, dentre vários robôs desenvolvidos, de um modelo para realizar o experimento.

Os programas devem ser desenvolvidos no editor de texto do ambiente, que dispõe, à sua direita, de um cenário para simulação dos comandos. Na Figura 5 é apresentada a interface gráfica do RoboMind FURB.

Figura 5 – RoboMind FURB



Fonte: Benitti et al. (2008).

Os comandos a serem usados variam em função do modelo de robô selecionado, mas toda a mecânica tem como base a indicação de quantidade de casas que o robô deve percorrer. No Quadro 3 é apresentado um conjunto de instruções que podem ser utilizadas para desenhar a letra "T" no cenário. Primeiramente é ativada a função do pincel branco para colorir o chão através do comando `pintarBranco`, as próximas instruções realizam a movimentação do robô, e, por fim, a função `pararPintar` desativa o recurso de colorir o chão.

Quadro 3 – Instruções do RoboMind FURB

```
pintarBranco()
andarFrente(3)
virarDireita()
andarFrente(2)
andarTrás(4)
pararPintar()
```

Fonte: elaborado pelo autor.

2.4 FERRAMENTAS ATUAIS

Desenvolvido por Torrens (2014), o Robotoy é um software educacional que tem objetivo permitir que crianças elaborem programas para robôs Lego Mindstorms NXT. Com uma linguagem de programação simplificada, a ferramenta evidencia o foco na solução dos problemas propostos.

No Quadro 4 encontra-se um programa desenvolvido na linguagem Robotoy. O programa soluciona o seguinte problema: o robô deve mostrar no display a quantidade de

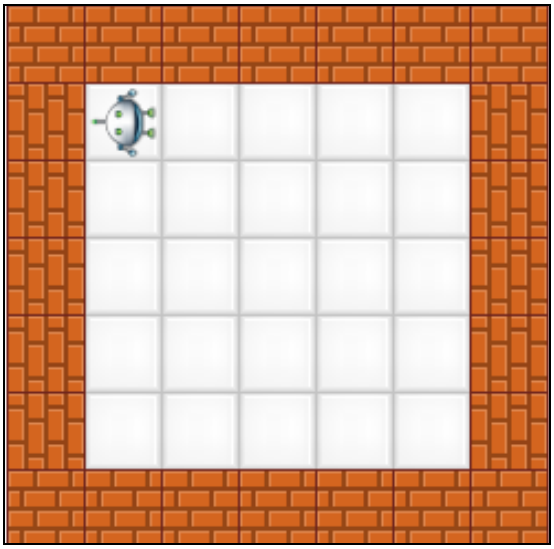
colunas e de linhas que um cenário possui (TORRENS, 2014, p. 71). A Figura 6 traz um possível cenário com o estado inicial do robô.

Quadro 4 – Programa em Robotoy para contagem de colunas e linhas

```
número linhas <- 1
número colunas <- 1

enquanto não tem obstáculo
   andar para frente 1
   colunas <- colunas + 1
fim do enquanto

virar para a direita 1
enquanto não tem obstáculo
   andar para frente 1
   linhas <- linhas + 1
fim do enquanto

texto qtdLinhas <- "Linhas: " . linhas
texto qtdColunas <- "Colunas: " . colunas
escrever qtdLinhas
escrever qtdColunas
emitir som
```

Fonte: Torrens (2014, p. 71).

Figura 6 – Cenário para contagem de colunas e linhas



Fonte: adaptado de Torrens (2014, p. 71).

Para desenvolvimento da ferramenta Robotoy, Torrens (2014) definiu alguns requisitos, sendo que os principais são:

> a) disponibilizar uma linguagem de programação para que o usuário possa programar (Requisito Funcional - RF);
> b) disponibilizar um software para que o usuário possa manipular os programas definidos na linguagem (RF);
> c) disponibilizar uma camada intermediária que viabilize o suporte a várias plataformas de robôs (RF);
> [...]
> e) permitir que programas definidos na linguagem possam ser implantados em robôs LEGO Mindstorms NXT (RNF);
> f) realizar a implantação de código no robô LEGO Mindstorms NXT através da API leJOS (RNF);
> g) possuir a inteligência do robô LEGO Mindstorms NXT implementada com a API leJOS (RNF); [...] (TORRENS, 2014, p. 27).

A linguagem Robotoy foi definida para viabilizar a execução em várias plataformas de robôs, bem como permitir a montagem de qualquer tipo de robô, sendo o único componente obrigatório o *brick* para robôs Lego Mindstorms NXT. Segundo LEGO Group (2016), o kit Lego Mindstorms NXT 2.0 possui, além do *brick*, 3 servo motores, 1 sensor ultrassônico, 2 sensores de toque, 1 sensor de cor e outras 612 peças para montagem dos robôs, conforme ilustrado na Figura 7.

Figura 7 – Kit Lego Mindstorms NXT

Fonte: Plug (2016).

A partir dessa característica, Batista (2016) estendeu a ferramenta desenvolvida por Torrens (2014) para permitir "[...] que um mesmo programa possa ser executado tanto em robôs do tipo Lego Mindstorms NXT como em robôs construídos com base na placa Arduino Mega 2560 [...]" (BATISTA, 2016, p. 6). Isto é, foi adicionado suporte à programação de robôs Arduino. Nesse sentido, foram incluídas na ferramenta as seguintes funcionalidades: (a) escolha da plataforma de robô a ser utilizada (Lego Mindstorms NXT ou Arduino); (b) possibilidade de configuração do robô Arduino; (c) tradução direta dos comandos da linguagem Robotoy para a linguagem do Arduino. Dessa forma, estabelece os seguintes requisitos principais para estender a Robotoy:

> a) gerar código para Arduino a partir dos comandos da linguagem Robotoy (Requisito Funcional - RF);
> b) disponibilizar uma tela para que o usuário escolha para qual plataforma será gerado o programa: Lego Mindstorms NXT ou Arduino (RF);
> c) disponibilizar uma tela de configuração para o robô Arduino (RF);
> d) fazer o envio do programa traduzido para o Arduino (RF);
> e) executar os programas em robôs montados com base no Arduino (Requisito Não Funcional - RNF); [...] (BATISTA, 2016, p. 24).

No entanto, independentemente da plataforma de robô usada, o custo[1] ainda pode ser elevado para adoção da Robotoy como ferramenta de ensino-aprendizagem por escolas de

[1] Batista (2016) gastou aproximadamente R$480,00 para adquirir um kit Arduino Mega 2560 (placa e demais componentes). Enquanto um kit Lego Mindstorms EV3, similar ao NXT, conforme afirma a autora, é vendido na loja virtual oficial da Lego por $349,99, ou seja, aproximadamente R$1.133,93.

educação básica. Assim, para viabilizar o uso da Robotoy, Silva (2016) desenvolveu um simulador 2D, facilitando a validação dos exercícios em cenários complexos. Para que isso ocorra, implementou-se um conversor para uma linguagem intermediária em formato script, simulando um interpretador de comandos (SILVA, 2016).

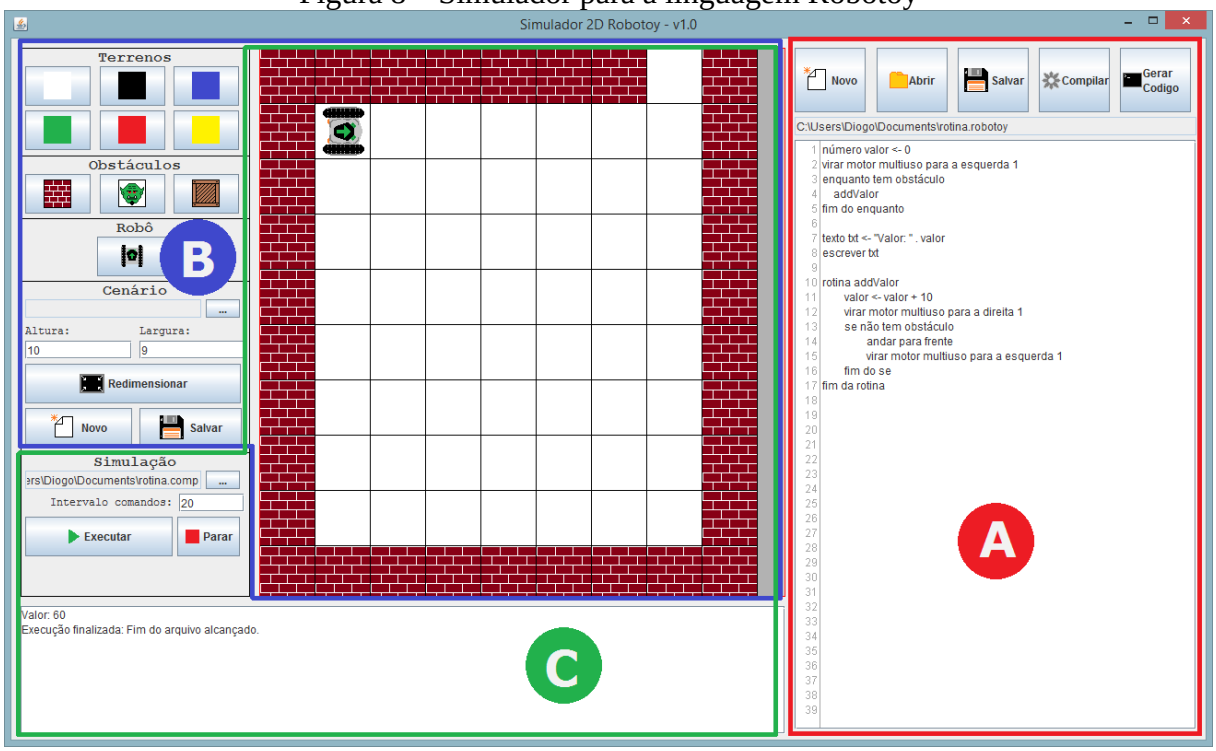
Na Figura 8 pode-se observar que o simulador contém um editor de cenário para a inclusão de obstáculos, alteração de terreno e redimensionamento do campo. Além disso, o cenário só pode conter apenas um robô que, após ser selecionado, pode ser rotacionado por inteiro ou apenas a sua cabeça. Foram definidos como principais requisitos do simulador 2D:

a) possuir uma interface para a elaboração de programas na linguagem Robotoy [visto na Figura 8(a)] (Requisito Funcional - RF);
b) possuir uma interface para criar e editar cenários 2D a fim de serem a base das simulações [visto na Figura 8(b)] (RF);
c) traduzir os programas Robotoy em um *script* que possa ser interpretado pelo simulador 2D (RF);
d) executar os programas traduzidos no simulador 2D [sendo a área para controle e execução da simulação vista na Figura 8(c)] (RF);

[...]

h) utilizar a linguagem Robotoy como linguagem de alto nível para programar as ações dos robôs que serão simuladas (RNF). (SILVA, 2016, p. 23).

Figura 8 – Simulador para a linguagem Robotoy



Fonte: Silva (2016, p. 29).

3 DESENVOLVIMENTO

Este capítulo está divido em cinco seções, onde são apresentados os principais requisitos da ferramenta desenvolvida, a especificação da ferramenta, a especificação e a implementação do reconhecimento por voz, as ferramentas utilizadas e a operacionalidade da implementação e os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DA APLICAÇÃO

A aplicação deve:

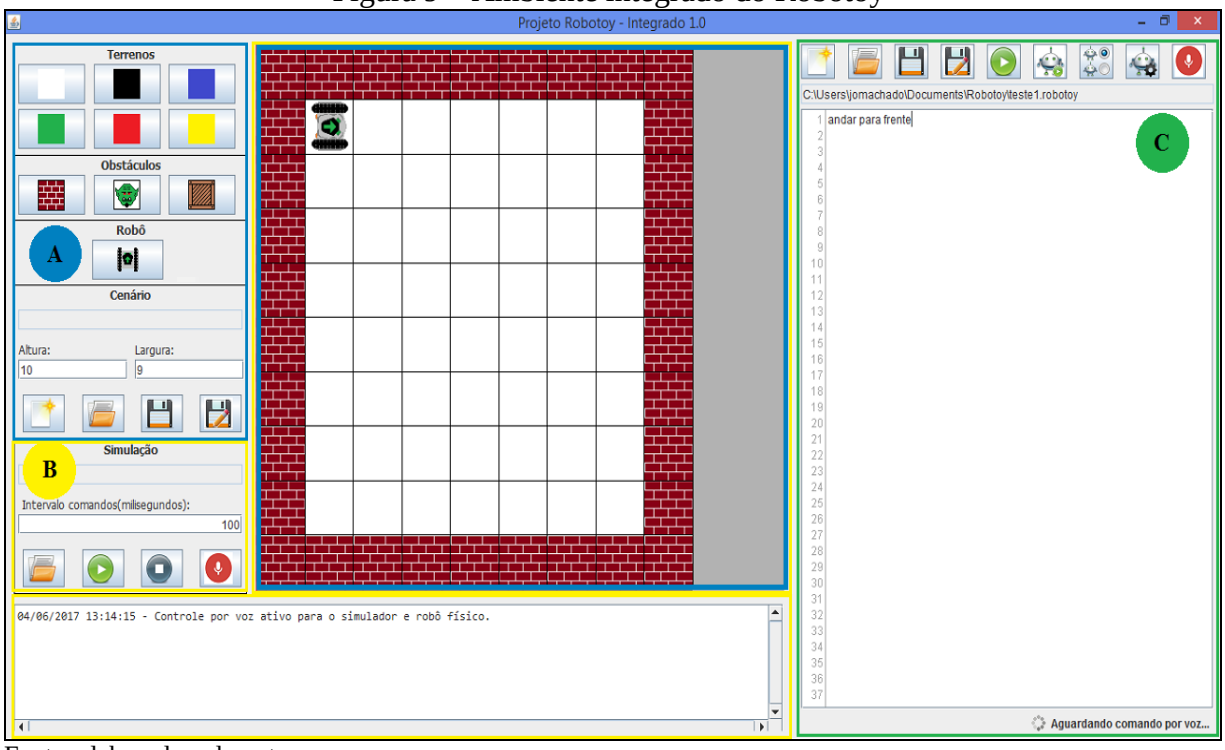
a) possuir um módulo para elaboração e compilação dos programas na linguagem Robotoy (Requisito Funcional - RF);
b) conter um módulo para criação, edição e simulação 2D dos programas elaborados (RF);
c) possibilitar a programação dos robôs através de comando por voz, tanto no editor de programas quanto no simulador (RF);
d) permitir o controle de robôs Lego Mindstorms NXT através do reconhecimento dos comandos por voz (RF);
e) permitir o envio dos programas para as plataformas Lego Mindstorms NXT e Arduino (RF);
f) executar os programas nas plataformas Lego Mindstorms NXT e Arduino (Requisito Não Funcional - RNF);
g) ser implementada na linguagem Java no ambiente de desenvolvimento Eclipse (RNF);
h) utilizar a Microsoft Speech Platform para reconhecimento de comandos por voz (RNF);
i) ter o módulo de reconhecimento por voz implementado na linguagem C# no ambiente de programação Visual Studio (RNF).

3.2 ESPECIFICAÇÃO DA FERRAMENTA

A ferramenta foi desenvolvida para integrar em um único ambiente a programação e a simulação de robôs na linguagem de programação Robotoy. O ponto de partida foi o simulador desenvolvido por Silva (2016), que permitia elaborar programas em Robotoy, construir cenários 2D e simular os programas implementados. Para permitir a programação de robôs Lego Mindstorms NXT e Arduino, foi necessário integrar as funcionalidades da ferramenta implementada por Torrens (2014) juntamente com a extensão proposta por Batista

(2016). Assim, para explicar a ferramenta desenvolvida, torna-se necessário que a mesma seja vista como a composição de três módulos. Conforme pode ser visto na Figura 9, tem-se: o gerenciamento dos cenários (área destacada em azul), o controle de simulação (área destacada em amarelo) e o gerenciamento dos programas (área destaca em verde). No módulo de gerenciamento dos programas é possível criar, editar e salvar programas. Ele também é responsável pela compilação/tradução e envio dos programas tanto para o simulador quanto para os robôs, sendo que a configuração e a seleção dos robôs estão incorporadas ao mesmo. O controle de simulação possibilita a execução dos programas nos cenários. Já o módulo de gerenciamento dos cenários permite criar, editar, redimensionar e salvar os cenários, incluindo os objetos que os compõem, tais como terrenos, obstáculos e robô.

Figura 9 – Ambiente integrado do Robotoy



Fonte: elaborado pelo autor.

As classes que compõem os três módulos estão organizadas em onze projetos, três dos quais foram desenvolvidos ou adaptados nesse trabalho a fim de incluir o recurso de reconhecimento de voz e possibilitar a interação entre os ambientes desenvolvidos por Torrens (2014), Batista (2016) e Silva (2016). O Quadro 5 relaciona o nome e a descrição de cada um dos projetos da ferramenta Robotoy.

Quadro 5 – Descrição dos projetos

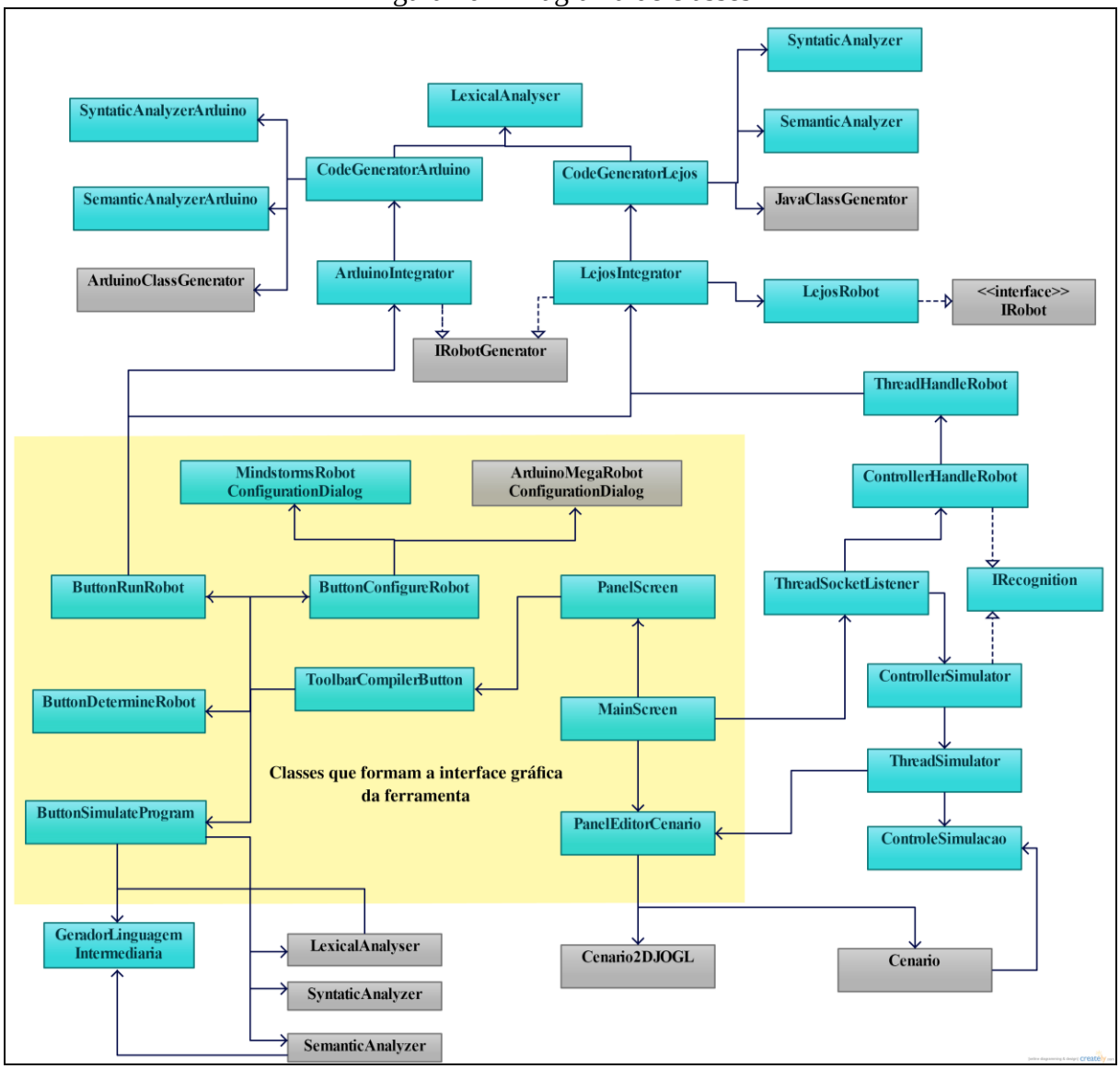
Projeto/Arquivo	Descrição
`robotoy-ui` definido por Torrens (2014)	classes que constituem a interface do módulo de gerenciamento dos programas
`robotoy-compiler` definido por Torrens (2014) e adaptado por Batista (2016)	analisadores léxico, sintático e semântico, responsáveis pela geração dos códigos para robôs Lego NXT e Arduino
`robotoy-compiler-test` definido por Torrens (2014)	testes JUnit para o compilador
`robotoy-lejos-robot` definido por Torrens (2014)	código para robôs Lego NXT correspondente a cada comando da linguagem Robotoy
`arduino-robot-methods.ino` definido por Batista (2016)	arquivo para o Arduino IDE, com as funções correspondentes a cada comando da linguagem Robotoy
`robotoy-generic-robot` definido por Torrens (2014)	interfaces e definições genéricas do que é um robô programável na linguagem Robotoy , utilizadas pelo robô Lego NXT
`robotoy-lib` definido por Torrens (2014) e Batista (2016)	bibliotecas de terceiros utilizadas pelo ambiente de programação e simulação Robotoy
`robotoy-integration` definido por Torrens (2014) e Batista (2016)	definição genérica e implementações de integradores com as plataformas `LejosIntegrator` e `ArduinoIntegrator`
`robotoy-commom` definido por Torrens (2014) e adaptado por Batista (2016)	rotinas comuns entre os projetos
`robotoy-environment`	classes que constituem o módulo de controle da simulação e do reconhecimento por voz, além da tela principal para interação com o usuário
`robotoy-recognition`	rotinas responsáveis pelo recebimento das informações enviadas pela API de reconhecimento de voz
`robotoy-simulator` definido por Silva (2016) e adaptado pelo autor	classes que constituem o módulo de gerenciamento dos cenários

Fonte: elaborado pelo autor.

No diagrama apresentado na Figura 10 pode-se observar as principais classes que compõem o aplicativo, destacando os objetos adicionados ou alterados, representados em azul, que permitiram a inclusão do reconhecimento de voz e a integração das ferramentas. Pode-se observar, na área destacada em amarelo, as classes que constituem a interface gráfica da ferramenta, tendo como classe principal a `MainScreen`, que se relaciona com as classes `PanelScreen`, `PanelEditorCenario` e `ThreadSocketListener`. A classe `PanelScreen` representa a área de gerenciamento dos programas, contendo o *toolbar* com os grupos de botões `ButtonRunRobot`, `ButtonSimluteProgram`, `ButtonDetermineRobot` e `ButtonConfigureRobot e`. Os botões `ButtonDetermineRobot` e `ButtonConfigureRobot` representam, respectivamente, as opções para seleção e configuração dos robôs, instanciando as telas de configuração de cada plataforma. Já os botões `ButtonRunRobot` e

`ButtonSimulateProgram` realizam a compilação e execução dos programas nos robôs e no simulador.

Figura 10 – Diagrama de classes



Fonte: elaborado pelo autor.

Para a integração dos comandos reconhecidos por voz, a classe `ThreadSocketListener` representa o *socket* de rede que, ao receber o comando, dispara um evento para objetos que implementam a interface `IRecognition`, ou seja, os controladores `ControllerSimullator` e `ControllerHandleRobot`. Estes dois controladores contêm uma *thread* instanciada que realiza a execução em paralelo dos comandos recebidos da API de reconhecimento por voz. Por fim, salienta-se que os objetos `ArduinoIntegrator` e `LejosIntegrator` sofreram também alterações. Isso tornou-se necessário, pois, ao iniciar o controle de voz do robô, deve ser iniciada imediatamente uma conexão Bluetooth com o computador para envio e execução de cada comando reconhecido, diferentemente da função

de envio de um programa completo, que aguarda a intervenção do usuário para efetuar a execução do programa no robô. Sendo assim, um novo parâmetro foi acrescentado indicando o tipo de operação: controle do robô por voz ou execução completa de um programa.

3.3 RECONHECIMENTO POR VOZ

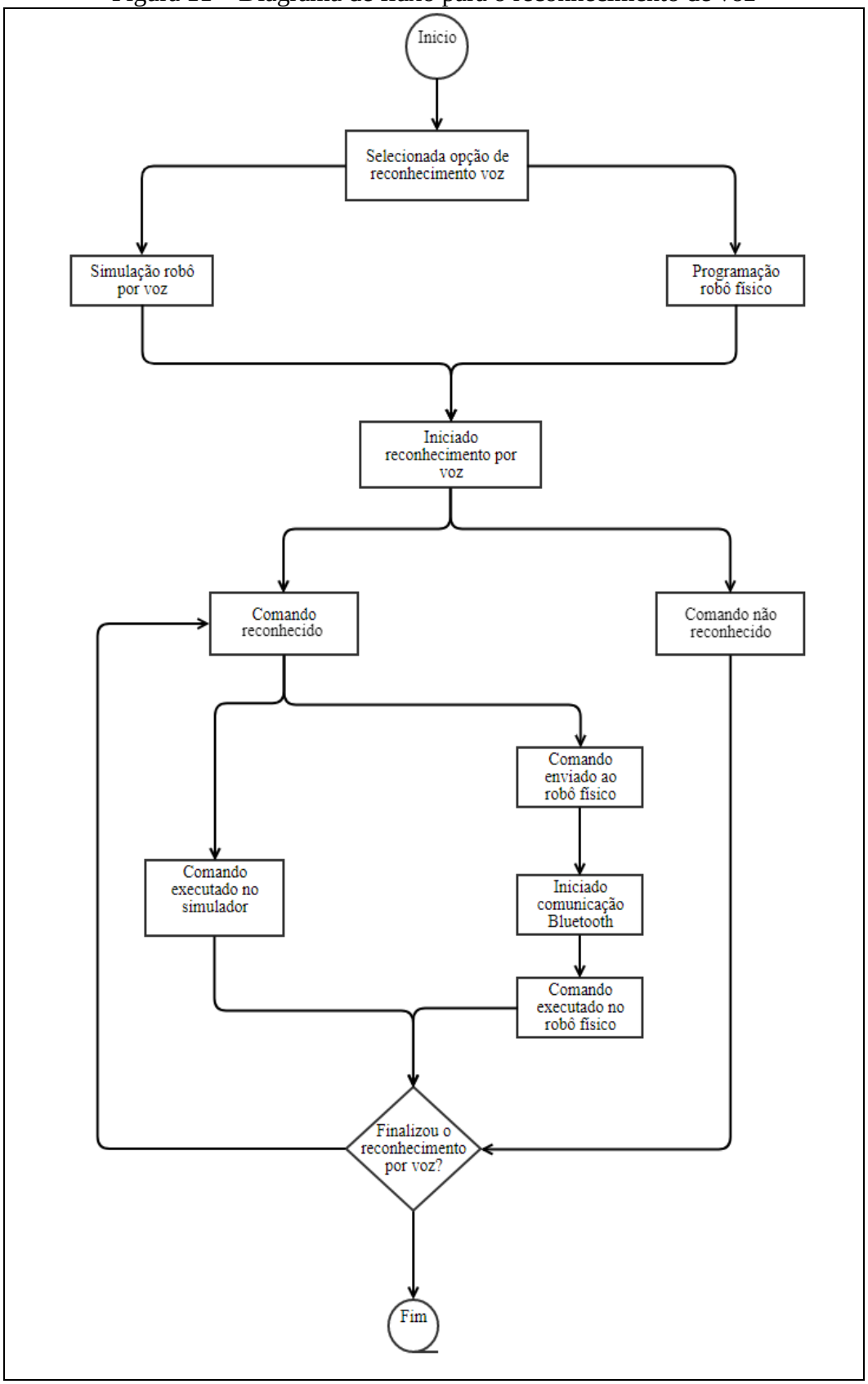
Após integradas as ferramentas de Torrens (2014), Batista (2016) e Silva (2016), foi desenvolvido um aplicativo para possibilitar a simulação e a programação de robôs através de comandos por voz. A Figura 11 ilustra o diagrama do fluxo para o reconhecimento de voz.

Ao escolher a opção de desenvolvimento de programa no Robotoy utilizando o reconhecimento de voz, o usuário pode optar pela execução dos comandos no simulador ou nos robôs físicos Lego Mindstorms NXT. Em seguida, o reconhecimento de voz é habilitado e a API começa initerruptamente a monitorar a entrada de áudio. Assim que um comando é reconhecido, no caso da opção ter sido pela simulação, o comando é compilado e executado imediatamente no cenário simulado. Já no caso da opção ter sido pela execução no robô físico, o comando é enviado para o robô e é iniciada uma comunicação Bluetooth para que seja possível determinar o instante de finalização da execução do comando. É importante ressaltar que, caso um comando esteja sendo executado e um novo comando seja reconhecido, o aplicativo aguarda a finalização da execução do primeiro comando para posteriormente enviar o próximo.

Conforme descrito no Apêndice A, apenas alguns comandos da linguagem Robotoy são reconhecidos por voz, especificamente os comandos de controle de fluxo `se` e `enquanto` e as ações `andar`, `virar` e `emitir som`. São reconhecidos também números entre 1 e 20 e os comandos para verificar a existência ou não de obstáculos e para comparar se uma cor identificada é igual ou diferente de outra cor. As ações `girar roda` e `parar de girar` foram suprimidas, pois não possuem correspondentes no simulador, conforme a especificação feita por Silva (2016). A ação `escrever <texto>` não é reconhecida, pois seria possível especificar somente um número finito de mensagens (correspondente ao `<texto>`) a serem reconhecidas e escritas pelo robô. Ainda, a linguagem Robotoy possui declaração e uso de variáveis, expressões (aritméticas, relacionais e lógicas), declaração e uso de rotinas. Embora seja possível escrever programas em Robotoy com essas estruturas, as mesmas não são reconhecidas por voz em função das suas construções sintáticas. Para fazer a especificação dos comandos a serem reconhecidos por voz, optou-se pela notação XML, já que a Microsoft Speech Platform não possibilita a especificação da gramática utilizando notação Backus Naur Form (BNF). No Quadro 6 é apresentado um trecho da gramática da linguagem Robotoy

adaptada, mostrada no Apêndice B, e o trecho correspondente da gramática utilizada no reconhecimento de voz.

Figura 11 – Diagrama de fluxo para o reconhecimento de voz



Fonte: elaborado pelo autor.

Quadro 6 – Gramática da linguagem x gramática do reconhecimento de voz

Gramática da linguagem Robotoy

```
...
<comando>           ::= <controle_fluxo> | ...| <acao>
<controle_fluxo>    ::= <se> | <enquanto>
<se>                ::= se <condicao> \n <lista_comandos> <senao> fim do se
<senao>             ::= ε | senao \n <lista_comandos>
<enquanto>          ::= enquanto <condicao> \n <lista_comandos> fim do enquanto
<condicao>          ::= ...
<acao>              ::= <andar> | ... | <emitir_som>
<andar>             ::= andar para <frente_ou_tras> <qdade_opcional>
<emitir_som>        ::= emitir som
<frente_ou_tras>    ::= frente | tras
<qdade_opcional>    ::= ε | <expressao>
<expressao>         ::= ...
```

Gramática do reconhecimento de voz

```
<grammar ...>
 <rule id="main">
   <one-of>
     <item> <ruleref uri="#controle_fluxo" /> </item>
     <item> <ruleref uri="#acao"/></item>
   </one-of>
 </rule>
 <rule id="controle_fluxo">
   <one-of>
     <item> se <ruleref uri="#condicao"/> </item>
     <item> senão </item>
     <item> fim do se </item>
     <item> enquanto <ruleref uri="#condicao"/> </item>
     <item> fim do enquanto </item>
   </one-of>
 </rule>
 ...
 <rule id="acao">
   <one-of>
     <item> andar para
            <item> <ruleref uri="#frente_tras" /> </item>
            <item repeat="0-1"> <ruleref uri="#qdade_opcional" /> </item>
     </item>
     ...
     <item> emitir som </item>
   </one-of>
 </rule>
 <rule id="frente_tras">
   <one-of>
     <item> frente </item>
     <item> trás </item>
   </one-of>
 </rule>
 ...
 <rule id="qdade_opcional">
   <one-of>
     <item> 1 </item>
     <item> 2 </item>
     <item> 3 </item>
     ...
   </one-of>
 </rule>
</grammar>
```

Fonte: elaborado pelo autor.

Conforme pode ser observado no Quadro 6, foi definida a regra principal (`main`) que agrupa dois grupos de comandos `controle_fluxo` e `acao`, representando respectivamente, os

comandos de seleção e repetição e as ações a serem executadas pelo robô. O nó `<one-of>` indica que a sentença só pode conter um `<item>` da lista, ou seja, a sentença será um comando de controle de fluxo ou ação. Observa-se que o atributo `<item repeat>` tornou-se necessário para determinar a quantidade de vezes que a regra `<ruleref>` constará na sentença, permitindo assim definir construções sintáticas opcionais.

Para a implementação do reconhecimento de voz, utilizou-se a Microsoft Speech Platform. Diante disso, desenvolveu-se um aplicativo em C# que realiza a troca de dados através de canal de comunicação em rede utilizando o protocolo User Datagram Protocol (UDP). O Quadro 7 apresenta as rotinas do aplicativo responsáveis pela inicialização do motor de reconhecimento de voz e o carregamento da gramática, respectivamente representados pelos objetos das classes `SpeechRecognitionEngine` e `Grammar` (linhas 1 a 8). Após a inicialização dos objetos, é necessário determinar a forma de entrada de áudio (linha 9), podendo ser através do dispositivo padrão do áudio do sistema (usado no aplicativo) ou pela especificação de arquivos no formato `.wav`.

Quadro 7 – Código de inicialização da Microsoft Speech Platform

```
1  private SpeechRecognitionEngine recEngine;
2  const string _IDIOMA = "pt-Br";
3  const string _FILE_GRAMATICA = @"Gramatica\grammar.xml";
4  string pathGramatica = string.Concat(AssemblyDirectory.TrimEnd().AddBs(),
5  _FILE_GRAMATICA);
6  ...
7  Grammar gramar = new Grammar(pathGramatica);
8  recEngine.LoadGrammarAsync(gramar);
9  recEngine.SetInputToDefaultAudioDevice();
```

Fonte: elaborado pelo autor.

Deve-se também realizar a seleção dos comandos reconhecidos que serão processados posteriormente pela ferramenta Robotoy. Ao identificar uma sentença válida, o aplicativo executa o código apresentado no Quadro 8, onde somente serão armazenadas para processamento as sentenças que possuem um grau de confiança igual ou superior a 70% (linha 4). Em seguida, os comandos previamente selecionados serão adicionados a uma lista para posterior envio ao Robotoy.

Quadro 8 – Reconhecimento dos comandos

```
1  string mensagemLog = String.Format("Grau de confiação: {0} - Sentença
2  Reconhecida: {1}", e.Result.Confidence, e.Result.Text);
3  log.Add(mensagemLog);
4  if (e.Result.Confidence >= 0.70) {
5    string command = AjustaComando(e.Result.Text);
6    robotoyIntegration.AddComando(command);
7  }
```

Fonte: elaborado pelo autor.

Como último fluxo de execução, implementou-se uma *thread* que realiza a leitura dos comandos reconhecidos para posterior envio à aplicação principal, conforme mostra o Quadro 9. Neste contexto, a rotina contém um laço de repetição que a cada 200 milissegundos

conecta-se ao *socket* disponível na ferramenta Robotoy. Os comandos são enviados em formato UTF-8 (8-bit Unicode Transformation Format) e, caso ocorra algum erro no envio, é efetuada outra tentativa imediatamente em seguida.

Quadro 9 – *Thread* de comunicação com o Java

```
1   if (HasComando()) {
2     log.Add("Enviando comando reconhecido ao Robotoy");
3     if (Connecting())   {
4       string sendMessage = RemoverComando();
5       if (!SendSocket(sendMessage))      {
6         AddComando(sendMessage, true);
7         return;
8       }
9     }
10    else {
11       return;
12    }
13  Thread.Sleep(_TIME_CHECK_COMANDO)
```

Fonte: elaborado pelo autor.

3.4 IMPLEMENTAÇÃO

Nesta seção são mostradas as técnicas e as ferramentas utilizadas e a operacionalidade da implementação.

3.4.1 Técnicas e ferramentas utilizadas

As técnicas e ferramentas utilizadas no desenvolvimento do trabalho foram:

a) no desenvolvimento da aplicação Robotoy utilizou-se a linguagem de programação Java através da IDE Eclipse;
b) na implementação do ambiente gráfico optou-se pela biblioteca gráfica `Swing`;
c) para o desenvolvimento e envio dos códigos referentes aos robôs Lego Mindstorms NXT e Arduino, fez-se necessária a instalação do *plugin* leJOS e da IDE Arduino;
d) na implementação do reconhecimento de voz foi utilizada a linguagem de programação C# através do ambiente de programação Visual Studio, usando a Microsoft Speech Platform.

3.4.2 Operacionalidade da implementação

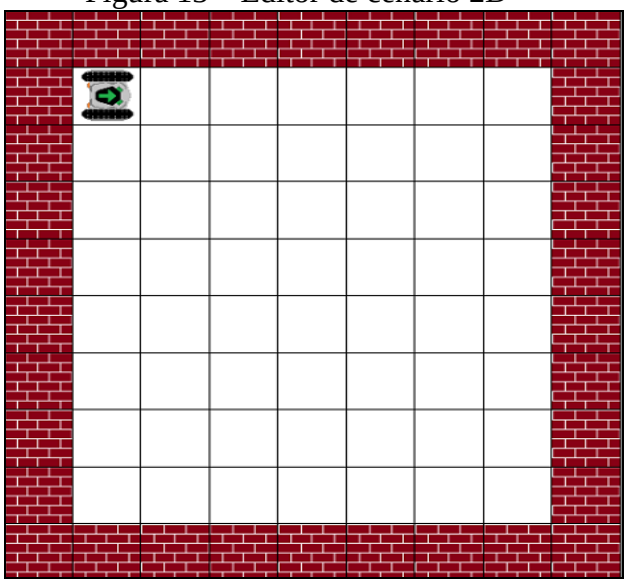
Conforme apresentado na Figura 9 (da seção 3.2), a ferramenta é composta por três módulos. Sendo assim, é necessário que seja explanado cada um destes módulos separadamente. Primeiramente é descrito o módulo para gerenciamento dos cenários, cuja interface possui uma barra de ferramentas e um editor de cenário, ambos desenvolvidos por Silva (2016). A barra de ferramentas mostrada na Figura 12 tem como objetivo permitir ao usuário o gerenciamento dos objetos a serem dispostos no cenário 2D, bem como criar um

cenário novo (D), abrir um cenário já existente (E), salvar ((F) e (G)) ou redimensionar o cenário que está sendo editado, isto é, alterar a quantidade de células do cenário (campos `altura` e `largura` (C)). Para acrescentar objetos, deve-se selecionar um objeto entre `Terrenos`, `Obstáculos` e `Robô` na barra de ferramentas (A) e clicar na posição desejada no editor de cenários (Figura 13). A pasta e o nome do arquivo de um cenário previamente desenvolvido e carregado no editor são apresentados no campo `Cenário` (B).

Figura 12 – Barra de ferramentas para edição de cenários

Fonte: elaborado pelo autor.

Figura 13 – Editor de cenário 2D



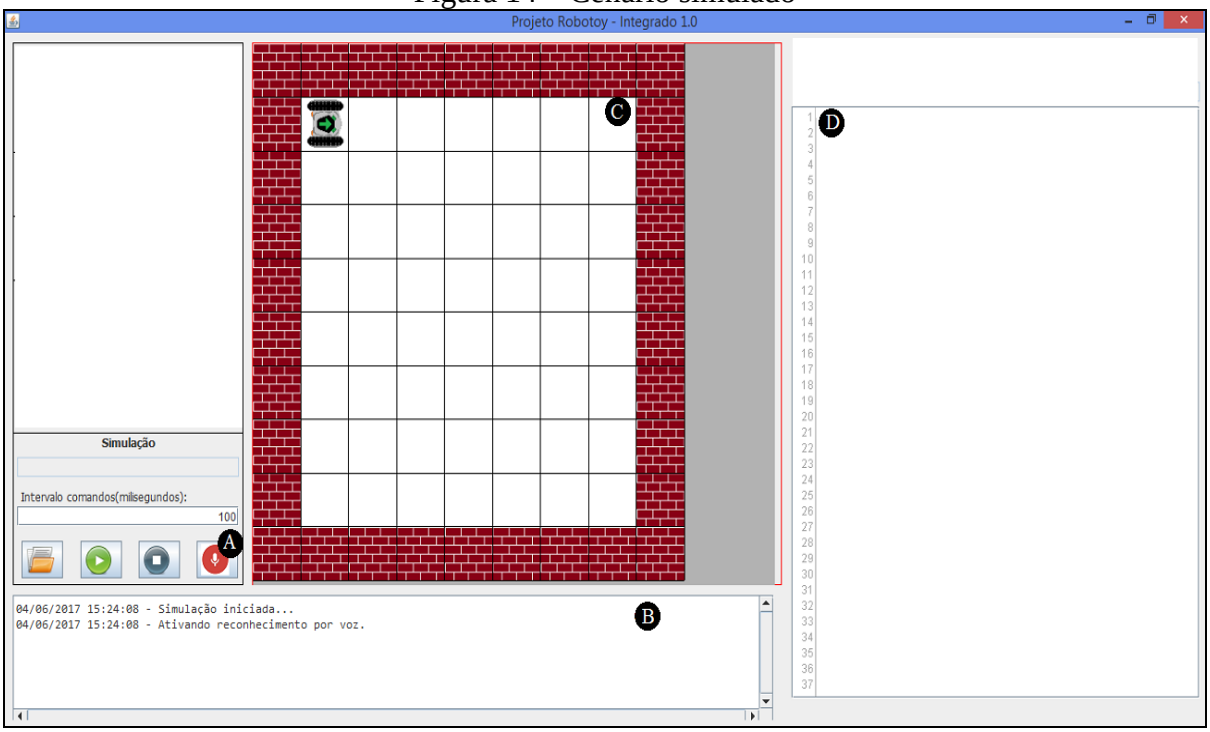
Fonte: elaborado pelo autor.

Na Figura 14 é apresentada a interface do módulo de controle de simulação, inicialmente implementada por Silva (2016), que teve o recurso de controle por voz adicionado às suas funcionalidades. Ao selecionar o botão de `Reconhecer comando por voz` (A), são apresentadas no display (B) as mensagens "Simulação iniciada..." e "Ativando reconhecimento por voz.", indicando que o aplicativo está pronto para reconhecer os comandos ditos pelo usuário. Cada comando reconhecido é executado imediatamente no cenário simulado (C) e mostrado no editor de programas (D). Caso não seja possível executar um comando, serão apresentadas mensagens no display. Por exemplo, caso o robô identifique um obstáculo à sua frente, a mensagem "Existe um obstáculo impedindo o robô de andar." será apresentada.

O módulo de gerenciamento dos programas tem a interface mostrada na Figura 15, contendo uma área para edição de programas (J) e nove botões com as seguintes funcionalidades: (A) botão `Novo`, permite criar um programa novo; (B) botão `Abrir`, possibilita a seleção de programas com a extensão `.robotoy`, exibindo o programa selecionado na área de edição (J); botões (C) `Salvar` e (D) `Salvar como`, salvam os comandos do editor de programas em arquivos `.robotoy` na pasta selecionada pelo usuário;

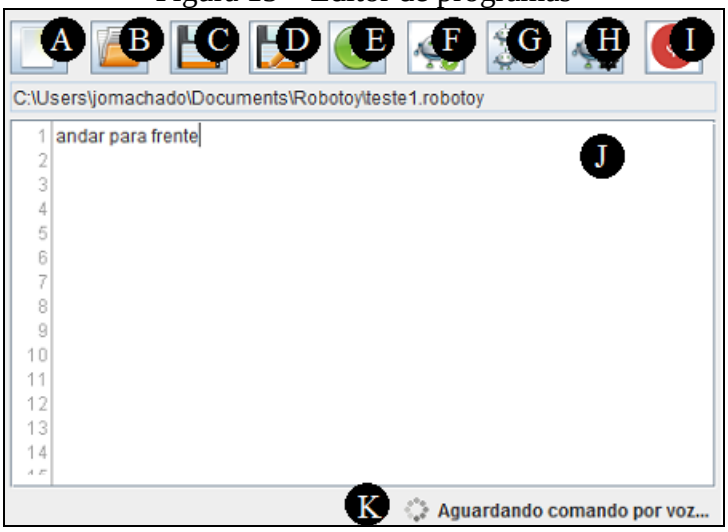
(E) botão `Simular programa`, compila os comandos para iniciar a simulação do programa no cenário 2D; (F) botão `Enviar programa`, compila o programa para a plataforma de robô selecionada e envia-o ao robô físico; (G) botão `Selecionar robô`, permite a seleção do tipo de plataforma de robô físico; (H) botão `Configurar robô`, possibilita a configuração da plataforma de robô selecionada; (I) botão `Controlar robô por voz`, inicia o reconhecimento de voz e, para cada comando reconhecido, executa a ação no robô Lego Mindstorms NXT, adicionando o comando na área de edição de programas (J).

Figura 14 – Cenário simulado



Fonte: elaborado pelo autor.

Figura 15 – Editor de programas



Fonte: elaborada pelo autor.

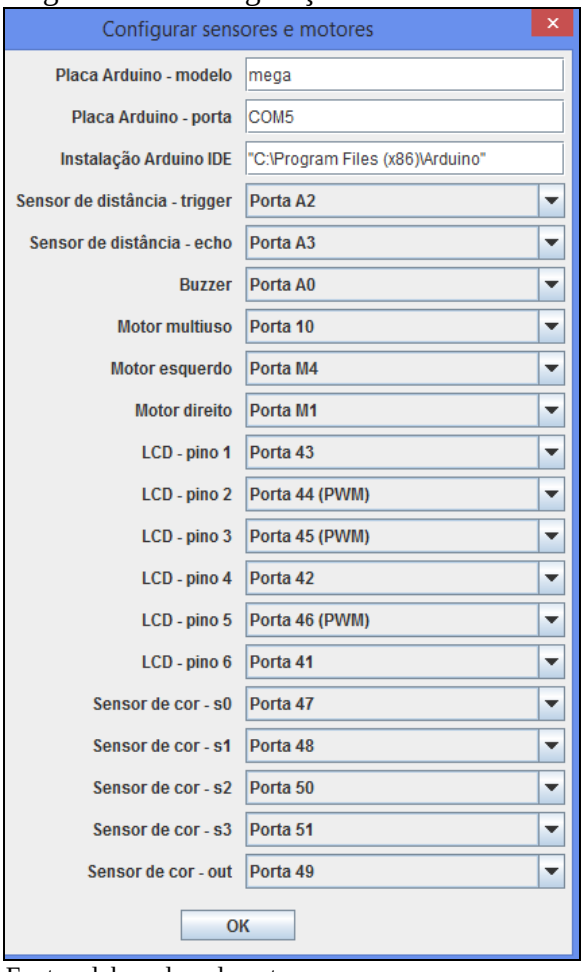
Ao selecionar o controle de robô por voz, na barra de status de comunicação (Figura 15 (K)) são exibidas mensagens para que seja possível determinar o estado atual da comunicação entre o ambiente de programação Robotoy e o robô. Quanto um comando é reconhecido, o mesmo é compilado e enviado ao robô através de Bluetooth ou cabo USB. No momento em que se inicia a execução do programa no robô Lego Mindstorms NXT, um canal de comunicação via Bluetooth é aberto com o computador para que, ao término da execução de um comando, seja possível identificar que o robô está disponível para executar um próximo comando. Nesse caso, na barra de status de comunicação é apresentada a mensagem "Aguardando comando por voz...".

Conforme pode ser observado na Figura 16 e na Figura 17, as opções para configuração dos robôs permaneceram as mesmas desenvolvidas por Torrens (2014) e Batista (2016). Contudo, a configuração *Device Address* foi incluída na configuração dos robôs Lego Mindstorms NXT para que seja possível realizar a conexão Bluetooth com o robô, possibilitando assim o controle do robô por voz.

Figura 16 – Configuração do robô Lego Mindstorms NXT

Fonte: elaborado pelo autor.

Figura 17 – Configuração do robô Arduino



Fonte: elaborado pelo autor.

3.5 ANÁLISE DOS RESULTADOS

Nesta seção são mostrados os resultados obtidos com o desenvolvimento da aplicação. Inicialmente é apresentada uma análise entre o grau de confiança no reconhecimento de voz detectado pela *engine* Julius e pela API Microsoft Speech Platform. Em seguida, tem-se um comparativo do tempo necessário para o envio dos programas às plataformas Lego Mindstorms NXT e Arduino. A última subseção traz uma comparação entre os trabalhos correlatos e o desenvolvido.

3.5.1 Comparativo entre as bibliotecas de reconhecimento de voz

A implementação da funcionalidade de reconhecimento de voz demandou algumas alterações no decorrer do desenvolvimento do trabalho para que se aprimorasse o grau confiabilidade na identificação das sentenças. Inicialmente, o reconhecimento de voz foi implementado usando a *engine* Julius. No entanto, esta apresentou problemas ao ser exposta em ambientes com ruído, pois sentenças diferentes são reconhecidas quando a entrada de áudio capta qualquer som não esperado. Para contornar o problema, o grau de confiança no reconhecimento de voz foi aumentado. Mas alguns comandos ou palavras corretamente pronunciadas não eram reconhecidas. Assim, num segundo momento, utilizou-se a API Microsoft Speech Platform. Essa demonstrou uma performance melhor em ambientes com ruído, além de manter um nível de confiabilidade idêntico.

A Tabela 1 e a Tabela 2 mostram os resultados obtidos em um experimento realizado com três interlocutores em um ambiente controlado sem ruído, utilizando o recurso de reconhecimento de voz das duas bibliotecas. Composto por dois homens com idades de 32 e 59 anos e uma mulher com 54 anos, o experimento validou o grau de confiança na pronúncia dos comandos. Para a *engine* Julius, o grau de confiança de 90% resultou em um nível eficaz para o filtro, já que supera a média de todos os comandos captados. Já para API Microsoft Speech Platform, optou-se por utilizar o grau de confiança de 70%, valor eficaz tanto em ambiente com ou sem a existência de ruído, além de comprovar os resultados atingidos pelo o experimento.

Portanto, a ausência do recurso de remoção total de ruído da *engine* Julius tornou-se o maior impedimento para a escolha dessa biblioteca de reconhecimento de voz. Além disso, a API Microsoft Speech Platform demonstrou um bom grau de eficiência na utilização do microfone acoplado aos notebooks, dispensando a compra de um dispositivo externo de entrada de áudio.

Tabela 1 – Análise do reconhecimento de voz pela *engine* Julius

Comando	H32	H59	M54	média	desvio padrão
andar para frente	0,87	0,96	0,93	0,92	0,05
andar para trás	0,95	0,95	0,93	0,94	0,01
andar para frente 3	0,87	0,99	0,8	0,89	0,10
andar para trás 4	0,99	0,99	0,94	0,97	0,03
andar para frente 13	0,99	0,93	0,87	0,93	0,06
andar para trás 14	0,99	0,89	0,5	0,79	0,26
virar para a direita	0,94	0,95	0,92	0,94	0,02
virar para a esquerda	0,95	0,84	0,96	0,92	0,07
virar motor multiuso para a esquerda	0,95	0,95	0,95	0,95	0,00
emitir som	1,00	1,00	1,00	1,00	0,00
enquanto não tem obstáculo	0,99	1,00	0,99	0,99	0,01
se cor identificada igual branco	0,98	0,99	0,80	0,92	0,11
senão	0,99	0,94	0,99	0,97	0,03
fim do se	0,99	0,99	0,91	0,96	0,05
fim do enquanto	1,00	0,99	1,00	1,00	0,01
Total da média/desvio padrão				0,94	0,07

Fonte: elaborado pelo autor.

Tabela 2 – Análise de reconhecimento de voz pela Microsoft Speech Platform

Comando	H32	H59	M54	média	desvio padrão
andar para frente	0,70	0,87	0,79	0,79	0,09
andar para trás	0,81	0,86	0,80	0,82	0,03
andar para frente 3	0,71	0,87	0,76	0,78	0,08
andar para trás 4	0,82	0,85	0,74	0,80	0,06
andar para frente 13	0,71	0,82	0,88	0,80	0,09
andar para trás 14	0,82	0,81	0,75	0,79	0,04
virar para a direita	0,71	0,83	0,89	0,81	0,09
virar para a esquerda	0,78	0,76	0,85	0,80	0,05
virar motor multiuso para a esquerda	0,72	0,83	0,83	0,79	0,06
emitir som	0,81	0,90	0,86	0,86	0,05
enquanto não tem obstáculo	0,81	0,84	0,84	0,83	0,02
se cor identificada igual branco	0,82	0,77	0,50	0,70	0,17
senão	0,85	0,90	0,77	0,84	0,07
fim do se	0,73	0,82	0,80	0,78	0,05
fim do enquanto	0,89	0,88	0,62	0,80	0,15
Total da média/desvio padrão				0,80	0,04

Fonte: elaborado pelo autor.

3.5.2 Comparativo do tempo de envio de programas

A Tabela 3 apresenta um comparativo do tempo necessário para o envio dos programas para as plataformas dos robôs através de seus métodos de conexão. Para a elaboração do comparativo, foram realizados cinco testes com o envio do comando `andar para frente` para um robô Lego Mindstorms NXT e para um robô Arduino, apresentados no

Apêndice C. O tempo registrado contempla tanto a compilação do programa no ambiente Robotoy como o envio do programa compilado aos robôs.

Tabela 3 – Comparativo de tempo para o envio dos programas aos robôs (em segundos)

teste	Arduino (USB)	Lego NXT (USB)	Lego NXT (Bluetooth)	Lego NXT controle por voz (Bluetooth)
1	30,88	8,95	18,98	24,26
2	29,35	9,31	17,23	24,35
3	30,58	8,98	16,28	22,9
4	29,55	8,38	16,85	24,5
5	29,98	9,03	17,4	23,45
total média	30,07	8,93	17,35	23,89

Fonte: elaborado pelo autor.

Confrontando as plataformas Arduino e Lego NXT por meio do envio do programa pela USB, são necessários em média apenas 8,93 segundos para o Lego NXT. Além disso, o kit Lego NXT permite o envio dos programas através de Bluetooth, tornando-se prático o processo de manipulação do robô, porém lento em comparação à conexão USB, mas com tempo menor, se comparado ao envio de programas para o Arduino via USB.

Por fim, compara-se o recurso adicional de controle por voz dos robôs Lego NXT. Em comparação aos métodos citados anteriormente, o recurso de controle por voz realiza duas conexões Bluetooth com o robô, primeiramente para o envio do comando compilado e, em seguida, para que seja possível determinar a finalização da execução do mesmo. O tempo necessário para finalização deste processo torna-se maior em relação aos demais. Além disso, vale a pena mencionar a dificuldade em sincronizar estes dois processos. Após a finalização do *batch* que realiza a compilação e envio dos programas, é necessário aguardar alguns segundos para que o kit Lego NXT comece a execução do programa. Mas não foi possível mensurar este tempo. Sendo assim, várias tentativas de comunicação via Bluetooth são efetuadas pelo computador para que seja possível sincronizar os processos.

3.5.3 Comparativo entre os trabalhos correlatos

O Quadro 10 apresenta de forma comparativa as características dos trabalhos correlatos e da ferramenta desenvolvida. Pode-se observar que os softwares Webots, Robotoy não integrado (TORRENS, 2014; BATISTA, 2016; SILVA, 2016), Robotoy (trabalho desenvolvido) e RoboEduc permitem a customização de robôs, ou seja, o usuário pode modificar a estrutura dos robôs adaptando a cada cenário proposto, enquanto o RoboMind FURB utiliza modelos de robôs pré-definidos, limitando assim o número de problemas e

cenários onde os robôs possam ser inseridos. Todas as ferramentas dispõem de um ambiente para simulação com o intuito de tornar desnecessária a aquisição de um kit de robótica, porém somente o ambiente de simulação do Webots permite a inclusão de múltiplos robôs, proporcionando a interação entre os robôs na busca da solução a partir da programação separada de cada um.

Quadro 10 – Comparativo entre os trabalhos correlatos e a ferramenta desenvolvida

trabalhos / características	RoboEduc (SILVA, 2009)	Webots (CYBERBOTICS, 2016)	RoboMind FURB (BENITTI et al. 2008)	Robotoy (TORRENS, 2014; BATISTA, 2016; SILVA, 2016)	Robotoy (trabalho desenvolvido)
customização de robôs	X	X		X	X
múltiplos robôs		X			
plataforma de robô suportada	Lego NXT, Lego RCX, H-EDUC	Khepera, Hemisson, Lego NXT, entre outros	Lego NXT	Lego NXT, Arduino	Lego NXT, Arduino
linguagem de programação própria	X		X	X	X
tipo de programação	textual, gráfica	textual	textual	textual	textual
ambiente de simulação	X	X	X	X	X
reconhecimento de comandos por voz					X

Fonte: elaborado pelo autor.

Referente à linguagem de programação, exceto pelo Webots, os demais contam com uma linguagem própria, motivando assim os usuários que não possuem conhecimento prévio em linguagens de programação comerciais o estudo da lógica de programação. Contudo, esses ambientes que possuem uma linguagem de programação própria contêm um conjunto menor de instruções em comparação ao Webots. Ainda no quesito linguagem de programação, apenas no RoboEduc tem-se a possibilidade de programação com o uso de componentes gráficos, incluindo um nível inicial intuitivo aos alunos no princípio do desenvolvimento da lógica de programação.

Por fim, todos incluem suporte à programação de kit Lego Mindstorms NXT, sendo que RoboEduc e Robotoy possibilitam o envio de programas a kits de baixo custo, como H-EDUC e Arduino, respectivamente. Essa alternativa permite que as instituições de ensino possam determinar o kit adequado a sua realidade financeira, além de abranger um maior

público a partir dos kits de baixo custo. No entanto, nenhum deles permite programar os robôs com comando por voz, funcionalidade desenvolvida nesse trabalho como extensão do Robotoy.

Foram apresentadas duas categorias de ambientes para desenvolvimento e programação de robôs. Os softwares RoboEduc, RoboMind FURB e Robotoy possuem a característica comum de serem ambientes educativos, no qual os alunos efetuam exercícios para o desenvolvimento do processo de ensino-aprendizagem. Por outro lado, o Webots é uma aplicação comercial para desenvolvimento de robôs comerciais, sendo necessário um ambiente fisicamente realista para modelagem, programação e simulação dos robôs.

Todavia, o ambiente educativo Robotoy passou por adaptações, porém tais transformações foram implementadas separadamente. A partir disso, desenvolveu-se a integração em um único ambiente para programação e simulação de robôs Arduinos e Lego Mindstorms NXT. Desta forma, tem-se a intenção de proporcionar uma maior inserção do Robotoy nas instituições de ensino. Além disso, incluiu-se o recurso de entrada de comando por voz, adicionando outra forma de interação e aumentando a usabilidade.

4 CONCLUSÕES

Sabe-se que no Brasil a educação enfrenta grandes desafios. Nessa perspectiva pedagógica, a robótica educacional, como mediadora da relação entre o professor e o aluno, auxilia no desenvolvimento da aprendizagem, por meio do compartilhamento de novas ideias e experiências. Neste sentido, a linguagem Robotoy consiste em um instrumento facilitador para o aluno na construção do seu conhecimento tecnológico.

Diante disso, esse trabalho apresentou a integração em uma única ferramenta dos trabalhos desenvolvidos por Torrens (2014), Batista (2016) e Silva (2016), adicionando o recurso de entrada de comando por voz. Para tanto, optou-se por utilizar a Microsoft Speech Platform para a implementação do reconhecimento de voz, que demonstrou resultados melhores em ambientes com ruído em comparação a *engine* Julius. Na elaboração da gramática para o reconhecimento de voz, a especificação permitiu a estruturação das sentenças de forma similar à gramática elaborada por Torrens (2014). No entanto, apenas um subconjunto de comandos Robotoy são reconhecidos por voz. Foram selecionados apenas os comandos que possibilitam tanto o controle do ambiente de simulação como a programação física dos robôs similar a forma como as pessoas falam. Salienta-se que o comparativo entre os métodos de envio dos programas ao robô físico identificou lentidão no recurso de controle por voz devido à dificuldade na sincronização dos processos e o tempo necessário para comunicação Bluetooth com o robô Lego NXT.

A partir dos resultados obtidos, pode-se concluir que o trabalho cumpre os objetivos propostos, pois fornece um ambiente para a elaboração de programas na linguagem Robotoy tanto para o ambiente simulado como para os robôs Lego Mindstorms NXT e Arduino. Acrescenta também a possibilidade de construir programas através de comando por voz, executando-os simultaneamente no simulador como no robô físico. Embora tenha sido constatada lentidão no processo de controle por voz dos robôs, a integração da ferramenta e a inclusão do reconhecimento por voz, como esperado, fornece usabilidade e estimula uma nova forma de interação.

4.1 EXTENSÕES

Como sugestão de extensão para trabalhos futuros propõe-se:

a) adicionar ao kit Arduino um módulo Bluetooth para que seja possível o envio de comandos por voz aos robôs;
b) permitir a interação de múltiplos robôs no ambiente de simulação 2D, opção semelhante à disponibilizada no Webots;

c) criar um módulo para elaboração de cenários de simulação 3D;
d) desenvolver uma linguagem gráfica, similar à do Roboeduc, para que o usuário possa realizar a programação dos robôs por meio do recurso de arrastar e soltar;
e) permitir a inclusão de códigos nativos para o Lego Mindstorms NXT e Arduino em programas escritos na linguagem Robotoy;
f) melhorar a performance do controle por voz dos robôs Lego Mindstorms NXT.

REFERÊNCIAS

AZEVEDO, Samuel; AGLAÉ, Akynara; PITTA, Renata. Minicurso: introdução a robótica educacional. In: REUNIÃO ANUAL DA SOCIEDADE BRASILEIRA PARA O PROGRESSO DA CIÊNCIA, 62., 2010, Natal. **Anais eletrônicos...** São Paulo: SBPC/UFRN, 2010. Não paginado. Disponível em: <http://www.sbpcnet.org.br/livro/62ra/minicursos/MC%20Samuel%20Azevedo.pdf>. Acesso em: 07 set. 2016.

BARROS, Renata P. **Evolução, avaliação e validação do software RoboEduc**. 2011. 92 f. Dissertação (Mestrado em Engenharia da Computação) – Programa de Pós-Graduação em Engenharia Elétrica e de Computação, Universidade Federal do Rio Grande do Norte, Natal. Disponível em: <http://repositorio.ufrn.br:8080/jspui/handle/123456789/15375>. Acesso em: 07 set. 2016.

BATISTA, Juliana C. **Robotoy:** ferramenta para ensino de programação para crianças usando robô Arduino. 2016. 64 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

BENITTI , Fabiane B. V. et al. **Bem vindo ao RoboLAB**: robótica educativa na FURB. [Blumenau?], [2008]. Disponível em: <http://robolab.inf.furb.br/>. Acesso em: 21 ago. 2016.

BRESOLIN, Adriano A. **Estudo do reconhecimento de voz para o acionamento de equipamentos elétricos via comandos em português**. 2003. 107 f. Dissertação (Mestrado em Automação Industrial) – Programa de Pós-Graduação em Automação Industrial, Universidade do Estado de Santa Catarina, Joinville. Disponível em: <http://www.tede.udesc.br/tde_arquivos/8/TDE-2006-05-19T093420Z-193/Publico/Adriano%20de%20Andrade%20Bresolin.pdf>. Acesso em: 02 nov. 2016.

CASTILHO, Maria I. **Robótica na educação:** com que objetivos? [Porto Alegre], [2009]. Disponível em: <http://www.pucrs.br/eventos/desafio/2007/mariaines.php#raclog>. Acesso em: 15 set. 2016.

CYBERBOTICS. **Webots robot simulator** [S.l.], 2016. Disponível em: <https://www.cyberbotics.com>. Acesso em: 23 ago. 2016

FORNAZA, Roseli; WEBBER, Carine G. Robótica educacional aplicada à aprendizagem em física. **Renote**, [Porto Alegre], v. 12, n. 1, p. 1-10, jul. 2014. Disponível em: <http://seer.ufrgs.br/index.php/renote/article/view/50275/31405>. Acesso em: 02 nov. 2016.

GOMES, Cristiane G. et al. A robótica como facilitadora do processo de ensino-aprendizagem de matemática no ensino fundamental. In: PIROLA, Nelson A. (Org). **Ensino de ciências e matemática IV**: temas de investigação. São Paulo: UNESP/Cultura Acadêmica, 2010. p. 205-221. Disponível em: <http://books.scielo.org/id/bpkng/pdf/pirola-9788579830815-11.pdf>. Acesso em: 28 ago. 2015.

KLAUTAU, Aldebaro et al. **Reconhecimento de voz para o português brasileiro**. [S.l.], 2012. Disponível em: <http://www.laps.ufpa.br/falabrasil/downloads.php>. Acesso em: 21 out. 2016.

LEE, Akinobu; KAWAHARA, Tatsuya. Recent development of open-source speech recognition engine Julius. In: ASIA-PACIFIC SIGNAL AND INFORMATION PROCESSING ASSOCIATION ANNUAL SUMMIT AND CONFERENCES, 1., 2009, Hokkaido. **Proceedings...** Hokkaido: APSIPA, 2009. p. 131-137. Disponível em: <https://julius.osdn.jp/paper/julius-APSIPA2009.pdf>. Acesso em: 15 set. 2016.

LEE, Akinobu. **The Julius book**. [S.l.], 2009. Disponível em: <http://osdn.dl.sourceforge.jp/julius/37581/Juliusbook-part-4.1.2-en.pdf>. Acesso em: 22 out. 2016.

LEGO GROUP. **Lego Mindstorms NXT 2.0**. [S.l.], 2016. Disponível em: <https://shop.lego.com/en-US/LEGO-MINDSTORMS-NXT-2-0-8547>. Acesso em: 02 nov. 2016.

MAISONNETTE, Rogers. **A utilização dos recursos informatizados a partir de uma relação inventiva com a máquina**: a robótica educativa. [Paraná], [2002]. Disponível em: <http://www.proinfo.gov.br/upload/biblioteca.cgd/192.pdf>. Acesso em: 02 nov. 2016.

MARTINS, José A. **Avaliação de diferentes técnicas para reconhecimento de fala**. 1997. 161 f. Tese (Doutorado em Engenharia Elétrica e de Computação) – Faculdade de Engenharia Elétrica e de Computação, Universidade Estadual de Campinas, Campinas. Disponível em: <http://www.bibliotecadigital.unicamp.br/document/?code=vtls000127015>. Acesso em: 15 set. 2016.

MCROBERTS, Michael. **Arduino básico**. São Paulo: Novatec, 2011.

MICROSOFT CORPORATION**. Microsoft speech platform**. [S.l.], 2016a. Disponível em: <https://msdn.microsoft.com/en-us/library/jj127857.aspx>. Acesso em: 21 mar. 2017.

_____. Microsoft speech platform. [S.l.], 2016b. Disponível em: <https://msdn.microsoft.com/en-us/library/jj127916.aspx>. Acesso em: 21 mar. 2017.

MIRANDA, Leonardo C.; SAMPAIO, Fábio F.; BORGES, José A. S. RoboFácil: especificação e implementação de um kit de robótica para a realidade educacional brasileira. **Revista Brasileira de Informática na Educação**, [Porto Alegre?], v. 18, n. 3, p. 46-58, 2010. Disponível em: <http://www.br-ie.org/pub/index.php/rbie/article/view/1275>. Acesso em: 07 set. 2016.

PERICO, Alisson; SHINOHARA, Cindi S.; SARMENTO, Cristiano D. **Sistema de reconhecimento de voz para automatização de uma plataforma elevatória**. 2014. 96 f. Trabalho de Conclusão de Curso (Engenharia Industrial Elétrica - Ênfase em Automação) – Departamento Acadêmico de Eletrotécnica, Universidade Tecnológica Federal do Paraná, Curitiba. Disponível em: <http://nupet.daelt.ct.utfpr.edu.br/tcc/engenharia/doc-equipe/2012_2_15/2012_2_15_monografia.pdf>. Acesso em: 02 nov. 2016.

PLUG. **Como arrumar as peças?** [S.l.], 2016. Disponível em: <http://www.plug.pt/forum/index.php?topic=61.270>. Acesso em: 02 nov. 2016.

QUINTANILHA, Leandro. Irresistível robô. **ARede**, São Paulo, n. 34, não paginado, mar. 2008. Disponível em: <www.revista.arede.inf.br/site/edicao-n-34-marco-2008/3920-irresistivel-robo>. Acesso em: 23 out. 2016.

ROBOMIND ACADEMY. **Introduction:** what is Robo/RoboMind? [S.l.], 2014. Disponível em: <http://www.robomind.net/en/introduction.htm>. Acesso em: 02 set. 2016.

ROZAK, Mike. **Talk to your computer and have it answer back with the Microsoft Speech API**. [S.l.], 1996. Disponível em: <https://www.microsoft.com/msj/archive/s233.aspx>. Acesso em: 04 jun. 2017.

SANTOS, Franklin L. et al. REDUC: a robótica educacional como abordagem de baixo custo para o ensino de computação em cursos técnicos e tecnológicos. In: WORKSHOP DE INFORMÁTICA NA ESCOLA, 16., 2010, Santos. **Anais eletrônicos...** [S.l.]: SBC, 2010. P. 1304-1313. Disponível em: <http://www.br-ie.org/pub/index.php/wie/article/view/2053>. Acesso em: 05 nov. 2016.

SCHIVANI, Milton. **Contextualização no ensino de física à luz da teoria antropológica do didático: o caso da robótica educacional**. 2014. 220 f. Tese (Doutorado em Educação) – Faculdade de Educação, Universidade de São Paulo, São Paulo. Disponível em: <http://www.teses.usp.br/teses/disponiveis/48/48134/tde-01122014-104322/>. Acesso em: 05 nov. 2016.

SILVA, Alzira F. **RoboEduc**: uma metodologia de aprendizado com robótica educacional. 2009. 135 f. Tese (Doutorado em Engenharia da Computação) – Faculdade de Engenharia Elétrica e de Computação, Universidade Federal do Rio Grande do Norte, Natal. Disponível em: <https://repositorio.ufrn.br/jspui/handle/123456789/15128>. Acesso em: 07 set. 2016.

SILVA, Diogo. **Um simulador 2D para a linguagem Robotoy**. 2016. 54 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TEVAH, Rafael T. **Implementação de um sistema de reconhecimento de fala contínua com amplo vocabulário para o português brasileiro**. 2006. 91 f. Dissertação (Mestrado em Engenharia Elétrica) – Programa de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro. Disponível em: <http://www.eletrica.ufrj.br/teses/textocompleto/2006053001.pdf >. Acesso em: 02 nov. 2016.

TORRENS, Maria G**. Robotoy:** ferramenta para uso de robótica no ensino de programação para crianças. 2014. 71 f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

WING, Rowan. **Intro to controllers**. [Boulder?], 2011. Disponível em: <http://correll.cs.colorado.edu/?p=852>. Acesso em: 07 set. 2016.

APÊNDICE A – Comandos Robotoy reconhecidos por voz

O Quadro 11 detalha os comandos da linguagem Robotoy que são reconhecidos por voz.

Quadro 11 – Comandos reconhecidos por voz

`andar para <#frente_tras> <#qdade_opcional>` onde: `<#frente_tras>` pode ser frente ou trás e `<#qdade_opcional>` é um valor opcional entre 1 e 20.

Definição:
Este comando faz com que o robô se movimente para frente ou trás de 1 a 20 passos. Caso não seja informada a quantidade de passos, o robô se movimentará apenas uma vez.
Exemplos (do que deve ser falado/reconhecido):
`andar para frente`
`andar para trás 20`

`virar para a <#esquerda_direita> <#qdade_opcional>` onde: `<#esquerda_direita>` pode ser esquerda ou direita e `<#qdade_opcional>` é um valor opcional entre 1 e 20.

Definição:
Este comando gira o robô 90º para a direita ou para a esquerda de 1 a 20 vezes. Caso não seja informada a quantidade, o robô girará apenas uma vez.
Exemplos:
`virar para a esquerda`
`virar para a direita 2`

`virar motor multiuso para a <#esquerda_direita> <#qdade>` onde: `<#esquerda-direita>` pode ser esquerda ou direita e `<#qdade>` é um valor entre 1 e 20.
Definição: Este comando faz com que o motor multiuso, responsável por sustentar o motor de distância, gire 90º em uma direção (direita ou esquerda) de 1 a 20 vezes. **Exemplos:** `virar motor multiuso para esquerda 1`
`emitir som`
Definição: Este comando emite um som no *buzzer* do robô e, no caso do simulador, emite um bip na saída de som padrão do sistema. **Exemplo:** `emitir som`
`se <#condição> <#comando> senão <#comando> fim do se` onde: `<#condição>` deve ser conforme especificado a seguir, `<#comando>` pode ser uma ou mais ocorrências de `andar`, `virar`, `emitir som` ou um comando de controle de fluxo, e a cláusula `senão <#comando>` é opcional.

Definição:
Trata-se de um comando de seleção, ou seja, se a `<#condição>` for verdadeira, os `<#comando>`s subsequentes serão executados, caso contrário, os `<#comando>`s subsequentes ao `senão`, se existirem, serão executados.
Exemplo:
`se não tem obstáculo`
`  andar para frente`
`senão`
`  andar para trás`
`fim do se`

`enquanto <#condição> <#comando> fim do enquanto`

onde:

`<#condição>` deve ser conforme especificado a seguir e `<#comando>` pode ser uma ou mais ocorrências de `andar`, `virar`, `emitir som` ou um comando de controle de fluxo.

Definição:

Trata-se de um comando de repetição, ou seja, caso a `<#condição>` seja verdadeira, os `<#comando>`s serão executados repetidas vezes até que a `<#condição>` se torne falsa.

Exemplo:

```
enquanto cor identificada = vermelho
    andar para frente
fim do enquanto
```

`<#condição>`

Definição:

Uma `<#condição>` pode ser:

```
<#condição>::= <#negação> tem obstáculo |
               cor identificada <#operador_relacional> <#cor>
<#negação>::= não | ε
<#operador_relacional>::= igual | diferente
<#cor>::= branca | branco | preta | preto | vermelha | vermelho |
          amarela | amarelo | verde | azul
```

Fonte: elaborado pelo autor.

APÊNDICE B – BNF da linguagem Robotoy

No Quadro 12 é apresentada a BNF da linguagem Robotoy, inicialmente especificada por Torrens (2014), mas alterada no decorrer do desenvolvimento deste trabalho. Observa-se que os comandos `andar` e `virar`, na especificação inicial, não precisavam ser seguidos pela quantidade de vezes que o movimento deveria ser executado, o que, nesse caso, seria repetido indefinidamente pelo robô Lego Mindstorms NXT. Já na programação do robô Arduino (BATISTA, 2016) e do simulador (SILVA, 2016), esses comandos sem o parâmetro fazem com que o robô se locomova apenas uma vez. Assim, a semântica dos comandos foi unificada, tendo efeito idêntico em todos os robôs, ou seja, `andar` e `virar` sem parâmetro faz com que o robô execute o comando apenas uma vez. Além disso, os comandos `parar de andar` e `parar de virar` foram removidos, pois tornaram-se desnecessários e não foram implementados por Batista (2016) e Silva (2016).

Quadro 12 – Especificação sintática da linguagem

```
<programa>          ::= <lista_comandos> <lista_rotinas>
<lista_comandos>    ::= ε | \n <lista_comandos> | <comando> \n <lista_comandos>
<comando>           ::= <controle_fluxo> |
                        <definicao_variavel_ou_invocacao_metodo> |
                        <acao>
<controle_fluxo>    ::= <se> | <enquanto>
<se>                ::= se <condicao> \n <lista_comandos> <senao> fim do se
<senao>             ::= ε | senao \n <lista_comandos>
<enquanto>          ::= enquanto <condicao> \n <lista_comandos> fim do enquanto
<definicao_variavel_ou_invocacao_metodo>
                    ::= <tipo> id <declaracao> | id <declaracao_opcional>
<declaracao>        ::= "<-" <expressao>
<declaracao_opcional>::= ε | "<-" <expressao>
<tipo>              ::= numero | texto | cor
<expressao>         ::= <aritmetica>
<aritmetica>        ::= <elemento> <add_sub>
<add_sub>           ::= ε | + <elemento> <add_sub> | - <elemento> <add_sub>
<elemento>          ::= <termo> <mul_div_mod>
<mul_div_mod>       ::= ε | * <termo> <mul_div_mod> |
                        / <termo> <mul_div_mod> |
                        % <termo> <mul_div_mod>
<termo>             ::= + <termo> |  - <termo> | id <concatenar_char> |
                        numerico  | ( <aritmetica> ) | literal <concatenar_char> |
                        <cor> | cor identificada
<cor>               ::= branca | branco | preta | preto | vermelha | vermelho |
                        amarela | amarelo | verde | azul
<concatenar_char>   ::= ε | . <opcoes>
<opcoes>            ::= id <concatenar_char> |
                        ( <aritmetica> ) <concatenar_char> |
                        literal <concatenar_char>
<acao>              ::= <andar> | virar  <virar> | <girar_roda> |
                        <parar> | <escrever> | <emitir_som>
<andar>             ::= andar para <frente_ou_tras>  <qdade_opcional>
<virar>             ::= para a <esquerda_ou_direita> <qdade_opcional> |
                        motor multiuso para a <esquerda_ou_direita> <qdade>
<girar_roda>        ::= girar roda <esquerda_ou_direita> para <frente_ou_tras>
<parar>             ::= parar de girar
<escrever>          ::= escrever <expressao>
<emitir_som>        ::= emitir som
<frente_ou_tras>    ::= frente | tras
<esquerda_ou_direita> ::= esquerda | direita
```

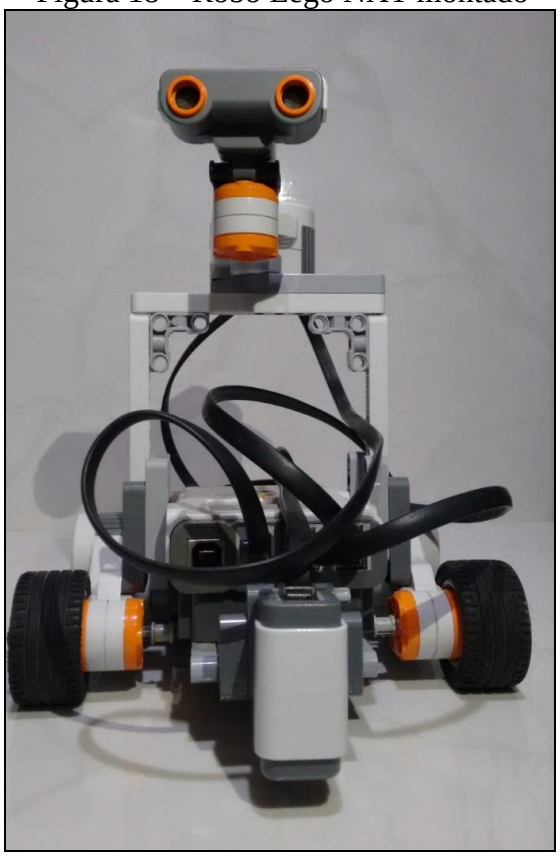
```
<qdade_opcional>    ::= ε | <expressao>
<qdade>             ::= <expressao>
<condicao>          ::= <tipo_condicao> <condicao_extra>
<tipo_condicao>     ::= <verificar_obstaculo> | <comparar>
<condicao_extra>    ::= ε | e <condicao> | ou <condicao>
<verificar_obstaculo>::= <nao> tem obstaculo
<nao>               ::= ε | nao
<comparar>          ::= <expressao> <operador_relacional> <expressao>
<operador_relacional>::= > | < | >= | <= | = | =/
<lista_rotinas>     ::= ε | <rotina> <lista_rotinas>
<rotina>            ::= rotina id \n <lista_comandos> fim da rotina \n
```

Fonte: elaborado pelo autor.

APÊNDICE C – Robôs Lego Mindstorms NXT e Arduino utilizados no trabalho

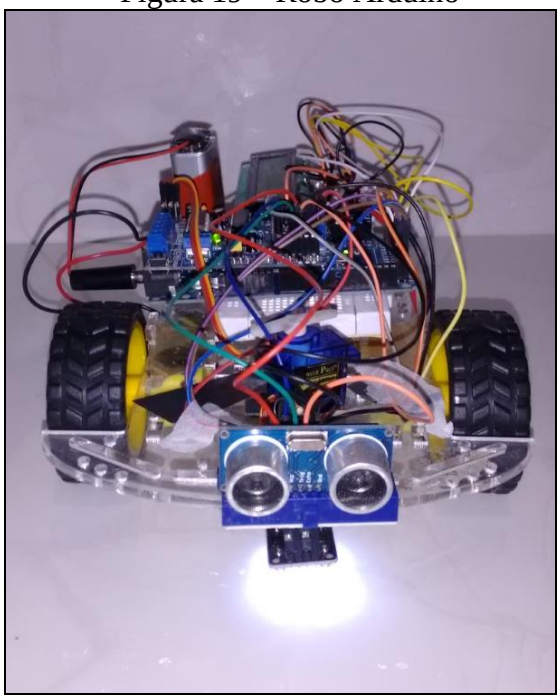
A Figura 18 e a Figura 19 apresentam, respectivamente, os robôs Lego Mindstorms NXT e Arduino usados no desenvolvimento do trabalho. Como base para a montagem do robô Lego Mindstorms NXT, utilizou-se o modelo criado por Torrens (2014), diferentemente da plataforma Arduino, cujo robô usado foi cedido por Batista (2016).

Figura 18 – Robô Lego NXT montado



Fonte: elaborado pelo autor.

Figura 19 – Robô Arduino



Fonte: elaborado pelo autor.