

RODOLFO EGON PERHS

Desenvolvimento de solução mobile para acionamento de robôs autônomos de sumô.

Guaratinguetá – SP
2017

Rodolfo Egon Perhs

Desenvolvimento de solução mobile para acionamento de robôs autônomos de sumô.

Trabalho de Graduação apresentado ao Conselho de Curso de Graduação em Engenharia Elétrica da Faculdade de Engenharia do Campus de Guaratinguetá, Universidade Estadual Paulista, como parte dos requisitos para obtenção do diploma de Graduação em Engenharia Elétrica.

Orientador (a): Daniel Julien B. da S. Sampaio

Guaratinguetá - SP
2017

P439d	Perhs, Rodolfo Egon Desenvolvimento de solução móbile para acionamento de robôs autônomos de sumô / Rodolfo Egon Perhs – Guaratinguetá, 2017. 108 f : il. Bibliografia: f. 37 Trabalho de Graduação em Engenharia Elétrica – Universidade Estadual Paulista, Faculdade de Engenharia de Guaratinguetá, 2017. Orientador: Prof. Dr. Daniel Julien Barros da Silva Sampaio 1. Robótica. 2. Smartphone. Gerenciamento da informação. I. Título
	CDU 007.52

DESENVOLVIMENTO DE SOLUÇÃO MOBILE PARA ACIONAMENTO DE ROBÔS
AUTÔNOMOS DE SUMÔ.

RODOLFO EGON PERHS

ESTE TRABALHO DE GRADUAÇÃO FOI JULGADO ADEQUADO COMO
PARTE DO REQUISITO PARA A OBTENÇÃO DO DIPLOMA DE
"GRADUADO EM ENGENHARIA ELÉTRICA"

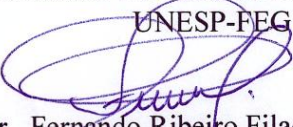
APROVADO EM SUA FORMA FINAL PELO CONSELHO DE CURSO DE
GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Prof. Dr. Leonardo Mesquita
Coordenador

BANCA EXAMINADORA:


Prof. Dr. Daniel Julien Barros da Silva Sampaio
Orientador/UNESP-FEG


Prof. Dr. Samuel Euzédice de Lucena
UNESP-FEG


Prof. Dr.. Fernando Ribeiro Filadelfo
UNESP-FEG

Fevereiro de 2017

Dedico este trabalho de modo especial, à
minha família, à equipe FEG – Robótica e a todos os amigos
que me assistiram nos últimos anos.

AGRADECIMENTOS

Se nomeasse um a um todos àqueles a quem devo meus agradecimentos pela participação ativa e contribuição única em minha vida até a presente data, certamente falharia; Sendo assim, opto por agradecer ao subjetivo: Agradeço a Deus, à minha família sanguínea, àqueles que compartilharam os palcos musicais comigo, aos professores que destoaram da massa, aos amigos especiais que tive o prazer de encontrar, e por fim, aos membros da equipe FEG-Robótica, da qual tenho orgulho em dizer que participei.

“If I have seen further it is by standing on the shoulders
of Giants.”

Isaac Newton

RESUMO

Este trabalho fundamenta-se no desenvolvimento de uma aplicação *mobile* dedicada à dinâmica dos combates de robôs autônomos nas categorias de sumô. Os objetivos desta solução para *smartphones* com sistema operacional *Android* almejam exemplificar a construção de uma interface homem máquina (IHM) simples, funcional e fácil de ser manipulada pelo usuário, bem como explorar os recursos da comunicação wireless, essencialmente a tecnologia Bluetooth. Para tanto, escolheu-se trabalhar com o ambiente de desenvolvimento integrado *Android Studio*, o qual oferece amplos recursos para elaboração de códigos para tomadas de decisão e ações de controle utilizando o processamento interno dos aparelhos celulares, análise de resultados de forma dinâmica, bem como facilidades na configuração da identidade visual da solução *mobile*. Munido destes recursos, entende-se que o desenvolvimento de uma aplicação voltada aos campos da robótica autônoma, integrando sistemas embarcados, cada vez mais populares e difundidos por grandes empresas do ramo da tecnologia como *Intel Corporation* e outras, vai de encontro com as tendências mundiais para o universo da eletrônica e dos sistemas a serem desenvolvidos no século XXI, principalmente quando se alinha esta perspectiva com outra tendência de mercado: o desenvolvimento de aplicativos para *smartphones*, os quais têm se popularizado a cada dia mais, numa infinidade de aplicações variando desde entretenimento até sérias aplicações industriais para gerenciamento de informações. Desta forma, este trabalho busca sugerir caminhos importantes a serem considerados para o desenvolvimento de uma metodologia extensível às mais variadas necessidades e tendências globais, não se limitando em expor com exclusividade a solução para acionamento remoto de robôs autônomos, mas sim permitindo que aquele que fizer uso deste trabalho possa sentir-se mais confiante para iniciar desenvolvimentos diversos com comunicação wireless e *smartphones*, seja nos domínios da robótica autônoma e móvel ou não.

PALAVRAS-CHAVE: Robótica. Smartphone. Gerenciamento da informação.

ABSTRACT

This work sustains itself in the development of a mobile application which shall be dedicated to the dynamics of sumo robots battles. The main objectives of this solution for smartphones with Android operational system aims to exemplify the construction of a simple and functional Human Machine Interface, easy to be handled by the user, as well as explore the resources available in wireless communication, essentially the Bluetooth technology. For that, it was choose to use the integrated development environment (IDE) named Android Studio, which offers ample resources to elaborate codes for decision making process and action control, using the internal processing hardware from mobile phones – smartphones, dynamic analysis of results, as well as facilities to create and configure the visual identity of the mobile solution. Provided with these resources its understood that the development of an application focused on the fields of autonomous robotics, integrating embedded systems, increasingly popular and widespread for major companies around the world such as Intel Corporation and others, matches with the global tendencies for the universe of electronics and systems to be developed in the 21th century, principally when it gets aligned with another market trend: the development of applications for smartphones, the ones which had become more and more popular across the years in a vast infinity of possible application from entertainment to serious data management for industries. Thus this work aims to suggest important paths to be considered in the development of a methodology extensible to the most varied needs and global trends, not limited in exposing only the solution for remote actuation for autonomous robots, but allowing the ones who use the next pages to feel more confident to start in several developments with wireless communication and smartphones, even that they are not in the dominions of mobile autonomous robotics.

KEYWORDS: Robotics. Smartphone. Information management.

LISTA DE ILUSTRAÇÕES

Figura 1 – Interface do Android Studio.....	23
Figura 2 – Cronograma do projeto	24
Figura 3 – Tela principal do aplicativo.....	27
Figura 4 – Break-down da tela principal	28
Figura 5 – Fluxograma de funcionamento.....	30
Figura 6 – Procedimento de teste (identidade visual)	32
Figura 7 – Procedimento de teste (travas funcionais e funcionamento)	32
Figura 8 – Procedimento de teste (range de aplicação).....	33

LISTA DE TABELAS

Tabela 1 – Componentes gráficos	28
Tabela 2 – Componentes gráficos x Funcionalidade	29
Tabela 3 – Resultados de teste (login)	33
Tabela 4 – Resultados de teste (identidade visual)	34
Tabela 5 – Resultados de teste (travas e funcionalidades).....	34
Tabela 6 – Resultados de teste (range de aplicação).....	35

LISTA DE ABREVIATURAS E SIGLAS

APK	Aplicativo para <i>smartphone</i>
BT	Bluetooth
FEG	Faculdade de Engenharia de Guaratinguetá
IDE	Integrated Developing Envirolment (Ambiente de Desenvolvimento Integrado)
IHM	Interface Homem Máquina
IoT	Internet of Things (Internet das coisas)
IR	Infra Red (Infra-vermelho)
PIC	Peripheral Interface Controller (Interface Controladora de Periféricos)
POO	Programação Orientada a Objetos
WC XII	Winter Challenge 12 (Campeonato de robôs)

SUMÁRIO

1	INTRODUÇÃO	12
1.1	A MODERNIZAÇÃO DOS PROCESSOS INDUSTRIAIS E O SURGIMENTO DA ROBÓTICA	13
1.2	A EVOLUÇÃO DA TECNOLOGIA APLICADA ÀS ROTINAS PESSOAIS – A ERA SMART	13
2	DESENVOLVIMENTO (REVISÃO TEÓRICA)	15
2.1	DINÂMICAS DO COMBATE DE ROBÔS DE SUMÔ AUTÔNOMO	15
2.2	DISPOSITIVOS DE CONTROLE	17
2.2.1	Microcontroladores e sistemas embarcados	17
2.2.2	Smartphones	18
2.3	COMUNICAÇÃO ENTRE DISPOSITIVOS	19
2.3.1	Comunicação serial	19
2.3.2	Comunicação wireless	20
2.3.3	Comunicação Bluetooth	20
2.4	INTERFACES GRÁFICAS E A INTERAÇÃO HOMEM-MÁQUINA	22
2.4.1	IDE Android Studio	22
3	DESENVOLVIMENTO (METODOLOGIA APK)	24
3.1	DEFINIÇÃO DAS FUNCIONALIDADES	25
3.2	DEFINIÇÃO DAS APLICAÇÕES (LAYOUT GRÁFICO)	25
3.3	TESTES DE VALIDAÇÃO	31
4	CONCLUSÃO	36
	REFERÊNCIAS	37
	BIBLIOGRAFIA CONSULTADA	38
	APENDICE A – Linhas de código	39
	APÊNDICE B – Regras da competição WINTER CHALLENGE XII	100

1 INTRODUÇÃO

A notável evolução tecnológica a partir do final do século XX surpreende a história trazendo à tona desenvolvimentos direcionados não mais somente à produção industrial, mas com um grande foco no cidadão comum e suas rotinas diárias, popularizando a incorporação de recursos tecnológicos ao dia-a-dia da sociedade; incorporação esta, sustentada por diversos fatores, dentre os quais se destacam os processamentos computacionais já enraizados na cultura moderna e presentes nas mais diferentes atividades e produtos de consumo; afinal de contas, realizar compras numa rede de supermercados, acessar sua conta bancária ou ainda conectar-se a uma rede de contatos estabelecendo uma comunicação efetiva, ainda que à distância, tornou-se uma atividade dependente da computação e seus derivados. Além disto, os chamados eletrodomésticos avançam mais e mais na utilização de processamentos computadorizados para melhoria de suas performances e ganho de competitividade através de diferenciais tecnológicos frente à forte concorrência do livre-mercado globalizado, inserindo o processamento computacional no interior das residências, através de intensas preocupações com o design intuitivo e de fácil entendimento das interfaces homem-máquina, responsáveis por auxiliar o usuário comum na adequada operação dos equipamentos eletrônicos. E para explicar este comportamento histórico, FERNANDES et al. (2012) propõe: “A partir da curiosidade, percepção, imaginação e necessidade; o homem foi desenvolvendo técnicas e criando a cada dia novos produtos, começaram a produzir artesanalmente diversos produtos, como calçados, móveis, casas, roupas, etc.”; portanto não seria questionável esperar que estes mesmo fatores motivadores conduzissem a sociedade para auxiliar no desenvolvimento de dispositivos para realização das múltiplas tarefas executadas pelo homem. Assim surgiu o conceito transformador da robótica, elaborada inicialmente sobre o prisma da realização de trabalhos forçados ou escravos, como refere-se o termo Robota, divulgado primeiramente numa peça de 1921 de Karel Tchépek, traduzida para o português sob o título de “A fábrica de robôs”; sendo esta a primeira referência reconhecida às origens do termo robótica.

1.1 A MODERNIZAÇÃO DOS PROCESSOS INDUSTRIAIS E O SURGIMENTO DA ROBÓTICA

Como referenciamos à peça de Tchápek, as origens do termo robótica estabelece sua base no conceito de realização de trabalhos forçados ou ainda escravos, isto, pois o conceito de robôs surgiu na história como uma estratégia para a realização de tarefas repetitivas, propondo uma atuação de alta repetibilidade, com precisão, exatidão e velocidade teoricamente melhores do que as factíveis para seres humanos.

Quando se fala em tarefas repetitivas, onde exatidão, velocidade e precisão são fatores essenciais, logo o raciocínio volta-se às operações fabris iniciadas na revolução industrial do século XVIII onde o uso de equipamentos projetados para substituir e/ou aperfeiçoar as atividades desenvolvidas pela mão-de-obra humana marca um ponto importante na história.

Obviamente, a evolução tecnológica da sociedade foi muito além dos “spinning mules” de Samuel Crompton, e à medida que se exploravam mais e mais a utilização de máquinas para tarefas previamente exclusivas do homem, muita especulação surgia no tocante à possibilidade de os chamados robôs industriais substituírem por completo a mão-de-obra convencional, todavia este temor não passou até então das obras de ficção, ainda que a participação de robôs industriais seja cada vez maior em linhas de produção.

1.2 A EVOLUÇÃO DA TECNOLOGIA APLICADA ÀS ROTINAS PESSOAIS – A ERA SMART

A tecnologia por si é sempre transformadora. Miranda (2002) reforça esta alegação ao afirmar:

Na modernidade (a partir do séc. XVI), devido a fatores históricos, sociais, culturais, econômicos, políticos, a tecnologia sofre e propicia transformações profundas. E muito além de alterar padrões de comportamento, a tecnologia, a partir da modernidade, contribui para alterar a relação do ser humano com o mundo que o cerca, implicando no estabelecimento de uma outra cosmovisão, diferentemente daquela dos gregos ou dos medievais. (MIRANDA, 2002)

O século XXI surge na história sustentado primordialmente pela popularização dos computadores, e num crescente avanço das tecnologias de transmissão de dados, especialmente a internet. Em concordância com o trabalho de COUTINHO (2014), observa-se, no entanto, um marco no ano de 2007, quando a Apple lança seu primeiro dispositivo móvel, batizado por Iphone; reescrevendo o conceito de *smartphone* e iniciando o que seria uma verdadeira revolução na vida de consumidores de todas as níveis econômicos, dos mais variados países. Pode-se ainda dizer que a maneira de se comunicar foi totalmente reinventada a partir de então. O conceito trazido pela Apple foi tão bem recebido pelos usuários, que em menos de 10 anos após o lançamento do primeiro Iphone, todos os players do mercado de telecomunicação desenvolvem

atualmente quase que integralmente seus aparelhos no conceito smart, restando nas prateleiras do varejo poucos modelos de aparelhos celulares convencionais, como os anteriores ao ano de 2007.

Importante é entender a razão do sucesso dos *smartphones* a nível mundial. Atribuiu-se inicialmente seu sucesso à incorporação da internet às suas funcionalidades, oferecendo uma conexão sem fronteiras entre contatos, algo tão grande quanto o usuário possa comportar no mundo digital, não importando limites físicos, fuso-horários, ou ainda o conteúdo da mensagem, pois além das conversações usuais, fotos, vídeos e os mais diversos arquivos de mídia são compartilhados em questões de segundos por todo o globo. Esta facilidade, aliada à dinâmica dos aplicativos, que nada mais são do que softwares desenvolvidos pelo fabricante do aparelho ou outros desenvolvedores ao redor do mundo, com intuitos assertivos, auxiliando em todo o tipo de tarefa organizacional ou simplesmente trazendo entretenimento na forma de jogos aplicados a todas as idades e públicos-alvo.

Afinal de contas, segundo COUTINHO (2014, p.50) “Não é somente no campo dos negócios e das relações humanas que os *smartphones* têm causado impactos. Na realidade, aplicações diretas de sua tecnologia já podem ser observadas em áreas como medicina, educação, segurança e transporte.”; o que confere posição especial aos *smartphones*.

2 DESENVOLVIMENTO (REVISÃO TEÓRICA)

A interação entre tecnologias vanguardistas, como os *smartphones* e a robótica, por exemplo, demonstram ser uma tendência para os novos tempos do século XXI. Todas estas tecnologias têm sido amplamente estudadas e aprimoradas, gerando adeptos convictos, como MAIA que engrandece a utilização da robótica ao afirmar que:

As transformações que ocorrem, causadas pelo advento dos robôs, muitas vezes podem não estar visíveis para grande parte das pessoas que não convivem no ambiente fabril, contudo a ascensão da robótica nas fábricas faz parte da mesma tendência que vem determinando, nos últimos anos, a crescente automatização dos bancos, do comércio e das empresas em geral, causados pelo advento da informática. (MAIA)

Já LEMOS (2010) é taxativo ao esclarecer que “as tecnologias sem fio estão transformando as relações entre pessoas e espaços urbanos, criando novas formas de mobilidade.” O que por sua vez demonstra a importância de integração entre tecnologias buscando facilidades de acesso que os *smartphones* são capazes de ofertar de forma convincente. Especialmente sobre *smartphones*, FINCOTTO (2014) demonstra estar convencido da qualidade inerente à utilização destes dispositivos e seus sistemas operacionais, ao comentar sobre o sistema operacional Android:

“A plataforma Android se apresenta como um sistema operacional para dispositivos móveis robusto e flexível, e que pode ser utilizado para aplicações comerciais no ambiente corporativo.”

Incorporando estes adventos tecnológicos e seus impactos na sociedade atual, faz-se claro a necessidade de ir além e explorar outras possibilidades do emprego destas tecnologias modernas, e, particularmente contempladas neste trabalho voltado a aplicação de um software para *smartphones*, encontra-se o controle de robôs via Bluetooth em uma modalidade de entretenimento nomeada combate de sumôs autônomos.

2.1 DINÂMICAS DO COMBATE DE ROBÔS DE SUMÔ AUTÔNOMO

Previamente à discussão acerca da dinâmica do combate de robôs de sumô, é de suma importância ressaltar que projeto de robô por parte da equipe FEG-Robótica, busca o desenvolvimento de um agente reativo simples, o que segundo Stuart Russel e Peter Norvig (2004, p.46) seriam agentes que “selecionam ações com base na percepção atual, ignorando o restante do histórico”. Esta informação, ainda que podendo ser desconsiderada para o desenvolvimento do software de acionamento, ajuda a situar o desenvolvedor na aplicação, o que tonifica suas possibilidades de ações otimizadas.

Para o desenvolvimento de uma solução dedicada, como a proposta deste trabalho, faz-se necessário o conhecimento profundo da aplicação em questão: o combate de robôs de sumô autônomo e, ainda mais especificamente, segundo as regras da Robocore, elaboradas para o evento Winter Challenge XII – anexo B; uma vez que o foco do

projeto era o desenvolvimento de um aplicativo para acionamento via Bluetooth de robôs de sumô autônomo que deveriam disputar a competição Winter Challenge XII.

Ainda que sem conhecimento prévio do design mecânico dos robôs, devemos nos atentar às principais regras desta aplicação, as quais seguem abaixo, conforme o anexo B:

“A partida é disputada por duas equipes, cada uma composta por um ou mais integrantes... Uma partida consistirá em 3 (três) rounds, cada round terá um tempo nominal de 1 (um) minuto, podendo ser acrescentado, a critério do juiz, 30 segundos totalizando um tempo total de 1 minuto e 30 segundos (1’ 30’’)... Os robôs autônomos deverão entrar em operação automaticamente em não menos do que cinco (5) segundos após autorização do juiz e comando dado por um membro da equipe.”

Como parte da dinâmica do combate de robôs de sumô autônomo destaca-se a multiplicidade de robôs dentro das diversas equipes competidoras. É muito comum que uma mesma equipe trabalhe com mais de um robô por categoria, visando melhores colocações e validações de diferentes conceitos de projeto eletro-mecânico. E sendo a variedade de robôs, ou ainda, oponentes, elevada; entende-se que mais do que as nuances do projeto eletro-mecânico do robô, as estratégias a serem adotadas pelo robô, em cada rodada, podem se tornar decisivas para o resultado final do combate. Tratando-se de robôs autônomos, a estratégia deve ser assumida com a maior assertividade possível, para que de acordo com o comportamento do adversário, o robô possa decidir a melhor maneira de vencer a disputa. É importante ressaltar que a maior parte dos robôs competidores utiliza uma única estratégia por robô, deixando assim um caminho claro de se explorar multiplicidade de estratégias pré-programadas e escolhidas de acordo com cada combate.

Uma vez esclarecidos os principais pontos da dinâmica do combate de sumôs autônomos, compreende-se que o aplicativo móvel proposto deve ser capaz de oferecer os seguintes recursos:

- Escolher qual o robô deverá ser controlado, uma vez que a equipe FEG – Robótica, para a qual o aplicativo foi desenvolvido, possui 5 robôs de sumô divididos entre 2 categorias;
- Escolher qual a estratégia o robô deverá assumir para cada um dos 3 rounds, uma vez que a versatilidade de estratégias tem se mostrado historicamente efetiva na busca por melhores resultados;

- Acionamento preciso após 5 segundos da autorização do juiz, já que este é um critério crucial para a aplicação.

2.2 DISPOSITIVOS DE CONTROLE

2.2.1 Microcontroladores e sistemas embarcados

Tomando por base uma abordagem simplista, tem-se que microcontroladores nada mais são do que computadores em miniatura, reduzidos a uma única estrutura, que através de um baixíssimo consumo de energia: une as funções de processamento digital, leitura e escrita em memória RAM e ROM, além de outras funcionalidades que viabilizam a interação com periféricos diversos.

Microcontroladores são computadores de propósito específico. Eles possuem tamanho reduzido, baixo custo e baixo consumo de energia. Devido a esses fatores há diversos segmentos, que os utilizam, tais como a indústria automobilística, de telecomunicações, de brinquedos, de eletrodomésticos, de eletroeletrônicos, bélica, etc. (SILVA, 2009)

A história dos microcontroladores remete à década de 1970 quando em 1971 o *Smithsonian Institution*, creditando os engenheiros da *Texas Instruments*, criam o primeiro microcontrolador, que somente seria comercializado em 1974 sob o título de TMS 1000 com funções de memória para leitura, escrita, processador e relógio numa só pastilha, o que já previa um mercado de aplicação intitulado sistemas embarcados.

MEHL (2011) define sistemas embarcados como “sistemas computacionais de uso específico, com seus recursos computacionais como memória e poder de processamento projetados restritamente para este propósito especial.” Desta forma, podem ser entendidos como um sistema desenhado para executar um conjunto limitado de instruções, de forma pré-definida, com limitantes bem claros no tocante a dimensões e custos, principalmente. Este tipo de aplicação utiliza-se de um microcontrolador unido a alguns periféricos, responsáveis por estabelecer um tipo de comunicação com o usuário, ainda que sejam periféricos simples com a função de apenas ilustrar o status de funcionamento, sem necessariamente responder de forma inteligente a comandos do usuário.

Exemplos destes sistemas popularizam-se nos campos da robótica móvel e domótica (automatização de processos para residências) dado ao baixo custo e simplicidade de projeto, se comparado com soluções equivalentes que fariam uso de um

hardware com capacidade mais elevadas, tido como *overspec* (acima das especificações), o que apesar de atender e até mesmo superar as necessidades da aplicação, tornam o projeto de elevado custo e complexa implementação já que um hardware mais robusto tende a possuir de forma isolada seu microprocessador e em consequência disto, um ambiente para configuração e programação mais próximo a um sistema operacional, capaz de ofertar inúmeros recursos, que podem, todavia facilmente serem desconsiderados de acordo com a aplicação. O sucesso e popularização dos sistemas embarcados têm atingido grande destaque principalmente com os conceitos de IoT (*Internet of Things*) e a variedade de sistemas pré-concebidos como *Arduino*, *BeagleBone*, *Raspberry* e outros que, comercializados a baixo custo e em formato livre (*open source*), tornaram possível que até mesmo crianças possam iniciar atividades básicas de engenharia fazendo uso de microcontroladores robustos e de fácil utilização, sem a necessidade de longos anos de experiência nas áreas da microeletrônica e ciências da computação.

Os Sistemas Embarcados encontram-se cada vez mais presentes em nosso dia-a-dia, e com uma utilização e importância crescente, tornam-se necessários estudos nas áreas de projeto em hardware, software e interfaceamento com base em sistemas embarcados. (CHASE, 2007)

2.2.2 *Smartphones*

Torres (2009, p.393) classifica *smartphone* como sendo um “celular que oferece recursos avançados similares aos de um *notebook*”. E sendo assim, possui uma capacidade de hardware para processamento significativamente maior do que a interna ao controlador utilizado nesta aplicação da FEG – Robótica, um PIC da família 18F.

Dentre as utilidades já incorporadas dos *smartphones*, destaca-se o recurso de comunicação via Bluetooth, poderosa tecnologia para transmissão de dados.

Além disto, *smartphones* permitem a criação de aplicativos, que podem ser customizáveis de acordo com a necessidade do usuário. Desta forma, a criação de um aplicativo para sistema operacional *Android* ganhou espaço devido às facilidades de ser considerado *open source*, o que ampliaria a fonte de consulta para o desenvolvimento, aliado ao fato de aproximadamente 86% da equipe de robótica (usuário final do aplicativo) dispor de um *smartphone* possuidor deste sistema operacional, em múltiplas versões.

2.3 COMUNICAÇÃO ENTRE DISPOSITIVOS

Comunicação pode ser entendida sobre múltiplos aspectos. Para o conteúdo apresentado neste trabalho, comunicação deverá ser entendida como a troca de informações entre sujeitos, ou objetos, o que neste caso, chamaremos por dispositivos. Uma comunicação bem estabelecida pressupõe que o conteúdo da mensagem a ser transmitida por dada fonte emissora, seja convenientemente interpretada pela fonte receptora, sofrendo, portanto, a menor distorção proveniente do meio de transmissão por onde a mensagem circulará até atingir seu destino.

Quando falamos em comunicação entre pessoas, uma rápida associação é feita com idiomas, que nada mais são do que formas padronizadas para representar uma mensagem utilizando de uma sequência de caracteres provenientes de um alfabeto alfanumérico. Já ao falarmos de comunicação entre dispositivos, especialmente dispositivos eletrônicos computadorizados, é mais apropriado assumir que o alfabeto será convertido para níveis lógicos, representados por zeros (0) e uns (1), configurando o que é chamado de *bit*. Obviamente, assim como existem inúmeros idiomas distintos, há também codificações diferentes para a transmissão destes *bits* que em conjunto formam as chamadas palavras. Um exemplo pode ser a palavra 0010 – uma palavra de quatro *bits*, ou seja, quatro caracteres distribuídos entre zeros (0) e uns (1).

Dispositivos eletrônicos podem ser configurados para transmitir e/ou receber, de acordo com suas características de hardware, estas chamadas palavras, ou ainda, mensagens, segundo diversos protocolos, que seriam equivalentes a idiomas na comunicação convencional entre pessoas. Para o projeto em questão, adotou-se a comunicação serial entre um celular com sistema operacional *Android* e um microcontrolador programável PIC - *Microchip* da família 18F, onde ainda, o meio de transmissão seria o ar através de um protocolo característico para a transmissão de dados via tecnologia *Bluetooth*.

2.3.1 Comunicação serial

Quando se trata de enviar um conteúdo, expresso em uma palavra constituída de certo número de bits, uma técnica muito conhecida é transmitir esta mensagem de forma dita serial, o que nada mais é do que enviar o conteúdo das palavras *bit a bit*, um de cada vez, seguindo um intervalo de tempo bem definido para cada *bit* durante a transmissão.

2.3.2 Comunicação *wireless*

Conforme sugere o nome, a comunicação *wireless* representa o conceito de transmissão de dados sem a utilização de fios e cabos, ou ainda, em outra interpretação, pode-se associar a comunicação *wireless* a uma forma de comunicação onde o meio de propagação do conteúdo da mensagem não é um elemento de matéria rígida como tradicionalmente há o costume, e sim, através da propagação das mensagens a serem transmitidas de uma fonte a outra, através do ar livre. Obviamente esta proposta de comunicação traz benefícios interessantes, principalmente em termos de conforto ao usuário, quesito extremamente relevante nas últimas décadas, onde a praticidade e velocidade se tornam cada vez mais algo imprescindível e cotidiano. Como outras vantagens ainda podem ser observadas a diversidade de informações que podem ser transmitidas para mais de um elemento receptor ao mesmo tempo, uma vez que o conteúdo está disponível no ar livre. Todavia, como toda tecnologia, também possui algumas limitações como o alcance de sinal limitado de acordo com os equipamentos e ambiente de instalação, além de uma susceptibilidade maior a ruídos ocasionados por obstáculos e reflexões de superfícies.

Ainda assim, as limitações podem ou não impedir a aplicação, o que geralmente pode ser superado através da escolha de equipamentos, devidamente instalados, de acordo com cada situação. Isto se afirma facilmente ao depararmos com os inúmeros exemplos de comunicação *wireless* que abrangem toda a extensão do planeta como: Rádio *Broadcast*, Telefonia móvel, Infravermelho, *Bluetooth*, Comunicação via satélite, Internet via Wi-Fi.

2.3.3 Comunicação *Bluetooth*

A tecnologia de comunicação chamada *Bluetooth* foi criada em 1994, com o objeto direto de ofertar uma alternativa à comunicação tradicional de sistemas eletrônicos através de fios e cabos. Esta alternativa propunha um conceito de funcionalidade por ondas de rádio emitidas e recebidas por dispositivos eletrônicos que possuíssem a tecnologia *Bluetooth*, ou seja, a aplicação *Bluetooth* oferta, através de configurações, uma perspectiva de comunicação bidirecional, onde os elementos incorporados à rede possam tanto enviar quanto receber mensagens; além disto, a proposta *Bluetooth* possibilita que a transmissão de conteúdo ocorra a mais de um

dispositivo por vez, já que a utilização de ondas de rádio influencia obviamente na propagação da mensagem ao ar livre, disponibilizando assim a informação para mais de um dispositivo por vez, sendo o limite de dispositivos interligados uma característica das versões distintas da tecnologia *Bluetooth*. Esta interligação de dispositivos é chamada de pareamento entre dispositivos. Uma vez que dois dispositivos estão devidamente pareados, a comunicação pode iniciar.

A Bluetooth device uses radio waves instead of wires or cables to connect to a phone or computer. A Bluetooth product, like a headset or watch, contains a tiny computer chip with a Bluetooth radio and software that makes it easy to connect. When two Bluetooth devices want to talk to each other, they need to pair. Communication between Bluetooth devices happens over short-range, ad hoc networks known as piconets. A piconet is a network of devices connected using Bluetooth technology. The network ranges from two to eight connected devices. When a network is established, one device takes the role of the master while all the other devices act as slaves. Piconets are established dynamically and automatically as Bluetooth devices enter and leave radio proximity. (How does Bluetooth work? Disponível em <<https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics>>. Acesso em 06 de junho de 2016.)

É importante ressaltar que uma vez que a propagação das mensagens ocorre no ar livre, há certamente limitantes quanto à distância que os dispositivos podem se afastar de tal forma que a comunicação prevaleça. Entre outros pontos, pode-se entender como as principais limitações da comunicação *Bluetooth* como:

- Limite de distância entre dispositivos para comunicação;
- Comunicação suscetível à interferência de obstáculos físicos e realimentação negativa consequente de superfícies reflexivas;

De toda forma, há certamente fortes atrativos para a utilização desta tecnologia, como segue:

- Baixo consumo de energia para transmissão de mensagens;
- Utilização simples e aberta para incorporação em projetos de baixo custo;
- Tecnologia confiável já presente em mais de um bilhão de dispositivos ao redor do mundo o que auxilia na comunicação com eletrônicos já presentes no mercado como celulares, fones de ouvido, carros, etc.

2.4 INTERFACES GRÁFICAS E A INTERAÇÃO HOMEM-MÁQUINA

2.4.1 IDE Android Studio

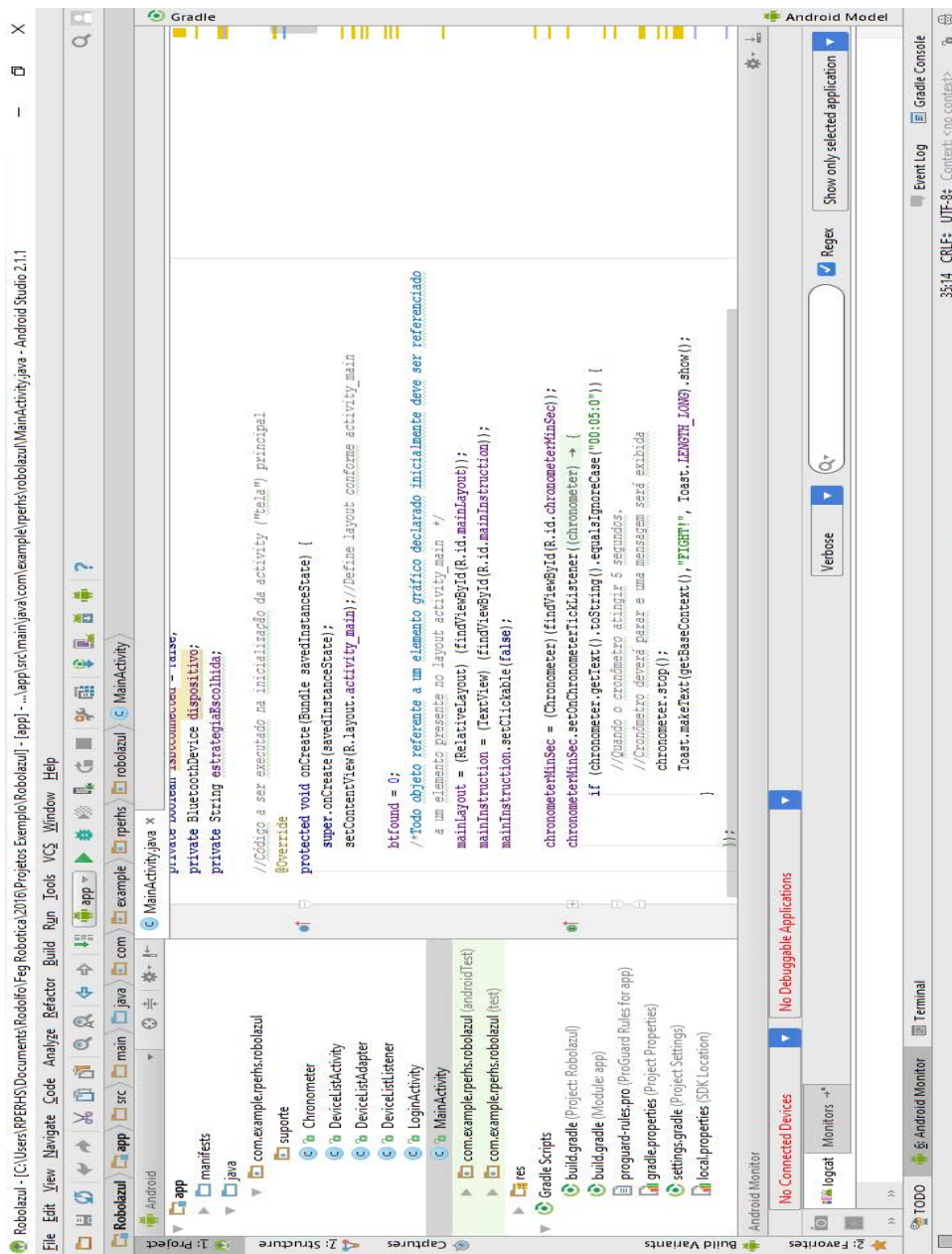
A programação de aplicativos voltados para o sistema operacional *Android* se torna, como já comentado previamente, algo extremamente interativo e de implementação relativamente simples uma vez que os conceitos de engenharia de software, programação de computadores em linguagem do tipo POO (programação orientada a objetos) estejam bem consolidados, isto porque *Android* é uma linguagem do tipo *open source*, onde toda documentação está amplamente disponível e permite personalizações e contribuições ao redor do mundo, para as mais diversas implementações. Todavia, ainda se faz necessário a utilização da ferramenta adequada para a preparação dos códigos e os testes preliminares. Para tanto, optou-se pelo ambiente de desenvolvimento integrado *Android Studio*, tido como o IDE oficial para a programação em *Android*.

Android Studio provides the fastest tools for building apps on every type of Android device. World-class code editing, debugging, performance tooling, a flexible build system, and an instant build/deploy system all allow you to focus on building unique and high quality apps. (Android Studio The Official IDE for Android. Disponível em <<https://developer.android.com/studio/index.html>>. Acesso em 06 de junho de 2016.)

Dentre algumas funcionalidades de destaque do *Android Studio*, cabem ressaltar:

- Edição de códigos de forma inteligente, seguindo códigos de cores e dicas durante a escrita;
- Emulador rápido e eficiente, permitindo o teste bem próximo do real ainda no ambiente virtual, podendo ser configurado como diferentes dispositivos compatíveis com Android;
- Integração com ferramentas de desenvolvimento colaborativo como GitHub.

Figura 1 – Interface do Android Studio



Fonte: Produção do próprio autor

3 DESENVOLVIMENTO (METODOLOGIA APK)

O desenvolvimento de um aplicativo para *smartphones* não difere em nada do desenvolvimento de um software comum. É preciso definir previamente alguns tópicos de elevada relevância para iniciar efetivamente a criação de linhas de código que trarão vida ao software desejado.

As etapas básicas para o desenvolvimento de um software podem ser obtidas através de um sistema simples de perguntas e respostas como:

- Qual a necessidade que o software deverá solucionar?
- Quais os recursos que o software deverá apresentar para atender a necessidade?
- Quais os limites da aplicação? Exemplos: Quantidade de usuários; interação entre usuários, simultaneidade de acessos; distâncias de atuação, etc.
- Quais os recursos se tem disponíveis para implementação do software?
- Qual o cronograma a ser considerado entre o conceito do projeto até sua execução ser necessária livre de falhas?

Uma vez contempladas todas as perspectivas supracitadas, faz-se necessário o desenvolvimento de um cronograma de atividades que permita visualizar as necessidades do projeto em cada etapa de seu desenvolvimento, e assim, possa-se estipular *deadlines* factíveis para que o projeto esteja em uma versão funcional capaz de atender às necessidades da equipe FEG-Robótica. Abaixo, temos o retrato do cronograma considerado para o desenvolvimento do projeto em questão:

Figura 2 – Cronograma do projeto

CRONOGRAMA DO PROJETO						
Conceitualização	Revisão Bibliográfica	Desenvolvimento	Testes	Aperfeiçoamento		
• Avaliação de necessidades (FEG-Robótica) • Avaliação de recursos disponíveis • Definição do hardware e escopo do projeto	• Estudo de linguagens (Java – Android, etc.) • Estudo da evolução tecnológica no mundo • Estudo da tecnologia Bluetooth	• Definição das interfaces gráficas • Desenvolvimento das interfaces gráficas • Desenvolvimento das funcionalidades • Definição dos procedimentos de teste	• Execução dos procedimentos de testes	• Avaliação dos resultados de testes • Aperfeiçoamento das funcionalidades • Reexecução dos procedimentos de testes • Reavaliação dos resultados • Consolidação da versão 1.0 • Treinamento para FEG-Robótica		
Jan/16	Feb/16	Mar/16	Abr/16	Mai/16	Jun/16	Jul/16

Fonte: Produção do próprio autor

3.1 DEFINIÇÃO DAS FUNCIONALIDADES

Como citado no tópico 2.1 DINÂMICAS DO COMBATE DE ROBÔS DE SUMÔ AUTÔNOMO, as funcionalidades do aplicativo proposto para a equipe FEG – Robótica hão de serem capazes de permitirem ao usuário:

- Escolher qual o robô deverá ser controlado, uma vez que a equipe FEG – Robótica, para a qual o aplicativo foi desenvolvido, possui 5 robôs de sumo divididos entre 2 categorias;
- Escolher qual a estratégia o robô deverá assumir para cada um dos 3 rounds, uma vez que a versatilidade de estratégias tem se mostrado historicamente efetiva na busca por melhores resultados;
- Acionamento preciso após 5 segundos da autorização do juiz, já que este é um critério crucial para a aplicação.

Explorando um pouco mais estas funcionalidades, se faz necessário refletir sobre como poderão ser explorados recursos disponíveis para a programação de dispositivos *Android*, claramente entende-se que será necessário escolher entre listas distintas qual o robô a ser considerado, assim como qual estratégia, lembrando que cada robô possui estratégias distintas, portanto, necessária uma referência cruzada dinâmica de acordo com o robô selecionado.

3.2 DEFINIÇÃO DAS APLICAÇÕES (LAYOUT GRÁFICO)

Uma vez estabelecidas quais serão as funcionalidades que o aplicativo deverá executar, é necessário desenhar uma forma gráfica de induzir o usuário à utilização do aplicativo. Tratando-se de uma forma de programação extremamente ligada ao visual já que por trás de qualquer ação executada existe uma interação homem-máquina através de um conjunto de elementos gráficos que comportam diferentes interações com o usuário. Por exemplo, imagine uma chave do tipo *ON-OFF*, ou mais propriamente chamado de “switch” em linguagem Java: Seria adequado associarmos a este componente gráfico a exibição de um relatório textual das últimas ações tomadas pelo usuário? Provavelmente escolher um componente gráfico próprio para a exibição de textos, os chamados *TextViewers*, seria uma solução mais adequada. Desta forma, será apresentado inicialmente o conceito básico para cada componente utilizado segundo a referência *Android Developer*.

Text View: Exibe textos para o usuário, podendo ser permitido a edição de seu conteúdo.

Buttons / ImageButtons: Consiste em um texto ou ícone (podendo ainda ser uma composição de ambos), que informa qual ação ocorrerá quando o usuário tocá-lo.

Switch: São botões de dois estados, onde a interação do usuário consiste em deslizar uma chave entre as duas possibilidades de estado (*On* e *Off*).

RadioButton: Trata-se de um botão de dois estados que pode estar checado ou não; onde checado seria o mesmo que ativo. Quando um *radio Button* não está ativo, o usuário pode pressioná-lo para ativá-lo, todavia de forma contrária ao *CheckBox*, um *radio Button* não pode ser desativado por um novo clique. Para tanto, é comum sua aplicação nos chamados *RadioGroups*, onde uma série de *radio buttons* estão inseridos e apenas um deles pode estar ativo, e para isso, a ativação de um *radio Button* desativa todos os outros inseridos no *radio Group*.

Após tomar conhecimento da funcionalidade básica dos componentes gráficos, ainda é preciso compreender como inseridos nas chamadas *activities*, ou ainda, em termos da língua portuguesa, as chamadas telas do aplicativo. Cada tela possui seu layout gráfico, que nada mais é do que a distribuição arbitrária e definida de seus componentes gráficos. Imagine cada *activity* como um *container*, onde se distribui convenientemente elementos. A distribuição desses elementos pode seguir algumas fórmulas pré-estabelecidas, as quais são identificadas como layouts em linguagem de programação. Abaixo os principais layouts utilizados na programação de aplicativos para sistema operacional *Android*:

LinearLayout (Horizontal): Esta distribuição divide o container em colunas, adicionando sempre o próximo elemento à direita do anterior. Importante ressaltar que as colunas podem ser de larguras distintas.

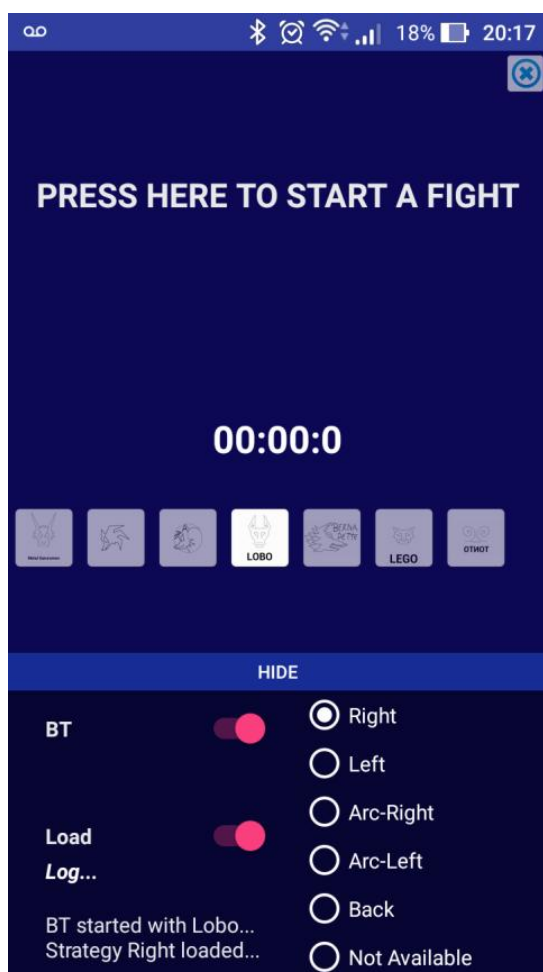
LinearLayout (Vertical): Similar ao caso anterior, porém divide o container em linhas ao invés de colunas. Sendo assim, adicionando sempre o próximo elemento abaixo do componente gráfico anterior.

RelativeLayout: Muito utilizado para designs avançados, esta distribuição adiciona elementos gráficos em posições cartesianas relativas a algum dos componentes já inseridos no layout, sem distinção se acima ou abaixo, à esquerda ou à direita.

Agora, uma vez fixados os conceitos necessários para a criação do layout gráfico das telas (*activities*) do aplicativo, já podemos explorar como as mesmas foram

constituídas para atender às funcionalidades pré-estabelecidas. Abaixo um retrato de como o aplicativo foi constituído graficamente:

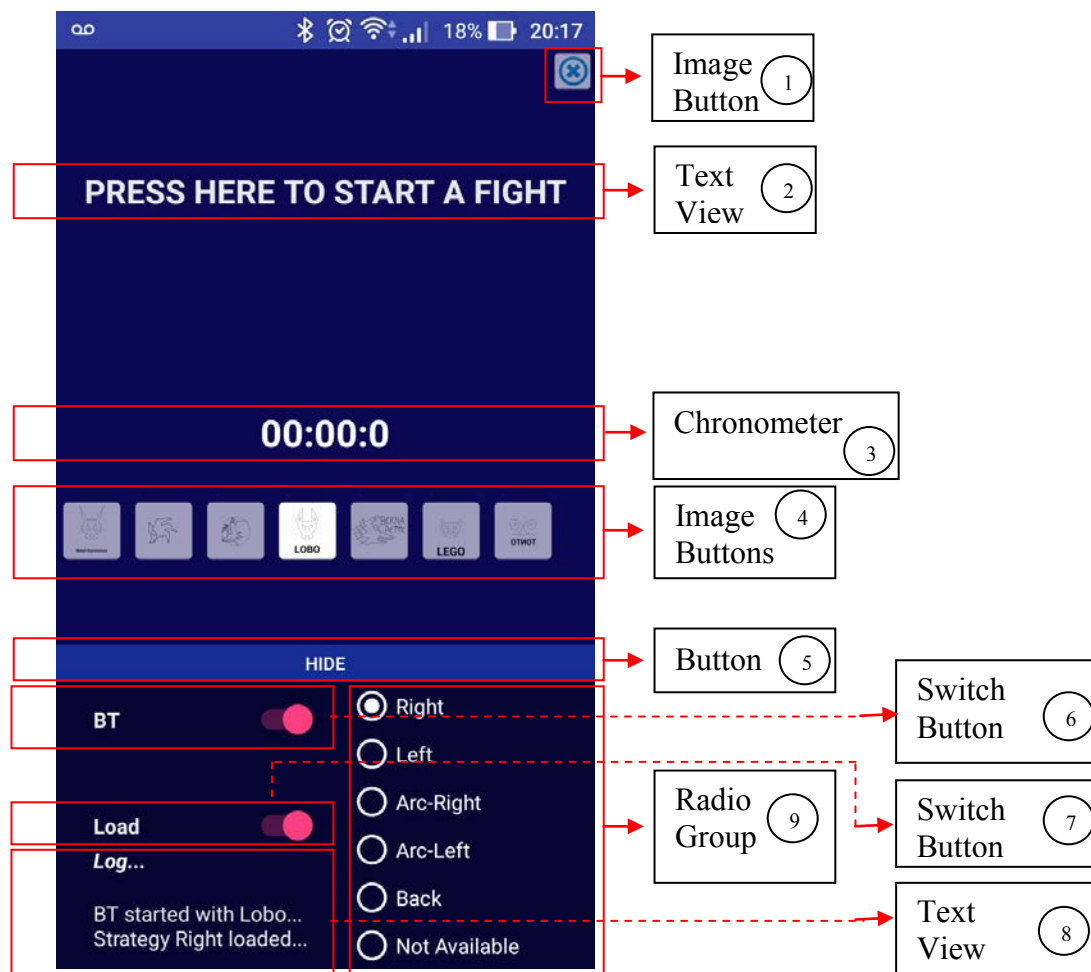
Figura 3 – Tela principal do aplicativo



Fonte: Produção do próprio autor

Para iniciar a análise, primeiramente identifiquemos quais os componentes estão presentes no layout.

Figura 4 – Break-down da tela principal



Fonte: Produção do próprio autor

Tabela 1 – Componentes gráficos

Número do componente	Referência no programa (Ver anexo A)
1 – Image Button	closeButton
2 – Text View	mainInstruction
3 – Chronometer	chronometerMinSec
4 – Image Buttons (Text View)	robot1, robot2, robot3, ... , robot7 (goneRobot*)
5 – Button (Button)	hideOptions (showOptions*)
6 – Switch Button	BTcommunicationState
7 – Switch Button	loadProgram
8 – Text View	Log
9 – Radio Group Radio Buttons	strategies strategy1, strategy2, ... , strategy6

Fonte: Produção do próprio autor

Nota: *Alguns elementos estão ocultos, ou ainda disfarçados, na maior parte da execução do aplicativo como o botão showOptions que reveza atuação com o botão hideOptions, e também os TextViews goneRobot1, goneRobot2, ... , goneRobot7 responsáveis por causar a impressão de foco exclusivo em um robô de cada vez, representado por cada um dos ImageButton robot1, robot2, ... , robot7.

Resta compreender mais profundamente a funcionalidade associada a cada componente gráfico, princípio básico para que o código associado possa ser bem interpretado. As funcionalidades associadas a cada elemento gráfico pode ser encontrada na tabela abaixo.

Tabela 2 – Componentes gráficos x Funcionalidade

Número do componente	Funcionalidade
1	Encerrar o aplicativo
2	Fornecer informações a respeito do próximo passo a ser tomado pelo usuário para o adequado funcionamento do aplicativo
3	Apresentar a contagem de 5s com precisão de décimos de segundo
4 (4)	Através de uma identidade visual criada para cada robô, selecionar qual robô deverá ser considerado para o próximo acionamento (Realizar uma ocultação parcial do elemento gráfico principal afim de causar a impressão de foco apenas para o robô selecionado)
5 (5)	Ocultar as opções de acionamento – abaixo do elemento 5 (Exibir as opções de acionamento)
6	Estabelecer comunicação Bluetooth com o dispositivo adequado a ser exibido numa lista de dispositivos pareados no momento da comutação da chave
7	Preparar o caracter a ser enviado durante a comunicação de acordo com a estratégia selecionada
8	Exibir um relatório das ações tomadas pelo usuário durante execução do aplicativo
9	Grupo de radioButtons que representam as estratégias plausíveis de serem executadas por cada robô. A escolha desta estratégia é a função do elemento.

Fonte: Produção do próprio autor

Uma vez que o código fonte, desenvolvido originalmente para este trabalho (exceção à classe `Chronometer.java` disponibilizada em <<https://github.com/f2prateek/Stopwatch/blob/master/src/com/psrivastava/stopwatch/Chronometer.java>>) encontra-se comentado e totalmente disponível no Apêndice A, limitaremos nossa abordagem de códigos para o tratamento dos eventos oriundos da interatividade com o usuário a um fluxograma que descreve substancialmente a sequência lógica para que o aplicativo execute adequadamente suas funcionalidades, as quais são: o envio de um caractere específico, de acordo com a estratégia escolhida, para dispositivo *Bluetooth* previamente pareado e escolhido pelo usuário durante a sequência de comandos do aplicativo, de tal forma que o robô escolhido após um tempo não inferior a 5 segundos inicie sua ação na arena de combate (dojô), devendo ser desligado também remotamente imediatamente após comando do usuário.

Figura 5 – Fluxograma de funcionamento



Fonte: Produção do próprio autor

3.3 TESTES DE VALIDAÇÃO

Todo projeto necessita de ao menos 3 fases antecessoras ao seu lançamento para uso e consumo conforme idealização inicial. Estas fases comportam a elaboração do conceito e cronograma, o desenvolvimento das atividades e etapas propostas, e uma terceira fase responsável por validar se o projeto desenvolvido atende ao conceito de sua formação. Esta terceira fase podem ocorrer incluir simulações computacionais em ambiente virtual adequado, mas certamente terá seu maior peso de julgamento sobre a validação empírica, através de uma sequência ordenada de testes reais, que disponibilizem desta forma uma segurança aos projetistas de que o produto desenvolvido poderá ser aplicado de forma segura, com resultado positivo garantido, e evitando que qualquer comportamento imprevisível tenha uma origem num erro sistêmico de projeto. CRESPO et al. (2004) defende que “Ao se adotar uma técnica de teste (Funcional ou Estrutural) é necessário escolher um critério para a elaboração dos casos de teste com a finalidade de testar os elementos do software”. No caso de robôs autônomos de combate na categoria sumô, é essencial que qualquer comportamento do robô nos momentos de luta corresponda à interpretação esperada. Indo mais além, quando se faz referência a aplicação *mobile* que deverá ser capaz de escolher o robô a ser comandado, sua estratégia e com exatidão enviar um comando assertivo para iniciar e interromper a luta, é essencial que se comprove através de testes próximos do cenário real, que tudo o que se espera, acontecerá sem interferências provenientes de erros associados ao projeto; ainda que interferências externas possam ocorrer e inviabilizar a execução de qualquer projeto.

Para a execução dos testes de validação do aplicativo *mobile*, decidiu-se, em conjunto com a equipe FEG-Robótica, um formato de validação 100% prática, sem considerar simulações computacionais. Para tanto, elencaram-se inicialmente todas as funcionalidades que deveriam ser colocadas à prova:

- Identidade visual para associação dos comandos e escolhas a serem realizadas para cada um dos sete robôs da equipe;
- Travas do aplicativo para que o qualquer comando só seja disparado após o usuário seguir todas as etapas corretas para utilizar o aplicativo;
- Efetiva transmissão de conteúdo entre o *smartphone* e os módulos *Bluetooth* utilizados para receber a mensagem e colocá-las à disposição dos robôs.
- *Range* de aplicação e interferências provenientes da distância, localidade e multiplicidade de comunicações via ondas.

Para tanto, concebeu-se uma metodologia, ou ainda um procedimento, de teste para cada um dos tópicos contemplados acima. Abaixo, temos o retrato de cada um destes procedimentos desenvolvidos exclusivamente para validação empírica do aplicativo *mobile* desenvolvido para o acionamento via *Bluetooth* dos robôs de combate da categoria sumo da equipe FEG-Robótica:

Figura 6 – Procedimento de teste (identidade visual)



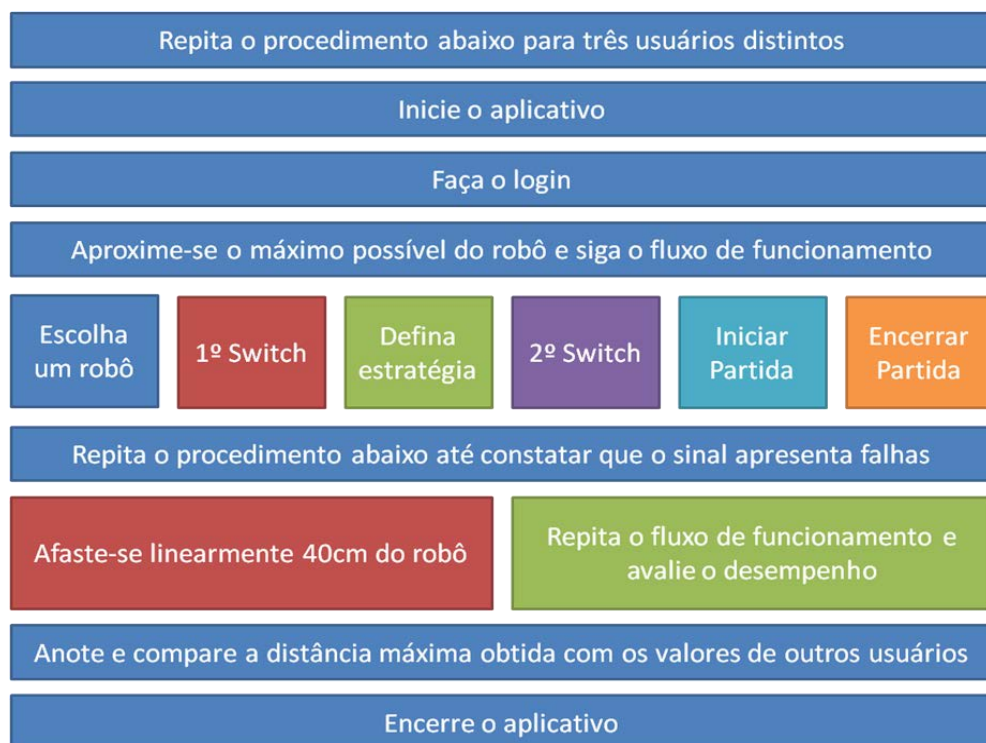
Fonte: Produção do próprio autor

Figura 7 – Procedimento de teste (travas funcionais e funcionamento)



Fonte: Produção do próprio autor

Figura 8 – Procedimento de teste (range de aplicação)



Fonte: Produção do próprio autor

Adiante, temos os resultados dos testes realizados para validação descritos em forma de tabela.

Tabela 3 – Resultados de teste (login)

Procedimento de teste - login					
Smartphone	Login		Senha		Resultado
	Correto	Incorreto	Correto	Incorreto	
1	x			x	Aprovado
		x	x		Aprovado
	x		x		Aprovado
2	x			x	Aprovado
		x	x		Aprovado
	x		x		Aprovado
3	x			x	Aprovado
		x	x		Aprovado
	x		x		Aprovado

Fonte: Produção do próprio autor

Tabela 4 – Resultados de teste (identidade visual)

Procedimento de teste - identidade visual					
Smartphone	Robô	Logo	Texto	Opções	Resultado
1	Sonic	OK	OK	OK	Aprovado
	Tails	OK	OK	OK	Aprovado
	Zoiudinho	OK	OK	OK	Aprovado
	Lobo	OK	OK	OK	Aprovado
	Bernadete	OK	OK	OK	Aprovado
	MetalGarurumon	OK	OK	OK	Aprovado
	OTNOT	OK	OK	OK	Aprovado
2	Sonic	OK	OK	OK	Aprovado
	Tails	OK	OK	OK	Aprovado
	Zoiudinho	OK	OK	OK	Aprovado
	Lobo	OK	OK	OK	Aprovado
	Bernadete	OK	OK	OK	Aprovado
	MetalGarurumon	OK	OK	OK	Aprovado
	OTNOT	OK	OK	OK	Aprovado
3	Sonic	OK	OK	OK	Aprovado
	Tails	OK	OK	OK	Aprovado
	Zoiudinho	OK	OK	OK	Aprovado
	Lobo	OK	OK	OK	Aprovado
	Bernadete	OK	OK	OK	Aprovado
	MetalGarurumon	OK	OK	OK	Aprovado
	OTNOT	OK	OK	OK	Aprovado

Fonte: Produção do próprio autor

Tabela 5 – Resultados de teste (travas e funcionalidades)

Procedimento de teste - travas e funcionalidades							
Usuário	1º Switch	Estratégia	2º Switch	Iniciar	Encerrar	Esperado	Resultado
A	1	2	3	4	5	ok	ok
	1	3	2	4	5	não ok	não ok
	2	3	1	4	5	não ok	não ok
	3	1	2	4	5	não ok	não ok
B	1	2	3	4	5	ok	ok
	1	4	3	2	5	não ok	não ok
	2	1	3	4	5	não ok	não ok
	3	4	5	1	2	não ok	não ok
C	1	2	3	4	5	ok	ok
	2	3	1	4	5	não ok	não ok
	5	2	3	1	4	não ok	não ok
	4	2	1	3	5	não ok	não ok

Fonte: Produção do próprio autor

Tabela 6 – Resultados de teste (range de aplicação)

Procedimento de testes - range de aplicação								
Usuário	Distância (m)	Resultado	Usuário	Distância (m)	Resultado	Usuário	Distância (m)	Resultado
A	0	ok	B	0	ok	C	0	ok
	0,4	ok		0,4	ok		0,4	ok
	0,8	ok		0,8	ok		0,8	ok
	1,2	ok		1,2	ok		1,2	ok
	1,6	ok		1,6	ok		1,6	ok
	2	ok		2	ok		2	ok
	2,4	ok		2,4	ok		2,4	ok
	2,8	ok		2,8	ok		2,8	ok
	3,2	ok		3,2	ok		3,2	ok
	3,6	ok		3,6	ok		3,6	ok
	4	ok		4	ok		4	ok
	4,4	ok		4,4	ok		4,4	ok
	4,8	ok		4,8	ok		4,8	ok
	5,2	ok		5,2	ok		5,2	ok
	5,6	ok		5,6	ok		5,6	ok
	6	ok		6	ok		6	ok
	6,4	ok		6,4	ok		6,4	ok
	6,8	ok		6,8	ok		6,8	ok
	7,2	ok		7,2	ok		7,2	ok
	7,6	ok		7,6	ok		7,6	ok
	8	ok		8	ok		8	ok
	8,4	não ok		8,4	ok		8,4	não ok

Fonte: Produção do próprio autor

Desta forma, concluiu-se que o aplicativo estava em condições de ser aprovado segundo as necessidades da equipe FEG-Robótica para um *range* de aplicação de até 8m de distância, podendo ser assim considerado um software bem sucedido segundo o critério de PRESSMAN (2011, p. 7) que expõe: “Quando um software é bem-sucedido – atende às necessidades dos usuários, opera perfeitamente durante um longo período, é fácil de modificar e, mais fácil ainda de utilizar – ele é realmente capaz de mudar as coisas para melhor.”

4 CONCLUSÃO

Tem-se como certeza de que a robótica trata-se de um vasto campo de estudo, o qual engloba diversificadas formas de atuação: partindo da automatização de processos industriais até mesmo a inteligência artificial inserida no campo de entretenimento através de combate de robôs autônomos. Deve-se notar, todavia que independentemente da aplicação, a robótica estará sempre conectada a algum tipo de interface com o mundo humano; comunicando-se para que haja sinergia entre os objetivos do usuário e a tarefa executada pela estrutura robótica que o atende. Comunicar-se, assim como no mundo das relações humanas, é algo de certa complexidade e pode ocorrer através de variadas tratativas, ainda que no mundo computacional dos robôs a essência do alfabeto seja reduzida a zeros e uns distribuídos de forma ordenada.

Ainda que se possa entender o procedimento lógico para adaptar comandos expressos por sequências de *bits* com níveis lógicos altos e baixos, a utilização de ferramentas adequadas podem minimizar os esforços e permitir uma fácil saída para a complexa questão de comunicação. Como ferramentas adequadas para o universo da programação computacional de dispositivos, deve-se citar desde o ambiente virtual de criação das linhas de códigos, como também as tecnologias empregadas para a efetiva transmissão das mensagens (*Android Studio* e *Bluetooth*, no caso do desenvolvimento deste trabalho). Inúmeras são as possibilidades, alocadas segundo uma série de vantagens e desvantagens, inclusive financeiras, para as quais mais uma vez se prova a célebre constatação de que a ferramenta mais adequada é certamente aquela que se sabe utilizar, com um aditivo heurístico, que reforça a estratégia de se dominar a necessidade, para então sim buscar a implementação de uma solução dedicada que não gere excessivo desperdício de recursos, e muito menos deixe de cumprir a proposta.

Como último tópico, pode-se também concluir que há uma grande relevância em acompanhar os desenvolvimentos tecnológicos da sociedade, pois em mundo que se dinamiza a cada dia mais, a incorporação de recursos adentra o campo dos objetos pessoais, e num simples *smartphone*, reside uma capacidade significativa de hardware e software comuns a grupos de ilimitada quantidade de usuários, capaz de simplificar e auxiliar na concentração de funcionalidades disponíveis ao toque de um dedo, podendo oferecer interação e bons resultados com algo já mais consagrado na história como uma lista de contatos telefônicos, mas também para algo mais atual como o controle de acionamento remoto para robôs autônomos de sumô.

REFERÊNCIAS

- GOOGLE INC. **Android studio the official ide for android**. Disponível em: <<https://developer.android.com/studio/index.html>>. Acesso em: 06 jun. 2016.
- CHASE, O. **Sistemas Embarcados**. [S.l.]: SBA Jovem, 2007. 7p.
- COUTINHO, G. L. **A Era dos smartphones: um estudo exploratório sobre o uso dos smartphones no Brasil**. 2014. 67p. Monografia (Bacharelado em Publicidade e Propaganda) – Universidade de Brasília, Brasília, 2014.
- CRESPO, A. N. et al. **Uma metodologia para teste de software no contexto da melhoria de processo**. Brasília: 3º SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 2004.
- FERNANDES, E. da R.; ZITKE, V. A. A Evolução da técnica e o surgimento da tecnologia no contexto econômico e educacional. In: CONGRESSO INTERNACIONAL DE HISTÓRIA DA UFG, 3., 2012, Jataí. **Anais...** Jataí: História e Diversidade Cultural, 2012.
- FINCOTTO, M. A.; SANTOS, M. T. P. Um foco na plataforma android. **Revista T.I.S.: Tecnologias, infraestrutura e software**, São Carlos, v. 3, n. 2, p. 151-161, mai-ago. 2014.
- Bluetooth SIG, Inc. **How does bluetooth works?** Disponível em: <<https://www.bluetooth.com/what-is-bluetooth-technology/bluetooth-technology-basics>>. Acesso em: 06 jun. 2016.
- LEMO, A. Celulares, funções pós-midiáticas, cidade e mobilidade. **Urbe: Revista Brasileira de Gestão Urbana**, v. 2, n. 2, p. 155-166, 2010.
- MAIA, D. V. de A. **Automação industrial e robótica**. Universidade Federal do Rio Grande do Norte. 2011.5p.
- MEHL, E. L. de M. **Sistemas eletrônicos embarcados**. Universidade Federal do Paraná. 2011. 18p.
- MIRANDA, A. L. **Da natureza da tecnologia: uma análise filosófica sobre as dimensões ontológica, epistemológica e axiológica da tecnologia moderna**. 2002. 161f. Dissertação (Programa de Pós-Graduação em Tecnologia) – Centro Federal de Educação Tecnológica do Paraná, Curitiba, 2002.
- PRESSMAN, R. S. **Engenharia de software: uma abordagem profissional**. Porto Alegre: AMGH Editora, 2011.
- RUSSEL, S.; NORVIG, P. **Inteligência artificial**. Rio de Janeiro: Elsevier, 2004.
- SILVA, D. F. **Sistema de comunicação bluetooth utilizando microcontrolador**. 2009. 74p. Trabalho de conclusão de curso (Graduação em Engenharia da Computação) – Escola Politécnica de Pernambuco, Recife, 2009.
- TORRES, Cláudio. **A Bíblia do marketing digital**. São Paulo: Novatec editora, 2009.

BIBLIOGRAFIA CONSULTADA

BLOCH, J. **Effective Java**. 2. ed. Santa Clara: Addison-Wesley, 2008.

Prateek Srivastava. **Chronometer java**. Disponível em:
<<https://github.com/f2prateek/Stopwatch/blob/master/src/com/psrivastava/stopwatch/Chronometer.java>>. Acesso em: 06 jun. 2016.

DEITEL, P.; DEITEL, H.; DEITEL, A. **Android para programadores: uma abordagem baseada em aplicativos**. 2. ed. São Paulo: Bookman, 2015.

SANTOS, F. M. V. **Robótica Industrial**. Aveiro: Universidade de Aveiro, 2003.

SILVEIRA, R. M. C. F. **Ciência e tecnologia: transformando a relação do ser humano com o mundo**. Santa Catarina: NEPET – Núcleo de Estudos e Pesquisas em Educação Tecnológica/ UFSC, 2005.

TCHAPEK, K. **A fábrica de robôs**. São Paulo: Hedra, 2010.

APÊNDICE A – Linhas de código

Código de desenvolvimento gráfico do APK Robolazul

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/
    apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:background="#0c0853"
6      android:nestedScrollingEnabled="false"
7      android:textAlignment="center"
8      android:id="@+id/mainLayout">
9
10     <com.example.rperhs.robolazul.Chronometer
11         android:layout_width="wrap_content"
12         android:layout_height="wrap_content"
13         android:id="@+id/chronometerMinSec"
14         android:layout_above="@+id/mainActivityCenter"
15         android:layout_centerVertical="true"
16         android:layout_centerHorizontal="true"
17         android:layout_marginBottom="25dp"
18         android:textColor="#ffffff"
19         android:textStyle="bold"
20         android:textSize="25dp"/>
21
22     <TextView
23         android:layout_width="wrap_content"
24         android:layout_height="wrap_content"
25         android:textAppearance="?android:attr/
    textAppearanceSmall"
26         android:text="Small Text"
27         android:id="@+id/mainActivityCenter"
28         android:layout_centerVertical="true"
29         android:layout_centerHorizontal="true"
30         android:visibility="invisible" />
31
32     <TextView
33         android:layout_width="match_parent"
34         android:layout_height="wrap_content"
35         android:gravity="center_horizontal"
36         android:textAppearance="?android:attr/
    textAppearanceLarge"
37         android:text="CHOOSE YOUR ROBOT"
38         android:id="@+id/mainInstruction"
39         android:textStyle="bold"
40         android:textColor="#e8e9ea"
41         android:layout_alignParentTop="true"
42         android:layout_centerHorizontal="true"

```

```
43         android:layout_marginTop="80dp"
44         android:onClick="startRobot"
45         android:clickable="true" />
46
47     <RelativeLayout
48         android:layout_width="match_parent"
49         android:layout_height="match_parent"
50         android:background="#64000000"
51         android:layout_alignParentRight="true"
52         android:layout_alignParentEnd="true"
53         android:id="@+id/optionsLayout"
54         android:layout_below="@+id/hideOptions">
55
56         <Switch
57             android:layout_width="150dp"
58             android:layout_height="30dp"
59             android:id="@+id/BTcommunicationState"
60             android:layout_alignParentTop="true"
61             android:layout_alignParentLeft="true"
62             android:layout_alignParentStart="true"
63             android:text="BT"
64             android:textColor="#e8e9ea"
65             android:layout_marginLeft="25dp"
66             android:layout_marginTop="10dp"
67             android:textStyle="bold" />
68
69         <ScrollView
70             android:layout_width="match_parent"
71             android:layout_height="match_parent"
72             android:id="@+id/scrollView"
73             android:layout_gravity="center_horizontal"
74             android:layout_toRightOf="@+id/
BTcommunicationState"
75             android:layout_toEndOf="@+id/
BTcommunicationState" >
76
77             <RadioGroup
78                 android:orientation="vertical"
79                 android:id="@+id/strategies"
80                 android:layout_width="match_parent"
81                 android:layout_height="match_parent"
82                 android:layout_toEndOf="@+id/
BTcommunicationState"
83                 android:layout_toRightOf="@+id/
BTcommunicationState"
```

Código de desenvolvimento gráfico do APK Robolazul

```

84         android:layout_alignParentTop="true"
85         android:gravity="center_vertical|
center_horizontal">
86
87         <RadioButton
88             android:layout_width="wrap_content"
89             android:layout_height="wrap_content"
90             android:text="Strategy 1"
91             android:id="@+id/strategy1"
92             android:textColor="#ffffff"
93             android:enabled="true"
94             android:clickable="true"
95             android:checked="true"
96             android:buttonTint="#ffffff"
97             android:gravity="left|center_vertical
"
98             android:layout_gravity="left"
99             android:layout_marginLeft="20dp" />
100
101         <RadioButton
102             android:layout_width="wrap_content"
103             android:layout_height="wrap_content"
104             android:text="Strategy 2"
105             android:id="@+id/strategy2"
106             android:textColor="#ffffff"
107             android:enabled="true"
108             android:buttonTint="#ffffff"
109             android:gravity="left|center_vertical
"
110             android:layout_gravity="left"
111             android:layout_marginLeft="20dp" />
112
113         <RadioButton
114             android:layout_width="wrap_content"
115             android:layout_height="wrap_content"
116             android:text="Strategy 3"
117             android:id="@+id/strategy3"
118             android:textColor="#ffffff"
119             android:enabled="true"
120             android:buttonTint="#ffffff"
121             android:gravity="left|center_vertical
"
122             android:layout_marginLeft="20dp"
123             android:layout_gravity="left" />
124

```


Código de desenvolvimento gráfico do APK Robolazul

```

125         <RadioButton
126             android:layout_width="wrap_content"
127             android:layout_height="wrap_content"
128             android:text="Strategy 4"
129             android:id="@+id/strategy4"
130             android:textColor="#ffffff"
131             android:enabled="true"
132             android:buttonTint="#ffffff"
133             android:gravity="left|center_vertical"
134
135             android:layout_marginLeft="20dp"
136             android:layout_gravity="left" />
137
138         <RadioButton
139             android:layout_width="wrap_content"
140             android:layout_height="wrap_content"
141             android:text="Strategy 5"
142             android:id="@+id/strategy5"
143             android:textColor="#ffffff"
144             android:singleLine="true"
145             android:checked="false"
146             android:clickable="true"
147             android:enabled="true"
148             android:buttonTint="#ffffff"
149             android:gravity="left|center_vertical"
150
151             android:layout_gravity="left"
152             android:layout_marginLeft="20dp" />
153
154         <RadioButton
155             android:layout_width="wrap_content"
156             android:layout_height="wrap_content"
157             android:text="Strategy 6"
158             android:id="@+id/strategy6"
159             android:textColor="#ffffff"
160             android:singleLine="true"
161             android:checked="false"
162             android:clickable="true"
163             android:enabled="true"
164             android:buttonTint="#ffffff"
165             android:gravity="left|center_vertical"
166
167             android:layout_marginLeft="20dp"
168             android:layout_gravity="left" />

```

Código de desenvolvimento gráfico do APK Robolazul

```

167         </RadioGroup>
168     </ScrollView>
169
170     <Switch
171
172         android:id="@+id/loadProgram"
173         android:layout_centerVertical="true"
174         android:layout_alignParentLeft="true"
175         android:layout_alignParentStart="true"
176         android:textColor="#e8e9ea"
177         android:text="Load"
178         android:layout_marginLeft="25dp"
179         android:layout_width="150dp"
180         android:layout_height="45dp"
181         android:textStyle="bold"
182         android:layout_marginTop="10dp"
183         android:layout_marginBottom="10dp" />
184
185     <ScrollView
186         android:layout_width="wrap_content"
187         android:layout_height="wrap_content"
188         android:id="@+id/logScroll"
189         android:layout_alignParentBottom="true"
190         android:layout_alignLeft="@+id/loadProgram"
191         android:layout_alignStart="@+id/loadProgram"
192         android:layout_below="@+id/loadProgram"
193         android:foregroundGravity="top">
194
195         <TextView
196             android:layout_width="wrap_content"
197             android:layout_height="match_parent"
198             android:textAppearance="?android:attr/
199 textAppearanceSmall"
200             android:id="@+id/logActions"
201             android:layout_alignParentBottom="true"
202             android:layout_below="@+id/log"
203             android:layout_alignRight="@+id/log"
204             android:textColor="#e8e9ea"
205             android:gravity="top|start" />
206     </ScrollView>
207
208     <TextView
209         android:layout_width="wrap_content"
210         android:layout_height="wrap_content"
211         android:textAppearance="?android:attr/

```

```

210 textAppearanceSmall
211     android:text="Log..."
212     android:id="@+id/Log"
213     android:layout_above="@+id/logScroll"
214     android:layout_alignLeft="@+id/logScroll"
215     android:layout_alignStart="@+id/logScroll"
216     android:textColor="#ffffff"
217     android:textStyle="bold|italic"
218     android:layout_marginTop="5dp" />
219
220 </RelativeLayout>
221
222 <Button
223     android:layout_width="match_parent"
224     android:layout_height="25dp"
225     android:text="SHOW"
226     android:id="@+id/showOptions"
227     android:background="#832452dd"
228     android:textSize="12dp"
229     android:textColor="#ffffff"
230     android:onClick="doShowOptions"
231     android:visibility="gone"
232     android:layout_alignParentBottom="true" />
233
234 <HorizontalScrollView
235     android:layout_width="match_parent"
236     android:layout_height="wrap_content"
237     android:id="@+id/horizontalScrollView"
238     android:layout_alignTop="@+id/mainActivityCenter"
239     android:layout_centerHorizontal="true" >
240
241     <RelativeLayout
242         android:orientation="horizontal"
243         android:layout_width="match_parent"
244         android:layout_height="match_parent">
245
246         <ImageButton
247             android:layout_width="wrap_content"
248             android:layout_height="wrap_content"
249             android:id="@+id/robot1"
250             android:background="@mipmap/robot1"
251             android:layout_alignParentTop="true"
252             android:layout_alignParentLeft="true"
253             android:layout_alignParentStart="true" />
254

```


Código de desenvolvimento gráfico do APK Robolazul

```

255         <TextView
256             android:layout_width="wrap_content"
257             android:layout_height="wrap_content"
258             android:textAppearance="?android:attr/
textAppearanceMedium"
259             android:id="@+id/goneRobot1"
260             android:background="#640c0853"
261             android:hint="
"
262             android:layout_alignParentTop="true"
263             android:layout_alignParentLeft="true"
264             android:layout_alignParentStart="true"
265             android:layout_alignBottom="@+id/robot1"
/>
266
267         <ImageButton
268             android:layout_width="wrap_content"
269             android:layout_height="wrap_content"
270             android:id="@+id/robot2"
271             android:background="@mipmap/robot2"
272             android:layout_alignParentTop="true"
273             android:layout_toRightOf="@+id/robot1"
274             android:layout_toEndOf="@+id/robot1" />
275
276
277         <TextView
278             android:layout_width="wrap_content"
279             android:layout_height="wrap_content"
280             android:textAppearance="?android:attr/
textAppearanceMedium"
281             android:id="@+id/goneRobot2"
282             android:background="#640c0853"
283             android:layout_alignParentTop="true"
284             android:layout_alignBottom="@+id/robot2"
285             android:layout_toLeftOf="@+id/robot3"
286             android:hint="
" />
287
288         <ImageButton
289             android:layout_width="wrap_content"
290             android:layout_height="wrap_content"
291             android:id="@+id/robot3"
292             android:background="@mipmap/robot3"
293             android:layout_alignParentTop="true"
294             android:layout_toRightOf="@+id/robot2"
295             android:layout_toEndOf="@+id/robot2" />
296

```

Código de desenvolvimento gráfico do APK Robolazul

```

297         <TextView
298             android:layout_width="wrap_content"
299             android:layout_height="wrap_content"
300             android:textAppearance="?android:attr/
textAppearanceMedium"
301             android:id="@+id/goneRobot3"
302             android:background="#640c0853"
303             android:layout_alignParentTop="true"
304             android:layout_alignBottom="@+id/robot3"
305             android:layout_toLeftOf="@+id/robot4"
306             android:hint="                " />
307
308         <ImageButton
309             android:layout_width="wrap_content"
310             android:layout_height="wrap_content"
311             android:id="@+id/robot4"
312             android:background="@mipmap/robot4"
313             android:layout_alignParentTop="true"
314             android:layout_toRightOf="@+id/robot3"
315             android:layout_toEndOf="@+id/robot3" />
316
317         <TextView
318             android:layout_width="wrap_content"
319             android:layout_height="wrap_content"
320             android:textAppearance="?android:attr/
textAppearanceMedium"
321             android:id="@+id/goneRobot4"
322             android:background="#640c0853"
323             android:layout_alignParentTop="true"
324             android:layout_alignBottom="@+id/robot4"
325             android:layout_toLeftOf="@+id/robot5"
326             android:hint="                " />
327
328         <ImageButton
329             android:layout_width="wrap_content"
330             android:layout_height="wrap_content"
331             android:id="@+id/robot5"
332             android:background="@mipmap/robot5"
333             android:layout_alignParentTop="true"
334             android:layout_toRightOf="@+id/robot4"
335             android:layout_toEndOf="@+id/robot4" />
336
337         <TextView
338             android:layout_width="wrap_content"
339             android:layout_height="wrap_content"

```

```

340         android:textAppearance="?android:attr/
textAppearanceMedium"
341         android:id="@+id/goneRobot5"
342         android:background="#640c0853"
343         android:layout_alignParentTop="true"
344         android:layout_alignBottom="@+id/robot5"
345         android:layout_toRightOf="@+id/robot4"
346         android:hint="                " />
347
348     <ImageButton
349         android:layout_width="wrap_content"
350         android:layout_height="wrap_content"
351         android:id="@+id/robot6"
352         android:background="@mipmap/robot6"
353         android:layout_alignParentTop="true"
354         android:layout_toRightOf="@+id/robot5"
355         android:layout_toEndOf="@+id/robot5" />
356
357     <TextView
358         android:layout_width="wrap_content"
359         android:layout_height="wrap_content"
360         android:textAppearance="?android:attr/
textAppearanceMedium"
361         android:id="@+id/goneRobot6"
362         android:background="#640c0853"
363         android:layout_alignParentTop="true"
364         android:layout_alignBottom="@+id/robot6"
365         android:layout_toRightOf="@+id/robot5"
366         android:hint="                " />
367
368     <ImageButton
369         android:layout_width="wrap_content"
370         android:layout_height="wrap_content"
371         android:id="@+id/robot7"
372         android:background="@mipmap/robot7"
373         android:layout_alignParentTop="true"
374         android:layout_toRightOf="@+id/robot6"
375         android:layout_toEndOf="@+id/robot6" />
376
377     <TextView
378         android:layout_width="wrap_content"
379         android:layout_height="wrap_content"
380         android:textAppearance="?android:attr/
textAppearanceMedium"
381         android:id="@+id/goneRobot7"

```


Código de desenvolvimento gráfico do APK Robolazul

```

382         android:background="#640c0853"
383         android:layout_alignParentTop="true"
384         android:layout_alignBottom="@+id/robot7"
385         android:layout_toRightOf="@+id/robot6"
386         android:hint="                " />
387
388     </RelativeLayout>
389 </HorizontalScrollView>
390
391 <Button
392     android:layout_width="wrap_content"
393     android:layout_height="25dp"
394     android:text="HIDE"
395     android:id="@+id/hideOptions"
396     android:layout_below="@+id/chronometerMinSec"
397     android:layout_marginTop="100dp"
398     android:background="#832452dd"
399     android:layout_alignParentRight="true"
400     android:layout_alignParentEnd="true"
401     android:layout_alignParentLeft="true"
402     android:layout_alignParentStart="true"
403     android:textSize="12dp"
404     android:textColor="#ffffff"
405     android:onClick="doHideOptions" />
406
407 <ImageButton
408     android:layout_width="30dp"
409     android:layout_height="30dp"
410     android:id="@+id/closeButton"
411     android:background="@mipmap/close"
412     android:layout_alignParentTop="true"
413     android:layout_alignParentEnd="true"
414     android:onClick="closeActivity" />
415
416 <TextView
417     android:layout_width="wrap_content"
418     android:layout_height="wrap_content"
419     android:textAppearance="?android:attr/
textAppearanceMedium"
420     android:id="@+id/textView"
421     android:background="#640c0853"
422     android:hint="                "
423     android:layout_alignBottom="@+id/closeButton"
424     android:layout_alignStart="@+id/closeButton"
425     android:layout_alignParentTop="true"

```

Código de desenvolvimento gráfico do APK Robolazul

```
426         android:layout_alignParentEnd="true" />
427
428     </RelativeLayout>
429
```


Código de desenvolvimento gráfico do APK Robolazul

```

1 <LinearLayout xmlns:android="http://schemas.android.com/
  apk/res/android"
2     xmlns:tools="http://schemas.android.com/tools"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent"
5     android:gravity="center_horizontal"
6     android:orientation="vertical"
7     android:paddingBottom="@dimen/activity_vertical_margin"
8     "
9     android:paddingLeft="@dimen/activity_horizontal_margin"
10    "
11    android:paddingRight="@dimen/
  activity_horizontal_margin"
12    android:paddingTop="@dimen/activity_vertical_margin"
13    tools:context="com.example.rperhs.robolazul.
  LoginActivity">
14
15    <!-- Login progress -->
16    <ProgressBar
17        android:id="@+id/login_progress"
18        style="?android:attr/progressBarStyleLarge"
19        android:layout_width="wrap_content"
20        android:layout_height="wrap_content"
21        android:layout_marginBottom="8dp"
22        android:visibility="gone" />
23
24    <ScrollView
25        android:id="@+id/login_form"
26        android:layout_width="match_parent"
27        android:layout_height="match_parent">
28
29        <LinearLayout
30            android:id="@+id/email_login_form"
31            android:layout_width="match_parent"
32            android:layout_height="wrap_content"
33            android:orientation="vertical">
34
35            <android.support.design.widget.TextInputLayout
36                android:layout_width="match_parent"
37                android:layout_height="wrap_content">
38
39                <AutoCompleteTextView
40                    android:id="@+id/email"
41                    android:layout_width="match_parent"
42                    android:layout_height="wrap_content"

```

Código de desenvolvimento gráfico do APK Robolazul

```

41         android:hint="@string/prompt_email"
42         android:inputType="textEmailAddress"
43         android:maxLines="1"
44         android:singleLine="true" />
45
46     </android.support.design.widget.
TextInputLayout>
47
48     <android.support.design.widget.TextInputLayout
49         android:layout_width="match_parent"
50         android:layout_height="wrap_content">
51
52         <EditText
53             android:id="@+id/password"
54             android:layout_width="match_parent"
55             android:layout_height="wrap_content"
56             android:hint="@string/prompt_password"
57             android:imeActionId="@+id/login"
58             android:imeActionLabel="@string/
action_sign_in_short"
59             android:imeOptions="actionUnspecified"
60             android:inputType="textPassword"
61             android:maxLines="1"
62             android:singleLine="true" />
63
64     </android.support.design.widget.
TextInputLayout>
65
66     <Button
67         android:id="@+id/email_sign_in_button"
68         style="?android:textAppearanceSmall"
69         android:layout_width="match_parent"
70         android:layout_height="wrap_content"
71         android:layout_marginTop="16dp"
72         android:text="@string/action_sign_in"
73         android:textStyle="bold" />
74
75     </LinearLayout>
76 </ScrollView>
77 </LinearLayout>
78

```

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <RelativeLayout xmlns:android="http://schemas.android.com/
   apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent">
5
6      <ScrollView
7          android:layout_width="match_parent"
8          android:layout_height="match_parent"
9          android:id="@+id/dialogScrollView"
10         android:layout_alignParentTop="true"
11         android:layout_alignParentLeft="true"
12         android:layout_alignParentStart="true" >
13
14         <LinearLayout
15             android:orientation="vertical"
16             android:layout_width="match_parent"
17             android:layout_height="match_parent"
18             android:background="#b1000000">
19
20             <TextView
21                 android:layout_width="match_parent"
22                 android:layout_height="wrap_content"
23                 android:textAppearance="?android:attr/
   textAppearanceMedium"
24                 android:text="Medium Text"
25                 android:id="@+id/dialogStrategyText1"
26                 android:gravity="center"
27                 android:textColor="#f5f8f9" />
28
29             <VideoView
30                 android:layout_width="wrap_content"
31                 android:layout_height="wrap_content"
32                 android:id="@+id/dialogStrategyVideoView"
   />
33
34             <TextView
35                 android:layout_width="match_parent"
36                 android:layout_height="wrap_content"
37                 android:textAppearance="?android:attr/
   textAppearanceMedium"
38                 android:text="Medium Text"
39                 android:id="@+id/dialogStrategyText2"
40                 android:gravity="center"
41                 android:textColor="#f5f8f9" />

```

Código de desenvolvimento gráfico do APK Robolazul

```
42         </LinearLayout>
43     </ScrollView>
44
45
46 </RelativeLayout>
47
```


Código de desenvolvimento gráfico do APK Robolazul

```

1  <?xml version="1.0" encoding="utf-8" ?>
2  <LinearLayout xmlns:android="http://schemas.android.com/
   apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical" >
6
7      <LinearLayout
8          android:layout_width="match_parent"
9          android:layout_height="wrap_content"
10         android:orientation="horizontal" >
11
12         <TextView
13             android:layout_width="wrap_content"
14             android:layout_height="wrap_content"
15             android:paddingRight="5.0dip"
16             android:text="Name:"
17             android:textColor="#000000"
18             android:textSize="15sp"
19             android:id="@+id/item_btdevices_nameField" />
20
21         <TextView
22             android:id="@+id/item_btdevices_actualName"
23             android:layout_width="wrap_content"
24             android:layout_height="wrap_content"
25             android:paddingRight="5.0dip"
26             android:textColor="#000000"
27             android:textSize="15sp"
28             android:text="Australopitecus" />
29
30     </LinearLayout>
31
32     <LinearLayout
33         android:layout_width="match_parent"
34         android:layout_height="wrap_content"
35         android:orientation="horizontal" >
36
37         <TextView
38             android:layout_width="wrap_content"
39             android:layout_height="wrap_content"
40             android:paddingRight="5.0dip"
41             android:text="MACadress:"
42             android:textColor="#000000"
43             android:textSize="15sp"
44             android:id="@+id/item_btdevices_MACadress" />

```

Código de desenvolvimento gráfico do APK Robolazul

```
45
46     <TextView
47         android:id="@+id/
item_btdevices_actualMACadress"
48         android:layout_width="wrap_content"
49         android:layout_height="wrap_content"
50         android:paddingRight="5.0dip"
51         android:textColor="#000000"
52         android:textSize="15sp"
53         android:text="XXX" />
54     </LinearLayout>
55
56
57 </LinearLayout>
```

Código de desenvolvimento gráfico do APK Robolazul

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/
  apk/res/android"
3     android:layout_width="wrap_content"
4     android:layout_height="wrap_content"
5     android:background="#ffffff"
6     android:orientation="horizontal" >
7
8
9     <ListView
10         android:id="@+id/btdevices_list"
11         android:layout_width="match_parent"
12         android:layout_height="match_parent"/>
13
14
15 </LinearLayout>
16
```

Código de desenvolvimento gráfico do APK Robolazul

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <resources>
3     <color name="colorPrimary">#3F51B5</color>
4     <color name="colorPrimaryDark">#303F9F</color>
5     <color name="colorAccent">#FF4081</color>
6 </resources>
7
```


Código de desenvolvimento gráfico do APK Robolazul

```
1 <resources>
2     <!-- Default screen margins, per the Android Design
3         guidelines. -->
4     <dimen name="activity_horizontal_margin">16dp</dimen>
5     <dimen name="activity_vertical_margin">16dp</dimen>
6 </resources>
```

Código de desenvolvimento gráfico do APK Robolazul

```
1 <resources>
2
3     <!-- Base application theme. -->
4     <style name="AppTheme" parent="Theme.AppCompat.Light.
5         DarkActionBar">
6         <!-- Customize your theme here. -->
7         <item name="colorPrimary">@color/colorPrimary</
8         item>
9         <item name="colorPrimaryDark">@color/
10        colorPrimaryDark</item>
11        <item name="colorAccent">@color/colorAccent</item>
12    </style>
13 </resources>
```

Código de desenvolvimento gráfico do APK Robolazul

```

1 <resources>
2     <string name="app_name">Robolazul</string>
3
4     <string name="prompt_email">Email</string>
5     <string name="prompt_password">Password (optional)</
6     string>
7     <string name="action_sign_in">Sign in or register</
8     string>
9     <string name="action_sign_in_short">Sign in</string>
10    <string name="error_invalid_email">This email address
11    is invalid</string>
12    <string name="error_invalid_password">This password is
13    too short</string>
14    <string name="error_incorrect_password">This password
15    is incorrect</string>
16    <string name="error_access_denied">Invalid credentials
17    </string>
18    <string name="access_allowed">Login successful</string
19    >
20    <string name="error_field_required">This field is
21    required</string>
22    <string name="permission_rationale">"Contacts
23    permissions are needed for providing email
24    completions."
25    </string>
26    <string name="title_activity_login2">Sign in</string>
27
28    <string name="robot1_name">Metalgarurumon</string>
29    <string name="robot2_name">Sonic</string>
30    <string name="robot3_name">Tails</string>
31    <string name="robot4_name">Lobo</string>
32    <string name="robot5_name">Bernadete</string>
33    <string name="robot6_name">Zoiudinho</string>
34    <string name="robot7_name">Tonto</string>
35
36    <string name="Sonic_strategies_1">Right</string>
37    <string name="Sonic_strategies_2">Left</string>
38    <string name="Sonic_strategies_3">Arc-Right</string>
39    <string name="Sonic_strategies_4">Arc-Left</string>
40    <string name="Sonic_strategies_5">Inv-Arc-Right</
41    string>
42    <string name="Sonic_strategies_6">Inv-Arc-Left</string
43    >
44
45    <string name="Bernadete_strategies_1">Right</string>

```

Código de desenvolvimento gráfico do APK Robolazul

```

35     <string name="Bernadete_strategies_2">Left</string>
36     <string name="Bernadete_strategies_3">Arc-Right</
    string>
37     <string name="Bernadete_strategies_4">Arc-Left</string
    >
38     <string name="Bernadete_strategies_5">Inv-Arc-Right</
    string>
39     <string name="Bernadete_strategies_6">Inv-Arc-Left</
    string>
40
41     <string name="Tails_strategies_1">Right</string>
42     <string name="Tails_strategies_2">Left</string>
43
44     <string name="Metalgarurumon_strategies_1">R-Forward</
    string>
45     <string name="Metalgarurumon_strategies_2">L-Forward</
    string>
46     <string name="Metalgarurumon_strategies_3">R-Back</
    string>
47     <string name="Metalgarurumon_strategies_4">L-Back</
    string>
48
49     <string name="Lobo_strategies_1">Right</string>
50     <string name="Lobo_strategies_2">Left</string>
51     <string name="Lobo_strategies_3">Arc-Right</string>
52     <string name="Lobo_strategies_4">Arc-Left</string>
53     <string name="Lobo_strategies_5">Back</string>
54
55     <string name="Zoiudinho_strategies_1">Right</string>
56     <string name="Zoiudinho_strategies_2">Left</string>
57     <string name="Zoiudinho_strategies_3">Arc-Right</
    string>
58     <string name="Zoiudinho_strategies_4">Arc-Left</string
    >
59     <string name="Zoiudinho_strategies_5">Back</string>
60
61
62 </resources>
63

```

Código de desenvolvimento gráfico do APK Robolazul

```

1 <vector xmlns:android="http://schemas.android.com/apk/res/
  android"
2     android:width="24dp"
3     android:height="24dp"
4     android:viewportWidth="24.0"
5     android:viewportHeight="24.0">
6     <path
7         android:fillColor="#FF000000"
8         android:pathData="M6,18c0,0.55 0.45,1 1,1h1v3.5c0,
0.83 0.67,1.5 1.5,1.5s1.5,-0.67 1.5,-1.5L11,19h2v3.5c0,0.
83 0.67,1.5 1.5,1.5s1.5,-0.67 1.5,-1.5L16,19h1c0.55,0 1,-0
.45 1,-1L18,8L6,8v10zM3.5,8C2.67,8 2,8.67 2,9.5v7c0,0.83 0
.67,1.5 1.5,1.5s1.5,-0.67 1.5,-1.5L5,17.33 5,16.5v-7C5,8.67 4.33,8 3.5,8zM20.
5,8c-0.83,0 -1.5,0.67 -1.5,1.5v7c0,0.83 0.67,1.5 1.5,1.5s1
.5,-0.67 1.5,-1.5v-7c0,-0.83 -0.67,-1.5 -1.5,-1.5zM15.53,2
.16l1.3,-1.3c0.2,-0.2 0.2,-0.51 0,-0.71 -0.2,-0.2 -0.51,-0
.2 -0.71,0l-1.48,1.48C13.85,1.23 12.95,1 12,1c-0.96,0 -1.
86,0.23 -2.66,0.63L7.85,0.15c-0.2,-0.2 -0.51,-0.2 -0.71,0
-0.2,0.2 -0.2,0.51 0,0.71l1.31,1.31C6.97,3.26 6,5.01 6,
7h12c0,-1.99 -0.97,-3.75 -2.47,-4.84zM10,5L9,5L9,4h1v1zM15
,5h-1L14,4h1v1z"/>
9 </vector>
10

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

1 package com.example.rperhs.robolazul;
2
3 import android.content.Context;
4     import android.os.Handler;
5     import android.os.Message;
6     import android.os.SystemClock;
7     import android.util.AttributeSet;
8     import android.widget.TextView;
9     import java.text.DecimalFormat;
10
11 public class Chronometer extends TextView {
12     @SuppressWarnings("unused")
13     private static final String TAG = "Chronometer";
14
15     public interface OnChronometerTickListener {
16
17         void onChronometerTick(Chronometer chronometer);
18     }
19
20     private long mBase;
21     private boolean mVisible;
22     private boolean mStarted;
23     private boolean mRunning;
24     private OnChronometerTickListener
mOnChronometerTickListener;
25
26     private static final int TICK_WHAT = 2;
27
28     private long timeElapsed;
29
30     public Chronometer(Context context) {
31         this (context, null, 0);
32     }
33
34     public Chronometer(Context context, AttributeSet attrs
) {
35         this (context, attrs, 0);
36     }
37
38     public Chronometer(Context context, AttributeSet attrs
, int defStyle) {
39         super (context, attrs, defStyle);
40
41         init();
42     }

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```
43
44     private void init() {
45         mBase = SystemClock.elapsedRealtime();
46         updateText(mBase);
47     }
48
49     public void setBase(long base) {
50         mBase = base;
51         dispatchChronometerTick();
52         updateText(SystemClock.elapsedRealtime());
53     }
54
55     public long getBase() {
56         return mBase;
57     }
58
59     public void setOnChronometerTickListener(
60         OnChronometerTickListener listener) {
61         mOnChronometerTickListener = listener;
62     }
63
64     public OnChronometerTickListener
65     getOnChronometerTickListener() {
66         return mOnChronometerTickListener;
67     }
68
69     public void start() {
70         mStarted = true;
71         updateRunning();
72     }
73
74     public void stop() {
75         mStarted = false;
76         updateRunning();
77     }
78
79     public void setStarted(boolean started) {
80         mStarted = started;
81         updateRunning();
82     }
83
84     @Override
85     protected void onDetachedFromWindow() {
86         super .onDetachedFromWindow();
```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

87         mVisible = false;
88         updateRunning();
89     }
90
91     @Override
92     protected void onWindowVisibilityChanged(int
visibility) {
93         super .onWindowVisibilityChanged(visibility);
94         mVisible = visibility == VISIBLE;
95         updateRunning();
96     }
97
98     private synchronized void updateText(long now) {
99         timeElapsed = now - mBase;
100
101         DecimalFormat df = new DecimalFormat("00");
102
103         int hours = (int)(timeElapsed / (3600 * 1000));
104         int remaining = (int)(timeElapsed % (3600 * 1000)
);
105
106         int minutes = (int)(remaining / (60 * 1000));
107         remaining = (int)(remaining % (60 * 1000));
108
109         int seconds = (int)(remaining / 1000);
110         remaining = (int)(remaining % (1000));
111
112         int milliseconds = (int)((int)timeElapsed % 1000
) / 100);
113
114         String text = "";
115
116         if (hours > 0) {
117             text += df.format(hours) + ":";
118         }
119
120         text += df.format(minutes) + ":";
121         text += df.format(seconds) + ":";
122         text += Integer.toString(milliseconds);
123
124         setText(text);
125     }
126
127     private void updateRunning() {
128         boolean running = mVisible && mStarted;

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

129         if (running != mRunning) {
130             if (running) {
131                 updateText(SystemClock.elapsedRealtime())
132             ;
133                 dispatchChronometerTick();
134                 mHandler.sendMessageDelayed(Message.
135 obtain(mHandler,
136                 TICK_WHAT), 100);
137             } else {
138                 mHandler.removeMessages(TICK_WHAT);
139             }
140             mRunning = running;
141         }
142     }
143
144     private Handler mHandler = new Handler() {
145         public void handleMessage(Message m) {
146             if (mRunning) {
147                 updateText(SystemClock.elapsedRealtime())
148             ;
149                 dispatchChronometerTick();
150                 sendMessageDelayed(Message.obtain(this ,
151 TICK_WHAT),
152                 100);
153             }
154         }
155     };
156
157     void dispatchChronometerTick() {
158         if (mOnChronometerTickListener != null) {
159             mOnChronometerTickListener.onChronometerTick(
160 this);
161         }
162     }
163
164     public long getTimeElapsed() {
165         return timeElapsed;
166     }
167 }

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

1  //Explicitação do pacote onde o código está sendo
   desenvolvido
2  package com.example.rperhs.robolazul;
3
4  //Importação das bibliotecas necessárias
5  import android.app.Activity;
6  import android.app.ProgressDialog;
7  import android.bluetooth.BluetoothAdapter;
8  import android.bluetooth.BluetoothSocket;
9
10 import android.bluetooth.BluetoothDevice;
11 import android.content.Intent;
12 import android.os.Bundle;
13 import android.os.Handler;
14 import android.os.SystemClock;
15 import android.view.View;
16 import android.widget.Button;
17 import android.widget.CompoundButton;
18 import android.widget.CompoundButton.
   OnCheckedChangeListener;
19 import android.widget.ImageButton;
20 import android.widget.RadioButton;
21 import android.widget.RadioGroup;
22 import android.widget.RelativeLayout;
23 import android.widget.Switch;
24 import android.widget.TextView;
25 import android.widget.Toast;
26
27 import java.io.IOException;
28 import java.io.OutputStream;
29 import java.util.Set;
30 import java.util.UUID;
31
32 /**
33  * Created by RPERHS on 05/05/2016.
34  */
35 public class MainActivity extends Activity {
36
37     private Thread thread;
38
39     // Declaração de todos os componentes gráficos
   utilizados no layout principal
40     protected RelativeLayout mainLayout; //privacidade |
   super classe do elemento | nome do objeto
41     protected TextView mainInstruction;

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

42     protected Chronometer chronometerMinSec;
43
44     protected int choosedRobotId;
45     protected ImageButton robot1;
46     protected ImageButton robot2;
47     protected ImageButton robot3;
48     protected ImageButton robot4;
49     protected ImageButton robot5;
50     protected ImageButton robot6;
51     protected ImageButton robot7;
52     protected ImageButton close;
53
54     protected TextView goneRobot1;
55     protected TextView goneRobot2;
56     protected TextView goneRobot3;
57     protected TextView goneRobot4;
58     protected TextView goneRobot5;
59     protected TextView goneRobot6;
60     protected TextView goneRobot7;
61
62     protected Button showOptions;
63     protected Button hideOptions;
64
65     protected RelativeLayout optionsLayout;
66
67     protected RadioGroup strategies;
68     protected RadioButton strategy1;
69     protected RadioButton strategy2;
70     protected RadioButton strategy3;
71     protected RadioButton strategy4;
72     protected RadioButton strategy5;
73     protected RadioButton strategy6;
74
75     protected TextView logActions;
76
77     protected Switch BTCommunicationState;
78     protected Switch loadProgram;
79
80     //Declaração de variáveis globais para execução do
aplicativo
81     protected String choosedRobot;
82     protected String selectedBTaddress = null;
83     protected int btfound;
84     private BluetoothAdapter mBluetoothAdapter = null;
85     private BluetoothSocket btSocket = null;

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

86     private boolean isBtConnected = false;
87     private BluetoothDevice dispositivo;
88     private String estrategiaEscolhida;
89
90     //Código a ser executado na inicialização da activity
91     ("tela") principal
92     @Override
93     protected void onCreate(Bundle savedInstanceState) {
94         super.onCreate(savedInstanceState);
95         setContentView(R.layout.activity_main); //Define
96         layout conforme activity_main
97
98         btfound = 0;
99         /*Todo objeto referente a um elemento gráfico
100        declarado inicialmente deve ser referenciado
101        a um elemento presente no layout activity_main
102        */
103        mainLayout = (RelativeLayout) (findViewById(R.id.
104        mainLayout));
105        mainInstruction = (TextView) (findViewById(R.id.
106        mainInstruction));
107        mainInstruction.setClickable(false);
108
109        chronometerMinSec = (Chronometer) (findViewById(R.
110        id.chronometerMinSec));
111        chronometerMinSec.setOnChronometerTickListener(
112        new Chronometer.OnChronometerTickListener() {
113            public void onChronometerTick(Chronometer
114            chronometer) {
115                if (chronometer.getText().toString().
116                equalsIgnoreCase("00:05:0")) {
117                    //Quando o cronômetro atingir 5
118                    segundos.
119                    //Cronômetro deverá parar e uma
120                    mensagem será exibida
121                    chronometer.stop();
122                    Toast.makeText(getBaseContext(), "FIGHT!", Toast.LENGTH_LONG).show();
123                }
124            }
125        });
126
127        choosedRobotId = R.id.robot1; //fins de
128        inicialização
129        robot1 = (ImageButton) (findViewById(R.id.robot1)

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

116 );
117     robot1.setOnClickListener(new View.
    OnClickListener() {
118         @Override //Escolha dos listeners para quando
        o elemento for selecionado pelo usuário
119         public void onClick(View view) {
120             doChooseRobot(R.id.robot1);
121         }
122     });
123     robot2 = (ImageButton) (findViewById(R.id.robot2)
    );
124     robot2.setOnClickListener(new View.
    OnClickListener() {
125         @Override
126         public void onClick(View view) {
127             doChooseRobot(R.id.robot2);
128         }
129     });
130     robot3 = (ImageButton) (findViewById(R.id.robot3)
    );
131     robot3.setOnClickListener(new View.
    OnClickListener() {
132         @Override
133         public void onClick(View view) {
134             doChooseRobot(R.id.robot3);
135         }
136     });
137     robot4 = (ImageButton) (findViewById(R.id.robot4)
    );
138     robot4.setOnClickListener(new View.
    OnClickListener() {
139         @Override
140         public void onClick(View view) {
141             doChooseRobot(R.id.robot4);
142         }
143     });
144     robot5 = (ImageButton) (findViewById(R.id.robot5)
    );
145     robot5.setOnClickListener(new View.
    OnClickListener() {
146         @Override
147         public void onClick(View view) {
148             doChooseRobot(R.id.robot5);
149         }
150     });

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

151         robot6 = (ImageButton) (findViewById(R.id.robot6)
152         );
153         robot6.setOnClickListener(new View.
154         OnClickListener() {
155             @Override
156             public void onClick(View view) {
157                 doChooseRobot(R.id.robot6);
158             }
159         });
160         robot7 = (ImageButton) (findViewById(R.id.robot7)
161         );
162         robot7.setOnClickListener(new View.
163         OnClickListener() {
164             @Override
165             public void onClick(View view) {
166                 doChooseRobot(R.id.robot7);
167             }
168         });
169         close = (ImageButton) (findViewById(R.id.
170         closeButton));
171         close.setOnClickListener(new View.OnClickListener
172         () {
173             @Override
174             public void onClick(View view) {
175                 closeActivity();
176             }
177         });
178         goneRobot1 = (TextView) (findViewById(R.id.
179         goneRobot1));
180         goneRobot2 = (TextView) (findViewById(R.id.
181         goneRobot2));
182         goneRobot3 = (TextView) (findViewById(R.id.
183         goneRobot3));
184         goneRobot4 = (TextView) (findViewById(R.id.
185         goneRobot4));
186         goneRobot5 = (TextView) (findViewById(R.id.
187         goneRobot5));
188         goneRobot6 = (TextView) (findViewById(R.id.
189         goneRobot6));
190         goneRobot7 = (TextView) (findViewById(R.id.
191         goneRobot7));
192         showOptions = (Button) (findViewById(R.id.
193         showOptions));

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

182         hideOptions = (Button) (findViewById(R.id.
hideOptions));
183
184         optionsLayout = (RelativeLayout) (findViewById(R.
id.optionsLayout));
185         logActions = (TextView) (findViewById(R.id.
logActions));
186
187         strategies = (RadioGroup) (findViewById(R.id.
strategies));
188         strategy1 = (RadioButton) (findViewById(R.id.
strategy1));
189         strategy2 = (RadioButton) (findViewById(R.id.
strategy2));
190         strategy3 = (RadioButton) (findViewById(R.id.
strategy3));
191         strategy4 = (RadioButton) (findViewById(R.id.
strategy4));
192         strategy5 = (RadioButton) (findViewById(R.id.
strategy5));
193         strategy6 = (RadioButton) (findViewById(R.id.
strategy6));
194
195         BTCommunicationState = (Switch) (findViewById(R.
id.BTcommunicationState));
196         BTCommunicationState.setOnCheckedChangeListener(
new OnCheckedChangeListener() {
197             @Override
198             public void onCheckedChanged(CompoundButton
compoundButton, boolean isChecked) {
199                 if(isChecked)
200                 {
201                     boolean connectSuccess = true;
202                     mBluetoothAdapter = BluetoothAdapter.
getDefaultAdapter();
203                     if (mBluetoothAdapter == null) {
204                         Toast.makeText(
getApplicationContext(), "There is no Bluetooth device
available", Toast.LENGTH_LONG).show();
205                         finish();
206                     } else {
207                         if(mBluetoothAdapter.isEnabled())
{
208                             // tentar realizar conexao
209                             try{

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

210             changeIntent();
211         } catch (/*IOException*/
Exception e) {
212             connectSuccess = false;
213         }
214         mainInstruction.setText("
PLEASE LOAD A STRATEGY");
215         updateLogActions("BT started
with " + getChoosedRobot());
216     } else{
217         Toast.makeText(
getApplicationContext(), "Bluetooth Adapter is not
enabled, Please enable it and try again!", Toast.
LENGTH_LONG).show();
218         loadProgram.setChecked(false)
;
219     }
220 }
221 btfound++;
222 mainInstruction.setText("PLEASE LOAD
A STRATEGY");
223 }
224 else
225 {
226     loadProgram.setChecked(false);
227     mainInstruction.setText("PLEASE START
BT COMMUNICATION");
228     Toast.makeText(getApplicationContext(
), "Bluetooth communication successfully destroyed with "
+ getChoosedRobot() + "!", Toast.LENGTH_LONG).show();
229     updateLogActions("BT destroyed with "
+ getChoosedRobot());
230 }
231 }
232 });
233 loadProgram = (Switch) (findViewById(R.id.
loadProgram));
234 loadProgram.setOnCheckedChangeListener(new
OnCheckedChangeListener() {
235     @Override
236     public void onCheckedChanged(CompoundButton
compoundButton, boolean isChecked) {
237         if(isChecked)
238         {
239             if(BTCommunicationState.isChecked())

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

240         {
241             estrategiaEscolhida =
chooseStrategy();
242             Toast.makeText(
getApplicationContext(), "Strategy " + getChoosedStrategy
() + " successfully loaded to " + getChoosedRobot() + "!",
Toast.LENGTH_LONG).show();
243             updateLogActions("Strategy " +
getChoosedStrategy() + " loaded" );
244             mainInstruction.setText("PRESS
HERE TO START A FIGHT");
245             mainInstruction.setClickable(true
);
246         }
247         else
248         {
249             mainInstruction.setText("PLEASE
START BT COMMUNICATION");
250             Toast.makeText(
getApplicationContext(), "ERROR: No robot is
communicating", Toast.LENGTH_LONG).show();
251         }
252     }
253 }
254 else
255 {
256     mainInstruction.setText("PLEASE LOAD
A STRATEGY");
257     /*
258     CÓDIGO DE ALERTA PARA GUIAR PRÓXIMOS
PASSOS DO USUÁRIO
259     */
260 }
261 }
262 });
263
264     //Parametrização para inicialização do aplicativo
num cenário qualquer
265     doChooseRobot(R.id.robot1);
266     optionsLayout.setVisibility(View.GONE);
267     hideOptions.setVisibility(View.GONE);
268     showOptions.setVisibility(View.VISIBLE);
269
270     /*Ao retornar da lista de dispositivos bluetooth
pareados, onde o usuário deverá

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

271         informar qual módulo bluetooth HC-05 ele deverá
272         conectar para enviar as mensagens
273         necessárias durante a execução do aplicativo,
274         há de resgatar alguns valores que serão
275         armazenados momentaneamente para efetividade da
276         comunicação
277         */
278         if (getIntent().getExtras() != null)
279         {
280             selectedBTaddress = (String) getIntent().
getSerializableExtra("itemSelecionado");
281             String test;
282             test = (String) getIntent().
getSerializableExtra("robot");
283             Toast.makeText(getApplicationContext(),
selectedBTaddress + " " + test, Toast.LENGTH_SHORT).show()
;
284             if(test != null)
285             {
286                 switch (test)
287                 {
288                     case "Metalgarurumon":doChooseRobot(R
.id.robot1);break;
289                     case "Sonic":doChooseRobot(R.id.
robot2);break;
290                     case "Tails":doChooseRobot(R.id.
robot3);break;
291                     case "Lobo":doChooseRobot(R.id.robot4
);break;
292                     case "Bernadete":doChooseRobot(R.id.
robot5);break;
293                     case "Zoiudinho":doChooseRobot(R.id.
robot6);break;
294                     case "Tonto":doChooseRobot(R.id.
robot7);break;
295                     default:break;
296                 }
297                 try {//Tentativa de comunicação bluetooth
efetiva com o dispositivo escolhido
298                     if (btSocket == null || !
isBtConnected) {
299                         mBluetoothAdapter =
BluetoothAdapter.getDefaultAdapter();
300                         dispositivo = mBluetoothAdapter.
getRemoteDevice(selectedBTaddress);

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

298         btSocket = dispositivo.
createInsecureRfcommSocketToServiceRecord(UUID.fromString
("00001101-0000-1000-8000-00805F9B34FB"));
299         BluetoothAdapter.
getDefaultAdapter().cancelDiscovery();
300         btSocket.connect();
301         Toast.makeText(
getApplicationContext(), "Bluetooth communication
successfully established with " + getChoosedRobot() + "!"
, Toast.LENGTH_LONG).show();
302         updateLogActions("BT started with
" + getChoosedRobot());
303         BTCommunicationState.setChecked(
true);
304     }
305     } catch(IOException e) {
306     }
307 }
308
309 }
310 }
311
312 public void doTestChronometer()
313 {
314     //Inicializa o ponto de partida do cronômetro e
inica a contagem
315     chronometerMinSec.setBase(SystemClock.
elapsedRealtime());
316     chronometerMinSec.start();
317 }
318
319 public void doHideOptions(View view)
320 {
321     //Alteração no layout - escondendo as opções para
comunicação bluetooth
322     optionsLayout.setVisibility(View.GONE);
323     showOptions.setVisibility(View.VISIBLE);
324     hideOptions.setVisibility(View.GONE);
325 }
326
327 public void doShowOptions(View view)
328 {
329     //Alteração no layout - exibindo as opções para
comunicação bluetooth
330     optionsLayout.setVisibility(View.VISIBLE);

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

331         hideOptions.setVisibility(View.VISIBLE);
332         showOptions.setVisibility(View.GONE);
333     }
334
335     //Método responsável por retornar qual a estratégia
está selecionada
336     public String getChoosedStrategy()
337     {
338         String choosedStrategy;
339         switch(strategies.getCheckedRadioButtonId())
340         {
341             case R.id.strategy1: choosedStrategy =
strategy1.getText().toString();break;
342             case R.id.strategy2: choosedStrategy =
strategy2.getText().toString();break;
343             case R.id.strategy3: choosedStrategy =
strategy3.getText().toString();break;
344             case R.id.strategy4: choosedStrategy =
strategy4.getText().toString();break;
345             case R.id.strategy5: choosedStrategy =
strategy5.getText().toString();break;
346             case R.id.strategy6: choosedStrategy =
strategy6.getText().toString();break;
347             default: choosedStrategy = "Please select a
strategy!";break;
348         }
349         return choosedStrategy;
350     }
351
352     //Método para retornar qual robô está selecionado
353     public String getChoosedRobot()
354     {
355         switch(choosedRobotId)
356         {
357             case R.id.robot1: choosedRobot = getString(R.
string.robot1_name);break;
358             case R.id.robot2: choosedRobot = getString(R.
string.robot2_name);break;
359             case R.id.robot3: choosedRobot = getString(R.
string.robot3_name);break;
360             case R.id.robot4: choosedRobot = getString(R.
string.robot4_name);break;
361             case R.id.robot5: choosedRobot = getString(R.
string.robot5_name);break;
362             case R.id.robot6: choosedRobot = getString(R.

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

362 string.robot6_name);break;
363         case R.id.robot7: choosedRobot = getString(R.
string.robot7_name);break;
364         default: choosedRobot = "Pick a robot";break;
365     }
366     return choosedRobot;
367 }
368
369     //Método que define internamente o robô selecionado
pelo usuário de acordo com sua interação
370     public void doChooseRobot(int id)
371     {
372         switch(id)
373         {
374             case R.id.robot1: if(choosedRobotId!=id){
BTCommunicationState.setChecked(false);}setAllRobotsOff()
;goneRobot1.setVisibility(View.GONE); choosedRobotId = R.
id.robot1;setStrategies();break;
375             case R.id.robot2: if(choosedRobotId!=id){
BTCommunicationState.setChecked(false);}setAllRobotsOff()
;goneRobot2.setVisibility(View.GONE); choosedRobotId = R.
id.robot2;setStrategies();break;
376             case R.id.robot3: if(choosedRobotId!=id){
BTCommunicationState.setChecked(false);}setAllRobotsOff()
;goneRobot3.setVisibility(View.GONE); choosedRobotId = R.
id.robot3;setStrategies();break;
377             case R.id.robot4: if(choosedRobotId!=id){
BTCommunicationState.setChecked(false);}setAllRobotsOff()
;goneRobot4.setVisibility(View.GONE); choosedRobotId = R.
id.robot4;setStrategies();break;
378             case R.id.robot5: if(choosedRobotId!=id){
BTCommunicationState.setChecked(false);}setAllRobotsOff()
;goneRobot5.setVisibility(View.GONE); choosedRobotId = R.
id.robot5;setStrategies();break;
379             case R.id.robot6: if(choosedRobotId!=id){
BTCommunicationState.setChecked(false);}setAllRobotsOff()
;goneRobot6.setVisibility(View.GONE); choosedRobotId = R.
id.robot6;setStrategies();break;
380             case R.id.robot7: if(choosedRobotId!=id){
BTCommunicationState.setChecked(false);}setAllRobotsOff()
;goneRobot7.setVisibility(View.GONE); choosedRobotId = R.
id.robot7;setStrategies();break;
381             default: setAllRobotsOff();break;
382         }
383     }

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

384
385     //Retira foco visual de todos os robôs da lista
386     public void setAllRobotsOff()
387     {
388         goneRobot1.setVisibility(View.VISIBLE);
389         goneRobot2.setVisibility(View.VISIBLE);
390         goneRobot3.setVisibility(View.VISIBLE);
391         goneRobot4.setVisibility(View.VISIBLE);
392         goneRobot5.setVisibility(View.VISIBLE);
393         goneRobot6.setVisibility(View.VISIBLE);
394         goneRobot7.setVisibility(View.VISIBLE);
395     }
396
397     //Método responsável por atualizar o log de ações
tomadas pelo usuário
398     public void updateLogActions(String newLogEvent)
399     {
400         String actualText = logActions.getText().toString
401         ();
402         logActions.setText(actualText + "\n" +
403         newLogEvent + "...");
404     }
405
406     //Método responsável para iniciar a luta
407     public void startRobot(View view)
408     {
409         if (mainInstruction.getText()=="PRESS HERE TO
410         START A FIGHT")
411         {
412             if(BTCommunicationState.isChecked() &
413             loadProgram.isChecked())
414             {
415                 Toast.makeText(getApplicationContext(), "Are you ready?!", Toast.LENGTH_SHORT).show();
416                 doTestChronometer();
417                 Handler handler = new Handler();
418                 handler.postDelayed(new Runnable() {
419                     @Override
420                     public void run() {
421                         startFight(btSocket,
422                         estrategiaEscolhida);
423                         mainInstruction.setText("PRESS
424                         HERE TO STOP THE FIGHT");
425                     }
426                 }, 4600); /*Valor definido empiricamente

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

420 para juntamente com os atrasos oriundos
421 da execução do código, propagação do
422 sinal pelo ar e, por fim, interpretação
423 por parte do PIC a ser controlado, a o
424 robô inicie seus movimentos em 5 segundos*/
425     }
426     else
427     {
428         Toast.makeText(getApplicationContext(), "Please try again!", Toast.LENGTH_SHORT).show();
429     }
430     //Método para encerrar a luta
431     else
432     {
433         if(mainInstruction.getText()=="PRESS HERE TO STOP THE FIGHT")
434         {
435             stopFight(btSocket);
436             BTCommunicationState.setChecked(false);
437             mainInstruction.setText("PLEASE START BT COMMUNICATION");
438             chronometerMinSec.setBase(SystemClock.elapsedRealtime());
439         }
440     }
441 }
442
443 //Método que define qual caractere enviar de acordo
444 com a estratégia definida pelo usuário
445 /*Importante que os caracteres são os mesmos para as
446 6 possibilidades de Radio Button
447 Todavia cada PIC interpretará os caracteres de forma
448 diferente. Esta padronização visou
449 simplificar o processo de envio de mensagens para
450 acionamento de acordo com estratégia*/
451 public String chooseStrategy(){
452     String toSend = "h";
453     int strategy = strategies.getCheckedRadioButtonId
    ();
454     switch (strategy)
455     {
456         case R.id.strategy1: toSend = "a"; break;
457         case R.id.strategy2: toSend = "b"; break;

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

454         case R.id.strategy3: toSend = "c"; break;
455         case R.id.strategy4: toSend = "d"; break;
456         case R.id.strategy5: toSend = "e"; break;
457         case R.id.strategy6: toSend = "f"; break;
458     }
459     return toSend;
460
461 }
462
463     //Método que define uma mensagem de 10 caracteres
idênticos para encerrar a luta
464     /*Empiricamente notou-se que o envio de um único
caractere para encerrar a luta não apresentava
465     * bons resultados, dado às condições de programação
interna dos PICs responsáveis pelo controle
466     * autônomo dos robôs. Esta solução mostrou resultado
assertivo em todas as tentativas*/
467     public void stopFight(BluetoothSocket socket){
468         String parar = "h";
469         int i;
470         byte[] toSend = parar.getBytes();
471         try{
472             for(i = 0; i<10; i++) {
473                 OutputStream mmOutputStream = socket.
getOutputStream();
474                 mmOutputStream.write(toSend);
475             }
476         } catch(IOException e){
477
478         }
479     }
480     //Método que envia a mensagem para iniciar a luta
481     public void startFight(BluetoothSocket socket, String
strategy){
482         byte[] toSend = strategy.getBytes();
483         try{
484             OutputStream mmOutputStream = socket.
getOutputStream();
485             mmOutputStream.write(toSend);
486         } catch(IOException e){
487
488         }
489     }
490
491     //Método que altera o nome das estratégias de acordo

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

491 com o robô selecionado
492     public void setStrategies()
493     {
494         strategy1.setClickable(true);
495         strategy2.setClickable(true);
496         strategy3.setClickable(true);
497         strategy4.setClickable(true);
498         strategy5.setClickable(true);
499         strategy6.setClickable(true);
500         switch(getChoosedRobot())
501         {
502             case "Sonic":
503                 strategy1.setText(R.string.
Sonic_strategies_1);
504                 strategy2.setText(R.string.
Sonic_strategies_2);
505                 strategy3.setText(R.string.
Sonic_strategies_3);
506                 strategy4.setText(R.string.
Sonic_strategies_4);
507                 strategy5.setText(R.string.
Sonic_strategies_5);
508                 strategy6.setText(R.string.
Sonic_strategies_6);break;
509
510             case "Tails":
511                 strategy1.setText(R.string.
Tails_strategies_1);
512                 strategy2.setText(R.string.
Tails_strategies_2);
513                 strategy3.setText("Not Available");
strategy3.setClickable(false);
514                 strategy4.setText("Not Available");
strategy4.setClickable(false);
515                 strategy5.setText("Not Available");
strategy5.setClickable(false);
516                 strategy6.setText("Not Available");
strategy6.setClickable(false);break;
517
518             case "Bernadete":
519                 strategy1.setText(R.string.
Bernadete_strategies_1);
520                 strategy2.setText(R.string.
Bernadete_strategies_2);
521                 strategy3.setText(R.string.

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

521 Bernadete_strategies_3);
522         strategy4.setText(R.string.
    Bernadete_strategies_4);
523         strategy5.setText(R.string.
    Bernadete_strategies_5);
524         strategy6.setText(R.string.
    Bernadete_strategies_6);break;
525
526         case "Metalgarurumon":
527             strategy1.setText(R.string.
    Metalgarurumon_strategies_1);
528             strategy2.setText(R.string.
    Metalgarurumon_strategies_2);
529             strategy3.setText(R.string.
    Metalgarurumon_strategies_3);
530             strategy4.setText(R.string.
    Metalgarurumon_strategies_4);
531             strategy5.setText("Not Available");
    strategy5.setClickable(false);
532             strategy6.setText("Not Available");
    strategy6.setClickable(false);break;
533
534         case "Lobo":
535             strategy1.setText(R.string.
    Lobo_strategies_1);
536             strategy2.setText(R.string.
    Lobo_strategies_2);
537             strategy3.setText(R.string.
    Lobo_strategies_3);
538             strategy4.setText(R.string.
    Lobo_strategies_4);
539             strategy5.setText(R.string.
    Lobo_strategies_5);
540             strategy6.setText("Not Available");
    strategy6.setClickable(false);break;
541
542         case "Zoiudinho":
543             strategy1.setText(R.string.
    Zoiudinho_strategies_1);
544             strategy2.setText(R.string.
    Zoiudinho_strategies_2);
545             strategy3.setText(R.string.
    Zoiudinho_strategies_3);
546             strategy4.setText(R.string.
    Zoiudinho_strategies_4);

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

547         strategy5.setText(R.string.
Zoiudinho_strategies_5);
548         strategy6.setText("Not Available");
strategy6.setClickable(false);break;
549
550         case "Tonto":
551             strategy1.setText("Not Available");
strategy1.setClickable(false);
552             strategy2.setText("Not Available");
strategy2.setClickable(false);
553             strategy3.setText("Not Available");
strategy3.setClickable(false);
554             strategy4.setText("Not Available");
strategy4.setClickable(false);
555             strategy5.setText("Not Available");
strategy5.setClickable(false);
556             strategy6.setText("Not Available");
strategy6.setClickable(false);break;
557         default:break;
558     }
559 }
560
561     /*Método que finaliza a activity principal e exibe a
activity com a lista de dispositivos
562     bluetooth previamente pareados*/
563     public void changeIntent()
564     {
565         if(btfound == 1)
566         {
567             Intent intent = new Intent(this,
DeviceListActivity.class);
568             intent.putExtra("robot",choosedRobot);
569             startActivity(intent);
570             MainActivity.this.finish();
571         }
572     }
573
574     @Override
575     public void onPause()
576     {
577         super.onPause();
578     }
579
580     @Override
581     //Método que armazena qual dispositivo bluetooth foi

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

581 selecionado na lista de dispositivos
582     public void onResume()
583     {
584         super.onResume();
585         if (getIntent().getExtras() != null)
586         {
587             selectedBTaddress = (String) getIntent().
getSerializableExtra("itemSelecionado");
588         }
589     }
590
591
592     @Override
593     //Método que armazena qual dispositivo bluetooth foi
selecionado na lista de dispositivos
594     //Similar ao onResume, mas para o caso da activity
ser reinicializada
595     protected void onRestart() {
596         super.onRestart();
597         if (getIntent().getExtras() != null) {
598             selectedBTaddress = (String) getIntent().
getSerializableExtra("itemSelecionado");
599             Toast.makeText(getApplicationContext(),
selectedBTaddress, Toast.LENGTH_SHORT).show();
600         }
601     }
602
603     //Método que encerra o aplicativo
604     public void closeActivity()
605     {
606         MainActivity.this.onDestroy();
607         System.exit(0);
608     }
609
610 }
611

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

1 package com.example.rperhs.robolazul;
2
3 /**
4  * Created by RPERHS on 25/04/2016.
5  */
6
7 import android.animation.Animator;
8 import android.animation.AnimatorListenerAdapter;
9 import android.annotation.TargetApi;
10 import android.app.Activity;
11 import android.app.LoaderManager;
12 import android.content.CursorLoader;
13 import android.content.Loader;
14 import android.content.pm.PackageManager;
15 import android.database.Cursor;
16 import android.graphics.Color;
17 import android.net.Uri;
18 import android.os.AsyncTask;
19 import android.os.Build;
20 import android.os.Bundle;
21 import android.provider.ContactsContract;
22 import android.support.annotation.NonNull;
23 import android.support.design.widget.Snackbar;
24 import android.support.v7.app.AppCompatActivity;
25 import android.text.TextUtils;
26 import android.view.KeyEvent;
27 import android.view.Menu;
28 import android.view.View;
29 import android.view.inputmethod.EditorInfo;
30 import android.widget.AdapterView;
31 import android.widget.AutoCompleteTextView;
32 import android.widget.Button;
33 import android.widget.EditText;
34 import android.widget.TextView;
35 import android.widget.Toast;
36
37 import java.util.ArrayList;
38 import java.util.List;
39
40 import static android.Manifest.permission.READ_CONTACTS;
41
42 public class LoginActivity extends AppCompatActivity
43     implements LoaderManager.LoaderCallbacks<Cursor> {
44     private static final int REQUEST_READ_CONTACTS = 0;
45     private static final String[] DUMMY_CREDENTIALS = new

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

44 String[]{
45     "robotica.feg@gmail.com:roboschica"
46 };
47
48 private UserLoginTask mAuthTask = null;
49
50 private AutoCompleteTextView mEmailView;
51 private EditText mPasswordView;
52 private View mProgressView;
53 private View mLoginFormView;
54
55 @Override
56 protected void onCreate(Bundle savedInstanceState) {
57     super.onCreate(savedInstanceState);
58     setContentView(R.layout.activity_login);
59
60     mEmailView = (AutoCompleteTextView) findViewById(R
        .id.email);
61     populateAutoComplete();
62
63     mPasswordView = (EditText) findViewById(R.id.
        password);
64     mPasswordView.setOnEditorActionListener(new
        TextView.OnEditorActionListener() {
65         @Override
66         public boolean onEditorAction(TextView
        textView, int id, KeyEvent keyEvent) {
67             if (id == R.id.login || id == EditorInfo.
        IME_NULL) {
68                 attemptLogin();
69                 return true;
70             }
71             return false;
72         }
73     });
74
75     Button mEmailSignInButton = (Button) findViewById(
        R.id.email_sign_in_button);
76     mEmailSignInButton.setOnClickListener(new View.
        OnClickListener() {
77         @Override
78         public void onClick(View view) {
79             attemptLogin();
80         }
81     });

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

82
83         mLoginFormView = findViewById(R.id.login_form);
84         mProgressView = findViewById(R.id.login_progress)
85     ;
86
87     private void populateAutoComplete() {
88         if (!mayRequestContacts()) {
89             return;
90         }
91
92         getLoaderManager().initLoader(0, null, this);
93     }
94
95     private boolean mayRequestContacts() {
96         if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M
97         ) {
98             return true;
99         }
100         if (checkSelfPermission(READ_CONTACTS) ==
101         PackageManager.PERMISSION_GRANTED) {
102             return true;
103         }
104         if (shouldShowRequestPermissionRationale(
105         READ_CONTACTS)) {
106             Snackbar.make(mEmailView, R.string.
107             permission_rationale, Snackbar.LENGTH_INDEFINITE)
108             .setAction(android.R.string.ok, new
109             View.OnClickListener() {
110                 @Override
111                 @TargetApi(Build.VERSION_CODES.M)
112                 public void onClick(View v) {
113                     requestPermissions(new String
114                     [{READ_CONTACTS}, REQUEST_READ_CONTACTS]);
115                 }
116             });
117         } else {
118             requestPermissions(new String[] {READ_CONTACTS
119             }, REQUEST_READ_CONTACTS);
120         }
121         return false;
122     }
123
124     @Override
125     public void onRequestPermissionsResult(int

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

118 requestCode, @NonNull String[] permissions,
119                                     @NonNull int[]
    grantResults) {
120     if (requestCode == REQUEST_READ_CONTACTS) {
121         if (grantResults.length == 1 && grantResults[
0] == PackageManager.PERMISSION_GRANTED) {
122             populateAutoComplete();
123         }
124     }
125 }
126
127 private void attemptLogin() {
128     if (mAuthTask != null) {
129         return;
130     }
131
132     mEmailView.setError(null);
133     mPasswordView.setError(null);
134
135     String email = mEmailView.getText().toString();
136     String password = mPasswordView.getText().
toString();
137
138     boolean cancel = false;
139     View focusView = null;
140
141     if (!TextUtils.isEmpty(password) && !
isPasswordValid(password)) {
142         mPasswordView.setError(getString(R.string.
error_invalid_password));
143         focusView = mPasswordView;
144         cancel = true;
145     }
146
147     if (TextUtils.isEmpty(email)) {
148         mEmailView.setError(getString(R.string.
error_field_required));
149         focusView = mEmailView;
150         cancel = true;
151     } else if (!isEmailValid(email)) {
152         mEmailView.setError(getString(R.string.
error_invalid_email));
153         focusView = mEmailView;
154         cancel = true;
155     }

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

156
157         if (cancel) {
158             focusView.requestFocus();
159         } else {
160             showProgress(true);
161             mAuthTask = new UserLoginTask(email, password
162 );
163             mAuthTask.execute((Void) null);
164         }
165     }
166
167     private boolean isEmailValid(String email) {
168         return email.contains("@");
169     }
170
171     private boolean isPasswordValid(String password) {
172         return password.length() > 4;
173     }
174
175     @TargetApi(Build.VERSION_CODES.HONEYCOMB_MR2)
176     private void showProgress(final boolean show) {
177
178         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.
HONEYCOMB_MR2) {
179             int shortAnimTime = getResources().getInteger
(android.R.integer.config_shortAnimTime);
180
181             mLoginFormView.setVisibility(show ? View.GONE
: View.VISIBLE);
182             mLoginFormView.animate().setDuration(
shortAnimTime).alpha(
183                 show ? 0 : 1).setListener(new
AnimatorListenerAdapter() {
184                 @Override
185                 public void onAnimationEnd(Animator
animation) {
186                     mLoginFormView.setVisibility(show ?
View.GONE : View.VISIBLE);
187                 }
188             });
189
190             mProgressView.setVisibility(show ? View.
VISIBLE : View.GONE);
191             mProgressView.animate().setDuration(

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

191 shortAnimTime).alpha(
192             show ? 1 : 0).setListener(new
    AnimatorListenerAdapter() {
193             @Override
194             public void onAnimationEnd(Animator
    animation) {
195             mProgressView.setVisibility(show ?
    View.VISIBLE : View.GONE);
196             }
197             });
198     } else {
199
200         mProgressView.setVisibility(show ? View.
    VISIBLE : View.GONE);
201         mLoginFormView.setVisibility(show ? View.GONE
    : View.VISIBLE);
202     }
203 }
204
205 @Override
206 public Loader<Cursor> onCreateLoader(int i, Bundle
    bundle) {
207     return new CursorLoader(this,
208         Uri.withAppendedPath(ContactsContract.
    Profile.CONTENT_URI,
209             ContactsContract.Contacts.Data.
    CONTENT_DIRECTORY), ProfileQuery.PROJECTION,
210         ContactsContract.Contacts.Data.MIMETYPE +
211             " = ?", new String[]{
    ContactsContract.CommonDataKinds.Email
212             .CONTENT_ITEM_TYPE},
213         ContactsContract.Contacts.Data.IS_PRIMARY
    + " DESC");
214 }
215
216 @Override
217 public void onLoadFinished(Loader<Cursor>
    cursorLoader, Cursor cursor) {
218     List<String> emails = new ArrayList<>();
219     cursor.moveToFirst();
220     while (!cursor.isAfterLast()) {
221         emails.add(cursor.getString(ProfileQuery.
    ADDRESS));
222         cursor.moveToNext();
223     }

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

224
225         addEmailsToAutoComplete(emails);
226     }
227
228     @Override
229     public void onLoaderReset(Loader<Cursor> cursorLoader
230     ) {
231     }
232
233     private void addEmailsToAutoComplete(List<String>
234     emailAddressCollection) {
235         ArrayAdapter<String> adapter =
236             new ArrayAdapter<>(LoginActivity.this,
237                 android.R.layout.
238                 simple_dropdown_item_1line, emailAddressCollection);
239
240         mEmailView.setAdapter(adapter);
241     }
242
243     private interface ProfileQuery {
244         String[] PROJECTION = {
245             ContactsContract.CommonDataKinds.Email.
246             ADDRESS,
247             ContactsContract.CommonDataKinds.Email.
248             IS_PRIMARY,
249         };
250
251         int ADDRESS = 0;
252         int IS_PRIMARY = 1;
253     }
254
255     public class UserLoginTask extends AsyncTask<Void,
256     Void, Boolean> {
257
258         private final String mEmail;
259         private final String mPassword;
260
261         UserLoginTask(String email, String password) {
262             mEmail = email;
263             mPassword = password;
264         }
265
266         @Override

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

263         protected Boolean doInBackground(Void... params)
264     {
265         try {
266             Thread.sleep(2000);
267         } catch (InterruptedException e) {
268             return false;
269         }
270
271         for (String credential : DUMMY_CREDENTIALS) {
272             String[] pieces = credential.split(":");
273             if (pieces[0].equals(mEmail)) {
274                 return pieces[1].equals(mPassword);
275             }
276         }
277         return false;
278     }
279
280     @Override
281     protected void onPostExecute(final Boolean
282     success) {
283         mAuthTask = null;
284         showProgress(false);
285
286         if (success) {
287             Toast.makeText(getApplicationContext(),
288             getString(R.string.access_allowed), Toast.LENGTH_LONG).
289             show();
290         } else {
291             mPasswordView.setError(getString(R.string
292             .error_access_denied));
293             mPasswordView.requestFocus();
294         }
295     }
296
297     @Override
298     protected void onCancelled() {
299         mAuthTask = null;
300         showProgress(false);
301     }
302 }

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

1  package com.example.rperhs.robolazul;
2
3  import android.app.Activity;
4  import android.bluetooth.BluetoothDevice;
5  import android.view.View;
6  import android.view.ViewGroup;
7  import android.widget.AdapterView;
8  import android.widget.TextView;
9
10 import com.example.rperhs.robolazul.R;
11
12 import java.util.List;
13
14 /**
15  * Created by RPERHS on 06/07/2016.
16  */
17 public class DeviceListAdapter extends ArrayAdapter<
    BluetoothDevice> {
18
19
20     private final List<BluetoothDevice> devices;
21     private final Activity activity;
22
23     public DeviceListAdapter(Activity activity, int
    textViewResourceId, List<BluetoothDevice> devices) {
24         super(activity, textViewResourceId, devices);
25         this.activity = activity;
26         /**
27          * List de objetos do tipo BluetoothDevice
    recebida como parâmetro pelo Construtor.
28          */
29         this.devices = devices;
30     }
31
32     @Override
33     public View getView(int position, View convertView,
    ViewGroup parent)
34     {
35
36         BluetoothDevice bt = devices.get(position);
37         View view = activity.getLayoutInflater().inflate(R
    .layout.item_btdevices_list, null);
38         TextView btName = (TextView) view.findViewById(R.
    id.item_btdevices_actualName);
39         btName.setText(bt.getName().toString());

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```
40         TextView btAddress = (TextView) view.findViewById(  
    R.id.item_btdevices_actualMACadress);  
41         btAddress.setText(bt.getAddress().toString());  
42         return view;  
43     }  
44     /**  
45      * Retorna o número de itens que serão mostrados na  
    ListView.  
46      */  
47     @Override  
48     public int getCount()  
49     {  
50         return super.getCount();  
51     }  
52  
53     /**  
54      * Retorna um determinado item da ListView, de acordo  
55      * com a sua posição.  
56      */  
57     @Override  
58     public BluetoothDevice getItem(int position)  
59     {  
60         return devices.get(position);  
61     }  
62 }  
63
```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

1 package com.example.rperhs.robolazul;
2
3 import android.bluetooth.BluetoothAdapter;
4 import android.bluetooth.BluetoothDevice;
5
6 /**
7  * Created by RPERHS on 06/07/2016.
8  */
9 import java.util.ArrayList;
10 import java.util.List;
11
12 import android.app.Activity;
13 import android.bluetooth.BluetoothManager;
14 import android.content.Intent;
15 import android.os.Bundle;
16 import android.view.KeyEvent;
17 import android.view.Menu;
18 import android.widget.AdapterView;
19 import android.widget.ListView;
20 import android.widget.Toast;
21
22
23 import java.util.Set;
24
25 public class DeviceListActivity extends Activity
26 {
27
28     protected BluetoothAdapter mBluetoothAdapter = null;
29     /**
30      * Variável de instância do tipo ListView que fará
31      * referência à ListView do
32      * nosso Resource Layout e será manipulado via
33      * código para
34      * ser preenchido com os dados consultados no
35      * banco de dados.
36      */
37     private ListView deviceList;
38     // ID da opção "Apagar" do menu de contexto
39     private static final int MENU_APAGAR= Menu.FIRST;
40     public String robot;
41
42     @Override
43     public void onCreate(Bundle savedInstanceState)
44     {
45         if (getIntent().getExtras() != null)

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

43         {
44             robot = (String) getIntent().
getSerializableExtra("robot");
45         }
46         super.onCreate(savedInstanceState);
47         // Definimos o Resource Layout a ser
controlado pela Activity
48         setContentView(R.layout.
activity_btdevices_list);
49         /**
50          * Fazemos a ligação entre o objeto "
deviceList" e a View
51          * ListView do nosso Resource Layout.
52          */
53         deviceList= (ListView) findViewById(R.id.
btdevices_list);
54         // Registra o Context Menu para a ListView da
nossa tela de aplicação
55         registerForContextMenu(deviceList);
56     }
57
58     /**
59      * Retorna o objeto ListView já preenchido com a
lista de dispositivos pareados.
60      */
61     @Override
62     protected void onResume()
63     {
64         super.onResume();
65         // Chama o método que irá montar a ListView na
tela.
66         createListView();
67     }
68     public ListView getDeviceList()
69     {
70         return deviceList;
71     }
72     /**
73      * Monta a ListView com os dados a serem
apresentados na tela.
74      */
75     private void createListView()
76     {
77         BluetoothManager mBluetoothManager = (
BluetoothManager) getSystemService(BLUETOOTH_SERVICE);

```


Código de desenvolvimento das funcionalidades do APK Robolazul

```

78         mBluetoothAdapter = mBluetoothManager.
getAdapter();
79         Set<BluetoothDevice> pairedDevices =
mBluetoothAdapter.getBondedDevices();
80         List<BluetoothDevice> btlist = new ArrayList<
BluetoothDevice>();
81         if(pairedDevices.size()>0){
82             for(BluetoothDevice bt : pairedDevices){
83                 btlist.add(bt);
84             }
85         } else{
86             Toast.makeText(getApplicationContext(), "
There is no paired Bluetooth devices!", Toast.LENGTH_LONG
).show();
87         }
88         ArrayAdapter<BluetoothDevice> adapter = new
DeviceListAdapter(this, android.R.layout.
simple_list_item_1,btlist);
89         deviceList.setAdapter(adapter);
90         deviceList.setOnItemClickListener(new
DeviceClickListener(this));
91     }
92
93     @Override
94     public boolean onKeyDown(int keyCode, KeyEvent
event)
95     {
96
97         if (keyCode == KeyEvent.KEYCODE_BACK)
98         {
99             Intent intent = new Intent(this,
MainActivity.class);
100             startActivity(intent);
101             DeviceListActivity.this.finish();
102         }
103         return super.onKeyDown(keyCode, event);
104     }
105 }
106
107

```

Código de desenvolvimento das funcionalidades do APK Robolazul

```

1 package com.example.rperhs.robolazul;
2
3 import android.bluetooth.BluetoothDevice;
4 import android.content.Intent;
5 import android.view.View;
6 import android.widget.AdapterView;
7
8 /**
9  * Created by RPERHS on 06/07/2016.
10  */
11 public class DeviceListListener implements AdapterView.
    OnItemClickListener {
12
13     private final DeviceListActivity activity;
14     public DeviceListListener(DeviceListActivity activity)
15     {
16         this.activity = activity;
17     }
18     public void onItemClick(AdapterView<?> adapterView,
    View view, int posicao, long idPosicao)
19     {
20         Intent i = new Intent(activity, MainActivity.class
    );
21         BluetoothDevice bt = (BluetoothDevice)activity.
    getDeviceList().getItemAtPosition(posicao);
22         i.putExtra("itemSelecionado", bt.getAddress());
23         String robot = (String) activity.robot;
24         i.putExtra("robot", robot);
25         activity.startActivity(i);
26         activity.finish();
27     }
28 }
29

```

APENDICE B – Regras da competição WINTER CHALLENGE XII



Regras SUMÔ

Documento Revisado 28/06/2016 17:30

1.	Introdução	03
2.	Especificações dos Participantes	03
3.	Especificações dos Robôs	03
4.	O Dohyô (Ringue de Sumô)	05
5.	A Partida de Sumô	06
6.	Início, Paralisação, Continuação e Término de uma Partida	07
7.	Duração das Partidas	08
8.	Yukô	08
9.	Penalidades	09
10.	Lesões e Acidentes Durante a Partida	09
11.	Declaração de Objeções	10

1. Introdução

- Nome da Modalidade: Sumô
- Número de Robôs por Partida: Dois
- Duração da Partida: 3 minutos
- Classes Disponíveis: 25g-3 kg (1oz-6.6 lbs) *
- Dimensões máximas dos Robôs: Verificar item 2: "Especificações dos Robôs"
- Especificações do Dohyō: Verificar item 4.2: "Especificações"
- Especificações de Controle: Autônomos ou Rádio-Controlados **

* Verifique a disponibilidade de classes para cada evento.

** Autônomos competem contra autônomos e rádio-controlados contra rádio-controlados, são classes separadas e não competem uma contra a outra.

ATENÇÃO: Cada robô poderá participar apenas de uma única categoria durante o evento, por exemplo: um robô inscrito na categoria de combate não poderá participar na categoria do hockey e vice-versa.

2. Especificações dos Participantes

- 2.1. As modalidades Junior são única e exclusivamente destinada aos alunos do Ensino Fundamental e Médio com idade inferior a 18 anos;
- 2.2. Qualquer outro participante que não se enquadre nas condições estabelecidas no item 2.1 não poderá ser inscrito nas categorias Junior;
- 2.3. Todos os participantes da categoria Junior deverão apresentar uma autorização de seus pais ou responsáveis permitindo a participação no evento;
- 2.4. Não existe uma quantidade máxima de integrantes por equipe;
- 2.5. Na categoria Junior, é permitida a participação de um orientador com idade superior a 18 anos, responsável pela equipe, porém qualquer interferência do orientador na construção ou programação identificada pela organização da competição resultará na eliminação da equipe.

3. Especificações dos Robôs

3.1 Especificações

- 3.1.1 O robô deverá caber em um cubo ou quadrado com as dimensões referentes à sua classe.
- 3.1.2 A massa total do robô no início da partida deverá ser menor ou igual ao peso designado para sua respectiva classe:

Classe	Altura	Largura	Comprimento	Peso
3 kg Sumô	Ilimitada	20 cm	20 cm	3000 g
Mini Sumô	Ilimitada	10 cm	10 cm	500 g
Micro Sumô	5 cm	5 cm	5 cm	100 g
Nano Sumô	2.5 cm	2.5 cm	2.5 cm	25 g
Lego Sumô	Ilimitada	15.2 cm	15.2 cm	1000 g
Lego Sumô – Junior	Ilimitada	15.2 cm	15.2 cm	1000 g
Humanóide (RoboOne)	50 cm	20 cm	20 cm	3000 g

- 3.1.3 Robôs das classes Nano (25g), Micro (100g), Mini (500g), Lego (1000g) deverão ser obrigatoriamente autônomos. Robôs de 3 kg ou Humanóides poderão ser tanto autônomos quanto rádio-controlados.
- 3.1.4 Para robôs autônomos qualquer método de controle poderá ser empregado, desde que esteja completamente contido no robô e que não interaja com um sistema de controle externo (humano ou máquina).

3.1.5 Os robôs autônomos deverão entrar em operação automaticamente em não menos do que cinco (5) segundos após autorização do juiz e comando dado por um membro da equipe.

3.1.6 As classes de 3 kg e Humanóide podem ser rádio-controladas, conforme as regras da FCC (*Federal Communications Commission*) e aprovação dos organizadores do evento. Robôs rádio controlados deverão iniciar ao sinal dado pelo juiz principal.

Abaixo seguem as frequências e modulações permitidas:

- Frequências: 27MHz, 72MHz, 75MHz, 333MHz, 433MHz, 900MHz e 2.4GHz;
- Modulações: AM, FM, PCM, PPM e enlaces únicos, como DSM e outras;
- Demais frequências e modulações deverão ser aprovadas pela organização.

No caso de dois robôs que possuam a mesma frequência virem a se enfrentar, a preferência será do primeiro a ter feito a inscrição.

3.1.7 É obrigatória a fixação do nome do robô em uma superfície visível, permitindo que os espectadores e organizadores do evento possam identificar facilmente os robôs envolvidos na partida.

3.1.8 Os robôs poderão expandir seu tamanho após o início da partida, porém não será permitido se separar fisicamente devendo continuar como um único robô. A violação desta regra implicará na perda da partida. O desprendimento de peças, cujo o somatório de suas massas, seja inferior a 10g, não implicará na perda da partida. Caso um robô seja prejudicado por uma peça que tenha se desprendido de seu adversário, a ele será dado um ponto de Yuko.

3.2 Restrições

3.2.1 Dispositivos para interferência, tais como, porém, não limitados a sistemas de LEDs infravermelhos (IR) com intenção de saturar os sensores dos oponentes, não são permitidos.

3.2.2 Peças que possam quebrar ou danificar o *Dohyō* não são permitidas, tais peças serão avaliadas pelos juizes na inspeção de segurança, podendo ou não ser liberada para o uso. Não utilize peças que tenham, ou não, a intenção de danificar o robô adversário, seu operador e/ou *Dohyō*. Impactos e colisões normais não serão considerados como danos intencionais.

3.2.3 Dispositivos que possam armazenar líquido, pó, gás ou outras substâncias com intenção de lançá-las no oponente não são permitidos.

3.2.4 Nenhum dispositivo inflamável será permitido.

3.2.5 Dispositivos que lancem quaisquer objetos no oponente não são permitidos.

3.2.6 Substâncias para melhorar a tração não são permitidas. Pneus e outros componentes do robô que entrem em contato com a arena não devem ser capazes de pegar e segurar um cartão de 80 x 130 mm, com gramatura 180gr (cartolina) por mais de 2 (dois) segundos.

3.2.7 Dispositivos para aumentar a força normal, tais como bombas de vácuo ou ímãs, somente são permitidos na classe de 3 kg, não sendo permitidos nas outras classes.

3.2.8 Todas as quinas, incluindo, porém não limitado, a pá frontal, não podem ser afiadas o suficiente para arranhar ou danificar o *Dohyō*, outros robôs ou membros das equipes.

4. O *Dohyō* (Ringue de Sumô)

4.1 Interior

O interior do *Dohyō* é a superfície onde são realizadas as partidas, circulada por uma linha de borda, que também é parte integrante do *Dohyō*. Qualquer lugar fora dessa área delimitada é chamado de parte exterior.

4.2 Especificações

4.2.1 O *Dohyō* terá formato circular, e deve ser de tamanho e material apropriados para as respectivas classes de peso:

Detalhes do Dohyô

Classe	Espessura	Diâmetro	Material
3 kg/Humanóide	5,00 mm	154,0 cm	Placa de aço
Mini/Lego	2,50 cm	77,0 cm	Placa de MDF
Micro	1,25 cm	38,5 cm	Placa de MDF
Nano	0,625 cm	19,25 cm	Placa de MDF

4.2.2 As Shikiri (linhas de início) consistem em duas linhas marrons (ou equivalentes para absorção de luz infravermelha - IR) centradas no Dohyô com espessura e espaçamento para a respectiva classe. A distância de separação entre as linhas é medida pelos limites externos das mesmas.

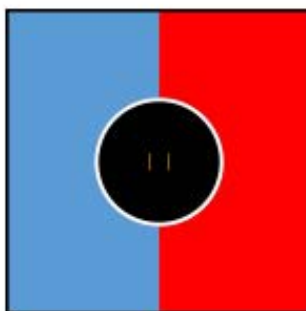
4.2.3 A linha de borda é uma faixa circular com a largura apropriada para a respectiva classe na extremidade externa da superfície de disputa. A área do Dohyô se estende até a borda externa dessa faixa circular.

Pintura do Dohyô

Classe	Largura da Shikiri	Comprimento da Shikiri	Separação da Shikiri	Largura da Borda
3 kg/Humanóide	2 cm	20 cm	20 cm	5 cm
Mini/Lego/Sumô	1 cm	10 cm	10 cm	2,5 cm
Micro Sumô	0,5 cm	5 cm	5 cm	1,25 cm
Nano Sumô	0,25 cm	2,5 cm	2,5 cm	0,625 cm



4.3 **Exterior do Dohyô:** Deverá haver um espaço apropriado para a classe em questão além da borda exterior do Dohyô. Esse espaço pode ser de qualquer cor, material ou formato, contanto que os conceitos básicos dessas regras não sejam violados. Esta área, em cujo meio fica o Dohyô, deverá ser chamada de "área do Dohyô". Quaisquer partes da plataforma do Dohyô além das dimensões mínimas serão consideradas como dentro da área do Dohyô. Durante a realização de um round não será permitido a nenhum competidor entrar na área do Dohyô.



5. A Partida de Sumô

- 5.1. A partida é disputada por duas equipes, cada uma composta por um ou mais integrantes. Apenas é permitido dois membros de cada equipe na área de realização das partidas, enquanto os demais

membros deverão assistir a disputa junto ao público. Cada equipe competirá no *Dohyō* com um robô construído de acordo com as especificações do Item 3. A partida será iniciada ao comando do juiz principal e continua até um competidor conquistar dois pontos de Yukō. O vencedor da partida é determinado pelo juiz.

- 5.2. Uma partida consistirá em 3 (três) *rounds*, cada *round* terá um tempo nominal de 1 (um) minuto, podendo ser acrescentado, a critério do juiz, 30 segundos totalizando um tempo total de 1 minuto e 30 segundos (1' 30").
- 5.3. O juiz poderá decretar o fim de um *round* ao perceber que não está ocorrendo qualquer tipo de evolução por parte dos dois robôs competidores, sendo decretado empate nesse *round*.
- 5.4. A equipe que ganhar dois *rounds* ou receber 2 (dois) pontos de Yukō primeiro, dentro do tempo limite, será declarada vencedora. Uma equipe recebe um ponto de Yukō quando vence um *round*. Caso o tempo limite seja atingido antes de uma equipe atingir dois pontos de Yukō e uma das equipes tenha recebido um ponto de Yukō, enquanto a outra não tiver ganhado pontos, esta será a vencedora.
- 5.5. Caso a partida não for vencida por nenhuma equipe dentro do tempo limite, a decisão será realizada pelos juizes, por meio de pontuação seguindo critérios apresentados no Item 8.3.
- 5.6. Um ponto de Yukō será dado ao vencedor quando a decisão dos juizes ou por pontos for empregada.
- 5.7. É permitido ao competidor um tempo máximo de 5 minutos para troca de baterias dos robôs entre dois *rounds*.
- 5.8. É permitido ao competidor alterar a programação de seus robôs entre duas partidas, porém, é proibida qualquer tipo de alteração durante a partida, ou seja, entre cada um dos *rounds*.

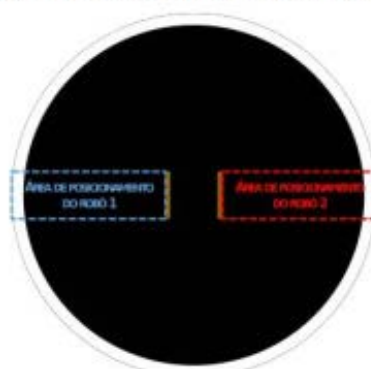
6. Início, Paralisação, Continuação e Término de uma Partida

- 6.1 **Início:** Mediante as instruções do juiz, as duas equipes se cumprimentam com uma reverência ao lado do *Dohyō*, se aproximam do *Dohyō* e posicionam o seu robô dentro da sua metade do *Dohyō*, atrás da linha *Shikiri* e/ou de acordo com as instruções do juiz.

Ao sinal do juiz os dois robôs devem ser simultaneamente posicionados com pelo menos uma parte de sua estrutura dentro da região delimitada pela linha *Shikiri* de sua metade do *Dohyō*, não sendo mais possível alterar o posicionamento dos robôs.

Não existe qualquer tipo de restrição quanto a orientação dos robôs, desde que respeitando as características supracitadas.

Robôs da categoria Lego® serão colocados no *Dohyō* de acordo com a indicação do juiz.



Nas partidas entre robôs autônomos, quando o juiz principal anunciar o início do *round*, um membro de cada time ativará os robôs e após uma pausa de 5 segundos os robôs podem começar a funcionar. Durante esses 5 segundos os membros das equipes devem deixar a área do *Dohyō*. Caso o robô necessite de qualquer tipo de dispositivo externo para iniciar ou finalizar o programa de controle do robô, este dispositivo deverá ficar dentro do recipiente posicionado no chão do lado de fora da área do

Dohyô.

Nas partidas entre robôs rádio controlados os pilotos deverão estar posicionados fora da área do *Dohyô* juntamente com os juizes. Assim que for anunciado o início da rodada pelo juiz, os robôs poderão começar a funcionar. Somente neste caso não é necessário haver o intervalo de 5 segundos.

- 6.2 **Paralisação e Continuação:** A partida é paralisação ou retomada conforme os anúncios dos juizes.
- 6.3 **Término:** A partida termina quando anunciado pelo juiz principal. Então as duas equipes recolherão os seus respectivos robôs da área do *Dohyô*.

7. Duração das Partidas

- 7.1. Uma partida será disputada por um total de 3 (três) minutos, iniciando e terminando mediante o comando do juiz, exceto nas partidas em que ocorreram *rounds* com extensão.
- 7.2. Um *round* estendido, quando concedido pelo juiz, deverá durar no máximo 1 minuto e 30 segundos.
- 7.3. Os seguintes períodos não são incluídos no tempo da partida:
 - 7.3.1. O tempo entre o anúncio de um *Yukô* pelo juiz e a continuação da partida. O intervalo padrão entre os *rounds* será de 30 (trinta) segundos.
 - 7.3.2. O tempo decorrido entre o anúncio da paralisação e a continuação da partida.

8. Yukô

- 8.1. Um ponto de *Yukô* será concedido quando:
 - 8.1.1. Um robô legalmente forçar o corpo do robô oponente a tocar o espaço fora do *Dohyô*.
 - 8.1.2. Quando o robô adversário tocar o espaço externo ao *Dohyô* por si mesmo.
 - 8.1.3. No caso do sumô humanóide, qualquer parte do robô adversário exceto a sola de seus pés (mãos, joelhos, costas, tórax, etc.) toque o *Dohyô*, quando ele é empurrado ou jogado para fora do *Dohyô*.
 - 8.1.4. Qualquer uma das situações acima ocorra ao mesmo tempo em que o fim da partida é anunciado.
 - 8.1.5. Quando o robô adversário perder uma ou mais peças com somatório de massas superior a 10g.
- 8.2. Um ponto de *Yukô* não será concedido quando:
 - 8.2.1. Um robô movido por rodas capotar no *Dohyô*, um ponto de *Yukô* não será contado e o *round* continuará até que aconteça uma das situações previstas no item 8.1.
- 8.3. Quando a decisão dos juizes for necessária para decidir o vencedor, os seguintes pontos serão levados em consideração, porém, prevalecendo sempre a decisão final do juiz:
 - 8.3.1. Méritos técnicos na movimentação e operação do robô
 - 8.3.2. Penalidades durante a partida
 - 8.3.3. Peso dos robôs
 - 8.3.4. Atitude dos competidores durante a partida
- 8.4. Um *round* em progresso será terminado sob as seguintes condições:
 - 8.4.1. Os robôs estarem presos ou orbitando um ao outro sem progresso perceptível por 5 (cinco) segundos. Em caso de dúvida quanto ao progresso da partida, o juiz poderá estender o tempo limite de observação por até 30 segundos.

- 8.4.2. Ambos o robôs se moverem sem fazer progresso ou pararem exatamente ao mesmo tempo, sem se tocar, por 5 (cinco) segundos. Entretanto, caso um robô pare de se movimentar antes, depois de 5 (cinco) segundos será declarado como sem vontade de lutar, neste caso o oponente receberá um ponto de Yukô, mesmo que depois este pare de funcionar. Caso ambos os robôs estejam se movendo e não esteja claro se há progresso, o juiz poderá estender o tempo da partida até 30 (trinta) segundos.
- 8.4.3. Caso ambos os robôs toquem o lado de fora do *Dohyô* ao mesmo tempo, não podendo ser determinado qual o fez primeiro.
- 8.5. **Violações:** Competidores fazendo qualquer uma das ações descritas nos itens 3.2, 8.6 ou 8.7, estarão violando as regras.
- 8.6. **Insultos:** Um competidor que insulte seu oponente ou os juizes, coloque um dispositivo de voz em seu robô com intenção de insultar, escreva palavras de teor ofensivo no corpo do robô ou faça qualquer ação ofensiva, estará violando essas regras.
- 8.7. **Violações Brandas:** Uma violação branda é declarada se um competidor:
- 8.7.1. Entrar na área do *Dohyô* durante a partida, exceto quando o competidor o faça para retirar o seu robô mediante o anúncio, feito pelo juiz, de um ponto de Yukô ou paralisação da partida. Entrar na área do *Dohyô* significa:
- 8.7.1.1. Uma parte do corpo do competidor está dentro da área do *Dohyô*
- 8.7.1.2. Um competidor coloca qualquer dispositivo mecânico dentro da área do *Dohyô* para apolar o seu corpo.
- 8.7.2. Faça as seguintes ações:
- 8.7.2.1. Peça a paralisação da partida sem razões apropriadas
- 8.7.2.2. Demore mais de 30 segundos para retomar a partida, exceto caso o juiz anuncie uma extensão de tempo
- 8.7.2.3. Cujo robô comece a operar antes de 5 (cinco) segundos após o juiz principal anunciar o início da partida
- 8.7.2.4. Faça ou fale algo que ponha em risco a equidade da partida

9. Penalidades

- 9.1. Competidores que violarem os itens 3.2 e 8.6 dessas regras perderão a partida. O Juiz dará dois pontos de Yukô ao oponente e pedirá ao(s) violador(es) que saiam da área do *Dohyô*. O(s) violador(es) não terá(ão) nenhum direito.
- 9.2. Cada violação branda descrita no item 8.7 será acumulativa. Duas destas violações serão convertidas em um ponto de Yukô para o adversário.
- 9.3. As violações descritas no item 8.7 deverão ser acumuladas apenas durante a partida em vigência.
- 9.4. Qualquer dano causado ao *Dohyô* de maneira intencional será punida com dois pontos de Yukô ao oponente decretando o fim da partida.

10. Lesões e Acidentes Durante a Partida

- 10.1. **Requisição de Paralisação da Partida:** Um competidor pode requerer a paralisação da partida quando ele ou algum membro de sua equipe sofrer uma lesão ou sofrer um acidente e não puder continuar a partida. Lesões e acidentes causados propositalmente para paralisar a partida implicarão na desclassificação da equipe da competição.
- 10.2. **Impossibilidade de Continuar a Partida:** Quando uma partida não puder continuar devido à lesão ou acidente com um competidor, o causador dessa lesão ou acidente perderá a partida. Quando não for

claro qual equipe é a responsável, aquela que não puder continuar a partida ou que solicite a paralisação será declarada a perdedora.

- 10.3. **Requisição por Parte de uma Equipe para Cuidar de uma Lesão ou Acidente:** A decisão sobre a continuação de uma partida paralisada em caso de lesão ou acidente ficará a cargo dos juizes. Tal processo não poderá demorar mais de 5 (cinco) minutos.
- 10.4. **Yukô Dado ao Competidor que Não Continuará:** O vencedor determinado de acordo com o Item 10.2 deverá ganhar dois pontos de Yukô. O perdedor que já tiver conquistado um ponto de Yukô será computado como tal. Quando a situação descrita no Item 10.2 ocorrer durante uma partida estendida, o ganhador receberá apenas um ponto de Yukô.

11. Declaração de Objeções

- 11.1. Nenhuma objeção deverá ser declarada contra a decisão dos juizes.
- 11.2. O Capitão de uma equipe pode apresentar objeções aos organizadores do evento antes do fim da partida, se houver dúvidas quanto à correta aplicação dessas regras. Caso não haja nenhum membro da organização do evento presente, as objeções poderão ser apresentadas aos juizes antes do fim da partida.

12. Aspectos Gerais

- 12.1. Todas as quinas afiadas do robô deverão estar por todo o tempo protegidas, até que seja solicitado a sua remoção para início do round.
- 12.2. Todos os robôs devem manter suas rodas suspensas por todo o tempo.