UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS
**DEPARTAMENTO DE COMPUTAÇÃO**
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

# INTERFACE COM O AEDROMO (AMBIENTE EXPERIMENTAL E DIDÁTICO COM ROBÔS MÓVEIS) UTILIZANDO SISTEMA ANDROID

Nome: Lucas Salmen Roberto Alves RA: 925012

Orientador: Rene Pegoraro

BAURU - SP
JANEIRO/2014

LUCAS SALMEN ROBERTO ALVES

# *INTERFACE COM O AEDROMO (AMBIENTE EXPERIMENTAL E DIDÁTICO COM ROBÔS MÓVEIS) UTILIZANDO SISTEMA ANDROID*

Trabalho de Conclusão de Curso do Curso de Bacharelado em Ciência da Computação da Universidade Estadual Paulista "Júlio de Mesquita Filho", Faculdade de Ciências, campus Bauru.

Orientador: Rene Pegoraro

BAURU

JANEIRO/2014

Alves, Lucas Salmen Roberto.
Interface com o AEDROMO (Ambiente Experimental e didático com robôs móveis) utilizando sistema Android / Lucas Salmen Roberto Alves, 2014
43 f. : il.

Orientador: Rene Pegoraro

Monografia (Graduação)– Universidade Estadual Paulista. Faculdade de Ciências, Bauru, 2014

1. AEDROMO. 2. Interface homem-computador. 3. Android. 4. Robôs móveis. I. Universidade Estadual Paulista. Faculdade de Ciências. II. Título.

# DEDICATÓRIA

Dedico este projeto primeiramente à minha mãe, Olga Salmen Roberto, e minha avó, Olga Salmen, que sempre me apoiaram e estiveram do meu lado sempre que eu precisei, que me moldaram como pessoa, me educaram e me deram humildade para reconhecer meus próprios erros.

Também dedico às pessoas que me ajudaram nessa jornada da universidade, aos meus amigos, Alan Peixinho, Daniel F. S. Santos, Daniel Zuniga, Felipe Avelar, Filipe Araújo, Gustavo Souza, Henrique Arruda, Hiroaki Shibukawa, Jorge Henrique e Marcel Henrique, e aos seguintes professores que sempre me ofereceram ajuda e atenção quando tive dúvidas, Andréa Vianna, Celso Socorro, Humberto Ferasoli, Luttgardes de Oliveira, Rene Pegoraro e Simone Prado.

# RESUMO

O AEDROMO (Ambiente Experimental e Didático com Robôs Móveis) é um ambiente versátil, amigável e escalável que suporta uma ampla gama de experimentos. Nele existe uma área semelhante a uma mesa, onde os objetos podem interagir entre si, dentre eles robôs e outros objetos, podendo assim realizar inúmeras atividades.

Atualmente o AEDROMO conta com computadores clientes que interagem com o sistema através de uma interface, podendo assim realizar a comunicação entre o usuário e o AEDROMO.

Este projeto oferece suporte a criação de uma nova forma de interface para o AEDROMO, podendo assim ser utilizado por dispositivos que executam o sistema Android, este aplicativo desenvolvido neste projeto servirá de base para futuros trabalhos a partir desta nova interface.



Palavras-chave: AEDROMO, interface homem-computador, Android, robôs móveis.

# ABSTRACT

The AEDROMO (Experimental and Didactic Environment with Mobile Robots) is a versatile, user friendly and scalable environment that supports a wide range of experiments. In it there is an area that is similar to a desk where objects can interact with each other, including robots and other objects, and thus can perform numerous activities.

In it's current state, AEDROMO has client computers that interact with the system through an interface, and thus realize the communication between the user and AEDROMO.

This project offer support to create a new form of interface for AEDROMO and can therefore be used for devices running Android, the app developed in this project will serve as a basis for future work on this new interface.



Keywords: AEDROMO, human-computer interface, Android, mobile robots.

# LISTA DE FIGURAS

# LISTA DE SIGLAS

ADT (*Android Developer Tools*)

AEDROMO (*Ambiente Experimental e Didático com Robôs  Móveis*)

API (*Application programming interface*)

GISDI (*Grupo de Integração de Sistemas e Dispositivos Inteligentes*)

IDE (*Integrated development environment*)

IP (*Internet Protocol*)

JDT (*Java Development Tools*)

JRE (*Java Runtime Environment*)

LED (*Light-Emitting Diode*)

LEPEC (*Laboratório de Ensino, Pesquisa e Extensão em Computação*)

SDK (*Software development kit*)

UDP/IP(*User Data-gram Protocol over Internet Protocol*)

UFES (*Universidade Federal do Espírito Santo*)

UI (*User interface*)

UML (*Unified Modeling Language*)

USP  (*Universidade de São Paulo*)

# SUMÁRIO

# 1. INTRODUÇÃO

A robótica com o passar dos anos vem crescendo, devido ao fato da tecnologia se tornar mais disponível, e com isso possibilitando a criação de robôs com novas aptidões. E cada vez mais, é possível pensar nas facilidades que eles oferecem para a sociedade (Savall et al., 1999).

Uma importância da robótica se dá na educação. Atualmente, o ensino de matérias como computação, matemática e engenharia se torna complicado. As aulas teóricas não conseguem prender a atenção dos alunos, entretanto, as aulas práticas não sofrem do mesmo problema, e isto acaba sendo a vantagem do ensino de robótica, pois o que é aprendido em aula, pode ser utilizado para a fabricação e programação de robôs, e aquilo que não se passava de teoria, acaba tendo um sentido ao ser percebido pelos alunos que a teoria realmente tem um propósito (Avanzato, 2000).

No Departamento de Computação da Faculdade de Ciências da Unesp Bauru, há o AEDROMO (*Ambiente Experimental e Didático com Robôs Móveis*) que foi desenvolvido pelo GISDI (*Grupo de Integração de Sistemas e Dispositivos Inteligentes*). Este ambiente tem o objetivo de ser uma introdução ao ensino de robótica à alunos. Ele apresenta uma área parecida com uma mesa, onde sobre ela ficam os robôs e os outros objetos que irão interagir, dessa forma realizando tarefas e atividades. Algumas vantagens dele são: o baixo custo de manutenção; facilidade de implementação de códigos; entre outras (Ferasoli Filho et al., 2006).

O usuário realiza suas atividades em um computador-cliente que é o responsável por se comunicar com o AEDROMO, através deste computador é feita a programação das ações de um determinado robô na arena.

Os clientes do AEDROMO se conectam a ele via Internet. Estes computadores são utilizados pelos usuários para a interação com o ambiente. Os computadores executam programas que servem de interface para os usuários, facilitando assim o seu uso. O fato destes computadores ficarem em postos fixos limita a interatividade entre o usuário com o computador-cliente e com os elementos físicos sobre a arena.

Com o recente barateamento da produção (e em consequência facilidade de acesso) de dispositivos móveis[1] como celulares, vem surgindo projetos que utilizam dispositivos móveis

[1] Neste trabalho usa-se o termo dispositivo móvel referindo-se aos equipamentos que podem ser utilizados enquanto estão sendo carregados pelo usuário, como *smartphones* e *tablets*. Por outro lado, o termo dispositivo portátil faz referência aos dispositivos que constantemente são utilizados apoiados sobre um móvel.

como *smartphones* e *tablets* para controlar dispositivos automatizados como robôs (Aroca et al., 2012) (Goebel et al., 2011).

Estes novos dispositivos são pequenos computadores que usam sistemas operacionais diversos. Entre estes sistemas aparece o Android. Este é um sistema operacional portável, que tem requisitos muito menores do que um sistema operacional comum. Uma aplicabilidade comum do sistema é nos chamados *smartphones*, celulares com aplicativos, ou *tablets*.

AEDROMO é um ambiente que necessita de novas formas de interfaceamento com os usuários que pode se beneficiar do sistema operacional Android, portável e dinâmico, com uma interface que pode facilitar o controle dos robôs do ambiente remotamente. As aplicações do AEDROMO são muitas, e um dispositivo remoto com Android onde pessoas pudessem interagir simultaneamente aumentaria as possibilidades de utilização do AEDROMO.

## 1.1. PROBLEMA

Atualmente, uma dificuldade imposta aos usuários do AEDROMO é o fato dos programas clientes estarem em computadores em postos fixos que dificultam a interatividade entre o usuário e os elementos físicos sobre a arena que muitas vezes podem ser manipulados diretamente pelos usuários.

Esse problema seria reduzido ao se controlar um robô através de um dispositivo móvel, no caso, um controlador (*Smartphone* ou *Tablet*) com o sistema operacional Android, já que a capacidade de carregar estes dispositivos no ambiente é muito maior do que de computadores, sejam eles portáteis ou *notebooks*.

Outra possível vantagem seria a implementação de realidade aumentada no dispositivo, onde seria capaz de observar a arena e onde está cada um dos objetos, podendo assim programá-los diretamente através de toques na tela.

## 1.2. JUSTIFICATIVA

Os principais benefícios deste trabalho são basicamente:

I) A criação de novos modelos de interfaces aplicáveis ao AEDROMO, possibilitando assim uma maior gama de opções de desenvolvimento de interfaces indo além da regra que são os computadores padrões;
II) Permitir uma maior interatividade humano-*software* cliente-arena do AEDROMO, dessa forma utilizando o Android como dispositivo móvel para controle da interatividade, já que ele é menor que um computador tradicional, ou até mesmo um *notebook*, o usuário do AEDROMO pode interagir com a interface ao mesmo tempo que interage fisicamente com os objetos na arena;
III) Aumentar a portabilidade do sistema, transformando-o mais agradável e amigável ao usuário, este ponto enaltece o anterior, já que as possibilidades de desenvolvimento de interfaces no Android são ainda maiores.

## 1.3. OBJETIVOS

### *1.3.1. OBJETIVO GERAL*

Desenvolver um aplicativo base em Android que atuará como cliente do AEDROMO para servir de interface à realização das tarefas.

### *1.3.2. OBJETIVOS ESPECÍFICOS*

Elaborar uma comunicação entre o aplicativo e o computador-servidor. Identificar algumas interfaces Homem-Máquina para programação com o AEDROMO utilizando dispositivos móveis. Desenvolver rotinas clientes em Android para o AEDROMO proporcionando novas interfaces Homem-Máquina.

## 1.4. MÉTODO DE PESQUISA

Para este projeto, será utilizado o AEDROMO, com todos os seus equipamentos, disponíveis aos alunos nas dependências do departamento de computação, ele já conta com uma API estabelecida ao lado do servidor, além de um simulador que pode ser usado para teste antes

da execução no ambiente real, este simulador também será utilizado no desenvolvimento do projeto. Será utilizado também um computador disponível no LEPEC com a IDE Eclipse em conjunto com o SDK do Android para o desenvolvimento do aplicativo no dispositivo móvel. Para vias de testes, será necessário um dispositivo com Android, no caso um *smartphone* com sistema na versão 2.3.6 ou superior e também um simulador do dispositivo móvel, que será utilizado no desenvolvimento e também para substituição ao dispositivo real, este simulador já é disponibilizado junto ao SDK do Android, com todas as versões disponíveis.

Para se realizar o levantamento bibliográfico, utilizou-se buscas na Internet sobre artigos relacionados a este projeto, para pesquisas sobre APIs, serão feitas diretamente aos programas do AEDROMO e à documentações existentes.

## 2. FUNDAMENTAÇÃO TEÓRICA

Atualmente, o ensino de matérias como computação, matemática e engenharia se torna complicado. Os estudantes não tem tanto interesse por essas matérias teóricas quanto pelas práticas, no caso, essas matérias acabam sofrendo desvantagem por serem apresentadas apenas de uma forma muito metódica, entretanto, o ensino de robótica pode ser empregado de maneira prática, e esta matéria faz uso daquela parte teórica(Avanzato, 2000).

Hoje, no Departamento de Computação da Faculdade de Ciências da Unesp Bauru, há um interesse nesta área de robótica.

Um dos resultados deste interesse é o AEDROMO e foi desenvolvido pelo GISDI. As grandes vantagens deste ambiente são: i) o baixo custo de manutenção; ii) facilidade de implementação de códigos; iii) aprendizagem no conceito de máquinas, programação e inteligência artificial; iv) diversão e entretenimento; v) pesquisa e desenvolvimento de projetos (Ferasoli Filho et al., 2006).

Como pode ser observado na figura 1, o ambiente é composto de uma área onde os robôs interagem, uma arena ou área de trabalho, um computador-servidor que envia sinais aos robôs, uma câmera que fica acima do campo para identificar onde cada objeto está na arena e por fim um transmissor que é responsável pelo comunicação computador-robô.

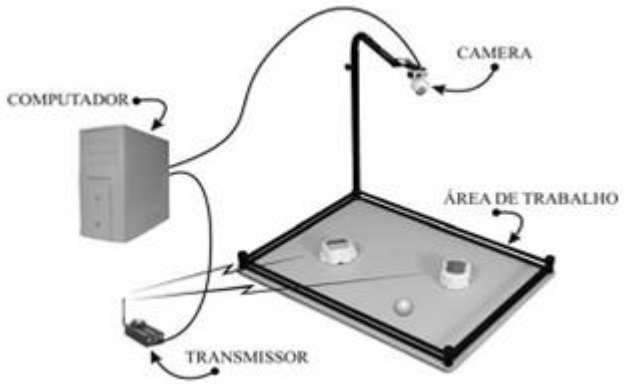


Figura 1. AEDROMO. Fonte: Ferasoli Filho et al. (2006).

Uma outra parte importante do projeto é o computador-cliente que faz a comunicação com o computador-servidor, esse computador-cliente é o responsável por programar as ações de um robô, a comunicação cliente-servidor é feita através de UDP/IP (*User Datagram Protocol over Internet Protocol*). O computador-servidor é responsável pela comunicação com os robôs e da implementação do sistema de visão computacional em conjunto à câmera (Alves et al., 2011).

A arquitetura em detalhe pode ser vista na figura 2.

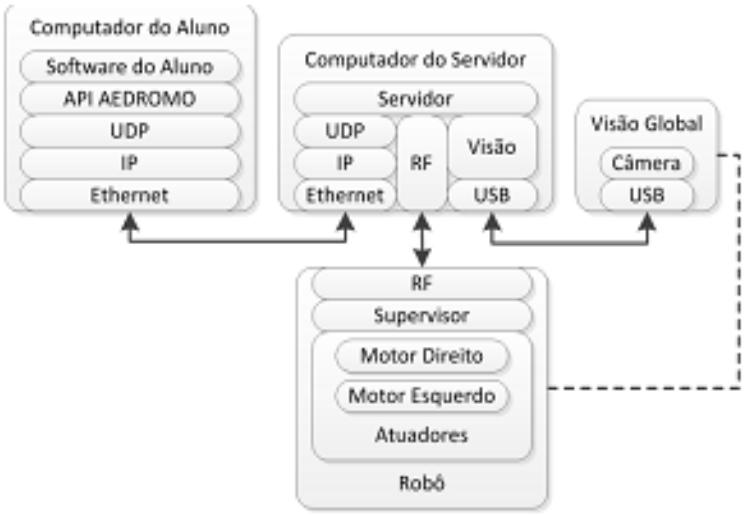


Figura 2. Arquitetura do AEDROMO. Fonte: Alves et al. (2011).

As imagens são obtidas através da câmera e são enviadas para o computador servidor, nele, essas imagens são adquiridas e processadas. Este processamento determina a posição cartesiana (x, y) de cada objeto (robôs, bolas, blocos etc) no campo e, exclusivamente para os robôs, os seus ângulos de orientação (Alves et al., 2011).

O grande objetivo do AEDROMO é o direcionamento didático que ele apresenta, diversas atividades podem ser realizadas neste ambiente (Ferasoli Filho et al., 2006), tais como:

I) Coletar: na área de trabalho é colocado algumas bolinhas de cores diferentes. Nesta atividade, o usuário tem como objetivo coletar certas bolinhas de uma determinada cor em um espaço pré-determinado, o tempo é o grande obstáculo nesse jogo. Pode também jogar com duas pessoas, assim o que coletar mais rápido vence.

II) Labirinto: nesta atividade a área de trabalho, se transforma em um labirinto, onde o objetivo é fazer o robô chegar até o final, pelo caminho mais curto ou com o melhor tempo.

III) Tênis: o campo é dividido em duas partes, onde um robô ocupa cada uma das áreas. O objetivo é empurrar a bola através do campo do adversário até chegar na linha de fundo, mas sem o robô atacante ultrapassar a linha, enquanto o defensor deve evitar isso.

IV) Arrastando blocos: essa atividade é semelhante a primeira, porém nesse caso, são blocos, com pesos diferentes, é possível adicionar blocos com pesos maiores para que seja necessário o trabalho de dois robôs ao mesmo tempo para empurrá-lo.

Os clientes do AEDROMO, atualmente, são computadores de mesa ou portáteis que

executam programas que servem de interface com os usuários, para que estes desenvolvam as suas atividades. Esses computadores, sempre conectados através da Internet, até mesmo os *notebooks*, funcionam em postos fixos próximos ou remotamente, o que limita a interatividade entre o usuário com o computador-cliente e os elementos físicos sobre a arena que muitas vezes podem ser manipulados diretamente pelos usuários.

Por outro lado, com o maior acesso a dispositivos móveis como celulares, vem surgindo iniciativas que empregam dispositivos móveis como *smartphones* e *tablets* para controlar dispositivos automatizados como robôs (Aroca et al., 2012)(Goebel et al., 2011)(Chen et al., 2010)(Lopes, 2013)(Sethi, 2013)(Sugiura et al., 2009).

Inclusive algumas dessas ideias que foram aplicadas em outros projetos podem ser implementadas no próprio AEDROMO em conjunto com um *smartphone*, porém a abordagem seria diferente, ainda assim a interface a ser implementada no dispositivo e a interação homem-máquina poderia ser aplicada (Goebel et al., 2011)(Sethi, 2013)(Sugiura et al., 2009). Mais adiante estas interfaces serão apresentadas.

Estes aparelhos móveis tem aparecido comercialmente com seus preços cada vez menores, popularizando-os. São pequenos computadores que usam sistemas operacionais também diversos, entre estes sistemas aparece o Android que pode ser programado em Java em dispositivos com preços acessíveis. Este é um sistema operacional portável, que tem requisitos muito menores do que um sistema operacional comum.

A estrutura do sistema Android é dividida em quatro camadas: camada de aplicação, camada do framework da aplicação, camada do sistema de execução e camada do núcleo Linux. O Android executa uma máquina virtual Java sob a camada deste núcleo (Xie et al., 2012). Esta estrutura pode ser vista na figura 3.

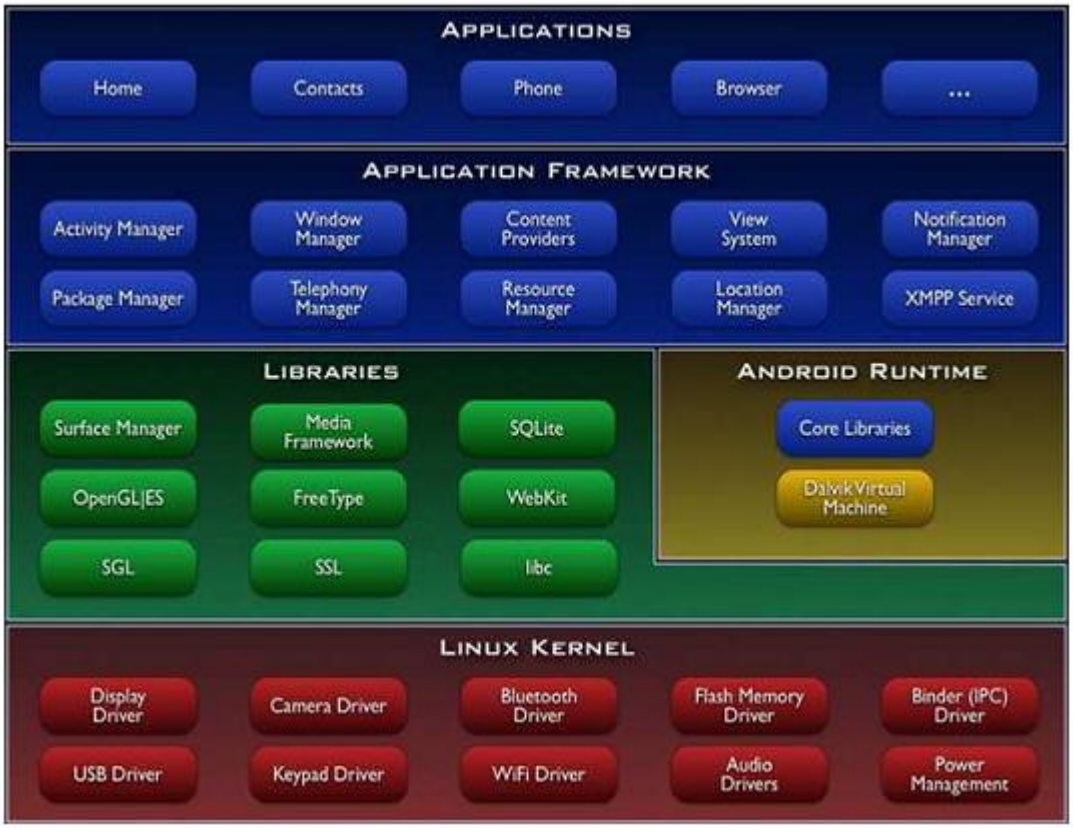


Figura 3. Estrutura do sistema Android. Fonte: Xie et al. (2012).

Uma aplicabilidade comum do sistema é nos chamados *smartphones*, celulares com aplicativos, ou *tablets*. Os aplicativos são programas para diversas finalidades, eles podem ser criados por usuários utilizando Java junto a IDE (*Integrated development environment*) Eclipse com o SDK (*Software development kit*) do Android, mesmo assim é possível criar códigos em outras linguagens como Python ou C, esta é uma vantagem de se utilizar um núcleo Linux (Aroca et al., 2012). Detalhadamente, cada aplicação apresenta sua própria máquina virtual Dalvik, que é um *software* que executa os aplicativos do Android, o que ela faz é converter arquivos do formato de máquina virtual Java, para o formato Dalvik, que é o *software* que todo dispositivo Android, contém. Esta máquina virtual reside no núcleo do Linux através do processo de gerenciamento. A Dalvik suporta a interface nativa Java (JNI – Java Native Interface) no modo de programa, as aplicações Android podem chamar a biblioteca compartilhada de desenvolvimento de C/C++ através da JNI para executar um método de programação "Java + C" (Xie et al., 2012), podendo assim programar com a linguagem C. A API (*Application Programming Interface*) do Android é completa e utiliza os inúmeros sensores que compõem o dispositivo.

Os *smartphones* atuais tem uma variedade de sensores que podem ser explorados para

robôs. Por exemplo, muitos tem câmeras, *wi-fi*, *bluetooth*, alto-falantes e microfone (Aroca et al., 2012). Outro aspecto interessante do Android é a sua UI (*User interface*), a interface do usuário, ela é toda preparada para toque, ou seja, tudo pode ser controlado com o dedo, isso facilita ao utilizar o sistema.

No último trimestre de 2012 o Android era o sistema operacional líder, em participação de mercado, com 70,1%, para medidas de comparação, o segundo lugar é da Apple, com o seu sistema iOS, com 21% (IDC, 2013).

Os dispositivos móveis, entre eles os que usam o Android, oferecem ferramentas para a criação de novas interfaces, a seguir são citados alguns trabalhos onde se nota a importância das novas interfaces utilizando dispositivos móveis.

Uma interface, o mascote eletrônico, foi idealizada pela própria Unesp, em conjunto com a USP de São Carlos e com a UFES (Universidade Federal do Espírito Santo) e se trata de pequenos robôs com a capacidade de interagir com crianças portadoras de deficiência, o objetivo deles é servirem como companheiros para as crianças, realizando tarefas e interagindo com ela. Quando o robô não está próximo da criança, ele age como um animal de estimação de maneira randômica (Lopes, 2013). No último protótipo criado pelo grupo, visto na figura 4, o robô apresenta um *smartphone*, onde este é responsável por aumentar a interatividade com a criança, essa interatividade é aumentada pelo fato, de que na tela, uma “carinha” se comunica com a criança, mudando a expressão facial na tela e proferindo frases como “Que legal”, “Que susto” e “Cansei” (Lopes, 2013). Coisa que com um robô, feito de LEDs e rodas, não seria possível obter, e assim, o robô com a “carinha” se torna muito mais amigável para a criança, conquistando um dos principais objetivos, que é a atenção e a felicidade da criança.


Figura 4. Último protótipo criado pelo grupo de pesquisa. Fonte: Lopes. (2013).

Essas interfaces não utilizam apenas Android, em um caso mais específico, o iLabArm e o iLabBot são projetos do Dr. Vikram Kapila, professor de engenharia mecânica no Instituto Politécnico da Universidade de Nova Iorque, este projeto faz uso do smartphone, no caso, um iPhone ou iPad, para controlar um braço robótico a distância, a comunicação é feita através da rede Wi-Fi, o controle sobre o robô é realizado utilizando os sensores de acelerômetro do smartphone, e com o gesto de "pinça" na tela de toque do smartphone, o robô reproduz o mesmo movimento na extremidade do braço, para poder pegar objetos (Sethi, 2013). O controle realizado com o acelerômetro do smartphone é exemplificado na figura 5.

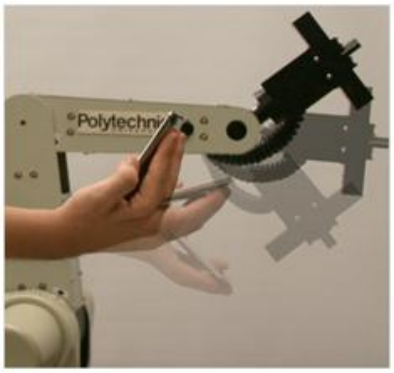

Figura 5. iLabArm sendo controlado pelo smartphone. Fonte: Jared Alan Frank e David Lopez. (2011). apud Sethi. (2013).

Uma outra interface interessante que faz uso do sistema iOS é o projeto Walky, representado na figura 6, um projeto criado na Universidade de Keio no Japão. Como no caso do iLabArm, o objetivo também é movimentar um robô remotamente através do smartphone, porém no Walky, o sensor mais importante utilizado é a própria tela de toque, através de gestos intuitivos, o robô bípede os reproduz (Sugiura et al., 2009).

Figura 6. Operação do robô bípede utilizando um gesto para a reprodução do movimento. Fonte: Sugiura et al. (2009).

A grande vantagem da utilização do smartphone é que a interface é muito mais natural do que com controles ou botões (Sugiura et al., 2009). Alguns movimentos que podem ser reproduzidos pelo robô através dos gestos na tela do smartphone podem ser vistos na figura 7.

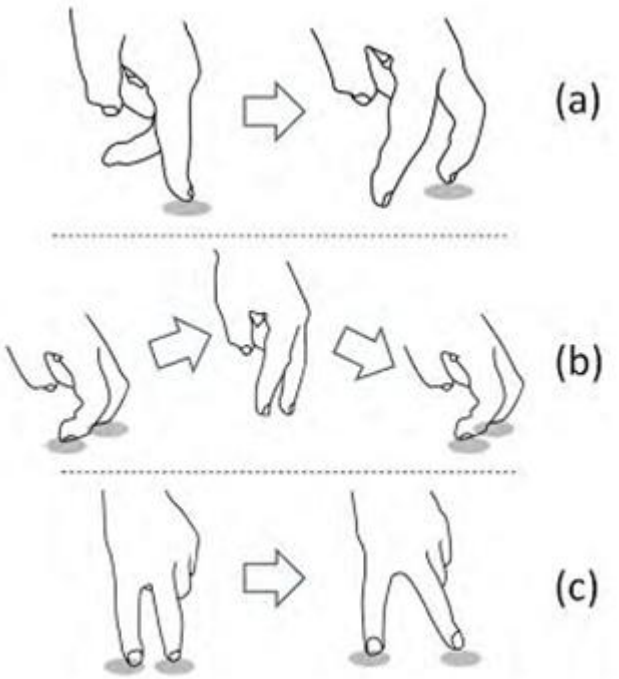


Figura 7. Gestos de entrada para o robô. (A) movimento de andar para frente e para trás; (B) movimento de pulo; (C) passos laterais. Fonte: Sugiura et al. (2009).

Além dessas interfaces, pode-se citar também o projeto de controle de robôs LEGO Mindstorms NXT utilizando a Plataforma Android (Goebel et al., 2011), que propõe o uso do Android para exemplo de controle sobre os robôs didáticos da LEGO, pelo fato da plataforma apresentar diversos sensores e facilidade de implementação, e ser de grande utilidade para cursos de robótica.

É importante citar sobre o aspecto da interface, já que inúmeras situações utilizam de diversas interfaces para controle, como visto. No caso deste trabalho a importância se dá devido à maior interação humano-software e cliente-arena do AEDROMO e para aumentar a portabilidade do sistema, transformando-o mais agradável e amigável ao usuário, além de possibilitar uma entrada de dados mais intuitiva, já que como visto no exemplo do Walky e do iLabArm, o smartphone, com seus inúmeros sensores, facilita essa interatividade.

O AEDROMO é um ambiente que necessita de novas formas de interfaceamento com os usuários que pode se beneficiar do sistema operacional Android, portável e dinâmico, com uma ótima interface com seu manuseio através do dedo que poderá facilitar mais o controle remoto dos robôs do ambiente.

As possibilidades de aplicações do AEDROMO são enormes, e um dispositivo remoto com Android onde pessoas pudessem interagir simultaneamente só aumentaria esse leque de possibilidades.

# 3. DESENVOLVIMENTO

Para o desenvolvimento prático do aplicativo e testes realizados com o simulador do AEDROMO, foi utilizado o Eclipse com o SDK do Android, este pacote é mais conhecido como ADT (Android Developer Tools), e pode ser feito o download através do próprio site da Google, visto na figura 8, ele apresenta todas as ferramentas necessárias para criar um aplicativo de Android. Vale ressaltar que esta versão do Eclipse possui também o JDT (Java Development Tools) que são as ferramentas necessárias para criar um programa em Java, já que os códigos de Android também são escritos em linguagem Java, com algumas funções específicas para um melhor aproveitamento do sistema.

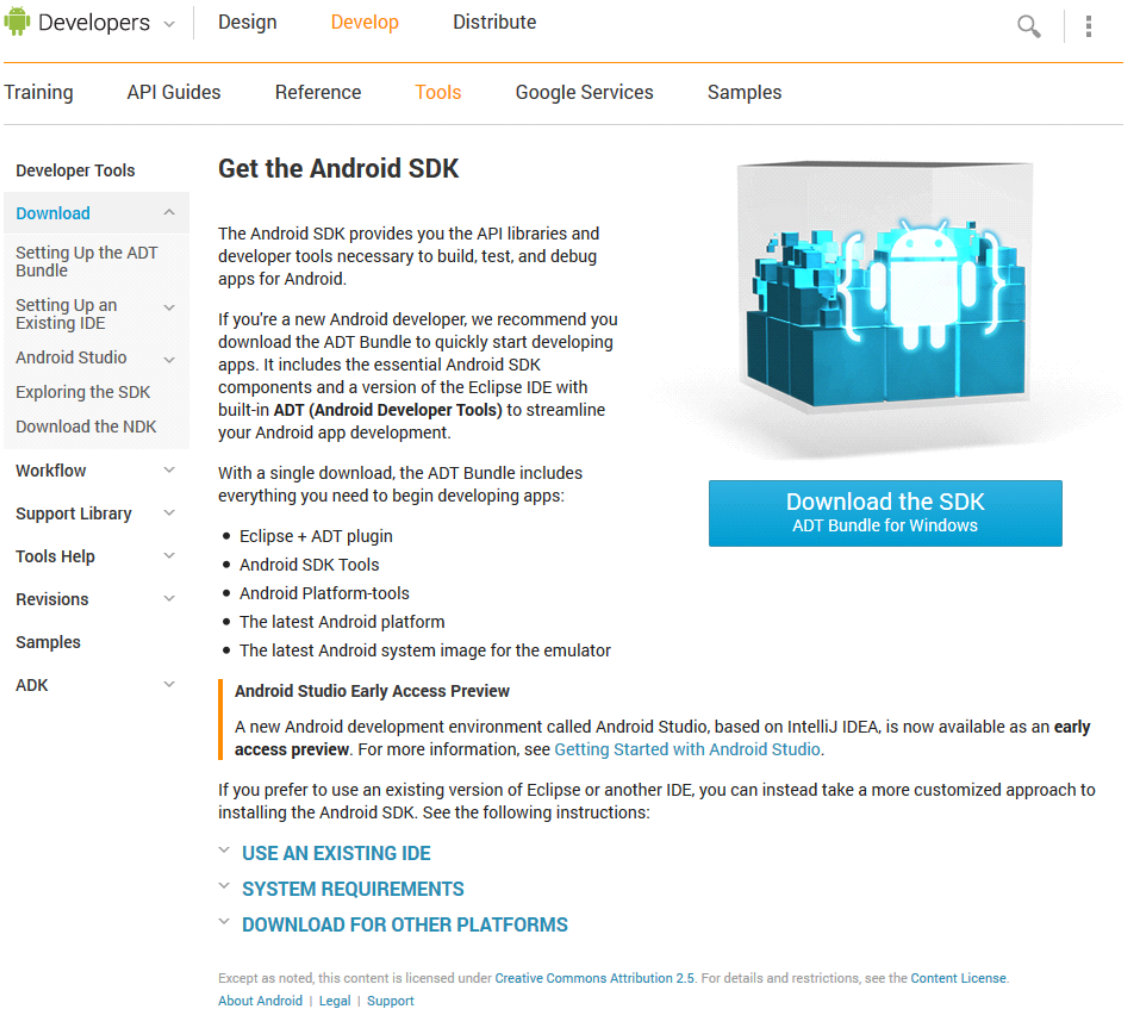


Figura 8. Site da Google para download do ADT do Android com o Eclipse. Extraído de: http://developer.android.com/sdk/index.html; Acesso em 24/12/2013

No primeiro momento do desenvolvimento foi feito um levantamento das funções e classes do simulador do AEDROMO, como o próprio é escrito em Java, foi simples utilizá-lo, para isso foi necessário apenas importar o projeto para o eclipse. Uma particularidade deste simulador é o fato dele utilizar o Java 3D, algo que não vem com o Eclipse por padrão.

Para a instalação desta biblioteca é necessário entrar no site da *oracle* e fazer o *download* da versão 1.5.1 do Java 3D, instalá-la e se certificar que o caminho da pasta é o mesmo do JRE que o eclipse está fazendo uso, caso isso não ocorra, é necessário mover os arquivos j3dcore-d3d.dll, j3dcore-ogl.dll, j3dcore-ogl-chk.dll e j3dcore-ogl-cg.dll para a pasta bin do JRE, como visto na figura 9. Esses arquivos do java3d tem a extensão "dll", pois o sistema operacional utilizado foi o windows, caso este fosse o linux a extensão seria "so".

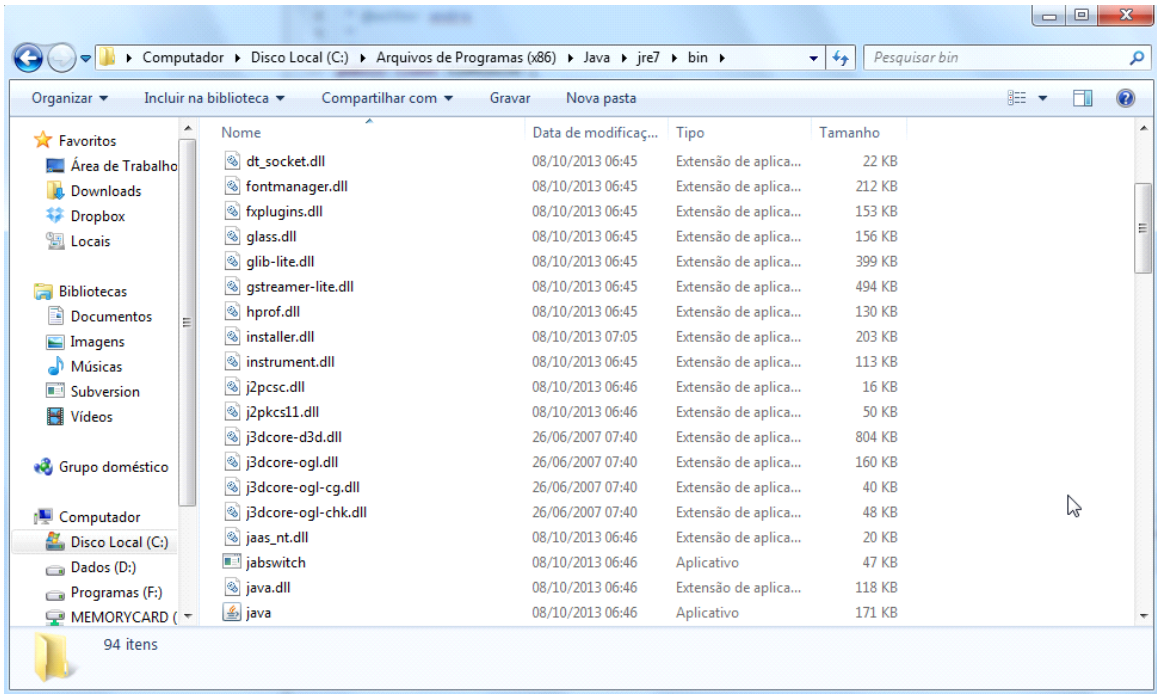


Figura 9. Arquivos DLL do Java3d.

Uma outra particularidade é a respeito do próprio JRE, a versão que vem no pacote do Eclipse com o ADT é a 6, enquanto que a versão utilizada pelo simulador é a 7, portanto há uma incompatibilidade entre as versões, e é necessário instalar a nova versão do JRE e depois colocar no path do Eclipse para que ele utilize essa nova versão ao invés da antiga da própria IDE, esse passo é visto na figura 10.

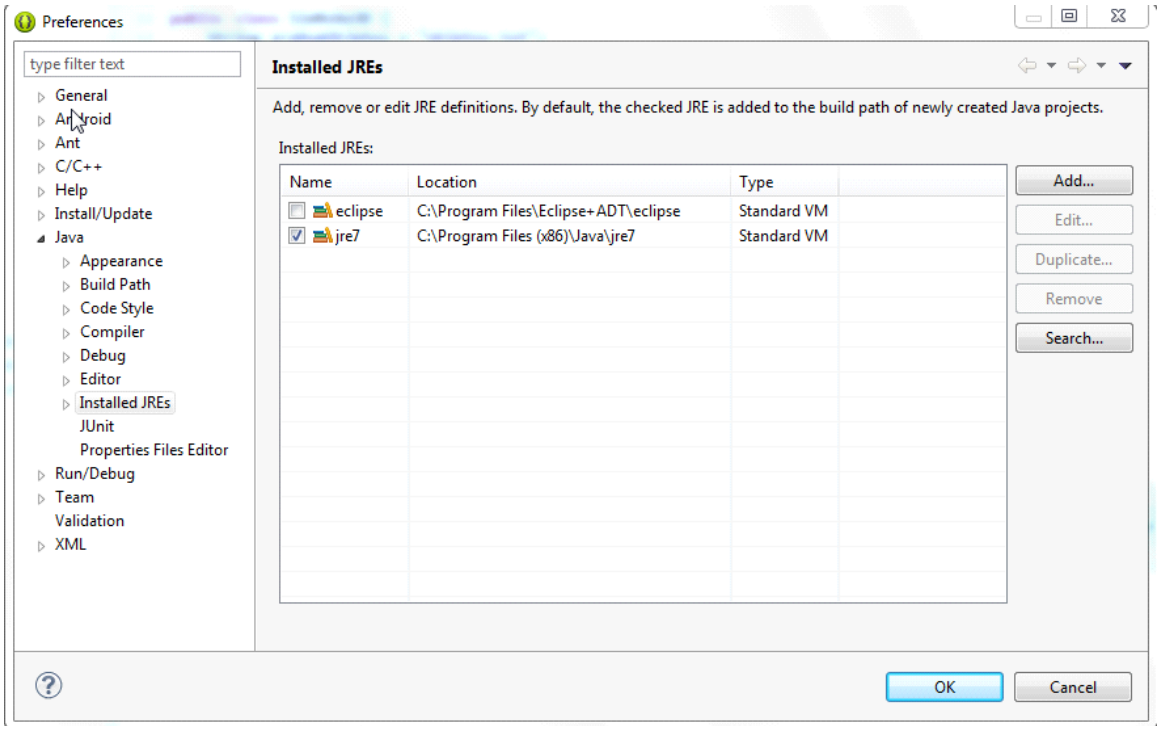


Figura 10. Alterando o Path do JRE do Eclipse para o JRE 7.

O AEDROMO utiliza um esquema de troca de informações por *sockets* para poder enviar as informações, neste pacote contêm o IP, o porto e os bytes de comando. Estes *sockets* são enviados através de conexão UDP.

O IP a ser colocado deve ser o do computador servidor. Duas situações podem vir a ocorrer, a primeira onde este computador está na mesma rede que o dispositivo Android, neste caso o IP deve ser o local da rede interna, no caso de estarem conectados em redes distintas, deve ser conectado pelo IP externo do computador servidor, e não o IP da rede local.

O porto é usado para determinar qual robô será movimentado, caso seja o 7879 é o robô 1, caso seja o porto 7878 é o robô 2. Assim é bem simples de se determinar qual robô movimentar.

Os bytes de comando são utilizados para ditar qual o comando que o robô deve executar, cada um desses bytes são pré-determinados e cada um corresponde a um determinado movimento, por exemplo, ir para frente, ir para trás, virar para um determinado lado, cada um desses movimentos é um passo.

Como são enviados dez pacotes por segundo, a verificação de onde o robô está é feita através de um loop que fica rodando até a verificação da coordenada pretendida em x e y, com o estado atual do robô. A estrutura do pacote enviado pode ser visto na figura 11.

```
void enviaComando(int cmd,int numPorto) {
    try {
        byte b[] = new byte[1];
        b[0] = (byte) cmd;
        DatagramPacket packet;
        packet = new DatagramPacket(b, b.length,
                InetAddress.getByName(numIP), numPorto);
        socket.send(packet);
    } catch (UnknownHostException e) {
        e.printStackTrace();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figura 11. Estrutura do Datagrama.

Este simulador do AEDROMO basicamente consiste em dois projetos, um mais complexo que faz a parte de servidor, e outro mais simples que é o cliente.

O projeto do servidor abre uma janela do *Java 3D*, vista na figura 12, onde se encontra uma simulação do AEDROMO, nela contêm a mesa, os dois robôs e três bolas para os robôs possam interagir. O campo original tem um comprimento de 80 cm e largura de 60 cm, o campo virtual o representa da mesma maneira, de tal forma que tudo seja proporcional. Uma peculiaridade deste campo é o fato dele ser cortado ao meio, no caso, cada robô tem seu próprio ponto 0,0, isso significa que pedir para um robô ir para o ponto 40,10 não significa que esse ponto será o mesmo para o outro robô, já que cada um tem sua própria origem, isso é feito para facilitar o uso do robô quando utilizado por mais de 1 pessoa. Por exemplo, quando uma pessoa está movimentando o robô 2, ele não precisa ficar se preocupando em passar parâmetros grandes como 74,49, podendo passar o ponto 6,11 que é o complemento daquela posição, neste caso, o ponto 6,11 é o 74,49 para o robô 1.

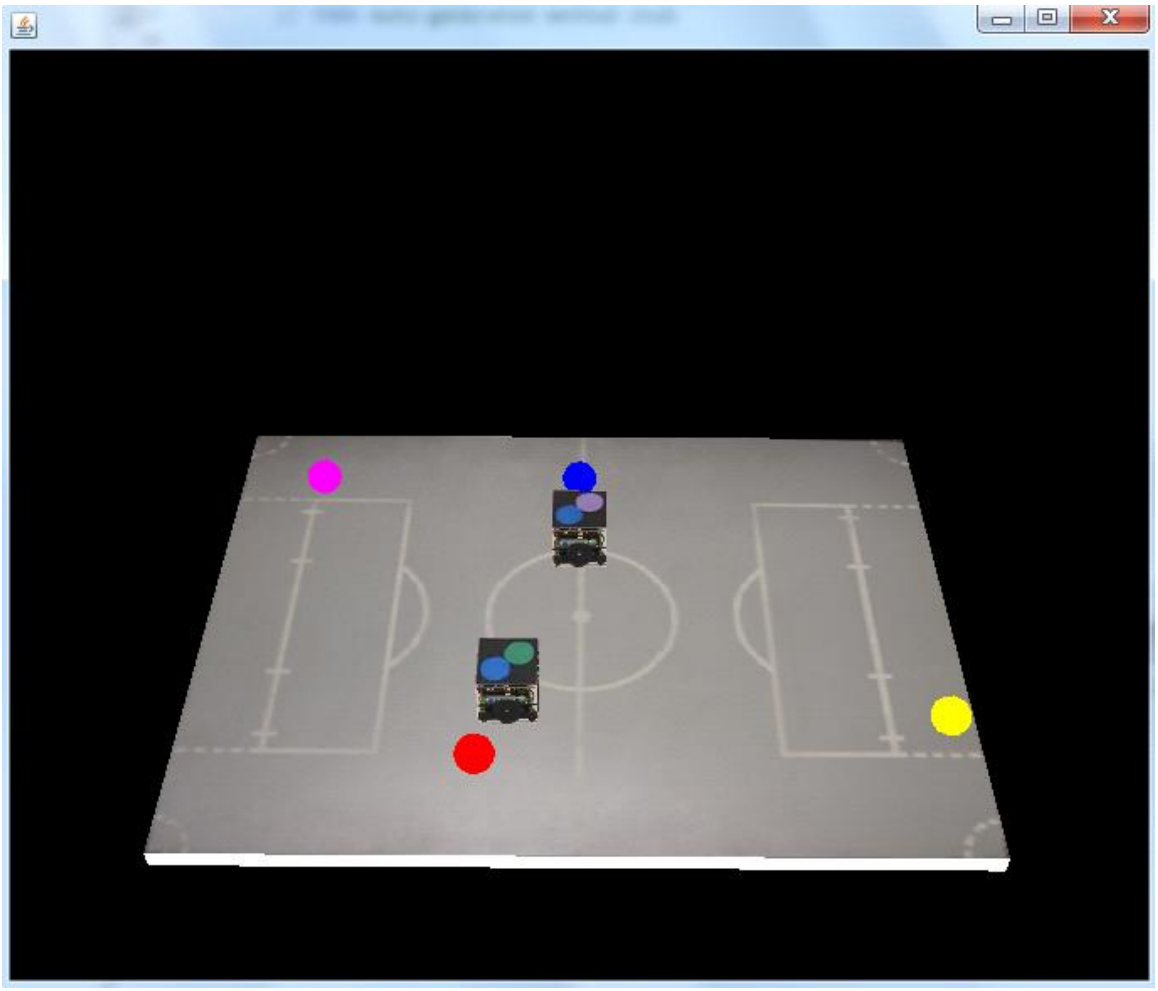

Figura 12. Janela do Simulador.

O servidor é composto de 11 classes, visto na figura 13. Algumas classes são básicas como a de Colisão, utilizada para o tratamento de colisões entre os objetos; a classe da Bola que trata da proporção e do desenho da mesma; a classe EstadoRobos que cria as variáveis que serão utilizadas para descrever o estado do robô, como o ponto x, y e o ângulo, além das variáveis de estado anterior e posterior; a classe EstadoObjetos que tem o mesmo propósito que a anterior. Outra classe básica é a que trata das ações do mouse, a EscutaMouse. Além de classes de constantes físicas, como a CTE e a Cor, que apresentam os valores constantes úteis para o resto do programa, e do AEDROMO, que é a SimRobo3D, que é a classe principal e instancia a arena 3D.

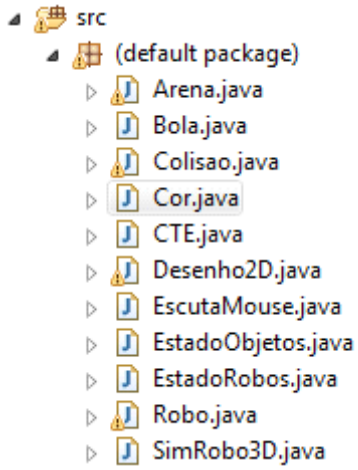


Figura 13. Classes do Servidor.

O servidor ainda conta com outras classes mais complexas e importantes. No caso a Desenho2D, que monta a arena em 2D, no caso, verificando as colisões entre os objetos, entre o robô e as bolas, entre as paredes do campo etc, e adicionando todos os parâmetros para a física do mundo criado. Outra classe importante é da Arena propriamente dita. Esta classe é responsável por colocar os objetos na arena, colocar as texturas nos próprios, fazer a atualização do desenho a cada passo dos objetos, além de tratar os bytes para a comunicação com o servidor, estes bytes são tratados da mesma forma que no servidor original, com a única diferença de ser java e não C. Por fim, a última importante classe é a Robo, que é a responsável geral pelo robô, além de fazer o tratamento da movimentação do robô.

Algumas partes tiveram que ser ajustadas no simulador, originalmente ele estava com as coordenadas trocadas, portanto os pontos de origem não estavam no padrão do espaço cartesiano.

A parte do cliente é mais simples, e tem uma classe responsável por fazer uma conexão com o projeto do servidor e mandar um comando pré-determinado através do uso do *socket* do java, e assim poder movimentar o robô até o lugar desejado. Este projeto do cliente por se tratar de um exemplo é uma aplicação base para outras atividades e não apresenta interface, sendo assim só é possível determinar o local de destino do robô através da inserção no próprio código.

No caso, o aplicativo de Android visa criar uma interface base para outros projetos com o intuito de utilizar o Android para controle dos robôs na Arena, portanto, o essencial é ser possível controlá-los através de um dispositivo com o sistema Android e fazer isso remotamente, adicionando uma maior interatividade homem-arena ao projeto, já que dessa forma é possível ficar o dispositivo Android em uma mão e interagir com os objetos na outra, estando ao lado da

arena.

O primeiro passo foi criar um roteiro para o projeto do aplicativo, com algumas regras básicas, a principal como sendo um aplicativo disponível para as mais diversas versões disponíveis do Android. O Android conta com um problema referente a fragmentação de versões (Business Insider 2014), como este sistema é aberto, a atualização das versões dependem das empresas que produzem os seus dispositivos próprios e das operadoras de telefonia. Dessa forma acontece o fenômeno da fragmentação, onde existem muitos *smartphones* com versões defasadas do Android, enquanto outros com as versões mais atualizadas. Outra resolução importante é que aplicativo fosse simples, ou seja, sem muitos comandos complexos para poder facilitar o entendimento e a consequente utilização do mesmo.

Tendo como parâmetro esse objetivo de criar um aplicativo abrangente para as mais diversas versões do Android, foi decidido que ele abrangeria desde a API 8 até a API 17, sendo assim possível executá-lo na versão 2.2 (referente ao SDK 8) até a versão 4.2 (referente ao SDK 17). Dessa forma, o aplicativo poderia ser executado em uma grande parcela de dispositivos. Este ponto é importante pois como este é um aplicativo meramente acadêmico, não é esperado que os dispositivos Android sejam muito avançados ou potentes, portanto é de suma importância essa larga variedade de versões disponíveis para o aplicativo.

O primeiro grande problema foi conseguir essa integração de disponibilidade entre as versões. Em um primeiro momento a parte visual do programa seria dirigida através de abas que ficariam na parte de baixo da tela, podendo alterar as telas através do toque, esta ideia pode ser vista na figura 14.

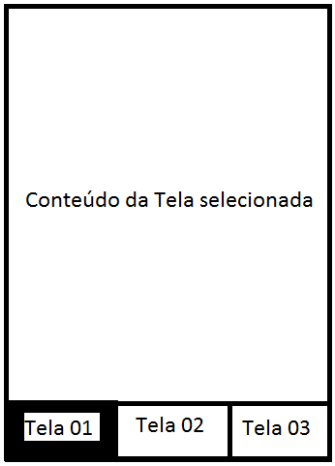


Figura 14. Ideia de tela com abas.

Isso seria feito através da utilização das classes "TabActivity", implementadas pelo próprio SDK do Android, este sempre foi um recurso comum ao Android em suas primeiras versões, sendo que uma enorme quantidade de aplicativos utilizava essa navegação por abas para o Android.

Neste caso, tudo é controlado por um controlador das abas, ele que as cria chamando outras classes que também são herdeiras da TabActivity.

Infelizmente essa abordagem não foi possível, pois a classe "TabActivity" ficou obsoleta com a chegada da API 13. A recomendação é que se utilize algo chamado de "Fragments", porém o aplicativo ainda assim iria parecer algo antigo já que essa não é mais a interface padrão dos novos aplicativos de Android.

Foi escolhido então utilizar um layout mais atual, no caso, fazendo uso da "ActionBar", a diferença é que nesse caso a interface é muito mais discreta e parece fazer parte do aplicativo, diferente do que acontecia com as abas na parte debaixo do aplicativo. A "ActionBar" básica pode ser vista na figura 15.

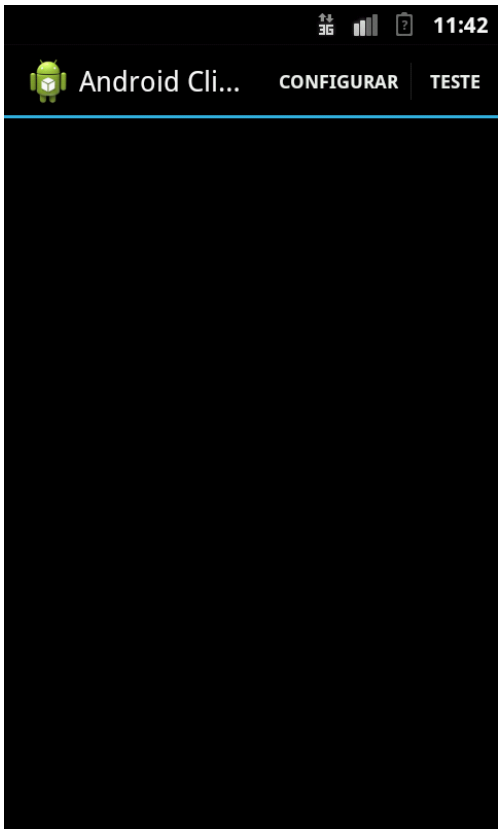


Figura 15. Tela utilizando a ActionBar.

Porém, esta classe só foi adicionada nas versões mais recentes do Android, mais especificamente na versão 3.0 (do SDK 11), ou seja, essa interface não estaria disponível para as versões mais antigas do Android, que era um dos principais objetivos deste projeto. Para contornar este problema, foi necessário procurar na internet alguma solução possível de retrocompatibilidade.

A solução achada foi utilizar uma biblioteca feita por Wharton (2014), chamada de ActionBarSherlock, ela oferece uma abordagem simples, e retrocompatível com as versões antigas do Android.

Neste caso, é necessário apenas chamar esta biblioteca e utilizá-la no lugar da ActionBar normal, já que ela é uma herdeira da ActionBar, a única coisa extra que ela faz é oferecer a retrocompatibilidade com as versões antigas do Android.

Resolvido o problema do *layout* da interface, o próximo passo foi criar uma interface básica, onde nela consistiria basicamente uma tela onde existiria os comandos para os passos do robô, enquanto que uma outra teria as configurações de qual robô seria movimentado e sobre o IP a ser conectado, estas telas são apresentadas na seção de resultados. Assim a interface básica seria construída e poderia ser feito o controle dos robôs através do dispositivo Android.

O passo seguinte foi criar uma alternativa segura para o envio e recebimento de dados entre a interface e o AEDROMO. O método escolhido foi o de utilizar o próprio conceito de *sockets* do Java, onde é criado um datagrama para os *sockets*, onde é enviado as informações para o servidor contendo os movimentos necessários.

No caso do Android, para fazer qualquer tipo de conexão de dados é necessário alterar as permissões do aplicativo, isso acontece para que evite que usuários utilizem um aplicativo sem saber o que ele está usando, ou seja, para cada módulo, como por exemplo, telefone, agenda, mensagens, *internet*, *bluetooth* etc. o aplicativo deve conter no seu arquivo de manifesto as suas permissões para que o usuário ao fazer o *download* na loja oficial da Google saiba o que o aplicativo utiliza. Na figura 16 pode ser visto o arquivo de manifesto.

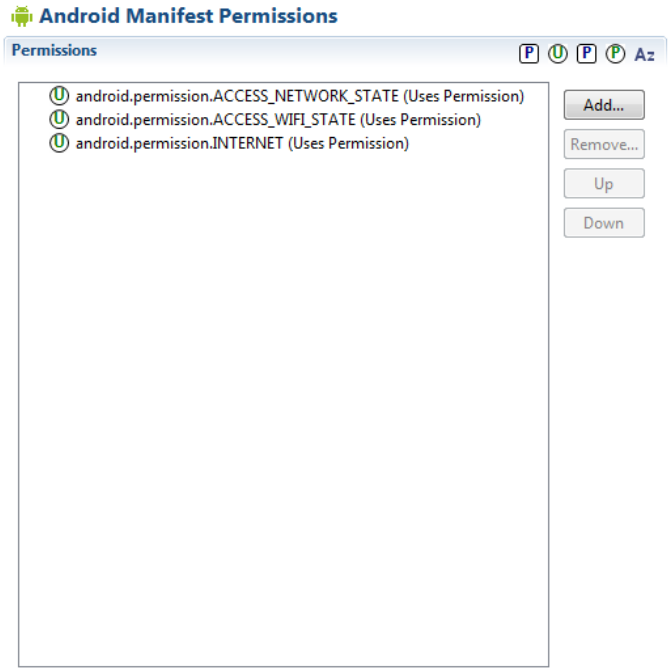


Figura 16. Arquivo de manifesto mostrando as permissões.

Segundo esta figura 16, o aplicativo pede autorização para poder verificar o estado da rede *wi-fi* e de dados, ou seja, se o dispositivo está conectado na rede, e também pede permissão para poder utilizar a *internet* efetivamente, sendo assim possível fazer a troca de dados entre o dispositivo e o servidor.

Resolvido o problema da conexão, era necessário criar uma forma para que o *loop* de envio dos datagramas não travasse a utilização do aplicativo, já que na primeira abordagem (onde apenas a conexão de *socket* era feita) não era possível utilizar o dispositivo depois de enviar o comando, fazendo com que o aplicativo ficasse inutilizável.

A segunda abordagem utilizava *thread*, como são utilizadas pelo Java, ou seja, a classe implementava um "Runnable", dessa forma quando o botão fosse pressionado ele acionava o método *run* e então a *thread* começaria a executar, infelizmente esse método não era possível, o Android recomenda a própria *thread* criada na API própria para esse fim, a figura 17 mostra a classe.

```
1  package com.tcc.androidclienttoraedromo.socket;
2
3  import java.io.IOException;
15
16
17 public class RoboClienteTeste2 implements Runnable {
```

Figura 17. Runnable.

Apenas na terceira abordagem foi possível, nesse caso implementando o "AsyncTask", a *thread* da própria API do Android (a classe pode ser vista na figura 18). Este método funciona como uma *thread* normal, roda apenas uma vez, e é dividida em quatro partes.

```
public class RoboClienteTeste3 extends AsyncTask <String,Void,Void>{

    class Estado {
        float angulo;
        float dAngulo;
        float x, y;
        float dx, dy;
    }

    final int indice = 1;
    final int MAX_OBJETOS = 100;
    //número de robôs que o cliente controla
    final int NUM_ROBOS = 1;
    int numeroObjetos;

    DatagramSocket socket;
    Estado estado[] = new Estado[MAX_OBJETOS];


    @Override
    protected void onPreExecute() {
        // TODO Auto-generated method stub
        super.onPreExecute();
```

Figura 18. AsyncTask.

A primeira parte o "onPreExecute" é usada antes da *thread* ser executada, para mostrar uma barra de progresso por exemplo. A segunda parte que é o "doInBackground" é a *thread* em si, e a única parte que foi efetivamente utilizada (já que não é exibida nenhuma barra de progresso), nela está contido o *loop* principal com o código para fazer a conexão entre o servidor e o aplicativo, na figura 19 tem o início do código. A terceira parte "onProgressUpdate" é um método que executa no mesmo tempo que a *thread*, é durante o "doInBackground", e é usado para poder mostrar *logs* enquanto a *thread* é executada. A quarta e última é a "onPostExecute", parecida com a "onPreExecute", só que após o término da *thread*. As únicas que são obrigatórias ter no código são a "onPreExecute", doInBackground" e a "onPostExecute".

```
    @Override
    protected Void doInBackground(String... params) {
        int numPorto = 0000;
```

Figura 19. Método doInBackground.

Feito isso a interface base foi concluída e a conexão entre dispositivo-servidor, foi concluída, terminando assim o desenvolvimento do aplicativo.

# 4. RESULTADOS

Como resultado obtêm-se o aplicativo base para desenvolvimento de futuras interfaces, este aplicativo tem o controle remoto sobre o AEDROMO. Pode-se observar nas figuras 20 e 21, as duas telas básicas para o uso do aplicativo.

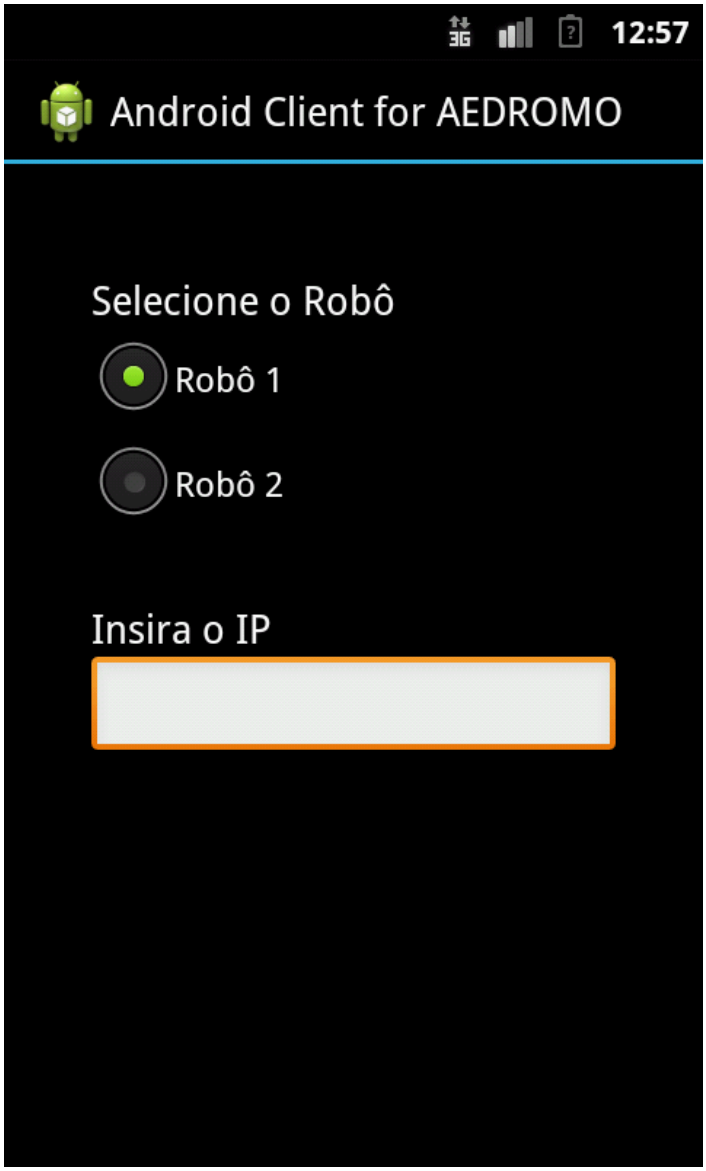


Figura 20. Tela das configurações.

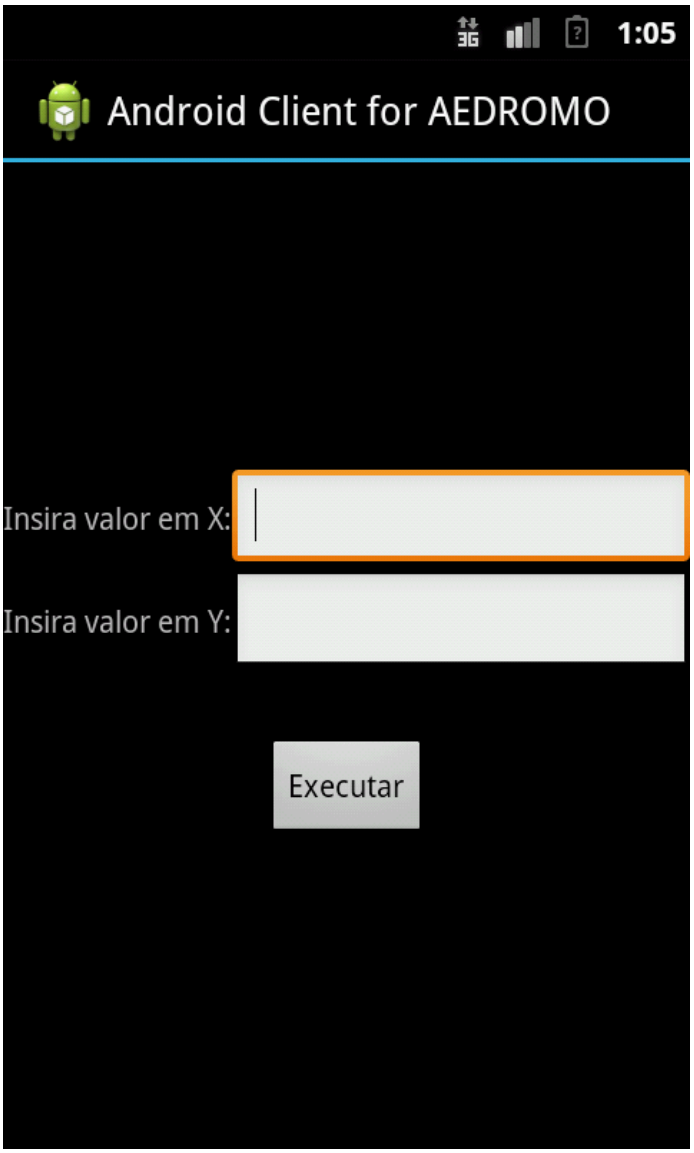


Figura 21. Tela dos parâmetros para execução.

Este aplicativo, uma vez iniciado apresenta um menu com 2 opções, para assim poder acessar as 2 telas principais do aplicativo, a primeira opção (vista na figura 20), é a tela de configurações. Nela é apresentado um grupo de botões para escolher qual robô será controlado e um campo de texto que é onde se coloca o IP a ser acessado, este campo aceita apenas números, e o formato do texto deve ser o padrão para IP, com números e pontos. As informações assim que postas nesta tela podem ser obtidas fazendo um acesso ao próprio componente, ou seja, não é necessário clicar em algum botão para que estes dados sejam guardados em uma determinada variável, dessa forma após os dados serem colocados nos campos basta apertar o botão de voltar do dispositivo Android para retornar à tela principal.

Já a segunda tela (vista na figura 21) apresenta apenas as coordenadas e um botão de executar para dar início a comunicação entre o dispositivo e o AEDROMO. As coordenadas são colocadas em um campo de texto que aceita apenas números. Ao clicar neste botão de executar as informações são coletadas, tanto das coordenadas, quanto das configurações, e em seguida a conexão é realizada e os movimentos são feitos no ambiente.

A parte referente ao estudo do AEDROMO foi motivado para poder caracterizá-lo na intenção de abranger ainda mais o projeto original e servir como uma literatura única para os futuros trabalhos que venham a utilizar Android para desenvolvimento de interfaces. Este estudo foi possível através do levantamento bibliográfico de alguns artigos criados pelo GISDI e também pelo estudo do simulador criado para o AEDROMO, também foi feito um entendimento sobre o mesmo para poder formalizar e explicitar sobre o funcionamento a respeito desta ferramenta que é o simulador.

Por fim, o aplicativo criado tem a capacidade de se conectar ao servidor do AEDROMO e realizar os comandos básicos como mexer qualquer um dos robôs através de suas próprias coordenadas.

# 5. CONCLUSÃO

A contribuição principal deste trabalho é um aplicativo que serve de base para desenvolvimento de futuras interfaces em dispositivos móveis para a interação com o AEDROMO.

Além disso a própria concepção deste aplicativo quebra com um paradigma que é o de utilizar um dispositivo móvel como um *smartphone*, para se conectar a um universo interativo, mostrando assim que é possível utilizar o AEDROMO de outras formas, não apenas através dos computadores em postos fixos.

Como produto final temos um aplicativo e um texto que sintetiza não só uma simples aplicação que serve como complemento para uso do AEDROMO, mas também toda uma documentação de suporte, explicando o propósito do projeto, a necessidade de novas tecnologias para a interação com o AEDROMO, a explicação do simulador e como utilizá-lo para testes. Este aplicativo apresenta um código robusto que pode ser facilmente modificado para futuros trabalhos.

Para o futuro, baseado neste trabalho podem ser criadas novas interfaces, utilizando como parâmetro as interfaces aqui apresentadas na seção de fundamentação teórica e fazendo uso do sistema Android e os sensores do dispositivo.

# REFERÊNCIAS

Alves, S. F. R., Ferasoli Filho, H., Pegoraro, R., Caldeira, M. A. C., Yonezawa, W., Rosário, J., Ambiente Educacional de Robótica Direcionado a Aplicações em Engenharia. **Anais do SBAI - Simpósio Brasileiro de Automação Inteligente**, 2011.

Aroca, R. V., Oliveira, A. P. B. S., Gonçalves, L. M. G., Towards Smarter Robots With Smartphones. **5th workshop in applied robotics and automation**, Bauru, Brasil, 2012.

Avanzato, R., Mobile robotics for freshman design, research, and high school outreach. **2000 IEEE International Conference on Systems, Man, and Cybernetics**, 2000.

Business Insider. Chart of the day: Android's Fragmentation Problem. Disponível em: <http://www.businessinsider.com/chart-of-the-day-androids-fragmentation-2013-9>. Acesso em: 07 de janeiro de 2014.

Chen, M., Chen, J., Chang, T. **Android/OSGi-based vehicular network management system**. *Computer Communications*, v.34, p.169-183, 2010.

Ferasoli Filho, H., Pegoraro, R., Caldeira, M. A. C., Rosário, J., AEDROMO- An Experimental and Didactic Environment with Mobile Robots. **Proceedings of The 3rd International Conference on Autonomous Robots and Agents**, Palmerston North, Nova Zelândia, 2006.

Goebel, S., Jubeh, R., Raesch, S. -L., Zuendorf, A., Using the Android Platform to control Robots. **Proceedings of 2nd International Conference on Robotics in Education**, Viena, Áustria, 2011.

IDC. Android and iOS Combine for 91.1% of the Worldwide Smartphone OS Market in 4Q12 and 87.6% for the Year. Disponível em: <http://www.idc.com/getdoc.jsp?containerId=prUS23946013#.UUyTDjs-26_>. Acesso em: 22 março 2013.

Lopes, R. J., Mascote Eletrônico. Unesp Ciência, n.39, p.xii-xv, março, 2013.

Savall, J., Avello, A., Briones, L., Two Compact Robots for Remote Inspection of Hazardous Areas in Nuclear Power Plants. **1999 IEEE International Conference on Robotics and Automation**, Detroit, Estados Unidos, 1999.

Sethi, C., iPhone Apps for Robot Control. Disponível em: <http://www.asme.org/kb/news---articles/articles/robotics/iphone-apps-for-robot-control>. Acesso em: 07 maio 2013.

Sugiura, Y., Kakehi, G., Withana, A. I., Fernando, C. L., Sakamoto, D., Inami, M., Igarashi, T., Walky: Operating Method for a Bipedal Walking Robot through Personalized Gestures Expression of Fingers. **ACM SIGGRAPH Asia 2009 Emerging Technologies, pp79-79**, 2009.

Wharton, J., ActionBarSherlock - Biblioteca de compatibilidade com o Android. Disponível em: <https://github.com/JakeWharton/ActionBarSherlock>. Acesso em: 08 de janeiro de 2014.

Xie, C., Zhou, J., Li, S., Jia, L., The Design And Implementation of Mobile Monitoring System of Transmitting Station Based on Android Platform. **2012 International Conference on Mechanical an Electronics Engineering**, Liaoning, China, 2012.