Instituto Federal do Rio Grande do Norte
Campus Natal - Central
Diretoria de Gestão e Tecnologia da Informação
Tecnologia em Análise e Desenvolvimento de Sistemas

# Relatório técnico de desenvolvimento da biblioteca Mirobot-Poti

**Mateus Oliveira Costa Bezerra**

Natal-RN
Janeiro de 2017

**Mateus Oliveira Costa Bezerra**

# Relatório técnico de desenvolvimento da biblioteca Mirobot-Poti

Trabalho de conclusão de curso de graduação do curso de Tecnologia e Análise em Desenvolvimento de Sistemas da Diretoria de Gestão e Tecnologia de Informação do Instituto Federal do Rio Grande do Norte como requisito parcial para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

*Linha de pesquisa*:
Nome da linha de pesquisa

Orientador

Leonardo Ataide Minora, Mestre

TADS – Curso de Tecnologia em Análise e Desenvolvimento de Sistemas
DIATINF – Diretoria Acadêmica de gestão e Tecnologia da Informação
CNAT – Campus Natal - Central
IFRN – Instituto Federal do Rio Grande do Norte

Natal-RN

Janeiro de 2017

Trabalho de Conclusão de Curso de Graduação sob o título *Relatório técnico de desenvolvimento da biblioteca Mirobot-Poti* apresentada por Mateus Oliveira Costa Bezerra e aceita pelo Diretoria de Gestão e Tecnologia da Informação do Instituto Federal do Rio Grande do Norte, sendo aprovada por todos os membros da banca examinadora abaixo especificada:

---

Leonardo Ataide Minora, Mestre
Presidente
DIATINF – Diretoria Acadêmica de Gestão e Tecnologia da Informação
IFRN – Instituto Federal do Rio Grande do Norte

---

Nome completo do examinador e titulação
Examinador
Diretoria/Departamento
Instituto

---

Nome completo do examinador e titulação
Examinador
Diretoria/Departamento
Universidade

Natal-RN, data da defesa (dia, mês e ano).

*Citação*

Autor

# Relatório técnico de desenvolvimento da biblioteca Mirobot-Poti

Autor: Mateus Oliveira Costa Bezerra
Orientador: Leonardo Ataide Minora, Mestre

## Resumo

O aumento de interesse pela área da programação voltada para jovens e crianças vem com o aumento da necessidade de novas estratégias e metodologias de ensino que se adequem a esse público. Além dessa necessidade, existe a necessidade de meios alternativos para o ensino de programação nos cursos superiores e técnicos, principalmente no contexto brasileiro já que a área de informática esta diretamente ligada à língua inglesa, na qual nem todos os alunos falantes do português tem domínio. Seguindo esses dois contextos nasce a proposta desse trabalho, que é juntar uma forma alternativa de ensino de programação, chamada de programação tangível, a qual nasceu para ser voltada ao ensino de programação para crianças; com uma linguagem de programação em português desenvolvida para o ensino de lógica de programação, o Potigol. Essa junção vai ser feita através da construção de uma biblioteca para a linguagem Potigol, que possibilite programar as ações de um robô chamado de mirobot.

*Palavras-chave*: programação tangível, potigol, mirobot.

# Título do trabalho (em língua estrangeira)

Author: Mateus Oliveira Costa Bezerra
Supervisor: Leonardo Ataide Minora, Mestre

## Abstract

O resumo em língua estrangeira (em inglês *Abstract*, em espanhol *Resumen*, em francês *Résumé*) é uma versão do resumo escrito na língua vernícula para idioma de divulgação internacional. Ele deve apresentar as mesmas características do anterior (incluindo as mesmas palavras, isto é, seu conteúdo não deve diferir do resumo anterior), bem como ser seguido das palavras representativas do conteúdo do trabalho, isto é, palavras-chave e/ou descritores, na língua estrangeira. Embora a especificação abaixo considere o inglês como língua estrangeira (o mais comum), não fica impedido a adoção de outras linguas (a exemplo de espanhol ou francês) para redação do resumo em língua estrangeira.

*Keywords*: Keyword 1, Keyword 2, Keyword 3.

# Sumário

**1 Introdução** p. 8

1.1 Objetivos . . . . . p. 9

1.1.1 Objetivo Geral . . . . . p. 9

1.1.2 Objetivos Específicos . . . . . p. 9

1.2 Metodologia . . . . . p. 9

1.3 Delimitação do trabalho . . . . . p. 10

1.4 Organização do trabalho . . . . . p. 10

**2 Mirobot e Potigol: tecnologias alternativas para o ensino de programação** p. 11

2.1 Mirobot: uma proposta de uso de robôs para ensino de lógica de programação . . . . . p. 11

2.1.1 Resumo da arquitetura do Mirobot . . . . . p. 12

2.1.2 Códigos e projetos do Mirobot . . . . . p. 12

2.2 Linguagem Potigol uma linguagem em português para o ensino de lógica de programação de computadores . . . . . p. 13

**3 Desenvolvimento da biblioteca mirobot-poti** p. 14

3.1 Definição dos métodos . . . . . p. 14

3.2 Implementação dos métodos . . . . . p. 16

3.3 O Potigol e o empacotamento da biblioteca . . . . . p. 17

3.4 Teste da implementação com exemplo de código . . . . . p. 17

**4 Considerações Finais** p. 19

4.1 Principais contribuições . . . . . p. 19
4.2 Limitações e dificuldades . . . . . p. 19
4.3 Trabalhos futuros . . . . . p. 20

**Referências** p. 21

**Apêndice A – Primeiro apêndice** p. 22

**Anexo A – Primeiro anexo** p. 23

# 1 Introdução

Atualmente o ensino de programação e robótica voltado aos jovens e crianças vêm ganhando espaço em praticamente todo o mundo, no entanto existe a fama da alta complexidade envolvida com esses dois assuntos. Por causa dessa alta complexidade, pesquisadores e professores procuraram metodologias e estratégias diferentes para um ensino de programação bem como de robótica. Uma das novas estratégias de ensino programação é baseada na programação tangível (HORN; JACOB, 2007) que busca contornar os problemas da alta complexidade ligada à informática, com o princípio de modificar a forma de programar que normalmente é usando recursos virtuais e intocáveis, por objetos que podem ser tocados e vistos no mundo real.

O Google mantém uma plataforma baseada na programação tangível chamada de *Project Bloks* (`https://projectbloks.withgoogle.com`) que possibilita a criação de programas através da combinação de pequenos módulos no formato de blocos que se encaixam. Cada bloco tem um comportamento específico e pode ser combinado com outros blocos de diversas formas produzindo novos comportamentos mais complexos. Dentro dessa mesma plataforma também existe um pequeno robô que consegue produzir desenhos usando uma caneta, chamado de mirobot. A programação das ações do mirobot pode ser feita de diversas formas, que vai desde a combinação dos módulos do Project Bloks, até a construção desses comportamentos por meio de linguagens de programação como: JavaScript, Python e Scratch.

Outra dificuldade no aprendizado de programação, além do alto nível de abstração ligado à computação, são as sintaxes complexas nas linguagens de programação. Principalmente quando a linguagem de programação é baseada em um idioma diferente da língua materna dos alunos. Pensando nisso, foi que no ano de 2011 iniciou-se no Instituto Federal de Educação, Ciências e Tecnologia do Rio grande do Norte (IFRN), por meio do professor Leonardo Reis Lucena e alunos bolsistas, a projeto da linguagem Potigol (LUCENA; LUCENA, 2016). A proposta do Potigol é ser uma linguagem de programação que tem sua sintaxe baseada no idioma português, multiparadigma, com suporte de softwares

para o ensino de programação dos falantes desse mesmo idioma.

Este trabalho busca juntar essas duas experiências já citadas, de forma que elas possam entrar em sintonia, levando em conta que elas têm o mesmo objetivo final, auxilar no ensino de programação. Dessa forma o objetivo principal do trabalho é permitir que exista uma alternativa além das linguagens já suportadas para a programação das ações do mirobot.

## 1.1 Objetivos

### 1.1.1 Objetivo Geral

Este trabalho tem por objetivo geral implementar uma biblioteca que permita programar na linguagem Potigol comportamentos de um robô Mirobot.

### 1.1.2 Objetivos Específicos

Fazem parte dos objetivos específicos deste trabalho:

- Descrever o protocolo de troca de mensagens do robô Mirobot;
- Desenvolver a biblioteca mirobot-poti na linguagem de programação Java;
- Implementar exemplo de utilização da biblioteca Mirobot-Poti em Potigol utilizando um simulador para o robô;

## 1.2 Metodologia

Esse trabalho seguiu as seguintes etapas. Primeiramente foram pesquisadas e entendidas as ferramentas já existentes desenvolvidos pelo Google, entre elas estão: mirobot-sim, um simulador do robô mirobot escrito em Javascript; mirobot-py e mirobot-js, que são implementações das bibliotecas para comunicação com o robô nas respectivas linguagens: Python e Javascript. Após essa etapa foi definida a interface(assinatura dos métodos) da biblioteca mirobot-poti que seria posteriormente implementada pora comandar os comportamentos do mirobot, a qual a nomeação foi feita com base no protocolo do mirobot diponibilizado em: `http://learn.mirobot.io/docs/understanding-the-mirobot-protocol/`. A próxima etapa foi entender como fazer a importação de uma biblioteca externa na linguagem Potigol, a qual foi compreendida através de simples testes compilando códigos da

linguagem Java e importando o resultado em códigos no Potigol. Depois foi feita uma segunda validação com a biblioteca se comunicando com o simulador do mirobot. E por fim, foram implementados todos os métodos anteriormente nomeados e criado um exemplo de uso da utilização da biblioteca mirobot-poti.

## 1.3 Delimitação do trabalho

Devido a dificuldades na importação do robô, foi utilizado um simulador ao invés do próprio robô. Por outro lado, a ausência do robô não teve impacto no desenvolvimento da biblioteca já que existia todo um ambiente pronto para suportar os testes necessários.

## 1.4 Organização do trabalho

O trabalho está organizado na seguinte estrutura: este primeiro capítulo, com uma introdução sobre o trabalho; o segundo capítulo com o referencial teórico utilizado relativo ao mirobot e a linguagem de programação Potigol, o terceiro capítulo que contém a descrição da implementação e os testes, e por fim o quarto capítulo com as considerações finais do trabalho.

# 2 Mirobot e Potigol: tecnologias alternativas para o ensino de programação

## 2.1 Mirobot: uma proposta de uso de robôs para ensino de lógica de programação

Um dos principais objetivos da programação tangível é tornar o que é complexo e virutal em algo simples que pode ser visto e manuseado facilmente no mundo real (HORN; SOLOVEY; JACOB, 2008; HORN; JACOB, 2007; MCNERNEY, 2000). Esse mesmo pensamento pode ser estendido da seguinte forma: durante o aprendizado de programação é comum que os alunos apliquem seus conhecimentos para manipular recursos virtuais como: números, cadeias de caracteres (strings) e valores booleanos. Como forma de alternativa ao uso dos objetos virtuais, existe a possibilidade de se usar robôs e programar as suas ações. Tenta-se com isso tornar mais intuitivo para visualizar e aprender a lógica de programação, principalmente para crianças, pois os resultados dos seus programas serão mostrados em um mundo real e não em uma tela de computador.

Devido a esse cenário, a empresa Google desenvolveu o Mirobot. O Mirobot é em um robô cujas ações podem ser programadas tanto através das linguagens de programação: Scratch, Javascript e Python; como também através um módulo que também implementa a noção de programação tangível chamado *Project Bloks*. A proposta do mirobot é ser um robô que recebe comandos de uma determinada fonte via Wi-Fi e ao realizar essas ações ele vai marcando o seu trajeto com uma caneta, resultando em um desenho. Entre as principais ações que ele realiza estão: se movimentar para frente ou para atrás, girar para esquerda ou para direita, descer ou levantar a caneta (possibilitando com que ele desenhe ou não seu trajeto quando desejado).

### 2.1.1 Resumo da arquitetura do Mirobot

O Mirobot é composto basicamente por um chassi de madeira que é formado por: duas rodas responsáveis pela sua movimentação, uma terceira roda no formato esférico (que ajuda na sustentação e movimentação do robô) e um orifício onde é possível fazer o encaixe de uma caneta. Já na parte dentro do seu corpo, existem dois motores de passo, cada um responsável por uma das rodas, os quais são encarregados pela sua movimentação. Além desses dois motores, ainda há um servomoto que fica responsável pelo acionamento da caneta. Na parte de controle ele contém um arduíno nano responsável pelo acionamento dos motores, e a comunicação via Wi-Fi e o tratamento dos dados fica por conta de um controlador principal, o qual é desenvolvido pela mesma empresa do mirobot.

A comunicação com o Mirobot é feita via Wi-Fi e ele trabalha com a tecnologia chamada websocket (FETTE; MELNIKOV, 2011). O websocket consiste em uma tecnologia que oferece um meio de comunicação bidirecional através de um único soquete, essa tecnologia normalmente é usada em browsers web e servidores web. As mensagens que são enviadas e recebidas pelo Mirobot estão no formado JSON (Javascript Object Notation).

Todo esse projeto é mantido em um repositório no portal Github nos endereços: chassi em `https://github.com/mirobot/mirobot-chassis`; firmware arduino em `https://github.com/mirobot/mirobot-arduino` e o firmware wifi em `https://github.com/mirobot/mirobot-wifi`.

### 2.1.2 Códigos e projetos do Mirobot

O Mirobot é rodeado por um grande ecossistema de projetos relacionados a ele no repositório Github (`https://github.com/mirobot`), entre eles estão:

- *mirobot-sim*, que é um simulador que tem a função de receber as mensagens em JSON via websocket e simular a parte de comunicação do robô;
- *mirobot-py*, mirobot-js e mirobot-scratch que são bibliotecas para o controle do mirobot para as respectivas linguagens Python, Javascript e Scratch;

Além dos projetos no Github ainda existe uma gama de aplicativos web com plataformas de interação com o mirobot, podem ser encontradas em: `http://apps.mirobot.io/`. Para cada uma das linguagens de programação há uma aplicação web que possibilita comunicação com o robô, em todas essas aplicações, além da possibilidade de se conectar

com o robô físico e enviar os comandos, é possível acompanhar o seu trajeto por meio de um simulador. Ainda existem outros aplicativos relacionados a comunicação com o mirobot, entre eles estão: point & click, que é uma interface para definir o trajeto do mirobot apenas usando clicks na tela de um simulador; e remote control, que atua como um remoto possibilitando o envio das ações para o mirobot por meio de botões na interface web.

## 2.2 Linguagem Potigol uma linguagem em português para o ensino de lógica de programação de computadores

No ano de 2011 no Instituto Federal de Educação, Ciências e Tecnologia (IFRN), Campi Natal/Central (CNAT), foi iniciado o projeto da linguagem de programação Potigol pelo professor Leonardo Reis Lucena e alunos bolsistas. A linguagem nasceu com a proposta principal de auxiliar o ensino das disciplinas de programação de computadores, e leva como principal característica a sua sintaxe em português. Entre algumas outras características da linguagem estão: multiparadigma, tipagem estática com inferência de tipos, projetada para ser usada por alunos iniciantes.

Atualmente a linguagem Potigol com tutorial e seus softwares utilitários estão disponíveis em `http://potigol.github.io` e seus projetos com seus códigos-fontes em `https://github.com/potigol/`.

# 3 Desenvolvimento da biblioteca mirobot-poti

Este capítulo está dividido em quatro seções, as quais simboliza as etapas do desenvolvimento da biblioteca mirobot-poti. A primeira seção descreve o protocolo de funcionamento do mirobot e a definição da classe mirobot, assim como os seus métodos. A segunda seção, apresenta o processo de implementação da biblioteca e, após o seu desenvolvimento exemplificar com um caso de uso a usando o potigol. A terceira seção descreve a parte de empacotamento do projeto. A quarta seção finaliza com exemplos de código e testes da implementação. O código do projeto que foi desenvolvida está disponível em: `https://github.com/ensino-de-programacao/mirobot-potigol`.

## 3.1 Definição dos métodos

O início do desenvolvimento do mirobot-poti se baseou principalmente na biblioteca mirobot-py (`https://github.com/mirobot/mirobot-py`). O mirobot-py contém os artifícios necessários para se comunicar com mirobot por meio da linguagem Python, esse código se baseia em uma classe principal que representa o robô mirobot e seus métodos (ações).

```
from mirobot import Mirobot

# Connect to Mirobot
mirobot = Mirobot()
mirobot.autoConnect()

# Put the pen down
mirobot.pendown()

# Move forward 100mm
mirobot.forward(100)

# Move back 100mm
mirobot.back(100)

# Turn left 45 degrees
mirobot.left(45)

# Turn right 45 degrees
mirobot.right(45)

# Lift the pen up
mirobot.penup()

# Beep for a second
mirobot.beep(1000)

# Print the state of the collision sensors
print(mirobot.collideState())

# Print the state of the line following sensors
print(mirobot.followState())

# Disconnect from Mirobot
mirobot.disconnect()
```

Figura 1: Definição de alguns métodos da classe principal do mirobot-py.

Como pode ser visto na Figura 1, esses são os métodos da classe Mirobot. Dentre eles estão os que têm o propósito de se conectar com o robô: tanto automaticamente, por meio de uma detecção automática dos dispositivos mirobot; quanto manualmente, inserindo o endereço IP do robô e a porta de comunicação desejada; e por último um método para se desconectar e fechar a conexão WebSocket. Além dos métodos de conexão, ainda tem os de movimentação, e entre eles estão: a movimentação para trás e para frente (medida em milímetros), movimentação para os lados (girar sobre o próprio eixo usando o grau como medida) direito e esquerdo, e a ação "beep" que recebe um valor em segundos da duração do som, por último, descer e levantar a caneta. E para finalizar, os métodos que são responsáveis por: informar a velocidade em que os pacotes estão sendo enviados e recebidos via Wi-Fi (ping), retorna as informações captadas pelo sensor de colisão, retornar as informações captadas pelo sensor de seguimento de linha.

Baseando-se nessas definições, foi criado a classe Mirobot e seus métodos na linguagem Java. Para poder seguir o padrão do potigol, os nomes dos métodos foram traduzidos do inglês para o português. Depois de finalizada a etapa de definição dos métodos foi passado para a etapa de implementação.

## 3.2 Implementação dos métodos

A implementação da biblioteca mirobot-poti, foi feita através da linguagem Java. Para a comunicação via websocket foi usado o projeto Glassfish Tyrus(`https://tyrus.java.net/`). O critério de escolha pelo Glassfish Tyrus foi devido a possibilidade de usar um servidor *standalone*, ou seja, um servidor que roda dentro da própria aplicação, caso contrário seria necessário um servidor externo para abrigar a aplicação.

Primeiramente foi estudado o funcionamento da biblioteca tyrus e feito alguns exemplos simples de comunicação websocket. Depois de entendido a parte básica, foi seguido o próximo passo de criação de um socket handler. O websocket handler é o responsável por enviar e receber as mensagens do mirobot, ele representa o baixo nível de comunicação e está diretamente ligado ao websocket. As mensagens enviadas e recebidas pelo socket handler são no formato JSON (Java Object Notation), e seguem a seguinte estrutura:

```
1 {"cmd":"left", "id": "Ir7l002c", "arg": "90"}
```

Figura 2: Exemplo de comando no formato JSON.

Como pode ser visto na Figura 2, as mensagens JSON que são enviadas do mirobot se baseiam em três informações: o "cmd" que representa o comando desejado que o mirobot realize; o "id" que representa a identificação da mensagem que foi enviada (posteriormente vai ser explicado a sua utilidade); e o "arg" que representa o argumento que vai ser passado ao comando, se o comando não precisar de argumento, como é o caso do comando usado para descer a caneta ou de verificar o ping, esse atributo é ignorado. Então basicamente o funcionamento da biblioteca em linhas gerais, se resume em traduzir um comando para o formato JSON e enviá-lo via websocket obedecendo a estrutura exemplificada na Figura 2.

O mirobot a cada comando recebido retorna uma mensagem em JSON para informar a situação da execução do comando recebido. O atributo "id" serve para identificar o comando o qual o mirobot está informando a situação. Outro ponto importante é que, alguns comandos do mirobot demoram mais para ser executados que outros, dependendo do tamanho do argumento. Para exemplificar essa questão temos que levar em conta o seguinte fator, se for enviado um comando para que o mirobot ande 10 mm para frente, ele irá demorar um certo tempo para poder se deslocar os 10 mm, já se for enviado o mesmo comando mudando o argumento para 20 mm, o mirobot vai demorar o dobro de tempo para completar o percurso.

Considerando que o mirobot não empilha as mensagens recebidas, e executa apenas uma por vez, fica a cargo da biblioteca fazer esse controle, pois se o robô receber um comando enquanto estiver no meio da execução de outro, irá ocorrer um erro. A estratégia usada para resolver esse problema foi a seguinte: todos os comandos que serão enviados para o mirobot são guardados em uma fila, e vão sendo enviados à medida que o tempo estimado de duração de um comando simples seja cumprido, já no caso dos comandos que têm variação de duração em decorrer do tamanho do argumento passado, usa-se esse mesmo argumento e o comando enviado para se estimar o tempo que vai ser esperado até o envio do próximo comando da fila.

Depois de completa esses passos, foi incorporado a classe mirobot o WebsocketHandler, o qual é usado por todos os seus métodos já definidos anteriormente. Depois de finalizada essa etapa foi gerado o arquivo da biblioteca, importado e testado no Potigol.

## 3.3 O Potigol e o empacotamento da biblioteca

Em paralelo com o desenvolvimento da biblioteca mirobot-poti, foi testado a importação do formato jar do Potigol, todas as tentativas iniciais foram sem sucesso pois o Potigol não conseguia fazer a importação do arquivo jar. Depois de muitas tentativas foi descoberto que o class-path do Potigol não adicionava os arquivos jar automaticamente, pois apesar de incluso no class-path o diretório no qual o arquivo jar se encontrava, só era achado dentro do diretório os arquivos class. O problema foi solucionado descompactando o arquivo jar no diretório principal do Potigol. Outra forma que daria para resolver esse problema seria adiconando manualmente o caminho completo do arquivo jar no class-path.

No momento de compilação do projeto mirobot-poti utilizando o maven para automatizar a compilação, foi usado um plugin do maven chamado “jar-with-dependences” para que na hora da compilação as dependências do projeto (Glassfish Tyrus) seja inserido dentro do arquivo jar, possibilitando que o projeto após compilado dependa apenas de um único arquivo jar para ser executado.

## 3.4 Teste da implementação com exemplo de código

A última etapa do trabalho ficou com o teste na linguagem Potigol usando a biblioteca mirobot-poti para se comunicar com o simulador do robô (mirobot-sim). O para a realização do teste foi usado uma adaptação na classe mirobot, pois não foi encontrado

uma forma de instanciar um novo objeto dentro do Potigol. Essa adaptação foi o uso do padrão singleton para, a qual a classe fica responsável por gerar uma instancia da classe e retornar a variável no Potigol. Após a implementação do teste, foi completa a etapa de desenvolvimento do trabalho.

```
use "br.edu.ifrn.mirobot"

mirobot = Mirobot.pegarInstancia

mirobot.conectar("ws://localhost:8899")

escreva "Usando o Mirobot."

mirobot.ping

mirobot.descerCaneta
mirobot.esquerda("10")
mirobot.moverParaTras("20")
mirobot.direita("90")
mirobot.moverParaFrente("40")
mirobot.levantarCaneta

mirobot.beep("10")

mirobot.desconectar

escreva "Fim da execução."
```

# 4 Considerações Finais

Com o término da fase de desenvolvimento desse trabalho foi possível alcançar o objetivo geral, permitir programar comportamentos de um robô mirobot na linguagem Potigol. Em relação aos objetivos específicos, todos foram alcançados como o planejado: foi descrito o protocolo usado no mirobot, desenvolvido a biblioteca mirobot-poti e implementado um exemplo em Potigol do uso da biblioteca.

## 4.1 Principais contribuições

A contribuição pessoal foi no aprendizado de novas tecnologias, tais como: o websocket, desenvolvimento de uma biblioteca na liguagem Java e o contato com a linguagem Potigol. Além da oportunidade de interagir com novas áreas como foi o caso da programação tangível.

A principal contribuição para o projeto Potigol é permitir criar uma "sala de aula" mais interativa. Os alunos irão programar em uma linguagem em português comportamentos de um robô, tal como desenvolver a lógica para desenho de um quadrado e de outros objetos geométricos mais complexos.

## 4.2 Limitações e dificuldades

A limitação principal do projeto foi a ausência de robô para teste real devido a um problema na importação do equipamento. Além da ausência do robô físico para os testes, uma outra limitação foi que a biblioteca mirobot-poti não foi implementada com uso de threads. Uma biblioteca que usou threads foi a mirobot-py. Por causa dessa limitação, todos os comportamentos são síncronos, não sendo possível o enfileiramento de comandos controlados pela própria biblioteca mirobot-poti.

Um obstáculo no projeto foi entender como o Potigol usa bibliotecas externas e assim

poder fazer o mirobot-poti uma alternativa no ensino, ou seja, integrar na programação a biblioteca apenas quando necessário. Foram realizadas algumas interações com a equipe do projeto Potigol para que a biblioteca fosse usada apenas especificando como biblioteca externa (uso de *classpath*).

Ficou o *bug* na saída dos comandos com o uso específico do comando java -cp potigol.jar:mirobot.jar:. br.edu.ifrn.potigol.Principal -w UsandoMirobot.poti, que resultou na mensagem: *aguarde...*, após um tempo a mensagem era apagada e não tinha nenhum resultado de sucesso nem de erro.

## 4.3 Trabalhos futuros

A partir deste trabalho, existem algumas possibilidades de continuação no desenvolvimento de software, desenvolvimento de rôbos bem como na avaliação do ensino de programação. Quanto ao desenvolvimento de software, é possível destacar:

- Desenvolvimento de um *shell* permita enviar comandos diretamente ao um mirobot, mirobot-poti-shell;
- Extensão do mirobot-poti-shell para permitir comandos potigol, inclusive a carga de bibliotecas em tempo de execução;
- Desenvolver outras bibliotecas que permitam manipulação de outros robôs, como por exemplo o robô músico do Google ou carros robôs usados nas olimpíadas de robótica nacionais.

No processo de ensino de programação, é importante fazer testes controlados em laboratório para avaliar a efetividade do uso de potigol para manipular o robô assim como para o desenvolvimento de lógica de programação.

# Referências

FETTE, I.; MELNIKOV, A. *The WebSocket Protocol*. [S.l.], December 2011. `http://www.rfc-editor.org/rfc/rfc6455.txt`. Disponível em: <http://www.rfc-editor.org/rfc/rfc6455.txt>.

HORN, M. S.; JACOB, R. J. K. Designing tangible programming languages for classroom use. In: *Proceedings of the 1st International Conference on Tangible and Embedded Interaction*. New York, NY, USA: ACM, 2007. (TEI 07), p. 159–162. ISBN 978-1-59593-619-6. Disponível em: <http://doi.acm.org/10.1145/1226969.1227003>.

HORN, M. S.; SOLOVEY, E. T.; JACOB, R. J. K. Tangible programming and informal science learning: Making tuis work for museums. In: *Proceedings of the 7th International Conference on Interaction Design and Children*. New York, NY, USA: ACM, 2008. (IDC '08), p. 194–201. ISBN 978-1-59593-994-4. Disponível em: <http://doi.acm.org/10.1145/1463689.1463756>.

LUCENA, L. R.; LUCENA, M. Potigol, a programming language for beginners. In: *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*. New York, NY, USA: ACM, 2016. (ITiCSE '16), p. 368–368. ISBN 978-1-4503-4231-5. Disponível em: <http://doi.acm.org/10.1145/2899415.2925493>.

MCNERNEY, T. S. MASTER OF SCIENCE IN MEDIA ARTS AND SCIENCES, *Tangible Programming Bricks: An approach to making programming accessible to everyone*. jun. 2000.

# APÊNDICE A – Primeiro apêndice

Os apêndices são textos ou documentos elaborados pelo autor, a fim de complementar sua argumentação, sem prejuízo da unidade nuclear do trabalho.

# ANEXO A – Primeiro anexo

Os anexos são textos ou documentos não elaborado pelo autor, que servem de fundamentação, comprovação e ilustração.