UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

CIBELE ALVES DA SILVA REIS
HENRIQUE REINALDO SARMENTO
VINICIUS ZARAMELLA

FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL: UMA PLATAFORMA ROBÓTICA CONTROLADA POR SMARTPHONE

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA

2014

CIBELE ALVES DA SILVA REIS
HENRIQUE REINALDO SARMENTO
VINICIUS ZARAMELLA

FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL: UMA PLATAFORMA ROBÓTICA CONTROLADA POR SMARTPHONE

Projeto da disciplina de Trabalho de Conclusão de Curso de Engenharia Elétrica apresentado ao Departamento Acadêmico de Informática da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de "Engenheiro em Computação".

Orientador: Prof. Dr. Leonelo Dell Anhol Almeida

Co-orientador: Prof. Dr. Cesar Augusto Tacla

CURITIBA

2014

RESUMO

REIS, Cibele; SARMENTO, Henrique; ZARAMELLA, Vinicius. FERRAMENTA DE AUXÍLIO AO DESENVOLVIMENTO DO PENSAMENTO COMPUTACIONAL: UMA PLATAFORMA ROBÓTICA CONTROLADA POR SMARTPHONE . 118 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

A dificuldade dos alunos em aprender lógica e linguagens de programação faz com que várias soluções tecnológicas sejam criadas para auxiliar neste processo de ensino-aprendizagem. Entre as soluções tecnológicas, duas abordagens utilizadas são a robótica e a programação a partir de linguagens gráficas. A literatura indica que estas ferramentas podem auxiliar jovens a raciocinar de forma sistemática e desenvolver o pensamento computacional, e que este resultado pode ser potencializado com aplicação da metodologia de aprendizado baseado em problemas. Sendo assim, este trabalho descreve o desenvolvimento da Plataforma Coffee, constituída por um ambiente *Web* de programação em blocos, e um *smartphone* como unidade controladora de uma base robótica. Esta plataforma pode ser facilmente expandida para funcionar com diversos dispositivos móveis e kits robóticos. Outra realização deste trabalho foi a aplicação da plataforma em sala de aula com o objetivo de avaliar a ferramenta produzida e a motivação dos alunos ao utilizá-la. Além da avaliação da ferramenta pelos alunos foi realizada uma avaliação heurística da usabilidade do ambiente *Web* de programação, nesta foi possível verificar que a plataforma é esteticamente agradável e consistente em seu conteúdo. Ao final do estudo foi possível constatar que a solução apresentada pode servir como estímulo para os alunos e auxiliar na compreensão de conceitos de programação.

Palavras-chave: Robótica Móvel, Smartphone, Programação Visual, Pensamento Computacional, Ferramenta de Ensino

ABSTRACT

REIS, Cibele; SARMENTO, Henrique; ZARAMELLA, Vinicius. TOOL TO AID THE DEVELOPMENT OF COMPUTATIONAL THINKING: A ROBOTIC PLATFORM CONTROLLED BY SMARTPHONE. 118 f. Trabalho de Conclusão de Curso – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2014.

The difficulty of students in learning logic and programming languages leads to the creation of many technological solutions to assist in the teaching-learning process. Among these solution, two common approaches are robotics and graphical programming based languages. Researches indicate that these tools can help youth to think systematically and develop computational thinking, and that this result can be enhanced with the application of a problem based learning methodology. Therefore, this work describes the development of Plataforma Coffee, which involves a block programming Web environment and a smartphone as a robot controller unit. This platform can be easily expanded to work with various mobile devices and robotic kits. Another accomplishment of this work was the application of the platform in a classroom with the aim of evaluating the tool produced and the students' motivation while using it. Besides the tool evaluation by student, it was performed a heuristic usability evaluation of the Web programming environment, in which was verified that the platform is aesthetically pleasing and consistent in its content. At the end of this work, it was possible to observe that the presented solution can serve as a stimulus for students and assist in understanding programming concepts.

Keywords: Mobile Robotics, Smartphone, Visual Programming, Computational Thinking, Teaching Tool

LISTA DE FIGURAS

LISTA DE SIGLAS

ABET	*Accreditation Board for Engineering and Technology*
ADT	*Android Development Tools*
AJAX	*Asynchronous JavaScript XML*
API	*Application Programming Interface*
CRUD	*Create, Remove, Update, Delete*
CSS	*Cascading Style Sheets*
DC	*Direct Current*
DOM	*Document Object Model*
DTMF	*Dual-Tone Multi-Frequency*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
JSON	*JavaScript Object Notation*
MIT	*Massachusetts Institute of Technology*
MR	Modelo Relacional
MV*	*Model View Family*
MVC	*Model View Controller*
PBL	*Problem Based Learning*
PET	Programa de Educação Tutorial
PWM	*Pulse-Width Modulation*
REST	*Representational State Transfer*
RP	Robótica Pedagógica
SDK	*Standard Development Kit*
SPA	*Single Page Application*
TCLE	Termo de Consentimento Livre e Esclarecido
TTL	*Transistor-Transistor Logic*
UART	*Universal Asynchronous Receiver Transmitter*
UTFPR	Universidade Tecnológica Federal do Paraná
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Alunos que entram em cursos de graduação que envolvem em seu currículo o ensino de computação se deparam com matérias que abrangem linguagens e lógica de programação. Parte desses alunos tem dificuldade em entender o funcionamento dessa forma de solucionar problemas devido a diversos fatores que envolvem método de ensino e de estudo (GOMES; HENRIQUES; MENDES, 2008). Ainda segundo Gomes, Henriques e Mendes (2008), o aprendizado de linguagens de programação ainda é complicado devido ao fato da matéria ser apresentada de maneira expositiva (com apresentações, diagramas, textos, etc) quando deveria ser apresentada de maneira mais centrada na resolução de problemas.

Tendo em vista a dificuldade no aprendizado e compreensão de linguagens e lógicas de programação, surgiram, dentre outras soluções, as linguagens gráficas ou baseadas em *script* e aplicações robóticas voltadas ao ensino. Sendo assim, a proposta deste projeto é unir ambas as soluções citadas em apenas um produto e aplicá-lo em salas de aula.

Para alunos de disciplina de computação, a introdução de linguagens gráficas e, consequentemente, o desenvolvimento do pensamento computacional (entre outros aspectos, capacidade de solucionar problemas de forma lógica e sequencial), visa diminuir a dificuldade no aprendizado e compreensão de linguagens e lógica de programação. Ferramentas que utilizam linguagens gráficas como, por exemplo, Scratch (WANG; ZHOU, 2011) auxiliam jovens a raciocinar de forma sistemática e são acessíveis pela Internet, entretanto são limitadas à interface gráfica e seus resultados são restritos às ações do computador. Algo que normalmente desperta o interesse dos alunos é a utilização de ferramentas que apresentam resultados palpáveis, como por exemplo experiências em laboratório e, no caso de estudantes na área de computação, experiências com robótica (AROCA et al., 2013).

Uma abordagem que vem sendo amplamente utilizada na robótica é a substituição de um microcontrolador como unidade principal de processamento por um celular (AROCA; GONÇALVES; OLIVEIRA, 2012) e, como exemplos de projetos, há o CALLY

(YIM; SHAW, 2009) e o N-Bot (AROCA, 2012). Esta solução se mostra interessante devido ao crescimento da utilização de aparelhos celulares no Brasil, passando de 55,7 milhões de pessoas em 2005 para 271,1 milhões em 2013 e 136,45 cel/100 hab (TELECO, 2013).

A utilização de robótica para o ensino não é novidade, visto que há ferramentas muito conhecidas, tal qual o Lego Mindstorm (LEGO, 2013), entretanto, o custo de aquisição de utensílios do gênero nem sempre é acessível. Seguindo esta mesma ideia, Zanetti et al. (2013) propuseram um minicurso para introduzir a prática da robótica já em escolas, proporcionando aos alunos uma vivência real e contínua com tecnologias. Zanetti et al. (2013) ainda utilizaram no contexto do ensino a metodologia *Problem Based Learning* (PBL) que propõe que o aluno aprenda enquanto resolve um problema, realizando assim um papel ativo em seu aprendizado ao buscar uma solução, e trabalhar sozinho para isso (KOLMOS et al., 2007).

O produto final deste projeto une algumas tendências atuais, como a utilização de um celular como unidade integrante de um robô e a utilização de linguagens gráficas. Esta união resulta em um ambiente *Web* de programação em blocos e em uma base robótica móvel cujo processamento principal é realizado por um aparelho celular. Esta ferramenta, nomeada Plataforma Coffee, pode ser facilmente expandida visto que pode funcionar com diversos dispositivos móveis e kits robóticos desde que adaptações sejam feitas.

Este trabalho ainda propõe a aplicação da Plataforma Coffee em sala de aula com o intuito de analisar a resposta dos alunos em relação à ferramenta.

1.1 OBJETIVO GERAL E OBJETIVOS ESPECÍFICOS

O objetivo geral deste projeto é auxiliar no desenvolvimento do pensamento computacional e, consequentemente, auxiliar no ensino de linguagens de programação a alunos de computação por meio de um sistema que envolve um ambiente de programação em blocos e uma plataforma robótica.

Os objetivos específicos são:

- Desenvolver uma interface de programação cuja manipulação seja simples e de linguagem intuitiva;
- Desenvolver a comunicação entre a interface de programação e a plataforma robótica;
- Aplicar a ferramenta em salas de aula;

- Realizar um estudo de caso da ferramenta produzida.

1.2 ORGANIZAÇÃO DO DOCUMENTO

Este documento está dividido em sete capítulos. No presente capítulo é apresentada a justificativa da escolha do trabalho, bem como seus respectivos objetivos. O levantamento bibliográfico e a apresentação de projetos relacionados são apresentados no capítulo dois. Também são apresentados conceitos e referenciais teóricos de ferramentas necessárias para o desenvolvimento do projeto proposto. O capítulo três aborda a metodologia utilizada onde duas abordagens são levantadas. No capítulo quatro os recursos de hardware e software são descritos. O capítulo cinco tem como foco a implementação, os resultados do sistema desenvolvido e a avaliação preditiva da usabilidade. O capítulo seis trata da implantação do projeto em salas de aula e os resultados obtidos através da avaliação da ferramenta. Por fim, a última seção deste trabalho apresenta a conclusão e a discussão sobre trabalhos futuros.

2 LEVANTAMENTO BIBLIOGRÁFICO

Nesta seção são apresentados os fundamentos teóricos para o desenvolvimento do projeto proposto.

2.1 ENSINO EM ÁREAS DA COMPUTAÇÃO

Quando o ensino de linguagens e lógicas de programação é abordado, há diversos fatores que influenciam o sucesso do aprendizado por parte do aluno. Estes fatores vão desde o entendimento do problema abordado, do paradigma selecionado, passando pela linguagem escolhida e por fim as metodologias de ensino utilizadas (RIBEIRO; BRANDÃO; BRANDÃO, 2012).

Esta seção aborda os tópicos referentes a essas etapas do ensino da computação. Primeiramente na seção 2.1.1 são explicadas as dificuldades encontradas em ensinar e aprender a lógica de programação, em seguida, a seção 2.1.2 apresenta o conceito de pensamento computacional, logo após, na seção 2.1.3 a metodologia PBL é descrita e por fim, na seção 2.1.4, as ferramentas computacionais utilizadas no projeto são discutidas.

2.1.1 DIFICULDADES NO APRENDIZADO DE LINGUAGENS E LÓGICA DE PROGRAMAÇÃO

De acordo com Lima e Leal (2013), o processo de ensino e aprendizagem de programação de computadores é complexo para docentes e discentes e vem apresentando resultados insatisfatórios. Os autores ainda afirmam que o método de apresentação teórica de conceitos seguida de exemplos e listas de exercícios não são suficientes para motivar o envolvimento dos alunos.

Além da dificuldade em envolver os alunos na aprendizagem de linguagens e lógicas de programação, estes chegam nas universidades apresentando dificuldades de raciocínio lógico e pouca criatividade. Sendo assim, os autores ressaltam que para o aprendizado apresentar resultados positivos, é necessária uma mudança tanto na postura do aluno

quanto na do professor, além de um modelo de construção de projetos contextualizados (LIMA; LEAL, 2013).

Os autores Gomes, Henriques e Mendes (2008) evidenciam uma diversidade de fatores que complicam o aprendizado de linguagens e lógicas de programação, entre eles a utilização de métodos de ensino não centralizados nos alunos, turmas grandes que dificultam a promoção de um ensino personalizado e a supervisão inadequada às necessidades de cada aluno. Lima e Leal (2013) complementam as dificuldades apresentadas por Gomes, Henriques e Mendes (2008) ao afirmar que a programação de computadores exige dedicação e envolve raciocínio lógico e capacidade de abstração, capacidades estas nem sempre presentes nos alunos de graduação. Lima e Leal (2013) ainda afirmam que estas capacidades devem ser exploradas e incentivadas desde o Ensino Fundamental e Médio. Esta discussão está presente em diversos meios e é relacionada ao termo "Pensamento Computacional", abordado na próxima seção.

2.1.2 PENSAMENTO COMPUTACIONAL

De acordo com Wing (2006), pensamento computacional é uma habilidade que não apenas cientistas da computação devem dominar. Este conceito envolve a solução e reformulação de problemas, a projeção de sistemas, entre outros, sendo assim uma habilidade analítica que deveria pertencer a todos desde os primeiros anos de vida (WING, 2006).

O pensamento computacional é a capacidade de abstração e decomposição de problemas selecionando aspectos relevantes e simplificando assim suas soluções. Wing (2006) ainda afirma que para pensar computacionalmente são necessárias algumas características, como:

- A capacidade de modelar problemas e sistemas passando por vários níveis de abstração, e não o ato de programação;
- A maneira com que os humanos resolvem problemas. Não há um poder computacional sem a inteligência e criatividade de um ser humano.

Wing (2008) complementa os tópicos acima afirmando que o pensamento computacional é um tipo de pensamento analítico que abrange diversas áreas como matemática, engenharia e ciência. Sendo assim, a autora ainda diz que o pensamento computacional não necessita de máquinas ou computadores e futuramente não será utilizado apenas para

auxiliar na modelagem de sistemas cada vez mais complexos, mas também para analisar a quantidade massiva de dados gerados e coletados pela humanidade.

2.1.3 METODOLOGIA PBL

O uso de uma metodologia de estudo voltada à resolução de problemas torna-se relevante para a formação de engenheiros profissionais, visto que requisitos da *Accreditation Board for Engineering and Technology* (ABET) [1] para contratação de pessoal envolvem habilidades de processo como, por exemplo (KOLMOS et al., 2007):

- Análise e resolução de problemas;
- Gerenciamento de projetos e liderança;
- Habilidades analíticas e pensamento crítico;
- Conhecimento multidisciplinar;
- Inovação e criatividade;
- Habilidades sociais.

Além disso, Mills e Treagust (2003) afirmam que o ensino convencional não tem capacitado os engenheiros adequadamente com as habilidades requeridas pela indústria: comunicação forte; trabalho em equipe; perspectiva de questões sociais, ambientais e econômicas; e capacidade de aplicação de conhecimentos práticos. Estudos mostram que alunos que aprendem através da didática tradicional absorvem pouca informação, e alunos que aprendem através da resolução de problemas podem contextualizá-los e reforçar o conhecimento através desta resolução (KOLMOS et al., 2007).

Além da resolução de problemas complexos, o processo envolve uma forma de ensino centrada no aluno, e não no professor. Isso ocorre, pois o aluno participa mais na aula através do compartilhamento de suas decisões, pensamentos e dúvidas a respeito de um problema, e não somente através de perguntas como no ensino tradicional. Os alunos também são incentivados a se reunirem em pequenos grupos e discutirem sobre possíveis resoluções do problema, portanto, eles desenvolvem suas habilidades sociais. Outra diferença em relação à didática tradicional é que o papel do professor muda para

[1] Incorporação norte-americana que avalia qualidades que estudantes de engenharia e tecnologia devem ter ao terminar seus cursos em universidades

facilitador agindo como instrutor de procedimentos e consultor de conteúdos adicionais, tornando-se um guia para a resolução dos problemas.

Para a aplicação dessa metodologia é necessário haver preparação, tanto do material quanto dos facilitadores que devem ao invés de dar uma palestra, agir como consultores fazendo com que os alunos reflitam sobre suas conclusões através de perguntas como "Como você sabe disso?" e "Você está fazendo quais suposições?" (KOLMOS et al., 2007). Alunos podem resistir à essa nova metodologia de estudo, porém geralmente a aceitam e tornam-se mais confiantes, visto que a metodologia em geral torna os estudantes mais responsáveis por suas escolhas.

Um problema interessante segundo Kolmos et al. (2007) deve possuir as seguintes características:

- É desafiador e tem relação com o mundo real;
- É complexo e estruturado;
- Gera múltiplas hipóteses para resolução;
- Requer esforço do grupo;
- É consistente com resultados esperados;
- É construído a partir de conhecimentos passados dos alunos;
- Desenvolve a habilidade cognitiva dos alunos;

O processo para desenvolver, implementar e avaliar um problema é (KOLMOS et al., 2007):

- Identificar a ideia principal;
- Descrever os resultados esperados;
- Estruturar um problema complexo;
- Dividir o problema em etapas;
- Desenvolver um tutorial;
- Auxiliar estudantes em identificar recursos.

O processo para a aplicação da metodologia PBL é mostrado na figura 1. Inicialmente o facilitador apresenta o cenário do problema aos alunos, que por sua vez identificam os fatos envolventes. Com esse segundo passo, eles devem ser capazes de representar a problemática e criar hipóteses de soluções para esta. Logo após, os alunos podem identificar deficiências em seus conhecimentos a respeito do escopo do problema. Neste caso eles devem pesquisar e entender sobre o assunto para então aplicar os novos conhecimentos adquiridos. Finalmente, os estudantes avaliam suas hipóteses baseando-se no que aprenderam de novo e, posteriormente, refletem sobre suas conclusões.

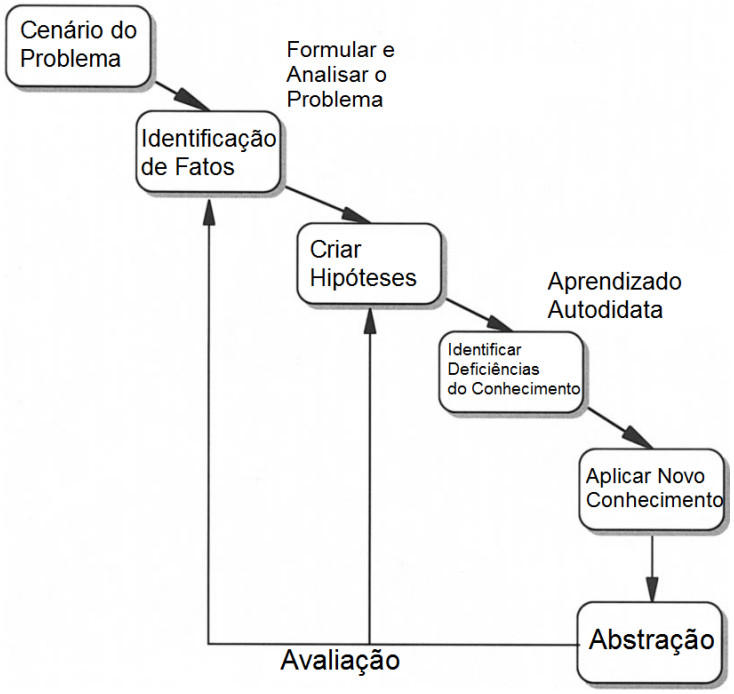

Figura 1: Diagrama do Ciclo do Aprendizado Baseado em Problemas
fonte: adaptado de Mills e Treagust (2003).

2.1.4 FERRAMENTAS COMPUTACIONAIS

De acordo com Neto e Schuvartz (2007), a utilização de ferramentas computacionais pode tornar o processo de aprendizado dos alunos mais dinâmico, ágil e prazeroso, aproximando a teoria da prática. Apesar das ferramentas computacionais apresentarem as vantagens anteriormente citadas, Neto e Schuvartz (2007) afirmam que ainda é um desafio fazer com que sua utilização realmente tenha um impacto quando aplicada na área da informática. Além disso, estas ferramentas devem se adaptar às necessidades de cada perfil de aluno, providenciando um atendimento individualizado e respeitando o ritmo de cada usuário do ambiente (NETO; SCHUVARTZ, 2007). Nesta seção são abordados dois

tipos de ferramentas computacionais: a programação visual e a robótica.

2.1.4.1 Programação Visual

Atualmente os programas de computador são na maior parte ainda desenvolvidos em linguagens textuais, pois estas mostram-se como a melhor maneira para dar instruções precisas ao computador. Porém, nesta abordagem existe uma defasagem grande entre o programa escrito e o que será visualizado em sua execução (HEAVEN, 2012). Esta defasagem pode ser reduzida através de soluções que permitem o usuário analisar os resultados de instruções dinamicamente. Uma dessas soluções é a programação visual que é intuitiva, de fácil entendimento e mais atrativa para alunos em fase inicial de aprendizagem (UTTING et al., 2010). Outra vantagem é que o programador pode focar na análise do problema e sua solução, não sendo necessária a verificação de erros de sintaxe de uma linguagem textual específica (GROVER; PEA, 2013).

Programação visual é uma linguagem que prioriza em seu ambiente de desenvolvimento uma sintaxe majoritariamente visual. Dentre estas linguagens de caráter educacional podemos citar Alice, Greenfoot e Scratch (UTTING et al., 2010). Segundo Utting et al. (2010), estas três linguagens, também consideradas como sistemas, permitem que os usuários escrevam programas relacionados aos seus interesses, como histórias, jogos, simulações, entre outros (UTTING et al., 2010).

2.1.4.2 Robótica

De acordo com Pio, Castro e Júnior (2006), apesar de grande parte do esforço dos professores e alunos ser adquirir habilidades necessárias para a produção de software, os produtos da prática de programação em cursos de graduação passam a ter forma somente após a passagem dos alunos por várias disciplinas, e ainda assim, pouco do que é realizado faz parte do mundo físico e tangível em que vivemos. A partir deste cenário, Zanetti et al. (2013) propõem a inclusão da robótica e de novas tecnologias como um elemento motivador que possibilita a participação dos alunos no processo de aprendizagem. Os autores ainda afirmam que, além das questões de motivação e compreensão do conhecimento, a inclusão da robótica no ensino pode explorar o trabalho em equipe, expressão oral e escrita e outras competências além daquelas que os currículos escolares empregam.

A utilização da Robótica Pedagógica (RP) faz com que os alunos desenvolvam habilidades para a solução de problemas, o desenvolvimento baseado em projetos e o trabalho em equipes (PIO; CASTRO; JÚNIOR, 2006). Além disso, a RP é um processo interativo e conciliatório entre o concreto e o abstrato, faz com que a resolução de um pro-

blema envolva etapas como: concepção, implementação, construção, automação e controle de um mecanismo (ZANETTI et al., 2013).

Gaudiello e Zibetti (2012), além de mencionarem a Robótica Pedagógica como uma tecnologia interativa e transparente, introduziram a diferença entre "robô para utilizar" e "robô para desenvolver". Os autores ainda afirmam que o aluno deixa de ser apenas um consumidor de tecnologia e passa a ser um autor da mesma. Neste mesmo contexto, Zanetti et al. (2013) ainda mencionam a exploração "construcionista" do conhecimento, auxiliando o aluno a criar e se apropriar de suas próprias ideias.

A RP permite a criação de um ambiente educativo imersivo e alternativo aos métodos tradicionais e pode utilizar objetos semelhantes aos brinquedos que as crianças possuem em casa, como por exemplo, os blocos de montagem Lego (GAUDIELLO; ZIBETTI, 2012).

A RP é uma ferramenta promissora cuja utilização permite aos professores e alunos construir uma visão simplificada da robótica, realizar projetos cada vez mais complexos e diversos conforme o conhecimento é consolidado e motivar os estudantes a adotarem métodos de aprendizagem diferentes. Além disso, seus usuários desenvolvem competências cognitivas e metacognitivas ligadas, respectivamente, à resolução de problemas e ao pensamento computacional (GAUDIELLO; ZIBETTI, 2012).

Por fim, cabe ressaltar que a utilização da RP requer que o professor deixe de ser um distribuidor incontestável de conhecimento e passe a mediar a interação entre os alunos e os robôs, ou ainda, as ideias dos alunos e sua praticidade de execução (GAUDIELLO; ZIBETTI, 2012).

2.2 TECNOLOGIAS

Esta seção aborda as tecnologias envolvidas no desenvolvimento do projeto em questão.

2.2.1 DISPOSITIVOS MÓVEIS

De acordo com Santos e Silva (2013), dispositivos móveis possuem características semelhantes aos computadores tradicionais, diferenciando-se no tamanho compatível para ser transportado no bolso. Ainda segundo a autora, a mobilidade destes aparelhos é dada devido à sua comunicação por meio de redes sem fio.

Santos e Silva (2013) afirmam que a tecnologia dos dispositivos móveis se encontra em expansão, assim como o aumento do poder de processamento agregado destes aparelhos. Tais mudanças abriram novos caminhos para a utilização destes dispositivos para diversas finalidades, sendo uma delas a educação.

O alto grau de mobilidade e transparência dos dispositivos móveis propicia novos métodos de interação que facilitam o processo de ensino-aprendizagem, principalmente para crianças (SANTOS; SILVA, 2013). Os autores afirmam que o desenvolvimento de aplicativos educacionais para dispositivos móveis no Brasil ainda é pouco explorado, embora seja promissor.

Aroca (2012) interpreta o alto poder computacional dos *smartphones* como um bom recurso para o controle de robôs e comprova esta tendência (GUIZZO; DEYLE, 2012). Além do alto poder de processamento, os *smartphones* normalmente possuem câmera, bússola, acelerômetro, Wi-Fi, auto-falantes, microfones, entre outros sensores que podem ser úteis para a robótica. Outro fator interessante é a produção destes aparelhos em grande escala, tornando mais barato comprar um telefone com todos esses recursos do que um microcontrolador e cada um dos sensores mencionados (AROCA, 2012).

Além dos recursos de hardware, os aparelhos celulares atuais também possuem sistemas operacionais e ambientes com facilidades para implementação de sistemas robóticos (AROCA, 2012). Um exemplo destes sistemas é o Android.

Android é o sistema operacional de aparelhos móveis mais utilizado do mundo (ANDROID, 2013). Este sistema operacional é baseado em Linux e permite, portanto, o desenvolvimento de aplicativos em diversas linguagens suportadas pelo mesmo, como Java, Python, C, entre outras (AROCA, 2012).

A depuração dos códigos desenvolvidos para o Android pode ser feita via cabo USB ou até mesmo via rede, permitindo assim, uma correção de aplicações à distância (AROCA, 2012). Além disso, há diversas bibliotecas que podem ser aproveitadas ao se levar em conta o desenvolvimento e a pesquisa em robôs; e facilitam a implementação de tarefas que poderiam ser complicadas, como por exemplo o comando por voz (AROCA, 2012).

Outra opção para desenvolver aplicativos móveis é a utilização do *framework* Apache Cordova (CORDOVA, 2014). Este *framework* é um conjunto de *Application Programming Interface* (API) que combinado com jQuery Mobile, por exemplo, permite o desenvolvimento de aplicações utilizando apenas *HyperText Markup Language* (HTML), *Cascading Style Sheets* (CSS) e JavaScript e torna desnecessária a utilização de linguagens

nativas como Java e Objective-C.

O Apache Cordova proporciona o acesso a funções nativas dos aparelhos celulares a partir de bibliotecas JavaScript, permitindo assim a utilização dos sensores e atuadores dos dispositivos móveis sem implicar na limitação de seus respectivos sistemas operacionais. O *framework* está disponível para as plataformas Android, iOS, Blackberry, Windows Phone, Palm WebOS, Bada e Symbian (CORDOVA, 2014).

Este *framework* foi construído para ser um sistema extensível. Tendo isso em vista, o Cordova separa o núcleo, mínimo necessário para executar o *framework*, do resto das funcionalidades que são adicionadas através de *plugins* conforme a necessidade de cada aplicativo. Para as funcionalidades mais comuns, como por exemplo o acesso a informações de localização, existe uma série de *plugins* oficiais disponibilizados no site do *framework*. Nos casos em que seja necessária a extensão das bibliotecas nativas, o Cordova oferece a possibilidade de se desenvolver seus próprios *plugins*.

Os *plugins* do Cordova consistem em três partes, sendo elas o código nativo, uma interface JavaScript e um arquivo de descrição ("plugin.xml"). O código nativo deve ser implemetado em todas plataformas que se deseje dar suporte. A interface JavaScript é única para todas as plataformas, permitindo assim a padronização do código na visão do desenvolvedor. Já o arquivo de descrição contém todos os dados necessários para que o *framework* possa compilar corretamente o código, gerando assim os aplicativos nativos para cada sistema suportado.

2.2.2 TROCA DE DADOS ATRAVÉS DO CANAL DE ÁUDIO

De acordo com Aroca, Burlamaqui e Gonçalves (2012), a troca de dados via canais de áudio é uma técnica útil em diversas aplicações e pode ser interessante se utilizada em dispositivos móveis.

A troca de dados via canais de áudio pode ser aplicada, dentre outras maneiras, a partir da utilização do *Dual-Tone Multi-Frequency* (DTMF) ou da comunicação serial.

O DTMF foi criado na década de 50 e continua sendo amplamente utilizado até os dias atuais em projetos de novos telefones. Esta técnica codifica dezesseis caracteres diferentes através de um par de frequências sonoras e se mostra robusta a ruídos e outros sons que não caracterizam um dígito DTMF (AROCA; BURLAMAQUI; GONÇALVES, 2012).

A comunicação serial consiste na transmissão de bits sequencialmente através de

um canal de comunicação ou barramento. A comunicação serial utilizada neste projeto é assíncrona e requer a implementação de um protocolo explicado na seção 3.4.3.

2.3 METODOLOGIAS DE AVALIAÇÃO DE FERRAMENTAS

Segundo Sharp, Rogers e Preece (2007), o processo de avaliação mostra o quanto uma ferramenta atende às necessidades do usuário em um ambiente específico. Um exemplo de projeto refere-se ao *Olympic Message System*[2], onde três princípios foram seguidos para sua realização:

- Focar nos usuários e suas tarefas;
- Observação, medições e análise de performance de usuários com o sistema;
- Projetar iterativamente.

Nesse exemplo, a avaliação foi feita através da análise da adaptação do usuário com a ferramenta. Quanto mais rápida essa adaptação, melhor é o sistema. Outro ponto importante é a iteratividade do projeto, visto que a avaliação não deve ocorrer somente na versão final do produto, pois sempre há melhorias à serem feitas.

Um dos motivos para avaliar um produto é a verificação de que o usuário pode utilizá-lo com sucesso, e de que ele aprecia a ferramenta. Projetos que utilizam essa metodologia atingem seus objetivos propostos com eficácia, pois os desenvolvedores podem corrigir problemas que foram detectados pela avaliação. Há dois tipos de frequência em que podem ocorrer as avaliações, durante (formativas) ou após o final (sumativa) de um produto.

Um dos paradigmas para avaliação chama-se "*Quick and dirty*", sua principal característica é obter comentários de usuários ou consultores, e verificar se as ideias do projeto satisfazem as necessidades dos usuários. Não há preocupação em documentar cuidadosamente essas observações, portanto os encontros são geralmente informais e ocorrem em qualquer momento do projeto. Um segundo paradigma é o teste de validação, na qual usuários são avaliados pelo tempo de resolução e quantidade de erros cometidos em um problema. Essa abordagem é feita em ambientes controlados com um protótipo ou produto final. O terceiro modelo é chamado estudos de campo realizado em um ambiente

[2]Um sistema de troca de mensagens para esportistas de diversos países que participaram da Olimpíada de 1984, e que poderiam ainda não ter contato com computadores. Além disso, não houve treinamento algum para seu uso (GOULD et al., 1987).

menos controlado onde os avaliadores fazem anotações dos usuários que utilizam a ferramenta, geralmente realizada nas etapas iniciais de um projeto. A quarta abordagem é a preditiva, sua principal característica é que usuários não são envolvidos, e é feita de forma que os próprios avaliadores façam a análise do projeto criando problemas e verificando em quanto tempo demoram para resolvê-los.

Exemplos de técnicas amplamente utilizadas para avaliação são (SHARP; ROGERS; PREECE, 2007):

- Observar usuários através de notas, áudios, vídeos e logs;
- Perguntar a opinião de usuários sobre o produto através de questionários e entrevistas;
- Perguntar a especialistas suas opiniões e sugestões para problemas;
- Testar a performance dos usuários através da comparação de *designs* diferentes;
- Modelar a performance de usuários para predizer a eficácia da interface de usuário.

O foco neste projeto são os paradigmas de Estudos de Campo e Avaliação Preditiva.

2.3.1 ESTUDOS DE CAMPO

De acordo com McGill (2012), são diversas as ferramentas de ensino aplicadas em sala de aula para ensinar programação, entretanto são poucas as análises que verificam o efeito dessas ferramentas na motivação dos alunos. Este trabalho contempla um estudo em relação à motivação dos alunos utilizando o paradigma dos Estudos de Campo.

Esse paradigma é realizado em ambientes naturais para o usuário, e seu objetivo é a compreensão do que usuários fazem naturalmente com o artefato e como a tecnologia os afeta. Os principais objetivos dessa avaliação são (SHARP; ROGERS; PREECE, 2007):

- Identificar oportunidades para novas tecnologias;
- Determinar requerimentos para *design*;
- Facilitar a introdução de tecnologia;
- Avaliar tecnologia.

Nesse modelo um avaliador coleta dados de usuários que utilizam os artefatos. Esse avaliador pode ser interno ou externo à avaliação. No primeiro caso o avaliador é um observador, coletando dados dos usuários que são analisados quantitativamente e qualitativamente. No segundo caso o avaliador é um participante da avaliação e seu papel é compreender o que ocorre em um ambiente social que está sendo analisado.

As técnicas utilizadas em Estudos de Campo podem ser através da observação dos usuários e de perguntas aos mesmos via entrevistas ou discussões.

Para este projeto, o Estudo de Campo visa analisar o perfil dos estudantes em relação à afinidade com programação e expectativas de utilizar a plataforma de ensino Coffee. Para uma análise posterior à aplicação da ferramenta, serão avaliados os aspectos de motivação citados por McGill (2012), sendo eles: atenção, relevância, confiança e satisfação.

2.3.2 AVALIAÇÃO PREDITIVA DA USABILIDADE

Segundo Sharp, Rogers e Preece (2007), nesse tipo de avaliação especialistas utilizam seu próprio conhecimento para detecção de possíveis problemas de usabilidade do produto. Outro ponto considerável desse método de avaliação é que usuários do produto não precisam estar presentes.

Especialistas podem avaliar o produto através de heurísticas, opiniões baseadas em experiências próprias, conhecimentos através de pesquisas estudadas, e usabilidade do produto.

As técnicas da Avaliação Preditiva são opiniões dos especialistas à respeito de suas heurísticas que podem predizer a eficácia de uma interface, realizada geralmente no começo de um projeto; e modelagem do desempenho de tarefas que prediz a eficácia de uma interface.

A usabilidade de um produto refere-se à interação do usuário com um determinado artefato, preocupando-se se aquele consegue atingir seus objetivos com facilidade e satisfação. Dentro desse contexto, há cinco características que definem um produto com uma usabilidade agradável ao usuário (NIELSEN, 1993).

- Facilidade de Aprendizagem: Envolve a facilidade de uso do produto por um usuário. Assim, o usuário é capaz de tornar-se experiente rapidamente caso o produto seja de fácil aprendizagem.

- Eficiência: Envolve a alta produtividade que o produto pode oferecer a um usuário experiente. Ou seja, um usuário experiente pode completar seus objetivos utilizando poucos recursos (tempo e esforços em geral).
- Facilidade de Relembrar: As etapas de uso do sistema precisam ser fáceis de relembrar, portanto esse atributo verifica se o entendimento do produto pelo usuário mesmo após o seu afastamento é fixado.
- Erros: Envolve a pequena taxa de erros que ocorrem no sistema. Além disso, trata também da capacidade do sistema retornar a um estado estável após alguma ocorrência.
- Satisfação Subjetiva: Capacidade do sistema oferecer uma interface agradável ao usuário.

Além disso, a norma ISO 9241-11 (ISO, 1998) especifica o termo usabilidade como uma medida que avalia se usuários atingem determinados objetivos com eficácia, eficiência e satisfação em um contexto específico de uso.

2.4 TRABALHOS CORRELATOS

Esta seção descreve os trabalhos correlatos ao proposto nesta monografia.

2.4.1 SCRATCH

Scratch é um ambiente de programação visual desenvolvido pelo Massachusetts Institute of Technology (MIT). Este projeto teve início em 2003 e seu software e *Website* se tornaram públicos apenas em 2007. O sistema permite a criação de diversos tipos de alocação, como histórias animadas, jogos, cartões de datas comemorativas, tutoriais, entre outros. No Scratch a programação é feita ao agrupar blocos coloridos de comandos, permitindo ainda que os projetos nele desenvolvidos sejam salvos no computador ou compartilhados no site do ambiente (MALONEY et al., 2010).

O Scratch oferece a possibilidade de controlar movimentos de animações através dos blocos mencionados. Há um fórum para discussão de projetos, e em sua *Wiki* (MIT, 2013b), há diversos tutoriais de como utilizar a ferramenta e criar *scripts* sendo possível criar animações, e blocos que representam animações de objetos. A figura 2 mostra o ambiente de programação da ferramenta.

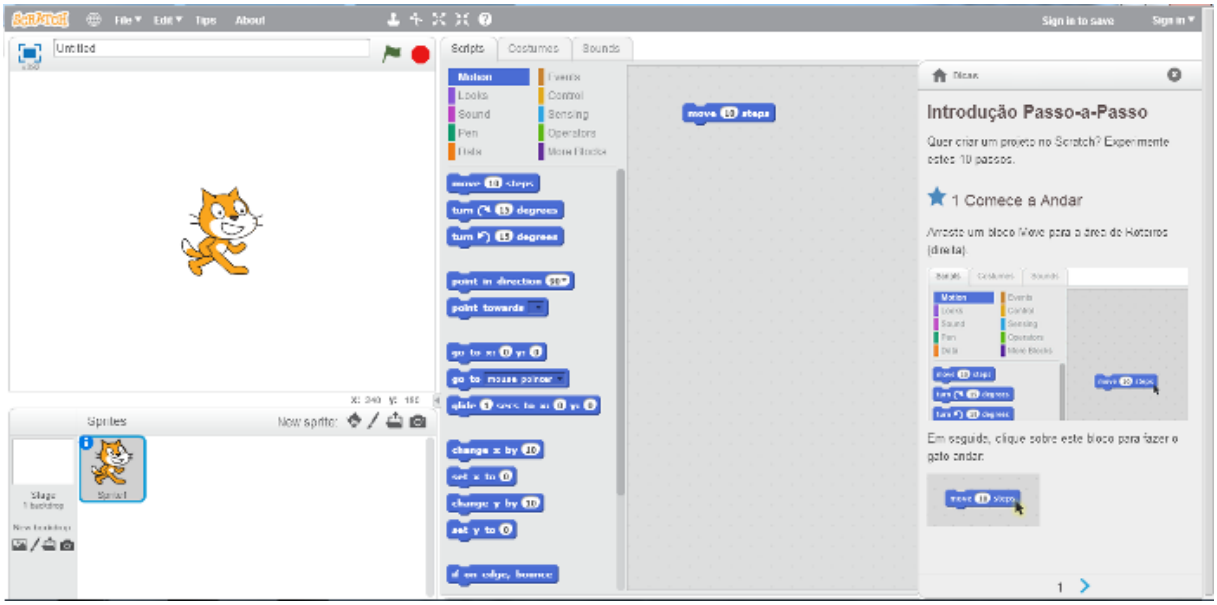

Figura 2: Interface Gráfica do Aplicativo Web Scratch
fonte: MIT (2013a)

A ferramenta é gratuita e proporciona aos usuários o compartilhamento e acesso de diversos projetos, possibilitando assim uma experiência social encorajante que introduz a lógica de programação àqueles que não possuem contato prévio na área (MALONEY et al., 2010).

Wang e Zhou (2011) utilizaram o Scratch para desenvolver o pensamento computacional de alunos do ensino médio. Inicialmente o modo de ensino de programação proposto desperta o interesse dos alunos através de um problema em que o Scratch possa ser aplicado e que seja familiar para eles. O professor então guia os estudantes para compreender o problema proposto e analisar como a solução pode ser feita utilizando-se o Scratch. Logo após, o conhecimento adquirido através do Scratch é transferido para o ambiente de programação tradicional. Além disso, a solução de vários exercícios e o compartilhamento é incentivado pelo professor.

2.4.2 LEGO MINDSTORM

O kit Lego Mindstorms foi introduzido no mercado em 1998, porém sua versão mais recente foi lançada em setembro de 2013, o Lego Mindstorm EV3. Estes kits normalmente vêm equipados de blocos de montar, engrenagens, motores, sensores, interface programável, entre outros. A interface programável é um microprocessador e sua versão mais recente é a EV3 que promete uma programação mais intuitiva a partir de um meca-

nismo gráfico (LEGO, 2013). A figura 3 mostra as peças do kit do Lego, e na figura 4 é possível observar o funcionamento do ambiente de programação.

Figura 3: Kit Lego Mindstorms EV3
fonte: LEGO (2013)

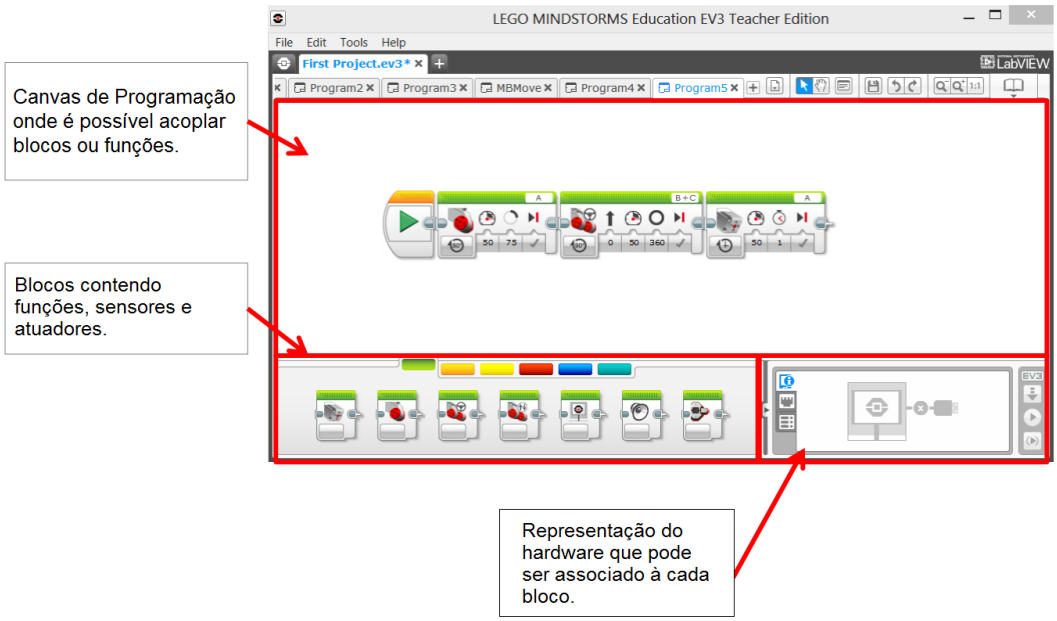

Figura 4: Ambiente de Programação do Lego EV3
fonte: Adaptado de LEGO (2014)

O kit Lego Mindstorms é usado amplamente na educação de Ciência da Computação para alunos do ensino médio e universitários na Alemanha (KISS, 2010). O sistema pode ser usado para o ensino de programação em dois tipos diferentes. Primeiramente, pode ser utilizada uma interface visual baseada em blocos para a programação do robô,

esse sistema é chamado *Robotic Invetion System*. Essa estrutura permite que os usuários enviem comandos de movimentação ao Lego através desses blocos. Além disso, é possível utilizar laços, estruturas condicionais e *threads*.

A segunda forma de programação é através do sistema *Lego Java Operating System* que permite que o usuário utilize ferramentas Java para escrever o programa de controle do robô. Segundo Kiss (2010), mulheres apresentaram dificuldades no aprendizado de programação na Alemanha, e seria possível aplicar o Lego para auxiliar essa questão.

Em outra situação, como introdução à programação para alunos do ensino médio e primeiro ano da faculdade, o kit Lego Mindstorms foi utilizado para desenvolver as habilidades para resolução de problemas (KATO; HIROYUKI, 2010). O curso foi divido em quatro estágios dos quais alunos de diferentes níveis de escolaridades foram direcionados. Cada estágio representa diferentes níveis de problemas ou desafios em que os alunos deveriam se reunir em grupos para a sua resolução. Também foi desenvolvida uma central online com uma interface amigável para comunicação do professor e grupos de alunos para sua melhor comunicação.

No segundo estágio desse curso foi utilizado um ambiente de programação visual e, logo após, uma introdução à linguagem C foi realizada. As atividades nesse estágio foram voltadas ao ensino através de jogos e envolveram a ambientação com a ferramenta, controle dos atuadores, e uso dos sensores do robô.

2.4.3 CALLY

Cally é um robô de quatro rodas e dois braços que possui um celular como processamento principal. Para sua base mecânica foi utilizado o Bioloid Expert Toolkit (YIM; SHAW, 2009). O robô possui funcionalidades de reconhecimento de objetos, expressões faciais e gestos com mãos. A câmera é responsável por fazer todas as detecções e, além disso, o robô detecta ações humanas e faces demonstradas no celular de outros robôs Cally (YIM; SHAW, 2009).

Figura 5: Robô Cally
fonte: Yim e Shaw (2009)

2.4.4 N-BOT

N-Bot é uma plataforma robótica educacional cujo objetivo principal é fazer com que a construção de robôs se torne popular entre estudantes de todas as idades (AROCA; GONÇALVES; OLIVEIRA, 2012). O controle deste robô é todo feito por áudio, visto que este é utilizado para enviar comandos para atuadores e coletar os estados dos sensores. Sua plataforma desenvolvida não necessita de softwares adicionais ou drivers especiais, tornando assim sua utilização mais simples para aqueles que não possuem conhecimento ou permissões, no caso de computadores de universidades para instalar softwares (AROCA; GONÇALVES; OLIVEIRA, 2012).

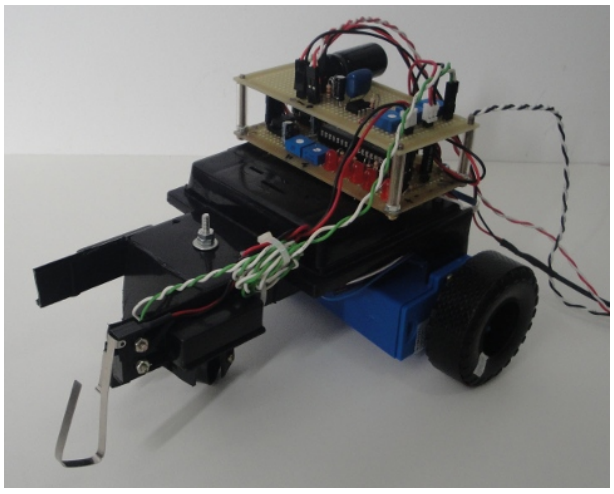

Figura 6: Robô N-Bot
fonte: Aroca (2012)

De acordo com Aroca (2012), o N-Bot pode utilizar um *smartphone* como unidade de controle. Em seu ambiente de programação, é possível utilizar o Blockly (BLOCKLY, 2013) que é uma linguagem baseada em blocos. Outra opção é a programação diretamente do *smartphone* através de *scripts* para Android utilizando SL4A (LUCAS; GREYLING, 2011). Ainda é possível possuir um computador como processador do robô, neste caso, é possível programá-lo em JavaScript através de um aplicativo *Web*. As três soluções propostas envolvem o envio de sinais de áudio para o robô.

O principal objetivo educacional do projeto foi aumentar o interesse dos alunos em robótica (AROCA, 2012). Duzentos e quatro alunos do ensino superior com média de 21 anos de idade atenderam a uma aula que explicava como construir o robô, e questionários foram aplicados previamente e posteriormente à aula para verificar se a opinião dos alunos havia mudado. Ao final da palestra 71% dos alunos disseram que seria interessante possuir o próprio robô, sendo que na pesquisa prévia somente 58.5% havia esse interesse.

2.4.5 ROMO

Romo é um robô que utiliza um *smartphone* iOS como base de processamento (ROMO, 2014). Apesar do robô ser considerado um brinquedo, no site oficial é possível baixar a *Standard Development Kit* (SDK) do projeto. Sendo assim, desenvolvedores podem desenvolver seus próprios projetos utilizando o sistema. As principais funcionalidades do robô envolvem o uso da câmera e o *touchscreen* na tela do celular para interação do usuário, além disso, ele possui uma representação amigável com feições humanas que podem ser comandadas para simular sentimentos e comportamentos humanos graficamente. O robô também possui o controle da inclinação em que o *smartphone* é posicionado, chamada de *tilt* pelos criadores, através disso é possível fazer um controle mais robusto do ângulo da câmera do *smartphone*.

Também é possível utilizar um *tablet* no lugar de um *smartphone* sem que haja grandes diferenças. O aplicativo para dispositivos móveis possui diversos jogos para a interação com o robô. É possível utilizar um sequenciador de instruções baseado em blocos para a programação do robô. Este sistema é mais focado em funcionalidades e não em estruturas de programação.

Figura 7: Robô Romo
fonte: Romo (2014)

2.4.6 SMARTBOT

O Smartbot é um robô que pode utilizar um *smartphone* ou um microcontrolador (como o Arduino (ARDUINO, 2014)) como base de processamento e pode ser utilizado para o entretenimento e educação (SMARTBOT, 2014). O *smartphone* pode possuir diferentes sistemas operacionais como Android, Iphone, Ipod e Windows Phone. Ele também oferece uma SDK para a programação para desenvolvedores programarem nos diversos sistemas operacionais.

As principais funcionalidades desenvolvidas para o robô são (SMARTBOT, 2014):

- Controlar o robô utilizando outro *smartphone*;
- Utilizar o robô para tirar fotos do usuário automaticamente;
- Fazer o robô seguir um desenho de rotas planejadas através de outro *smartphone*;
- Rastreador de faces;
- Jogos envolvendo a interação com o robô.

No site oficial (SMARTBOT, 2014) é possível observar que planeja-se adicionar uma programação baseada em blocos nessa plataforma sendo possível utilizar as operações de controle do robô, sensores, temporizadores, condições, laços, cálculos, e variáveis. Na tela do *smartphone* o robô possui feições humanas que são amigáveis ao usuário.

Figura 8: Robô Smartbot
fonte: Smartbot (2014)

2.4.7 RESUMO

A tabela 1 mostra uma visão geral dos trabalhos correlatos ao trabalho proposto em relação aos aspectos considerados relevantes para este projeto. A tabela 2 compara diferentes ambientes de programação entre os projetos, e a tabela 3 faz uma comparação dos robôs utilizados nas abordagens.

Projeto	**Possui robô**	**Voltado ao ensino de programação**	**Utiliza celular**	**Permite compartilhamento de projetos**	**Publicações científicas**
Scratch	Não	Sim	Não	Sim	[MALONEY et al. 2010, WANG; ZHOU 2011]
Cally	Sim	Não	Sim	Não	Não encontrado
N-Bot	Sim	Sim	Sim	Não	[AROCA; GONÇALVES; OLIVEIRA 2012, AROCA 2012]
Lego Mindstorm	Sim	Sim	Não	Não	[KISS 2010, KATO; HIROYUKI 2010]
Romo	Sim	Não	Sim	Não	Não encontrado
Smartbot	Sim	Não	Sim	Não	Não encontrado
Coffee	Sim	Sim	Sim	Sim	-

Tabela 1: Comparação Geral Entre Trabalhos Correlatos

Projeto	**Ambiente de programação para usuários**
Scratch	Acessado pela *Web*. Baseado em blocos. Possibilidade de controle de animações e resolução de problemas através da programação visual.
Cally	Não há interface de programação para o usuário.
N-Bot	Acessado pela *Web*. Baseado em blocos utilizando o Blockly. Possibilidade de programar em C.
Lego Minds-torms	Requer instalação do ambiente de programação. Baseado em blocos. Possibilidade de programar componentes de hardware do sistema.
Romo	Requer instalação de aplicativo no *smartphone*. Possibilidade de enviar comandos ao robô. Programação através de um sequenciador baseado em blocos, porém não possui estruturas avançadas de programação. Possui SDK para desenvolvedores.
Smartbot	Requer instalação de aplicativo no *smartphone*. Possibilidade de enviar comandos ao robô, ainda não possui interface de programação para os usuários. Possui SDK para desenvolvedores.
Coffee	Ambiente de programação acessado pela *Web*. Baseada em blocos utilizando Blockly. Possibilidade de visualização do código gerado em Javascript.

Tabela 2: Tabela Comparativa de Ambientes de Programação

Projeto	**Custo (US$)**	**Principais características**
Cally	Não encontrado	Robô possui dois braços mecânicos e quatro rodas.
N-Bot	14	Motores controlados por canal de áudio. Utiliza *smartphone* como principal processamento. Possibilidade de adição de sensores no robô.
Lego Mindstorm EV3	350	Possibilidade de montar o robô de formas diferentes através das peças do kit. Possibilidade de adição de sensores no robô.
Romo	129	Robô possui feições humanas na tela do *smartphone*, possibilidade de controlar a inclinação em que o *smartphone* é posicionado. Funcionalidades que utilizam a câmera do *smartphone*.
Smartbot	238	Robô possui feições humanas na tela do *smartphone*. Funcionalidades que utilizam a câmera do *smartphone*.
Coffee	Variável[3].	Utiliza o *smartphone* como principal processamento. Motores controlados por um microcontrolador. Comunicação entre *smartphone* e microcontroladores através do canal de áudio. Possibilidade de adição de sensores no robô.

Tabela 3: Tabela Comparativa de Componentes do Robô

A revisão realizada nesta seção evidencia que há diversos trabalhos semelhantes em relação à educação e à utilização de dispositivos móveis na robótica, entretanto, não há uma plataforma que permita a utilização de um robô e a publicação e compartilhamento de projetos simultaneamente.

[3]O custo depende da plataforma robótica que é utilizada. Para a aplicação do projeto, foi utilizado o robô do Programa de Educação Tutorial (PET) que custa em torno de R$500 se montado com peças compradas no Brasil. Exemplos de preços de robôs com microcontroladores acoplados podem ser encontrados em Shop (2014) na faixa de US$80

3 METODOLOGIA

Este capítulo apresenta a metodologia utilizada para o desenvolvimento do projeto proposto.

As duas próximas seções contêm as abordagens desenvolvidas, sendo que a primeira foi a ideia inicial e a segunda é uma solução que corrige os problemas gerados pela primeira abordagem. A seção 3.3 apresenta a comparação entre as soluções I e II. As seções 3.4 e 3.5 descrevem, respectivamente, as tecnologias utilizadas e os passos metodológicos.

3.1 FUNDAMENTOS DA ABORDAGEM I

Como mostra a figura 9, para esta abordagem, o projeto desenvolvido seria composto de um *smartphone*, um ambiente *Web* de programação e um decodificador DTMF. Existe também um componente externo ao projeto porém necessário para a execução completa deste, que seria a plataforma robótica comandada pelo *smartphone*.

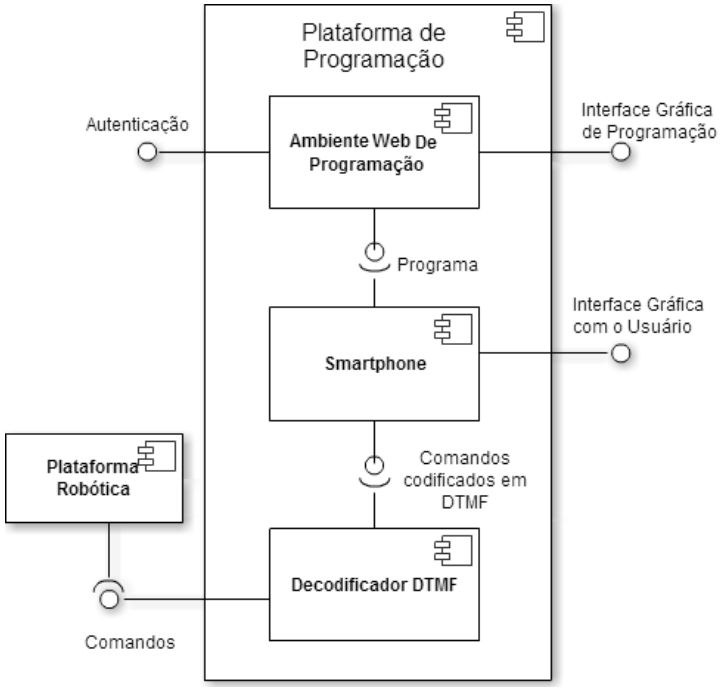

Figura 9: Diagrama em Blocos Geral do Sitema Proposto pela Abordagem I

Para a abordagem I, entende-se como base robótica os componentes físicos que são controlados pelo *smartphone*, o qual possui a parte lógica do robô. Neste escopo, a base é limitada a dois motores *Direct Current* (DC). O componente de decodificação de sinais DTMF é acoplado à plataforma robótica para que através deste, o *smartphone* possa enviar comandos para os motores.

Para esta solução, o *smartphone*, representado na figura 10, possuiria seu processamento particionado em cinco blocos, sendo eles: saída de áudio, codificador DTMF processamento principal, sensores e comunicação com o ambiente *Web*. O bloco de comunicação receberia um programa proveniente do ambiente *Web* e o carregaria no bloco de processamento principal, que por sua vez poderia receber dados dos sensores. O processamento principal ainda determinaria as ações do robô e as codificaria utilizando o bloco de codificação DTMF. Após isso, os comandos seriam enviados para a saída de áudio.

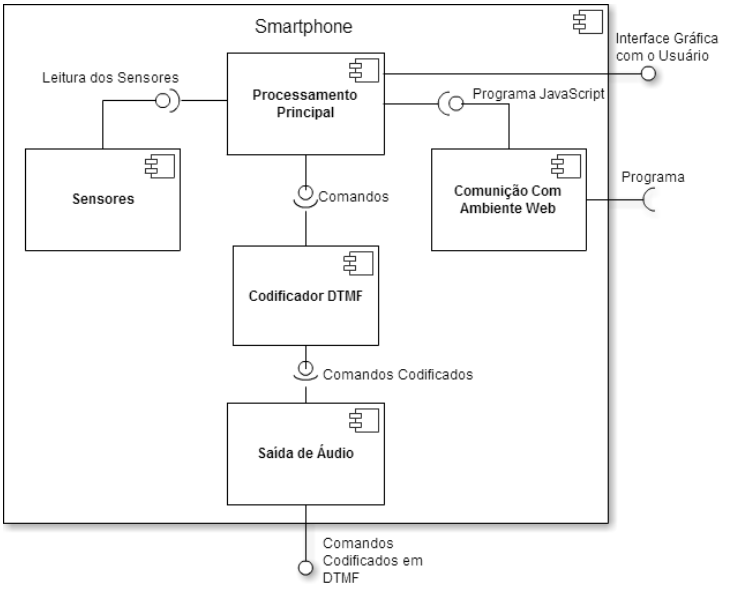

Figura 10: Diagrama de Componentes do Aplicativo *Smartphone* Proposto Pela Abordagem I

O ambiente *Web* de programação, ilustrado na figura 11, é constituído por três blocos e inclui a comunicação, um ambiente de programação visual baseado na biblioteca Blockly (BLOCKLY, 2013) e um gerenciador de contas. O gerenciador de contas, como o próprio nome sugere, gerencia as contas dos usuários, agregando os projetos já criados ao ambiente de programação visual, que por sua vez permite que o usuário programe apenas arrastando blocos. A biblioteca Blockly faz a conversão do programa gráfico para a linguagem JavaScript, e a comunicação, quando requisitada envia esta tradução para o celular.

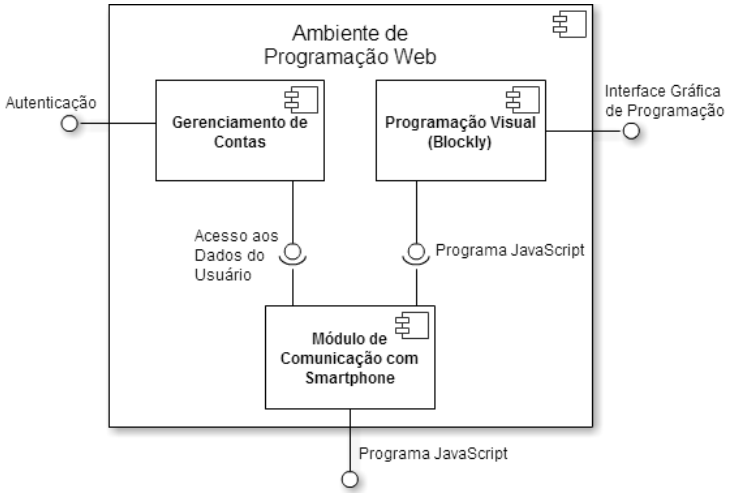

Figura 11: Diagrama de Componentes do Ambiente *Web* de Programação Proposto Pela Abordagem I

3.2 FUNDAMENTOS DA ABORDAGEM II

Como mostra a figura 12, as abordagens I e II diferem essencialmente na concepção da base robótica e na troca do decodificador DTMF para um adaptador áudio-serial.

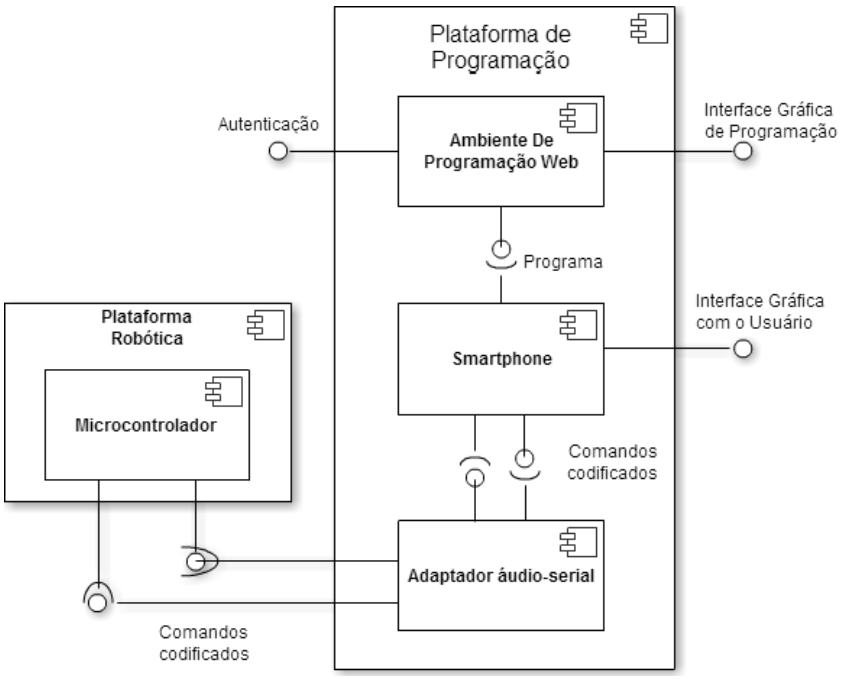

Figura 12: Diagrama em Blocos Geral do Sitema Proposto pela Abordagem II

A definição da base robótica foi modificada para a abordagem II. Esta que anteriormente não possuia capacidade de processamento sofreu a adição de um microcontrolador e, consequentemente, uma unidade de controle. Ou seja, a nova arquitetura mantém o *smartphone* como processador principal e adiciona o microcontrolador para executar as rotinas de mais baixo nível como gerar o sinal de controle dos motores DC, fazer o controle e ler os sensores da base robótica.

Esta solução mantém a partição de cinco blocos de processamento do *smartphone* representados na figura 13, entretanto o codificador DTMF é substituído por um codificador das mensagens trocadas entre o dispositivo móvel e a base robótica e o bloco de saída de áudio passa a ser de entrada também.

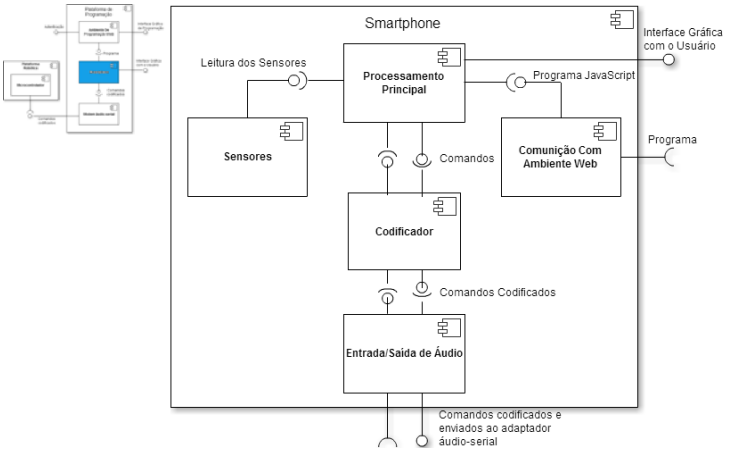

Figura 13: Diagrama de Componentes do Aplicativo *Smartphone* Proposto Pela Abordagem II

Para a abordagem II, o ambiente de programação *Web* continua constituído por três blocos, entretanto o que anteriormente era apenas gerenciamento de contas passou a ser também de projetos (vide figura 14). Esta alteração permitiu que os usuários compartilhassem projetos e os tornassem públicos.

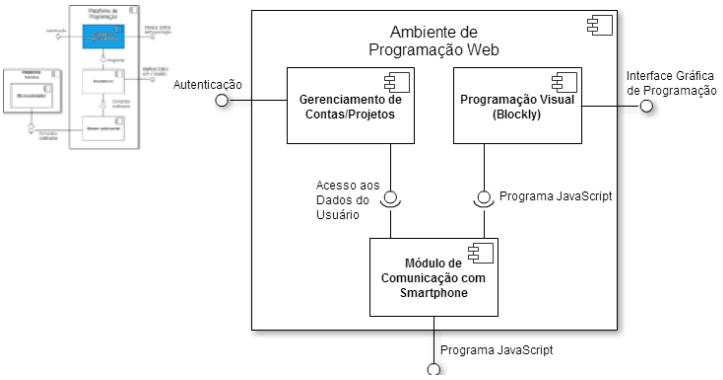

Figura 14: Diagrama de Componentes do Ambiente *Web* de Programação Proposto Pela Abordagem II

3.3 COMPARAÇÃO ENTRE AS ABORDAGENS I E II

A tabela 4 sumariza as principais diferenças entre as abordagens I e II.

	Abordagem I	**Abordagem II**
Smartphone	Única unidade de processamento.	Processador principal.
Aplicação *Web*	Gerenciamento de contas.	Gerenciamento de contas e projetos.
Plataforma Robótica	Sem microcontrolador, sem sensores, não gera PWM.	Com microcontrolador, com sensores.
Comunicação Entre o *Smartphone* e a Plataforma Robótica	Unidirecional. DTMF.	Bidirecional. Serial.
Controle dos Motores	PWM gerado pelo celular, irregular e de baixa frequência.	PWM gerado pelo microcontrolador.

Tabela 4: Comparação entre as abordagens I e II

Cabe ressaltar que a abordagem I foi desenvolvida até que suas limitações fossem encontradas e entendidas. Ou seja, ao perceber que a não utilização de uma unidade de processamento na base robótica implicaria em limitações na movimentação e inclusão de sensores, a equipe optou por alterar levemente o projeto e concluí-lo de maneira mais robusta.

Todas as limitações encontradas são detalhadas no capítulo 5.

3.4 TECNOLOGIAS

Esta seção apresenta as tecnologias utilizadas para o desenvolvimento do projeto.

3.4.1 APLICATIVO SMARTPHONE

O aplicativo *smartphone* foi desenvolvido em Cordova para que não houvesse uma grande limitação do sistema operacional do dispositivo móvel. Para tanto, o ambiente de desenvolvimento integrado Aptana Studio 3 (APTANA, 2014) foi utilizado.

O Cordova não possuia *plugins* voltados ao interfaceamento da comunicação por áudio necessários para as abordagens I e II, portanto eles tiveram que ser desenvolvidos. A equipe optou pelo desenvolvimento deles para apenas um sistema operacional (Android), entretanto esta escolha não invalidou o quesito multiplataforma do *framework* pois ainda

assim a expansão para outras plataformas é facilitada.

Dentre os sistemas operacionais de *smartphones* para o desenvolvimento dos *plugins*, foi selecionado o Android, pois este está presente na maioria dos celulares existentes, inclusive nos de baixo custo (BUCERZAN; RATIU; MANOLESCU, 2013). O sistema operacional Android é baseado em Linux (CHEN; CHEN; CHANG, 2011) e a linguagem de programação utilizada no desenvolvimento de aplicativos para este é Java, pois os kits de desenvolvimento padrão disponibilizados pela Google são direcionados para esta linguagem.

O *Android Development Tools* (ADT) já inclui o Android SDK entre outros pacotes necessários para acessar todas as funcionalidades, programar e depurar dispositivos com Android. Android SDK é uma biblioteca de alto nível para programação de aplicativos voltados ao sistema operacional Android. Este permite que o programador foque na aplicação e abstraia as diferentes particularidades de cada dispositivo móvel. Outra característica a destacar é a possibilidade de utilizar com facilidade funções de altíssimo nível que são muito úteis em projetos relacionados a robótica, como por exemplo reconhecimento de voz.

O segundo passo foi a escolha da versão do sistema operacional a ser utilizada como alvo para o aplicativo. As versões do sistemas mais atuais são compatíveis com programas desenvolvidos para as versões mais velhas, porém o contrário não acontece. Portanto, para abranger o maior número possível de celulares, foi necessário um balanço entre funcionalidades disponíveis e quantidade de celulares que utilizam a versão em questão. Após o estudo, a versão *Android Jelly Bean 4.1.x* foi selecionada. Com esta escolha são eliminados 40,9% dos celulares com Android (ver figura 15) (ANDROID, 2013), porém a equipe levou em conta a projeção futura deste cenário e chegou a conclusão de que é a API que mais se encaixa nos requisitos do projeto.

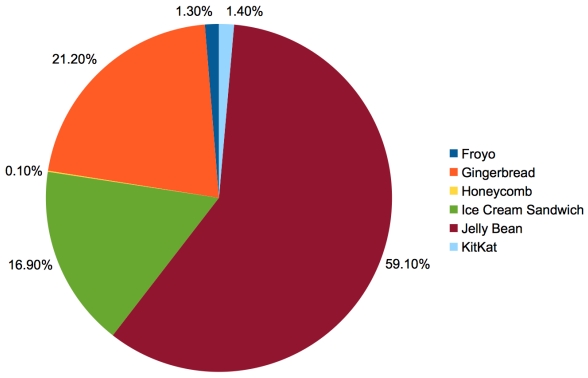

Figura 15: Proporção de Celulares Android e Suas Respectivas Versões
fonte: ANDROID (2014)

Outra decisão tomada é em relação ao ambiente de desenvolvimento integrado, ou *Integrated Development Environment* (IDE), que foi utilizado no desenvolvimento dos *plugins*. Os ambientes encontrados para desenvolvimento Android foram *Eclipse* e *Android Studio*. Após considerar os dois, a equipe selecionou o *Android Studio*, que apesar de estar na versão 0.3.7 se mostrou um ambiente estável, com atualizações frequentes e com suporte oficial do Google.

As principais características do Android Studio são:

- Suporte para *build* baseado em Gradle;
- Ferramentas de refatoramento específicas para Android;
- Ferramenta Lint para tratar performance, usabilidade, compatibilidade de versões entre outros;
- *Wizard* com *templates* para criação de componentes comuns para Android;
- Editor de *layout* que permite arrastar componentes de interface;
- Suporte para *Google Cloud Platform*.

3.4.2 AMBIENTE WEB DE PROGRAMAÇÃO

Para o desenvolvimento do ambiente *Web* foi utilizado o modelo de programação *Model View Controller* (MVC). Através deste modelo, a divisão do que é mostrado, *View*, de interação, *Controller*, e a parte lógica que possui os dados da aplicação, *Model*,

são separados e, logo, facilitam o gerenciamento, realização de testes e desenvolvimento paralelo do projeto.

No padrão de projeto de software MVC, o *Controller* recebe requisições do usuário, o *Model* aplica as regras de negócio e faz acesso ao banco de dados (caso seja necessário) utilizando os dados inseridos pelo usuário na requisição, e então um *View*, que pode corresponder à uma página HTML, é retornada ao usuário. O *framework* que implementa o MVC utilizado na aplicação *Web* foi o ASP.NET MVC 5 (MICROSOFT, 2013). Os principais motivos para esta escolha são:

- Plataforma livre e *open-source*;
- Suporte da IDE *Microsoft Visual Studio* que possui ferramentas avançadas para debug e organização do projeto;
- Suporte para gerenciamento de contas através da ferramenta *Role Management*;
- Suporte *Entity Framework*, capaz de acessar banco de dados através de uma abstração superior à *queries*, utilizando-se objetos e propriedades.

As linguagens de programação utilizadas no aplicativo *Web* foram: C# para o servidor, e JavaScript para o lado do cliente. Para a criação dinâmica dos conteúdos HTML do lado do servidor é utilizado o Razor, uma *view engine* lançada com a versão 3 do ASP.NET MVC.

Entre outras funções, o JavaScript é utilizado para manipulação da biblioteca Blockly. Esta biblioteca permite a criação de algoritmos em um ambiente de programação com blocos gráficos. Outra vantagem do Blockly é a possibilidade de gerar *scripts* a partir de um programa em blocos gráficos, tais *scripts* são gerados na linguagem JavaScript e posteriormente processados pelo *smartphone*.

O banco de dados utilizado no projeto é o *Microsoft SQL Server*. O principal motivo para a sua escolha foi a presença de suporte gráfico integrado no *Microsoft Visual Studio*. Para o desenvolvimento do *design* do *website* optou-se por utilizar o Twitter Bootstrap (TWITTER, 2014).

3.4.3 TROCA DE DADOS COM A BASE ROBÓTICA

A saída de som do *smartphone* para a troca de dados com a base robótica foi utilizada com o intuito de abranger grande parte dos dispositivos móveis. Sendo assim, foram utilizadas duas abordagens diferentes: DTMF e comunicação serial.

A abordagem do DTMF é de baixo custo pois, visto que esta tecnologia é amplamente utilizada, seus decodificadores são comuns. Além disso, para esta solução, a utilização de um microcontrolador foi descartada.

A abordagem da comunicação serial implicou na necessidade de utilizar um microcontrolador, entretanto, a solução se mostrou mais robusta e foi escolhida para a continuidade do projeto.

A comparação e descrição de ambas as soluções está descrita na seção 3.3.

3.4.4 AVALIAÇÃO PREDITIVA DA USABILIDADE

Para a maior compreensão e imparcialidade da análise da usabilidade do sistema foi utilizado um formulário baseado em Inspeção Heurística que basicamente é um conjunto de heurísticas utilizadas para avaliar um produto. Além disso, os avaliadores fazem essa análise individualmente (NIELSEN, 1993).

O formulário utilizado para avaliação de usabilidade possui heurísticas gerais descritas por Nielsen (1993). Essas heurísticas podem ser aplicadas em diferentes contextos, portanto, uma especificação delas para *websites* foi utilizada Rutter (2004). A lista contendo heurísticas gerais e suas especificações para o contexto *Web* foi a seguinte:

- Visibilidade do status do sistema:
  - Envolve o tempo de carga do *website*.
- Compatibilidade do sistema com o mundo real:
  - Envolve a coesão do *website* com o escopo do conteúdo que deve ser contido nele e o conhecimento do público-alvo.
- Consistência e padrões:
  - Localização e qualidade de *links* oferecidos pelo *website*. Além disso, consistência na escrita e gramática de todo *website*.
- Prevenção de erros:
  - Teste do *website* em diferentes sistemas operacionais e navegadores.
- Estética e design minimalista:
  - Envolve a qualidade do *background*, cor, gráficos, *layout*, fontes do *website*.

- Help e Documentação:
    - Envolve *metatags*, *copyright* e informações para contato.

Para a avaliação dessas heurísticas em um *website*, é possível utilizar os critérios estabelecidos por Rutter (2004) que descreve quatro classificações de completude para cada categoria: “Excelente”, “Bom”, “Satisfatório” e “Precisa de Melhorias”.

Caso a avaliação de completude não seja “Excelente”, é necessário descrever o motivo e atribuir a níveis de severidade para identificar quão prejudicial o problema é para o projeto e estabelecer prioridades para a sua correção. Os níveis de severidade para cada problema foram classificados por Nielsen (1993):

0. Não concordo que seja um problema de usabilidade;
1. Problema cosmético - corrigir se houver tempo extra;
2. Problema pequeno - baixa prioridade na correção;
3. Problema grave - alta prioridade na correção;
4. Problema catastrófico - correção obrigatória para entrega do produto.

Uma relação das heurísticas gerais e *Web* pode ser encontrada em Almeida e Santana (2008).

3.4.5 AVALIAÇÃO DA FERRAMENTA

Para que a análise da aplicação da ferramenta contemplasse também a motivação, o questionário aplicado foi desenvolvido com base no trabalho de McGill (2012).

O questionário aplicado é composto por duas partes, uma aplicada antes e outra após o curso. A primeira parte visa traçar o perfil dos alunos, coletar informações de seus dispositivos móveis e analisar alguns componentes da motivação ao introduzir a plataforma de ensino em sala de aula.

A segunda parte do questionário busca “medir” alguns componentes da motivação dos alunos após a aplicação da ferramenta de ensino. Para tanto, foram aplicadas questões na escala Likert (LIKERT, 1932) relacionadas à atenção, satisfação, confiança e relevância ao se tratar da plataforma de ensino. Os alunos também foram questionados em relação às suas expectativas futuras envolvendo cursos de programação e robótica.

3.5 PASSOS METODOLÓGICOS

Após definido o tema e feito o levantamento bibliográfico, o projeto teve sequência com a pesquisa das tecnologias empregadas para seu desenvolvimento, sendo que nesta etapa houve a decisão das mesmas. Feito isso, os passos metodológicos que foram seguidos até a conclusão do produto foram:

1. Utilização da saída do decodificador DTMF para o controle dos motores e testes de funcionamento:

   Estudo do hardware;

   Estudo do software;

   Desenvolvimento do hardware;

   Desenvolvimento do software;

   Integração hardware/software;

   Teste de funcionamento do bloco e verificação de limitações.

2. Utilização da saída de áudio para a comunicação serial com a base robótica:

   Estudo do hardware;

   Estudo do software;

   Desenvolvimento do hardware;

   Desenvolvimento do software;

   Integração hardware/software;

   Teste de funcionamento do bloco.

3. Desenvolvimento da interface *Web* de programação:

   Levantamento de requisitos da interface *Web* de programação;

   Estudo da biblioteca Blockly (BLOCKLY, 2013);

   Desenvolvimento do ambiente de programação visual;

   Implementação do tradutor de linguagem visual para JavaScript;

   Criação da página *Web*;

   Upload do aplicativo *Web* no servidor;

   Teste de funcionamento do bloco;

4. Desenvolvimento do aplicativo do dispositivo móvel:

    Levantamento de requisitos da aplicação Android;

    Estudo da API Android;

    Estudo do bloco de saída de áudio;

    Desenvolvimento da interface com o usuário;

    Desenvolvimento do software de controle do DTMF;

    Desenvolvimento da comunicação entre o dispositivo móvel e a base robótica;

    Desenvolvimento da comunicação com o servidor *Web*;

    Teste de execução de *scripts* em JavaScript;

    Interfaceamento dos sensores com a aplicação Android;

    Teste de funcionamento do bloco.

5. Desenvolvimento do banco de dados:

    Planejamento do banco de dados;

    Modelagem ER do banco de dados;

    Criação do banco de dados;

    Integração do banco de dados com a aplicação *Web*;

    Teste de funcionamento do bloco.

6. Avaliação Preditiva de Usabilidade

    Estudo das avaliações;

    Desenvolvimento do formulário;

    Aplicação da avaliação com os integrantes da equipe.

7. Estudo de Caso:

    Estudo das avaliações;

    Desenvolvimento de materiais (tutoriais, questionários, etc.);

    Aplicação das avaliações com alunos.

Após concluído o produto, ele foi aplicado ao ensino de programação de alunos de uma disciplina de computação e foi feito um estudo em cima de diversos fatores, como a resposta dos alunos a este tipo de incentivo e a usabilidade do produto para o ensino.

4 RECURSOS DE HARDWARE E SOFTWARE

Este capítulo apresenta todos os recursos de hardware e software essenciais ao desenvolvimento do projeto proposto.

4.1 RECURSOS DE HARDWARE

Os recursos de hardware necessários para o desenvolvimento do projeto proposto são:

1. *Smartphone* com sistema operacional Android[1];
2. Computador para o desenvolvimento de códigos;
3. Computador com sistema operacional Windows para desenvolvimento do aplicativo *Web*;
4. Plataforma robótica com:
   - Motores;
   - Bateria;
   - Ponte H;
   - Microcontrolador;
5. Carregador de bateria;
6. Cabo mini-USB/USB;
7. Adaptador serial-áudio;
8. *Pulse-Width Modulation* (PWM): técnica amplamente utilizada no controle da potência de motores DC (BRAGA, 2009).

[1]Para este projeto, os modelos de *smartphone* utilizados foram LG Nexus 4 e Samsung Galaxy S2 GT I9100

Entre os recursos necessários apresentados, a equipe possuia os itens 1, 2, 3, 5 e 6. O item 7 foi desenvolvido pela equipe no decorrer do projeto, e o item 4 pertence ao grupo PET (PETECO, 2014) e foi emprestado durante todo o período de desenvolvimento do projeto.

4.1.1 PLATAFORMA ROBÓTICA DO PET

A figura 16 mostra o robô do PET.

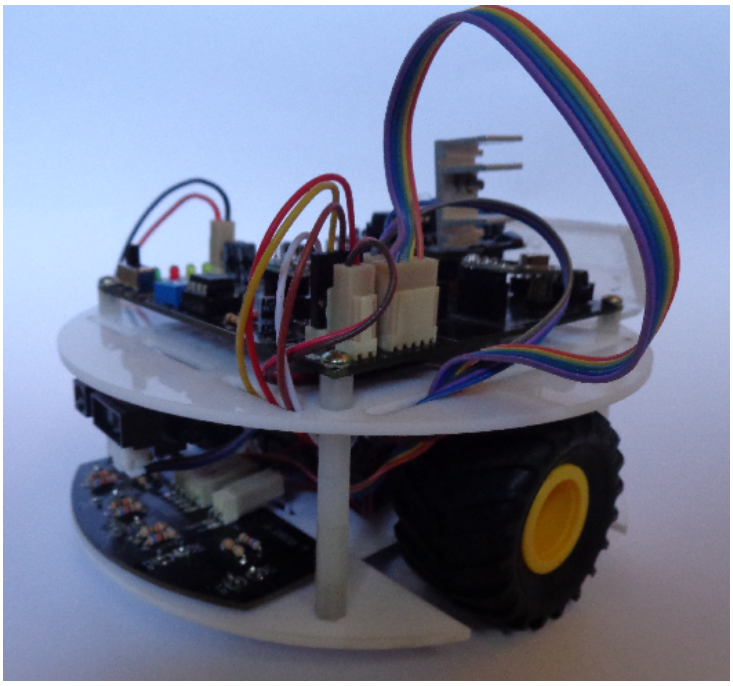

Figura 16: O Robô do PET

O robô possui os seguintes componentes:

- Base mecânica feita de acrílico (desevolvida pelo PET);
- Placa de Circuito Impresso (desevolvida pelo PET);
- Microcontrolador: ATMEGA328P-PU;
- Dois motores;
- Duas caixas de redução;
- Duas rodas;
- Ponte H;

- Bateria LiPo 7,2V para alimentação dos motores e da placa;
- Reguladores de tensão de 5V e 3V;
- Sensores:
  - 1 Infravermelho;
  - 5 Optointerruptores;
  - 2 Encoders.

4.2 RECURSOS DE SOFTWARE

Como recursos de software, são necessários:

1. Linguagens de utilizadas:
   - C/C++;
   - C#;
   - Java;
   - JavaScript;
   - HTML;
   - CSS;
2. Ambientes de desenvolvimento integrado:
   - Microsoft Visual Studio 2013;
   - Android SDK;
   - Aptana Studio 3;
   - Eclipse;
3. *Frameworks*:
   - Asp.net;
   - Android SDK;
   - Apache Cordova;
4. Ferramentas de Versionamento:

- GitHub;
- Team Foundation Sever;

5. Bibliotecas:
    - Bootstrap;
    - Blockly;
    - backbone.js;
    - jQuery;
    - topcoat.css;
    - underscore.js;
    - slimscroll.js;
    - prettify.js.
6. Técnicas:
    - *Asynchronous JavaScript XML* (AJAX): Técnica utilizada no lado cliente para gerar eventos assíncronos em aplicativos *Web* (W3SCHOOLS, 2014);
    - *JavaScript Object Notation* (JSON): Notação utilizada para transmissão de dados através de texto que é facilmente legível pelo ser-humano (JSON, 2014);
7. Arquiteturas aplicadas no *Web Service*:
    - *Representational State Transfer* (REST): Arquitetura que utiliza o protocolo HTTP, e pode realizar chamadas ao *Web Service* (FIELDING, 2000).

Não houveram gastos com recursos de software visto que todos são gratuitos ou podem ser acessados gratuitamente pela equipe.

5 A PLATAFORMA COFFEE

A plataforma proposta pelo projeto chama-se Coffee. O principal motivo para a escolha do nome foi a associação com a bebida que além de ser apreciada pelos integrantes da equipe, foi um estimulante para o desenvolvimento e conclusão desse projeto.

O sistema final permite que o usuário realize as seguintes ações:

- Crie sua própria conta;
- Possua uma página pessoal contendo sua lista de projetos;
- Edite, remova, crie, compartilhe e publique projetos;
- Baixe sua lista de projetos em seu *smartphone*;
- Acople seu *smartphone* na base robótica;
- Execute seus programas.

As funcionalidades da plataforma foram identificados baseando-se em sistemas como o Scratch (MALONEY et al., 2010) para o compartilhamento de projetos, Cally (YIM; SHAW, 2009), N-Bot (AROCA; GONÇALVES; OLIVEIRA, 2012), Romo (ROMO, 2014) e SmartBot (SMARTBOT, 2014) para o uso de um *smartphone* acoplado ao robô. Além disso, a publicação de projetos na página principal foi ideia da própria equipe.

Sendo assim, esta seção descreve o desenvolvimento de cada componente do sistema que torna seu funcionamento possível.

5.1 APLICATIVO *WEB*

Nesta seção, será tratada a parte do servidor *Web* que envolve o banco de dados, *back-end* e *front-end*. *Back-end* ou *server-side* corresponde à parte de processamento de dados requisitados ou inseridos pelo usuário e geralmente envolve o acesso ao banco de

dados para efetivar tais operações. *Front-end* ou *client-side* é a interface entre o usuário e o servidor, e abrange as chamadas ao servidor, e a visualização dos resultados obtidos.

5.1.1 BANCO DE DADOS

Conforme a figura 17, o banco de dados foi dividido em quatro tabelas, sendo elas: "ApplicationUser", "Project", "SharedProject" e "User_Proj". Estas correspondem, respectivamente, às informações do usuário do sistema, aos projetos, projetos compartilhados entre usuários e a relação N para N entre usuários e projetos.

As informações armazenadas em"ApplicationUser" correspondentes à conta de usuário, são: id, seu nome real, nome de usuário, e-mail, *password hash* e *security stamp*.

A tabela "Project" armazena informações do projeto, sendo elas: nome, descrição, data de criação, data em que o projeto foi salvo pela última vez, dados do Blockly no formato *Extensible Markup Language* (XML), código do Blockly em Javascript, e uma *flag* que informa se o projeto é público ou não.

A tabela "User_Proj" armazena os IDs de "ApplicationUser" e "Project". "SharedProject" armazena os IDs do usuário que compartilhou, do destinatário e do próprio projeto compartilhado. Além disso, a tabela "SharedProject" possui uma *flag* que informa o status de compartilhamento (aceito ou não aceito) do projeto.

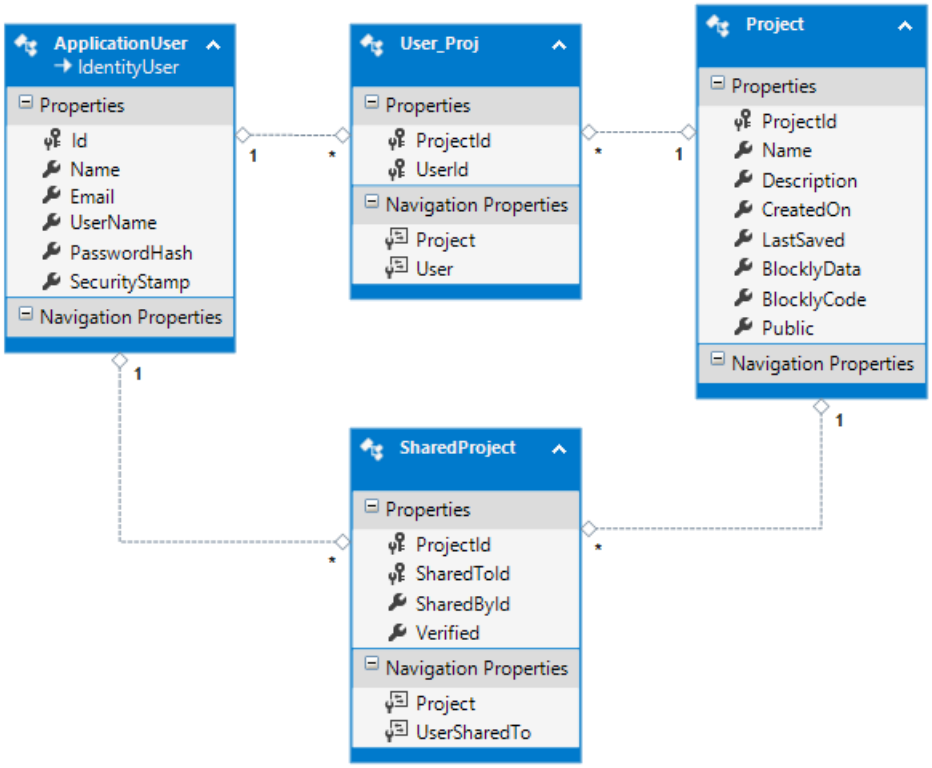

Figura 17: Modelo Relacional do Banco de Dados

5.1.2 *BACK-END* (LADO DO SERVIDOR)

O servidor *Web* responde a dois tipos de requisições de usuários. O primeiro tipo de acesso é através da página *Web*, nessa situação o usuário acessa o servidor a partir de um navegador. O segundo tipo de requisição é através de um *smartphone*, nesse modo as requisições ao aplicativo *Web* são feitas utilizando a arquitetura REST (FIELDING, 2000).

5.1.2.1 Tratamento de Requisições do Navegador *Web*

Esta seção descreve a arquitetura do servidor para requisições de acesso através de um navegador. As operações desse tipo de acesso envolvem gerenciamento de contas e de projetos do Blockly. Essas operações têm como resposta o retorno de páginas (*Views* no MVC) e a atualização ou inserção de dados do banco de dados.

Os *Controllers* envolvendo requisições a partir do navegador são de usuários e projetos e envolvem todas as operações necessárias para o gerenciamento de conta, gerenciamento de projetos e o ambiente de programação em blocos. Portanto, operações *Create, Remove, Update, Delete* (CRUD) foram implementadas para manipular projetos e contas

de usuário. Outra funcionalidade desenvolvida foi o compartilhamento de projetos entre usuários e a comunidade (através de projetos públicos disponíveis à todos que acessam o servidor).

As responsabilidades do controlador de contas são:

- Autenticar *login* e senha;
- Retornar página de registro de uma nova conta;
- Registrar nova conta;
- Retornar página com informações da conta de usuário;
- Retornar página para alteração de senha;
- Alterar senha de conta;
- Retornar página para alteração de e-mail;
- Alterar e-mail de conta.

As principais responsabilidades do controlador de projetos são:

- Retornar página com a lista de projetos associados à conta de usuário;
- Criar novo projeto;
- Deletar projeto;
- Retornar página com informações da configuração de um projeto;
- Salvar alterações da configuração de um projeto;
- Carregar blocos e código de determinado projeto;
- Salvar blocos do projeto;
- Retornar página com projetos públicos e busca de projetos públicos;
- Retornar página como resultado de uma busca de projetos públicos;
- Clonar projeto público para o repositório da conta;
- Retornar página com uma lista de novos projetos que foram compartilhados por outros usuários;

- Compartilhar projeto.

Um terceiro controlador chamado *Home* é responsável pela apresentação inicial na tela do usuário, e suas responsabilidades são:

- Retornar página principal e inicial;
- Retornar página com informações sobre o projeto Coffee;
- Retornar página com informações para contato.

Algumas páginas necessitam que o usuário esteja logado para seu acesso, portanto, foram utilizados os atributos de autorização do ASP.NET para essa verificação. Basicamente esses atributos funcionam da seguinte forma: as páginas que requerem determinada autenticação para serem executadas passam por um filtro que verifica se o usuário está autenticado no sistema no momento da requisição. Caso o usuário não esteja autenticado, uma página de erro é retornada.

Além disso, utilizou-se esta mesma técnica para verificar o acesso de projetos de usuários. Desta maneira, somente projetos de usuários contidos na tabela "User_Proj" poderiam ser acessados.

5.1.2.2 Tratamento de Requisições do Aplicativo *Smartphone* - *Web* API

As requisições utilizadas pelo *smartphone* envolvem: autorização para *login*, carregar lista de projetos e carregar código de projeto. Todas essas requisições utilizam somente as operações GET do protocolo HTTP, e além disso, retornam um objeto JSON (JSON, 2014) com as informações requisitadas pelo usuário.

As responsabilidades do controlador do *smartphone* são:

- Autenticar *login* e senha - retorna uma *flag*;
- Carregar código de projeto - retorna o código em JavaScript;
- Carregar lista de projetos - retorna uma lista com informações dos projetos do usuário: ID, nome, última data em que o projeto foi salvo, e data de criação e descrição.

Foi utilizado um atributo de autorização do ASP.NET para verificar a autenticação de usuários ao fazerem acesso às duas últimas requisições mencionadas acima.

5.1.3 *FRONT-END* (LADO DO CLIENTE)

Nesta seção são abordadas as páginas *Web* (*Views*) retornadas ao usuário, assim como a parte do projeto que utiliza a biblioteca Blockly.

5.1.3.1 Página *Web*

A página *Web* foi criada com o intuito de ser atrativa para os alunos. Esta solução funciona melhor em *desktops* e *notebooks* e não requer instalações para ser utilizada, bastando apenas um navegador e acesso à Internet.

As ações proporcionadas pelas páginas são:

- Possuir um perfil de usuário associando cada projeto criado ao seu criador;
- Criar, editar, excluir, salvar e configurar projetos;
- Compartilhar projetos com outros usuários;
- Tornar públicos os projetos do usuário;
- Pesquisar projetos públicos;
- Copiar projetos públicos à lista de projetos do usuário;
- Editar configurações de projeto;
- Editar configurações de conta.

Os dois primeiros tópicos da lista permitem que projetos criados por um usuário fiquem salvos e associados à sua conta, permitindo a sua posterior edição, exclusão ou configuração.

Além das ações mencionadas, há o compartilhamento que permite o desenvolvimento colaborativo entre usuários. Por outro lado, a publicação de projetos permite a visualização e cópia dos mesmos por outros usuários.

A figura 18 representa a página inicial da Plataforma Coffee. Em seu topo existe um menu de acesso rápido presente em todas as páginas do aplicativo. Em sua parte inferior, há uma lista dos últimos quatro projetos públicos disponibilizados por usuários.

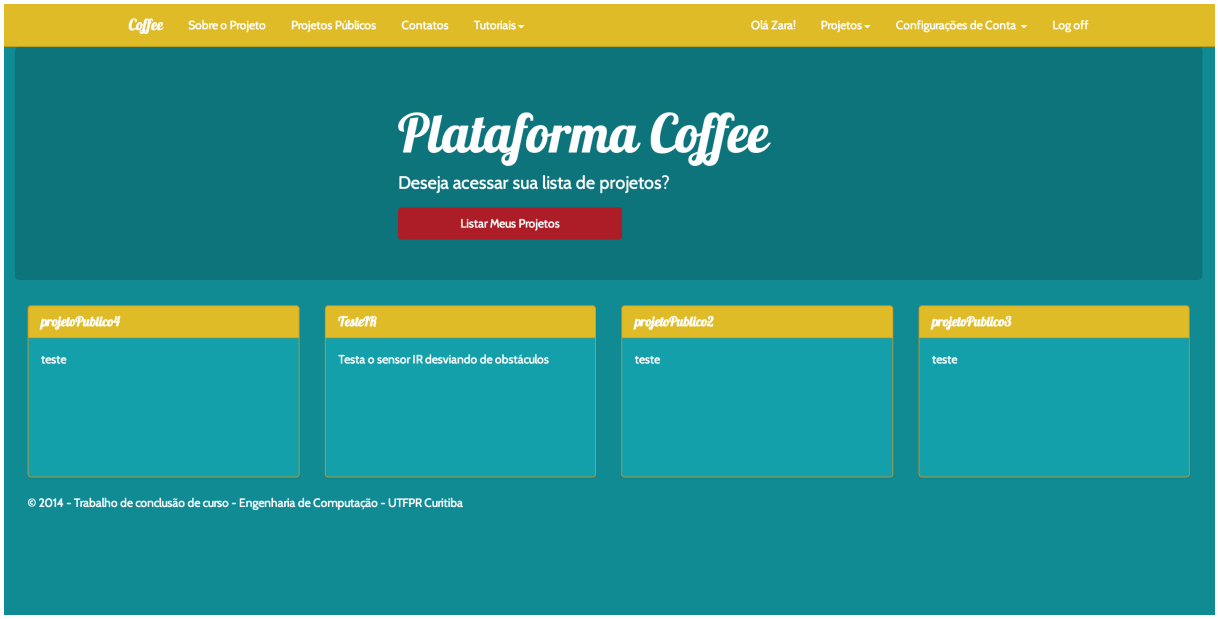

Figura 18: Página *Web* - Página Inicial

Caso o usuário clique no botão "Listar Meus Projetos" que aparece na figura 18, ele será direcionado a sua lista de projetos mostrada na figura 19. Além disso, é possível acessar a criação, configuração e compartilhamento de projetos nessa mesma página.

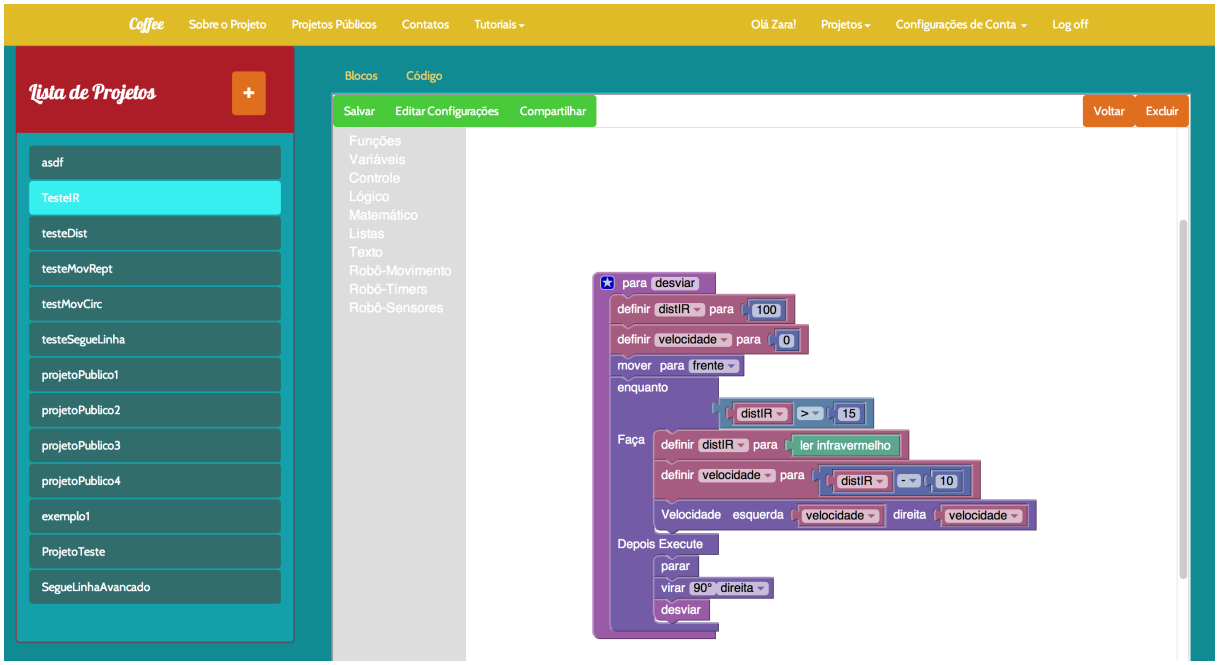

Figura 19: Página *Web* - Lista de Projetos

A figura 20 mostra a busca por projetos públicos que pode ser acessada sem que o usuário esteja conectado à sua conta. Esta busca pode ser realizada por nome de usuário

ou de projeto e ordenada por data de criação ou publicação do mesmo. Além disso, o sistema permite a clonagem do projeto para a sua lista caso o usuário esteja conectado.

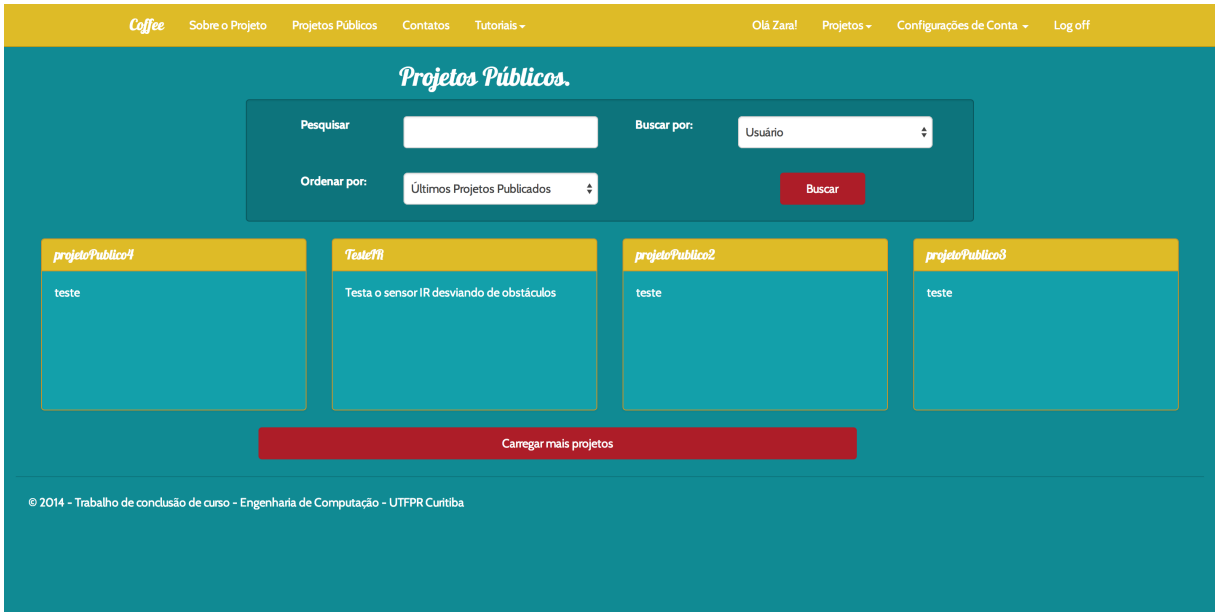

Figura 20: Página *Web* - Projetos Públicos

Todos os projetos desenvolvidos na página *Web* são escritos exclusivamente a partir de uma linguagem de blocos como mostra a figura 21, além disso, como mostrado na figura 22 é possível verificar o programa gerado pelos blocos em JavaScript.

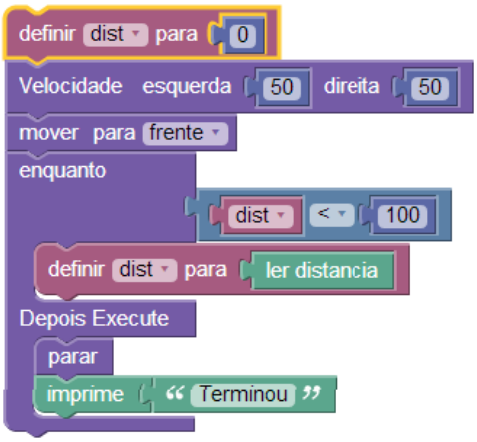

Figura 21: Página *Web* - Linguagem de Blocos

```
dist = 0;
app.Robot.set_speed(50,50);
app.Robot.assign_action(mov,['frente']);
app.Robot.assign_loop('while','dist < 100',function(){
  dist = (app.Robot.ler_distancia());
},function(){
  app.Robot.assign_action(parar,[]);
  window.alert('Terminou');
});
```

Figura 22: Página *Web* - Tradução dos Blocos Para JavaScript

Além da linguagem Razor, que oferece um complemento através da programação em C# e *"templating"* nas páginas, outras tecnologias foram utilizadas para a implementação da página *Web*.

O AJAX foi utilizado para atender às requisições assíncronas geradas pelo usuário em que não deve ocorrer a atualização da página inteira. Um exemplo disso é quando o usuário salva um projeto. Neste caso é realizada uma requisição POST ao servidor utilizando AJAX e a página *Web* não sofre alterações em seu estado.

Além do Bootstrap, outras bibliotecas foram utilizadas para que a visualização da página se tornasse mais agradável. Para uma impressão formatada do código em JavaScript gerado a partir dos blocos de um projeto e para a rolagem de alguns elementos HTML, foram utilizadas, respectivamente, as biblioteca prettify.js (PRETTIFY, 2014) e slimScroll.js (ROCHALA, 2014).

5.1.3.2 Blockly

A linguagem de programação em blocos foi escolhida por ser utilizada em diversos projetos de programação como o Scratch (MIT, 2013a). Além disso, a linguagem é de fácil utilização visto que não requer conhecimentos prévios de programação, e as formas são específicas de maneira que apenas blocos compatíveis se encaixem.

O Blockly (BLOCKLY, 2013) proporciona a criação e customização de blocos, o que permitiu que blocos específicos para movimentação, leitura de sensores, temporização e controle do robô fossem criados.

A criação dos blocos pode ser feita pelo *Block Factory* (BLOCKLY, 2014). O resultado desta criação é um código em JavaScript que permite a customização. Segue um exemplo de bloco customizado:

```
1 Blockly.Blocks['ler_acelerometro'] = {
2     init: function () \{
```

```
3           this.setHelpUrl('');
4           this.setColour(160);
5           this.appendDummyInput()
6               .appendField("ler acelerometro no eixo");
7           this.appendDummyInput()
8               .setAlign(Blockly.ALIGN_CENTRE)
9               .appendField(new Blockly.FieldDropdown([["x", "x"], ["y", "y
                    "], ["z", "z"]]), "eixo");
10          this.setInputsInline(true);
11          this.setOutput(true, "Number");
12          this.setTooltip('Esta funcao retorna um valor');
13      }
14  };
15
16  Blockly.JavaScript['ler_acelerometro'] = function (block) {
17
18      var eixo = block.getFieldValue('eixo');
19      var code = 'app.Robot.ler_acelerometro("'+eixo+'")';
20      return [code, Blockly.JavaScript.ORDER_NONE];
21
22  };
```

O *Block Factory* permite que diversas características do bloco sejam escolhidas, como: cor, formato, encaixes, entre outros. Cabe ressaltar que o código gerado pelo *"generator stub"* deve ser editado manualmente para prover o resultado esperado pelo bloco.

Todos os blocos criados para a plataforma de ensino Coffee estão presentes na imagem 23.

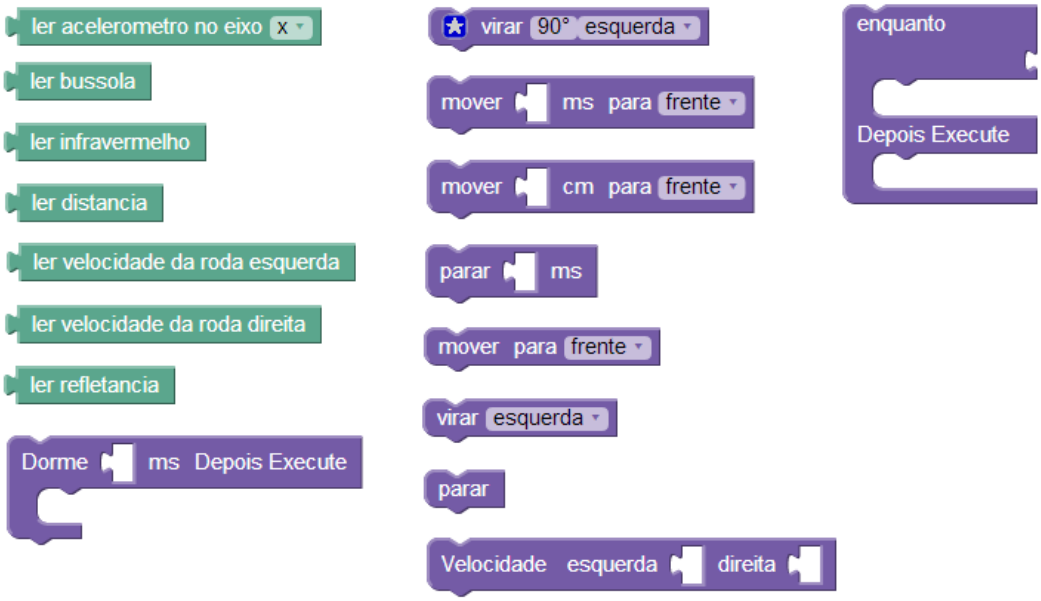

Figura 23: Blocos Criados

O bloco "enquanto", presente na figura 23, é o bloco de controle que teve que ser criado para substituir o bloco de repetição *"while"* pré-existente no Blockly. Esta substituição teve que ser realizada devido ao JavaScript não ser *multi-thread*, resultando no travamento do programa ao utilizar o bloco em questão (maiores detalhes são explicados na seção 5.2.3).

Embora seja uma solução para a limitação do JavaScript, o "enquanto", dividido nas partes "enquanto" e "depois execute", possui algumas limitações (ver seção 5.2.3), sendo elas:

- Não é possível inserir nenhum bloco de movimentação que receba dois parâmetros dentro da parte "enquanto", exceto o que define a velocidade das rodas esquerda e direita;
- É necessário declarar uma variável antes do bloco "enquanto" caso ela seja utilizada como condição do mesmo;
- Só é possível utilizar blocos "enquanto" encadeados caso estejam dentro da parte "depois execute", ou seja, esses blocos não podem estar aninhados, e é recomendável que esta seja a única maneira de utilizar mais de um bloco deste tipo no programa.

Cabe ressaltar que há códigos lógicamente corretos que se encaixam nestas restrições, entretanto não podem ser utilizados para esta aplicação.

Para uma melhor visualização da aplicação dos blocos, são apresentados quatro exemplos no decorrer do texto, sendo eles, testes utilizados para: distância, movimento, refletância e velocidade.

O teste de distância, imagem 24, faz com que o robô continue andando para frente até que tenha percorrido 100 centímetros.

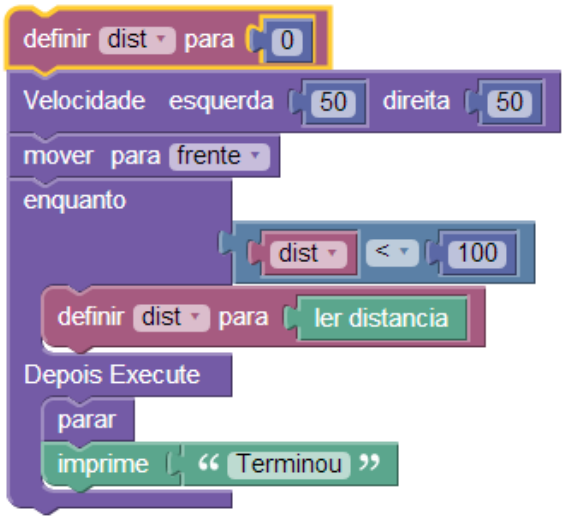

Figura 24: Teste de Distância

O teste de movimento, imagem 25, faz com que o robô se movimente desenhando um quadrado.

Figura 25: Teste de Movimento

O teste de refletância, imagem 26, faz com que o robô siga uma linha. Note que a leitura destes sensores retorna uma lista enumerada de 1 a 5, em que cada posição retorna o valor de um dos cinco sensores de refletância presentes no robô.

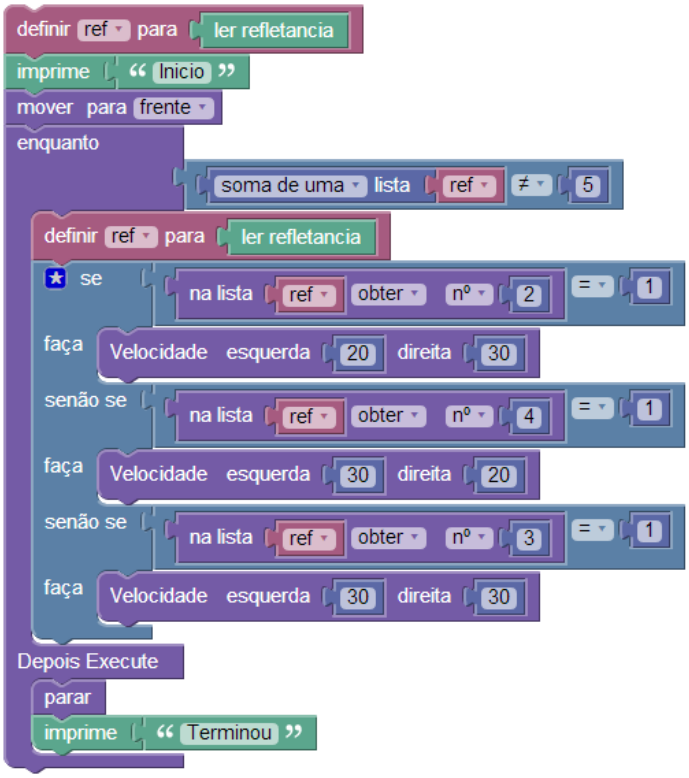

Figura 26: Teste de Refletância

O teste de velocidade, imagem 27, faz com que o robô aumente a velocidade de suas rodas gradativamente até atingir um determinado valor.

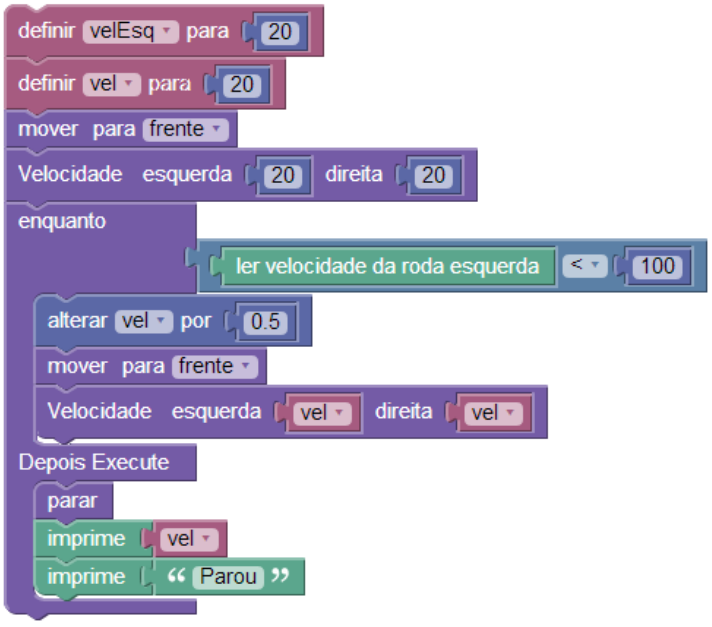

Figura 27: Teste de Velocidade

5.2 APLICATIVO SMARTPHONE

O aplicativo *smartphone* foi desenvolvido com o intuito de permitir que o usuário faça o *download* de sua lista de projetos criada no aplicativo *Web* e os execute junto à plataforma robótica. Ele possui três telas, sendo elas a de *login*, listagem de projetos e tela de projeto, todas ilustradas na figura 28, respectivamente.

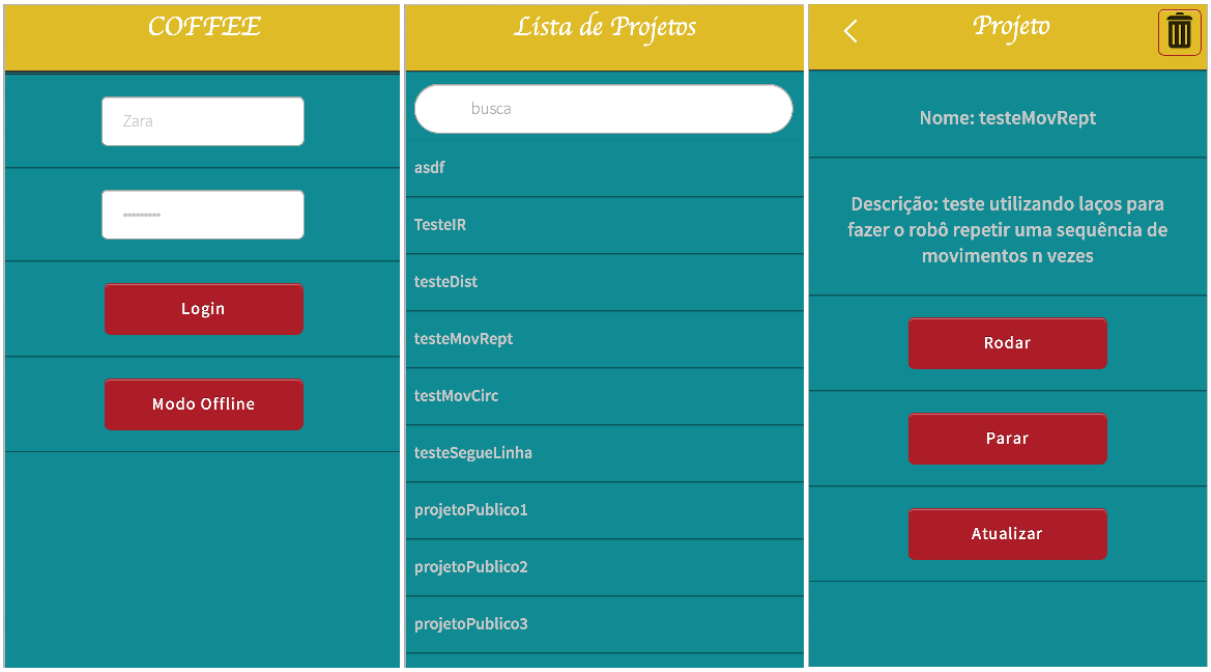

Figura 28: Telas do Aplicativo *Smartphone*

A execução dos projetos proporcionada pelo aplicativo *smartphone*, desenvolvido utilizando Cordova, requer o desenvolvimento de um protocolo de comunicação entre a base robótica e o dispositivo móvel.

Os detalhes de implementação contendo a estrutura do aplicativo, o protocolo de comunicação com a base robótica, a organização da lista de tarefas do robô e o desenvolvimento de *plugins* específicos estão descritos nas subseções seguintes.

5.2.1 ESTRUTURA DO APLICATIVO *SMARTPHONE*

O código do aplicativo *smartphone* foi estruturado utilizando a biblioteca backbone.js (BACKBONE, 2014), a qual permite a utilização de padrões de projeto da família *Model View Family* (MV*), como por exemplo o MVC utilizado no aplicativo *Web*. A *backbone* proporciona a persistência dos dados do lado do cliente e não apenas do ser-

vidor, diminuindo assim a quantidade de requisições necessárias durante a execução do aplicativo *smartphone*.

A persistência proporcionada pela biblioteca *backbone* permite a utilização do conceito de *Single Page Application*[1] (SPA), o que torna desnecessária a atualização de páginas. Sendo assim, o conteúdo do aplicativo pode ser alterado dinamicamente com a utilização do JQuery para monipular o *Document Object Model* (DOM) da página.

Sempre que o aplicativo necessita de algum dado do servidor, são utilizadas requisições AJAX, e trazidos objetos JSON através da arquitetura REST. Devido a isso, a quantidade de dados trocados entre o servidor e o aplicativo é reduzida, proporcionando assim ao usuário respostas mais rápidas e semelhantes às que seriam obtidas com a utilização de um aplicativo nativo.

O *layout* do aplicativo *smartphone* foi estruturado utilizando as bibliotecas *topcoat* e *underscore*, sendo a primeira um conjunto de recursos CSS e a outra uma ferramenta para realizar *"templating"* em JavaScript.

O *"templating"* consiste em renderizar os conteúdos HTML a partir de um modelo, substituindo marcações por dados dinâmicos. O código abaixo segue como exemplo.

```
1 <div>
2     <a href="\#project/<\%=id\%>">
3     <p><\%=Name\%></p>
4   </a>
5 </div>
```

Neste caso, a marcação é tudo que está entre os caracteres "<%" e "%>", fazendo com que as variáveis "Name" e "id" sejam substituídas em tempo de execução pelo valor delas no projeto em questão.

5.2.2 BANCO DE DADOS

O armazenamento dos dados dos usuários e projetos é realizado através do WebSQL, um banco de dados que é excutado no próprio navegador. O model relacional (MR) deste está representado na figura 29.

[1] É uma aplicação web que durante toda a sua execução não atualiza a página, matendo assim a persistência dos dados sem a necessidade de comunicação com o servidor.

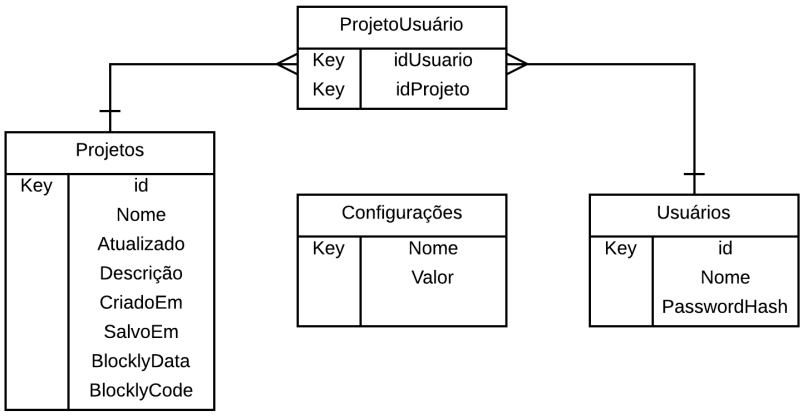

Figura 29: Modelo Relacional do Banco de Dados do *Smartphone*

Conforme representado na figura 29, o banco de dados possui quatro tabelas, sendo elas "ProjetoUsuario", "Configuracoes", "Projetos" e "Usuarios".

A tabela "ProjetoUsuario" representa a relação N para N de usuários com projetos. Isso se deve à função de compartilhamento e resulta na possibilidade de um projeto estar relacionado a N usuários.

A tabela "Configuracoes" representa quaisquer configurações do aplicativo que deva ser persistente entre execuções do aplicativo, por exemplo o último usuário logado. As relações "Usuarios" e "Projetos" armazenam respectivamente dados de projetos e usuários, sendo que o código do projeto que será executado está no campo "BlocklyCode".

A biblioteca *backbone* utiliza os dados do banco de dados e apenas faz requisições ao Aplicativo *Web* quando os dados não são encontrados ou estão desatualizados.

5.2.3 AGENDAMENTO DE TAREFAS

Como o aplicativo *smartphone* foi desenvolvido, grande parte, em JavaScript. A utilização de *threads* foi impossibilitada devido ao fato desta linguagem ser baseada em eventos. Portanto, um agendamento de tarefas tornou-se necessário.

Este agendamento consiste em manter uma lista de quais serão os próximos comandos enviados à base robótica. Para que houvesse um controle de início e término das ações, a utilização do *Deferred* do JQuery foi necessária. Este objeto mantém uma função de *callback* que é chamada após a finalização da tarefa em questão.

A função de *callback* é a mesma para todas as tarefas que são inseridas na lista.

Ela faz o controle de quais tarefas devem ser executadas e como devem ser os seus respectivos tratamentos.

Existem dois tipos de tarefas que podem ser inseridas na lista, sendo elas condicionais ou não condicionais.

As tarefas condicionais são:

- Virar "X" graus para a esquerda ou direita;
- Mover "X" centímetros para frente ou para trás;
- Mover durante "X" milisegundos para frente ou para trás;
- Parar durante "X" milisegundos;
- Laço condicional.

As tarefas não condicionais são:

- Virar para a esquerda ou direita;
- Mover para frente ou para trás;
- Parar.

Para o primeiro grupo, as tarefas sempre são inseridas no fim da lista de execução. Já para o outro grupo, as tarefas serão inseridas no fim da lista caso sejam diferentes da última ação não condicional listada.

Cabe ressaltar que a tarefa de atribuição de velocidade pode ser executada assim que chamada ou inserida no fim da lista de execução, dependendo da ação atual. O primeiro caso ocorre quando a ação atual é não condicional, já o segundo, quando é condicional. Este mecanismo permite que a mudança de velocidade possa ser realizada tanto em laços quanto em sequências de movimentos.

Sempre que a tarefa de atribuição de velocidade é chamada, uma variável é atualizada com o respectivo valor e nenhum comando é enviado à base robótica. Para que o robô receba a nova velocidade atribuída, um temporizador que envia estes dados é utilizado periodicamente. Os dados são enviados somente quando a variável apresenta uma mudança de valor.

Outro detalhe importante é que a estrutura de repetição “enquanto” nativa do JavaScript não pôde ser utilizada uma vez que a variável presente na condição da estrutura normalmente é atualizada por outro evento do código. Como não há *threads* nesta linguagem, o programa nunca sairia do laço e seria completamente travado.

Sendo assim, optou-se por criar a seguinte função em JavaScript:

```
1     function enquanto( funcao, condicao, funcaoDepois);
```

A função acima é chamada por um temporizador que verifica a condição periodicamente. Caso ela seja verdadeira, a sequência de comandos contida no parâmetro “funcao” é executada. Caso seja falsa, o temporizador é desligado, o laço é finalizado, a “funcaoDepois” é executada e a lista de execução é retomada. Cabe ressaltar que ambos parâmetros “funcao” e “funcaoDepois” são definidos pelo usuário. Um exemplo de utilização desta função é representado abaixo:

```
1 enquanto(
2     function(){
3         x = x +1
4     },
5     "x < 10",
6     function(){
7          alert("terminou"};
8 });
```

Para a chamada periódica da função apresentada, a função window.setInterval(), nativa do JavaScript, foi utilizada.

A leitura dos sensores do *smartphone* e da base robótica é executada periodicamente através de um temporizador e é disponibilizada em variáveis que podem ser acessadas pelo código em execução. Esta ação faz com que seja possível acessar os dados dos sensores sem que haja qualquer tipo de espera.

5.2.4 *PLUGINS* CORDOVA

Não foi possível encontrar na biblioteca padrão do Cordova ou na Internet *plugins* que satisfizessem as necessidades para implementar a comunicação entre a base robótica e o *smartphone* para as abordagens I e II deste projeto. Sendo assim, foi necessário desenvolver os *plugins* de DTMF e áudio-serial.

Um *plugin* do Cordova consiste em duas partes, sendo elas o código nativo, Android neste caso, e uma interface JavaScript. A opção pelo sistema operacional Android

não limita o a expansão do sistema para outras plataformas, apenas requer o desenvolvimento do código nativo para estas.

O *plugin* de DTMF está presente no repositório GitHub (ZARAMELLA; REIS; SARMENTO, 2014b) e possui a seguinte função:

- DTMF.startTone(succesCallBack, errorCallBack, int tone, int duration)

Esta função executa um tom DTMF por uma duração "duration" em milisegundos.

O *plugin* da comunicação áudio-serial está disponível no repositório GitHub (ZARAMELLA; REIS; SARMENTO, 2014a) e possui as seguintes funções:

- SerialAudio.sendBytes(succesCallBack, errorCallBack, String strToSend);
- SerialAudio.receiveByte(succesCallBack, errorCallBack);
- SerialAudio.startReading(succesCallBack, errorCallBack);
- SerialAudio.stopReading(succesCallBack, errorCallBack).

5.2.5 PROTOCOLO DE COMUNICAÇÃO ENTRE O *SMARTPHONE* E A BASE ROBÓTICA

Para que o *smartphone* e a base robótica se comunicassem, um protocolo e uma codificação de comandos foram criados. O protocolo exemplificado na figura 30 consiste apenas de um *start byte*, "K", e um *stop byte*, "Z" e encapsula os dados a serem trocados. Já a codificação está representada na tabela 5.

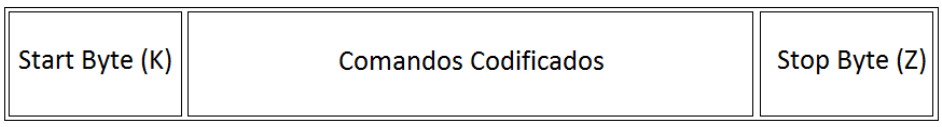

Figura 30: Pacote de Dados

Comando	Direção	Descrição
T(D/E)X[2]	*Smartphone* para robô	Faz com que o robô vire "X" graus para a "E"squerda ou "D"ireita, sendo "X" um inteiro positivo entre 0 e 360.
M(F/T)X[2]	*Smartphone* para robô	Faz com que o robô se mova para "F"rente ou para "T"rás por "X" ms.
m(F/T)X	*Smartphone* para robô	Faz com que o robô se mova para "F"rente ou para "T"rás por "X" cm.
PX[2]	*Smartphone* para robô	Faz com que o robô pare por "X" ms.
VXXYY	*Smartphone* para robô	Determina a velocidade das rodas esquerda ("XX") e direita ("YY"), sendo "XX" e "YY" valores inteiros entre 00 e 50 mm/s.
IX[3]	Robô para *smartphone*	É a distância "X" lida pelo sensor infravermelho em cm.
PXX[3]	Robô para *smartphone*	É a distância "XX" lida pelo encoder em cm percorrida pelo robô.
VXY[3]	Robô para *smartphone*	É a velocidade das rodas esquerda ("X") e direita ("Y") lida pelo *encoder*.
RX[4]	Robô para *smartphone*	É a leitura dos sensores de refletância.
F	Robô para *smartphone*	É a notificação de que o robô finalizou o último comando dado.

Tabela 5: Tabela de Codificação de Comandos

5.3 ABORDAGEM I: DTMF COMO CONTROLADOR DE MOTORES DC

Para a abordagem I, o controle dos motores DC é realizado diretamente pela saída de áudio do *smartphone*, ou seja, o sinal de controle dos motores é gerado pelo dispositivo móvel. Esta abordagem utiliza o DTMF de maneira que o *smartphone* emite o sinal e cada motor recebe dois bits de controle. Para tanto, um circuito decodificador de DTMF,

[2] O "X" é opcional para estes comandos. Neste caso, eles são executados até que outro comando seja dado.

[3] Cada "X" é representado por um byte e deve ser lido como número e não caracter. A presença dele é obrigatória no comando. Caso o comando tenha mais de um byte, o MSB está mais à esquerda.

[4] Para este comando são utilizados apenas os cinco bits menos significativos do byte e cada um representa a leitura de um dos sensores de refletância do robô.

mostrado na figura 31, foi desenvolvido.

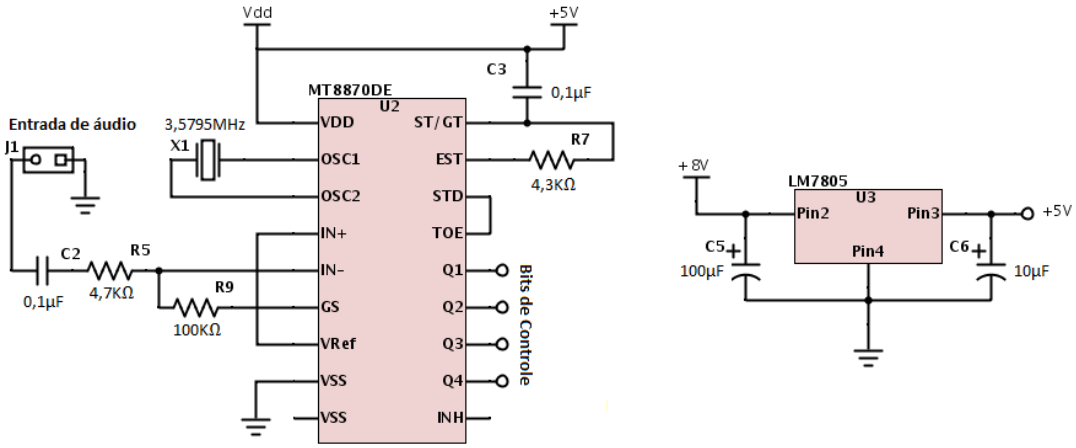

Figura 31: Diagrama Esquemático do Circuito Decodificador DTMF
fonte: baseado em Aroca (2012).

O circuito decodificador da figura 31 foi baseado no mesmo utilizado por Aroca (2012).

Apesar da Abordagem I ser funcional, a qualidade do sistema resultante observada durante o seu desenvolvimento foi baixa. O principal problema foi gerar um PWM adequado. A frequência alcançada desse sinal foi de 10Hz, e com este PWM os motores apresentaram trepidações. Os testes foram realizados utilizando-se um *smartphone* com o sistema operacional Android.

Foram caracterizados dois motivos para esse problema. O primeiro é decorrente do funcionamento do sistema operacional Android que não possui temporizadores precisos que, inclusive, não atingem frequências muito superiores às geradas. O segundo motivo, que poderia acabar diminuindo ainda mais essa faixa de frequência, é que o sistema operacional pode estar atarefado com outros aplicativos e destinar pouca capacidade de processamento ao aplicativo deste projeto. Portanto, nessa situação, poderia causar uma certa instabilidade no sinal enviado por DTMF.

Um segundo problema, foi a quantidade de sensores que a plataforma poderia utilizar. Com essa abordagem, havia uma limitação em relação à quantidade desses dispositivos utilizados pelo robô, e somente os sensores do *smartphone* poderiam ser utilizados. Isso ocorreria pois a comunicação desta solução era unidirecional. A utilização de *encoders*, infravermelho e dos sensores de refletância acoplados à plataforma robótica permitiriam a criação de problemas mais interessantes.

Durante a primeira abordagem, também tornou-se interessante a adição da funcionalidade de compartilhamento de projetos entre alunos e com o público, visto que isso ajudaria os alunos a divulgar sua soluções de problemas com seus colegas. A Abordagem II solucionou todas essas três restrições envolvendo a limitação de geração do PWM, sensoriamento externo ao *smartphone* e a possibilidade de compartilhamento de projetos.

5.4 ABORDAGEM II: ADAPTADOR ÁUDIO-SERIAL

Para a abordagem II, o controle dos motores DC é realizado por um microcontrolador dedicado. Apesar disso, o *smartphone* é responsável por enviar os comandos que determinam o funcionamento do microcontrolador, e, consequentemente, do robô.

A comunicação entre o *smartphone* e o microcontrolador é realizada através do canal de áudio que envia dados serialmente e sem modulação. Para que esta comunicação seja possível, um adaptador é utilizado, tornando compatíveis os níveis de entrada e saída do áudio com os níveis de tensão *Transistor-Transistor Logic* (TTL).

5.4.1 HARDWARE DO ADAPTADOR

O adaptador representado na figura 32 é uma versão modificada do circuito desenvolvido pelo RobotsEverywhere (2014a).

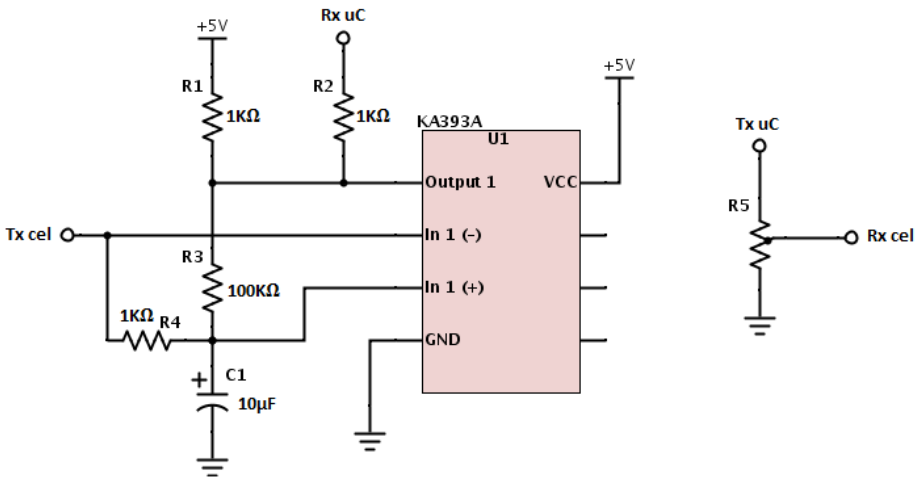

Figura 32: Diagrama Esquemático do Adaptador P2-Serial
fonte: baseado em RobotsEverywhere (2014a).

O circuito da figura 32 apresenta algumas melhorias em relação ao do RobotsEverywhere (2014a), sendo elas:

- Utilização de um comparador ao invés de um amplificador operacional sem realimentação;
- Capacidade de auto ajustar-se com diferentes níveis de *offset* na entrada;
- Comunicação bidirecional.

Estas melhorias permitiram que diversos modelos de *smartphones* pudessem ser utilizados sem necessidade de ajuste manual. Além disso, a comunicação serial, expandida também em software, permitiu que houvesse uma troca de dados entre o *smartphone* e a base robótica. Vale salientar que esta abordagem é compatível com qualquer plataforma robótica que possua um microcontrolador com pelo menos um periférico *Universal Asynchronous Receiver Transmitter*(UART) disponível.

5.4.2 INTERFACE DE SOFTWARE DO ADAPTADOR

Para que a comunicação entre o *smartphone* e a base robótica fosse completa, foi necessário desenvolver uma biblioteca que implementasse a comunicação serial em *software*.

Esta biblioteca, desenvolvida apenas para o sistema operacional Android, utiliza os conversores analógico-digital e digital-analógico de áudio do *smartphone* para produzir e ler os sinais analógicos. A máxima taxa de amostragem destes conversores é de 48kHz, o que limita a comunicação em aproximadamente 24Kb por segundo.

A comunicação serial implementada foi a 8N1 que consiste em 10 bits por pacote, sendo o primeiro um *start bit* seguido de 8 bits de dados e um *stop bit* sem bits de paridade. O *baudrate* foi fixado em 9600 bits por segundo, pois este, além de ser uma taxa de transmissão padrão, apresentou uma menor quantidade de leituras incorretas.

Para a entrada de dados do *smartphone*, foi necessário um pré-processamento pois o sinal era conforme a figura 33.

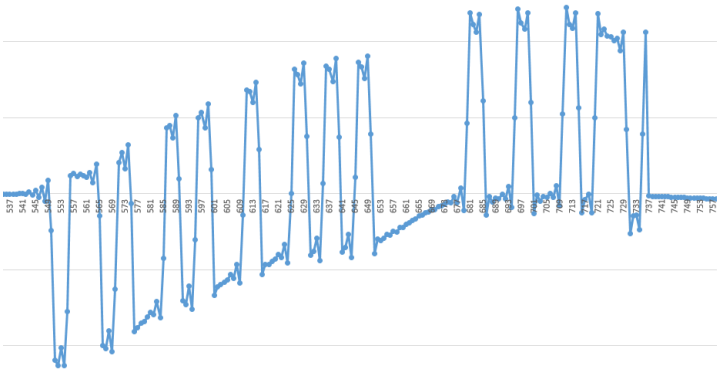

Figura 33: Leitura do Sinal de Entrada do *Smartphone*

Este sinal de entrada (figura 33) dificultava a determinação dos níveis lógicos pois havia uma flutuação de baixa frequência da componente DC. O pré-processamento é realizado a partir de um filtro passa-alta (leitura atual menos leitura passada) e resulta no sinal representado pela figura 34 obtida na mesma forma que o sinal da figura 33.

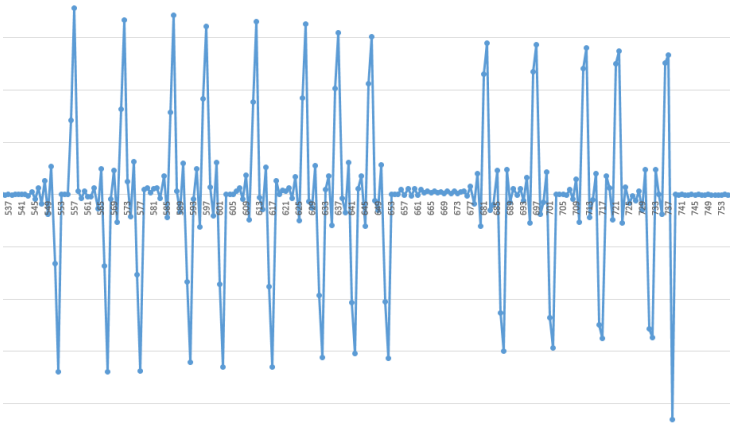

Figura 34: Sinal Após Filtro Passa-Alta

O resultado da aplicação do filtro passa-alta é um sinal sem variação na componente DC, o que permitiu que os níveis lógicos fossem determinados com uma maior precisão.

Para a obtenção desses dados, foi necessário utilizar o software do adaptador áudio-serial em conjunto com o depurador do Android (Android Debug Bridge). Sendo assim, foi possível parar a execução do programa dentro da rotina que processava os sinais da entrada de áudio do *smartphone* e então obter duas listas com os valores lidos. A primeira continha os dados crus e a outra, os dados pós-processados. Ambas foram

utilizadas para gerar, respectivamente, os gráficos das figuras 33 e 34.

Para a saída dos dados do *smartphone*, o algoritmo desenvolvido foi baseado em um código pré-existente disponibilizado por RobotsEverywhere (2014b). A modificação principal para este projeto foi que o *offset* gerado por *software* foi retirado. Anteriormente, o nível lógico 0 era -128 e o 1 era 16, com o novo código eles foram para, respectivamente, -72 e 72. Esta modificação foi necessária para que a comunicação funcionasse com o *hardware* desenvolvido.

5.5 COMPARAÇÃO ENTRE AS ABORDAGEM I E II

A abordagem II se mostrou vantajosa em relação à I em diversos fatores. Primeiramente, o controle do robô ficou mais robusto, preciso e independente da capacidade de processamento do *smartphone*.

Outra melhoria foi a possibilidade de adicionar sensores à base robótica, pois a comunicação que anteriormente era unidirecional passou a ser bidirecional, permitindo assim que o *smartphone* recebesse estas informações através do canal de áudio.

Além disso, a transmissão dos comandos que anteriormente era limitada a 50ms, passou para 4ms por comando, diminuindo assim o atraso entre a tomada de decisão pelo *smartphone* e a execução das ações pelo robô.

5.6 ANÁLISE DA AVALIAÇÃO PREDITIVA DA USABILIDADE

Esta seção apresenta a análise da Plataforma Coffee pelos próprios integrantes da equipe. Esta pesquisa resultou em seis heurísticas gerais e quinze específicas, além de observações que justificam a avaliação realizada.

5.6.1 RESULTADOS

O formulário contendo as heurísticas para *websites* (ver Apêndice A) foi respondido pelos três integrantes da equipe. Estes atribuíram graus de completude e severidade (ver seção 3.4.4) para as categorias avaliadas. Para a realização dessa análise, os critérios de avaliação estabelecidos por Rutter (2004) para heurísticas *Web* foram utilizados. A tabela 6 apresenta os resultados obtidos nessa avaliação.

Heurísticas Gerais	**Heurísticas *Web***	**Avaliação**			
		Excelente	Bom	Satisfatório	Precisa de Melhorias
Visibilidade do Status do Sistema	Tempo de carga do *website*	3	0	0	0
Compatibilidade do Sistema com o Mundo Real	Conhecimento do Público Alvo	3	0	0	0
	Escopo do Conteúdo	3	0	0	0
	Conteúdo	3	0	0	0
Consistência e Padrões	Navegação	3	0	0	0
	Escrita e Gramática	3	0	0	0
	Links	3	0	0	0
Prevenção de Erros	Compatibilidade	0	3	0	0
Estética e *Design* Minimalista	Fontes	3	0	0	0
	Layout	3	0	0	0
	Escolha das Cores	3	0	0	0
	Background	3	0	0	0
Help e Documentação	Informações de Contato	2	1	0	0
	Copyright	0	2	0	1
	Metatags	0	0	0	3

Tabela 6: Tabela com os resultados da Avaliação Preditiva da Usabildiade

A heurística que avalia o tempo de carga do *website*, foi avaliada como "Excelente" pelos três integrantes da equipe. A observação feita por eles indicou que nenhum relato de atraso foi identificado ao carregá-lo.

A segunda heurística geral analisada, compatibilidade do sistema com o mundo real, indica que o conhecimento sobre o público-alvo, o conteúdo do *website* e seu escopo foram avaliados como "Excelente". Os principais motivos desta atribuição foram as constatações de que o conteúdo do *website* era interessante para os usuários finais, era objetivo com o propósito da aplicação, e que as informações contidas nele eram precisas.

O terceiro grupo de heurísticas analisadas corresponde à consistência de padrões no *website*. As categorias analisadas foram: navegação; escrita e gramática; e *links*. Todas foram marcadas como "Excelente" segundo os integrantes. As justificativas envolveram afirmações de que a navegação pelas páginas era feita com facilidade e que o painel superior facilitava esse propósito. Além disso, os integrantes não encontraram erros de gramática e os *links* apontavam para páginas com qualidade esperada por um usuário.

A quarta heurística geral avaliada, foi a de prevenção de erros em relação à compatibilidade entre sistemas operacionais e diferentes navegadores. Esse item foi avaliado como "Bom" para os três integrantes. O problema indicado por eles foi que apesar do *website* ter sido testado nos navegadores Chrome, Firefox, Safari e Internet Explorer, ele não foi testado em outros. Entretanto a severidade apontada para esse foi baixa, sendo declarada como um problema cosmético para dois integrantes e pequeno para outro. O principal motivo indicado foi o fato de haver um botão mal disposto e o tamanho de alguns elementos ficarem fora do esperado em alguns navegadores, entretanto, essa ocorrência não afetaria a compreensão do *website* em sentido algum.

A avaliação para a quinta heurística: "Estética e *design* minimalista" foi avaliada como "Excelente" para os integrantes. A principal observação dos avaliadores foi que as fontes, o *layout* para alguns navegadores específicos, a escolha das cores e o *background* apresentaram uma disposição agradável. Um dos comentários relevantes foi que o contraste de cores dos botões facilitava suas localizações. Outra observação foi que a compreensão do código JavaScript gerado, ao traduzir o programa em blocos, foi facilitada pela formatação utilizada na área de texto.

Para a última heurística geral, *help* e documentação as avaliações foram dispersas. A primeira observação levantada foi que o *website* não possuía data de publicação nem edição. A severidade deste problema foi avaliada como pequena para todos, pois não afetaria o uso do *website* pelo público-alvo.

O segundo problema identificado foi em relação ao *Copyright*. Ele foi caracterizado como grave para um integrante e pequeno para outros dois. Um dos comentários para esse problema foi que o Blockly não foi citado em momento algum. Desta maneira, os alunos interessados por essa forma de programação poderiam ser limitados ao uso da Plataforma Coffee. Outro integrante mencionou que a ferramenta Blockly não é referenciada por outros *websites* que a utilizam.

A terceira observação realizada por todos os integrantes foi que *Metatags* não foram empregadas pelo *website*. Esse problema foi classificado com nível de severidade

pequeno por dois avaliadores e cosmético por outro. Uma justificativa apontada foi que para a avaliação do projeto em sala de aula, o *link* do *website* poderia ser passado aos alunos. Cabe ressaltar que a severidade avaliada poderia ser maior caso a utilização da Plataforma Coffee fosse realizada em maior escala, visto a falta de utilização de *Metatags* implica na dificuldade de aparição em resultados de ferramentas de pesquisa.

A principal justificativa apontada por um integrante foi que para a avaliação do projeto em uma sala de aula o *link* do *website* poderia ser passado. aos alunos e, portanto, o problema não seria relevante, porém tornava-se relevante para a sua futura divulgação visto que a afetaria na busca da Plataforma Coffee em resultados de pesquisa.

5.6.2 DISCUSSÃO DOS RESULTADOS DA AVALIAÇÃO HEURÍSTICA

Como pode ser observado, não houve variação significativa entre as resposta dadas no formulário. Para grande parte das categorias que foram avaliadas, a resposta "Excelente" foi a mais afirmada.

Isso indica que os avaliadores consideraram o conteúdo do *website* consistente em relação à navegação, gramática e links. Além disso, foi apontado que a estética é agradável e a página não foge de seu escopo.

Um dos motivos que pode explicar esse fato foram as diversas revisões no conteúdo do *website* com o objetivo de facilitar o seu uso e torná-lo agradável. O *framework* MVC e as ferramentas de versionamento para o compartilhamento de projetos entre os integrantes da equipe demonstraram-se fundamentais para o desenvolvimento da plataforma.

Apesar de grande parte dos problemas identificados possuírem níveis de severidade baixo para a aplicação desenvolvida, as correções podem ser realizadas em projetos futuros da plataforma.

Cabe ressaltar que a avaliação preditiva da usabilidade foi realizada apenas por três pessoas, sendo que todas participaram do projeto e desenvolvimento do *website*. Portanto, o resultado pode ter sido enviesado e para uma análise mais robusta, outros especialistas poderiam contribuir em sua avaliação.

A heurística geral "Estética e *design* minimalista" poderia ser analisada por *designers*, pois envolvem conhecimentos específicos da área.

Além disso, a participação de professores poderia trazer contribuições relevantes através de suas observações em relação à heurística geral "Compatibilidade do sistema com o mundo real" que depende do conhecimento dos interesses dos estudantes e do propósito

do *website*.

6 ESTUDO DE CASO

Este capítulo apresenta a análise da aplicação da ferramenta em sala de aula e os passos realizados para o seu cumprimento.

6.1 ELABORAÇÃO DOS PROBLEMAS PARA A APLICAÇÃO EM SALA DE AULA

Para a elaboração dos problemas, as características descritas por Kolmos et al. (2007) foram utilizadas (seção 2.1.3). Além disso, tutoriais também foram feitos para que os alunos pudessem recorrer a eles caso aparecessem dúvidas.

Os problemas elaborados para a aplicação em sala de aula foram desenvolvidos considerando uma turma de Computação I[1] no início do semestre e tiveram como objetivo o ensino de estruturas do pensamento computacional, como: sequência de instruções, laços de repetição, condições e variáveis. A elaboração da lista de problemas para a aplicação em sala de aula foi baseada em outros experimentos realizados por PETECO (2014) e McGill (2012).

Os problemas apresentaram diferentes níveis de dificuldades e o mais avançado era desafiador e complexo por envolver diversas estruturas de programação. Alguns deles possuíam múltiplas formas de resolução. Um exemplo disso foi o problema do robô seguir a linha para o qual foram encontradas três soluções distintas antes de sua aplicação em sala de aula.

Assim, a seguinte lista de problemas dentro do seu contexto de aprendizado foi criada:

1. Ambientação com o robô:
    - Realizar movimentos simples (frente, trás, esquerda, direita).
2. Aprendizado de algoritmos básicos e sequenciais:

[1] Computação I é uma disciplina lecionada na Universidade Tecnológica Federal do Paraná (UTFPR). Sua ementa é apresentada no Anexo A.

- Seguir um caminho pré-determinado.
- Fazer com que o robô faça formatos geométricos com sua movimentação.

3. Aprendizado de Laços:
   - Fazer com que o robô faça formatos geométricos com sua movimentação utilizando laços de repetição:
     - i Desenhar triângulo três vezes.
     - ii Desenhar quadrado quatro vezes.
     - iii Desenhar círculo infinitamente.
4. Aprendizado de Condições:
   - Fazer com que o robô aponte somente para um ponto cardeal, mesmo quando é rotacionado manualmente.
5. Aprendizado de Variáveis:
   - Realizar aceleração e desaceleração da velocidade do robô.
6. Aprendizado de algoritmos mais complexos inseridos em um contexto maior:
   - Robô segue a linha.
   - Robô desvia de objetos.

6.2 DINÂMICA DA APLICAÇÃO DA PLATAFORMA COFFEE

Esta seção apresenta a dinâmica da aplicação da Plataforma Coffee em sala de aula. Ela contempla a descrição de todos os passos seguidos nesta atividade, sendo eles: apresentação da plataforma, exemplo de utilização, aplicação de questionários (ver Apêndice B), resolução de exercícios pelo corpo discente e discussão sobre a ferramenta.

A Plataforma Coffee foi aplicada em uma turma de Computação I no fim do semestre. Esta turma era não regular, o que implica em uma diversidade de cursos, experiências prévias com programação e interesses.

Esta dinâmica contou com a participação de sete alunos em toda a sua duração e foi realizada em dois dias diferentes espaçados em uma semana. Cada dia contemplou um cenário distinto e, consequentemente, comportamentos distintos por parte dos alunos. Em ambos os dias apenas um robô, situado na frente da sala de aula, estava disponível para os testes dos programas criados pelos alunos.

No primeiro dia, havia uma entrega do projeto da disciplina de Computação I e a utilização do Coffee era optativa. Este fato dividiu a turma entre os que ainda não haviam entregado o projeto e os que estavam disponíveis para utilizar a plataforma. A duração do curso nesse dia foi de duas horas.

Inicialmente a ideia geral e os objetivos da plataforma Coffee foram apresentados aos alunos. Além disso, um exercício simples foi exposto e resolvido em sala, o que tornou possível mostrar o funcionamento do aplicativo e do acoplamento de blocos. O exercício envolvia uma rota simples que o robô deveria executar, então uma imagem que descrevia esta rota foi apresentada e sua solução passo a passo foi realizada na plataforma.

Logo após o exemplo, os alunos foram informados que a dinâmica era voluntária e não haveria qualquer tipo de bonificação para aqueles que participassem. Além disso, aqueles que o fizessem deveriam assinar um Termo de Consentimento Livre e Esclarecido (TCLE).

Após consentir com a dinâmica, o corpo discente preencheu o questionário pré-curso. Em seguida, foram apresentados outros problemas aos alunos de forma que eles tentassem resolvê-los por conta própria ou em grupos.

Durante as atividades do primeiro dia houveram algumas falhas técnicas com o robô que puderam afetar as impressões dos alunos. Inicialmente houve um problema em que uma variável de velocidade do robô não estava sendo reiniciada no início de cada execução, isso resultou em comportamentos inesperados. Após a identificação deste problema, o robô era reinicializado toda vez que um novo programa era executado.

No segundo dia os alunos puderam se dedicar totalmente ao uso da plataforma, pois não haviam apresentações ou entregas de trabalhos. Os alunos que não tiveram contato com a ferramenta no primeiro dia puderam fazer os exercícios iniciais. A resolução de problemas mais complexos, como fazer o robô seguir uma linha preta ou desviar de obstáculos, foi proposta para aqueles que já haviam passado pelo básico no curso anterior.

Após uma hora de atividades os alunos foram convidados a responder o questionário pós-curso e, depois disso, uma discussão sobre a ferramenta foi iniciada e as observações e comentários dos alunos foram registrados. Nesse segundo dia não houveram falhas técnicas significativas.

A discussão em sala de aula foi importante para coletar as informações que não seriam respondidas pelos questionários. Estes dados auxiliaram na obtenção de uma pesquisa mais completa e detalhada.

Em ambos os dias, os integrantes da equipe desenvolvedora da Plataforma Coffee procuraram agir como facilitadores e instigaram os alunos a desenvolver sua capacidade de raciocínio sem dar soluções imediatas às questões dos alunos. Portanto, o ciclo do PBL (ver figura 1) foi parcialmente seguido e procurou-se centralizar o aprendizado nos alunos, e no desenvolvimento do pensamento computacional deles.

Inicialmente, a identificação de fatos foi feita pelos próprios alunos, que buscaram criar hipóteses para solucionar os problemas. Quando um aluno possuía deficiência de conhecimento em alguma estrutura de programação ou funcionamento de algum bloco, foi recomendado o estudo desses pelos tutoriais desenvolvidos caracterizando, assim, o incentivo de um aprendizado autodidata. Entretanto, alguns alunos insistiram em procurar entender melhor a solução sem os tutoriais através de tentativa e erro. Após a compreensão do novo conhecimento, eles aplicaram-no no problema que estavam resolvendo e finalmente, um resumo do que foi feito para solucionar o problema foi apresentado pelo facilitador para que os alunos refletissem sobre as etapas da solução.

Uma das deficiências de conhecimento dos alunos percebida pelos facilitadores foi em relação ao pensamento sequencial de um programa em blocos. O aluno em questão havia apresentado todos os blocos e estruturas separadamente para a solução de um problema, entretanto ele não conseguiu integrá-los para resolvê-lo. Em outra situação houve a confusão do valor verdadeiro de uma condição e a movimentação que o robô deveria fazer caso essa condição fosse ativada. Para essas situações os facilitadores identificaram essas deficiências e desenvolveram o pensamento dos alunos procurando corrigir os conceitos errôneos desses.

Para esta pesquisa, apenas os alunos que responderam ambos os questionários foram considerados. O mapeamento daqueles que responderam o pré-curso e pós-curso foi realizado através da idade e curso. Cabe ressaltar que este método funcionou devido ao número reduzido de alunos na dinâmica e à característica da turma, o que não implicou em coincidências na combinação utilizada. Para aplicações futuras, este cruzamento de dados deve ser realizado com uma identificação sem ambiguidades que mantenha o anonimato, por exemplo um apelido fictício para cada aluno.

6.3 RESULTADOS OBTIDOS

A pesquisa realizada contou com doze alunos no primeiro dia e dez no segundo, entretanto apenas sete responderam os questionários pré e pós-curso. Portanto, para fins de consistência, apenas estes sete alunos foram considerados na análise.

Os sete alunos que participaram da aplicação da Plataforma Coffee em ambos os dias possuíam dispositivos móveis com *Wi-Fi* e conector externo para fone de ouvido. Além destes recursos, todos os *smartphones* possuíam câmeras, um não possuía GPS e três possuíam acelerômetro. Estes dados estão apresentados na figura 35.

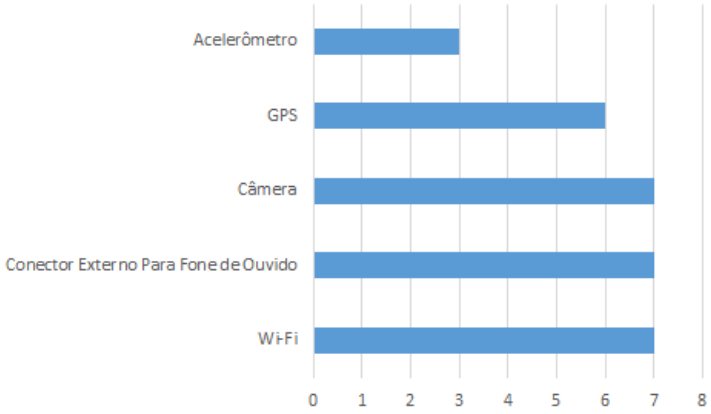

Figura 35: Recursos dos *Smartphones* dos Alunos

Os próximos resultados são divididos em duas partes. A primeira apresenta medidas de expectativas, reflexões dos alunos e componentes da motivação (atenção, relevância, confiança e satisfação). A segunda parte é separada pela Plataforma Coffee e seus subgrupos (Blockly e robô).

6.3.1 MEDIDAS DAS EXPECTATIVAS, REFLEXÕES E COMPONENTES DA MOTIVAÇÃO DOS ALUNOS

Esta subseção apresenta os resultados dos questionários aplicados pré e pós-curso voltados às medidas das expectativas, reflexões e componentes da motivação dos alunos. Estes resultados são apresentados nos diversos subgrupos que seguem.

O primeiro subgrupo é o da atenção. Este componente estava presente nos questionários pré e pós-curso e os resultados estão presentes na figura 36.

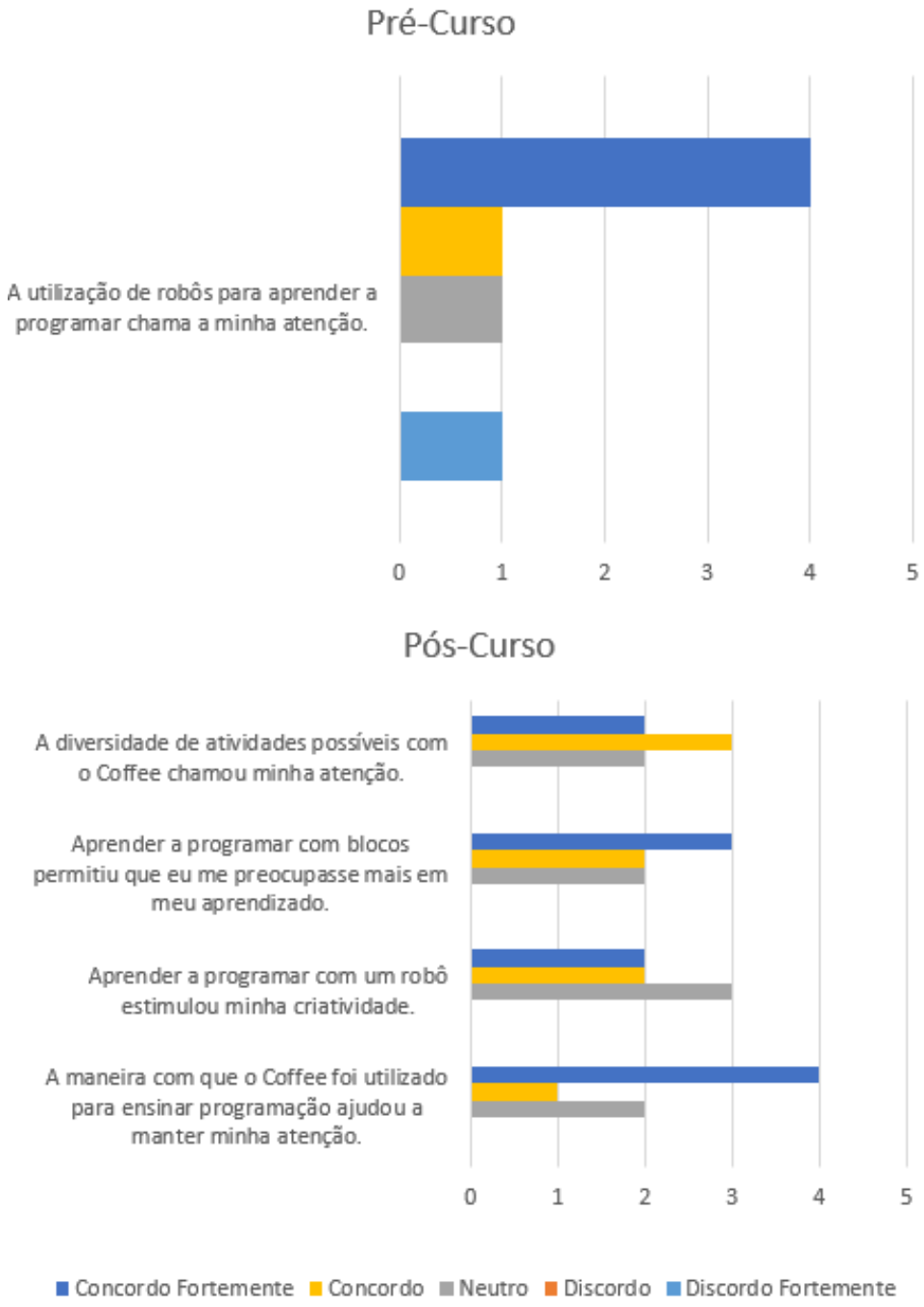

Figura 36: Gráfico de Perguntas e Resultados da Atenção

O gráfico presente na imagem 36 mostra que no questionário pré-curso um aluno não apresenta indícios de atenção em utilizar robôs em sala de aula enquanto os outros tendem do neutro ao positivo neste mesmo ponto. Já no questionário pós-curso, todas as questões relacionadas à atenção tenderam de respostas neutras a positivas.

O segundo subgrupo é o da relevância, também presente em ambos os questioná-

rios. As perguntas e resultados são ilustrados na figura 37.

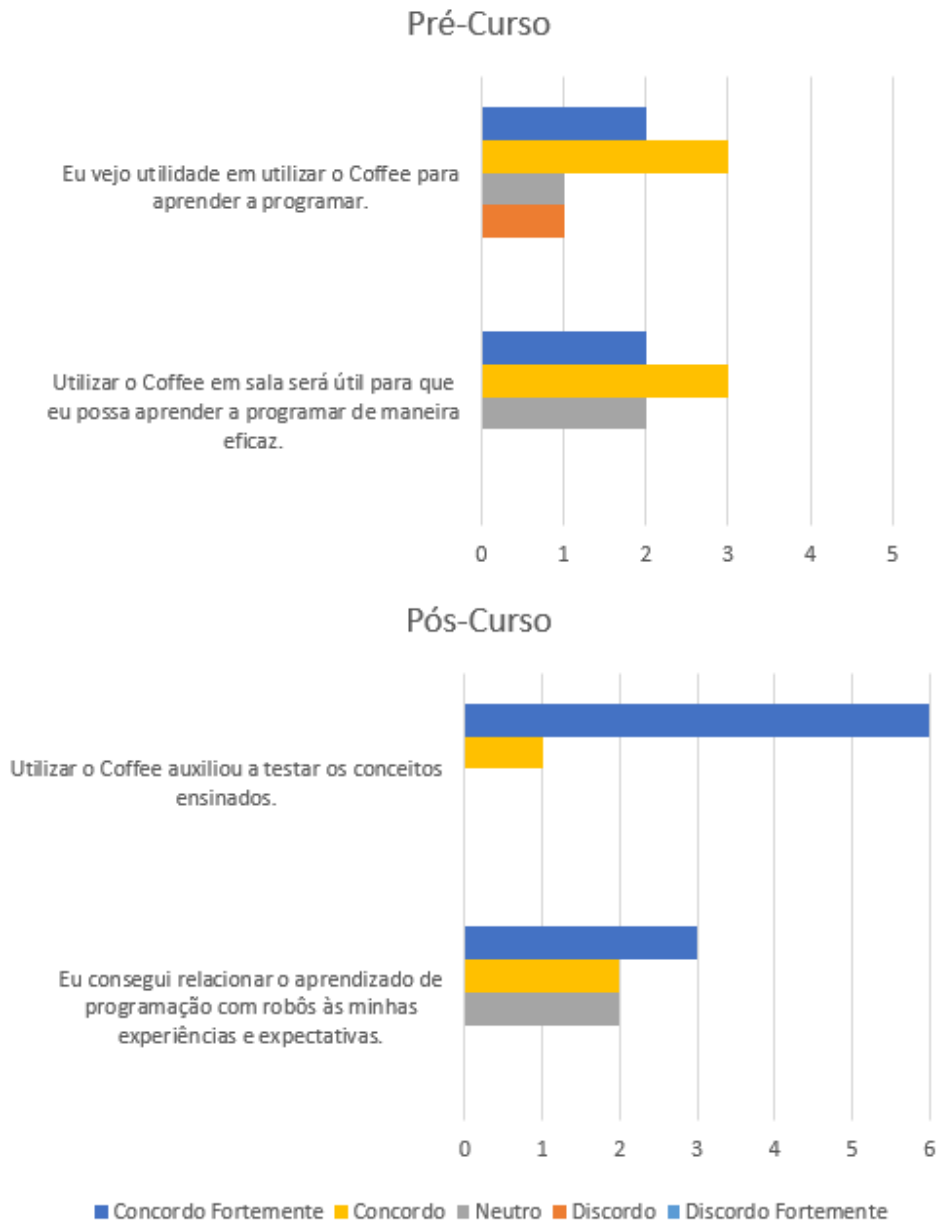

Figura 37: Gráfico de Perguntas e Resultados da Relevância

A figura 37 mostra que no questionário pré-curso houve neutralidade e discordância na questão: "Eu vejo utilidade em utilizar o Coffee para aprender a programar". Entretanto, no pós-curso os alunos responderam apenas positivamente quando questionados sobre o auxílio da plataforma em testar os conceitos ensinados.

O terceiro subgrupo questionado no pré e no pós-curso é o da confiança, ilustrado na figura 38.

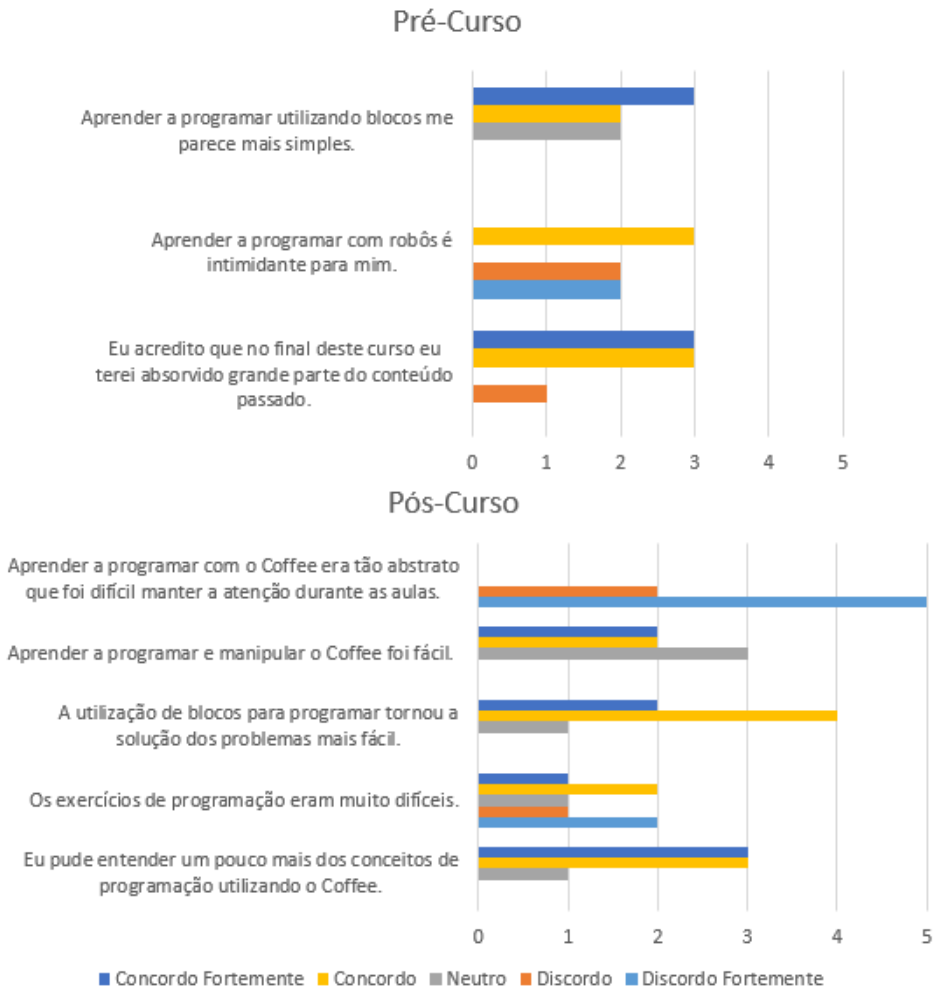

Figura 38: Gráfico de Perguntas e Resultados da Confiança

A figura 38 ilustra uma dispersão em relação à confiança dos alunos, principalmente quando abordados em relação à utilização de robôs no pré-curso e ao nível de dificuldade dos problemas apresentados. Outro ponto relevante é que dos sete alunos, seis concordaram que a utilização de blocos na programação tornou a solução dos problemas mais fácil.

O quarto componente analisado contempla apenas uma pergunta questionada antes do curso e seu resultado está exposto na figura 39.

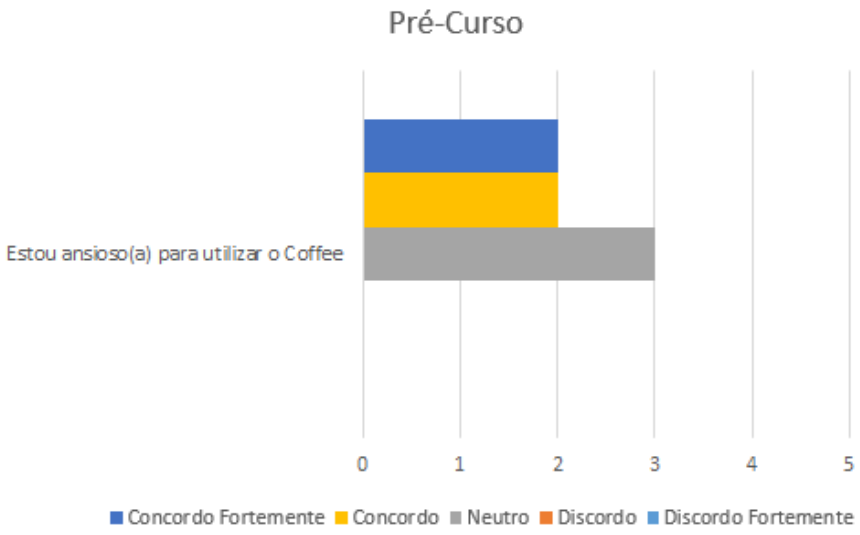

Figura 39: Gráfico de Perguntas e Resultados das Atitudes e Percepções

O gráfico presente na figura 39 mostra que inicialmente a plataforma não cativou todos os alunos. Dos sete, três se mostraram indiferentes em relação à sua utilização.

Ao serem questionados sobre a satisfação de utilizar a Plataforma Coffee, os alunos apresentaram, em geral, uma resposta positiva. Conforme ilustrado na figura 40, dos sete alunos, seis não se sentiram entediados ou desestimulados ao utilizar a ferramenta e um se mostrou indiferente.

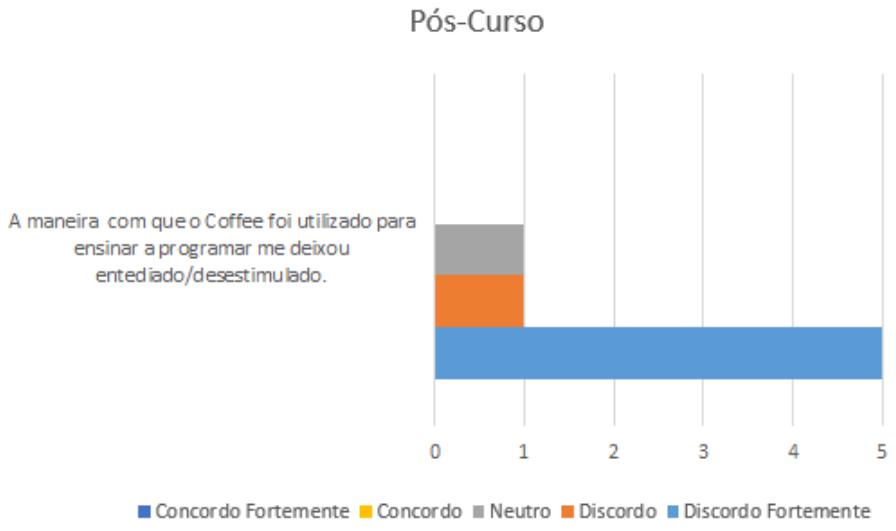

Figura 40: Gráfico de Perguntas e Resultados da Satisfação

O último grupo de questões realizadas no questionário pós-curso abordou os alunos em relação às suas expectativas e cursos futuros voltados à programação. Os resultados

estão presentes na figura 41.

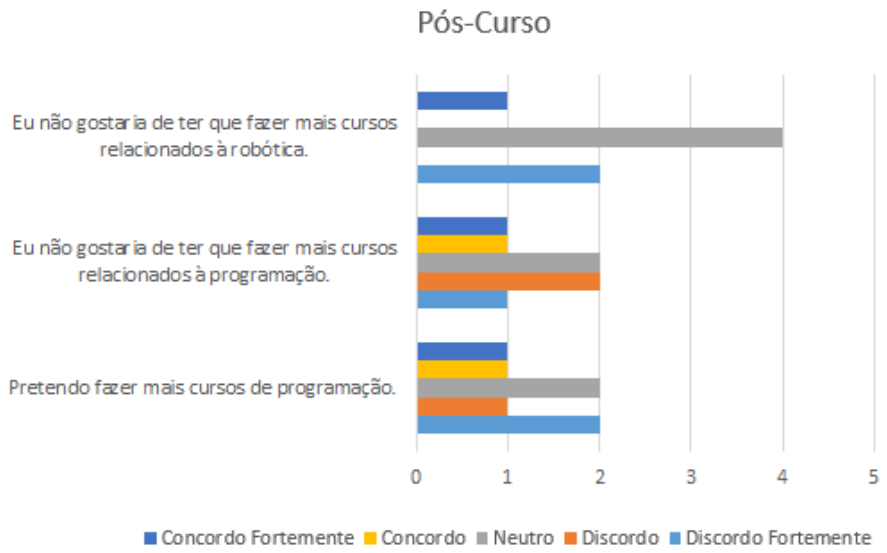

Figura 41: Gráfico de Perguntas e Resultados das Expectativas e Cursos Futuros

Conforme indicado na figura 41, os alunos possuem respostas divergentes em relação às suas expectativas em fazer outras matérias de robótica e programação.

6.3.2 MEDIDAS DA PLATAFORMA COFFEE E SEUS SUBGRUPOS BLOCKLY E ROBÔ

Esta subseção apresenta os resultados dos questionários aplicados pré e pós-curso voltados à análise da Plataforma Coffee e seus subcomponentes Blockly e robô.

Os primeiros gráficos, apresentados na figura 42, ilustram as respostas dos alunos em relação à utilização do Blockly. Todas as respostas relacionadas à simplicidade e facilidade de utilizar blocos foram de neutras a positivas.

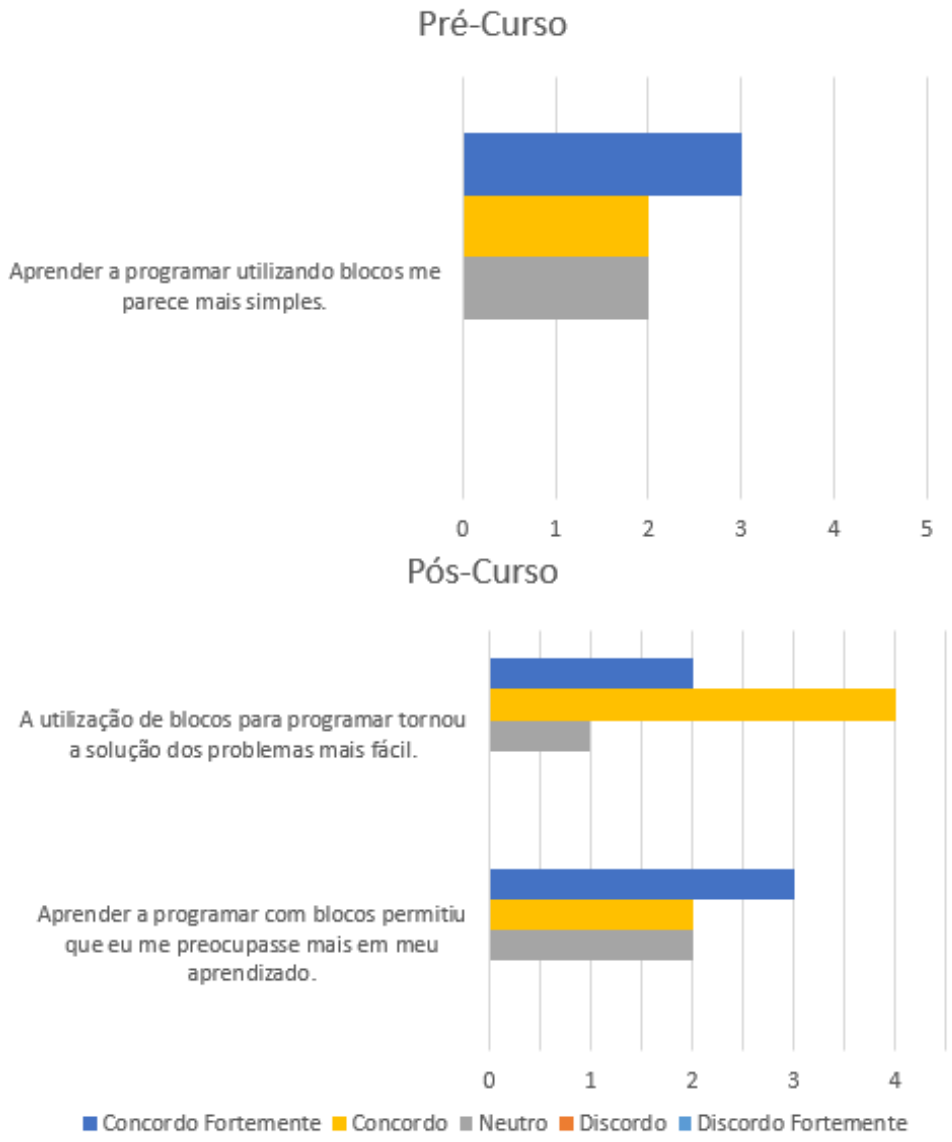

Figura 42: Gráfico de Perguntas e Resultados Referentes ao Blockly

A figura 43 apresenta as respostas dos alunos quando perguntados em relação à utilização de um robô.

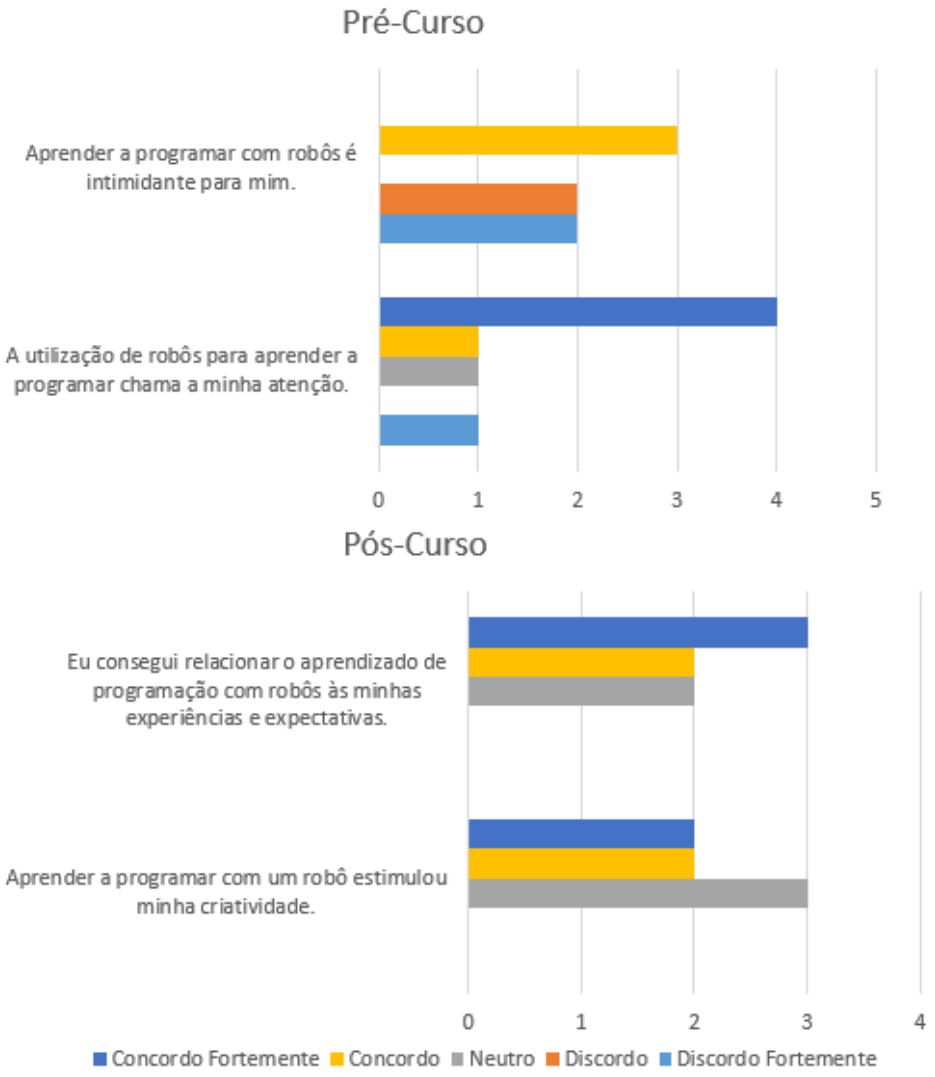

Figura 43: Gráfico de Perguntas e Resultados Referentes ao Robô

Conforme ilustrado na figura 43, no questionário pré-curso, três dos sete alunos declararam que se sentiam intimidados em utilizar robôs. Além disso, cinco deles responderam que programar o robô chamava a atenção. No pós-curso as respostas tenderam de neutras a positivas.

A última figura (44), indica as respostas dos alunos quando questionados em relação à Plataforma Coffee.

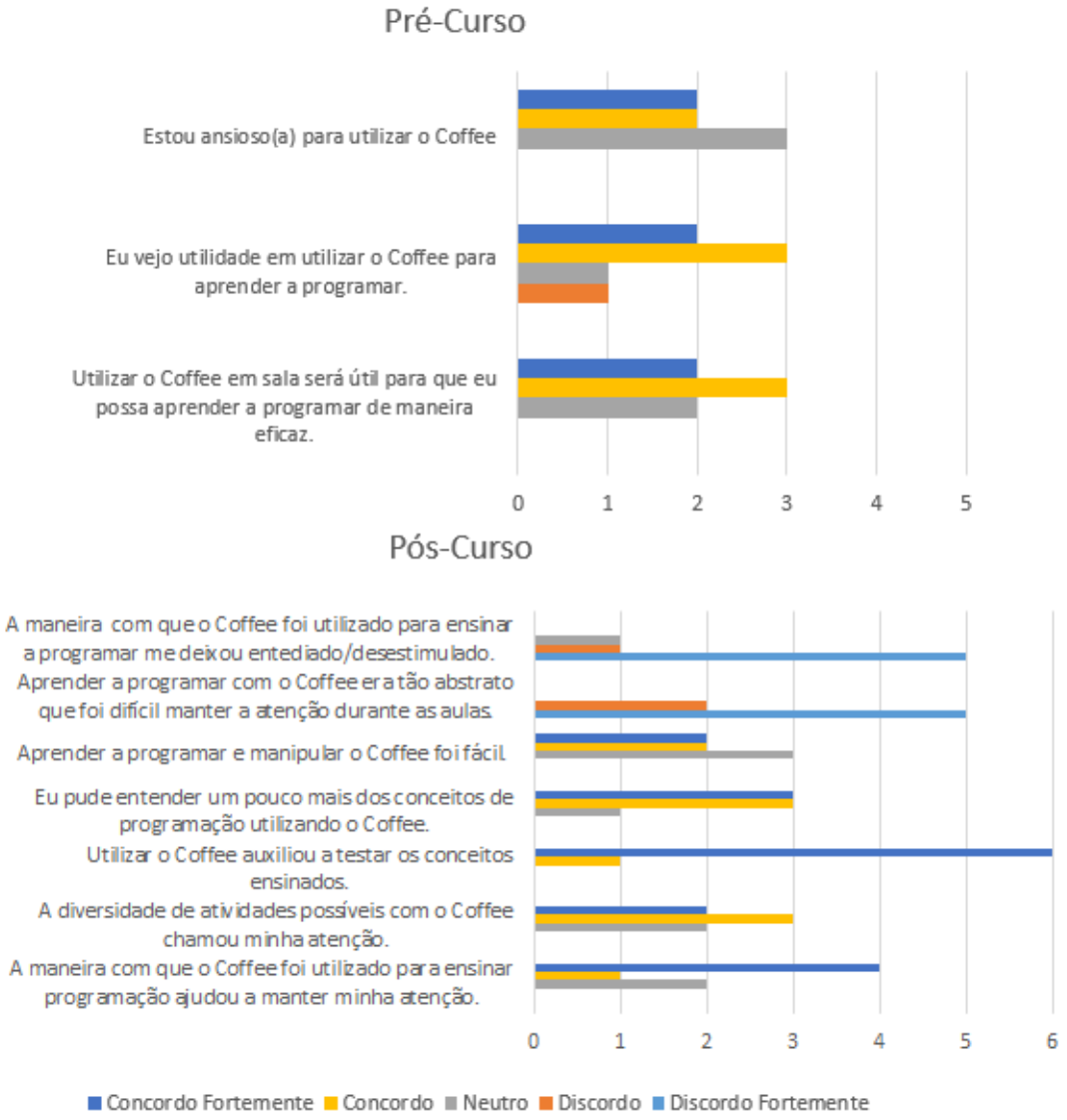

Figura 44: Gráfico de Perguntas e Resultados Referentes à Plataforma Coffee

Todas as respostas apresentadas na figura 44 tiveram uma tendência positiva. No questionário pré-curso, um dos alunos não via utilidade em utilizar a Plataforma Coffee para aprender a programar. É interessante notar que em todas as respostas do questionário pós-curso as funções da ferramenta apresentada foram de neutras a positivas.

6.3.3 OBSERVAÇÕES REALIZADAS DURANTE AS APLICAÇÕES EM SALA DE AULA

Esta subseção apresenta as observações dos alunos e facilitadores durante a dinâmica realizada com a turma de Computação I.

Durante as aplicações da Plataforma Coffee em sala de aula, apareceram diversas

questões sobre os blocos. Este fato levou às constatações de que a semântica deles deve ser revisada e deve haver um rearranjo dos grupos de blocos para facilitar a interação entre o aluno e a interface de programação.

Outro ponto interessante verificado durante a aplicação da plataforma é que os alunos não tiveram qualquer dúvida relacionada aos mecanismos da página *Web*, ou seja, criaram contas e projetos antes da apresentação dessas funções.

Para contemplar questões além das que o questionário propunha, a discussão em sala de aula foi realizada. Ela permitiu que os alunos expressassem suas impressões sobre a plataforma. Ficou evidente a demanda por um simulador integrado à página *Web* por questões de praticidade ao testar o código antes de aplicá-lo ao robô. Outra requisição foi a de suportar outras linguagens de programação que fossem textuais.

Os alunos mencionaram apenas uma modificação referente ao robô, sendo ela a criação de uma camada externa que permitisse omitir os componentes eletrônicos e aumentar a robustez do mesmo em seu manuseio.

Um dos comentários mais relevantes da discussão foi o pedido de que plataformas como o Coffee fossem utilizadas desde o início em turmas de Computação I sem que esta ideia tivesse sido apresentada anteriormente. Os alunos ainda alegaram que a ferramenta era interessante para diminuir o preconceito com programação e auxiliar na compreensão de diversos conceitos da disciplina.

6.4 DISCUSSÃO DO ESTUDO DE CASO

Analisando o gráfico apresentado na imagem 35 na seção 6.3, percebe-se que todos os alunos que utilizaram a Plataforma Coffee possuíam *smartphones* com os requisitos mínimos necessários (*Wi-Fi* e conector externo para fone de ouvido) para utilizá-la. Este fato possibilita que, posteriormente, cada aluno possa manusear sua própria plataforma.

Um outro dado importante levado em consideração foi a variedade de cursos dos alunos da turma, sendo eles: Licenciatura em Física, Bacharelado em Química, Engenharia Elétrica e Engenharia Mecânica. Esta variedade pode ter influênciado esta pesquisa.

Observando as respostas relacionadas às componentes da motivação, os alunos de Física e Química mostraram uma maior indiferença e até negatividade quando questionados em relação à utilização de robôs e da Plataforma Coffee. Entretanto, esta negatividade sofreu alterações após a aplicação em sala de aula.

Há indícios de que o aluno do curso de Química mudou sua opinião sobre a plataforma entre o início e o fim do curso. Inicialmente ele discordava fortemente que a utilização de robôs para aprender a programar chamava sua atenção. No questionário pós-curso, ele se tornou neutro neste ponto. Isso indica que é possível que a Plataforma Coffee auxilie na componente atenção da motivação.

Para a componente relevância da motivação, houve uma melhora geral entre as respostas pré e pós-curso. Isso indica que mesmo aqueles que se mostraram mais resistentes em utilizar a ferramenta conseguiram encontrar alguma utilidade para a mesma.

Outro fator importante é que o nível de conhecimento e a capacidade de abstração para resolver os problemas propostos era diferente para cada aluno. Isso foi observado em sala de aula e comprovado com o questionário. Quando questionados em relação à dificuldade dos exercícios, as respostas foram dispersas.

Um dos motivos que pode ter afetado a divergência nas respostas sobre a dificuldade dos exercícios aplicados em sala de aula foi a semântica utilizada nos blocos. Isto foi perceptível devido às diversas perguntas que surgiram durante o curso. Esta constatação poderia não ter sido alcançada somente com a avaliação preditiva e os questionários. Por outro lado, a avaliação preditiva da usabilidade indicou que a página *Web* possuia uma boa navegabilidade. Este fato pode ter influenciado na adaptação dos alunos à ferramenta, e consequentemente nas respostas positivas quando questionados sobre a facilidade de utilizá-la.

Inicialmente alguns alunos não se sentiam confiantes em utilizar robôs para aprender a programar. É possível que esta falta de confiança seja relacionada à ausência de uma carcaça na plataforma robótica, omitindo assim os componentes eletrônicos. Esta possibilidade foi levantada durante a discussão em sala de aula e não seria cogitada apenas com os questionários e observações realizadas pela equipe.

A falta de confiança no pré-curso não foi refletida no pós-curso. Possibilidades levantadas por meio da observação dos facilitadores e da discussão final são: a utilização da metodologia PBL apresentada por Kolmos et al. (2007) e a experiência dos alunos na utilização do robô.

Foi constatado que apesar da falta de confiança em utilizar robôs para aprender a programar e do diferente nível de conhecimento dos alunos, nenhum deles se sentiu intimidado a ponto de não querer testar a plataforma. Durante o segundo dia de aula houve um envolvimento geral da sala, sendo que cada um dos alunos permaneceu nos problemas que se sentia confortável para solucioná-lo.

Quando questionados sobre a ansiedade de utilizar a Plataforma Coffee, três dos sete alunos se mantiveram neutros. Isso demonstra que a utilização de robôs e a programação em blocos não implica na curiosidade e ansiedade de todos os alunos. Entretanto, cabe ressaltar que apesar desta indiferença, nenhum deles declarou se sentir desestimulado ou desinteressado ao utilizar a ferramenta. Apenas um aluno se manteve neutro em ambas as respostas.

As expectativas futuras dos alunos foram distintas e provavelmente não estão relacionadas à utilização da Plataforma Coffee. Isso se deve principalmente a dois fatores, o primeiro é que a aplicação em sala de aula não foi ideal. Seria interessante que os alunos pudessem utilizar a ferramenta desde o início do semestre, consolidando assim o conhecimento em programação com o auxílio da plataforma. Esta observação também foi realizada por um aluno em sala de aula, o que fortalece a constatação. Outro fator que precisa ser verificado com um número maior de alunos é que aqueles que são indiferentes à utilização de robôs nem sempre vão mudar suas perspectivas em relação ao assunto. Cabe ressaltar também que o objetivo não é fazer com que todos programem profissionalmente, dada a diversidade de cursos, mas sim melhorar o aprendizado por meio de uma abordagem mais motivadora.

Declarações de entusiasmo e o interesse geral da turma indicam que a Plataforma Coffee possui potencial e deveria ser aplicada e estudada por um período prolongado para a confirmação das conclusões relatadas.

7 CONSIDERAÇÕES FINAIS

O desenvolvimento do projeto apresentado resultou em duas abordagens diferentes. A primeira envolvia a ausência de um microcontrolador na base robótica, entretanto, diversas limitações apareceram no decorrer do trabalho e levaram ao desenvolvimento de uma segunda abordagem. Esta segunda solução corrigiu os problemas da primeira e adicionou uma maior robustez ao controle do robô.

O Apêndice C mostra os passos reais para a conclusão desse projeto. Pode ser observado que ao todo, foram necessárias 956 horas de trabalho, entre elas, 756 de desenvolvimento e 45 de avaliação da ferramenta.

O desenvolvimento da plataforma foi realizado com o auxílio de algumas ferramentas. Entre elas os *frameworks* ASP.NET MVC5 e Cordova, utilizados respectivamente, para a construção dos aplicativos *Web* e *smartphone*. A biblioteca Blockly foi importante para o desenvolvimento do ambiente de programação em blocos. Além disso, os principais componentes de hardware foram o CI K393A para o adaptador áudio-serial e uma placa de desenvolvimento baseada no Arduino Duemilanove (ATMEGA328P-PU).

Em toda a sua execução, o projeto foi desenvolvido com o objetivo de auxiliar no desenvolvimento do pensamento computacional de alunos que cursavam disciplinas de computação. O produto final, nomeado Plataforma Coffee, consiste de:

- Uma página *Web* que permite o desenvolvimento de programas utilizando uma linguagem gráfica em blocos;
- Uma base robótica móvel constituída de sensores, atuadores e um microcontrolador para realizar o controle dos movimentos;
- Um adaptador áudio-serial que proporciona a comunicação entre a base robótica e o *smartphone*;
- Um aplicativo *smartphone* que permite que ao acoplar o dispositivo móvel na base robótica, o programa criado na página *Web* possa ser executado.

A Plataforma Coffee, da maneira que foi desenvolvida, permite que seus componentes sejam expandidos. Um exemplo disso é que a base robótica permite a adição de outros sensores e que o aplicativo *smartphone* poderá ser executado em outros sistemas operacionais desde que um pequeno trecho de código adicional seja desenvolvido. Outro ponto importante é que a base robótica é fracamente acoplada ao restante da plataforma, implicando na possível substituição da mesma conforme desejado.

Além do desenvolvimento da Plataforma Coffee, outro fator importante do projeto foi a aplicação da ferramenta em sala de aula e, consequentemente, o desenvolvimento do estudo de caso. Este passo permitiu que houvesse uma avaliação do produto pelos alunos, o que possibilitou que observações e constatações fossem realizadas.

Em relação ao interesse dos alunos em programação com robótica, foi observado que no questionário pré-curso haviam quatro respostas negativas. Entretanto, no pós-curso houveram apenas respostas neutras e positivas. Além disso, quanto à componente motivacional "Relevância", houveram três respostas neutras e uma negativa no questionário respondido no pré-curso, este número foi reduzido à apenas duas respostas neutras no pós-curso.

Uma das constatações foi que a utilização de robôs em sala de aula não implica no interesse geral da turma. Para o caso deste estudo, os alunos eram de diversas áreas do conhecimento e nem todos demonstraram interesse em utilizar a Plataforma Coffee.

Outra questão relevante foi a avaliação preditiva da usabilidade que fez com que os integrantes da equipe refletissem sobre a página *Web* desenvolvida. Como resultado, esta avaliação indicou que o *website* é consistente em seu conteúdo e é esteticamente agradável, o que pode ter influenciado na adaptabilidade dos alunos com a Plataforma Coffee.

Por fim, esta pesquisa indicou que a Plataforma Coffee possui potencial e deveria ser aplicada e estudada a longo prazo para verificar se as impressões obtidas neste trabalho são mantidas.

7.1 TRABALHOS FUTUROS

A ferramenta desenvolvida se mostrou uma solução viável para a aplicação em salas de aula. Entretanto, as observações ressaltadas pelos alunos indicaram que ainda há ajustes a serem realizados para que a plataforma seja mais atrativa.

Futuramente, o Coffee deveria poder ser visto como um brinquedo, omitindo a

parte eletrônica e aumentando sua robustez para permitir assim que qualquer pessoa possa manuseá-lo sem necessitar de conhecimentos específicos. Além disso, a página *Web* poderia sofrer a adição de um simulador com a finalidade de reduzir a constante utilização da base robótica para verificar a resposta dos programas. Isso permitiria que alunos sem esta ferramenta pudessem utilizar a plataforma.

Ainda em relação à página *Web*, poderia ser adicionada a possibilidade de programar textualmente. Esta modificação permitiria que os alunos evoluíssem seus conhecimentos sem deixar de utilizar a Plataforma Coffee. Além disso, os alunos que já possuem experiência em programação também poderiam utilizá-la.

O *website* poderia ainda contar com uma avaliação preditiva da usabilidade mais robusta, ou seja, seria interessante que especialistas de outras áreas como *design* e pedagogia pudessem avaliar a ferramenta.

Em pesquisas futuras, o ambiente ideal para a aplicação deste projeto seria uma turma de computação no início do semestre, permitindo assim que o conhecimento em programação fosse construído gradativamente e assimilado à outras linguagens de programação.

Para aplicações a longo prazo, seria interessante analisar as implicações de explorar os recursos de compartilhamento e publicações de projetos , adicionar acompanhamentos periódicos, coletar logs e adicionar a avaliação do professor, visto que ele possui grande influência no ambiente da sala de aula.

REFERÊNCIAS

ALMEIDA, L. D. A.; SANTANA, V. F. **Websites Atendendo a Requisitos de Acessibilidade e Usabilidade**. 2008. Disponível em: <http://warau.nied.unicamp.br/warauv2/>. Acesso em: 20 de agosto de 2014.

ANDROID. **Android, The World's Most Popular Mobile Platform**. 2013. Disponível em: <http://developer.android.com/about/>. Acesso em: 16 de dezembro de 2013.

ANDROID. **Android Dashboard**. 2014. Disponível em: <http://developer.android.com/about/dashboards/>. Acesso em: 12 de fevereiro de 2014.

APTANA. **Aptana Studio 3**. Aptana Inc., 2014. Disponível em: <http://http://www.aptana.com/>. Acesso em: 08 de abril de 2014.

ARDUINO. 2014. Disponível em: <http://www.arduino.cc/>. Acesso em: 15 de agosto de 2014.

AROCA, R.; GONÇALVES, L.; OLIVEIRA, P. Towards Smarter Robots with Smartphones. **Robocontrol 2012**, 2012.

AROCA, R. V. **Plataforma Robótica de Baixíssimo Custo para Robótica Educacional**. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2012.

AROCA, R. V.; BURLAMAQUI, A. F.; GONÇALVES, L. M. Method for Reading Sensors and Controlling Actuators Using Audio Interfaces of Mobile Devices. **Sensors**, Molecular Diversity Preservation International, v. 12, n. 2, p. 1572–1593, 2012.

AROCA, R. V. et al. Increasing Students' Interest With Low-Cost CellBots. IEEE, 2013.

BACKBONE. **Backbone**. 2014. Disponível em: <http://backbonejs.org/>. Acesso em: 12 de junho de 2014.

BLOCKLY. **A Visual Programming Editor**. 2013. Disponível em: <https://code.google.com/p/blockly/>. Acesso em: 22 de novembro de 2013.

BLOCKLY. **Block Factory**. 2014. Disponível em: <https://blockly-demo.appspot.com/static/apps/blockfactory/index.html>. Acesso em: 21 de março de 2014.

BRAGA, N. C. Controles pwm de potência. **Instituto Newton C.Braga**, 2009.

BUCERZAN, D.; RATIU, C.; MANOLESCU, M.-J. SmartSteg: A New Android Based Steganography Application. **International Journal of Computers, Communications & Control**, v. 8, n. 5, 2013.

CHEN, M.-C.; CHEN, J.-L.; CHANG, T.-W. Android/OSGi-based Vehicular Network Management System. **Computer Communications**, Elsevier, v. 34, n. 2, p. 169–183, 2011.

CORDOVA, A. **Apache Cordova**. Apache Software Foundation, accessed, 2014. Disponível em: <http://cordova.apache.org/>. Acesso em: 08 de abril de 2014.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures. Chapter 5: Representational State Transfer (REST)**. Tese (Doutorado) — University of California, 7 2000. Disponível em: <http://www.ics.uci.edu/ fielding/pubs/dissertation/rest_arch_style.htm>. Acesso em: 07 de agosto de 2014.

GAUDIELLO, I.; ZIBETTI, E. La robotique Educationnelle: Etat des lieux et Perspectives. **Psychologie Française**, Elsevier, 2012.

GOMES, A.; HENRIQUES, J.; MENDES, A. Uma Proposta para Ajudar Alunos com Dificuldades na Aprendizagem Inicial de Programação de Computadores. **Educação, Formação & Tecnologias-ISSN 1646-933X**, v. 1, n. 1, p. 93–103, 2008.

GOULD, J. D. et al. The 1984 olympic message system: A test of behavioral principles of system design. **Commun. ACM**, v. 30, n. 9, p. 758–769, 1987.

GROVER, S.; PEA, R. Computational Thinking. In: **K-12 A Review of the State of the Field**. [S.l.]: Educational Researcher, 2013. v. 42, n. 1, p. 38–43.

GUIZZO, E.; DEYLE, T. Robotics trends for 2012. **IEEE Robotics & Automation Magazine**, IEEE, v. 19, n. 1, p. 119–123, 2012.

HEAVEN, D. Visual programming means anyone can be a coder. **New Scientist**, Elsevier, v. 215, n. 2879, p. 22, 2012.

ISO. **Ergonomic Requirements for Office Work with Display Terminals (VDTs). Part 11: Guidance on usability.** Geneva, Switzerland, 1998.

JSON. 2014. Disponível em: <http://json.org/>. Acesso em: 15 de julho de 2014.

KATO, S.; HIROYUKI, T. A Style and Tool for Group Exercise of Introductory Programming with LEGO Robot Control as Pre-Education Event. **IEEE**, 2010.

KISS, G. Using The Lego-Mindstorm KIT in German Computer Science Education. In: IEEE. **8th IEEE International Symposium on Applied Machine Intelligence and Informatics**. [S.l.], 2010.

KOLMOS, A. et al. Problem Based Learning. **TREE-Teaching and Research in Engineering in Europe**, 2007.

LEGO. **LEGO MINDSTORM Education EV3 Product Overview**. 2013. Disponível em: <http://education.lego.com>. Acesso em: 02 de dezembro de 2013.

LEGO. **First Lego League**. 2014. Disponível em: <http://www.firstlegoleague.org/>. Acesso em: 10 de fevereiro de 2014.

LIKERT, R. A Technique for the Measurement of Attitudes. **Archives of psychology**, n. 140, p. 1–55, 1932.

LIMA, M. R.; LEAL, M. C. Motivação Discente no Ensino-Aprendizagem de Programação de Computadores. **Educação & Tecnologia**, v. 17, n. 1, 2013.

LUCAS, J.; GREYLING, P. Practical Android Projects. Apress, 2011.

MALONEY, J. et al. The Scratch Programming Language and Environment. **ACM Transactions on Computing Education (TOCE)**, ACM, v. 10, n. 4, p. 16, 2010.

MCGILL, M. M. Learning to Program with Personal Robots: Influences on Student Motivation. **ACM Transactions on Computing Education (TOCE)**, ACM, v. 12, n. 1, p. 4, 2012.

MICROSOFT. **ASP.NET MVC**. 2013. Disponível em: <http://www.asp.net/mvc>. Acesso em: 03 de dezembro de 2014.

MILLS, J. E.; TREAGUST, D. F. Engineering Education - Is Problem-Based or Project-Based Learning the Answer? **Australian Journal of Engineering Education**, 2003. ISSN 1324-5821. Disponível em: <http://www.aaee.com.au/journal/2003/mills_treagust03.pdf>. Acesso em: 22 de fevereiro de 2014.

MIT. **Scratch**. 2013. Disponível em: <http://scratch.mit.edu>. Acesso em: 27 de outubro de 2013.

MIT. **Scratch Wiki Home**. 2013. Disponível em: <http://wiki.scratch.mit.edu/wiki/Scratch_Wiki_Home>. Acesso em: 27 de outubro de 2013.

NETO, W. C. B.; SCHUVARTZ, A. A. Ferramenta Computacional de Apoio ao Processo de Ensino-Aprendizagem dos Fundamentos de Programação de Computadores. In: **Anais do Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2007. v. 1, n. 1, p. 520–528.

NIELSEN, J. Heuristic Evaluation. In: NIELSEN, J.; MACK, R. (Ed.). **Usability Inspection Methods**. New York, NY: John Wiley & Sons, 1993.

PETECO. Robô fun. 2014. Disponível em: <http://dainf.ct.utfpr.edu.br/peteco/>. Acesso em: 02 de agosto de 2014.

PIO, J. L. d. S.; CASTRO, T. H. C. d.; JÚNIOR, A. N. d. C. A Robótica Móvel como Instrumento de Apoio à Aprendizagem de Computação. In: **Anais do Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2006. v. 1, n. 1, p. 497–506.

PRETTIFY. **Prettify**. 2014. Disponível em: <https://code.google.com/p/google-code-prettify/>. Acesso em: 07 de julho de 2014.

RIBEIRO, R. d. S.; BRANDÃO, L. d. O.; BRANDÃO, A. A. Uma Visão do Cenário Nacional do Ensino de Algoritmos e Programação: Uma Proposta Baseada no Paradigma de Programção Visual. In: **Anais do Simpósio Brasileiro de Informática na Educação**. [S.l.: s.n.], 2012. v. 23, n. 1.

ROBOTSEVERYWHERE. **Audio-Serial Adapter**. 2014. Disponível em: <http://www.robots-everywhere.com/re_wiki/index.php?n=Main.AudioSerial>. Acesso em: 15 de agosto de 2014.

ROBOTSEVERYWHERE. **Audio-Serial Adapter Code**. 2014. Disponível em: <http://robots-everywhere.com/re_site/audioserial/>. Acesso em: 15 de agosto de 2014.

ROCHALA, P. **SlimScroll**. 2014. Disponível em: <https://code.google.com/p/google-code-prettify/>. Acesso em: 20 de junho de 2014.

ROMO. **The programmable, Telepresence Robot Toy for Kids and Adults**. 2014. Disponível em: <http://www.romotive.com/>. Acesso em: 16 de maio de 2014.

RUTTER, J. P. Web Heuristic Evaluation III Web Conference. State College, Pennsylvania, 2004.

SANTOS, T. N. d.; SILVA, E. G. d. Proposta de aplicativo para dispositivos móveis que auxiliem no ensino de matemática. **Revista Técnico Científica do IFSC**, v. 1, n. 5, p. 578, 2013.

SHARP, H.; ROGERS, Y.; PREECE, J. Interaction Design: Beyond Human-Computer Interaction. **West Sussex, England: John Wiley & Sons**, 2007.

SHOP, R. 2014. Disponível em: <http://www.robotshop.com/>. Acesso em: 13 de agosto de 2014.

SMARTBOT. **Overdrive your Phone Turn it into a Smartphone Robot**. 2014. Disponível em: <http://www.overdriverobotics.com/>. Acesso em: 16 de maio de 2014.

TELECO. **Estatísticas de Celulares no Brasil**. 2013. Disponível em: <http://www.teleco.com.br/ncel.asp>. Acesso em: 12 de dezembro de 2013.

TWITTER. **Twitter Bootstrap**. 2014. Disponível em: <http://getbootstrap.com/2.3.2/>. Acesso em: 15 de março de 2014.

UTTING, I. et al. Alice, Greenfoot, and Scratch - A Discussion. **ACM Transactions on Computing Education (TOCE)**, ACM, v. 10, n. 4, p. 17, 2010.

W3SCHOOLS. 2014. Disponível em: <http://www.w3schools.com/>. Acesso em: 26 de julho de 2014.

WANG, X.; ZHOU, Z. The Research of Situational Teaching Mode of Programming in High School With Scratch. In: IEEE. **Information Technology and Artificial Intelligence Conference (ITAIC), 2011 6th IEEE Joint International**. [S.l.], 2011. v. 2, p. 488–492.

WING, J. M. Computational Thinking. **Communications of the ACM**, v. 49, n. 3, p. 33–35, 2006.

WING, J. M. Computational Thinking and Thinking About Computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, The Royal Society, v. 366, n. 1881, p. 3717–3725, 2008.

YIM, J.-D.; SHAW, C. D. CALLY: the Cell-phone Robot with Affective Expressions. In: IEEE. **Human-Robot Interaction (HRI), 2009 4th ACM/IEEE International Conference on**. [S.l.], 2009. p. 319–320.

ZANETTI, H. A. et al. Uso de Robótica e Jogos Digitais como Sistema de Apoio ao Aprendizado. **Jornada de Atualização em Informática na Educação**, v. 1, n. 1, p. 142–161, 2013.

ZARAMELLA, V.; REIS, C. A. S.; SARMENTO, H. R. **Cordova plugin serial-audio**. 2014. Disponível em: <https://github.com/vzaramel/cordova-plugin-serialaudio>. Acesso em: 11 de agosto de 2014.

ZARAMELLA, V.; REIS, C. A. S.; SARMENTO, H. R. **Cordova plugin to generate DTMF tons**. 2014. Disponível em: <https://github.com/vzaramel/cordova-plugin-dtmf>. Acesso em: 13 de agosto de 2014.

APÊNDICE A – FORMULÁRIO - AVALIAÇÃO PREDITIVA DA USABILIDIADE

Recorrer à esta página para verificar os critérios de avaliação: http://www.psu.edu/webconference/Web2004/Materials/Heuristic.pdf

Para cada categoria devem ser verificados os problemas identificados e seus respectivos níveis de severidade.

0.Não concordo que seja um problema de usabilidade

1.Problema cosmético - corrigir se houver tempo extra

2.Problema pequeno - baixa prioridade na correção

3.Problema grave - alta prioridade na correção

4.Problema catastrófico - correção obrigatória para entrega do produto

Formulário:

1.(Load Time) Visibilidade do status do sistema:

IDEAL: As páginas normalmente carregam rapidamente (menos de 10 segundos) em uma conexão de 54kbps, devido a gráficos pequenos, boa compreensão de sons e gráficos e divisão apropriada do conteúdo. INCOMPLETO: Páginas normalmente demoram mais que 15 segundos para carregar devido a grandes gráficos, sons, etc.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

2.(Content) Compatibilidade do sistema com o mundo real:

IDEAL: O website tem propósito e tema bem definido que é mantido por todo o website. INCOMPLETO: O website não possui propósito e tema bem definido.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

3.(Content accuracy) Compatibilidade do sistema com o mundo real:

IDEAL: Toda a informação fornecida pelo designer no website é precisa e todos os requisitos da tarefa foram alcançados. INCOMPLETO: Há várias imprecisões no conteúdo fornecido pelo designer ou vários dos requisitos não foram alcançados.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

4.(Interest) Compatibilidade do sistema com o mundo real.

IDEAL: O designer conhece o público-alvo do website e desenvolve o material visando a atender da melhor forma às necessidades desse público. INCOMPLETO: O designer não leva em consideração o público-alvo do website.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

5.(Links) Consistência e padrões

IDEAL: Todos os links apontam para website de alta qualidade e atualizados. INCOMPLETO: Menos de 3/4 dos links apontam para website de alta qualidade e atualizados.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

6.(Spelling and Grammar) Consistência e padrões

IDEAL: Não há erros na escrita, pontuação ou gramática no website. INCOMPLETO: Há mais que 5 erros na escrita, pontuação ou gramática no website.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

7.(Navigation) Consistência e padrões

IDEAL: Links para navegação são claramente nomeados, consistentemente localizados, permitem que o leitor se mova de uma página para páginas relacionadas e levam o usuário para onde ele espera ir. O usuário não se perde. INCOMPLETO: Alguns links não levam o usuário para os website descritos. O usuário normalmente se sente perdido.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

8.(Compatibility) Prevenção de erros

IDEAL: O website foi testado em diferentes sistemas operacionais e diferentes versões de navegadores. INCOMPLETO: O website não foi testado ou somente suporta um navegador ou um sistema operacional.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

9.(Background) Estética e design minimalista

IDEAL: Background é muito atrativo, consistente entre as páginas, incrementa o tema ou propósito do website e não atrapalha a legibilidade. INCOMPLETO: Background atrapalha a legibilidade do website.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

10.(Color choices) Estética e design minimalista

IDEAL: Cores de background, fontes, links formam uma paleta de cores agradável, não atrapalham o conteúdo e são consistentes entres as páginas do website. INCOMPLETO: Cores de background, fontes, links tornam o conteúdo difícil de compreender ou distrai o leitor.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

11.(Layout) Estética e design minimalista

IDEAL: O website tem um layout atrativo e usável. É fácil de localizar todos os elementos importantes. Espaço em branco, elementos gráficos e alinhamento são utilizados para organizar o material. INCOMPLETO: As páginas possuem uma aparência desordenada ou confusa. É freqüentemente difícil localizar elementos importantes.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

12.(Fonts) Estética e design minimalista

IDEAL: Fontes são consistentes, fáceis de ler e os tamanhos variam conforme os títulos e texto. O uso de estilos de fonte(e.g., itálico, negrito, sublinhado) é consistente e melhora a legibilidade. INCOMPLETO: Uma grande variedade de fontes, estilos e tamanhos de fontes são utilizados.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

13.(Meta Tags) Help e documentação.

IDEAL: As meta tags utilizadas descrevem muito bem a compreensão do material incluído na página. Descrição, título, autor, palavras-chave, classificação, tipo de recurso, entre outros, refletem precisamente o conteúdo das páginas. INCOMPLETO: As meta tags estão faltando. O conteúdo da páginas não é refletido em nenhuma meta tag.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

14.(Copyright) Help e documentação.

IDEAL: Todo material utilizado de outras fontes é devidamente referenciado e citado. INCOMPLETO: Materiais de fontes externas não são propriamente documentados ou o material foi utilizado sem permissão da fonte.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

15.(Contact Information) Help e documentação.

IDEAL: Cada página contém informação de autoria, data de publicação/última data de edição. Várias páginas não contêm informação de autoria, data de publicação/última data de edição. INCOMPLETO: Várias páginas não contêm informação de autoria, data de publicação/última data de edição.

☐ Excelente

☐ Bom

☐ Satisfatório

☐ Precisa de melhorias

APÊNDICE B – QUESTIONÁRIO

B.1 PRÉ-CURSO

1.Qual o seu curso?

2.Qual a sua idade?

3.Qual o seu gênero?

- **o** Masculino
- **o** Feminino

4.Você teve contato com programação antes da realização deste curso?

- **o** Sim
- **o** Não

5.Se sua resposta é sim, a sua experiência com programação foi:

☐ Muito ruim ☐ Ruim ☐ Neutro ☐ Boa ☐ Muito boa

6.Se sua resposta é não, a sua expectativa em aprender a programar é:

☐ Muito ruim ☐ Ruim ☐ Neutro ☐ Boa ☐ Muito boa

7.Você utiliza computadores/smartphones para:

- **o** Acessar a internet
- **o** Criar programas de computador
- **o** Utilizar redes sociais
- **o** Jogar
- **o** Outros

8.Você possui smartphone? Quando foi comprado?

- **o** Não possuo
- **o** Sim. Comprado em menos de 6 meses.
- **o** Sim. Comprado em menos de 1 ano.
- **o** Sim. Comprado em mais de 1 ano.

9.Quais recursos o seu celular possui?

- **o** Câmera
- **o** GPS
- **o** Acelerômetro
- **o** Wi-Fi
- **o** Conector externo para fone de ouvido
- **o** Não possui nenhum dos recursos mencionados
- **o** Não possuo celular

10.A utilização de robôs para aprender a programar chama a minha atenção.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

11.Utilizar a o Coffee em sala será útil para que eu possa aprender a programar de maneira eficaz.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

12.Eu vejo utilidade em utilizar o Coffee para aprender a programar.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

13.Eu acredito que no final deste curso eu terei absorvido grande parte do conteúdo passado.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

14.Aprender a programar com robôs é intimidante para mim.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

15.Aprender a programar utilizando blocos me parece mais simples.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

16.Estou ansioso(a) para utilizar o Coffee.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

B.2 PÓS-CURSO

1.Qual a sua idade?

2.Qual o seu curso?

3.A maneira com que o Coffee foi utilizado para ensinar programação ajudou a manter minha atenção.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

4.Aprender a programar com um robô estimulou minha criatividade.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

5.Aprender a programar com blocos permitiu que eu me preocupasse mais em meu aprendizado.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

6.A diversidade de atividades possíveis com o Coffee chamou minha atenção.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

7.Eu consegui relacionar o aprendizado de programação com robôs às minhas experiências e expectativas.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

8.Utilizar o Coffee auxiliou a testar os conceitos ensinados.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

9.Eu pude entender um pouco mais dos conceitos de programação utilizando o Coffee.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

10.Os exercícios de programação eram muito difíceis.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

11.A utilização de blocos para programar tornou a solução dos problemas mais fácil.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

12.Aprender a programar e manipular o Coffee foi fácil.

☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

13. Aprender a programar com o Coffee era tão abstrato que foi difícil manter a atenção durante as aulas.

    ☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

14. A maneira com que o Coffee foi utilizado para ensinar a programar me deixou entediado/desestimulado.

    ☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

15. Pretendo fazer mais cursos de programação.

    ☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

16. Eu não gostaria de ter que fazer mais cursos relacionados à programação.

    ☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

17. Eu não gostaria de ter que fazer mais cursos relacionados à robótica.

    ☐ Discordo fortemente ☐ Discordo ☐ Neutro ☐ Concordo ☐ Concordo fortemente

APÊNDICE C – CRONOGRAMA DE ATIVIDADES

		Nov-13				Dec-13				Jan-14				Feb-14				Mar-14				Apr-14				May-14				Jun-14				Jul-14				Aug-14				Sep-14			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Tarefa	Duração (horas)																																											
Monografia	155																																											
Levantamento bibliográfico	40																																											
Justificativa, objetivo geral e objetivos específicos	10																																											
Metodologia	30																																											
Recursos de hardware e software	5																																											
Desenvolvimento	30																																											
Resultados das avaliações da ferramenta	30																																											
Resumo e Conclusão	10																																											
Engenharia de Software	40																																											
Requisitos	10																																											
Casos de uso	10																																											
Redes de PERT	5																																											
Viabilidade	5																																											
Análise de riscos	5																																											
Cronograma	5																																											
Bloco DTMF (Abordagem I)	55																																											
Estudo do hardware	5																																											
Estudo do software (Plugin)	5																																											
Desenvolvimento do hardware	25																																											
Integração hardware/software	10																																											
Teste de funcionamento do bloco	10																																											
Adaptador Audio-Serial (Abordagem II)	85																																											
Estudo do hardware	5																																											
Estudo do software (Plugin)	5																																											
Desenvolvimento do hardware	35																																											
Integração hardware/software	10																																											
Teste de funcionamento do bloco	30																																											

Figura 45: Cronograma de Gantt das Atividades Realizadas: Parte 1

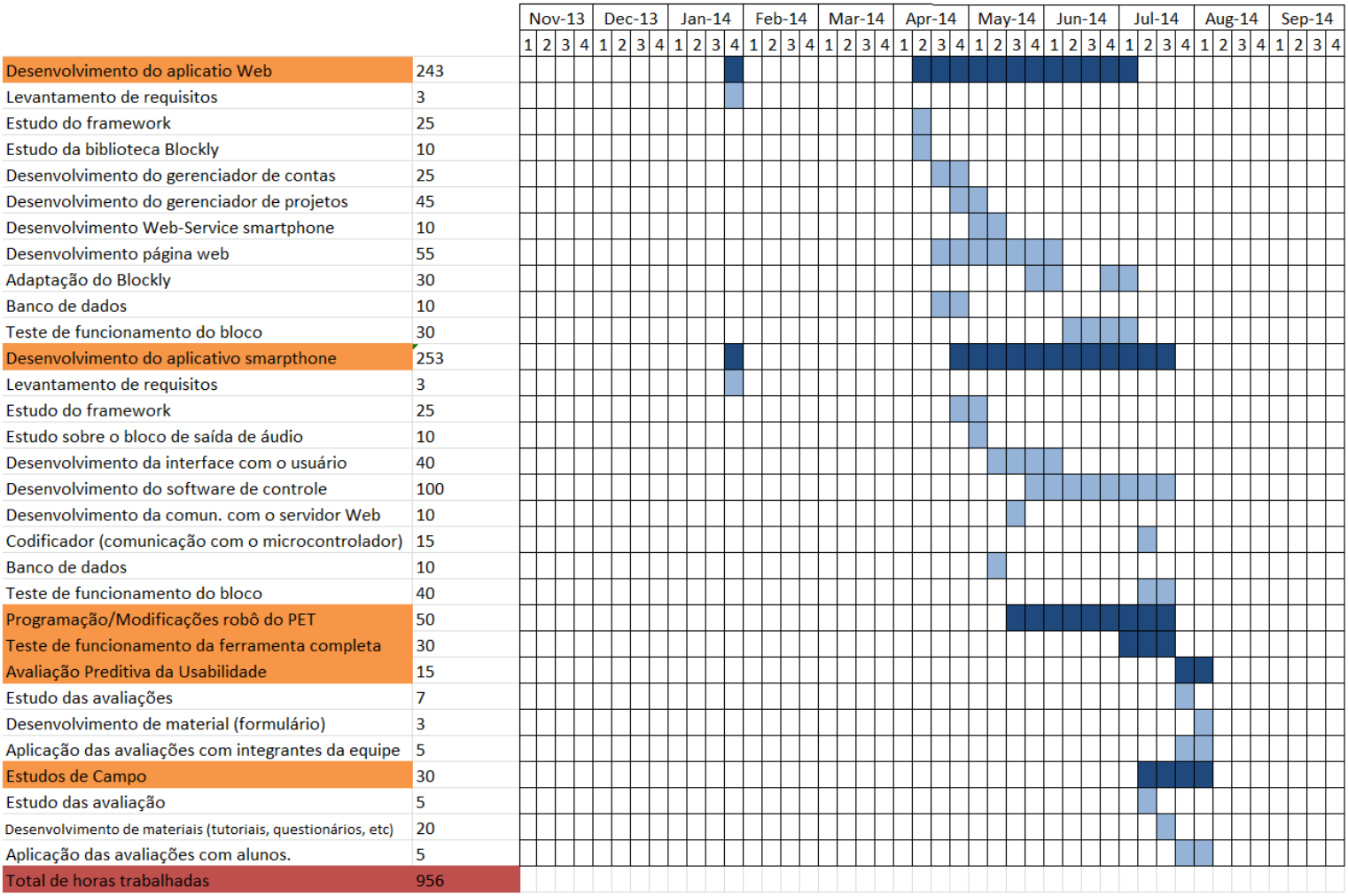

Figura 46: Cronograma de Gantt das Atividades Realizadas: Parte 2

ANEXO A – PLANO DE ENSINO COMPUTAÇÃO I

EMENTA	**CONTEÚDO**
1 Algoritmos Estruturados	1.1 Noções e definições básicas de algoritmos; 1.2 Estruturas simples (leitura, saída e atribuição); 1.3 Estruturas condicionais e estruturas de repetição; 1.4 Exemplos e aplicações.
2 Aspectos formais da linguagem de Programação.	2.1 Origem, conceitos e aspectos da linguagem; 2.2 Elementos básicos; 2.3 Tipos e definições de dados; 2.4 Dados numéricos; 2.5 Operadores aritméticos e relacionais; 2.6 Comando de atribuição; 2.7 Estrutura básica de um programa; 2.8 Comandos básicos de entrada e saída; 2.9 Biblioteca Matemáticas; 2.10 Expressões tipos e prioridade dos operadores; 2.11 Exemplos e aplicações.
3 Ambiente Operacional e Compiladores.	3.1 Descrição do ambiente operacional a ser utilizado; 3.2 Laboratórios, sistemas operacionais e compiladores; 3.3 Definições de regra para o desenvolvimento, construção e execução de programas.
4 Comandos da Linguagem de Programação.	4.1 Comandos de decisão; 4.2 Desvio incondicional;; 4.3 Comando de múltipla escolha; 4.4 Comandos de repetição; 4.5 Comando de interrupção e continuidade de laços; 4.6 Exemplos e aplicações.

5 Conjuntos e Matrizes numéricas.	5.1 Definições de arrays (tipos de dados numéricos); 5.2 Comandos de entrada e saida para arrays;; 5.3 Exemplos e aplicações.
6 Cadeia de caracteres.	6.1 Cadeia de caracteres; 6.2 Conceito e terminologia; 6.3 Operações básicas; 6.4 Funções de manipulação de cadeia de caracteres; 6.5 Exemplos e aplicações.
7 Subprogramas.	7.1 Regras para construções de subprogramas; 7.2 Tipos de variávies (local e global); 7.3 Passagem de parâmetros; 7.4 Funções com passagem de conjunto , matrizes e String; 7.5 Exemplos e aplicações.
8 Comandos da Linguagem de Programação.	8.1 Tipos de arquivos (texto e formatados); 8.2 Comandos para manipulação de arquivos; 8.3 Leitura e gravação de arquivos.

Disponível em: http://www.utfpr.edu.br/curitiba/cursos/licenciaturas/Ofertados-neste-Campus/matematica/planos-de-ensino/5o-periodo/if71a-computacao-1