UFBA

UNIVERSIDADE FEDERAL DA BAHIA
ESCOLA POLITÉCNICA
PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA INDUSTRIAL - PEI

DOUTORADO EM ENGENHARIA INDUSTRIAL

MAÍSA SOARES DOS SANTOS LOPES

Ambiente colaborativo para ensino aprendizagem de programação integrando laboratório remoto de robótica

PEI
Programa de Pós-Graduação em Engenharia Industrial

SALVADOR
2017

Maísa Soares dos Santos Lopes

Ambiente colaborativo para ensino aprendizagem de programação integrando laboratório remoto de robótica

Trabalho de Tese apresentado ao Programa de Pós-Graduação em Engenharia Industrial, da Universidade Federal da Bahia, como requisito parcial para a obtenção do título de Doutor em Engenharia Industrial.

Orientadores:
Prof. Dr. Antônio Cezar de Castro Lima
Universidade Federal da Bahia (PEI - UFBA)

Profa. Dra. Alzira Ferreira da Silva
Universidade Estadual do Sudoeste da Bahia (UESB)

Salvador - BA
2017

Lopes, Maísa Soares dos Santos
Ambiente colaborativo para ensino aprendizagem de programação integrando laboratório remoto de robótica / Maísa Soares dos Santos Lopes. -- Salvador, 2017.
103 f.

Orientador: Antônio Cezar de Castro Lima.
Coorientadora: Alzira Ferreira da Silva.
Tese (Doutorado - Programa de Pós-Graduação em Engenharia Industrial) -- Universidade Federal da Bahia, Escola Politécnica, 2017.

1. Programação de computadores. 2. Laboratório remoto. 3. Aprendizagem colaborativa suportada por computador. 4. Robótica móvel. I. Lima, Antônio Cezar de Castro. II. Silva, Alzira Ferreira da . III. Título.

"Ambiente colaborativo para ensino aprendizagem de programação integrando laboratório remoto de robótica"

MAÍSA SOARES DOS SANTOS LOPES

Tese submetida ao corpo docente do Programa de Pós-graduação em Engenharia Industrial da Universidade Federal da Bahia como parte dos requisitos necessários para a obtenção do grau de doutor em Engenharia Industrial.

Examinada por:

Prof. Dr. Marcio Luis Ferreira Nascimento_____
Doutor em Ciência e Engenharia dos Materiais pela Universidade Federal de São Carlos, Brasil, 2004;

Prof. Dr. Luiz Marcos Garcia Goncalves_____
Doutor em Engenharia de Sistemas e Computação pela Universidade Federal do Rio de Janeiro, Brasil, 1999;

Prof. Dr. Roque Mendes Prado Trindade_____
Doutor em Engenharia Elétrica e de Computação pela Universidade Federal do Rio Grande do Norte, Brasil, 2009;

Prof. Dr. Robson da Silva Magalhães_____
Doutor em Engenharia Industrial pela Universidade Federal da Bahia, Brasil, 2010;

Prof. Dr. Márcio Fontana_____
Doutor em Engenharia Elétrica pela Universidade Estadual de Campinas, Brasil, 2004.

Salvador, BA - BRASIL
Junho/2017

Dedicatória

A Hélio, Sofia e Artur que me ensinam que amar é uma decisão diária.

Agradecimentos

Durante o processo de construção deste trabalho foram muitos desafios, lutas, novos conhecimentos, realizações e alegrias. Chegar até aqui só foi possível graças ao auxílio de muitas pessoas. Agradeço de todo meu coração:

A Deus, por ser o meu Pastor fiel e misericordioso.

À toda minha família, por ser abrigo seguro.

A Cezar, por aceitar orientador este trabalho e acreditar que era possível desenvolvê-lo em Vitória da Conquista.

À Alzira, pela orientação, amizade e confiança dispensada a mim durante toda a caminhada.

Ao prof. Ivanor que plantou a semente deste trabalho.

A Roque, pela contribuição e otimismo que nos faz acreditar que somos capazes.

Aos alunos que fizeram e fazem parte do LARA, fundamentais na construção deste trabalho: Emerson, Gustavo, Helber, Iago, Igor Sodré, Isadora, João Victor, Jenifer, Jéssica, Luan, Lucas, Maone, Matheus Lima, Pablo, Raphael, Rodrigo, Stefanie, Thomas, Vinicius e Yuri.

Aos amigos e companheiros de doutorado Clênia, José Carlos e Agnaldo.

Às amigas Celina e Alexsandra, pelo carinho e incentivo.

À Cátia e seus alunos de Algoritmos e Programação I, pela colaboração no levantamento dos dados e na avaliação do ambiente.

Aos colegas da UESB, pelo apoio e incentivo.

À UESB, pelo apoio financeiro.

Ao PEI, por oferecer-me a possibilidade de realizar o doutorado, e aos seus professores e funcionários, pela gentileza e prestatividade.

Resumo

Programação de computadores é uma habilidade importante que os profissionais da área de Tecnologia de Informação devem possuir. Entretanto, o estudo de programação é considerado difícil e pouco atraente para a maioria dos alunos. Apesar dos vários estudos desenvolvidos na tentativa de melhorar esta situação, ainda existe necessidade de recursos educacionais que motivem e estimulem os alunos. Esta tese tem como objetivo propor uma arquitetura modular para um ambiente dinâmico voltado ao ensino aprendizagem de programação, baseada na integração de aprendizagem colaborativa suportada por computador, robótica móvel e laboratório remoto. A arquitetura proposta possibilita a contextualização do ensino e a utilização de estratégia de planejamento, codificação e teste de soluções de problemas e pode ser aplicada na educação à distância ou de forma complementar ao ensino tradicional. Para verificar a viabilidade da arquitetura, optou-se pela construção de um protótipo, denominado de LaraPC, um ambiente virtual de aprendizagem otimizado com suporte à experimento de robótico remoto, programação colaborativa, interação síncrona entre os usuários e ferramenta para análise de problema. Para compor o LaraPC, foi desenvolvido o ambiente para programação e controle de um robô móvel que integra um laboratório remoto e uma ferramenta de programação online onde é possível realizar atividades práticas individualmente ou em grupo. Os testes empíricos e de inspeção serviram de validação experimental do trabalho. Os resultados mostram que o conjunto de soluções adotadas comprova a capacidade da arquitetura em atender aos requisitos identificados e que, embora a validação de utilização ainda não esteja completa, o sistema é um ambiente com potencial para envolver e motivar os alunos na aprendizagem de programação.

Palavras chave: programação de computadores, laboratório remoto, aprendizagem colaborativa suportada por computador (CSCL), robótica móvel.

Abstract

Computer programming is an important skill that professionals in information technology area should have. However, to learn programming is considered a difficult and unattractive task for most students. Despite several efforts to improve this situation, there is still need for educational resources that motivate and stimulate students. This thesis propose a modular architecture for a dynamic environment focused on programming education, based on the integration of Computer Supported Collaborative Learning, mobile robotics and remote laboratory. Moreover, it also allows: contextualization of teaching, use of planning strategies, coding and testing solutions of problems and can be applied in distance education scenarios or to enhance traditional learning-teaching. To verify the feasibility of the proposed architecture, a prototype system called LaraPC was created, a virtual learning environment optimized with remote robotic experiment support, collaborative programming and synchronous user interaction. LaraPC comprises an environment for programming and control of a mobile robot that integrates a remote laboratory and an online programming tool, which could allow students to carry out practical activities individually or in a group. The empirical and inspection tests served as experimental validation of the work. The results show that the set of adopted solutions proves the capacity of the architecture to meet the identified requirements. Although the validation of use is not yet complete, the system is an environment with potential to engage and motivate students in programming learning.

Key words: computer programming, remote laboratory, Computer Supported Collaborative Learning (CSCL), mobile robotic.

Lista de Abreviaturas e Siglas

AP I	Algoritmos e Programação I
APC	Ambiente de Programação e Controle
API	*Application Programming Interface*
AVA	Ambiente Virtual de Aprendizagem
CSCL	*Computer Supported Collaborative Learning*
EJS	*Easy Java Simulation*
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
IDE	*Integrated Development Environment*
IHC	Interação Humano-Computador
LMS	*Learning Management System*
MOODLE	*Module Object-Oriented Dynamic Learning Environment*
PC	Programação Colaborativa
SCORM	*Sharable Content Object Reference Model*
UESB	Universidade Estadual do Sudoeste da Bahia

Lista de Figuras

Lista de Tabelas

Sumário

1 Introdução

Habilidades de programação são umas das principais competências que os profissionais da área de Tecnologia de Informação (TI) devem possuir (Law et al., 2010). Eles devem acompanhar a evolução tecnológica e propor soluções computacionais nas mais diversas áreas de conhecimento. Estas habilidades são introduzidas em cursos de programação, onde são abordados os princípios de lógica de programação, raciocínio lógico, processo de resolução de problemas, representação de soluções na forma de algoritmos e linguagem de programação. Entretanto, o estudo de programação é considerado difícil e pouco atraente para a maioria dos alunos (Kay, 2011).

Pesquisas mostram que a utilização de aprendizagem colaborativa (Brito et al., 2011; Fan, 2012; Serrano-Cámara et al., 2014) e de contexto (Cooper & Cunningham, 2010; Kay, 2011; Goadrich, 2014) podem atrair e motivar o ensino introdutório de computação. Entretanto, a junção destas duas abordagens em um único ambiente, que possa ser utilizado na educação presencial e a distância, ainda é pouco explorada.

Ambientes de aprendizagem colaborativa incentivam a construção do conhecimento, compreensão mais profunda e maior desenvolvimento de competências. A aprendizagem colaborativa suportada por computador (*Computer Supported Collaborative Learning* – CSCL) estuda como alunos podem aprender juntos com a ajuda das tecnologias digitais. Segundo Stahl, Koschmann & Suthers (2006),

> *CSCL impõe a colaboração entre os alunos; eles não estão simplesmente reagindo isoladamente a conteúdos publicados. Neste caso, a aprendizagem acontece através das interações entre os alunos. Eles aprendem através das suas perguntas, perseguindo conjuntamente linhas de raciocínio, ensinando um ao outro e vendo como os outros estão aprendendo. Suporte computacional para este tipo de colaboração é crucial para uma abordagem CSCL em e-learning*

Em programação de computadores, as ferramentas CSCL permitem que dois ou mais estudantes acessem e editem o mesmo código de programa e troquem ideias e soluções para os problemas. Essa prática é chamada de programação colaborativa (PC). Diversos trabalhos apontam os benefícios do CSCL no ensino de programação: aumento da motivação, aceleração no progresso da resolução de problemas, melhoria da qualidade do produto de software desenvolvido, entre outros (Fan & Sun, 2012a; Bravo, Duque & Gallardo, 2013, Serrano-Cámara et al., 2014).

Já a abordagem contextualizada propõe explicar a utilidade do que está sendo ensinado. Os alunos tendem a ficar por perto se eles entendem o valor do que eles estão aprendendo (Guzdial, 2010). Ferramentas como animação, mídias digitais, robótica, entre

outras podem ser utilizadas no ensino contextualizado de programação (Cooper & Cunningham, 2010). Entretanto, a ferramenta mais popular é o robô (Kay, 2011).

No ensino de programação de computadores, a robótica é vista como uma ferramenta de apoio a aprendizagem que estimula e motiva o interesse dos alunos (Anderson et al., 2011), além de minimizar as dificuldades relacionadas ao alto grau de abstração do conteúdo e a baixa capacidade dos alunos em resolver problemas (Rocha, 2006; Oliver et al., 2010; Ribeiro et al., 2011; Ressurreição, 2012; McGill, 2012; Celestino, 2013; Ferreira, 2013; Thomaz, 2013). Benitti (2012) considera a possibilidade de utilizar a robótica como ferramenta para o desenvolvimento de habilidades como raciocínio lógico, resolução de problemas e trabalho em equipe. Habilidades essenciais ao profissional de TI.

A robótica tem sido aplicada no ensino de programação em experimentos presenciais onde os alunos tem que estar fisicamente no laboratório e este geralmente tem restrições de horário para acesso. Uma solução para estas limitações de tempo e espaço é o uso de laboratórios remotos. Este tipo de laboratório permite que experimentos reais sejam acessados a qualquer hora e em qualquer lugar via internet (Ma & Nickerson, 2006; Tawfik et.al, 2013).

A maioria dos laboratórios remotos de robótica não é voltada para o ensino e aprendizagem de programação. Além de não permitir interação síncrona multiusuário, característica essencial em um ambiente de aprendizagem colaborativa, e não estar integrada a Ambientes Virtuais de Aprendizagem (AVA) (Bochicchio & Longo, 2010), falta suporte à ferramentas de programação ou ambientes de desenvolvimento integrado (*Integrated Development Environment* - IDE).

As IDEs possibilitam a construção e execução de programas, elas são geralmente desenvolvidas para atender às necessidades dos programadores profissionais e possuem vários conceitos e características difíceis de serem assimilados por programadores iniciantes (Pears et al., 2007). Além disto, a maioria das IDEs não possui recursos para programação colaborativa, o foco é na codificação de programas e não oferecem suporte ao planejamento de soluções.

Os estudantes novatos tendem a ignorar as fases de análise e projeto, desenvolvendo programas diretamente no computador usando tentativa e erro para encontrar uma solução e não entendem de fato o que estão fazendo (Tan et al., 2014). Estas fases são importantes pois ajudam a fixar conceitos e desenvolver habilidades de programação uma vez que os alunos são estimulados a pensar sobre o problema e propor soluções adequadas que devem ser implementadas, testadas, validadas e, se necessário, corrigidas e aprimoradas.

Considerando a importância da programação de computadores e as dificuldades inerentes ao seu ensino aprendizagem, torna-se relevante investigar e desenvolver recursos didáticos que possam contribuir para melhorar este processo, principalmente porque o insucesso nesta disciplina é apontado como uma das causas do elevado índice de evasão dos cursos de computação (Zanetti & Bonacin, 2014).

O foco desta tese é a colaboração multiusuário no ensino contextualizado de programação devido a problemática apresentada anteriormente. Este trabalho é parte do projeto LARA (Laboratório Remoto em AVA) que tem como objetivo projetar e implementar uma arquitetura pedagógica que integra recursos tecnológicos e metodologia de ensino para melhorar o processo de aprendizagem de disciplinas do curso de Ciência da Computação. O LARA envolve aspectos relacionados a laboratórios remotos, engenharia de software, robótica e metodologia pedagógica, como mostra a Figura 1.

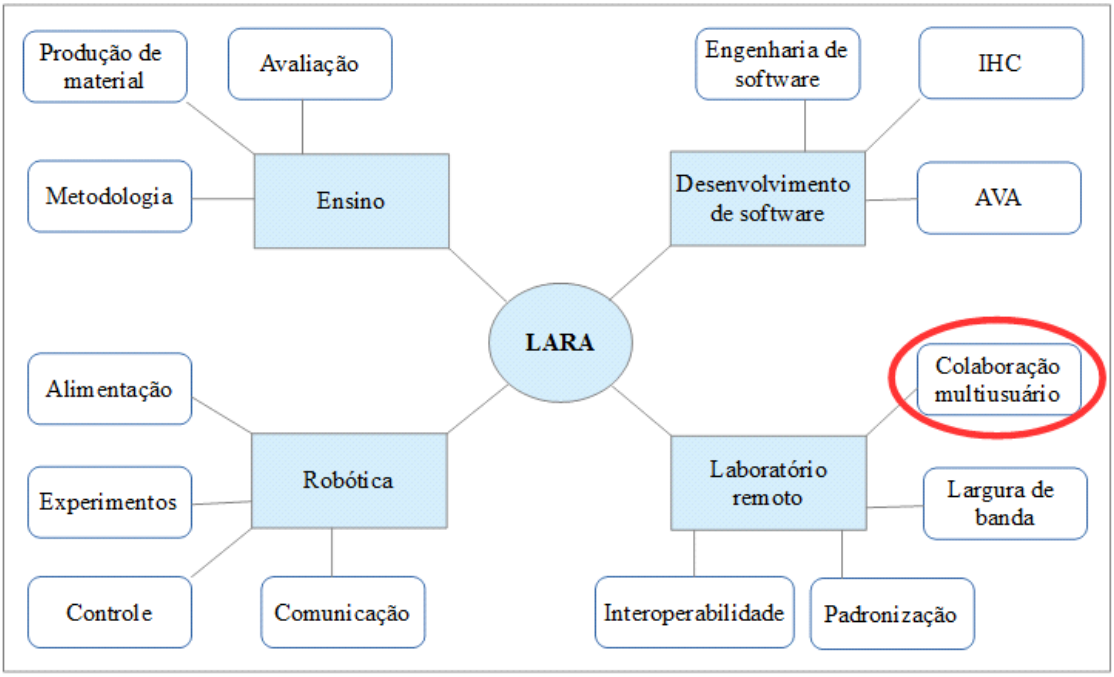


Figura 1 – Aspectos envolvidos no desenvolvimento do LARA

Sendo assim, este trabalho busca investigar se é possível aumentar o interesse do aluno no aprendizado de programação através do uso de uma arquitetura de um ambiente colaborativo para o ensino de programação utilizando laboratório remoto de robótica móvel. Surgem então as seguintes questões: Como utilizar laboratório remoto para incentivar o estudo de programação? Qual a arquitetura de um laboratório remoto para o ensino colaborativo de programação? Quais as ferramentas necessárias para um sistema colaborativo para ensino de programação usando laboratório remoto? Quais as

ferramentas necessárias em um laboratório remoto para suportar o ensino de programação que envolva as etapas de planejamento, codificação e depuração?

Como ferramenta colaborativa, a arquitetura deste sistema tem que considerar os aspectos tecnológicos referentes a infraestrutura como módulos, protocolos, sincronismo, gerenciamento de sessão, e os aspectos colaborativos como características, composição e dinamismo do trabalho em grupo. A dinâmica de trabalho, os serviços selecionados e a composição do ambiente de trabalho são fundamentais para proporcionar a colaboração, além de serem fatores especialmente críticos no contexto da aprendizagem colaborativa (Gerosa et al., 2004; Gadelha, 2011).

Segundo Sommerville (2003), a arquitetura de software estabelece a estrutura básica para um sistema e envolve a identificação dos seus componentes principais e a comunicação entre eles. Entretanto, "a integração de componentes não deve apenas ser feita a partir de um ponto de vista arquitetônico, mas também do ponto de vista dos elementos que constituem a interface gráfica do usuário (GUI), já que o sistema pode responder às interações do usuário e dar o *feedback* da forma mais adequada" (McCalla *apud* Jurado et al., 2012).

Assim, a arquitetura proposta considera a questão estrutural (módulos, protocolo e interface de comunicação entre os módulos), o processo de colaboração multiusuário e a usabilidade da interface do usuário.

1.1 Objetivos

Objetivo geral

Desenvolver uma arquitetura de um ambiente colaborativo para o ensino de programação utilizando laboratório remoto de robótica móvel

Objetivos Específicos

- Definir os componentes da arquitetura;
- Identificar as interfaces de comunicação entre os vários módulos da arquitetura;
- Definir métodos que possibilitem o desenvolvimento de ferramentas colaborativas síncronas para um laboratório remoto para o ensino de programação;
- Elaborar uma interface amigável que integre várias ferramentas colaborativas;
- Validar a utilização de laboratório remoto no ensino de programação.

1.2 Método da pesquisa

O tema explorado nesta tese é a organização de um sistema CSCL utilizando laboratório remoto de robótica para contextualização da aprendizagem de programação de computadores.

O presente trabalho é uma pesquisa aplicada, uma vez que tem por objetivo gerar conhecimentos para aplicação prática, dirigida à solução de problemas reais específicos (Kauark et al., 2010). Do ponto de vista da forma de abordagem do problema, é mista, baseada em dados qualitativos e quantitativos. Quanto a sua finalidade, é uma pesquisa exploratória, pois o objetivo é proporcionar a utilização de laboratórios remotos para contextualizar o ensino de programação em um ambiente de aprendizagem colaborativa (Gil, 2002; Filippo et al., 2012). Com relação aos procedimentos técnicos, pode ser considerada de caráter experimental, pois busca a obtenção de conhecimento e a descoberta de produtos (protótipos), processos e novos serviços.

O desenvolvimento da tese seguiu as seguintes etapas:

- Revisão bibliográfica – foram utilizados documentos científicos como fontes de informações, indicações e esclarecimentos, que ajudaram a elucidar questões da pesquisa e conhecer as contribuições já implementadas em ambientes colaborativos para ensino de programação e experimentação remota.
- Análise de ferramentas – foram analisadas ferramentas de programação colaborativa, laboratórios remotos e AVA para identificar os serviços que serviram como referência no desenvolvimento da arquitetura proposta.
- Criação de uma prova de conceito – para a verificação da viabilidade da aplicação da arquitetura, optou-se por desenvolver um protótipo como prova de conceito com a finalidade de verificar se a arquitetura proposta é passível de ser implementada.
- Validação – para validação do protótipo, foram utilizados testes de usabilidade da interface do usuário.

Devido à complexidade inerente ao desenvolvimento de sistemas colaborativos, optou-se por projetar uma arquitetura conceitual baseada em módulos que se inter-relacionam e possibilitam sua modificação e reutilização. Para o desenvolvimento do sistema foi utilizado a tecnologia de componentes que favorece a prototipação, a experimentação e o desenvolvimento iterativo (Fuks & Pimentel, 2011).

A Figura 2 mostra a abordagem de desenvolvimento utilizada neste trabalho. A primeira etapa é a análise que busca identificar, coletar e organizar informações relevantes do domínio. O projeto de aplicação visa especificar a arquitetura do sistema com base nos

requisitos levantados na análise. Então são selecionados componentes (ferramentas de software) adequados ao protótipo. Depois, os componentes selecionados são adaptados em função da arquitetura e são submetidos a teste para verificar se estão de acordo com as funcionalidades desejadas. Caso não sejam encontradas ferramentas adequadas, novos componentes são desenvolvidos para atender a arquitetura. A etapa seguinte é a integração dos componentes utilizando intercambio e armazenamento de dados e mecanismos de acoplamento de componentes. Por fim, o protótipo é testado (Fuks & Pimentel, 2011).

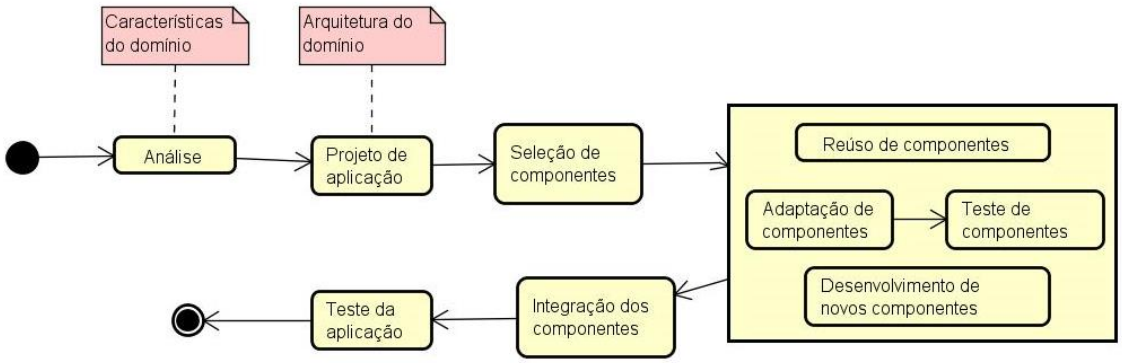


Figura 2 – Processo de desenvolvimento baseado em componentes
(Adaptado de Fuks & Pimentel (2011), pag. 368)

Considerando a importância da usabilidades para os sistemas CSCL, além de testar as funcionalidades dos componentes, foram testadas as interfaces do usuário através de método empírico, com participação do usuário, e teste de inspeção, onde é verificada a conformidade dos artefatos de software com um conjunto de diretrizes de usabilidade (Fernandez et al., 2011).

1.3 Organização do trabalho

Este trabalho está dividido em sete capítulos, contando com esta introdução. Inicialmente é apresentada uma revisão bibliográfica e estado da arte sobre ensino/aprendizagem de programação de computadores e suas dificuldades, a aprendizagem colaborativo suportada por computador e sua aplicação em programação, em seguida são apresentadas características dos laboratórios remotos, seus desafios educacionais e exemplos de laboratórios remotos de robótica móvel e o seu suporte a programação. O Capítulo 3 apresenta a arquitetura proposta para o sistema. O capítulo 4 apresenta as ferramentas selecionadas para compor o protótipo da arquitetura e o desenvolvimento do laboratório remoto de robótica móvel integrado a um ambiente de programação. O capítulo 5 descreve a integração das ferramentas selecionadas e interface

do protótipo. O capítulo 6 descreve a avaliação do laboratório remoto desenvolvido neste trabalho e a avaliação do protótipo da arquitetura. No capítulo 7, são apresentadas as conclusões e trabalhos futuros.

2 Referencial teórico

Este capítulo apresenta os conceitos que fundamentam essa pesquisa e seu estado da arte. Por se tratar de um projeto multidisciplinar, abrange algumas definições muito amplas que aqui são simplificadas para atender o escopo do trabalho. Inicialmente são apresentadas a importância e as dificuldades que envolvem o ensino de programação de computadores. A seção 2.2 conceitua aprendizagem colaborativa suportada por computador. Em seguida, é mostrada a aplicação de CSCL em programação e são apontados benefícios e exemplos de ferramentas. Na seção 2.4 são discutidas as características dos laboratórios remotos e as dificuldades educacionais enfrentadas por esse tipo de laboratório. A seção 2.5 apresenta laboratórios remotos de robótica móvel com suporte à programação.

2.1 Ensino de programação de computadores

O objetivo do ensino de programação é capacitar os alunos a desenvolverem soluções computacionais para resolver problemas do mundo real. No currículo de referência da Sociedade Brasileira de Computação (SBC) para os cursos de Ciência da Computação e Engenharia de Computação (SBC, 2005), programação é uma matéria do núcleo de fundamentos da computação que envolve a parte científica e as técnicas fundamentais à formação sólida dos egressos. O guia de referência da ACM e IEEE para cursos de Ciência da Computação (CS, 2013, p. 44) enfatiza que os alunos não devem perceber a computação apenas como aprender as especificidades de linguagens de programação, cuidados devem ser tomados para enfatizar os conceitos mais gerais da computação (abstração, decomposição, etc) no contexto de aprender a programar.

Portanto, programação não se restringe ao estudo de linguagem de programação, mas envolve um conjunto de princípios, técnicas e formalismos que visam o desenvolvimento de programas de qualidade (Santos & Costa, 2006). Segundo Koorsse et al. (2015), para desenvolver programas de computador são necessárias três etapas:

1. Entender o problema para formular uma solução – é necessário estudar a declaração do problema, o conjunto de requisitos e decidir sobre a melhor estratégia de programação a ser usada.
2. Produzir um algoritmo para resolver o problema – é necessário conhecer os princípios e conceitos de programação para selecioná-los e aplicá-los corretamente.
3. Traduzir o algoritmo para uma linguagem de programação – isto exige o conhecimento da sintaxe da linguagem de programação. O código do programa é testado e alterado até que o programa atenda ao conjunto original de requisitos e, assim, resolva o problema.

Os alunos devem, então, aprender o processo de resolução de problemas, desenvolver o raciocínio lógico e aprender linguagem de programação (Nandigam & Bathula, 2013). Este é um processo considerado difícil, especialmente para alunos iniciantes (Wang et al., 2011). Gomes et.al (2008) identificam um conjunto de fatores que complicam o aprendizado de programação: os métodos de ensino não são de fato centrado no aluno, as turmas grandes não permitem um ensino personalizado com *feedback* e supervisão adequados às necessidades de cada estudante; os conteúdos de programação são dinâmicos e abstratos, além disso, é comum começar ensinar os detalhes sintáticos de uma linguagem de programação antes que os alunos percebam qual a finalidade e utilidade de aprender programar; e os alunos estão habituados com o modelo de estudo baseadas em leituras, memorização de fórmulas e uma certa mecanização de procedimentos, mas programação exige intenso trabalho prático extraclasse, uma verdadeira compreensão dos assuntos e reflexão.

As dificuldades vão desde a natureza do conteúdo, conceitos abstratos e rigor da sintaxe das linguagens de programação, passando pelas dificuldades dos alunos, baixo nível de abstração e falta de habilidades para resolver problemas, até a inadequação dos métodos pedagógicos aplicados pelos docentes (Jenkins, 2002). Souza et al. (2016) enfatizam que os alunos tem dificuldades para aprender os conceitos de programação, para aplicar estes conceitos durante a construção de programas e falta motivação para realizar as atividades de programação. A maior parte do tempo das disciplinas é gasta ensinando a sintaxe da linguagem de programação e faltam sessões práticas *hand on* (Husain et al., 2013).

Várias abordagens tem sido aplicadas no intuito de amenizar tais problemas e dificuldades: aumento das atividades práticas *hand-on* (Wu et al., 2014) , utilização de método de caso (*case teaching*), visualização de programas e algoritmos usando, por exemplo, Scratch, iVProg e Alice (Ribeiro, Brandão & Brandão, 2012; Brandão et.al, 2012; Maloney, 2010; Andujar et al., 2013), utilização de *serious games* (Kazimoglu et.al, 2012; Piteira & Haddad, 2011; Muratet et al., 2009) e desenvolvimento de ambientes pedagógicos (Bini & Koscianski, 2009; Moreira & Favero, 2009; Verdú et.al, 2012; Chaves et.al, 2013; França & Soares, 2011). Além destas, estudos apontam a utilização de CSCL e robótica como estratégias para motivar e melhorar o ensino de programação.

2.2 Aprendizagem colaborativa suportada por computador

A aprendizagem colaborativa refere-se a situação em que duas ou mais pessoas aprendem ou tentam aprender algo juntas: conhecimentos, habilidades, competências, etc. Ela é definida por um conjunto de processos, que ajudam os alunos a interagirem em conjunto para realizar um objetivo específico (Serrano-Cámara et.al, 2014).

Com o avanço tecnológico surge a Aprendizagem Colaborativa Suportada por Computador (*Computer Supported Collaborative Learning* – CSCL). CSCL é uma área que estuda como a tecnologia pode apoiar os processos de aprendizagem através de esforços colaborativos entre estudantes trabalhando em uma determinada tarefa (Santoro, Borges & Santos, 1999). A CSCL propõe o desenvolvimento de softwares que propiciem a aprendizagem em grupo e que ofereçam atividades criativas de exploração intelectual e interação social (Stahl, Koschmann & Suthers, 2006).

Segundo Kumar et al. (2010), a promessa desses ambientes colaborativos é "promover oportunidade para facilitar a aprendizagem em contexto relativamente realistas, cognitivamente motivadores e socialmente enriquecidos". Para Soller (2002), os softwares CSCL devem oferecer ambiente semelhante a sala de aula, com espaços de trabalho compartilhados, apresentações on-line, notas de aula, material de referência, testes, notas de avaliação do estudante e bate-papo. Nestes espaços os alunos podem discutir entre si as suas estratégias de aprendizagem, as suas compreensões e as suas deficiências.

Sistemas CSCL são desenvolvidos para dar suporte ao processo de colaboração entre os alunos e, também destes com professor ou tutor. Por isto oferecem uma combinação de diversos meios de comunicação (e-mail, chat, fóruns de discussão, videoconferência, mensagem instantânea, etc), além de poder fornecer suporte pedagógico (Stahl, Koschmann & Suthers, 2006).

Sistemas de aprendizagem colaborativa ajudam a aumentar a interatividade e a acessibilidade aos vários recursos de aprendizagem de forma síncrona ou assíncrona. O desenvolvimento desses sistemas não é trivial, ele engloba tanto aspectos técnicos quanto de interação social e mistura três características importantes: onipresença – permite o acesso à informação e funcionalidades oferecidas pelo sistema em qualquer lugar e a qualquer tempo; consciência do contexto (*awareness*) – permite ao usuário compreender suas próprias atividades e também as atividades dos demais usuários no sistema; e colaboração (Almeida & Baranauskas, 2008).

2.3 CSCL em programação

A CSCL em programação de computadores, algumas vezes chamada de Programação Colaborativa (PC), é vista com a combinação de funções de interação social com funções de desenvolvimento de software e de depuração (Serrano-Cámara et.al, 2014). Como a CSCL, a programação colaborativa pode acontecer de forma síncrona ou assíncrona. A PC síncrona permite que vários usuários visualize e/ou edite o mesmo código ao mesmo tempo, programadores geograficamente distribuídos podem trabalhar em conjunto no mesmo documento e as mudanças realizadas por um programador são vistas pelos outros em tempo real (Fan, 2012; Bravo, Duque & Gallardo, 2013). Na PC assíncrona ou em tempo não

real, o usuário acessa um repositório de código, baixa o código para seu espaço de trabalho, realiza sua tarefa de programação de forma independente e atualiza suas alterações no repositório manualmente (Fan, 2012). Este tipo de trabalho pode causar sentimentos de isolamento e, consequentemente, reduz a motivação do estudante (Jara et al., 2009).

A colaboração em tempo real traz vários benefícios para o ensino aprendizagem de programação: intercâmbio de ideias entre os alunos, aumento da motivação para aprender, melhoria no processo de aprendizagem e no desempenho dos alunos, aceleração no processo de resolução de problemas, aumento na produtividade e melhoria da qualidade do código (Vizcaíno et al., 2000; Slavin, 1995; Wang & Lin, 2007; Fan & Sun, 2012a; Hwang et.al, 2012; Othman & Zain, 2015). Além disso, Serrano-Cámara et.al (2014) colocam que a programação colaborativa ajuda os alunos a aprender a dividir tarefas, a trabalhar juntos, a apoiar uns aos outros, aprender uns com os outros e a partilhar experiências para alcançar os objetivos de aprendizagem.

Além de permitir editar, compilar e executar código, um sistema de PC educacional deve possuir ferramentas para coordenação e suporte de comunicação, dispositivos de apoio a consciência, que informam sobre os usuários, os seus estados e as suas ações, e ferramentas de gestão de interação.

Serrano-Cámara et.al (2014) classificam as ferramentas CSCL para programação em três categorias: a) extensão dos instrumentos de CSCL, incorporando algumas características típicas de IDEs para suportar o trabalho de programação; b) integração de recursos CSCL em IDEs profissionais através de *plug-in*; c) sistemas CSCL com suporte a interpretação do código fonte e a visualização simultânea da sua animação. Vários trabalhos científicos sobre programação colaborativa propostos recentemente se encaixam nestas abordagens.

COLLECE (*COLLaborative Edition, Compilation and Execution of programs*) (Bravo, Duque & Gallardo, 2013) é um sistema CSCL que incorpora edição, compilação e execução de código. Ele possui ferramentas para gestão de usuário, sessão de trabalho e formulação de problemas, ferramenta de comunicação síncrona, bate-papo estruturado, ferramenta de coordenação e consciência do espaço de trabalho e ferramenta para análise de trabalho dos usuários e grupos. Entretanto, o COLLECE possui alguns inconvenientes, uma sessão de trabalho tem que ser agendada pelo professor e não permite a edição simultânea do código por vários usuários, uma ferramenta de coordenação do editor de texto define quem pode alterar o código.

RECIPE (*REal-time Collaborative Interactive Programming Environment*) (Shen & Sun, 2002) é um protótipo de sistema PC que permite que o compilador e depurador de um usuário seja convertido em aplicação colaborativa multiusuário (colaboração transparente),

além de possuir um controle de acesso flexível, onde os usuários podem solicitar e autorizar o acesso a uma sessão colaborativa. No entanto, ele não oferece ferramentas especializadas para a comunicação e não tem suporte adequado de consciência.

CoEclipse é um sistema que converte a IDE Eclipse de um único usuário em uma ferramenta de programação de colaboração multiusuário em tempo real. O grande diferencial desta ferramenta é que ela oferece recursos avançados de prevenção dos conflitos semânticos que podem ocorre durante uma sessão de programação colaborativa (Fan & Sun, 2012b).

Outras ferramentas adicionam *plug-ins* a IDEs para torná-las ambientes adequados a programação colaborativa, tais como: Cole-Programming (Jurado et al., 2013), Saros (Salinger et al., 2010) e Jazz Sangam (Devide et al., 2008). As desvantagens dessas ferramentas são a falta de orientação e assistência para desenvolver o código, falta de suporte a gestão de grupos e falta de edição colaborativa do código (Serrano-Cámara et.al, 2014).

ICI (*Idaho Collaborative* IDE) (Bani-Salameh et.al, 2008) é um ambiente colaborativo que combina um editor de código e um depurador de colaboração síncrono dentro de um ambiente virtual multiusuário 3D. No mundo virtual compartilhado, é possível ver os usuários que estão no ambiente e o que eles estão fazendo, convidar usuários para participar de uma sessão de trabalho e se comunicar via texto e VoIP (voz sobre IP). O usuário que iniciou a sessão é quem tem o controle dela, este controle pode ser passado para outro usuário. Quem controla a sessão tem autorização para alterar, compilar e executar código.

Nos últimos anos, também foram desenvolvidos vários softwares comerciais, tais como: SubEthaEdit[1], collabedit[2], Codeshare[3], CoderPad[4], e ferramentas de código aberto como FirePad [5]e EtherPad Lite[6] que são editores de texto colaborativo que utilizam *plug-ins* para suportar programação colaborativa.

2.4 Laboratório remoto

A expressão laboratório remoto é utilizada para definir experimento que é conduzido e controlado remotamente através da Internet. Diferente dos laboratórios virtuais, onde os experimentos são simulados, os laboratórios remotos utilizam componentes ou instrumentação reais em um local diferente de onde eles estão sendo

[1] https://www.codingmonkeys.de/subethaedit/
[2] http://collabedit.com/?ref=binfind.com/web
[3] https://codeshare.io/
[4] https://coderpad.io/
[5] https://firepad.io/
[6] http://etherpad.org/

manipulados. O usuário acessa e controla o computador do laboratório e pode acionar equipamentos, fazer observações, testar condições e coletar dados do experimento. A instalação de câmeras de vídeo no ambiente do laboratório permite que o usuário acompanhe em tempo real a execução do experimento (Chen et al., 2010; Ma & Nickerson, 2006).

Os laboratórios remotos modernos são ferramentas de software e hardware que possuem um conjunto típico de componentes (Gomes & Bogosyan, 2009): o objeto controlado (experimento, instrumento ou maquete experimental); os equipamentos e dispositivos de instrumentação que permitem a aquisição e controle de dados; o computador servidor do laboratório que assegura o controle e monitoramento do experimento; o servidor que faz a ligação, através da Internet, entre os usuários remotos e o servidor do laboratório; o servidor de câmera web; e a aplicação cliente que permite ao usuário acessar o experimento através de um navegador web. Estas plataformas devem fornecer um conjunto típico de serviços: ferramentas administrativas, gerenciamento e rastreamento de usuário, agendamento (reserva) e/ou fila para acessar os recursos e gerenciamento de experimentos, além de garantir escalabilidade, facilidade de manutenção do sistema, segurança e qualidade do serviço (Özbek & Kara, 2010; Orduña et.al, 2009; Guimarães et.al, 2011).

Atualmente, laboratórios remotos são aplicados em diversos cenários educacionais para a aprendizagem de disciplinas como engenharia, física, química, computação, etc (Heradio et al., 2016; Alkhaldi et al., 2016). Estudos mostram que este tipo de laboratório é uma ferramenta eficaz na compreensão conceitual de conteúdo (Gomes & Bogosyan, 2009; Machotka et.al, 2009; Lang, 2012; Brinson, 2015).

Ma & Nickerson (2006) realizaram uma vasta revisão da literatura sobre laboratórios *hand-on* tradicionais, remotos e virtuais na educação e constataram, entre outas coisas, que os laboratórios remotos e virtuais são eficientes na compreensão conceitual e habilidades profissionais. Mas o desempenho de aprendizagem não pode ser atribuído apenas à tecnologia do laboratório (*hand-on*, remoto ou virtual), é importante considerar fatores como motivação, colaboração dos pares, *feedback* corretivo de erros e riqueza dos meios de comunicação para desenvolver laboratórios mais interativos e imersivos.

O estudo realizado por Corter et.al (2011) compara a eficácia de laboratórios remotos e virtuais com os tradicionais *hand-on* e conclui que as novas tecnologias (remotos e simulados) podem ser usadas de forma eficaz para ensinar a compreensão conceitual, mas o modelo do laboratório é apenas um fator que afeta os resultados da aprendizagem, também devem ser considerados: a dificuldade do experimento, o processo de trabalho (individual ou em grupo), e a quantidade de tempo que as equipes dedicam a análise de

dados e a elaboração de relatórios. Para a educação a distância, deve haver um esforço para incorporar aos laboratórios ferramentas de colaboração eficazes.

É possível encontrar alguns laboratórios incorporados a AVA. As soluções mais comuns para essa integração são a utilização de *plug-ins* ou de padrões de empacotamento de conteúdo como o *Shareable Content Object Reference Model* (SCORM) (Gonzalez-Barbone & Anido-Rifon, 2008).

Vicente et al. (2010) apresentam um laboratório remoto para automação industrial que integra o sistema de autenticação e de reserva com o *Moodle* através do *plug-in M*eeting *Room Booking System* (MRBS) e de *scripts* em PHP. Barrios et al. (2013), Chaos et al. (2013) e Torre et.al (2013) descrevem o desenvolvimento de laboratórios remoto onde a interface do usuário é implementada em EJS (*Easy Java Simulations*) e os estudantes podem acessar o laboratório utilizando uma aplicação *stand-alone* ou um *applet* incorporado ao *Moodle*.

Os laboratórios remotos são recursos educacionais adicionais incorporados ao AVA. Mas, segundo Sancristobal et al. (2010), existem vários serviços duplicados entre estes dois sistemas: autenticação e autorização do usuário, gerenciamento de grupo, ferramentas administrativas, controle de usuário, e algumas vezes o agendamento. Para evitar redundância, alguns trabalhos sugerem que estes serviços sejam delegados ao AVA, mas que o agendamento e o rastreamento de usuário muitas vezes precisam ser compartilhados com o laboratório (Orduña et al., 2013).

O projeto LiLa (Library of Labs) (Mateos et.al, 2011) utiliza objetos SCORM para fornecer experimentos remotos. Estes objetos podem ser baixados e reutilizados em AVA compatível com o padrão SCORM. Apesar da portabilidade, Orduña et al. (2013) argumenta que esta é uma tecnologia do lado do cliente e, portanto, não pode conter qualquer código do servidor, não suporta um modo seguro para a troca de credenciais, não garante reservas e não retorna resultados aos AVAs.

Orduña et al. (2012) consideram que estas são soluções *ad-hoc*, pois são específicas para um laboratório e um ambiente de aprendizagem, elas não suportam a integração de outros laboratórios remotos nem a integração em outros AVAs. Os autores apresentam uma integração baseada em protocolo de federação que considera o AVA como um sistema consumidor e o sistema de gestão do laboratório remoto é o sistema provedor, onde o AVA gerencia a autenticação e autorização dos usuários, o sistema provedor gerencia o agendamento e o acesso aos laboratórios armazenando o que o usuário fez. Para demostrar esta solução, Orduña et al. (2013) descreve a ferramenta gateway4labs, que é uma abordagem de código aberto para a integração de vários laboratórios remotos em diferentes ambientes digitais de aprendizagem. Mas esta solução possui algumas

limitações: a API fornecida suporta apenas os pedidos de acesso do laboratório remota no momento do pedido, em vez de suportar também um esquema de reserva com base no calendário, e o sistema ainda não suporta o uso de recuperação. Outra dificuldade, é a necessidade de escrever código, mesmo que pouco, para a integração com alguns AVAs.

2.4.1 Colaboração multiusuário em laboratório remoto

Um laboratório remoto colaborativo permite que dois ou mais usuários realizem experiências remotas, ao mesmo tempo, como uma equipe. No entanto, a maioria dos laboratórios *online* não apoia o trabalho colaborativo em grupo (Heradio et.al, 2016). Soluções para esta limitação incluem (Torre et al., 2013):

- Implementação de laboratórios web em AVA, onde os laboratórios tiram proveito das ferramentas do AVA para apoiar interação através de ferramentas de comunicação síncrona e assíncrona. Esta abordagem pode ser vista em Lima (2013) que utiliza o SCORM (*Sharable Content Object Reference Mode*) para integrar laboratório online ao *Moodle*;
- Laboratórios *online* embutidos em mundos virtuais como *Second Life* (García-Zubia et.al, 2010). *Wonderland* (Fayolle et.al, 2011), *OpenSim* (Pereira, Paladini & Schaf, 2012), pois oferecem vários canais de colaboração entre os usuários;
- Laboratórios *online* incorporam colaboração multiusuário através de ferramentas de comunicação como bate-papo e videoconferência ou interação simultânea de vários participantes com o mesmo laboratório, como em Joolingen et al. (2005), Callaghan et al. (2007) e Jara et al. (2009). Neste caso, o laboratório atua com o meio de comunicação entre os usuários.

Além destes exemplos, Torre et al. (2013) incorporaram características das abordagens citadas e desenvolveram os *plug-ins EJSApp* e *EJSAppCollabSession* para o *Moodle* que permitem criar sessões de trabalho colaborativo com *applets* desenvolvidos em EJS. Em uma sessão colaborativa, os usuários assistem a execução do experimento e se comunicam via *Skype*[7].

Gravier et.al (2012) implementaram um sistema inteligente baseado em ontologias para codificar e executar políticas de colaboração entre usuários de laboratórios remotos. Este sistema prevê a existência de um tutor que coordena a sessão do laboratório e define as regras de colaboração de acordo com o contexto do ambiente de aprendizagem e metas pedagógicas.

[7] https://www.skype.com/pt-br/

2.5 Laboratório remoto de robótica no ensino de programação

Robótica na educação é uma tendência em expansão (Cruz-Martín et al., 2012). A capacidade dos robôs de explorar e interagir com o mundo real através de sensores e atuadores possibilita sua aplicação em uma série de atividades educativas que ajudam e promovem a aprendizagem (Benitti, 2012).

Os robôs permitem experiências *hand-on* com problemas reais tornando possível a visualização prática de conceitos abstratos. Neste sentido a robótica pode ser usada para contextualizar o ensino de programação, pois mostra a utilidade do que está sendo aprendido (Cooper & Cunningham, 2010; Guzdial, 2010). Pesquisas indicam que a robótica pode melhorar o aprendizado e aumentar a motivação em curso introdutórios de programação (McGill, 2012; Yamanishi et al., 2013; Ahmed & Alsaleh, 2011; Tsang et al., 2014).

Apesar das vantagens educacionais, a maioria dos laboratórios de robótica só está disponíveis para os alunos em períodos restritos. O uso de laboratório remoto possibilita compartilhamento de recurso e garante flexibilidade de tempo e espaço para os usuários. Atualmente, existem vários laboratórios remotos de robótica em uso no mundo, alguns utilizam robôs móveis (kulich et.al, 2013; Casini et.al, 2014), mas poucos dão suporte a ferramentas de desenvolvimento de programas necessárias ao ensino de programação.

A universidade de Deusto disponibiliza no laboratório remoto WebLab-Deusto três atividades de aprendizagem diferentes usando o robô Azkar-bot, são elas: *robot-proglist* – permite que o usuário escolha um programa pré-definidos para movimentar o robô; *robot-moviment* – permite mover o robô pra frente, pra trás e para ambos os lados utilizando os botões da interface gráfica ou o teclado; e *robot-standard* – permite que o usuário faça upload de código que será executado pelo robô remotamente (García-Zubia & Angulo, 2011).

A *University of Technology Sydney*, através do Instituto LabShare, disponibiliza acesso remoto a plataforma *iRobot Create* que permite explorar os conceitos de teleoperação de robôs, precisão de sensores, localização e mapeamento. Através do sistema web, o aluno pode controlar remotamente o robô e observar seus movimentos, os dados fornecidos por diferentes sensores e compará-los. Existem três modos diferentes de manipular o iRobot: 1) manualmente – utilizando os botões de direção presentes na interface ou as setas do teclado; 2) de forma autônoma (registro de dados) – ao clicar no botão “Start”, o robô tenta se localizar e, se for bem sucedido, começa a dirigir entre vários "pontos de interesse" dentro do labirinto de forma autônoma; ou 3) por *upload* de código – o aluno pode testar algoritmos de controle desenvolvidos no software de simulação de robô e enviá-los para o robô (iRobot, 2012).

Chaos et.al (2013) descrevem o desenvolvimento de um laboratório remoto de robótica que tem como objetivo permitir que os estudantes criem um mapa do meio ambiente do robô com base nas medições dos sensores e da posição do robô. O laboratório é composto por um robô NTX LEGO, que é controlado usando *Bluetooth*, uma câmara de alta resolução, que calcula a posição do robô sobre o piso, uma *webcam*, que dá ao usuário o retorno visual do funcionamento do robô, e o computador servidor do laboratório. O sistema é implementado com MATLAB, EJS e LabVIEW. Os estudantes podem acessar o laboratório através de uma aplicação *stand-alone* ou um *applet* incorporado ao *Moodle*, que contém o sistema de reserva e permite a colaboração entre os usuários.

A universidade de Siena oferece no *Automatic Control Telelab* (ACT) experimentos remotos onde os alunos podem interagir com robô para aprender conceitos básicos e avançados sobre controle, robótica, trajetória do robô móvel, controle ótimo e teoria dos jogos (Casini et.al, 2013).

Estes laboratórios remotos permitirem que o usuário envie código para ser executado pelo robô, entretanto, eles não fornecem um ambiente de programação onde os códigos podem ser escritos, editados e compilados.

Aroca et al. (2012) propõem um sistema web onde o usuário pode desenvolver seu próprio programa para controlar o robô. O sistema utiliza um smartphone como computador principal para controlar o robô e como servidor web. Entretanto, não há um controle que permita acompanhar as atividades e desempenho de cada aluno.

Petrovic & Balogh (2012) descrevem o robô móvel Robotnačka projetado para ensinar linguagem de programação Logo para iniciantes. O usuário pode acessar o Robô remotamente e controlá-lo a partir do ambiente de programação Imagine Logo que deve estar rodando na máquina do usuário.

Iturrate et al. (2013) mostram a integração de um laboratório remoto robótico com *Serious Games*. Os estudantes podem controlar o robô usando o teclado ou através do linguagem de programação gráfica *Google Blockly*[8]. O código é executado no computador que controla e envia comandos para o robô. O programa não é executado no Arduino[9] do robô, mas no computador.

Park & Lenskiy (2014) apresentam um laboratório remoto de robótica para ensino de programação C. O projeto de baixo custo simula sensores através de uma única câmera complementada com o algoritmo de visão computacional. Entretanto, o artigo não mostra o ambiente onde os alunos podem desenvolver os programas.

[8] https://developers.google.com/blockly/
[9] https://www.arduino.cc/

O SyRoTek é um laboratório remoto que permite o controle de uma plataforma multi-robôs em uma arena configurada dinamicamente. Ele é utilizado para o ensino de robótica móvel, inteligência artificial, engenharia de controle, entre outros. Além da página web, os usuários podem acessar a plataforma através da IDE NetBeans utilizando um conjunto de plug-ins disponibilizado pelo SyRoTek (Kulich et al., 2013). Esta solução permite que o usuário acesse o laboratório e simultaneamente utilize todos os recursos de programação disponíveis no NetBeans (Kosnar et al., 2011), mas possui o inconveniente da instalação da IDE e dos plug-ins na máquina cliente. Além disto, o SyRoTek incentiva o trabalho colaborativo de forma assíncrono utilizando o Subversion (SVN)[10], um sistema de controle de versão onde códigos-fonte são gerenciados e partilhados entre os alunos que trabalham na mesma equipe e com o professor.

Chen & Zhou (2015) propõem um ambiente de programação robótico baseado na Web denominado eRobotic. O ambiente permite a programação no estilo arrastar e soltar. Os objetivos do sistema são motivar os alunos e melhor a compreensão dos conceitos básicos de computação e programação.

[10] Version Control With Subversion - http://svnbook.red-bean.com/en/1.7/svn-book.pdf

3 Proposta de uma arquitetura para sistema colaborativo de programação com suporte a experimentação remota

Este capítulo apresenta a arquitetura geral do sistema de colaboração voltado ao ensino de programação usando laboratório remoto. O propósito da arquitetura é dar suporte ao ensino de programação contextualizado utilizando planejamento, codificação e teste de soluções de problemas. O sistema deve permitir o aprendizado ativo, onde o estudante é responsável pelo seu aprendizado e o professor é um facilitador, e possibilita o aprendizado colaborativo, onde estudantes realizam suas atividades em grupos pequenos, fornecendo e recebendo auxílio, estimulando a aprendizagem compartilhada e o desenvolvimento de habilidades sociais.

A maioria dos sistemas colaborativos de programação dão suporte apenas a codificação de algoritmos, o que, segundo Koorsse et al.(2015), representa apenas uma etapa no processo de aprendizagem de programação.

A partir do estado da arte apresentado no capítulo 2, foram identificados problemas e possíveis soluções para ferramentas educacionais relacionadas a programação de computadores. Características chaves para o uso de laboratórios remotos de robótica, como contexto em um sistema colaborativo, foram organizadas de maneira a definir uma base para o desenvolvimento de sistemas CSCL voltados para o ensino de programação de computadores. Este trabalho é original, pois, apesar de existirem ferramentas de CSCL para programação e alguns poucos laboratórios remotos de robótica integrados a IDE, até o momento, não foi observada pesquisa que propõe o estudo de arquitetura que integra todas as características aqui propostas.

3.1 Características importantes de ambiente de programação colaborativa com laboratório remoto

As principais características identificadas para a proposição de um sistema CSCL de programação baseado em laboratório remoto de robótica e programação colaborativa foram:

- Gerenciamento e organização de materiais didáticos de programação – Como ambiente educacional, é necessário que o sistema hospede material didático sobre programação de computadores.
- Apoio a vários meios de comunicação – a comunicação efetiva é fundamental ao processo de ensino aprendizagem, por isto, é necessário disponibilizar recursos que facilitem a comunicação síncrona e assíncrona entre os usuários.

- Suporte a estratégia de programação – No processo de aprendizagem de programação os alunos devem ser incentivados a pensar no problema, entender seus requisitos e propor solução algorítmica.
- Suporte a implementação – A solução algorítmica encontrada precisa ser traduzida para uma linguagem de programação, e o código gerado precisa ser testado e validado.
- Suporte a colaboração – Um aluno pode solicitar colaboração de colegas para realizar suas atividades. Este processo gera dados importantes como: participantes do grupo, contribuição de cada participante na resolução das atividades e tempo dedicado de cada um.
- Suporte a robótica educativa remoto – A robótica fornece um contexto para o ensino de programação, permitindo que os alunos visualizem a aplicação de conceitos abstratos. O acesso as atividades contextualizadas através de um laboratório remoto garante flexibilidade de tempo e espaço para os alunos realizarem suas atividades quando e onde quiserem.
- Suporte a avaliação dos dados gerados no processo de colaboração – Os professores devem ser capazes de acompanhar e mediar o processo de colaboração dos alunos.
- Utilização de ferramentas de código aberto – permitem a adaptação conforme a necessidade do projeto e não tem custo de licença.
- Usabilidade – Softwares educacionais devem ser fáceis de usar de modo que os alunos gastem tempo aprendendo o conteúdo e não a utilizar a ferramenta.
- Modularidade – A divisão do sistema em módulos possibilita o reuso de componentes de softwares.

Considerando estas características, a arquitetura foi organizada em módulos para garantir reuso e interoperabilidade, sua descrição é apresentada a seguir.

3.2 Componentes da arquitetura proposta

A aprendizagem de programação engloba resolução de problema, análise e planejamento de soluções, representação da solução em uma linguagem de programação (codificação), verificação sintática e semântica da solução, depuração e correção. A arquitetura proposta fornece suporte a todas estas etapas, além de possibilitar a aprendizagem em contexto, através do uso de laboratório remoto de robótica, e a colaboração entre os usuários. Os componentes fundamentais da arquitetura são: módulo de estudo, módulo de análise, módulo de codificação e módulo de experimentação. Além do repositório de dados onde todas as informações referentes e comuns aos módulos são armazenadas. O repositório é responsável por centralizar e sincronizar os dados entre os

módulos. Todos os módulos possuem conexão com o repositório de dados podendo ler e/ou escrever dados. A Figura 3 mostra a arquitetura proposta, seus módulos são descritos a seguir.

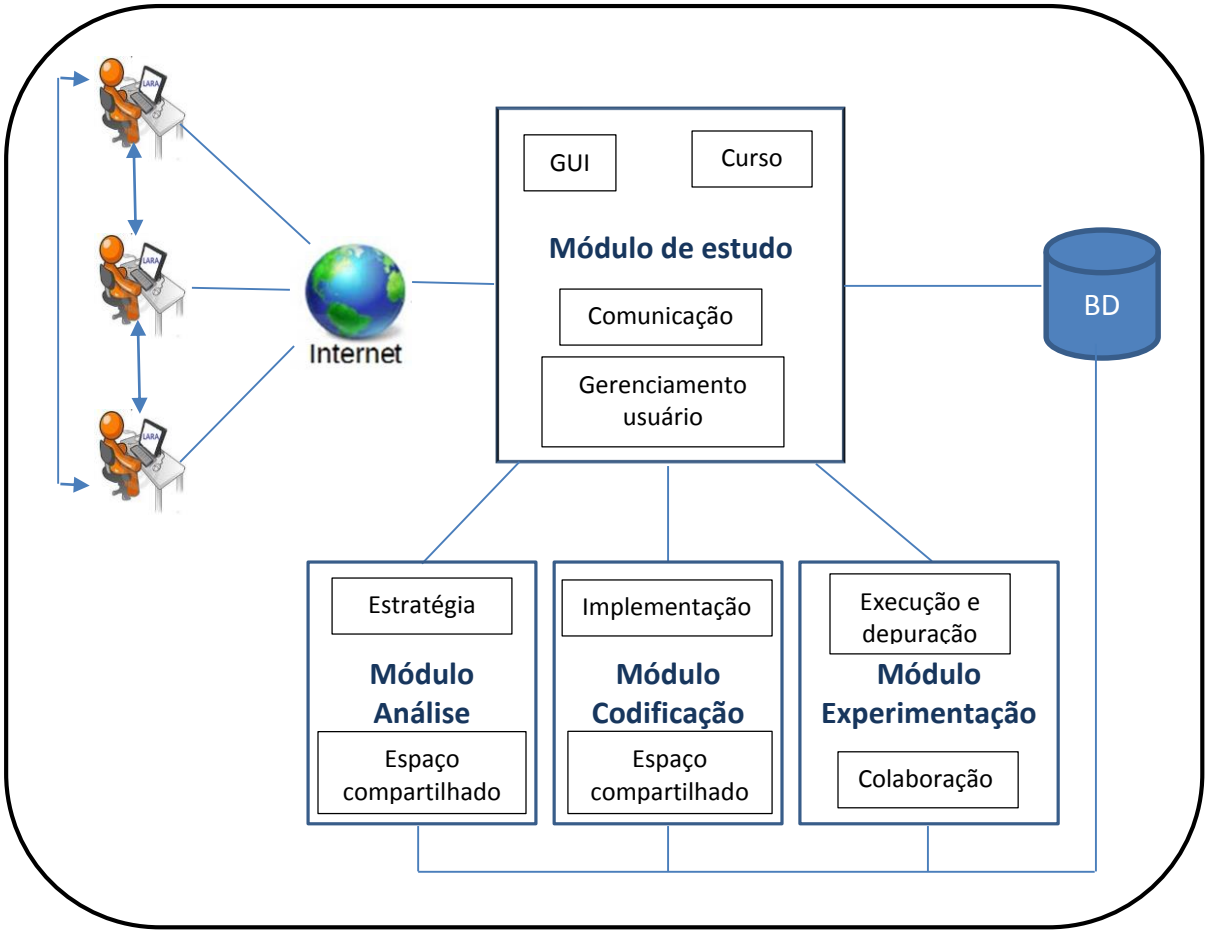


Figura 3 – Componentes da arquitetura proposta

3.2.1 Módulo de estudo

Como ambiente educacional, é natural que o sistema hospede material didático que ajuda os alunos na aquisição de conceitos teóricos e na realização de atividades práticas. Nesta arquitetura, este papel é desempenhado pelo módulo de estudo que, além de permitir o gerenciamento de mídias educacionais, deve garantir o gerenciamento dos usuários, a comunicação e a colaboração entre eles.

É através do modulo de estudo que o usuário tem acesso ao sistema completo. Ele permite ao aluno acessar os materiais, recursos e atividades disponibilizados pelo professor, realizar atividades utilizando os módulos de análise, codificação e

experimentação, solicitar e fornecer ajuda a outros usuários, além de se comunicar através de chats, fóruns, etc. O professor/tutor pode inserir e atualizar materiais e recursos didáticos, interagir com os alunos e acessar relatórios de acesso e de desempenho dos alunos.

Estes requisitos são implementados em ambientes virtuais de aprendizagem (AVA) muito utilizados na educação a distância. Para integrar o sistema, o AVA deve possuir interface web, código aberto para permitir a customização e um banco de dados que possibilite a interconexão das informações entre os outros módulos da arquitetura.

3.2.2 Módulo de análise

Como forma de estimular as boas práticas de programação que inclui o entendimento do problema e análise da solução, a arquitetura integra o módulo de análise que permite que o professor defina um modelo de plano que deve ser seguido e preenchido pelo aluno antes de iniciar a codificação.

As funcionalidades principais desse módulo são mostradas na Figura 4. O professor define a estrutura (modelo ou *template*) do planejamento para a solução de um problema. Por exemplo, o professor pode solicitar que o aluno informe o que ele entendeu do problema, quais são as entradas e saídas, quais as variáveis e seus tipos, um algoritmo para solucionar o problema, os componentes do grupo, as atribuições de cada componente, etc. O aluno deve pensar sobre o problema, criar e editar o plano com as informações solicitadas. Durante o desenvolvimento do plano ele pode solicitar que outros colegas colaborem na elaboração do plano. A colaboração permite a comunicação entre os alunos e a visualização e edição online do documento por todos os colaboradores. Um plano pode ser salvo e editado várias vezes.

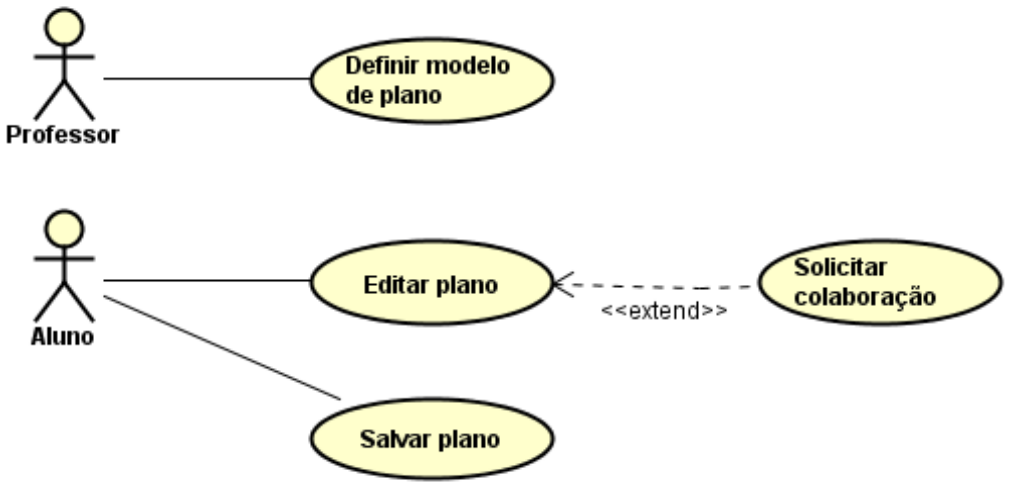


Figura 4 – Funcionalidades do módulo de planejamento

Para compor este modulo, pode ser utilizado *outliner*[11] ou editor de texto simples que possua interface web e código aberto.

3.2.3 Módulo de codificação

O módulo de codificação permite a implementação de algoritmos em linguagem de programação. Ele deve atender as seguintes características:

- Conter as três tarefas principais de programação (edição, compilação e execução) na interface principal do usuário;
- Editor de código que permite vários usuários alterarem o código simultaneamente;
- Ferramentas de coordenação para compilar ou executar o programa;
- Área de console para mostrar os erros de compilação;
- Permitir que cada usuário escolha qual trecho do código e do console quer visualizar;
- Ferramentas de consciência que permite cada colaborador sabre o que os outros estão fazendo no código;
- Comunicação via texto e/ou voz.

A Figura 5 mostra as funcionalidades básicas deste módulo. Ele permite que o usuário (aluno ou professor) crie, salve, abra, edite e compile programas escrito em uma linguagem de programação. Como no módulo de análise, o usuário pode solicitar colaboração de colegas e professores, isto possibilita a comunicação, a visualização e a edição online do código por todos os colaboradores.

Este módulo deve ser composto por um editor de código web e um compilador de linguagem de programação, ambas ferramentas de código aberto.

3.2.4 Módulo de experimentação

O módulo de experimentação ou modulo de laboratório remoto permite a manipulação e controle de dispositivos robóticos que ajuda os alunos no teste e depuração de soluções de problemas. Este módulo tem como objetivo permitir a contextualizado, pois os professores podem definir contextos de aplicações para as atividades e os alunos podem verificar onde e como aplicar os conceitos abstratos estudados.

[11] https://en.wikipedia.org/wiki/Outliner

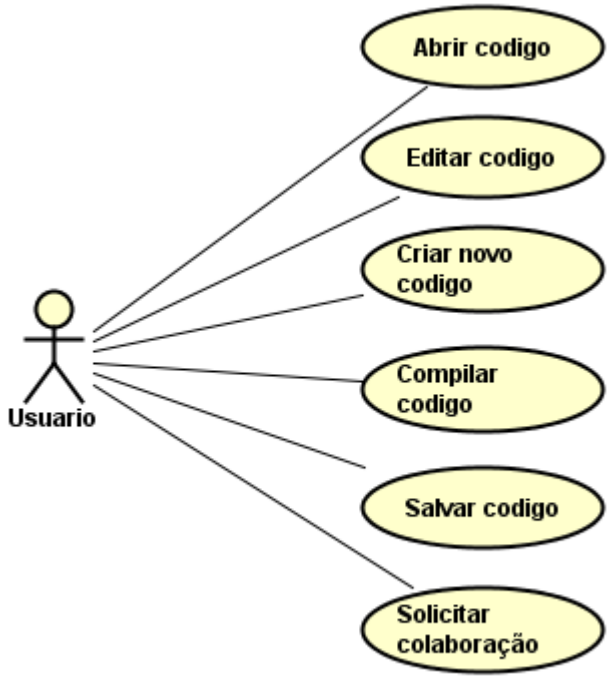


Figura 5 – Funcionalidades do módulo de codificação

O módulo de experimentação é basicamente responsável pelas seguintes funcionalidades do sistema (Figura 6): gerenciamento de experimento que permite cadastro, alteração, exclusão e bloqueio temporário do experimento para possível manutenção; gerenciamento de reserva, onde o usuário pode agendar dia e horário para iniciar uma sessão de experimento, alterar os dados da reserva e até mesmo cancelá-la; durante a sessão de experimento é possível programar e controlar o experimento e solicitar a colaboração de outros usuários. A colaboração durante uma sessão de experimento garante que todos os usuários, participantes da sessão, possam visualizar e também manipular o experimento.

Para compor este módulo, é necessário que o laboratório remoto tenha integração com uma IDE online, pois os códigos devem ser editados, compilados e testados dentro do próprio laboratório.

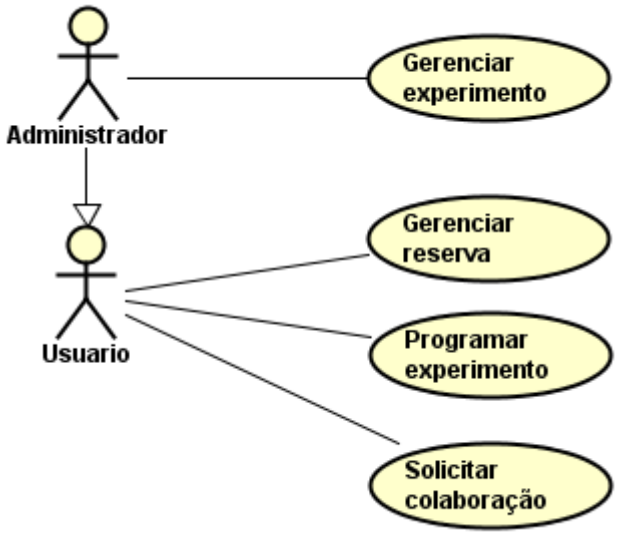


Figura 6 – Funcionalidades do módulo de experimentação

4 LaraPC – seleção e desenvolvimento de ferramentas para compor a arquitetura proposta

Este capítulo e o próximo abordam a implementação do protótipo da arquitetura apresentada no capitulo 3, chamado de Lara para Programação Colaborativa ou LaraPC. O objetivo do LaraPC é mostrar a viabilidade da arquitetura e a integração das ferramentas selecionadas para compor cada módulo.

As ferramentas, tecnologias e conceitos do LaraPC foram escolhidas com base nas características descritas no capitulo 3. Para os módulos de estudo e de análise, foram encontradas ferramentas que satisfazem os requisitos. Entretanto, os laboratório remotos de código aberto analisados não apoiam as atividades inerentes ao ensino de programação. Portanto, para compor o módulo de codificação e o módulo de experimentação, foi desenvolvido o laboratório remoto de robótica móvel integrado a uma IDE simples chamado de Ambiente de Programação e Controle (APC). As ferramentas selecionadas e o APC são descritas a seguir.

4.1 Ferramenta de aprendizagem virtual

Ambiente Virtual de Aprendizagem (AVA) ou Sistema de Gestão de Aprendizagem (Learning Management Systems - LMSs) é um sistema web que auxilia na disponibilização de cursos ou disciplinas on-line e permite a interação entre os alunos, professores e tutores envolvidos no processo de ensino aprendizagem. Os AVAs são comumente utilizados em cursos semipresenciais e de Educação a Distância (EaD).

Geralmente, os AVAs possuem capacidade de gerenciar e hospedar material de ensino, mídias eletrônicas, tutoriais, entre outros, além de oferecerem ferramentas para controle e administração de usuários, editores simples de HTML, ferramentas sociais para identificação, comunicação e colaboração entre usuários e ferramentas de avaliação e monitoramento.

Atualmente é possível encontrar vário AVAs de código aberto, tais como Moodle[12], Claroline[13], TelEduc[14] e Chamilo[15], e de licença comercial, como Blackboard[16], WebAssign[17] e WebAula[18].

Para compor o módulo de estudo da arquitetura proposta foi escolhido o *Modular Object-Oriented Dynamic Learning Environment* (MOODLE) que é hoje um dos AVAs mais

[12] https://moodle.org/
[13] http://www.claroline.net/
[14] http://www.teleduc.org.br/
[15] https://chamilo.org/chamilo-lms/
[16] http://br.blackboard.com/
[17] https://webassign.com/
[18] http://webaula.com.br/index.php/pt/

utilizados em instituições de ensino superior. Ele foi adotado pelo Ministério da Educação (MEC) e também pela Universidade Estadual do Sudoeste da Bahia (UESB) como repositório de materiais e ambiente para os cursos a distância.

O Moodle é um sistema web baseado na arquitetura cliente-servidor desenvolvido na linguagem de programação PHP com suporte para vários bancos de dados como MySQL e Postgresql. O seu código é livre e sua arquitetura é modular, o que facilita a inclusão de novas funcionalidade, adaptação e personalização de acordo com as necessidades de cada instituição.

4.2 Ferramenta de edição de texto

A ferramenta escolhida para compor o módulo de análise foi o EtherPad Lite[19], ou simplesmente Etherpad. Esta é uma ferramenta de código aberto para edição online de documentos de texto (pad) baseado na web que permite que grupo de usuários trabalhe em conjunto. Um usuário cria uma pad e pode convidar outras pessoas para colaborar na sua construção. Todos os colaboradores podem ver e editar qualquer parte do texto em tempo real. Para garantir a consciência de contexto, cada participante é identificado com uma cor e um nome, toda vez que um participante faz uma contribuição no texto, o trecho da contribuição fica marcado com a sua cor (Figura 7).

O convite do Etherpad consiste em enviar o endereço URL (*Uniform Resource Locator*) da pad para os convidados. Este endereço é composto da seguinte forma: `<servidor que está rodando a aplicação>/p/<chave da pad>`. Por exemplo, para acessar a pad teste do servido etherpad.net é utilizado a URL: https://etherpad.net/p/teste. Qualquer pessoa que acessar o endereço pode editá-la.

O EtherPad também permite criar versões do documento, visualizar a linha do tempo com todas as alterações do texto, importar e exportar o documento em vários formatos. Ele é implementado em JavaScript usando node.js e vem com traduções para muitas línguas incluindo o português. O seu banco de dados padrão é o DirtyDB baseado em arquivo, mas pode ser usado com outros bancos de dados como MySQL.

[19] http://etherpad.org/

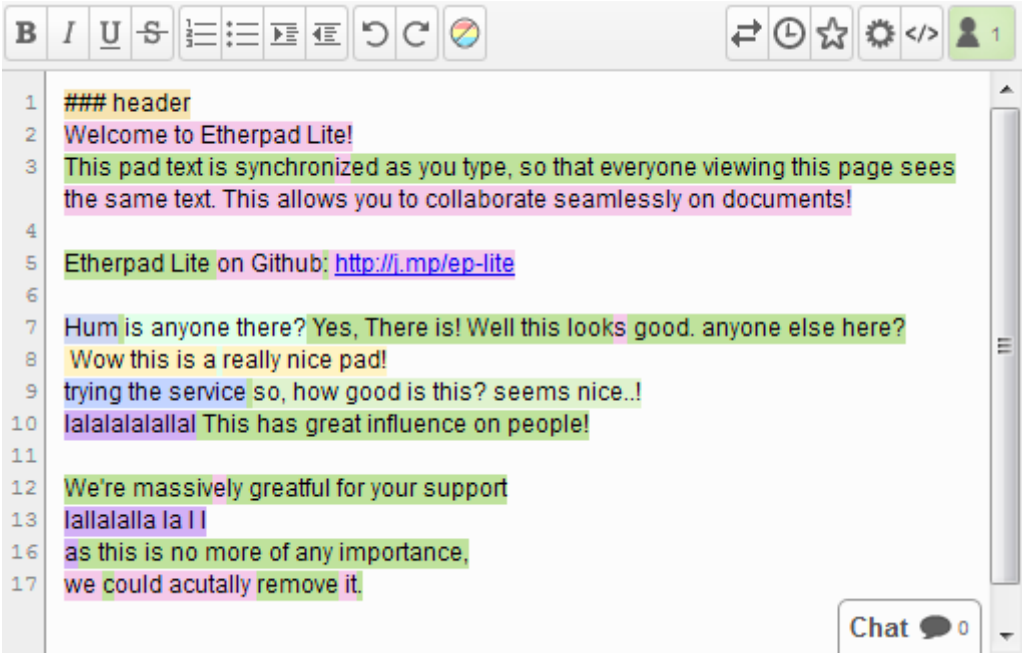


Figura 7 – Interface do EtherPad Lite

4.3 Ferramenta de suporte a colaboração

A colaboração deve estar presente em todas as etapas de aprendizagem suportada pela arquitetura proposta. A formação de grupos de estudo e a colaboração devem ser dinâmicos. Muitas funcionalidades de colaboração estão presentes no AVA, entretanto o Moodle possui limitações quanto a colaboração síncrona, pois não possuem recursos de consciência de contexto, nem de videoconferência e o bate-papo tem que ser agendado previamente. O EtherPad suporta a colaboração síncrona e o agrupamento dinâmico dos usuários.

Para estender a colaboração para todo o sistema, é utilizada a ideia de *party*[20] presente em jogos online. *Party* é um evento temporário que junta duas ou mais pessoas em seus computadores com o objetivo de jogar online. Uma *party* permite que vários jogadores participem simultaneamente de uma mesma partida podendo ser colaboradores ou rivais e proporciona uma forma de comunicação social.

No sistema proposto neste trabalho, um *party* é um grupo dinâmico formado por alunos para estudar programação de forma colaborativa. Em uma *party*, os participantes tem acesso aos módulos de análise, codificação e experimentação e as contribuições

[20] https://en.wikipedia.org/wiki/Party_game, https://pt.wikipedia.org/wiki/LAN_party, https://pt.wikipedia.org/wiki/Jogo_multijogador,

realizadas por um participante podem ser visualizadas pelos outros, além disso eles podem se comunicar através de um chat integrado aos módulos.

4.4 Ambiente de Programação e Controle (APC)

O APC é um laboratório remoto de robótica móvel integrado a uma ferramenta de programação. Ele permite que o usuário crie seus próprios códigos para manipular e controlar o robô em um ambiente web sem a necessidade de baixar ou instalar qualquer software adicional. A metodologia utilizada para desenvolver o APC foi uma adaptação do XP (*eXtreme Programing*) que, além dos testes de software, prever o teste de usabilidade do sistema.

Para acessar o APC, o usuário precisa estar cadastrado e possuir *login* e senha. As principais funcionalidades do sistemas são mostradas da Figura 8. O usuário pode fazer uma reserva, que consiste em agendar dia e horário para acessar o experimento. A reserva é importante para evitar conflito caso vários usuários tentem acessar o robô ao mesmo tempo. No dia e horário reservado, o usuário deve iniciar uma sessão de experimento, a sessão termina quando o usuário sai do laboratório ou quando acaba o tempo da reserva. Durante uma sessão é possível controlar o experimento, isto é, criar, editar, compilar programas e enviá-los para o robô executar, visualizar a execução através da câmera e cronometrar o tempo da sessão.

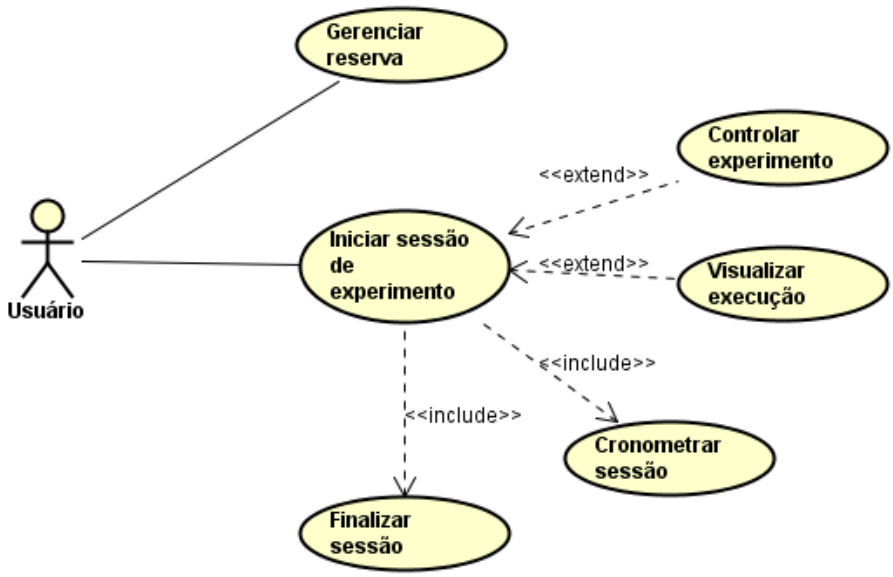


Figura 8 – Funcionalidades do APC

O APC é um sistema de hardware e software que possui os componentes típicos de um laboratório remoto (Tawfik et al., 2013) integrado a um IDE simples. Os componentes do APC são mostrados na Figura 9. O experimento do laboratório remoto é um robô móvel que se comunica com o servidor do laboratório via rede sem fio. O servidor do laboratório é responsável por controlar o experimento, ele compila e envia código para o robô. O servidor web faz a ligação entre o usuário e o laboratório. A interface do usuário disponibiliza as funções básicas de criação, edição, compilação e execução de código. Além disso, o usuário pode assistir em tempo real ao desempenho do robô através de uma câmera IP, receber *feedback* da compilação e enviar dados para o robô.

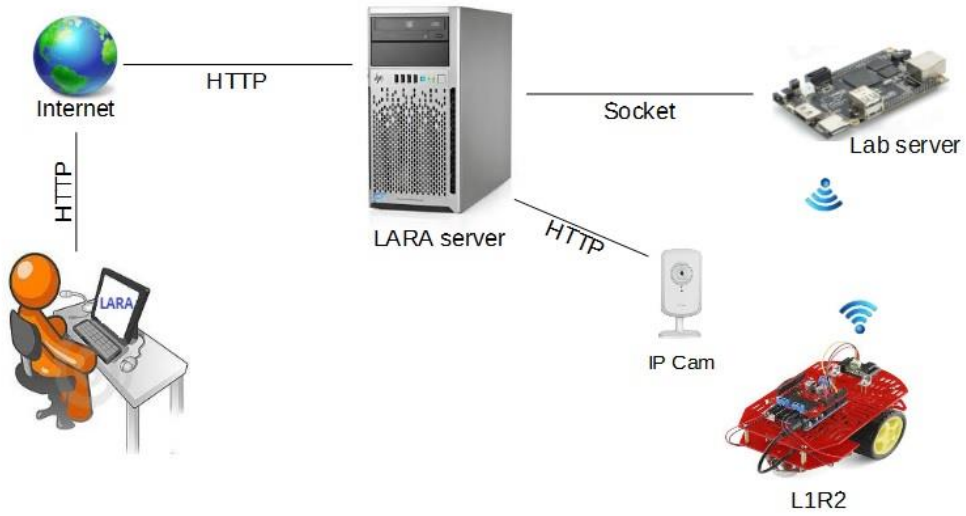


Figura 9 – Componentes do Ambiente de Programação e Controle

A seguir são descritos cada componente detalhadamente.

4.4.1 O robô

O robô, chamado L1R2 (*Lara Remote Robot*), é um carro seguidor de linha e resgate baseado nas regras da Olimpíada Brasileira de Robótica (OBR) para a modalidade prática (OBR, 2014). Ele deve operar em uma arena fechada. A arena utilizada neste projeto é uma superfície plana de formica branca com 2m de comprimento por 1,5m de largura (Figura 10).

O L1R2 é um robô educacional que possui as seguintes características: é de baixo custo, recebe e executa código remoto, tem alimentação que permite o seu uso 24 horas por dia durante toda a semana (24/7), e retorna automaticamente para um local da arena definida como posição inicial.

Muitos trabalhos apresentam kits de robótica, tipo LEGO, como solução de baixo custo (Benitti, 2012), entretanto estes kits são caros para a realidade de muitas instituições de ensino do Brasil. Para atender ao requisito de custo do projeto, optou-se por desenvolver todo o robô com equipamentos baratos e reciclados. A Figura 11 mostra a planta do carro, e como os componentes são arranjados no chassi. O L1R2 possui dois motores DC com torque de 800 gf.cm, que movimentam a roda esquerda e a roda direita. O terceiro suporte é feito de roda livre omni. O controle central do robô é realizado pelo microcontrolador Arduino Mega 2560. O sistema de odometria do robô consiste em dois codificadores quadráticos ópticos e uma bússola, magnetômetro gy-271, que permitem o controle de trajetória e orientação do robô. Para a função de seguir linha, são usados três pares de infravermelhos (transmissor e receptor). Três sensores ultra-sônicos localizados na frente e nos lados do robô são usados para detectar e evitar obstáculos. Os sensores de luz (LDR - Light Depend resistor) são usados para medir a intensidade da luz na arena.

Figura 10 – Arena do L1R2

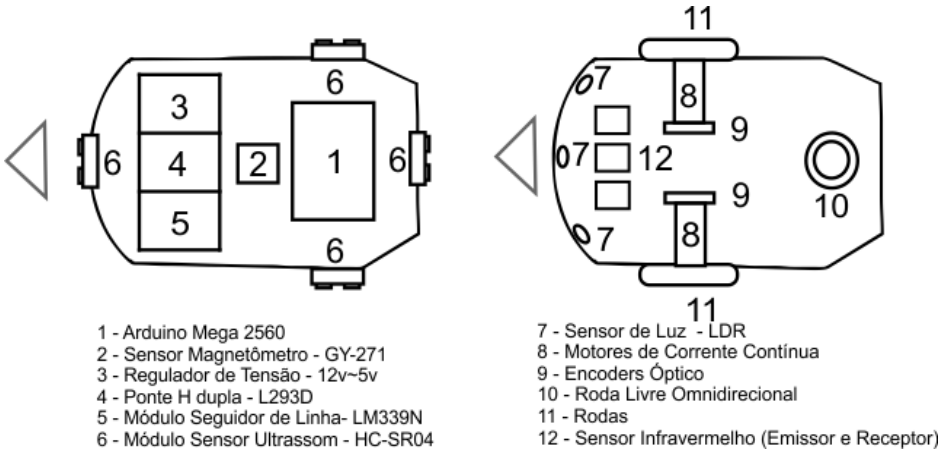


Figura 11 – Componentes do L1R2

Para receber o código remotamente, a comunicação entre o robô e o servidor do laboratório é feita por radiofrequência através do shield Pololu Wixel[21], que cria uma conexão lógica entre o servidor e o Arduino. Esta comunicação permite carregar código para o Arduino e enviar e receber textos através da porta serial do Arduino.

Para a alimentação do robô, as seguintes possibilidades foram avaliadas: uso de baterias recarregáveis, alimentação estilo carrinho bate-bate do iRobot (iRobot, 2012) e fiação tencionada. A alimentação deve garantir que o robô esteja disponível para o usuário durante toda a sessão de experimento. Uso de baterias foi descartado, pois o robô fica indisponível durante a recarrega. O modelo de carrinho bate-bate utilizado pelo iRobot usa um circuito paralelo para alimentação redundante. No projeto do L1R2, esta redundância seria feita por bateria recarregável, mas a implementação do controle de parada de carregar da bateria quando esta estivesse cheia demostrou ser inviável. Para garantir a autonomia de alimentação 24/7, foi utilizada a fiação tencionada por representar uma solução simples e barata. Neste modelo, o L1R2 é ligado a dois fios (um positivo e outro negativo) que ficam suspensos na arena de tal forma que se mantenham sempre esticado e não atrapalhem os movimentos do robô. O Arduino é alimentado com uma tensão de 12 V, e outros componentes recebem tensão de 5 V.

Na arena, foi definido um local, chamado de posição inicial, onde o L1R2 deve estar a cada início de sessão de laboratório. Para garantir o retorno automático do robô a posição inicial, foi colocado na arena uma fonte luminosa próximo a esta posição e implementado um seguidor de luz utilizando matemática intervalar (Trindade et al., 2010).

O L1R2 é programado na linguagem C Arduino que é baseada na linguagem de programação C/C++.

4.4.2 Servidor de laboratório

O servidor de laboratório é um minicomputador Cubieboard 2[22] com processador ARMv7 dual core de 1GHz, 1GB de memória RAM e sistema operacional Cubian R1. Ele é responsável por compilar o códigos do usuário e enviá-lo para o L1R2, bem como gerenciar a comunicação serial do Arduino.

A comunicação entre o servidor web e o servidor de laboratório é baseada na troca de mensagens. As mensagens seguem o modelo de protocolo de comunicação `comando+mensagem`, onde comando define a ação que o servidor de laboratório deve executar e mensagem contém informações complementares para a execução do comando.

[21] Pololu Wixel Shield for Arduino User's Guide https://www.pololu.com/docs/0J47
[22] http://cubieboard.org

Quando um usuário solicita a compilação de um código, o servidor de laboratório recebe o comando para compilar e a mensagem contendo o endereço do código. O servidor de laboratório faz o download do arquivo do código via link `http`, compila o código usando o IDE Arduino 1.5.6 Beta, cria uma pasta com data e hora atual do sistema para guardar o código e o resultado da compilação e retorna o resultado para o servidor web.

Quando o usuário solicita enviar o código para o robô, o servidor web emite o comando de enviar e a mensagem com o código. O servidor do laboratório repete o processo de compilação, se o código estiver sintaticamente correto, ele é então enviado para o robô através do Wixel Shield for Arduino. Depois desse processo, a comunicação serial entre o servidor do laboratório e o Arduino do robô é iniciada. Para implementar esta comunicação, foi usada a API Java Comm.

4.4.3 Servidor web

O servidor web do APC, ou servidor Lara, realiza a ligação entre o usuário e servidor do laboratório, como mostra a Figura 12. Ele permite que o usuário envie comandos e arquivos para o robô, além de fornecer feedback através do console, da porta serial do Arduino e da câmera de vídeo. O servidor web é também responsável pela comunicação entre os usuários via chat, pelo gerenciamento de reserva e pelo cronômetro da sessão de experimento.

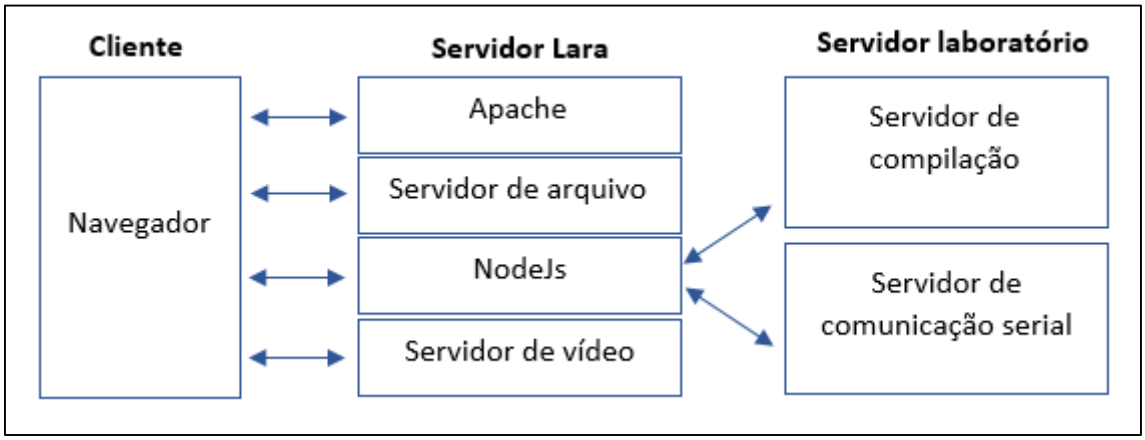


Figura 12 – Componentes de software do APC

O servidor web utilizado é o Apache versão 2.2.22-13+deb7u3. As páginas web são escritas em HTML5 e suas características em CSS3. Para gerar as páginas, é usado o PHP5 que faz as consultas ao banco de dados MySQL, e controla o acesso e o tempo de cada sessão de experimento. As funções disponíveis no APC foram desenvolvidas em JavaScript. O chat e a comunicação com o servidor do laboratório utilizam Sockets implementado em

NodeJs. Para transmitir as imagens da câmera IP, em tempo real, é usado o software VLC Media Player.

Quando o usuário solicita compilar código ou enviar código para o robô, o arquivo do código é salvo através de uma solicitação Ajax ao PHP e o socket.io do navegador emite a mensagem do evento correspondente a ação do usuário (compilar ou enviar). O socket que roda no servidor web envia o endereço do arquivo e a mensagem do evento para o servidor do laboratório que faz o download do arquivo e executa o evento. Depois, o servidor web emite uma mensagem de status (“compilando” ou “enviando”). O status é recebido pelo socket do navegador, que escreve o status no console e na serial e, então, desabilita a serial. Quando o servidor do laboratório envia a mensagem com o resultado do evento (erro ou sucesso) para o socket do servidor, esse transmite ao navegador, que libera a serial e escreve a mensagem no console. Nesse momento, é possível enviar dados para o robô através da porta serial. A figura 13 ilustra a sequência de tarefas realizada pelo servidor quando é solicitado o envio de código para o robô.

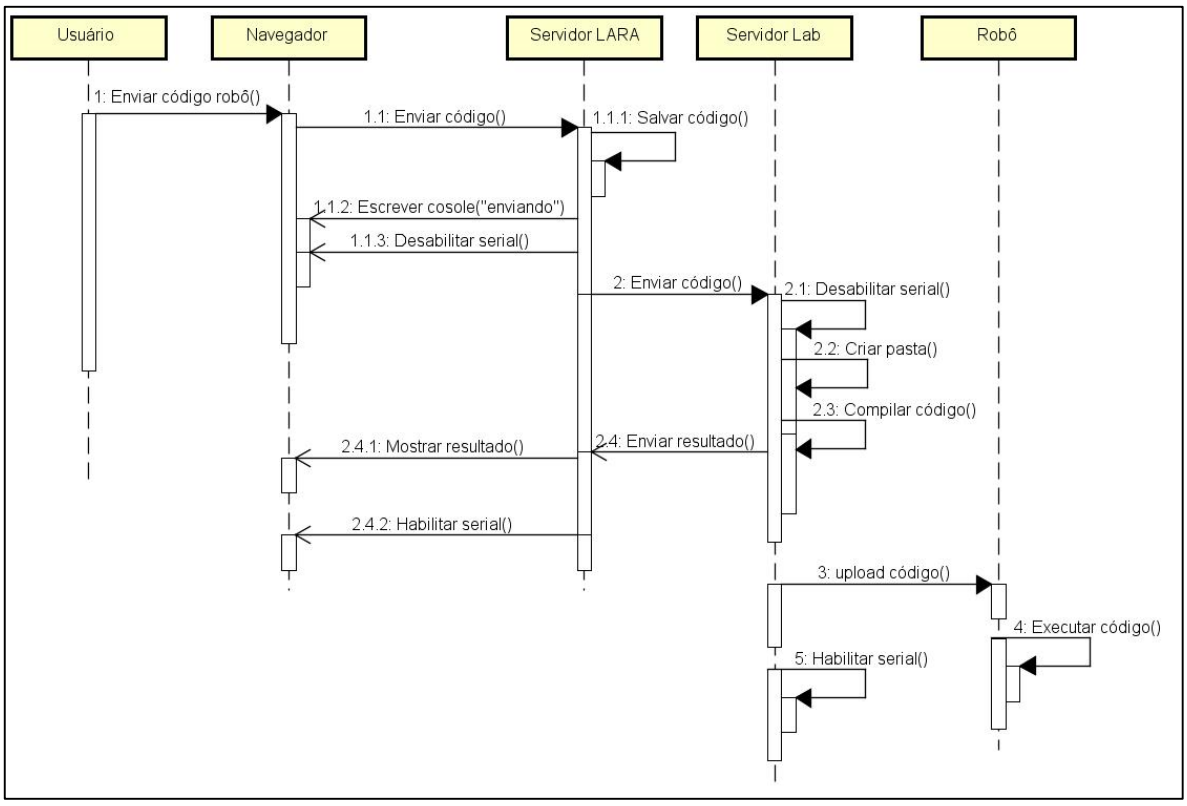


Figura 13 – Enviando código para o robô

4.4.4 Interface do usuário

A interface do usuário é uma parte muito importante no desenvolvimento de um laboratório remoto, pois ela deve habilitar o aluno para realizar as tarefas e fazer todas as

observações necessárias para atingir os objetivos de aprendizagem (Cooper, 2005). O aluno deve se concentra na realização de atividade, e não na utilização do software. A interação com o software deve ser natural e intuitiva proporcionando boa usabilidade (Costabile et.al, 2005).

Segundo a ISO 9241-11 (ISO, 1998), usabilidade é “a medida em que um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso”. Usabilidade é um atributo de qualidade relacionado a facilidade do uso do software (Juristo, Moreno & Sanchez-Segura, 2007). Refere-se à capacidade de aprendizado e memorização de um software, a eficiência de utilização, grau de propensão a erros e satisfação do usuário (Nielsen, 2012).

Por isto, a interface do APC foi desenvolvida considerando princípios de usabilidade como simplicidade e facilidade de uso. A Figura 14 mostra a janela de interface com usuário. Ela é dividia nas seguintes áreas: 1 – barra de ferramentas, 2 – editor de código, 3 – material didático, 4 – exemplos de código, 5 – visualização da câmera, 6 – console que exibe informações da compilação, 7 – porta serial e 8 – chat. A barra de ferramenta permite criar novo código, compilar código, enviar código para robô, salvar código, abrir código, abrir porta serial, abrir/fechar câmera, abrir chat, verificar as pessoas online, acompanhar o cronometro da sessão e sair do ambiente.

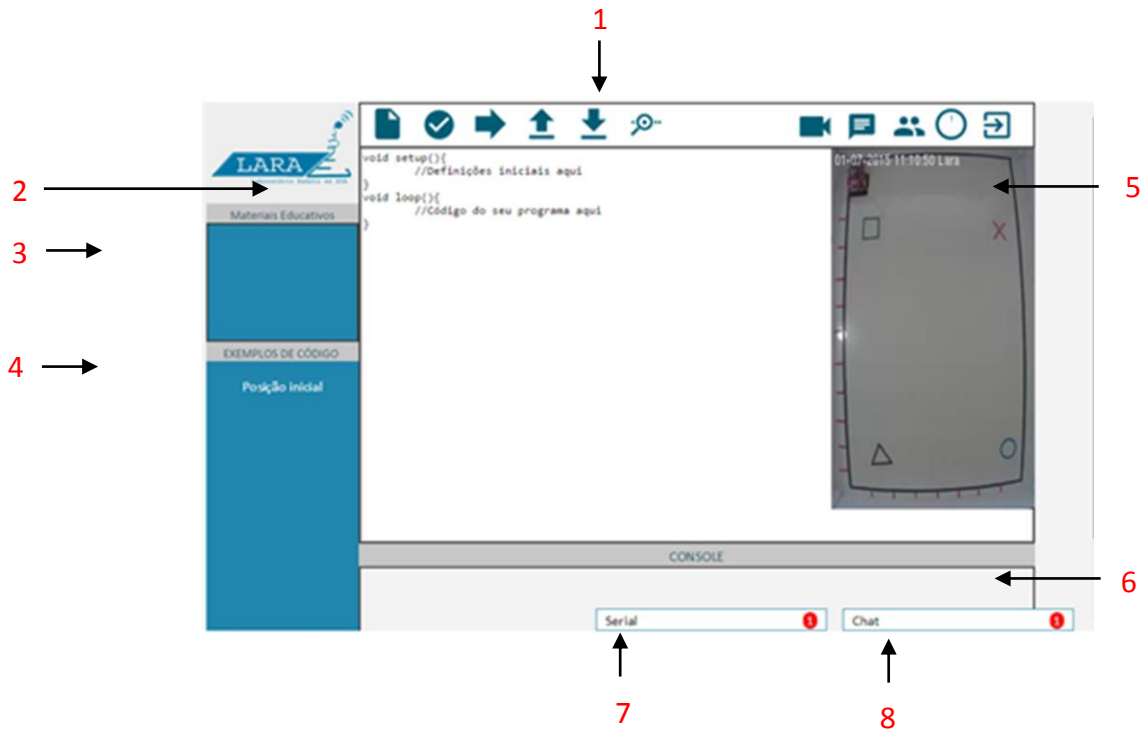


Figura 14 – Interface inicial do APC

Para avaliar a usabilidade do APC foi utilizado um método empírico (Fernandez et al., 2011) onde os usuários finais reais usaram o software para realizar um conjunto de tarefas predefinidas e o avaliador registrou os resultados para análise.

4.4.5 Teste de usabilidade

O teste de usabilidade do APC buscou avaliar principalmente o desempenho do usuário com o software, o objetivo não era chegar a resultados estatisticamente válidos, mas ter indicações de como melhorar a qualidade de uso da interface (Prates & Barbosa, 2003). Estes testes foram realizados pelo usuário em laboratório onde o avaliador tem um maior controle sobre o ambiente e sobre as atividades do usuário.

As etapas seguidas no processo de teste de usabilidade do APC foram: determinação do objetivo da avaliação, seleção das tarefas, seleção dos usuários participantes, elaboração do material para o teste, execução do teste e análise dos resultados.

Os objetivos do teste de usabilidade do APC foram: Identificar a opinião dos usuários sobre a interface do módulo, bem como as funcionalidades do mesmo e verificar a efetividade do sistema para controle do dispositivo robótico. Por ser um ambiente virtual de aprendizagem e tendo como foco o ensino de linguagem de programação, o APC precisa ser acessível independentemente do grau de conhecimento em ambientes virtuais que um usuário possa ter.

As tarefas selecionadas foram: abrir o arquivo de código criado em uma aula prática anterior; ler as instruções do arquivo e verificar seu entendimento; transferir o código para o robô; verificar se o robô executa as instruções de acordo com o código; e enviar comandos para o robô via porta serial.

Seguindo a recomendação de Nielsen (2000) e Turner et.al (2006), foram selecionados cinco usuários entre os alunos de graduação do Curso de Ciência da Computação da UESB, que cursavam o primeiro semestre e maiores de dezoito anos.

O teste foi realizado individualmente com cada usuário no laboratório do curso e foram utilizados os seguintes recursos: computador com acesso à Internet e rodando programa de log, roteiro do teste do avaliador, tarefas e questionário de avaliação para o usuário, cronômetro, gravador, lápis e caneta. Após o teste, cada usuário foi convidado a responder o questionário composto de duas parte.

As métricas utilizadas para elaboração do questionário e do roteiro de acompanhamento e observações do avaliador foram: satisfação subjetiva, layout gráfico, navegabilidade, terminologia, tempo de execução da tarefa, taxa de finalização da tarefa.

A parte inicial do questionário é composto de cinco questões de múltipla escolha que tem como objetivo conhecer o perfil do usuário quanto ao uso de recursos digitais (iniciante, intermediário, avançado), ao uso de ambientes virtuais de aprendizagem, faixa etária, conhecimento de programação e experiência com robótica.

A segunda parte do questionário foi usado para obter a opinião do usuário sobre o APC, como mostra a Tabela 1. Ele consiste de doze perguntas, sendo as dez primeiras de múltipla escolha e duas abertas. Nas questões objetivas, o usuário especificou o seu nível de concordância com a pergunta utilizando a escala Likert de cinco pontos (1: discordo totalmente; 2: discordo; 3: neutro; 4: concordo; 5: concordo totalmente). As questões foram:

1. Você ficou satisfeito em relação ao uso do sistema?
2. Os layouts das telas foram úteis para a interação no sistema?
3. As informações estavam dispostas de forma organizada?
4. As Mensagens que apareceram nas telas ajudaram a concluir as tarefas?
5. As Instruções para comandos ou funções eram objetivas, sem ambiguidades?
6. As Instruções para correção de erros eram objetivas, sem ambiguidades?
7. O sistema é fácil para retornar a um estado anterior?
8. O sistema é fácil de aprender?
9. Foi fácil lembrar os nomes e usos dos comandos?
10. O sistema possui falhas?
11. Qual a sua impressão sobre as telas do módulo de controle de dispositivos robóticos?
12. O que você modificaria na interface?

Todos os usuários que participaram do teste completaram as tarefas e responderam o questionário. Eles tinham entre dezoito e vinte e cinco anos, nunca tinham usado um ambiente virtual de aprendizagem, consideravam-se iniciante em programação e não tinham experiência com robótica. Dois se consideravam intermediários no uso de recursos digitais e os demais se consideravam iniciantes.

O grau de satisfação dos usuários que testaram o APC foi mensurado utilizando uma abordagem quantitativa que estabelece o ranking médio para cada questão objetiva do questionário. O ranking médio (R) é obtido calculado a média ponderada para cada questão, baseando-se na frequência das respostas, como mostra a Equação 1.

$$R = \Sigma(v_i * f_i)/5 \quad (1)$$

Onde:

R – ranking médio

v_i – valor da resposta i

f_i – frequência da resposta i

Os valores de R maiores que 3 são considerados como concordantes e os valores de R menores que 3, como discordantes. O resultado é mostrado na Figura 15.

Tabela 1 – Opinião dos usuários sobre o APC

Questão	Usuário 1	Usuário 2	Usuário 3	Usuário 4	Usuário 5
1	5	5	4	4	4
2	4	5	4	5	4
3	4	4	4	4	5
4	5	4	4	4	4
5	4	5	4	3	4
6	4	5	4	4	4
7	4	5	4	5	3
8	5	5	4	4	4
9	4	5	4	5	5
10	2	2	2	3	2
11	São de fácil visualização e interação, mas o tamanho da tela que mostra o robô poderia ser expandida.	As telas foram de grande ajuda para que houvesse uma melhor experiência do que estava sendo proposto.	De primeiro acesso, a tela do módulo de controle parece bem acessível para qualquer usuário com pouco conhecimento na área.	Observei, que o mesmo será muito interessante, pois dará ao usuário a oportunidade de ver seu código funcionar na prática, mesmo sem estar próximo.	Fácil de interagir, satisfatório.
12	O sistema demora para responder aos comandos, seria interessante acrescentar algo que aumentasse a velocidade de	A interface possui tudo necessário e é de fácil compreensão para quem a está utilizando, portanto não	Aparentemente, nada.	Alguns botões poderiam ser mais objetivos.	Eu acharia a interface satisfatória, então eu não modificaria nada.

	resposta aos comandos	modificaria o jeito na qual ela se encontra.			

As questões de 1 a 9 obtiveram ranking médio maior igual a 4, que significa que os usuários ficaram satisfeitos com o uso do APC. Os layouts das telas ajudam na interação com o sistema, as informações e mensagens são úteis e estão organizada, as instruções são objetivas, o sistema é fácil de aprender e os comandos são fáceis de lembrar. A questão 10 obteve ranking igual a 2,2, isso revela que o sistema tem poucas falhas.

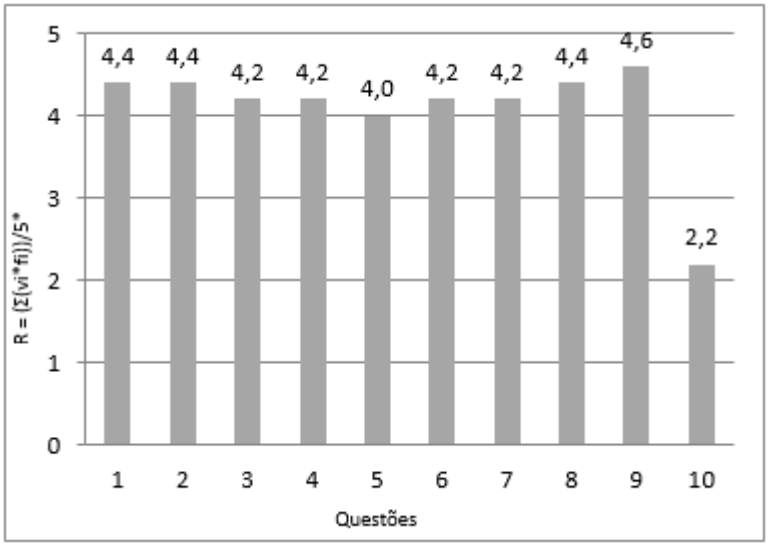


Figura 15 – Ranque médio de satisfação do APC

Apesar da avaliação geral do sistema ter sido muito boa, as respostas abertas e as observações do avaliador indicaram alguns aspectos que precisavam melhorar. Ao entrar no ambiente, o usuário não tem informação de onde ele está. Os usuários tiveram um pouco de dificuldade para identificar qual o botão para compilar o código. Os recursos para retornar o sistema para o estado anterior precisavam ser ampliados. O exemplo de código não foi encontrado facilmente. O sistema apresentou uma falha, pois, ao compilar o código, apareceu uma mensagem de alerta no console, mas o código estava correto. Os usuários tiveram um pouco de dificuldade para encontrar a janela de comunicação serial, e pontuaram a necessidade de melhorar a velocidade de resposta do sistema.

Com base nessas indicações, foram feitas as seguintes alterações no sistema:

- Foi colocado o nome do ambiente no início da página.
- Os textos dos rótulos dos botões de comando foram trocados para ficarem mais objetivos.

- Para facilitar o retorno do sistema ao estado anterior, duas soluções foram implementadas: colocar na barra de ferramenta o botão de retorno do robô a posição inicial e disponibilizar a opção de desfazer (crtl+z) no editor de código.
- Para facilitar a localização de exemplos, foi criado o botão “Exemplos” na barra de ferramentas.
- A mensagem de alerta enviada pelo sistema não é um erro, mas pode causar frustração no usuário. Este problema foi corrigido no servidor do laboratório.
- Para tornar a janela de comunicação serial mais visível, ela foi colocada ao lado da aba do console. A aba de comunicação serial é automaticamente habilita ou desabilita para indicar se ela pode ou não ser usada.
- Para aumentar a velocidade de resposta do sistema, duas medidas foram tomadas: melhorar a configuração do servidor virtual LARA e conectá-lo diretamente ao switch central da rede da UESB. Apesar dessas medidas, é importante observar que, em um sistema web, a velocidade depende também da latência da rede do usuário.
- Como o APC é integrado ao ambiente de aprendizagem Moodle, não é necessário manter a área de material didático.

A Figura 16 mostra a nova interface do APC com as alterações descritas.

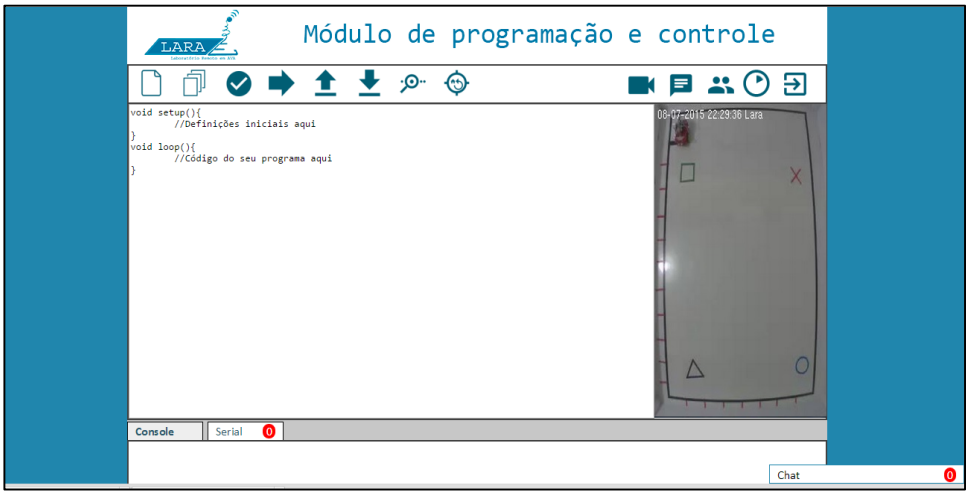


Figura 16 – Nova interface do APC

5 LaraPC – integração das ferramentas do protótipo da arquitetura

Este capítulo aborda a integração das ferramentas apresentadas no capitulo 4 e a GUI do LaraPC. Primeiro é apresentada a visão geral do sistema, em seguida os detalhes de implementação e, por fim, a interface do usuário. Sabe-se, entretanto, que é possível alcançar os objetivos da arquitetura através de implementações alternativas.

5.1 Visão geral do LaraPC

O LaraPC é utilizado para verificar a integração de várias ferramentas para o ensino de programação de forma a validar a arquitetura proposta. A estrutura utilizada para o desenvolvimento permite ao usuário acessar o ambiente para estudar programação, realizar atividades práticas e interagir com os outros usuários.

A Figura 17 apresenta a visão geral do LaraPC. O ambiente é acessado através do AVA Moodle que compõe o módulo de estudo. O Moodle possui funções de gerenciamento de usuários, de cursos, de objetos de aprendizagem, sistema instrucional de ensino-aprendizagem, entre outras. Para realizar as atividades práticas, o usuário deve realizar a análise do problema, codificação e teste de soluções acessando a área de realização de tarefas que é formado pelos módulos de análise, de codificação e de experimentação. O módulo de análise é composto pelo Etherpad Lite que permite criar e editar planos para resolução de problemas. Os módulos de codificação e experimentação foram agrupados na área de desenvolvimento. Nesta área é possível implementar algoritmos através do IDE online do APC, mesmo sem acessar o experimento, e testar e validar a solução através do robô remoto. O processo de colaboração é baseado em *party* que permite que grupos dinâmicos sejam formados durante a realização de atividades práticas e que os componentes do grupo se comuniquem via chat. A integração entre as funções e a interface do usuário é realização por socket, banco de dados compartilhado e solicitações HTTP.

5.1.1 Integração da base de dados

A base de dados do LaraPC é composta pela integração dos bancos de dados das ferramentas Moodle, APC e gerenciamento de colaboração. A Figura 18 mostra parte deste banco de dados. As tabelas user, role_assignments e role são do Moodle. Cada usuário do sistema possui um id único, um papel (role), que pode ser aluno, professor, tutor ou administrador, e suas permissões. As tabelas do APC guardam os dados referentes as reservas do experimento e as operações realizadas durante uma sessão como: iniciar sessão, finalizar sessão, compilar código, enviar código para o experimento, sair do laboratório, etc. Os dados da colaboração entre os usuários são armazenados basicamente nas tabelas *party*, integrante e notificação. Uma *party* (ou grupo) possui um identificador

único (id), um líder, data e hora em que foi criada. As informações sobre convites e expulsão das *parties* são armazenadas na tabela de notificação. Os usuários que participam do grupo (*party*) são os seus integrantes. Cada integrante possui registros na *party* para guardar quando entrou, quando saiu ou se foi expulso do grupo.

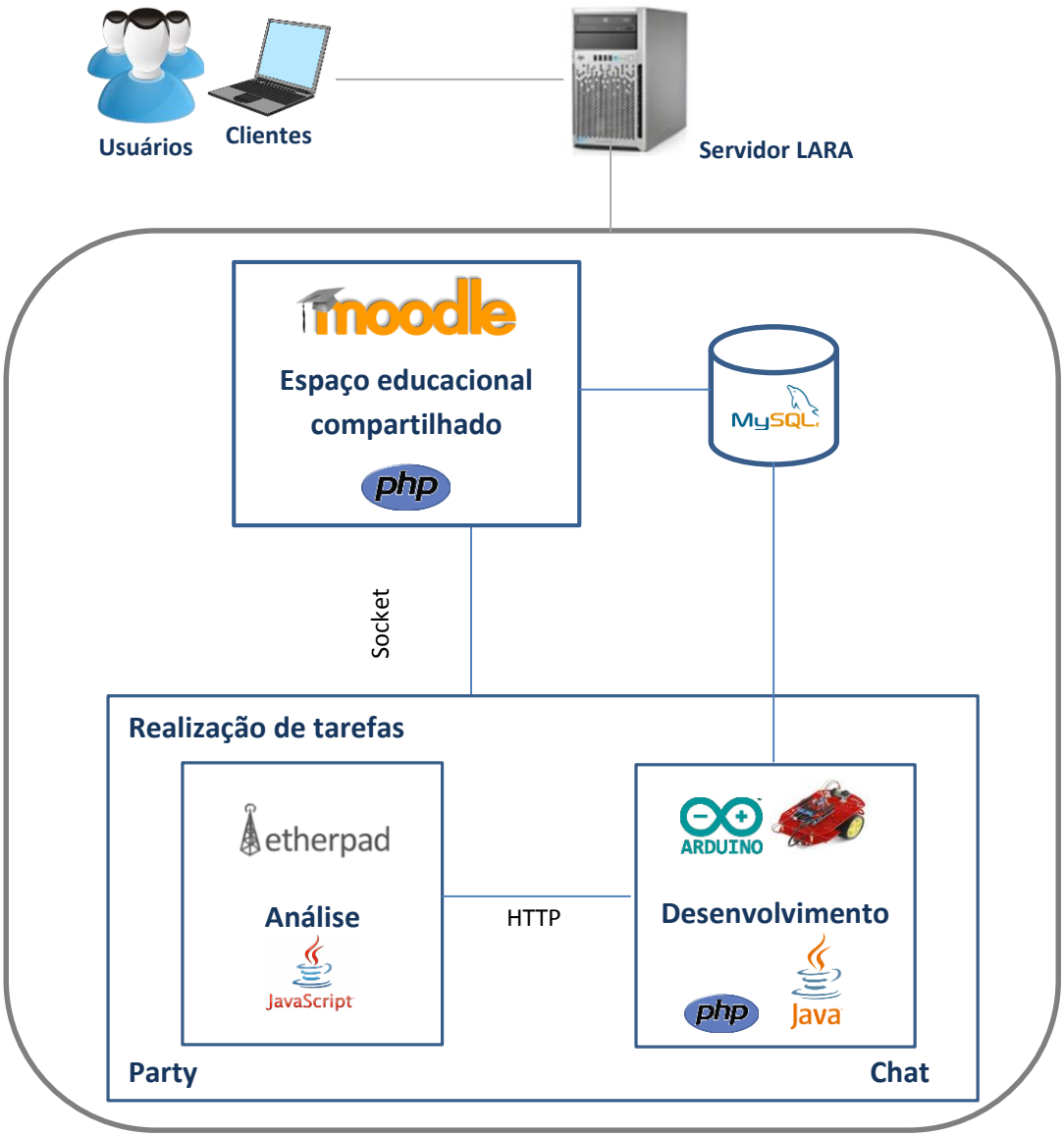


Figura 17 – Componentes do LaraPC

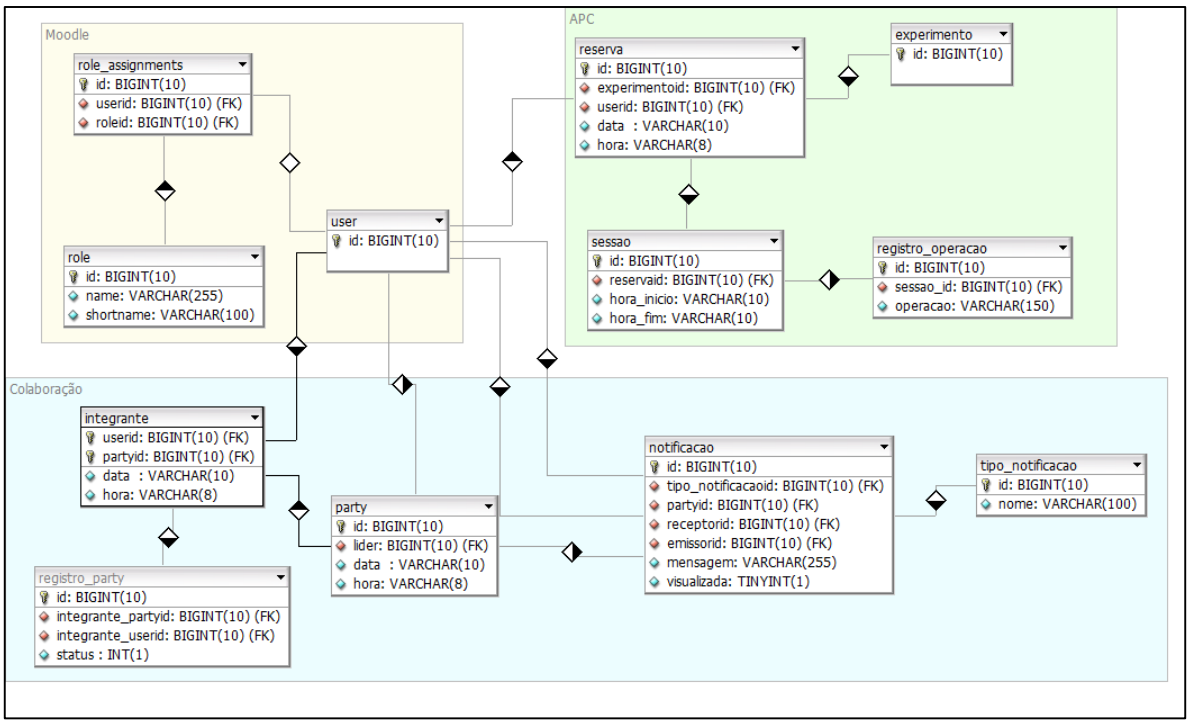


Figura 18 – Banco de dados do LaraPC

A integração da base de dados permitiu, entre outras coisas, utilizar o controle e o gerenciamento de usuários presentes no Moodle para controlar o acesso ao módulo de experimentação. Para acessar um experimento, é necessário efetuar uma reserva ou agendamento do experimento. O Moodle guarda todos os dados do usuário na variável global $USER. O APC consulta o Moodle e solicita o id e o nome do usuário. O código da Figura 19 mostra como é realizada a verificação da permissão de acesso do usuário a um experimento específico. O id do usuário é consultado e, então, é verificado qual a role deste usuário. O tipo de usuário é armazenado na sessão para verificar permissões de acesso ao experimento. Se o usuário for tutor, professor ou administrador, ele tem permissão de acesso liberado sem precisar de reserva. Se o usuário for aluno, é necessária verificar se ele tem reserva.

```
<?php
    require('conectar.php');
    require_once("../config.php") ;
    require_login();

    /* DEFINIÇÕES DE ID */
    $id = (int)$USER->id;
    $_SESSION['id-user'] = $USER->id;
    if($USER->id==1){
        header("Location: ../");
        exit;
    }
    /* NOME */
    $_SESSION['nome-user'] = $USER->firstname;
    /*BUSCA A ROLE DO USUÁRIO NO CURSO EM QUE ESTE ESTÁ  ATUALMENTE*/
    preg_match_all("/[a-zA-Z]*=(?<tipo>\d*)/",
get_user_roles_in_course($USER->id,2),$valores);
    /* GRAVA NA SESSÃO */
    $_SESSION['tipo-user'] = $valores['tipo'][2];
    $tipo = $valores['tipo'][2];
```

Figura 19 - Permissão de acesso a um experimento

Os documentos gerados durante uma *party* no modulo de análise e na codificação são armazenados no banco de dados do Etherpad e a chave de cada documento (pad) é montada a parir do id do usuário ou do id da *party*.

5.1.2 Gerenciamento de *party*

O gerenciamento do processo de colaboração do LaraPC é baseado na ideia de *party* (grupo) de jogos online e possui as seguintes características:

- Uma *party* é criada ou instanciada quando o primeiro usuário é convidado a participar de uma atividade, isto é, quando um convite é enviado;
- O usuário que envia o primeiro convite é o líder da *party*. A liderança pode ser passada para outro usuário da *party* a qualquer momento;
- Os usuários que participam da *party* são os seus integrantes;
- Um usuário só pode participar de uma *party* por vez;
- Um *party* pode ter no máximo 5 (cinco) integrantes;
- Só o líder da *party* pode: convidar usuário, excluir integrante, passar a liderança para outro integrante, compilar código e, caso tenho reserva, enviar código para o experimento;
- Durante uma *party,* todos os integrantes podem colaborar na análise e no desenvolvimento podendo ir de uma área a outra quando quiser;

- Para acessar o experimento durante uma *party*, é necessário que um integrante tenha uma reserva e este integrante deve o líder da *party* durante o período da reserva;
- A comunicação entre os integrantes da *party* é realizada através do chat, independente do modulo que o integrante esteja;
- Uma *party* termina quando o último integrante sai da *party*.

O gerenciamento das *parties* foi desenvolvido em Socket implementado em NodeJs. Para iniciar uma *party*, é necessário que o usuário convide outro usuário para participar do grupo. A Figura 20 mostra as atividades realizadas pelo usuário e pelo sistema para enviar um convite. O usuário deve entrar no ambiente destinado a realização de tarefas e selecionar a opção “Convidar para o grupo”. Neste momento o sistema mostra a lista dos usuários separados por usuários online e usuários off-line, onde deve ser selecionado o usuário que será convidado, isto é, o destinatário do convite. O sistema também verifica se o usuário que está fazendo o convite, o emissor do convite, faz para de uma *party* e se é seu líder, se as duas condições forem verdadeiras o convite é criado e enviado para o destinatário e no chat da *party* aparece uma mensagem informando o nome do convidado.

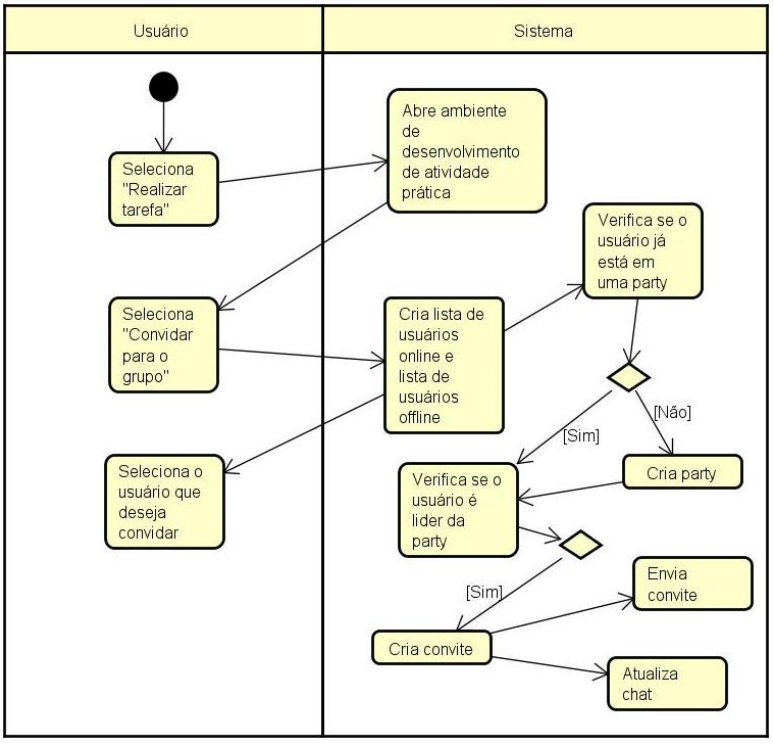


Figura 20 – Enviar convite para participar de um grupo

Se o emissor ainda não está em um grupo, uma nova *party* tem que ser criada. O processo de criação é mostrado na Figura 21. O sistema cria um objeto *party,* armazena

seus dados no banco de dados e coloca o usuário emissor como líder. Além disso, o sistema cria e criptografa a chave da *party*, esta chave é importante pois é usada para formar o endereço das pads do Etherpad. Depois as informações da *party* são enviadas para o usuário e a GUI atualizada.

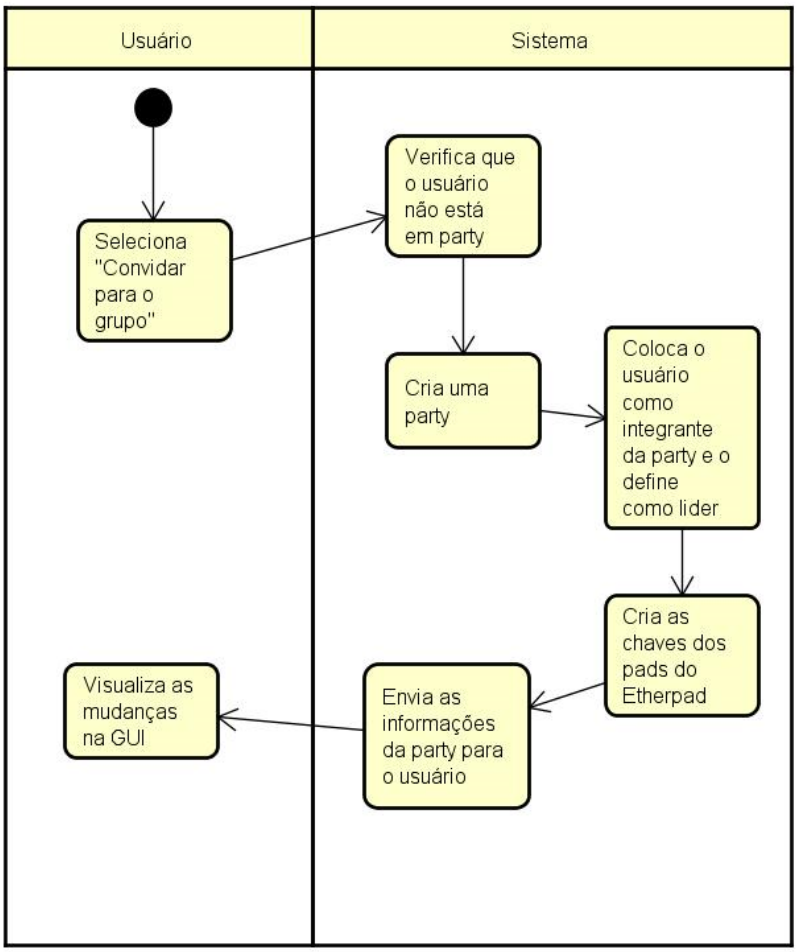


Figura 21 – Criar um grupo (ou party)

Após o sistema enviar o convite, o usuário destinatário o recebe e decide se vai aceitá-lo ou recusá-lo. Se aceitar, o sistema deve verificar se o convite ainda é válido, isto é, se o grupo ainda existe, e se o destinatário está ou não em outro grupo. Se o convidado estiver participando de outro grupo, ele é automaticamente removido e inserido na lista de integrantes da *party* que acabou de aceitar. Uma mensagem informando que o destinatário entrou no grupo aparece no *chat* e a GUI do destinatário é atualização com as pads da *party*, isto é, o texto do módulo de análise e o código do modulo de codificação são compartilhados com o novo integrante do grupo. Este processo é mostrado na Figura 22.

5.1.3 Integração com Etherpad

No LaraPC, o Etherpad é um serviço usado no módulo de análise e também no editor de código do APC. Em ambos os casos, o acesso é realizado através de iFrame, pois permite a integração entre as diferentes aplicações web de forma segura e independente.

Quando um usuário entra na área de realização de tarefas do LaraPC, são carregadas automaticamente uma pad para a área de desenvolvimento, e outra pad para o módulo de análise. As chaves das pads são formadas a partir do id do usuário ou da chave da *party*. Todas as vezes que um usuário está sem grupo, as pads carregadas são definidas pelo seu id. Quando o usuário está participando de uma *party*, as pads são formadas com base na chave da *party*.

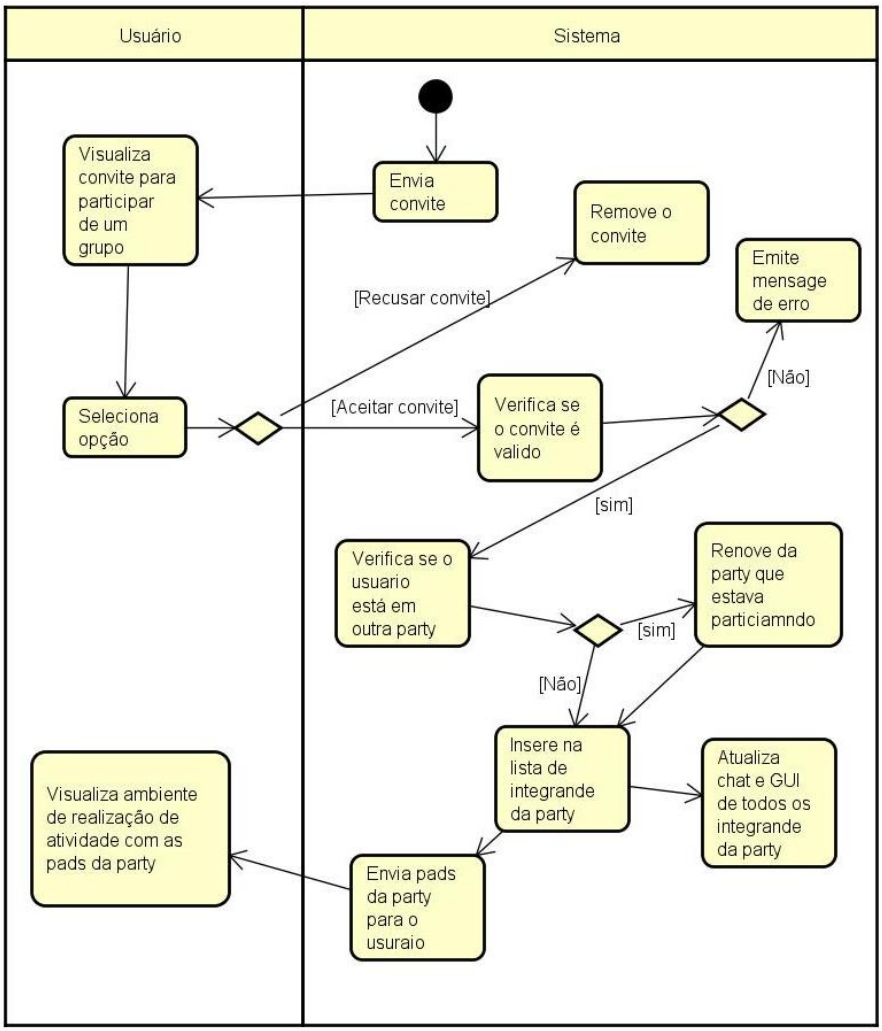


Figura 22 – Aceitar convite para entrar em um grupo

Quando uma *party* é criada (Figura 21), é formada a chave da *party* (`Key`) com base no seu id: `sha1("party"+id+"lara")` e também a chave das pads (Figura 23).

```
var chave = sha1('PARTY'+party.key+'LARA');
cliente.emit ('confirmaParty', party, chave, sha1(chave));
```

Figura 23 – Chaves das pads

A variável `chave` é a chave da pad do editor de código do APC e a pad da área de análise é a mesma chave criptografada novamente. O mesmo princípio é usado com o id do usuário.

A comunicação com o Etherpad é realizada através da API definida pela própria ferramenta, por meio de solicitações HTTP. Para cada solicitação, além dos parâmetros específicos, é necessário enviar os atributos `apikey` e `padID`. O atributo `apikey` é usado para garantir que apenas usuários autorizados possam utilizar a API. O atributo `padID` é a chave única da pad com a qual se deseja trocar informações. A figura 24 mostra o código para realizar uma atualização no texto da pad.

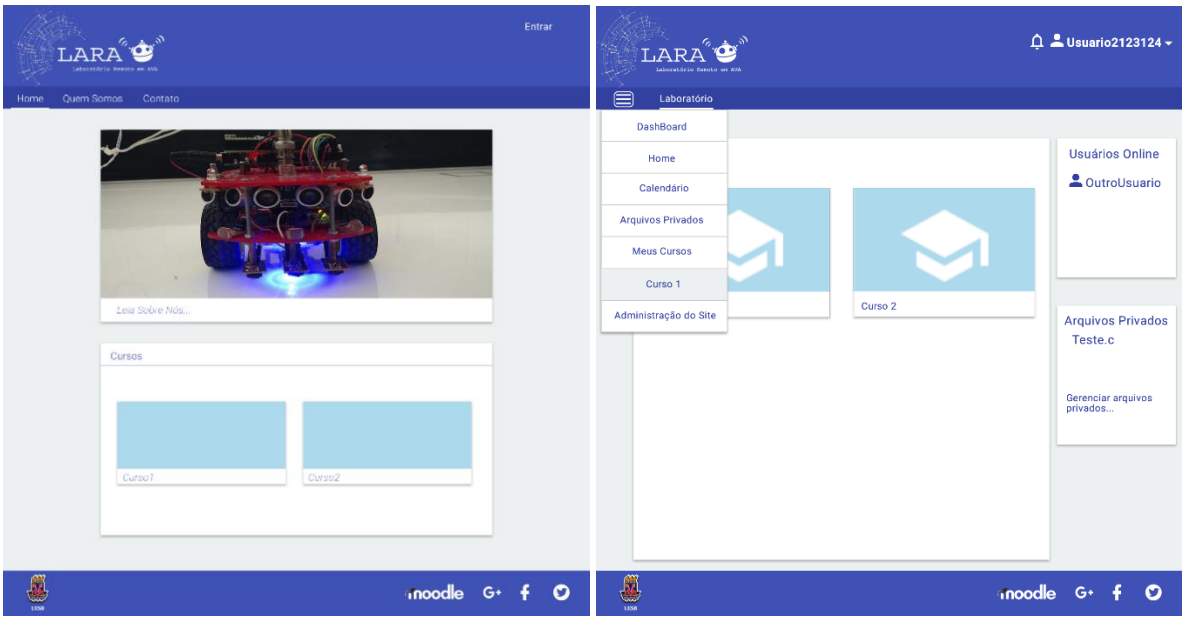
```
$.ajax({
      url: "http://localhost:3002/api/1/setText",
      dataType: "json",
      type: 'get',
      data: {apikey:chaveDaAPI,padID:idDaPad,text:data.codigo}
});
```

Figura 24 – Exemplo de comunicação com o Etherpad

A `url` define o servidor onde está sendo executada a aplicação, a porta, a versão da api (api/1/) e a operação desejada (setText). O `dataType` é definido como json pois a solicitação irá retornar um objeto Javascript como feedback da operação. O `type` de solicitação é definido como get. No campo `data`, são passados os parâmetros exigidos pela operação setText: apikey, padID e o texto que deve ser escrito na pad.

5.2 Interface do LaraPC

A interface do LaraPC utiliza os mesmos princípios aplicados na GUI do APC para tonar o ambiente agradável e contínuo, sem ruptura para o usuário. A Figura 25 mostra a página inicial do sistema (Figura 25a), e a página depois de logar (Figura 25b). No sistema, o usuário pode, entre outras coisas, acessar seus cursos e seus arquivos privados através do menu principal. Para acessar a área de realização de tarefas, o usuário deve clicar em Laboratório, na barra de ferramentas, e escolher “Reserva” ou “Acessar Laboratório”. Para fazer uma reserva, o aluno deve escolher o experimento, no caso L1R2, e fornecer o dia e o horário para a reserva.



(a) (b)

Figura 25 – Página inicial do LaraPC

Ao escolher “Acessar Laboratório”, o usuário é direcionado para o ambiente de realização de atividade mostrado na Figura 26. Neste ambiente é possível acessar a área de análise e a área de desenvolvimento. O ambiente possui: menu superior(1), abas (2), menu da aba selecionada (3) e área de trabalho (4). A barra de ferramenta superior possui o botão “Convidar para o grupo” do lado esquerdo, que permite convidar colegas para realizar a atividade em grupo, e o botão do lado direito para indicar se o aluno está ou não em uma sessão de laboratório. O botão fica vermelho para indicar que o laboratório remoto está off-line e verde quando o laboratório está online. O aluno pode selecionar a aba Desenvolvimento ou a aba Análise, a aba selecionada fica cinza.

Ao clicar no botão “Convidar para o grupo” aparece uma janela com a lista dos usuários online e dos off-line (Figura 27a) com o link “Convidar” ao lado dos nomes. O usuário convidado recebe o convite mostrado na Figura 27b.

Quando o convidado aceita o convite, a sua GUI é alterada (Figura 28). As pads do grupo são carregadas em seu navegador. O menu superior mostra um boneco para cada integrante do grupo, o botão do chat e o botão para sair do grupo. Na parte inferior da janela aparece o “Chat”. Ao passar o mouse sobre um boneco aparece o rótulo com nome de membro que ele representa, o boneco com a coroa representa o líder do grupo.

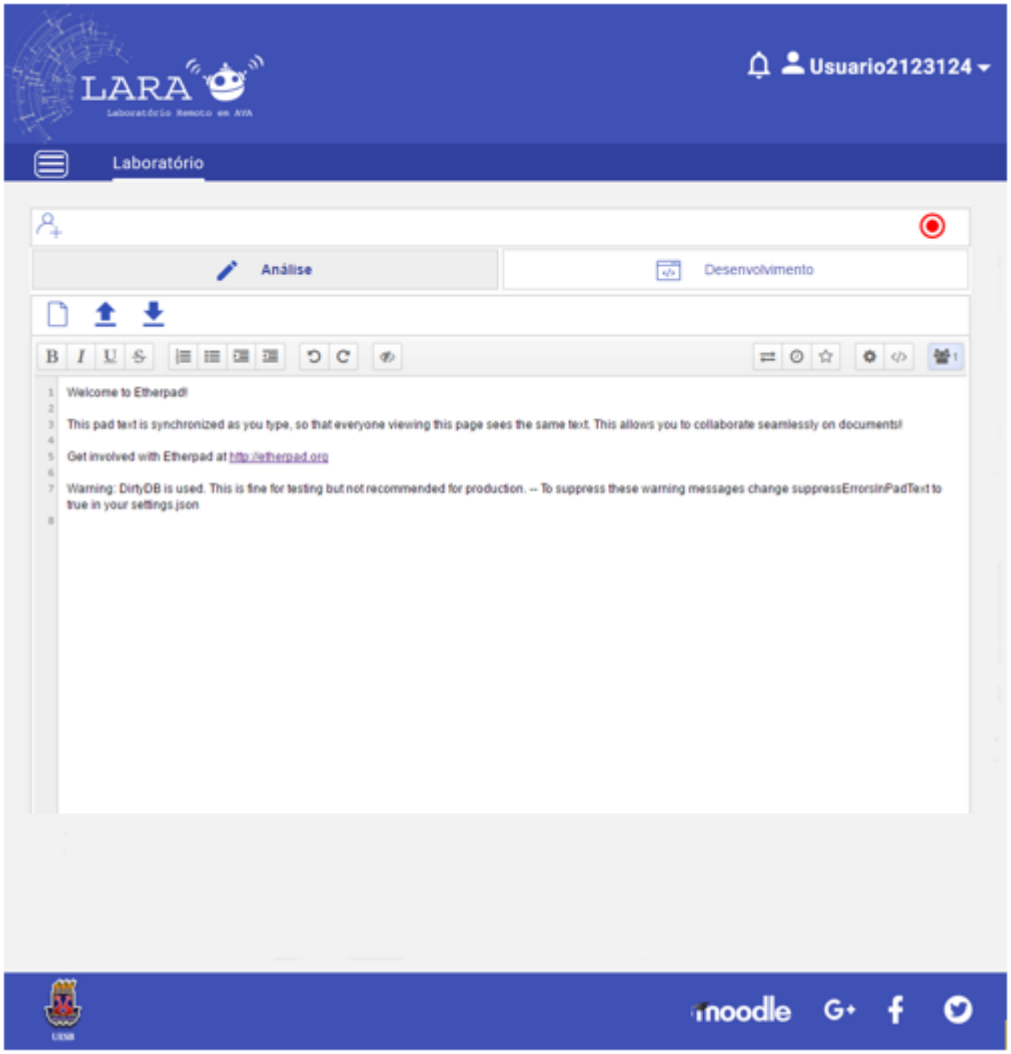


Figura 26 – Área de desenvolvimento de atividade

Figura 27 – Lista de usuários online e off-line e Convite recebido

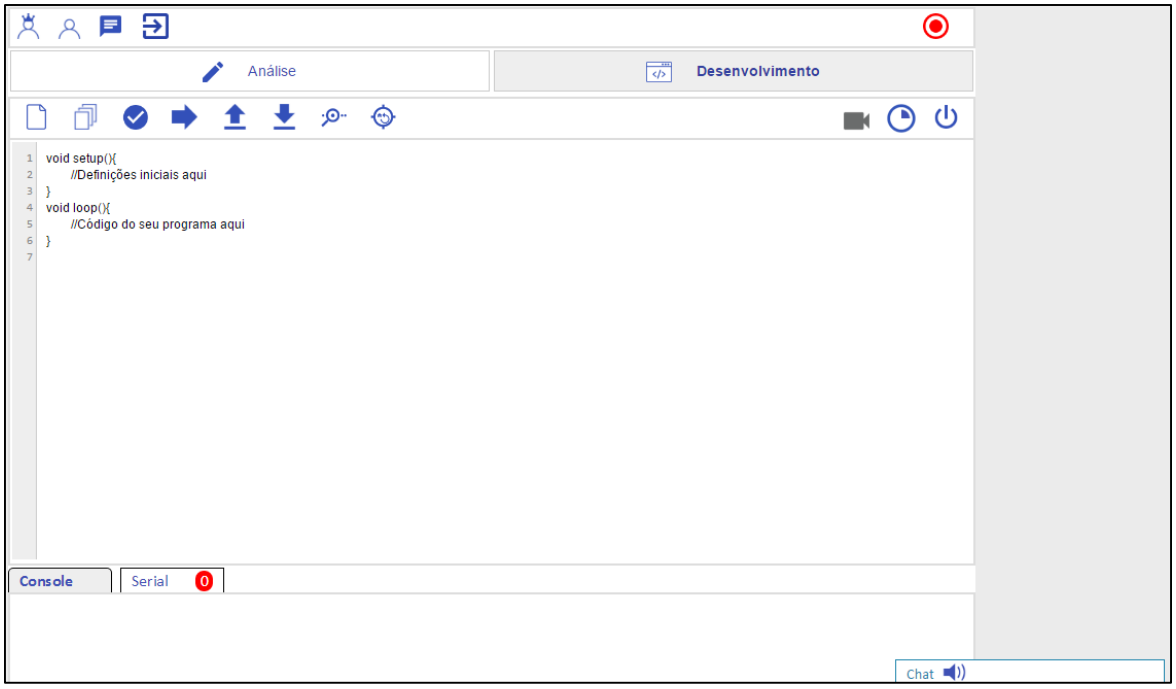


Figura 28 – Interface do convidado

A interface do líder do grupo é mostrada na Figura 29. Ele possui o botão para convidar outros usuários e ao clicar sobre um membro, ele pode passar a liderança ou expulsar do grupo.

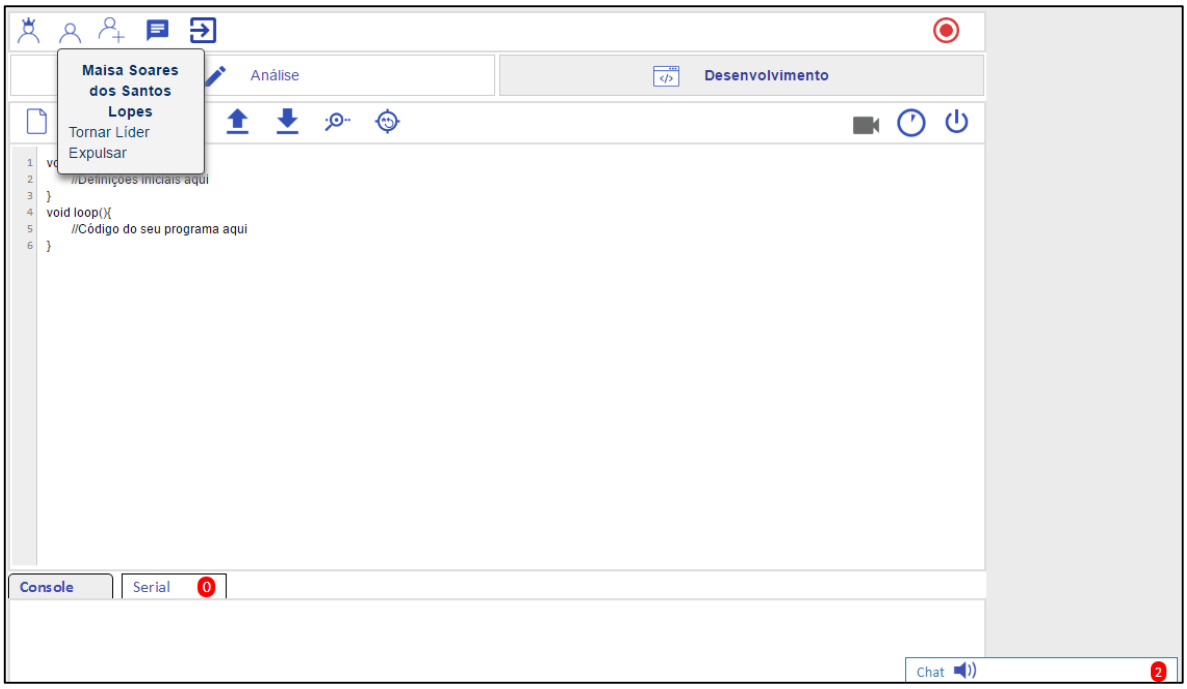


Figura 29 – Interface do líder

6 Validação

Este capítulo apresenta a validação do protótipo da arquitetura de um ambiente colaborativo de ensino de programação com suporte a experimentação remota. A avaliação foi realizada em duas etapas, em cada etapa foi utilizada uma metodologia distinta. Primeiro foi realizada a avaliação do APC através de um curso de programação para validar a ferramenta no ensino de programação. A segunda avaliação visa verificar a usabilidade do LaraPC através do método de inspeção voltado a ambientes colaborativos de ensino aprendizagem.

6.1 Avaliação do APC

Para avaliar o APC foi criado o Curso LARA de Introdução a Programação (CLIP). A descrição do curso, sua aplicação e resultados são descritos a seguir.

6.1.1 CLIP – Curso Lara de Introdução à Programação

O CLIP é um curso semipresencial com objetivo de ensinar os conceitos básicos de programação utilizando o robô móvel remoto, L1R2, para contextualizar as aplicações dos conceitos ensinados. A Figura 30 mostra o curso criado no Moodle.

Figura 30 - CLIP no Moodle

O curso consiste em quatro módulos, ao final de cada um os alunos devem resolver um problema:

- Módulo I - Comandos de sequência e variáveis
  Problema: Mover o robô na arena (direita, esquerda, frente, ré)
- Módulo II - Comandos de seleção e repetição

Problema: Mover o robô durante um período de tempo e/ou distância dentro de um laço

- Módulo III - Funções
  Problema: Detectar uma linha e executar o comando (parar, voltar a uma determinada posição, tomar decisão)
- Módulo IV - Modularização
  Problema: Usar as funções desenvolvidas para construir um seguidor de linha

As tarefas têm complexidade progressiva, à medida que os alunos avançam no curso, as atividades se tornam mais difíceis. A estratégia aplicada para resolver as atividades é "dividir para conquistar" onde a solução de problemas básicos contribui para a construção do seguidor de linha. A proposta usa a solução de problemas seguindo as seguintes etapas: compreender o problema, elaborar um plano, executar o plano e testar a solução. Em cada atividade, o estudante deve responder as seguintes questões:

Compreender o problema

a) Qual é o problema?
b) Existem restrições? Quais?
c) Tem alguma palavra ou termo desconhecido? Quais?

Traçar um plano

a) Já viu este problema antes ou algum similar? Se sim, pode usar o mesmo método para revolver este novo problema?
b) Pode dividir este problema em partes? Quais?
c) Quais as entradas do problema?
d) Qual a saída esperada do problema?
e) Quais ações são necessárias para resolver o problema?
f) Escreva um algoritmo para resolver o problema.

Executar o plano

a) Escolha um conjunto de entrada e siga os passos definidos no algoritmo. A saída encontrada é a desejada?
b) Quais comandas de linguagem de programação podem realizar as ações definidas no algoritmos?
c) Escreva o código na linguagem de programação.

Comprovar os resultados

a) Compilar o código. Está correto?

b) Se o programam estiver sintaticamente correto, enviar o código para o robô. O robô executou os comandos da forma esperada? Se não, quais foram os problemas?

No APC, para testar a solução, o aluno compila o código, se estiver sintaticamente correto ele pode ser enviado para o robô, caso contrário, o erro é mostrado na aba "Console". Quando o código é enviado para o L1R2, sua execução é visualizada através da câmera. Em caso de erro semântico, o aluno pode remodelar a solução e testá-la novamente. Durante a execução do experimento, através da aba “Serial”, o aluno pode enviar e receber dados do robô, tais como, atribuir valor a uma variável, ler o valor de sensores e de variáveis. Este recurso ajuda no processo de compreensão semântica do problema e fornece uma depuração de código simples, uma vez que o IDE Arduino não possui depurador.

Para o curso, foi implementada a biblioteca LARA.h que permite controlar e manipular o L1R2 com comandos de alto nível omitindo os detalhes da construção do robô. Por exemplo, para mover o robô, os seguintes comandos são usados: frente, direita, esquerda, ré e para. Figura 31 mostra a assinatura destas funções.

```
                        /** Funções para mover o robo **/
// SPEED_DEFAULT = 5 cm/seg
// DISTANCE_DEFAULT = 10 cm
// ANGLE_DEFAULT = 90 degrees

// Move o L1R2 para frente
void frente(int distancia= DISTANCE_DEFAULT, int velocidade = SPEED_DEFAULT);
void frente (float tempo, int velocidade = SPEED_DEFAULT);

//Move o L1R2 para traz (re)
void re(int distancia= DISTANCE_DEFAULT, int velocidade = SPEED_DEFAULT);
void re(float tempo, int velocidade = SPEED_DEFAULT);

// Move o L1R2 para direita
void direita (int anglo = ANGLE_DEFAULT);

//Move o L1R2 para esquerda
void esquerda(int anglo = ANGLE_DEFAULT);

// Faz o L1R2 parar
void parar();
```

Figura 31 – Assinatura das funções de movimento do L1R2

A biblioteca LARA.h facilita o uso de APC pois os comandos básicos de manipulação são simples e fáceis de lembrar. Os alunos podem se dedicar a aprender conceitos de lógica de programação, estratégia de resolução de problemas e boas práticas de programação, como o uso de comentários apropriados e padrão de codificação. Além disso, o uso do robô torna mais evidente a necessidade de depurar e testar os códigos para melhorar a qualidade do software desenvolvido.

6.1.2 Aplicação da avaliação

Para validar os benefícios do uso do APC com uma ferramenta educacional foi realizada uma oficina de programação com robô móvel. O objetivo da oficina foi avaliar a perspectiva de aprendizagem de programação e a experiência de uso do APC. Para isto, foram utilizados três instrumentos: a) questionário que mede a satisfação do aluno utilizando o sistema APC, b) coleta de dados de acesso ao ambiente e c) resultado acadêmico da oficina comparado ao resultado na disciplina Algoritmo e Programação I.

Foram aplicados dois questionários: 1) levantamento dos dados pessoais dos participantes e experiência com programação, robótica e laboratório remoto e 2) questionário com escala de Likert de cinco pontos (de 1 - discordo totalmente a 5 - concordo totalmente) para analisar a qualidade do ambiente, as percepções globais dos participantes sobre o APC e a qualidade das atividades realizadas no ambiente. O primeiro questionário foi aplicado no início da oficina quando os participantes preencheram uma ficha informando dados pessoais e experiências e o segundo ao final da oficina quando preencheram o questionário online.

Os dados de acesso foram coletados no banco de dados e através da análise dos relatórios do sistema.

A aprendizagem foi avaliada através de atividades e um desafio no final da oficina. A escala de pontuação utilizada foi de 0 a 10. O resultado foi comparado ao resultado em Algoritmos e Programação I (AP I), disciplina obrigatória do curso de Ciência da Computação da UESB.

Os alunos do primeiro semestre do curso de Ciência da Computação da UESB foram convidados a participar de uma oficina de extensão. A oficina aconteceu antes do início da disciplina AP I e contou com 20 participantes, sendo 3 do sexo feminino e 17 do sexo masculino. A idade dos alunos variou entre 17 e 24 anos. Apenas 5 participantes declararam ter alguma experiência com programação, mas nenhum tinha experiência com robótica nem com laboratório remoto.

Abaixo são analisados os resultados acadêmicos dos alunos e a percepção dos alunos sobre o APC.

6.1.3 Análise do resultado acadêmico

A média (M) da pontuação obtida pelos alunos na oficina foi 7,9 e o desvio padrão (DP) 1,9. A média indica que os alunos tiveram um desempenho satisfatório. Eles compreenderam e empregaram corretamente os comandos de movimento do robô (frente(), direita(), esquerda() e re()), além de entender a importância da sequência lógica de comandos. Também mostraram entendimento progressivo do conceito de variáveis. Nos primeiros programas foram usadas apenas variáveis explicitadas no enunciado do problema, mas à medida que a oficina foi avançando o conceito de variável foi aplicado adequadamente nas diversas atividades. Os comandos de seleção e repetição não eram familiares para a maioria dos alunos. As respostas das atividades mostram que eles entenderam os problemas e a necessidade de aplicar seleção e/ou repetição, mas alguns tiveram dificuldade para empregar corretamente estes comandos na programação. A maioria dos participantes também aplicou chamada de função e passagem de parâmetros corretamente.

O problema que mais desafiou os alunos foi implementar o seguidor de linha. Esta atividade foi realizada em grupo de quatro alunos. Apenas uma equipe completou a tarefa corretamente. As demais equipes escreverem códigos sintaticamente corretos, mas tiveram dificuldade para definir a condição do laço inicial do programa, pois requer entendimento sobre o uso de sensores. Além disto, os alunos tiveram dificuldade para avaliar expressões condicionais que utilizavam os operadores and/or. É importante ressaltar que nenhum membro da equipe que implementou o seguidor de linha tinha experiência previa com programação, mas foi a equipe com maior tempo de acesso ao APC.

O conteúdo apresentado na oficina corresponde ao conteúdo da primeira avaliação da disciplina Algoritmos e Programação I (AP I). A tabela 2 resume a pontuação obtida pelos alunos nesta avaliação. A disciplina tinha 31 alunos, destes 18 participaram da oficina (APC) e 13 não participaram (não APC). Pode ser observado que a pontuação dos alunos que fizeram a oficina é maior do que dos demais. Além disto, o desvio padrão das notas do grupo do APC é menor do que o grupo que não fez a oficina. Este fato indica que o desempenho acadêmico dos alunos do APC foi mais homogêneo.

Tabela 2 – Comparação do resultado acadêmico

	Nº	M	DP
APC	18	7,2	2,9
Não APC	13	5,8	4,4
Total	31	6,6	3,7

Estes resultados são interessantes, porque podem significar que a introdução de APC em um ambiente pedagógico tem potencial para envolver mais os alunos na aprendizagem de programação.

6.1.4 Análise de satisfação

Em termos gerais, os dados da pesquisa mostram que o APC foi avaliado muito positivamente pelos alunos. A Tabela 3 resume os resultados da pesquisa de opinião. Os alunos gostaram de usar o APC (90% - questão 2), eles acharam que controlar um robô móvel remoto utilizando um compilador online tornou a aprendizagem de programação mais interessante (100% - questão 3), além de considerar seu controle e manipulação fáceis (65% - questão 4). Isto demostra que o ambiente é fácil de usar, promoveu satisfação do usuário, o que garante que a interface satisfaz critérios de usabilidade.

Algumas questões abordaram o interesse do aluno em usar o APC no processo de aprendizagem. Os participantes externaram o desejo de usar o APC em seus estudos individuais (80% - questão 6), em disciplinas da graduação (80% - questão 7) e para aprender outros conteúdos além de programação (70% - questão 10). Estes resultados mostram que os alunos estão receptivos a usar o APC como ferramenta que promova a aprendizagem de diversos conteúdos relacionados a computação em um curso de graduação.

Outro elemento avaliado na pesquisa foi as atividades realizadas no APC. Os alunos consideraram as atividades interessantes (100% - questão 1), acessíveis (70% - questão 5), desafiadoras (95% - questão 8) e motivadoras (75% - questão 9). A partir destes dados, pode-se deduzir que os aluno se sentiram desafiados com as atividades propostas. Aliar uma ferramenta pedagógica com atividades desafiadoras é importante para promover a aprendizagem. Como os participantes não tinham experiência com programação, as atividades foram pensadas para manter a motivação. Com alunos mais avançados é necessário oferecer também atividades com níveis de dificuldade compatível com suas habilidades para aumentar a motivação e eficácia da aprendizagem.

A pesquisa também mostrou que os alunos tiveram problemas para realizar as atividades no ambiente (35% - questão 11 e desvio padrão maior igual a 1.0 nas questões 4, 5 e 6). Foi relatado que algumas vezes depois de enviar o código para o robô, o console era atualizado com a mensagem “Compilando e Enviando” e nada mais acontecia. Este dado é importante pois compromete a confiança no sistema. Para uma aprendizagem efetiva, é necessário que o sistema funcione corretamente e responda conforme o esperado. Erros recorrentes de funcionamento frustram os usuários e prejudicam o processo de aprendizagem. Este problema técnico foi resolvido substituindo o hardware do servidor de laboratório.

Outro dado importante, mostrado pela pesquisa, foi que os alunos utilizaram o processo de resolução de problema sugerido: entender o problema, propor, implementar e testar a solução (95% - questão 12), e afirmaram que os passos ajudaram na compreensão do problema (95% - questão 13). Isto indica que os módulos de análise e desenvolvimento da arquitetura podem ser úteis no processo de aprendizagem de programação.

Tabela 3 – Avaliação geral do APC

Questões	5	4	3	2	1	M	DP
1 As atividades do curso ajudaram a aprender programação de computadores	70%	30%	0%	0%	0%	4,7	0,5
2 Eu gostei de explorar o Ambiente de Programação e Controle do LARA	50%	40%	10%	0%	0%	4,9	0,7
3 Usar o APC fez a experiência de aprendizagem mais interessante do que uma aula regular em sala	75%	25%	0%	0%	0%	4,7	0,5
4 Eu achei o Módulo de Programação e Controle fácil de manipular / trabalhar	40%	25%	30%	5%	0%	3,9	1,0
5 Ter acesso a atividades online tornou mais fácil encontrar tempo para realizar minhas atividades práticas de programação	45%	25%	20%	10%	0%	4,00	1,1
6 Eu estou disposto a usar o APC para aprender sobre programação utilizando como meio de estudo individual	50,0%	30%	15%	0%	5%	4,2	1,1
7 Gostaria que o ambiente LARA fosse utilizado nas disciplinas da minha graduação	45%	35%	15%	5%	0%	4,2	0,9
8 Usar programação para interagir com robô móvel remoto tornou as atividades mais desafiadoras	70%	25%	5%	0%	0%	4,6	0,6
9 As atividades me deixou mais interessado em aprender o conteúdo do que eu estava antes do curso	55%	20%	25,0%	0%	0%	4,3	0,9
10 Eu estou interessado em ver diferentes conteúdos de aprendizagem no APC	50%	20%	30%	0%	0%	4,2	0,9

11	Eu tive problemas para usar o APC para completar as atividades	20%	15%	50%	15%	0%	3,4	1,0
12	Para resolver a atividades eu utilizei os seguintes passos: entender o problema, propor uma solução, implementar a solução e testar a solução	50%	45%	5%	0%	0%	4,5	0,7
13	Estes passos ajudaram na compreensão do problema	45%	50%	5%	0%	0%	4,4	0,6

A tabela 4 mostra o resumo da aquisição dos conceitos pelos alunos com uso do APC durante a oficina. Os dados demostram que houve uma boa assimilação sintática dos comandos, pois menos de 50% dos códigos continham erros e estes foram diminuindo com o uso do ambiente. Foram compilados 131 códigos e 348 códigos foram enviados para o robô. O número de código enviado é maior porque os aluno utilizaram pouco a função compilação, pois essa etapa é realizada todas as vezes que um código é enviado para o robô (ver seção 4.4.3). Foram encontrados 97 arquivos com erro de compilação, sendo o erro mais comum a falta de ponto e vírgula (;) no final dos comandos. Isto mostra a necessidade de abordar e enfatizar as boas prática de programação. Apenas 12 arquivos tinham os nomes dos comandos escrito errados, o que mostra que os comandos foram fáceis de assimilar. Os outros erros foram relacionados a: falta de chaves, erros com parênteses, utilização de caracteres inválidos, declaração das funções loop e setup e comando de atribuição.

Tabela 4 – Resumo da utilização do APC

Códigos	Qtd
Códigos compilados	131
Códigos enviados para o robô	348
Arquivos vazio	40
Códigos com erro	97
Falta de ponto e vírgula ;	36
Nome de comando errado	12
Erro com {	13
Outros erros	32

6.2 Avaliação do LaraPC

Avalição de sistemas CSCL é uma tarefa complicada e ainda não existe consenso sobre os métodos mais adequados para realizá-la. Entretanto alguns trabalhos propõem princípios e critérios que podem ser considerados nesta avaliação. Ewing & Miller (2002) sugerem cinco características a serem consideradas na avaliação de um ambiente de aprendizagem colaborativa: responsabilidade individual do aluno sobre o seu processo de aprendizagem; a aprendizagem ocorre em grupos pequenos; a comunicação durante a realização da tarefa é interativa e dinâmica; os alunos podem identificar seus papeis durante a realização da tarefa; os participantes devem ter um entendimento compartilhado dentro do ambiente onde estão inseridos. Gormley & Redfern (2007) definiram um conjunto de oito heurísticas baseadas na heurísticas de Nielsen (1995) para identificar as causas de problemas de usabilidade em ambientes web de colaboração e comunicação (WCC). Huang (2010) define um framework baseado na ISSO 9241-11 para avaliação de usabilidade de sistema CSCL. A estrutura consiste em 24 critérios agrupados em seis dimensões: eficácia, eficiência, colaboração, tolerância a erros, acessibilidade universal e satisfação.

As diretrizes escolhidas para guiar a avaliação do protótipo apresentado neste trabalho foram as definidas por Huang (2010) na dimensão colaboração. Esta dimensão visa avaliar os aspectos de usabilidade de um sistema que suporta ensino e aprendizagem colaborativa, foco principal da avaliação do LaraPC.

Os oito critérios definidos por Huang (2010) para colaboração são: Gestão do Usuário; Conscientização; Comunicação; Controle do Usuário / Controle do Moderador e do Professor; Compartilhamento / Gerenciamento de Arquivos / Conteúdo; Rastreamento de Processos / Notificação Automatizada; Proteção de Arquivo / Conteúdo; e Segurança. Um sistema CSCL deve permitir que os usuários gerenciem os membros da equipe no sistema pois uma equipe consiste em diferentes funções. Os recursos de conscientização devem garantir que os membros da equipe saibam quem está logado no sistema e o que cada membro está fazendo nas tarefas da equipe. O controle do moderador/professor deve permitir o monitoramento das atividades de uma equipe de tempos em tempos para garantir que o trabalho esteja sob controle e seja realizado corretamente. Os trabalhos da equipe precisam ser armazenados e protegidos no sistema e podem ser recuperado e acompanhado on-line sempre que necessário.

Para avaliação do LaraPC foi utilizado um método de inspeção por ser eficiente e de baixo custo (Santos et al., 2012). Os métodos de inspeção não exigem a participação de usuários finais reais, a inspeção é realizada por especialistas que verificam a conformidade dos artefatos de software com um conjunto de diretrizes ou princípios de usabilidade (Fernandez et al., 2011).

O objetivo do teste de inspeção da qualidade da interface do LaraPC foi identificar aspectos capazes de atrapalhar os usuários durante suas interações. A inspeção foi realizada por dois avaliadores, ambos verificaram a conformidade do sistema aos critérios da avaliação e apontaram as falhas encontradas. Para cada critério, foi definido a sua classificação através de uma escala de 0 (fraco) a 4 (fortíssimo) onde: 0 – Não representa um problema; 1 – Problema apenas estético: não precisa ser consertado a menos que haja tempo no projeto; 2 – Problema pequeno: o conserto deste problema é desejável, mas deve receber baixa prioridade; 3 – Problema grande: é importante consertá-lo, deve receber alta prioridade; 4 – Catastrófico: é imperativo conserta-lo antes do lançamento do produto.

As avaliações foram compatibilizadas e o resultado é mostrado na Tabela 5. O LaraPC possui boa usabilidade e não foram encontrados problemas críticos. A interface atende aos critérios de comunicação, controle do usuário / controle do professor, compartilhamento de arquivos e conteúdo, e segurança. Possui alguns problemas estéticos quanto ao gerenciamento de usuários e consciência. Os critérios de rastreamento de processos / notificação automatizada e proteção de arquivo / conteúdo foram classificados como problema pequeno.

Tabela 5 – Resultado da inspeção de usabilidade do LaraPC

Critérios de colaboração	**Comentário**	**Classificação**
1. Gerenciamento de usuários O sistema permite cadastrar usuários e suas permissões? O usuário pode gerenciar os membros da equipe?	O cadastro direto do usuário não é permitido. Porém, isso não caracteriza um problema já que a estrutura dos cursos e turmas se baseiam em convites direcionados. Durante a *party* o líder pode gerenciar os participantes por meio de convite, exclusão ou com a ação de tornar um dos participantes o líder da *party*. Mas, faltam recursos para especificar as atribuições e responsabilidades de cada membros do grupo em uma atividade.	1
2. Consciência (*awareness*) O sistema fornece mecanismos de conscientização?	Na aba Desenvolvimento não é possível identificar de maneira direta quem está contribuindo, para isso o	1

O sistema permite identificar as pessoas que participam de uma atividade colaborativa? O grupo consegue perceber a contribuição de cada membro na realização de atividades?	usuário precisa procurar a identificação na aba Análise. É possível a identificação dos membros de uma *party*, assim como a identificação das contribuições individuais realizadas tanto no módulo de Análise quanto no de Desenvolvimento.	
3. Comunicação O sistema fornece mecanismos de comunicação assíncrona? O sistema fornece mecanismos de comunicação síncrona? Os mecanismos de comunicação são suficientes?	Não há mecanismos de comunicação assíncrona no laboratório, todos os participantes de uma *party* podem se comunicar de maneira síncrona através do chat. Entretanto, o Moodle fornece comunicação assíncrona. Os mecanismos de comunicação são suficientes pois asseguram a comunicação dos participantes durante a execução de uma tarefa.	0
4. Controle do Usuário / Controle do professor e do Moderador O sistema permite ao professor dar instruções online e monitorar o trabalho em equipe? O sistema permite ao aluno gerenciar seus arquivos?	Partindo da premissa que o professor tenha acesso irrestrito ao laboratório, ele pode monitorar as atividades realizadas durante as *parties*, assim como dar instruções através do chat. O gerenciamento de arquivos no laboratório possibilita ao participante fazer upload e downloads dos seus arquivos nos dois módulos.	0
5. Compartilhamento de arquivos e conteúdo	Todos os membros da *party* acessam o mesmo conteúdo, tanto no módulo de Análise quanto no Desenvolvimento, de maneira que tanto o texto	0

O sistema fornecer meios e recursos para um grupo realizar tarefas e trabalhos em equipe? O sistema fornece interface 'WYSIWIS' (*What You See Is What I See* - O que você vê é o que eu vejo) durante a realização de atividade em equipe? O sistema permite o acesso fácil a uma base de conhecimento compartilhada?	quanto o código sendo editados estão disponíveis para download e upload por todos os participantes. Além disso, o módulo Análise fornece a possibilidade de compartilhamento do texto através de um link.	
6. Rastreamento de Processos / Notificação Automatizada O sistema permite o controle de versão através do qual os participantes podem recuperar versões anteriores de trabalhos? É possível enviar uma notificação para a equipe?	Há controle de versão no módulo Análise durante a sessão. Após o fim da sessão colaborativa, os dados não podem ser recuperados. A única maneira de notificar a equipe é por meio do chat.	2
7. Proteção de Arquivo / Conteúdo O sistema fornece o arquivamento de artefatos? O sistema permite que alunos realizem tarefas e trabalhos individualmente?	O sistema fornece a possibilidade de download dos arquivos nos dois módulos do laboratório. E o Moodle fornece espaço para armazenamento de arquivos para cada usuário. Seria interessante que essas ações pudessem acontecer de maneira conjunta e automática, mas ainda a critério do usuário. As atividades individuais não podem ser realizadas durante a sessão de uma *party*.	2
8. Segurança O usuário precisa fornecer nome e senha para acessar o sistema? O sistema parece seguro para armazenar os trabalhos e arquivos?	É necessária autenticação do usuário. Os usuário podem armazenar seus trabalhos e arquivos no espaço individual fornecido através do Moodle.	0

O sistema permite o gerenciamento de usuários através do Moodle e exige autenticação para o acesso. É possível realizar atividades em grupo e gerenciar os seus membros, mas não disponibiliza recursos para realizar o gerenciamento automático das responsabilidades dos membros do grupo. As atribuições e responsabilidades de cada participante podem ser registradas no documento de análise.

Cada membros do grupo é representado no ambiente e possui um cor, eles podem se comunicar de forma dinâmica através do chat e é possível identificar a contribuição de cada um. Entretanto, na aba Desenvolvimento é aconselhável inserir recurso para identificação direta da cor dos participantes. Esta identificação está presente apenas na aba Análise.

O sistema permite fazer download e upload dos artefatos da aba Análise e Desenvolvimento. Mas o sistema não fornece recursos para recuperar os dados de uma sessão de colaboração após seu encerramento, isto dificulta o controle de versão. Nos trabalhos individuais, é possível recuperar as versões anteriores do trabalho.

Apesar de possuir proteção de arquivo e conteúdo, o sistema não salva automaticamente os artefatos dos usuários em seu espaço de armazenamento no Moodle. A sugestão é fornecer ao usuário liberdade para escolher onde e como salvar seus trabalhos.

6.3 Considerações sobre as avaliações

Considerando as avaliações do APC e do LaraPC, é possível destacar que o sistema possui quatro características importantes: ubiquidade, usabilidade, aplicação no ensino contextualizado de computação e colaboração. O sistema pode ser acessado a qualquer hora de qualquer lugar através de um navegador web. O usuário tem liberdade para estudar a qualquer momento, pois o sistema fica disponível 24 horas por dia 7 dias por semana, sem a necessidade de instalar software ou esperar carregar a bateria do robô. Ele é integrado ao *Moodle*, o que permite inserir recursos e atividades, além de favorecer o uso de estratégias didáticas. A sua interface com o usuário é simples, fácil de usar e disponibiliza as principais funcionalidades de um editor de texto e de uma IDE. A biblioteca de comandos do robô também é bastante simples e intuitiva, o que facilita sua utilização e assimilação. O ambiente permite que os usuários se comuniquem e realizem atividades em grupo em tempo real. Além disto, a avaliação mostrou que o sistema é um ambiente com grande potencial para envolver os alunos na aprendizagem de conceitos abstratos como os de programação.

Considerando os objetivos apresentados para um ambiente CSCL para programação com suporte a contextualização através de laboratório remoto, a arquitetura proposta atende à maioria dos problemas apontados no ensino de programação e as características

propostas no capitulo 3. Entretanto melhorias e ampliações precisam ser feitas para atender plenamente alguns requisitos:

- Gerenciamento e organização de materiais didáticos através do AVA: O Moodle atende perfeitamente esta característica.
- Trabalho em grupo: a implementação de grupos dinâmicos garante a possibilidade de trabalhos em equipe, entretanto faltam recursos para atribuir e acompanhar as responsabilidades de cada membro na realização das atividades.
- Suporte a estratégia de programação: O módulo de análise permite que os alunos façam o planejamento das soluções de problemas, isto pode ajudar no entendimento do problema e da solução. Entretanto é necessário que a metodologia de ensino aprendizagem aplicada dê suporte a esta atividade.
- Suporte a implementação: Esta característica foi perfeitamente atendida através do compilador online presente no Ambiente de Programação e Controle.
- Suporte a colaboração: Além das ferramentas de comunicação presentes no Moodle, é possível interagir através do chat e desenvolver programação colaborativa, mas o professore precisa incentivar e mediar a colaboração. Também é preciso disponibilizar métodos para recuperar e retomar *party* já finalizadas, além de permitir a colaboração assíncrona.
- Suporte a robótica educativa remota: O laboratório remoto de robótica foi muito bem avaliado como ferramenta aplicável ao ensino de programação.
- Utilização de ferramentas de código aberto: característica plenamente atendida. Entretanto, a Cubieboard demostrou ter um processamento limitado provocando travamento do sistema em diversos momentos.
- Usabilidade: Atende a maioria dos aspectos analisados, mas necessita da opinião do usuário final.
- Modularidade: Apesar do sistema estar dividido em módulos, é necessário maior interoperabilidade.

Algumas características ainda precisam ser implementadas, apoio a comunicação via videoconferência e suporte as atividades do professore, como avaliação dos dados gerados no processo de colaboração.

7 Conclusões e trabalhos futuros

Este trabalho apresenta uma arquitetura modular com seus conceitos e complementos tecnológicos para auxiliar no ensino e aprendizagem de programação de computadores. A arquitetura foi fundamentada em características apontadas na literatura como importantes para o desenvolvimento de um ambiente colaborativo e motivador para a educação em programação. Foram definidos seus componentes, a comunicação entre eles e um método para tornar o laboratório remoto colaborativo e com suporte à programação. Foi também implementado um protótipo com GUI amigável para validar a utilização de laboratórios remotos no ensino de programação.

A arquitetura propõe a integração de ambiente virtual de aprendizagem, laboratório remoto, IDE e editor de texto online para compor um ambiente CSCL de programação. Para a caracterização do ambiente de ensino, foi utilizado AVA que permite gerenciamento de materiais didáticos e aplicação de estratégias e objetivos de aprendizagem. O laboratório remoto possibilita a realização de atividades práticas contextualizadas de forma ubíqua, superando as limitações de tempo e espaço dos laboratórios presenciais. A IDE online garante a programação colaborativa através da interação entre os usuários e do compartilhamento de código em tempo real. Para dar suporte as fases de análise e projeto no ensino de programação, foi incorporado um editor de texto colaborativo. Neste trabalho, portanto, foi apresentada uma alternativa para unir vantagens, baixar custos e proporcionar flexibilidade para criação de contextos educacionais de programação agregando componentes para resolução de problemas (analisar o problema a ser resolvido, formular soluções, produzir algoritmo), codificação de algoritmos, depuração e teste de soluções e colaboração.

A validação da arquitetura foi realizada através do desenvolvido do protótipo LaraPC. A escolha das ferramentas, tecnologias e conceitos para compor o protótipo foi baseada na capacidade de desempenhar funções e características idealizadas na arquitetura, serem livres e de código aberto. Foram utilizados no desenvolvimento do protótipo o Moodle, como módulo de estudo, o EtherPad Lite, como módulo de análise e editor de código compartilhado, o conceito de *party* de jogos online, para gerenciar a colaboração entre os usuários, e o Ambiente de Programação e Controle (APC) de um robô para os módulos de codificação e experimentação. O APC foi desenvolvido especialmente para compor esta proposta. O APC integra um laboratório remoto robótico com uma IDE on-line, que permite aos alunos analisar, implementar, testar e validar códigos para controlar um robô móvel de baixo custo, o L1R2, de maneira colaborativa ou individual.

As avaliações realizadas com o APC e com o LaraPC, embora tenham utilizado um pequeno grupo de estudantes e especialistas, apontam a eficiência destes ambientes como espaço de aprendizagem que pode motivar e atrair estudantes para o estudo de programação. A avaliação de satisfação do APC demostra que o ambiente é motivador e desperta o interesse dos alunos pela programação. A interface simples e fácil de usar ajuda neste processo, pois a interação com o software é intuitiva e permite que os alunos se concentrem na aprendizagem do conteúdo e não na utilização do sistema.

Agrupar laboratório remoto de robótica, colaboração multiusuário, ambiente virtual de aprendizagem, IDE online e editor de texto em um único ambiente é importante para evitar interrupções no processo de aprendizagem e ampliar as possibilidades do ensino de programação tradicional e a distância. Além de representando um avanço no estudo sobre aprendizagem de programação e laboratório remoto, uma vez que aumenta o interesse dos alunos pelo estudo da programação e torna o laboratório remoto mais semelhante ao ambiente presencial com espaço compartilhado, interação multiusuário, material didático, suporte as etapas de programação (análise, projeto, codificação e teste).

Essas características tornam o LaraPC uma ferramenta útil não só para o ensino de programação, mas também pode ser aplicado na aprendizagem de outros conteúdos importantes nos cursos de computação e engenharia, tais como: Algoritmos e estrutura de dados – é possível testar e escolher a estrutura de dados que melhor represente os dados do robô, o caminho seguido e o mapa do ambiente; Inteligência Artificial – é possível aprender técnicas de busca e problema de caminho mais curto; Análise de algoritmos – algoritmos de controle de robôs podem ser formalmente analisados e verificados sua eficiência na prática; Teoria dos grafos – é possível aplicar no mapeamento do ambiente do robô.

Em síntese, as principais contribuições desta tese, são:

- Uma ferramenta para aumentar o interesse do aluno no aprendizado de programação através do uso de um ambiente colaborativo para o ensino de programação utilizando laboratório remoto de robótica móvel;
- A estruturação de uma arquitetura modular para o ensino colaborativo de programação que integra ambiente virtual de aprendizagem, programação colaborativa, laboratório remoto, robótica móvel e IDE online;
- Desenvolvimento do laboratório remoto de robótica móvel de baixo custo integrado a ferramenta de programação com apoio a colaboração entre os usuários;
- Robô móvel de baixo custo com sensores que, além de seguir linha, permite executar experiências como localização de robô, mapeamento de ambiente, desvio de obstáculos e identificação de pontos de luz na arena;

- Validação do laboratório remoto de robótica com ferramenta para o ensino contextualização de programação.

Por se tratar de um tema relativamente novo, em questionamento e com outras alternativas de implementação, a pesquisa apresentada possibilita de realização de vários trabalhos futuros:

- Testar todo o LaraPC com usuários finais reais. O planejamento inicial para avaliação do sistema era no primeiro momento avaliar o APC como uma ferramenta para o ensino de programação e com base nos resultados obtidos fazer os ajustes necessários e realizar a avaliação do sistema como um todo. Entretanto, uma seria de incidentes não permitiram que este planejamento fosse cumprido. A primeira tentativa de avaliação do APC programada para o primeiro semestre de 2015 não pôde ser concluída devido à greve dos professores da UESB que durou três meses. A oficina só pode ser realizada completamente com os calouros de 2016 que iniciaram a graduação em meados de maio de 2016. Esta avaliação também testou a integração do APC com o Moodle. A segunda etapa da avaliação, onde a colaboração e a integração entre os módulos seriam testadas, ficou comprometida, pois os alunos voltaram as atividades do segundo semestre de 2016 em outubro e logo em seguida (dia 21/10) a UESB teve suas atividades suspensas devido a ocupação dos campi pelos estudantes em protesto contra a aprovação da PEC 241/55 do governo federal. A universidade só voltou a funcionar normalmente em fevereiro de 2017 inviabilizando a realização da avaliação com usuários reais.
- Continuar o desenvolvimento do LaraPC, tendo em vista a ampliação da gestão das *party* para recuperar trabalhos em grupo, suporte a videoconferência e desenvolvimento da parte administrativa e de professores prevista na arquitetura, mas não realizada;
- Desenvolvimento de tutores online utilizando tecnologia de sistemas multiagentes;
- Desenvolver uma versão do sistema para dispositivos móveis;
- Desenvolver um curso completo de programação de computadores no LaraPC e ministrá-lo;
- Integrar outros laboratórios remotos ao projeto;
- Testar outros hardwares abertos de baixo custo como servidor de laboratório;
- Desenvolver metodologia pedagógica para utilização do LaraPC.

8 Referencias

Ahmed, B., & Alsaleh, K. (2011). Robotics: Its effectiveness as a tool to teach engineering design and computer programming. *2011 IEEE Global Engineering Education Conference (EDUCON)*, 1018–1021.

Alkhaldi, T., Pranata, I., & Athauda, R. I. (2016). A review of contemporary virtual and remote laboratory implementations: observations and finding. *Journal of Computers in Education*, *3*(3), 329–351.

Almeida, L. D. A., & Baranauskas, M. C. C. (2008). Um Prospecto de Sistemas Colaborativos: Modelos e Frameworks. In Proceedings of the VIII Brazilian Symposium on Human Factors in Computing Systems (pp. 204–213).

Anderson, M., McKenzie, A., & Wellman, B. (2011). Affecting attitudes in first-year computer science using syntaxfree robotics programming. *ACM Inroads*, *2*(3), 51–57.

Andujar, M., Jimenez, L., Shah, J., & Morreale, P. (2013). EVALUATING VISUAL PROGRAMMING ENVIRONMENTS TO TEACH COMPUTING. *Journal of Computing Sciences in Colleges*, *29*(2), 140–148

Aroca, R. V., Gardiman, R. Q., & Goncalves, L. M. G. (2012). Web-Based Robot Programming Environment and Control Architecture. *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*, 27–32.

Barrios, A., Panche, S., Duque, M., Grisales, V. H., Prieto, F., Villa, J. L., Chevrel, P., et al. (2013). A multi-user remote academic laboratory system. Computers & Education, 62, 111–122.

Bani-Salameh, H., Jeffery, C., Al-Sharif, Z., & Doush, I. A. (2008). Integrating collaborative program development and debugging within a virtual environment. *Groupware: Design, Implementation, and Use. Springer Berlin Heidelberg*, 107–120.

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

Bini, E. M., & Koscianski, A. (2009). O ensino de programação de computadores em um ambiente criativo e motivador. VII Enpec.

Bochicchio, M. A., & Longo, A. (2010). Extending LMS with Collaborative Remote Lab Features. *2010 10th IEEE International Conference on Advanced Learning Technologies*, 310–314.

Brandão, L. D. O., Ribeiro, S., Brandão, A. A. F., & Paulo, S. (2012). A system to help teaching

and learning algorithms. *Frontiers in Education Conference (FIE), 2012. IEEE.*

Bravo, C., Duque, R., & Gallardo, J. (2013). A groupware system to support collaborative programming: Design and experiences. *Journal of Systems and Software*, *86*(7), 1759–1771.

Brinson, J. R. (2015). Learning outcome achievement in non-traditional (virtual and remote) versus traditional (hands-on) laboratories: A review of the empirical research. *Computers & Education*, *87*, 218–237.

Brito, S. R., Silva, A. S., Tavares, O. L., Favero, E. L., & Francês, C. R. L. (2011). Computer Supported Collaborative Learning for helping novice students acquire self-regulated problem-solving skills in computer programming. *The 2011 International Conference on Frontiers in Education: Computer Science and Computer Engineering*, 65–73.

Callaghan, M. J., Harkin, J., McColgan, E., McGinnity, T. M., & Maguire, L. P. (2007). Client–server architecture for collaborative remote experimentation. *Journal of Network and Computer Applications*, *30*(4), 1295–1308.

Casini, M., Garulli, A., Giannitrapani, A., & Vicino, A. (2013). Remote pursuer-evader experiments with mobile robots in the Automatic Control Telelab. Advances in Control Education, 66–71.

Casini, M., Garulli, A., Giannitrapani, A., & Vicino, A. (2014). A Remote Lab for Experiments with a Team of Mobile Robots. *Sensors*, 14(9), 16486–16507.

Celestino, H. A. D. S. (2013). APRENDIZAGEM DE ESTRUTURAS DE CONTROLO COM RECURSO À ROBÓTICA EDUCATIVA. Mestrado em Ensino de Informática - UNIVERSIDADE DE LISBOA.

Chaos, D., Chacón, J., Lopez-Orozco, J. A., & Dormido, S. (2013). Virtual and Remote Robotic Laboratory Using EJS, MATLAB and LabVIEW. *Sensors 2013*, (1424-8220), 2595–2612.

Chaves, J. O. M., Castro, A. F., Lima, R. W., Vinicius, M., Lima, A., & Ferreira, K. H. A. (2013). Integrando Moodle e Juízes Online no Apoio a Atividades de Programação. *II Congresso Brasileiro de Informática na Educação (CBIE 2013) XXIV Simpósio Brasileiro de Informática na Educação (SBIE 2013)*, (Cbie), 244–254.

Chen, X., Song, G., & Zhang, Y. (2010). Virtual and Remote Laboratory Development: A Review. Earth and Space 2010: Engineering, Science, Construction, and Operations in Challenging Environments, 368–368.

Chen, Y., & Zhou, Z. (2015). Robot as a Service in Computing Curriculum. *2015 IEEE Twelfth International Symposium on Autonomous Decentralized Systems*, 156–161.

Cooper, M. (2005). Remote laboratories in teaching and learning – issues impinging on widespread adoption in science and engineering education. *International Journal of Online Engineering (iJOE)*, *1*(1).

Cooper, S., & Cunningham, S. (2010). Teaching Computer Science in Context. *ACM Inroads*, *1*(1), 5–8.

Corter, J. E., Esche, S. K., Chassapis, C., Ma, J., & Nickerson, J. V. (2011). Process and learning outcomes from remotely-operated, simulated, and hands-on student laboratories. Computers & Education, 57(3), 2054–2067.

Costabile, M. F., De Marsico, M., Lanzilotti, R., Plantamura, V. L., & Roselli, T. (2005). On the Usability Evaluation of E-Learning Applications. In System Sciences, 2005. HICSS '05. Proceedings of the 38th Annual Hawaii International Conference on. IEEE (pp. 1–10).

Cruz-Martín, A., Fernández-Madrigal, J. a., Galindo, C., González-Jiménez, J., Stockmans-Daou, C., & Blanco-Claraco, J. L. (2012). A LEGO Mindstorms NXT approach for teaching at Data Acquisition, Control Systems Engineering and Real-Time Systems undergraduate courses. *Computers & Education*, *59*(3), 974–988.

CS (2013). Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. *ACM New York, NY, USA*, (978-1-4503-2309-3). Disponível em: http://www.acm.org/education/CS2013-final-report.pdf . Acessado em: 10/03/2014

Devide, J. V. S., Meneely, A., W., H. C., L., W., & Devetsikiotis, M. (2008). Jazz Sangam: A real-time tool for distributed pair programming on a team development platform. *Workshop on Infrastructure for Research in Collaborative Software Engineering, Atlanta, GA*.

Ewing, J., & Miller, D. (2002). A Framework for Evaluating Computer Supported Collaborative Learning. *Educational Technology & Society*, *5*(1), 112–118.

Fan, H. (2012). ATCoPE : Any-Time Collaborative Programming Environment for Seamless Integration of Real-Time and Non-Real-Time Teamwork in Software Development. *Proceedings of the 17th ACM International Conference on Supporting Group Work. ACM*, 107–116.

Fan, H., & Sun, C. (2012)a. Supporting semantic conflict prevention in real-time collaborative programming environments. *ACM SIGAPP Applied Computing Review*, *12*(2), 39–52.

Fan, H., & Sun, C. (2012)b. Achieving integrated consistency maintenance and awareness in real-time collaborative programming environments: The CoEclipse approach. *Proceedings*

of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 94–101.

Fayolle, J., Gravier, C., Yankelovich, N., & Kim, E. (2011). Remote Lab in Virtual World For Remote Control of Industrial Processes. *EEE International Conference on Multimedia and Expo (ICME), 2011 I*, 1–4.

Fernandez, A., Insfran, E., & Abrahão, S. (2011). Usability evaluation methods for the web: A systematic mapping study. *Information and Software Technology*, 53(8), 789–817.

Ferreira, S. M. C. (2013). A robótica educativa no ensino-aprendizagem de estruturas de seleção. Mestrado em Ensino de Informática - UNIVERSIDADE DE LISBOA

Filippo, D., Pimentel, M., & Wainer, J. (2012). Metodologia de pesquisa científica em sistemas colaborativos. Sistemas Colaborativos, 379–404

França, A. B., & Soares, J. M. (2011). Sistema de apoio a atividades de laboratório de programação via Moodle com suporte ao balanceamento de carga. XXII SBIE - XVII WIE, 710–719.

Fuks, H., Pimentel, M. (2011). Sistemas colaborativos. Elsevier Brasil.

Gadelha, B. F. (2011). *Uma Abordagem de Desenvolvimento de Groupware Baseada em Linha de Produto de Software e Modelo 3C de Colaboração*. *Tese de doutorado*. PUC-Rio.

García-Zubia, J., Irurzun, J., Angulo, I., Hernández, U., Castro, M., Sancristobal, E., Ruiz-de-Garibay, J. (2010). SecondLab : A Remote Laboratory under Second Life. *Education Engineering (EDUCON)*, 351–356.

García-Zubia, J. & Angulo, I. (2011) Introduction to Microcontrollers E-PRAGMATIC module. UNIVERSITY OF DEUSTO. September 26, 2011.

Gerosa, M. A., Raposo, A. B., Fuks, H., José, C., & Lucena, P. De. (2004). Uma Arquitetura para o Desenvolvimento de Ferramentas Colaborativas para o Ambiente de Aprendizagem AulaNet. Workshop Informática Na Educação Informática Na Educação, 168–177.

Gil, A. C. (2002). Como Elaborar Projetos de Pesquisa. EDITORA ATLAS (4th ed.). São Paulo.

Goadrich, M. (2014). Incorporating tangible computing devices into cs1 *. *Journal of Computing Sciences in Colleges*, *29*(5), 23–31.

Gomes, A., Henriques, J., & Mendes, A. J. (2008). Uma proposta para ajudar alunos com dificuldades na aprendizagem inicial de programação de computadores. *Educaçao, Formaçao & Tecnologias*, *1*, 93–103.

Gomes, L., & Bogosyan, S. (2009). Current Trends in Remote Laboratories. *IEEE Transactions on Industrial Electronics*, 56(12), 4744–4756.

Gonzalez-Barbone, V., & Anido-Rifon, L. (2008). Creating the first SCORM object. *Computers & Education*, 51(4), 1634–1647.

Gormley, P., & Redfern, S. (2007). The "WCC Heuristic Evaluation Set" – Towards a Discount Methodology for Evaluating the Usability of Web Communication and Collaboration (WCC) Groupware in Teaching and Learning. *EdTech*.

Gravier, C., Fayolle, J., Lardon, J., & O'Connor, M. J. (2012). Adaptive System for Collaborative Online Laboratories. *IEEE Intelligent Systems*, *27*, 11–17.

Guimarães, E. G., Cardozo, E., Moraes, D. H., & Coelho, P. R. (2011). Design and Implementation Issues for Modern Remote Laboratories. *IEEE Transactions on Learning Technologies*, 4(2), 149–161.

Guzdial, M. (2010). Does contextualized computing education help? *ACM Inroads*, *1*(4), 4

Heradio, R., de la Torre, L., Galan, D., Cabrerizo, F. J., Herrera-Viedma, E., & Dormido, S. (2016). Virtual and Remote Labs in Education: a Bibliometric Analysis. Computers & Education, 98, 14–38.

Huang, E. (2010). Identifying an effective framework for usability evaluation of Computer Supported Collaborative Learning System in Educational settings (Doctoral dissertation, Auckland University of Technology - Nova Zelândia).

Husain, M., Tarannum, N., & Patil, N. (2013). Teaching programming course elective: A new teaching and learning experience. 2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE), 275–279.

Hwang, W.-Y., Shadiev, R., Wang, C.-Y., & Huang, Z.-H. (2012). A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. *Computers & Education*, *58*(4), 1267–1281.

iRobot (2012). IROBOT RIG - LABORATORY USER GUIDE. Version 2.1. Disponível em: http://www.labshare.edu.au/media/rig_guides/irobot_userguide_v2-2.pdf Acessado em: 03/02/2014.

ISO (1998). ISO 9241-11, 98: Ergonomic Requirements for Office work with Visual Display Terminals. Part 11: Guidance on Usability. ISO

Iturrate, I., Martín, G., García-zubia, J., Angulo, I., Dziabenko, O., & Orduña, P. (2013). A Mobile Robot Platform for Open Learning based on Serious Games and Remote

Laboratories. In International Conference of the Portuguese Society for Engineering Education (CISPEE) (pp. 1–7). Porto.

Jara, C. a., Candelas, F. a., Torres, F., Dormido, S., & Esquembre, F. (2009). Synchronous collaboration of virtual and remote laboratories. *Computer Applications in Engineering Education*, *20*(1), 124–136.

Jenkins, T. (2002). On the Difficulty of Learning to Program. In Proceedings of 3rd Annual LTSN_ICS Conference. The Higher Education Academy (pp. 53–58).

Joolingen, W. R. van, Jong, T. de, Lazonder, A. W., Savelsbergh, E. R., & Manlove, S. (2005). Co-Lab: research and development of an online learning environment for collaborative scientific discovery learning. *Computers in Human Behavior*, *21*(4), 671–688.

Jurado, F., Redondo, M. a., & Ortega, M. (2012). Blackboard architecture to integrate components and agents in heterogeneous distributed eLearning systems: An application for learning to program. *Journal of Systems and Software*, *85*(7), 1621–1636.

Jurado, F., Molina, A. I., Redondo, M. A., & Ortega, M. (2013). Cole-Programming : Shaping Collaborative Learning Support in Eclipse. *IEEE-RITA*, *8*(4), 153–162.

Juristo, N., Moreno, A. M., & Sanchez-Segura, M.-I. (2007). Analysing the impact of usability on software design. Journal of Systems and Software, 80(9), 1506–1516.

Kauark, F. da S., Manhães, F. C., & Medeiros, C. H. (2010). Metodologia da Pesquisa: Um guia prático. Itabuna-BA: Via Litterarum.

Kay, J. S. (2011). Contextualized Approaches to Introductory Computer Science : The Key to Making Computer Science Relevant or Simply Bait and Switch ? In *SIGCSE '11 Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 177–182).

Kazimoglu, C., Kiernan, M., Bacon, L., & Mackinnon, L. (2012). A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. *Procedia - Social and Behavioral Sciences*, *47*, 1991–1999.

Koorsse, M., Cilliers, C., & Calitz, A. (2015). Programming assistance tools to support the learning of IT programming in South African secondary schools. *Computers & Education*, *82*, 162–178.

Kosnar, K., Faigl, J., Saska, M., Kulich, M., Chudoba, J., Stepán, P., & Preucil, L. (2011). SyRoTek: V012.2 - User's manual for SyRoTek e-learning system.

Kulich, M., Chudoba, J., Košnar, K., Krajník, T., Faigl, J., & Libor, P. (2013). SyRoTek — Distance Teaching of Mobile Robotics. *IEEE Transactions on Education*, 56(1), 18–23.

Kumar, V. S., Gress, C. L. Z., Hadwin, A. F., & Winne, P. H. (2010). Assessing process in CSCL: An ontological approach. *Computers in Human Behavior*, *26*(5), 825–834.

Lang, J. (2012). Comparative Study of Hands-on and Remote Physics Labs for First Year University Level Physics Students. Transformative Dialogues: Teaching & Learning Journal, 6(1), 1–25.

Law, K. M. Y., Lee, V. C. S., & Yu, Y. T. (2010). Learning motivation in e-learning facilitated computer programming courses. *Computers & Education*, *55*(1), 218–228.

Lima, J. F. (2013). Arquitetura em rede de compartilhamento de laboratórios on-line. Tese de doutorado em Engenharia Elétrica. Universidade de Brasília.

Ma, J., & Nickerson, J. V. (2006). Hands-On, Simulated, and Remote Laboratories: A Comparative Literature Review. *ACM Computing Surveys*, 38(3), 7–es.

Machotka, J., Nedic, Z., Nafalski, A., & Gol, O. (2009). A remote laboratory for collaborative experiments. *American Society for Engineering Education*.

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *ACM Transactions on Computing Education (TOCE)*, *10*(4), 1–15.

Mateos, V., Gallardo, A., Richter, T., Bellido, L., Debicki, P., & Villagrá, V. (2011). LiLa Booking System : Architecture and Conceptual Model of a Rig Booking System for On-Line Laboratories. *International Journal of Online Engineering (iJOE)*, *7*, 26–35.

McGill, M. M. (2012). Learning to Program with Personal Robots: Influences on Student Motivation. *ACM Transactions on Computing Education*, *12*(1), 1–32.

Moreira, M. P., & Favero, E. L. (2009). Um Ambiente para Ensino de Programação com Feedback Automático de Exercícios. Workshop Sobre Educação em Computação, 17, 429–438.

Muratet, M., Torguet, P., Jessel, J.-P., & Viallet, F. (2009). Towards a Serious Game to Help Students Learn Computer Programming. *International Journal of Computer Games Technology*, *2009*, 1–12.

Nandigam, D., & Bathula, H. (2013). Competing Dichotomies in Teaching Computer Programming to Beginner-Students. American Journal of Educational Research, 1(8), 307–312.

Nielsen, J. (1995). 10 usability heuristics for user interface design. Nielsen Norman Group. Available: https://www.nngroup.com/articles/ten-usability-heuristics/

Nielsen, J. (2000). Why you only need to test with 5 users. Nielsen Norman Group. Available: http://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/.

Nielsen, J. (2012) Usability 101: Introduction to Usability. January 4, 2012. Nielsen Norman Group. Available: http://www.nngroup.com/articles/usability-101-introduction-to-usability/

OBR. (2014). Regras Finais Nacional RoboCupJunior Rescue A OBR 2014 – Modalidade Prática. Disponível em <http://www.obr.org.br/wp-content/uploads/2013/04/rescueA_FINAL_2014_PORTUGUES.pdf>. Acessado em: 22/09/2014.

Oliver, J., Toledo, R., & Valderrama, E. (2010). A learning approach based on robotics in Computer Science and Computer Engineering. IEEE EDUCON 2010 Conference, 1343–1347.

Orduña, P., Irurzun, J., Hernández, U., Sancristobal, E., Martín, S., & Castro, M. (2009). Towards an Extensible WebLab Architecture. 3rd IEEE International Conference on E-Learning in Industrial Electronics, 2009. ICELIE '09., 115–120.

Orduña, P., Sancristobal, E., Emaldi, M., Castro, M., López-de-Ipina, D., & Garcia-Zubia, J. (2012). Modelling Remote Laboratories integrations in e-Learning tools through Remote Laboratories federation protocols. *Frontiers in Education Conference (FIE), 2012. IEEE*, 1–6.

Orduña, P., Uribe, S. B., Isaza, N. H., Sancristobal, E., Emaldi, M., Martin, A. P., Garcia-Zubia, J. (2013). Generic integration of remote laboratories in learning and content management systems through federation protocols. *Frontiers in Education Conference, 2013 IEEE*.

Othman, M., & Zain, N. M. (2015). Online collaboration for programming: Assessing students' cognitive abilities. Turkish Online Journal of Distance Education, 16(4), 84–97.

Özbek, M. E., & Kara, A. (2010). Software Technologies, Architectures and Interoperability in Remote Laboratories. 9th International Conference on Information Technology Based Higher Education and Training (ITHET), 2010, 402–406.

Park, J. S., & Lenskiy, A. (2014). Mobile Robot Platform for Improving Experience of Learning Programming Languages. *Journal of Automation and Control Engineering*, *2*(3), 265–269.

Pears, A., Seidman, S., Malmi, L., & Mannila, L. (2007). A survey of literature on the teaching of introductory programming. *ACM SIGCSE Bulletin*, *39*(4), 204–223.

Pereira, C. E., Paladini, S., & Schaf, F. M. (2012). Control and automation engineering education: Combining physical, remote and virtual labs. *International Multi-Conference on Systems, Sygnals & Devices*, 1–10.

Petrovic, P., & Balogh, R. (2012). Deployment of Remotely-Accessible Robotics Laboratory. *International Journal of Online Engineering*, *8*(2), 31–35.

Piteira, M., & Haddad, S. (2011). Innovate in your program computer class: an approach based on a serious game. *Proceedings of the 2011 Workshop on Open and Design of Communication*.

Prates, R. O., & Barbosa, S. D. J. (2003, July). Avaliação de Interfaces de Usuário–Conceitos e Métodos. *In Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação,* Capítulo (Vol. 6).

Ribeiro, P. C., Martins, C. B., & Bernardini, F. C. (2011). A Robótica como Ferramenta de Apoio ao Ensino de Disciplina de Programação em Curso de Computação e Engenharia. *Anais do XXII SBIE - XVII WIE*, 1108–1117.

Ribeiro, S., Brandão, L. D. O., & Brandão, A. A. F. (2012). Uma visão do cenário Nacional do Ensino de Algoritmos e Programação : uma proposta baseada no Paradigma de Programação Visual. Anais do Simpósio Brasileiro de Informática na Educação, 23(2004), 26–30

Ressurreição, R. P. L. da. (2012). A Consolidação de Conceitos de Programação Utilizando a Robótica Educativa. Mestrado em Ensino de Informática - UNIVERSIDADE DE LISBOA.

Rocha, R. (2006). Utilização da Robótica Pedagógica no Processo de Ensino-Aprendizagem de programação de Computadores. Dissertação (Mestrado em Educação Tecnológica) – Centro Federal de Educação Tecn. De Minas Gerais, Belo Horizonte

Salinger, S., Oezbek, C., Beecher, K., & Schenk, J. (2010). Saros: an eclipse plug-in for distributed party programming. *Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering*, 48–55.

Sancristobal, E., Castro, M., Harward, J., Baley, P., DeLong, K., & Hardison, J. (2010). Integration View of Web Labs and Learning Management Systems. *IEEE EDUCON Education Engineering* 2010, 81, 1409–1417.

Santoro, F. M., Borges, M. R. da S., & Santos, N. (1999). Um framework para estudo de ambientes de suporte à aprendizagem cooperativa. *Revista Brasileira de Informática na Educação*, (2).

Santos, N. S., Ferreira, L. S., & Prates, R. O. (2012). Um panorama sobre métodos de avaliação de sistemas colaborativos. *Proceedings - 9th Brazilian Symposium on Collaborative Systems, SBSC 2012*, (Dcc), 127–135.

Santos, R. P. dos, & 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. *Infocoomp, Journal of Computer Sciense*, *5*(1).

SBC (2005). "Currículo de Referência para Cursos de Graduação em Bacharelado em Ciência da Computação e Engenharia de Computação." *http://www.sbc.org.br/documentos-da-sbc/send/131-curriculos-de-referencia/760-curriculo-de-referencia-cc-ec-versao2005",* Acessado em 12-03-2014.

Serrano-Cámara, L. M., Paredes-Velasco, M., Alcover, C.-M., & Velazquez-Iturbide, J. Á. (2014). An evaluation of students' motivation in computer-supported collaborative learning of programming concepts. *Computers in Human Behavior*, *31*, 499–508.

Shen, H., & Sun, C. (2002). RECIPE : a prototype for Internet-based real-time collaborative programming. *Proc. of Intl. Conf. on Networks, Parallel and Distributed Processing*, 283–288.

Silva, T. R. Da, Medeiros, T., Medeiros, H., Lopes, R., & Aranha, E. (2015). Ensino-aprendizagem de programação: uma revisão sistemática da literatura. *Revista Brasileira de Informática Na Educação*, *23*(01), 182.

Slavin, R. E. (1995). Research on Cooperative Learning and Achievement : What We Know, What We Need to Know. *Center for Research on the Education of Students Placed at Risk, Johns Hopkins University*.

Soller, A. (2002). *Computational analysis of knowledge sharing in collaborative distance learning*. Tese de doutorado. University of Pittsburgh.

Sommerville, Ian (2003). Engenharia de software - São Paulo: Addison Wesley, 2003.

Souza, D. M. De, Batista, M. H. S., & Barbosa, E. F. (2016). Problemas e dificuldades no ensino de programação: Um mapeamento sistemático. *Revista Brasileira de Informática Na Educação*, *24*, 39–52.

Stahl, G., Koschmann, T., & Suthers, D. (2006). Computer-supported collaborative learning: An historical perspective. *Cambridge handbook of the learning sciences*, 409–426.

Tan, J., Guo, X., Zheng, W., & Zhong, M. (2014). Case-based teaching using the Laboratory Animal System for learning C/C++ programming. *Computers & Education*, *77*, 39–49.

Tawfik, M., Sancristobal, E., Martín, S., Díaz, G., Peire, J., & Castro, M. (2013). Expanding the Boundaries of the Classroom - Implementation of Remote Laboratories for Industrial Electronics Disciplines. *IEEE Industrial Electronics Magazine*, (march), 41–49.

Thomaz, S., Fernandes, C., Pitta, R., Torres, V., & Gonçalves, L. M. (2013). Web-based

configurable and multiplatform development environment for educational robotics. *2013 16th International Conference on Advanced Robotics, ICAR 2013*.

Torre, L. de la, Heradio, R., Jara, C. a., Sanchez, J., Dormido, S., Torres, F., & Candelas, F. a. (2013). Providing Collaborative Support to Virtual and Remote Laboratories. *IEEE Transactions on Learning Technologies*, 1–1.

Trindade, R. M. P., Bedregal, B. R. C., Neto, A. D. D., & Acioly, B. M. (2010). An interval metric. In New Advanced Technologies. InTech, 53, 323-340.

Tsang, E., Gavan, C., & Anderson, M. (2014). The practical application of LEGO® MINDSTORMS® robotics kits: does it enhance undergraduate computing students' engagement in learning the Java programming language? *In Proceedings of the 15th Annual Conference on Information technology education* (pp. 121–126).

Turner, C. W., Lewis, J. R., & Nielsen, J. (2006). Determining Usability Test Sample Size. *International Encyclopedia of Ergonomics and Human Factors*, 3(2), 3084–3088.

Verdú, E., Regueras, L. M., Verdú, M. J., Leal, J. P., de Castro, J. P., & Queirós, R. (2012). A distributed system for learning programming on-line. Computers & Education, 58(1), 1–10.

Vizcaíno, A., Contreras, J., Favela, J., & Prieto, M. (2000). An adaptive, collaborative environment to develop good habits in programming. *Intelligent Tutoring Systems. Springer Berlin Heidelberg*, 262–271.

Vicente, A. G., Muñoz, I. B., Luis, J., Galilea, L., & Revenga, P. A. (2010). Remote Automation Laboratory Using a Cluster of Virtual Machines. *IEEE Transactions on Industrial Electronics*, *57*(10), 3276–3283

Yamanishi, T., Sugihara, K., Ohkuma, K., & Uosaki, K. (2013). Programming instruction using a micro robot as a teaching tool. Computer Applications in Engineering Education, 23, 109–116.

Wang, S.-L., & Lin, S. S. J. (2007). The effects of group composition of self-efficacy and collective efficacy on computer-supported collaborative learning. *Computers in Human Behavior*, *23*(5), 2256–2268.

Wang, T., Su, X., Ma, P., Wang, Y., & Wang, K. (2011). Ability-training-oriented automated assessment in introductory programming course. Computers & Education, 56(1), 220–226.

Wu, H.-T., Hsu, P.-C., Lee, C.-Y., Wang, H.-J., & Sun, C.-K. (2014). The impact of supplementary hands-on practice on learning in introductory computer science course for freshmen. Computers & Education, 70, 1–8.

Zanetti, H. A. P., & Bonacin, R. (2014). Uso de semiótica e análise de normas em práticas de ensino de programação de computadores utilizando robótica pedagógica. *Revista Eletrônica de Tecnologia E Cultura*, 25–36.

Apêndice A – Produção científica decorrente da tese

Lopes, M. S. dos S., Brito, J. O., Trindade, R. M. P., Silva, A. F. Da, & Lima, A. C. D. C. (2016). Usability Evaluation of a Control and Programming Environment for Programming Education. International Journal of Software Engineering & Applications, 7(4), 11–21.

Lopes, M., Gomes, I., Trindade, R., Silva, A., & Lima, A. C. (2016). Web environment for programming and control of mobile robot in a remote laboratory. IEEE Transactions on Learning Technologies, 1–1. http://doi.org/10.1109/TLT.2016.2627565

Apêndice B – Montagem do L1R2

A Figura 32 mostra a arquitetura de componentes do L1R2 que ilustra a forma com que os sensores e atuadores do robô comunica-se com o microcontrolador Arduino Mega 2560. A Tabela 6 apresenta os componentes do robô e suas aplicações.

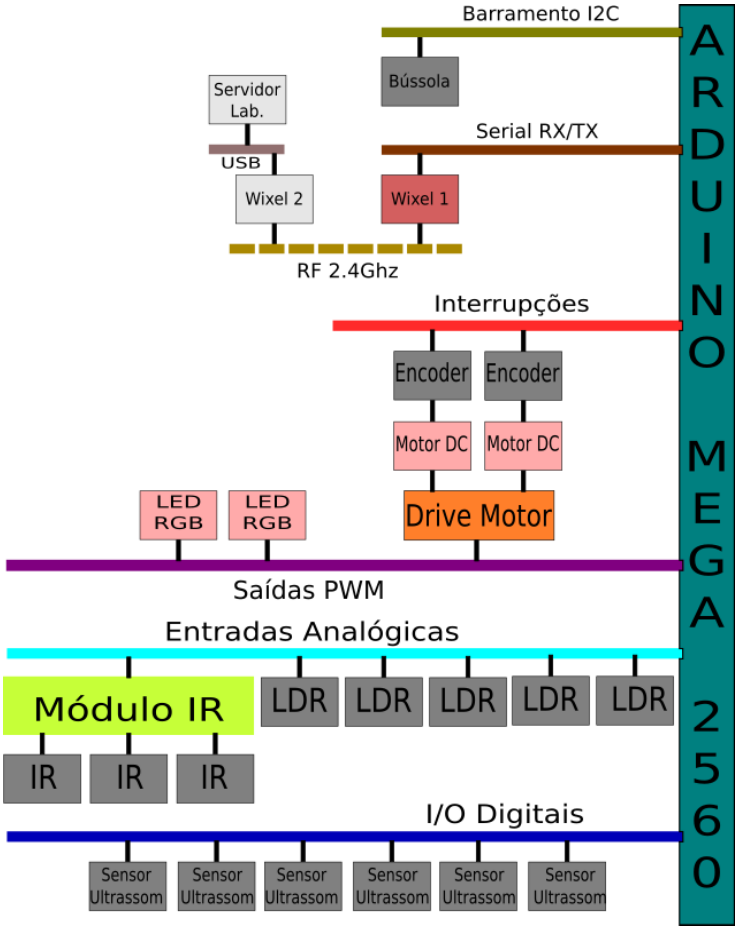


Figura 32 – Arquitetura de comunicação de componentes do L1R2

Autor: Iago P. Gomes

Tabela 6 – Descrição dos componentes do L1R2

Qtd.	Componente	Aplicação
1	Chassi magician	Base para o robô
6	Ultrassom HC-SR04	Usados para detecção e desvio de obstáculos na arena
3	Infravermelho tcrt5000 (emissor e receptor)	Usados para identificar a linha preta na arena

3	LDR (light dependent resistor) - 5mm	Usados para identificar a intensidade luminosa na arena
2	Motor CC CB605	Motores do robô
2	Encoders quadráticos	Usados para calcular o deslocamento do robô
1	Magnetômetro gy-271	Bússola, usado para calcular a orientação do robô
1	Drive de motor l298n	Usado para controlar os motores cc
1	Shield wixel	Usado para comunicação por rádio frequência com o servidor de laboratório
2	Led RGB 5mm	Usados como faróis do carrinho
1	Módulo seguidor de linha (CI comparador LM339N)	Usados para dar suporte a funcionalidade de seguir linha
2	Regulador de tensão 12v ~ 5v	Usados para converter a tensão de alimentação do robô de 12v para 5v, necessária para alimentar os sensores
1	Arduino mega 2560	Controlador do robô

Autor: Iago P. Gomes

UFBA
UNIVERSIDADE FEDERAL DA BAHIA
ESCOLA POLITÉCNICA

PROGRAMA DE PÓS GRADUAÇÃO EM ENGENHARIA INDUSTRIAL - PEI

Rua Aristides Novis, 02, 6º andar, Federação, Salvador BA
CEP: 40.210-630
Telefone: (71) 3283-9800
E-mail: pei@ufba.br
Home page: http://www.pei.ufba.br