Desenvolvimento de uma biblioteca para o controle de um robô móvel remoto

Gustavo L. de Oliveira[1], Iago P. Gomes[2], Raphael Christian dos S. Oliveira[2], Jenifer de J. Jang[2], Roque M. P. Trindade[3], Alzira F. Silva[3], Maísa S dos S. Lopes[3]

[1]Universidade Estadual do Sudoeste da Bahia (UESB)
Estrada do Bem Querer, Km 4, s/n - Bairro Universitário,
CEP 45031-900 - Vitória da Conquista, Bahia, Brasil.

{g.cdcomp, iago.pg00, raphach7, jeniferjangj}@gmail.com,
roquetrindade@uesb.edu.br, {arizlas, msslopes}@gmail.com

Abstract. *The programming teaching/learning process is a complex activity, because its high level of abstraction and formalism. The LARA, Remote Laboratory in VLE, offers experiments to improve this process, contextualizing with the use of robots. However, programming a robot requires knowledge about its hardware, making difficulties the beginner students understanding. This paper presents the development of a control library for L1R2, that allows the student programming it, without concerned with its implementation details. The test shown that 80% of the codes compiled and uploaded to the robot were syntactically correct, which indicates that the library is simple and easy to use, once the students did not have difficulty to use it nor to assimilate its commands.*

Resumo. *O processo de ensino/aprendizagem de programação é uma atividade complexa, devido ao alto grau de abstração e formalismo. O LARA, Laboratório Remoto em AVA, oferece experimentos para melhorá-lo, contextualizando com uso de robôs. Contudo, programar um robô exige conhecimentos de seus componentes físicos, o que dificulta a compreensão dos alunos iniciantes. Este trabalho apresenta o desenvolvimento de uma biblioteca para o L1R2, que permite ao aluno programá-lo sem se preocupar com os detalhes de sua implementação. O teste mostrou que 80% dos códigos compilados e enviados ao robô estavam sintaticamente corretos, o que indica que a biblioteca é simples e fácil de usar, pois os alunos não tiveram dificuldade no uso e assimilação dos comandos.*

1. Introdução

Devido às grandes mudanças tecnológicas que vêm ocorrendo nos últimos anos, o ensino e aprendizagem de programação tem se tornado cada vez mais fundamental, entretanto, por conta do rigor matemático, lógico e o alto nível de abstração exigidos nas resoluções de problemas de programação, seu ensino/aprendizagem torna-se uma tarefa árdua, especialmente nos semestres iniciais dos cursos de computação [dos Santos and Costa 2006]. Desta forma, diversas pesquisas têm sido desenvolvidas a fim de propor ferramentas que auxiliem nesse processo [Özmen and Altun 2014]. Uma delas é a robótica, pois pode ajudar no desenvolvimento de habilidades como raciocínio lógico, resolução de problemas e trabalho em equipe [Benitti 2012].

Entretanto, desenvolver robôs educacionais envolve conceitos de diferentes áreas, como a ciência da computação, a engenharia elétrica, a engenharia da computação, a matemática, etc [Niku 2013]. Por conta disso, a aplicação de tais robôs no processo de ensino/aprendizagem deve ser cuidadosamente procedida, evitando uma sobrecarga de conteúdo aos alunos [Alimisis 2013].

Tendo em vista essas questões, o LARA - Laboratório Remoto em Ambiente Virtual de Aprendizagem, disponibiliza experimentos remotos de robótica para o ensino de programação [Lopes et al. 2016]. Os alunos podem acessá-lo e programá-lo por meio da internet, em horários compatíveis com sua disponibilidade, além disso, o mesmo experimento com o mesmo *hardware* e infraestrutura pode ser compartilhado com vários alunos, o que garante escalabilidade, portabilidade e redução de custo [Kalúz et al. 2013].

Atualmente, o LARA possui um experimento de robótica móvel, em que os usuários controlam um robô , chamado de LARA *Remote Robot* (L1R2), utilizando uma IDE online. Neste caso, os alunos devem focar em aprender os conceitos de programação e não devem se preocupar com detalhes relacionados a construção do robô.

Esse experimento é baseado nas competições da Olimpíada Brasileira de Robótica (OBR) e na categoria *Junior Rescue A* da Robocup. Assim, os principais objetivos do L1R2 são seguir linha e desviar de obstáculos, ainda assim, o robô possui outros sensores e atuadores a disposição dos usuários, tornando o experimento mais versátil.

O presente trabalho discorre sobre a metodologia adotada para desenvolvimento do *hardware* e do *software*, com ênfase no desenvolvimento da biblioteca de controle do L1R2, que abstrai conceitos de programação de microcontroladores para facilitar seu uso, principalmente para alunos que estão aprendendo a programar. Além dessa seção, o artigo está dividido em: seção 2, que descreve a metodologia utilizada no projeto e implementação do sistema robótico; seção 3, onde é descrito a arquitetura de *hardware* do L1R2; seção 4, em que é apresentado a modelagem da biblioteca, e detalhes sobre a implementação; seção 5, onde são apresentados os resultados obtidos com sua utilização em um curso do LARA; e, seção 6, que mostra a conclusão do artigo.

2. Metodologia

O L1R2 como sistema robótico é constituído de uma parte de *software* e outra de *hardware* [Niku 2013], aumentando a dificuldade na sua construção e controle, pois tais partes possuem uma relação muito próxima, em que um único componente físico pode acarretar em mudanças consideráveis no desenvolvimento do programa, e vice-e-versa [Barros and Cavalcante 2010]. Para tanto, existem metodologias chamadas de concorrentes que permitem o projeto e implementação de robôs, considerando as características de suas partes, além de explorar esses conflitos [Zendoia et al. 2013]. No projeto do L1R2, foi utilizada a metodologia *Codesign*, que segundo [Thomas et al. 1993] é uma das melhores soluções para sistemas heterogêneos. A figura 1 apresenta o modelo da arquitetura utilizada.

Esta arquitetura foi baseada em outra proposta por [Thomas et al. 1993], todavia, os autores utilizam simulações para o teste prévio dos sistemas embarcados. No caso do L1R2, as simulações foram substituídas por testes de *hardware* e *software*, realizados previamente à integração de componentes no robô.

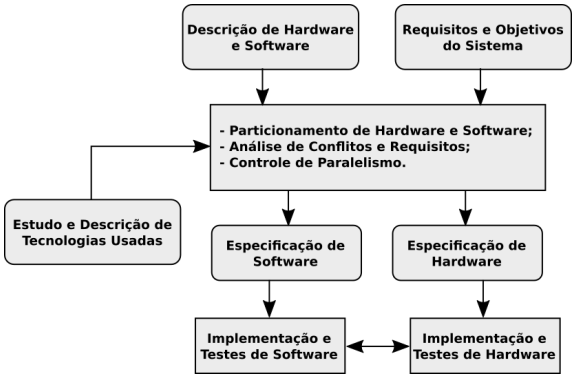

Figura 1. Arquitetura Codesign

A primeira fase do *Codesign* consiste em elencar as funções e objetivos do sistema de uma maneira mais genérica, em outras palavras, seria especificar o que é esperado do experimento. Após isso, os requisitos de cada parte são obtidos a partir da análise dessas funções, permitindo que os conflitos existentes sejam encontrados, analisados e resolvidos. A seguir são apresentados as funções do L1R2, que foram listados por meio de reuniões de toda a equipe do projeto, pois uma característica importante dessa fase é que os detalhes mais específicos sobre implementação devem ser evitados.

- **F01:** Os alunos poderão praticar durante os experimentos todos os conceitos abordados na disciplina de Algoritmos e Programação I, do curso de Ciência da Computação - UESB;
- **F02:** O experimento deve ficar online 24 horas por dia, todos os dias da semana;
- **F03:** O robô deve seguir as regras da Olimpíada Brasileira de Robótica, no que diz respeito à seguir linha e desviar obstáculos;
- **F04:** O robô deve voltar ao ponto inicial da arena sempre que for solicitado;
- **F05:** O aluno pode programar o robô utilizando funções de alto nível ou implementar suas próprias funções;
- **F05:** O robô deve se comunicar com o servidor de laboratório;
- **F06:** O robô deve executar suas manobras (frente, direita, esquerda, ré) corretamente.

A função F05 diz respeito a uma característica muito importante do experimento e objetivos do LARA. Como este foi projetado para que alunos pratiquem suas habilidades de programação, tornando mais contextualizado o processo de aprendizagem, fez-se necessário um meio que tornasse mais fácil o controle do robô, uma vez que em muitos casos a programação de microcontroladores exigem conhecimentos bastante específicos sobre a arquitetura do *hardware* [Martins et al. 2010]. Assim, foi projetada e implementada uma biblioteca de controle para L1R2, que omite aos usuários os detalhes de programação dos componentes (sensores e atuadores), mas permite também que aqueles em um nível mais avançado possa implementar suas próprias funções, neste caso, exercitando os conceitos de programação de sistemas embarcados e robôs móveis.

Outras funções também influenciaram na criação dessa biblioteca, como a F01, que exigem que o planejamento dos métodos e funções disponibilizadas permitam a prática dos assuntos da disciplina “Algoritmos e Programação I”, por exemplo: criação de variáveis; laços de repetições; desvios condicionais; criação e uso de funções e métodos;

passagem de parâmetros por referências; etc. E as funções F03 e F07 que, por sua vez, são importantes para a especificação dos requisitos funcionais da biblioteca.

3. Arquitetura de hardware

O L1R2 é formado por uma parte física e uma biblioteca de controle. Como o propósito da biblioteca é justamente abstrair detalhes de implementação dos componentes do robô (sensores, atuadores e o microcontrolador), então a definição da arquitetura de *hardware* norteia o desenvolvimento do *software*. Assim, a partir da análise das funções definidas na primeira fase da metodologia adotada e no estudo das competições que o experimento baseia-se, foi possível determinar todos os componentes necessários. Depois disso, analisou-se as influências destes no projeto da biblioteca, uma vez que cada um possui sua forma de controle e uso por meio do microcontrolador.

A figura 2 apresenta a arquitetura de comunicação entre os componentes e o microcontrolador. Nela é possível notar algumas características importantes, como o uso da plataforma de prototipação Arduíno Mega 2560 para controle do robô, e a forma com que os sensores e atuadores comunicam-se com esta.

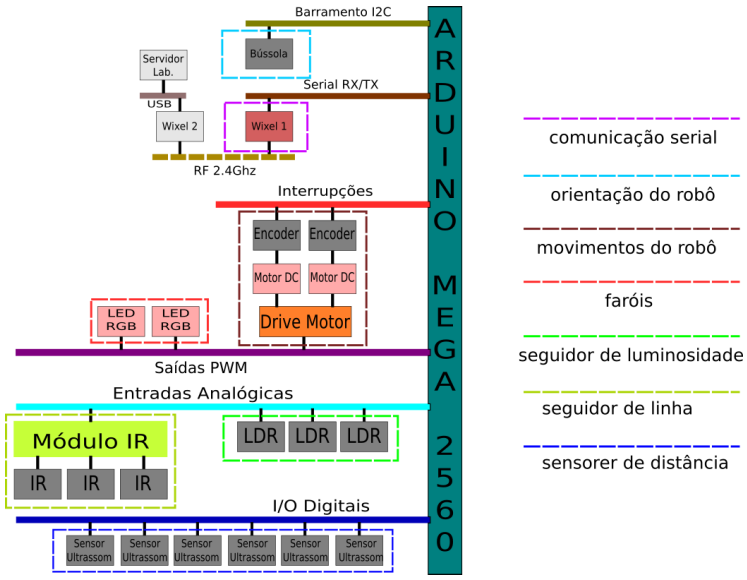

Figura 2. Arquitetura de comunicação de componentes L1R2

Na análise desta arquitetura, dois outros componentes merecem destaque. O primeiro caso é a bússola (sensor GY-271), ligada aos pinos que suportam o protocolo de comunicação I^2C (*Inter-Integrated Circuit*), que interliga periféricos em barramento. Para isso, o arduino possui uma biblioteca, chamada de "Wire.h"[1], que implementa os métodos necessários para a comunicação do microcontrolador e os sensores. Além do mais, como se trata de um sensor inercial, foi necessário a criação de um método de calibração e conversão da angulação obtida que é baseada na referência dos polos magnéticos, para uma local, permitindo ao usuário definir qual o "norte" do robô, ou

[1]A documentação da biblioteca está disponível no link: `https://www.arduino.cc/en/Reference/Wire`

seja, qual o ângulo escolhido como ponto 0. Estes métodos fazem parte de uma segunda biblioteca, chamada de "Compass.h", que foi baseada em outra[2] de mesmo nome.

No segundo caso, foram considerados as relações dos encoders, que estimam o deslocamento das rodas do robô, com a programação dos movimentos deste. Os sensores funcionam por meio de interrupções no microcontrolador, e com base nesses dados, são calculados o deslocamento longitudinal e orientação [Borenstein et al. 1996]. Assim, foi utilizado uma biblioteca chamada "Encoder.h"[3], para a leitura dos sensores, e a "Carro.h" que implementam as funções de movimento. Além disso, foi necessário reservar pinos específicos do controlador, por serem inerentes à interrupções implementadas em *hardware*.

4. A Arquitetura de software e o Desenvolvimento da Biblioteca

Junto ao particionamento de *hardware*, em que foram levantados os componentes do robô, uma arquitetura de *software* foi estabelecida. Esta esclarece a maneira como o controle é realizado e as funcionalidades disponibilizadas aos usuários. Como o controlador utilizado é a placa de prototipação Arduino Mega 2560, baseado no microcontrolador AT-Mega 2560, então a biblioteca foi desenvolvida conforme especificações e orientações disponíveis para a plataforma [Arduino 2017], utilizando a linguagem de programação C++ e o paradigma de orientação a objeto para alguns componentes. A Figura 3 exibe o diagrama de caso de uso, criado com base nas funcionalidades citadas acima.

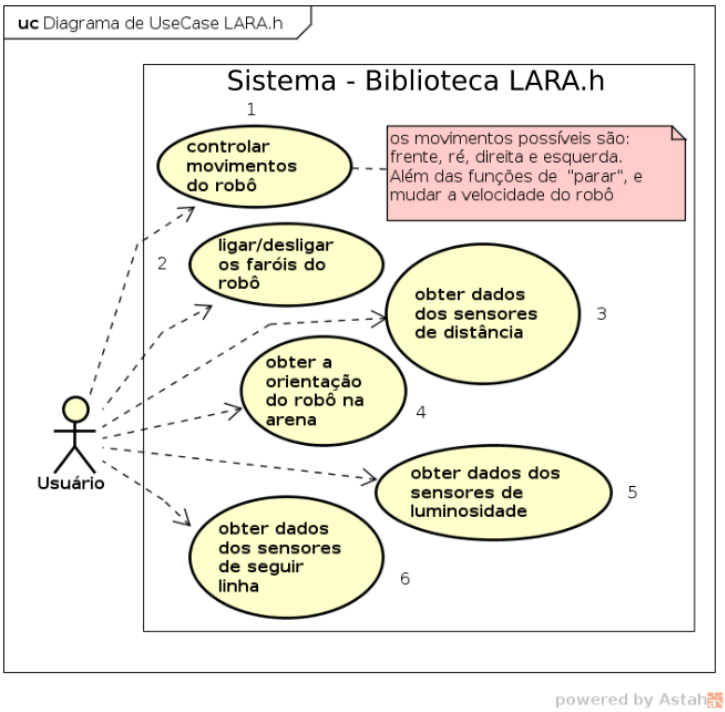

Figura 3. Diagrama de Use-Case Simplificado

[2]A documentação da biblioteca está disponível no link: `https://github.com/helscream/HMC5883L_Header_Arduino_Auto_calibration`

[3]A documentação da bibliotéca está no lilk: `https://www.pjrc.com/teensy/td_libs_Encoder.html`

Basicamente, o controle do L1R2 pode ser dividido em quatro grupos: controle de movimentos; controle dos sensores; controle estratégico; e, controle de atuadores. No primeiro grupo, que atende ao caso de uso 1, estão os métodos que permitem a manipulação dos motores do robô, fazendo-o realizar as manobras de "frente", "ré", "direita", "esquerda" e "parar", além de mudar sua velocidade. Por sua vez, no segundo grupo encontram-se as funções responsáveis pelas leituras de todos os sensores, atendendo aos casos de 3 a 6. Já no terceiro foram utilizadas as funções que obtém dados dos sensores, para inferir algumas informações, como o sensor de distância (Sensor Ultrassom) que está mais próximo ou distante de um obstáculo, ou o sensor de infravermelho (IR) que está sobre a linha marcada na arena, etc. Por fim, no quarto grupo são reunidos os métodos responsáveis pelo controle dos atuadores do robô, com exceção dos motores, atendendo principalmente ao caso de uso 2.

Dentre os grupos que compõem a biblioteca, o "controle de movimentos", e "controle dos sensores" são os que mais influenciam nas implementações de *software* e *hardware*, tendo em vista que atendem ao maior número de requisitos e funcionalidades do sistema robótico. Como cada sensor ou atuador possui uma forma específica de utilização, optou-se por modularizar as funções, dividindo-as em classes diferentes. A Figura 4 apresenta o diagrama de componentes da biblioteca.

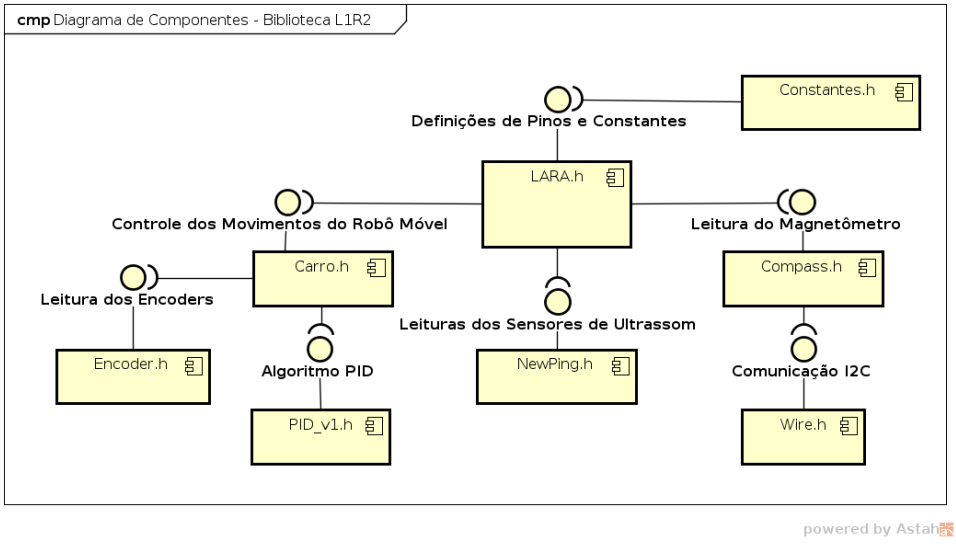

Figura 4. Diagrama de Componentes do LARA.h

A interface principal é a LARA.h, disponibilizada aos usuários do experimento, e que contém a definição de todas as funções públicas para o controle do robô. Essa utiliza as classes "Carro", "Compass" e "NewPing" para, respectivamente, controle dos movimentos do L1R2, leituras da bússola e leituras dos sensores de distância.

A "Compass", por sua vez, inclui a biblioteca "Wire.h" para comunicação pelo protocolo I^2C, e a classe "Carro" usa a "Encoder.h" para leitura dos encoders, e a "PID_v1.h" que contém a implementação do algoritmo de P.I.D. (Proporcional, Integral e Derivativo), utilizado juntamente com as equações odométricas do veículo para controlar os movimentos deste, reduzindo seus desvios laterais, que é um problema inerente à sua topologia diferencial [Muniandy and Muthusamy 2012]. Por fim, o arquivo de cabeçalho

"Constantes.h" possui a definição de algumas constantes que são utilizadas por mais de uma classe, como numeração de pinagem dos sensores e atuadores, além das constantes que são utilizadas pelos usuários, para referenciar, por exemplo, o identificador de um sensor.

De fato, o usuário tem acesso às funções e métodos definidos na classe LARA. Esta possui protótipos redefinidos no arquivo "Arduino.h", e é inicializada em "main.cpp", que pertencem ao ambiente de desenvolvimento da placa arduino, evitando a necessidade dos usuários do experimento incluírem a biblioteca para controle do L1R2, além de abstrair os conceitos quanto à utilização de objetos para os alunos dos semestres iniciais, uma vez que esse paradigma pode ainda não ter sido ensinado.

A Tabela 1 mostra algumas funções da classe LARA. As funções de 1 a 6 são responsáveis por ler os dados dos sensores, bastando passar como argumento seu índice (constantes criadas para demarcar a posição do componente no chassi do robô, por exemplo, DIREITA e ESQUERDA). Por sua vez, as de 7 a 9 pertencem ao grupo "controle estratégico". Já a 10 e a 11 são responsáveis por controlar os faróis do robô, sendo capazes também de mudar suas cores. Por fim, as funções de 12 a 19 controlam os movimentos do L1R2. Os movimentos laterais, direita e esquerda, recebem como argumento o ângulo de giro pretendido, e os movimentos longitudinais, frente e ré, podem ser definidas por um tempo ou distância. Algumas funções, como as de movimento, possuem declaradas em seus protótipos argumentos defaults, dispensando a passagem de parâmetros em suas chamadas.

Tabela 1. Algumas funções da biblioteca LARA.h

	Função
1	leituraIR(indexIR : int, type : int) : int
2	leituraSensorLuz(indexLDR : int) : int
3	distancia(indexUltrassom : int type : int) : int
4	leituraAnguloNorteMagnetico() : float
5	leituraAnguloReferencia() : float
6	getLeiturasIR(irDireita : int* , irEsquerda : int*, ir Centro : int*) : void
7	getLinha() : int
8	getMinDistancia() : int
9	getMaxDistancia() : int
10	ligarFarol(cor : Cor, led : int) : void
11	desligarFarol(led : int) : void
12	frente(tempo : long&, velocidade : int) : void
13	frente(distancia : int, type : int, velocidade : int) : void
14	re(tempo : long&, velocidade : int) void
15	re(distancia : int, type : int, velocidade : int) : void
16	direita(angulo : int) : void
17	esquerda(angulo : int) : void
18	parar() : void
19	setVelocidade(velocidade : int) : void

5. Análise e Discussão do resultado

O objetivo da biblioteca LARA.h é o controle do robô móvel, L1R2, abstraindo conceitos de programação de sistemas embarcados e detalhes de *hardware*, tornando mais simples o processo inicial de aprendizagem de programação.

Para avaliar a eficácia do proposto, foi realizado um curso semipresencial, com alunos do primeiro semestre do curso de Ciência da Computação, no primeiro período letivo de 2016. Ao todo foram 20 estudantes, sendo que destes: 3 do sexo feminino e 17 do sexo masculino; faixa etária dos 17 aos 24 anos; sendo que 5 alunos declararam ter alguma experiência com programação e nenhum participante tinha experiência com robótica. Durante o curso, os alunos resolviam problemas de programação usando o L1R2. Após a análise do problema (levantamento de requisitos, definição de processos, estruturação do algoritmo), estes deveriam implementar uma solução dentro do ambiente LARA usando a biblioteca e testando o código no robô.

A biblioteca foi avaliada por meio de dados obtidos da base de dados do sistema, que permitiu calcular a taxa de acerto e erro dos códigos enviados ao robô. Assim, foi possível definir quais os erros mais recorrentes e analisar a influência do uso da biblioteca nestes. A tabela 2 apresenta os dados [Lopes et al. 2016].

Tabela 2. Relatórios de Códigos do LARA

Códigos	Qtd.
Total de Códigos Enviados	479
Códigos compilados	131
Códigos enviados ao robô	348
Arquivos vazios	40
Códigos com erros	97
Falta de ponto e vírgula (;)	36
Nome de comando errado	12
Erros com {	13
Outros erros	32

Ao todo foram 479 códigos enviados pelos discentes, dentre os quais 97 possuíam erros sintáticos, ou seja, aproximadamente 20% do total continham erros na escrita do código. Estes erros foram classificados quanto sua natureza, analisando os códigos presentes nos arquivos. O mais comum foi a ausência no uso do ';', e isso pode ser explicado pela falta de experiência em programação dos alunos, uma vez que 75% dos participantes declararam não possuir nenhuma experiência.

Por sua vez, o fato mais importante para a análise da biblioteca é que aproximadamente 12.4% dos erros foram ocasionados pela escrita ou uso incorreto das funções da biblioteca LARA.h. Isso mostra que houve uma boa compreensão e assimilação das funções contidas na biblioteca.

6. Conclusão

O ensino de programação tem se tornado cada vez mais essencial para a sociedade moderna, entretanto, programar exige um nível de abstração e rigor lógico-matemático muito

alto. Por conta disso, diversas ferramentas são propostas para auxiliar esse processo, tornando-o mais prazeroso e eficaz. Entre elas, encontra-se a robótica, que explora a interdisciplinaridade da área para desafiar os alunos, cativando suas atenções às atividades.

O LARA é um laboratório remoto que oferece experimentos de robótica para amparar o ensino/aprendizagem de programação. O projeto conta com um experimento de robótica móvel, onde os usuários programam o robô L1R2. Entretanto, como o experimento pode ser utilizado por alunos sem experiência em programação, implementou-se uma biblioteca de controle para facilitar o uso do robô, abstraindo detalhes de implementação e programação de seus componentes físicos. A biblioteca "LARA.h", implementada em C++, utilizou os conceitos orientação a objetos e das boas práticas em programação, para modularizar suas funcionalidades, tornando sua implementação mais fácil, e contribuindo positivamente para sua manutenibilidade.

Como o desenvolvimento do L1R2 segue a metodologia *Codesign*, para projetos de sistemas heterogêneos, compostos por uma parte de *hardware* e *software*, os conflitos existentes entre os componentes do robô com a biblioteca de controle foram previamente encontrados, analisados e resolvidos. É o caso de F07, que exigiu a utilização de bibliotecas adicionais para corrigir os desvios laterais, que, por sua vez, são inerentes à topologia diferencial adotada no robô. Além do uso da bússola, que possui um protocolo específico para comunicação com o microcontrolador utilizado. Estas ocorrências tornam evidentes a necessidade de análises de *hardware* para se desenvolver um sistema de controle, assim como preconiza a metodologia adotada.

Para avaliar a biblioteca quanto a seu uso, realizou-se um curso, em que foram analisados a assimilação das funções implementadas. Dentre 97 códigos que continham erros e foram submetidos ao laboratório, apenas 12 continham erros provenientes de escrita ou uso da biblioteca, ou seja, apenas 12.4% de todos os erros. Isso indica que as funções criadas foram facilmente assimiladas, simplificando a programação e consequentemente a manipulação do L1R2.

Assim, no que diz respeito aos controles básicos do robô como leitura dos sensores, controle dos movimentos e controle de outros atuadores, a "LARA.h" mostrou-se eficaz e suficiente. Todavia, almeja-se agora que a biblioteca permita uma manipulação mais elaborada do robô, como o uso de processamento de imagem. Além disso, o LARA possui pretensões de adicionar novos robôs móveis ao laboratório, assim, será necessário modificar a biblioteca para que sua adaptação a estes não exijam muita re-implementação de código. Uma outra funcionalidade será adicionada a biblioteca é a possibilidade de ensino de Programação Orientada a Objeto em cursos oferecidos no ambiente.

Referências

Alimisis, D. (2013). Educational robotics: Open questions and new challenges. *Themes in Science and Technology Education*, 6(1):63–71. Acesso em: 19 jul. 2017.

Arduino (2017). Arduino style guide for writing libraries. Acesso em: 12 jul. 2017.

Barros, E. and Cavalcante, S. (2010). Introdução aos sistemas embarcados. *Artigo apresentado na Universidade Federal de Pernambuco-UFPE*, page 36. Acesso em: 19 jul. 2017.

Benitti, F. B. V. (2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education*, 58(3):978–988. Acesso em: 19 jul. 2017.

Borenstein, J., Everett, H., Feng, L., et al. (1996). *Where am I? Sensors and methods for mobile robot positioning*. University of Michigan. Acesso em: 07 set. 2016.

dos Santos, R. P. and Costa, H. A. X. (2006). Análise de metodologias e ambientes de ensino para algoritmos, estruturas de dados e programação aos iniciantes em computação e informática. *INFOCOMP Journal of Computer Science*, 5(1):41–50. Acesso em: 9 jul. 2017.

Kalúz, M., Fikar, M., et al. (2013). Simplifying the implementation of remote laboratories in educational environments using industrial hardware. In *Process Control (PC), 2013 International Conference on*, pages 522–527. IEEE. Acesso em: 9 jul. 2017.

Lopes, M., Gomes, I., Trindade, R., Silva, A., and Lima, A. C. (2016). Web environment for programming and control of mobile robot in a remote laboratory. *IEEE Transactions on Learning Technologies*. Acesso em: 9 jul. 2017.

Martins, L. E. G., de Souza Júnior, R., de Oliveira Jr, H. P., and Peixoto, C. S. A. (2010). Terase: Template para especificação de requisitos de ambiente em sistemas embarcados. In *WER*. Acesso em: 17 jul. 2017.

Muniandy, M. and Muthusamy, K. (2012). An innovative design to improve systematic odometry error in non-holonomic wheeled mobile robots. *International Symposium on Robotics and Intelligent Sensors 2012 (IRIS 2012)*, 41:436 – 442. Acesso em: 17 jul. 2017.

Niku, S. B. (2013). *Introdução à robótica–análise, controle, aplicações*. LTC, São Paulo, 1st edition edition.

Özmen, B. and Altun, A. (2014). Undergraduate students' experiences in programming: Difficulties and obstacles. *Turkish Online Journal of Qualitative Inquiry*, 5(3):1–27. Acesso em: 9 jul. 2017.

Thomas, D. E., Adams, J. K., and Schmit, H. (1993). A model and methodology for hardware-software codesign. *IEEE Design & test of computers*, 10(3):6–15. Acesso em: 9 jul. 2017.

Zendoia, J., Zapp, M., Agyapong-Kodua, K., Lohse, N., and Singh, M. (2013). Fundamentals of a co-design methodology for improving the performance of machine tools based on semantic representation. *International Journal of Computer Integrated Manufacturing*, 26(8):751–761. Acesso em: 07 set. 2016.