UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVIDIO NUNES DE BARROS
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ALLAN JHEYSON RAMOS GONÇALVES

APLICATIVO EM ANDROID PARA CONTROLE DE UNIDADES ROBÓTICAS MÓVEIS COM ARDUINO

PICOS – PI
2013

ALLAN JHEYSON RAMOS GONÇALVES

APLICATIVO EM ANDROID PARA CONTROLE DE UNIDADES ROBÓTICAS MÓVEIS COM ARDUINO

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí – UFPI, como requisito final para obtenção do título de Bacharel em Sistemas de Informação.

ORIENTADOR: PROF. MSc. ALGEIR PRAZERES SAMPAIO

PICOS - PI
2013

ALLAN JHEYSON RAMOS GONÇALVES

APLICATIVO EM ANDROID PARA CONTROLE DE UNIDADES ROBÓTICAS MÓVEIS COM ARDUINO

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros da Universidade Federal do Piauí – UFPI, como requisito final para obtenção do título de Bacharel em Sistemas de Informação.

Data de Aprovação:

_____
Prof. Algeir Prazeres Sampaio, MSc. (Orientador)

_____
Prof. Ismael de Holanda Leal, Esp. (Membro)

_____
Prof. Ivenilton Alexandre de Souza Moura, Esp. (Membro)

Eu, **Allan Jheyson Ramos Gonçalves**, abaixo identificado(a) como autor(a), autorizo a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação abaixo discriminada, de minha autoria, em seu site, em formato PDF, para fins de leitura e/ou impressão, a partir da data de hoje.

Picos-PI, 18 de abril de 2013.

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

G635a Gonçalves, Allan Jheyson Ramos.
Aplicativo em Android para controle de unidades robóticas móveis com Arduino / Allan Jheyson Ramos Gonçalves. – 2013.
CD-ROM : il. ; 4 ¾ pol. (57 p.)

Monografia(Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí. Picos-PI, 2013.
Orientador(A): Prof. MSc. Algeir Prazeres Sampaio

1. Android. 2. Arduino. 3. Robótica. I. Título.

CDD 005.1

Dedico este trabalho à minha família, em especial aos meus pais, Valdenora Léa Ramos e Reginaldo Borges Gonçalves, que sempre acreditaram no meu potencial e me deram total apoio ao longo desta caminhada, e ao meu avô, Valdemar de Moura Ramos (*In Memoriam)*, pelo exemplo de perseverança, os ensinamentos de vida compartilhados e pela inspiração para esta conquista que sempre foi tão importante para ele quanto é para mim.

AGRADECIMENTOS

A todos os meus colegas de turma por todos os momentos de aprendizado proporcionados ao longo desta jornada turbulenta e cheia de surpresas.

Ao meu orientador, Prof. MSc. Algeir Prazeres Sampaio, pelo auxílio e disposição para desenvolver este trabalho que não seria possível sem o seu conhecimento e sua experiência na área de TI.

Aos demais professores da UFPI campus Picos, que foram fundamentais para o meu crescimento profissional.

À minha mãe, Valdenora Léa Ramos, pelo espírito guerreiro e persistente de ser, que sempre foi referência para mim a cada nova etapa de desafios durante esta jornada.

Ao meu pai, Reginaldo Borges Gonçalves, pela orientação e pelas palavras de apoio a cada momento de insegurança que passei durante esta importante fase da minha vida.

Às minhas tias: Valderina Léha Ramos, Valdinez Léa Ramos e Vanusa Léha Ramos, por oferecerem a mim todo o afeto que uma mãe pode oferecer ao filho.

A todos os familiares e amigos que sempre me apoiaram e me fizeram crescer como pessoa e como profissional.

“O covarde nunca tenta, o fracassado nunca termina e o vencedor nunca desiste.”

(Norman Vicent Peale)

RESUMO

A comunicação entre dispositivos eletrônicos vem desenvolvendo-se de forma cada vez mais objetiva e diversificada. Esta evolução criou a necessidade de implantação de interfaces comuns entre os diversos tipos de dispositivos produzidos. O objetivo deste trabalho é desenvolver um sistema capaz de manipular um robô que utiliza a plataforma Arduino através de sinais digitais emitidos pelas interfaces de comunicação Bluetooth e Wi-Fi de um smartphone com sistema operacional Android instalado. O projeto é motivado pelo grande crescimento do uso da plataforma Arduino na área da robótica para a prototipação de novas ideias, atrelado ao crescimento do uso do sistema operacional Android nos dispositivos móveis em todo o mundo. O sistema foi desenvolvido para o controle de um robô móvel, mais precisamente um carro robô, com movimentos de avançar, retroceder e girar para ambos os lados. O usuário tem a opção de escolher qual interface deseja utilizar para controlar o robô e a partir daí, terá acesso ao modo de controle.

Palavras chave: Bluetooth. Controle. Robótica. Android. Arduino.

ABSTRACT

The communication between electronic devices is evolving from an increasingly diverse and objective. This development has created the need to implement common interfaces between different types of devices produced. The objective of this work is to develop a system capable of handling a robot that uses the Arduino platform digital signals emitted by communication interfaces Bluetooth and Wi-Fi on a smartphone with Android operating system installed. The project is motivated by the large growth in the use of the Arduino platform in robotics for prototyping new ideas, linked to the growing use of the Android OS on mobile devices worldwide. The system was developed to control a mobile robot, more precisely a robot car with movements forward, backward and turn to both sides. The user has the option to choose which interface you want to use to control the robot and from there, you will have access to the control mode.

Keywords: Bluetooth. Control. Robotic. Android. Arduino.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

2D	2 Dimensões
3D	3 Dimensões
CC	Corrente Contínua
IDE	*Integrated Development Environment*
IP	*Internet Protocol*
LED	*Light Emitting Diode*
LIPPO	Laboratório de Investigações e Pesquisas em Poéticas Digitais
MAC	*Media Access Control*
OS	*Operational System*
PWM	*Pulse Width Modulation*
RX	Via de Recepção de Dados
SDK	*Software Development Kit*
SMS	*Short Message Service*
SQL	*Structured Query Language*
TX	Via de Transmissão de Dados
USB	*Universal Serial Bus*

SUMÁRIO

Capítulo 1

INTRODUÇÃO

É cada dia mais comum a utilização de robôs móveis nas mais diversas atividades pessoais, profissionais e científicas. Todas as atividades que envolvem manipulação de unidades móveis exigem alguma forma de controle sobre esta, seja manualmente ou automaticamente realizada através de sensores e atuadores embarcados.

O uso de aplicações em dispositivos móveis para controle de robôs é uma alternativa bastante adequada para este fim, já que os aparelhos produzidos atualmente, mais conhecidos como *smartphones* possuem no mínimo uma interface para comunicação direta com outros dispositivos. Em sua grande maioria, possuem interface para comunicação *Wi-Fi* e *Bluetooth.*

Os smartphones funcionam com um sistema operacional embarcado e o Android OS vem ganhando cada vez mais espaço no mercado entre os utilizadores de dispositivos móveis. Isto se deve a sua distribuição livre e gratuita, além de disponibilizar aos usuários o seu código-fonte para que possam personalizar o sistema e desenvolver aplicativos livremente. Esta facilidade e flexibilidade fazem do Android um sistema operacional popular e com muitas possibilidades para o desenvolvimento de novos projetos.

Em uma conferência da Google em 2011, foi realizada uma demonstração em que ficou muito claro que o Android tem como novo objetivo o desenvolvimento de projetos que possam facilitar tarefas das pessoas no cotidiano. Para fazer isto é necessária a união de hardware e software e a alternativa encontrada foi incorporar a plataforma Arduino para auxiliar neste desenvolvimento.

Muitos projetos de robótica envolvem a utilização da plataforma Arduino, que é uma alternativa bastante expansível e descomplicada para o desenvolvimento de protótipos. Trata-se basicamente de um kit de desenvolvimento de software além de um kit de desenvolvimento de acessórios capaz de incorporar inúmeros sensores e módulos de comunicação para interagir com outros aparelhos.

Este trabalho apresenta uma aplicação envolvendo comunicação entre Android e Arduino para controle de um robô móvel. A aplicação poderá ser utilizada

ainda para outras finalidades, bastando realizar a programação adequada no Arduino para que os comandos do aplicativo Android sejam atendidos adequadamente.

1.1 Motivação

O grande avanço da robótica nos últimos anos vem despertando os estudiosos da área para a criação de novas possibilidades e aplicações utilizando sofisticados mecanismos atuadores. Robôs estão sendo utilizados nas mais diversas situações e estas cada vez menos prováveis para a concepção humana.

Torna-se evidente a necessidade de exploração dos mais diversos meios de controle para unidades robóticas móveis no atual cenário tecnológico que estamos vivendo. Muitas formas de comunicação podem ser aplicadas com a utilização da plataforma Arduino, dentre elas podem ser citadas o Bluetooth, Wi-Fi, Rádio Frequência, Infravermelho, dentre outras.

Esta grande variedade de formas de comunicação tornam o uso do Arduino cada vez mais crescente em projetos de robótica e automação, este foi um dos fatores que mais fortaleceram a ideia deste projeto. As possibilidades de criar um novo produto ou serviço utilizando esta tecnologia são inúmeras e por isso é cada vez maior o número de desenvolvedores que optam por utilizar o Arduino na concepção de um novo trabalho na área.

Fatores como estes fortalecem ainda mais os trabalhos e consequentemente tornam o processo de resolução de possíveis problemas ou imprevistos que venham a ocorrer durante a implementação, mais simples e rápidos através da troca de informações em fóruns de discussão e sites especializados.

1.2 Problema

A implementação de aplicativos em Android tem, em sua maioria, foco em proporcionar entretenimento e/ou comunicação entre pessoas. Este trabalho busca proporcionar uma forma de comunicação entre dispositivos como objetivo final, neste caso, entre Android e Arduino. A manipulação de robôs através de computadores já é realidade atualmente, mas não há um consenso ou uma forma

padrão de utilização destes controladores, muito menos no que tange aos dispositivos móveis.

Nos últimos anos a plataforma Arduino se tornou muito popular pela facilidade e rapidez na construção de um novo protótipo. Isto atraiu muitos novos entusiastas para a área da robótica.

A ideia é produzir um aplicativo “padrão” para o sistema operacional Android para que este possa ser utilizado para controlar qualquer robô móvel terrestre que utilize a plataforma Arduino, bastando que seja implementado um pequeno código de recepção dos dados na plataforma, que também é disponibilizado neste trabalho. Esta padronização poderia facilitar futuros projetos relacionados a este, evitando o retrabalho com a implementação da comunicação entre o Android e o Arduino.

1.3 Abordagem

Neste trabalho implementamos uma aplicação para o sistema operacional Android com a finalidade de obter uma forma de controle para unidade robóticas móveis que utilizem a plataforma Arduino, descrevendo suas funcionalidades e possibilidades de aplicação pelo usuário. Foi utilizada a linguagem de programação Java na IDE Eclipse no ambiente Windows com o SDK do Android para o desenvolvimento do aplicativo.

1.4 Objetivos

1.4.1 Objetivo geral

- Construir um aplicativo para o sistema operacional Android visando controlar uma unidade robótica com Arduino.

1.4.2 Objetivos específicos

- Obter uma comunicação estável entre Android e Arduino;
- Adquirir conhecimento sobre o desenvolvimento de aplicações para o sistema operacional Android;

- Construir uma unidade robótica móvel utilizando a plataforma Arduino;

1.5 Organização do documento

Este documento está organizado em 6 capítulos, onde o Capítulo 1 refere-se a definição da área do trabalho, a motivação para realizá-lo, o problema abordado e algumas contribuições do projeto para a área de estudo.

O Capítulo 2 traz o embasamento para a aplicação através do referencial teórico, abordando trabalhos e estudos já consolidados por outros autores e estudiosos da área da robótica, Android OS e da plataforma Arduino.

No Capítulo 3 é abordada a concepção do projeto, descrevendo alguns fatores que influenciaram para que o estudo pudesse ser realizado.

O Capítulo 4 mostra como a aplicação foi desenvolvida, seu funcionamento e algumas imagens da sua interface gráfica e recursos de controle sobre o Arduino.

Em seguida, no Capítulo 5 são descritos os resultados dos testes realizados com a aplicação e um robô em ambiente aberto, no próprio Campus Universitário da UFPI, Picos.

Concluindo, o Capítulo 6 apresenta uma síntese do trabalho, abordando alguns aspectos que influenciaram o desenvolvimento do projeto.

Capítulo 2

REFERENCIAL TEÓRICO

2.1 Android OS

A empresa Google, há alguns anos percebeu o grande crescimento do nicho de mercado dos dispositivos móveis no mundo. A exigência dos usuários tornou-se maior e os fabricantes desses aparelhos precisavam atender essa crescente necessidade de melhoria e variedade de recursos para os dispositivos. A partir desta percepção, o Google juntou-se com alguns fabricantes de aparelhos celulares e daí surgiu um grupo chamado de *Open Handset Alliance*, com o objetivo de criação de um sistema operacional de código livre para os celulares.

Observando o grande crescimento da utilização da internet em dispositivos móveis, a Google adquiriu o Android Inc., em 2005, para centralizar o seu desenvolvimento em uma plataforma de dispositivos móveis. As ideias revolucionárias da Apple introduzidas no iPhone em 2007 foram rapidamente adaptadas para inclusão destes recursos com distinções definidas, como um maior controle para os desenvolvedores. A *Open Handset Alliance* é um grupo de empresas liderado pelo Google que inclui operadores de telefonia móvel, fabricantes de aparelhos portáteis, fabricantes de componentes, provedores de plataformas e soluções de software para empresas de marketing. A partir de um ponto de vista de desenvolvimento de software, o Android fica bem ao centro do software livre (ABLESON, 2009).

Segundo LECHETA (2009, *apud* SILVA, 2009), o Android é uma plataforma para *smartphones,* baseada no sistema operacional Linux, possui diversos componentes, com uma variada disponibilidade de bibliotecas e interface gráfica, além de disponibilizar ferramentas para a criação de aplicativos.

2.1.1 Arquitetura do Android

O Android possui uma arquitetura dividida em cinco componentes: Aplicação, *Framework* de Aplicação, Bibliotecas, Ambiente de Execução e o *Kernel* do Linux, estrutura mostrada na **Figura 2.1** e melhor detalhada em seguida com uma apresentação das camadas com seus principais componentes e características.

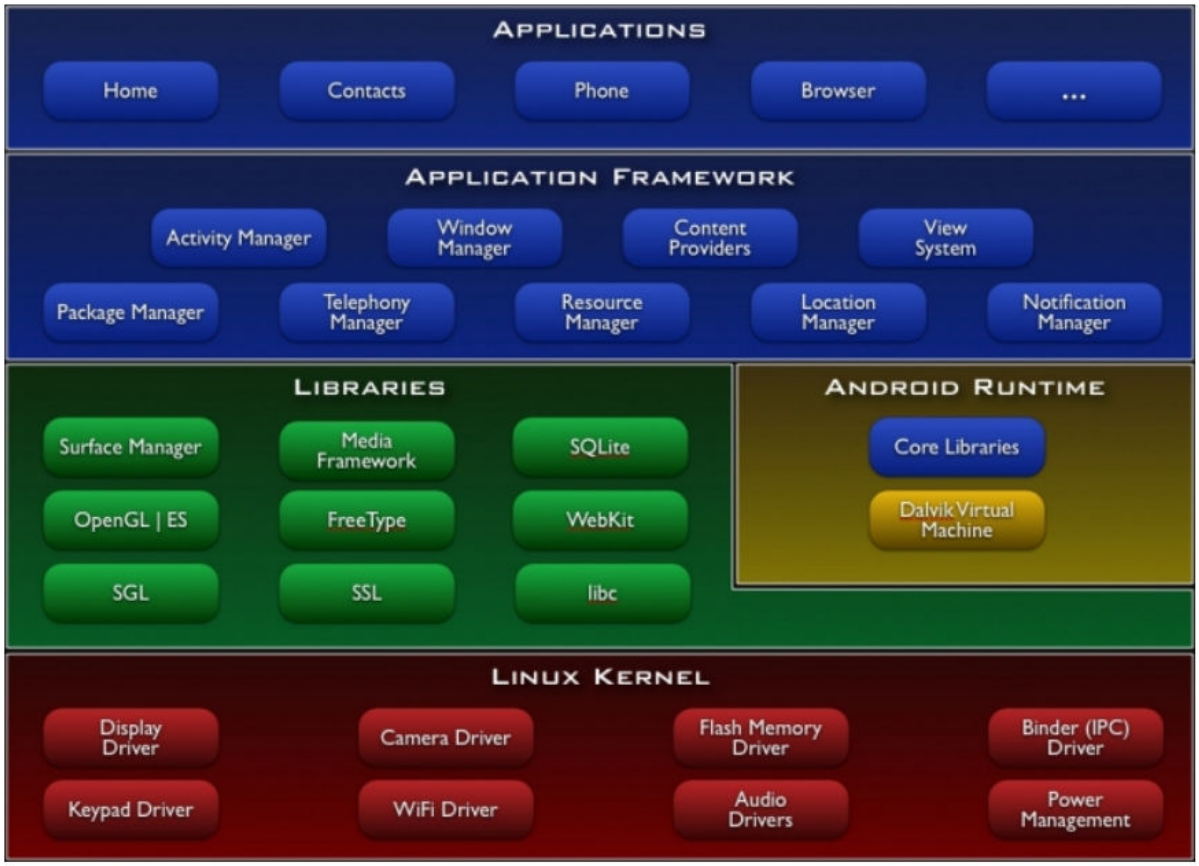


Figura 2.1.1 - Arquitetura do Android

Fonte: <http://developer.android.com>

Applications (Aplicações): o Android possui nativamente diversos aplicativos como programa de envio de mensagem SMS (*Short Message Service*), cliente de e-mail, mapas, calendário, navegador de internet e outros. A linguagem Java é utilizada no desenvolvimento de todos os aplicativos para Android (ANDROID, 2013). Os aplicativos desenvolvidos ficam nesta camada e são o que os usuários finais encontram no Android.

Application Framework (Framework de Aplicação): ambiente que disponibiliza inúmeros recursos para auxiliar o desenvolvedor a realizar o seu trabalho. É a parte do Android que está mais bem documentada e detalhada, pelo fato de ser esta a camada que fornece aos desenvolvedores criativos a possibilidade

de criarem aplicações inovadoras para o mercado (GARGENTA, 2011 *apud* FERNANDES; LAURINDO, 2011). Os recursos disponibilizados pelo Framework de Aplicação permitem o desenvolvimento de novos aplicativos com reaproveitamento das funcionalidades presentes na plataforma. A arquitetura do aplicativo é projetada para simplificar a reutilização de componentes e qualquer aplicação pode criar recursos próprios e estes podem ser disponibilizados para que outra aplicação possa utilizá-los. Este mecanismo permite ainda que componentes padrões sejam substituídos pelo usuário (ANDROID, 2013).

Libraries (Bibliotecas): o Android OS possui bibliotecas nativas C/C++, muitas delas produzidas pela comunidade de código livre. Elas facilitam o desenvolvimento de aplicações e possibilitam o acesso às funcionalidades de hardware. Entre elas estão um mecanismo de renderização web rápido utilizado pelo Safari, Chrome entre outros navegadores, um completo banco de dados SQL (*Structured Query Language*), uma implementação código aberto do Java, bibliotecas de gráfico 3D e camada de certificado de segurança (GARGENTA, 2011 *apud* FERNANDES; LAURINDO, 2011). O banco de dados do Android é o SQLite, que permite que o banco de dados seja utilizado pelos aplicativos para Android. Cada aplicativo pode criar vários bancos de dados que somente estarão disponíveis para a aplicação que os criou, garantindo assim a segurança dos dados do sistema. No SQLite são utilizados comandos SQL padrão, tornando o desenvolvimento de aplicativos que precisam de armazenamento ainda mais fácil (LECHETA, 2009 *apud* FERNANDES; LAURINDO, 2011).

Android Runtime (Ambiente de Execução Android): existe a necessidade de uma máquina virtual Java para execução dos aplicativos, já que todos os aplicativos do Android são escritos na linguagem Java. Esta máquina virtual Java chamada Dalvik é especial para executar aplicações Java em dispositivos móveis (LECHETA, 2009 *apud* FERNANDES; LAURINDO, 2011). Convencionalmente o código fonte Java é compilado utilizando o compilador Java e após esta, o bytecode é executado na JVM (*Java Virtual Machine* – Máquina Virtual Java), mas no Android após a compilação no compilador Java, o código é recompilado pelo compilador Dalvik para gerar um *bytecode* específico. Este *bytecode* Dalvik é que será executado na máquina virtual Dalvik (GARGENTA, 2011 apud FERNANDES; LAURINDO, 2011). Os códigos Java de aplicações para Android resultam na geração do *bytecode* (.class) normalmente, posteriormente este

será compilado para o formato ".dex" (*Dalvik Executable*), sendo esta a aplicação compilada, estes arquivos juntamente com os demais recursos da aplicação são compactados em um arquivo, o *Android Package File* que é o resultado final do aplicativo (LECHETA, 2009 *apud* FERNANDES; LAURINDO, 2011).

***Kernel Linux*:** o framework Java é utilizado no desenvolvimento de aplicações para Android, mas as bibliotecas padrão estão fora deste escopo. As bibliotecas Java utilizadas para o Android são otimizadas para um ambiente de recursos limitados. O Android invoca o *kernel* do Linux para execução de serviços solicitados por aplicações, mas não se trata de um Linux embarcado (STEELE; TO, 2011 *apud* FERNANDES; LAURINDO, 2011). O kernel Linux é responsável pela gerencia dos recursos primários do sistema, como threads, memória, processos, segurança do sistema de arquivos e outros drivers de hardware (LECHETA, 2009 *apud* FERNANDES; LAURINDO, 2011). Cada execução de uma aplicação gera um novo processo no sistema, se necessário por falta de memória ou por inatividade do processo, o Android pode encerrar este processos (LECHETA, 2009 *apud* FERNANDES; LAURINDO, 2011).

2.2 Arduino

Arduino é uma plataforma de prototipagem eletrônica de código aberto baseada em hardware e software flexíveis e fáceis de usar. Destina-se aos mais diversos tipo de pessoas e interesses, como artistas, designers ou qualquer pessoa interessada em criar objetos ou ambientes interativos (ARDUINO, 2013).

A utilização do Arduino na área da robótica é muito frequente pela grande flexibilidade de escalabilidade do sistema em desenvolvimento. É praticamente infinito o número de novas conexões que podem ser estabelecidas entre diversas placas, sensores e atuadores da plataforma. A programação embarcada também é muito atrativa e simples, o microcontrolador da placa é programado usando uma linguagem baseada em Wiring (Muito parecida com C) e o ambiente de desenvolvimento é bastante simples e objetivo.

O ambiente pode ser percebido através dos sensores que captam sinais de acordo com sua finalidade e transmitem esses sinais para o microcontrolador, que é responsável por gerenciar e executar todo o código

embarcado e através disto efetuar alguma ações pré-definidas na programação, como acender um *LED*, por exemplo.

As placas podem ser construídas à mão ou compradas pré-montadas, o software pode ser baixado gratuitamente. Os projetos de hardware estão disponíveis sob uma licença de código aberto, o usuário é livre para adaptá-los às suas necessidades (ARDUINO, 2013). A **Figura 2.2** mostra uma placa Arduino Duemilanove, que pode ser adquirida facilmente através da internet ou em lojas de eletrônica e robótica.

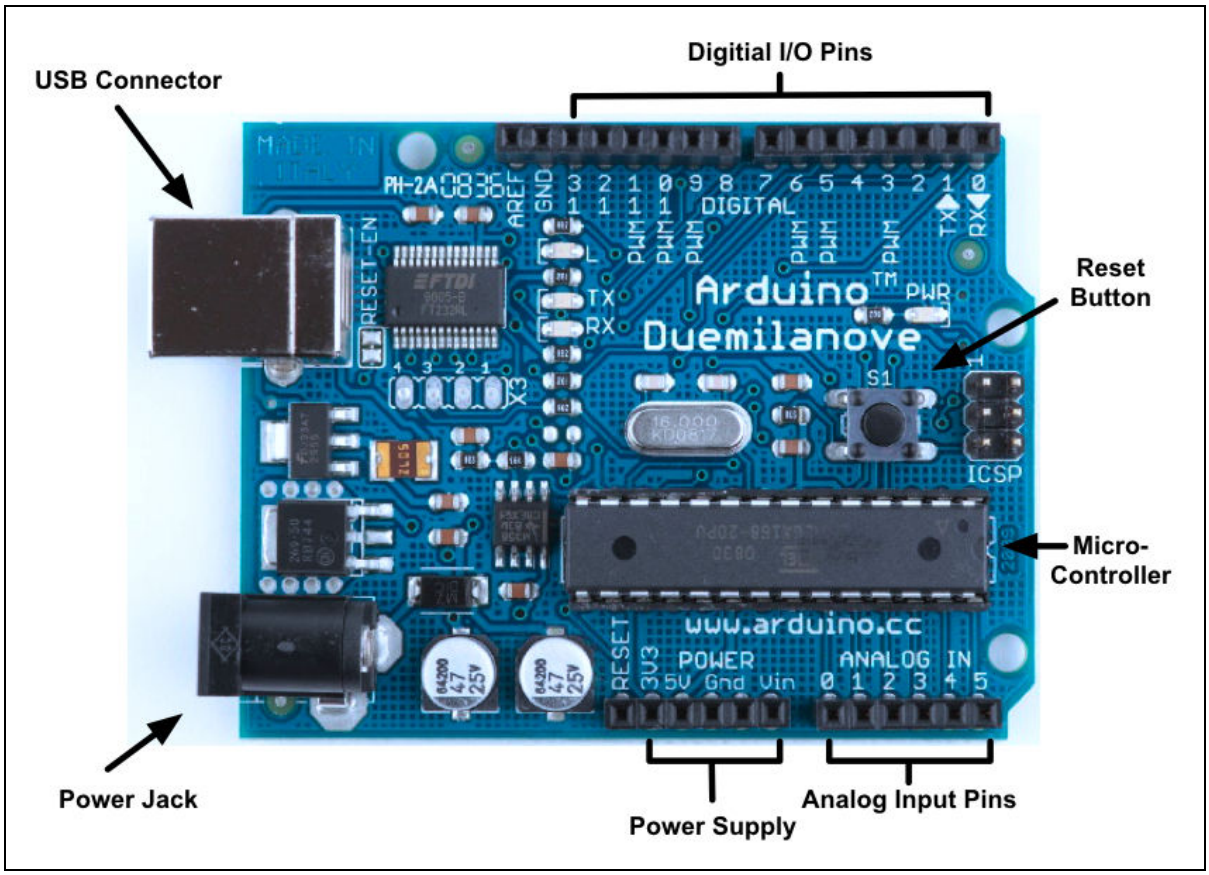


Figura 2.2.1 - Placa Arduino Duemilanove

Fonte: < http://uriedubot.wordpress.com/>

Trabalhar com Arduino é uma tarefa bem simples, não é necessário conhecimento em eletrônica, basta um conhecimento em programação básica e já é possível fazer alguns pequenos projetos com esta plataforma. Para programar a placa, há duas funções que necessitam ser declaradas: a função ***setup*()**, que será executada apenas uma vez, quando a placa for iniciada e a função ***loop*()**, que é

responsável por tratar dos eventos que irão ocorrer após o carregamento da função anterior, é nesta função que o tratamento de ações deve ser implementado.

A estrutura da placa permite a expansão do tamanho e complexidade das conexões e do código de acordo com a necessidade que o projeto requerer. A expansão pode se dar com periféricos conhecidos como Shields, que podem ser acoplados diretamente na placa ou interligados através de *jumpers* utilizando uma placa de ensaio conhecida como *protoboard*, que é uma placa com muitos furos e conexões condutoras para montagem de circuitos elétricos experimentais sem a necessidade de solda. A **Figura 2.3** mostra um Módulo de comunicação Bluetooth (mais à direita) e um chip ponte H, conectados diretamente a uma *protoboard*.

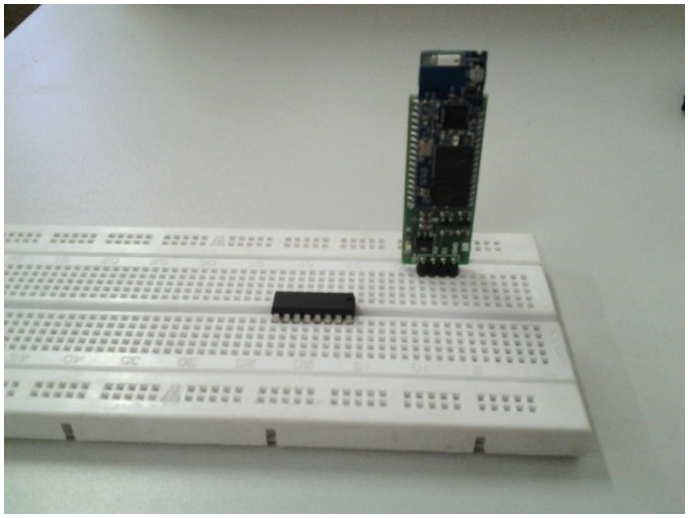


Figura 2.2.2 - *Protoboard* e Módulo Bluetooth

Fonte: o Autor

2.3 Robótica

A robótica é uma ciência que mescla física, mecânica, matemática, *design*, inteligência artificial, eletrônica e programação, e tem por objetivo compreender o processo de controle e montagem de "sistemas que interagem com o mundo real com pouca ou mesmo nenhuma intervenção humana" (MARTINS, 1993 *apud* ZANELATTO, 2004). Os sistemas que processam informações para efetuar ações são chamados de robôs.

O desejo de construir sistemas cada vez mais complexos sempre foi retratado nos filmes de ficção científica e estes foram grandes impulsionadores para

que as ideias pudessem ser estimuladas e desenvolvidas pela ciência. Muitas necessidades de serviços antes supridas totalmente por pessoas, atualmente estão sendo substituídas pela aplicação de robôs, se não completamente, em boa parte dos processos. Esta aplicação se dá principalmente da área industrial, em operações que envolvem grande precisão e esforço.

A palavra robótica (ATTROT, 2002 *apud* ZANELATTO, 2004) foi utilizada pela primeira vez por Isaac Asimov, um escritor que produziu várias obras de ficção científica envolvendo os robôs. Ele também criou as três leis da robótica, considerando que os robôs eram seres que não cometiam erros:

- Um robô não pode ferir um ser humano ou, por inação, permitir que um humano seja ferido;
- Um robô deve obedecer às ordens dadas por humanos, exceto quando isto conflitar com a Primeira Lei;
- Um robô deve proteger sua própria existência, a menos que isto conflite com a Primeira ou a Segunda Lei.

2.3.1 Robótica Móvel

Segundo a RIA (*Robot Institute of America*), um robô é um manipulador re-programável, multifuncional, projetado para mover materiais, peças, ferramentas ou dispositivos especiais, em movimentos variáveis, programados para a realização de uma variedade de tarefas (VIEIRA, 2011). GUERRA, 2009 *apud* VIEIRA, 2011 divide-os em:

- Robôs Manipulados: são dispositivos com vários graus de liberdade e operados manualmente;
- Robôs de Sequência Fixa: são dispositivos manipuladores que desempenham tarefas sucessivas de acordo com um método predeterminado e imutável, muito difícil de ser modificado;
- Robôs de Sequência Variável: dispositivos que desempenham tarefas sucessivas que podem ser facilmente modificadas;
- Robôs Repetitivos: são dispositivos que repetem uma sequência de tarefas gravadas, e são conduzidos ou controlados por um operador humano;

- Robôs de Controle Numérico: o operador alimenta o robô com um programa de movimentação ao invés de treiná-lo manualmente;
- Robôs Inteligentes: apresentam capacidade de compreender seu ambiente e a habilidade de executar tarefas apesar das mudanças que se apresentam no seu meio.

Capítulo 3

CONCEPÇÃO DO PROJETO

3.1 Projetos Atuais do LIPPO

O LIPPO (Laboratório de Investigação e Pesquisa em Poéticas Digitais) foi criado em agosto do ano de 2010, a partir de recursos obtidos através da participação da UFPI em consórcio formado entre a UNB (Universidade de Brasília), UFG (Universidade Federal de Goiás) e a UFPI (Universidade Federal do Piauí em Picos).

Atualmente o LIPPO conta com uma equipe de desenvolvedores de projetos e estagiários com foco na área de robótica, trabalhando tanto na construção de robôs como também na sua codificação embarcada utilizando a plataforma Arduino como microcontrolador do circuito.

Os projetos buscam estabelecer comunicação com a plataforma Arduino utilizando tecnologias e dispositivos variados. Os resultados obtidos com estas iniciativas proporcionam um leque de novas possibilidades para o laboratório e para a comunidade acadêmica.

3.2 Concepção

O desenvolvimento de aplicações Android vem crescendo cada vez mais desde a sua implantação gerida pelo Google. O grande fator que impulsionou esta grande massa de desenvolvedores foi a disponibilização do código fonte do sistema para que as empresas fabricantes de dispositivos, assim como os usuários mais avançados, pudessem personalizar seus sistemas e aplicações.

Além do código fonte disponível, existem ferramentas muito simples para o desenvolvimento em Android com a possibilidade de qualquer pessoa com um conhecimento básico em programação possa fazer alguns programas que utilizem comunicação em rede ou manipulação de objetos da tela do aparelho, por exemplo. Estes programas utilizam o recurso “arrastar e soltar”, através da criação de

diagramas de blocos no formato de peças de quebra-cabeças que representam as classes e os métodos utilizados na programação tradicional.

Um exemplo de aplicação é o AppInventor. Com ele é possível desenvolver uma aplicação online e baixar o arquivo “.apk“ para instalar em um *tablet* ou *smartphone*.

Estas facilidades fazem com que os usuários tenham maior interesse em conhecer melhor o sistema, fidelizando o uso destes com o Android. A partir do momento em que se conhece uma ferramenta como esta e há certo domínio sobre ela, é possível que um usuário antes leigo em programação torne-se em pouco tempo um bom desenvolvedor de aplicativos Android.

Juntamente com as ferramentas de criação de aplicativos, vem surgindo também muitas bibliotecas que facilitam o desenvolvimento de aplicações que exigem comunicação em rede com outros dispositivos. Uma delas é a biblioteca Amarino, que foi utilizada na segunda versão do aplicativo desenvolvido neste projeto.

Trata-se de uma biblioteca que implementa métodos de comunicação Bluetooth, gerenciando a conexão do Android com um módulo ligado ao Arduino. Para que este processo possa ocorrer com êxito, é necessário ainda que seja implementado um código específico em cada lado da comunicação, que podem funcionar como clientes, servidores ou ainda como ambos simultaneamente.

Observando esta gama de recursos, surgiu a ideia de se trabalhar com o Android para a manipulação de robôs móveis através desta plataforma. Além dos recursos lógicos disponíveis para a programação de sistemas, os *smartphones* e *tablets* que executam o Android, possuem algumas interfaces que viabilizam muitos projetos envolvendo o Arduino com a utilização de Shields e módulos de comunicação.

Alguns destes recursos foram implantados neste projeto. Primeiramente podem ser citadas as formas de comunicação utilizadas. Para que esta comunicação pudesse ser estabelecida, foi necessário avaliar as interfaces disponíveis no *smartphone* utilizado pelo autor do projeto. O modelo adotado foi o Samsung Galaxy Y - S5367, que possui interface de comunicação *Wi-Fi* e *Bluetooth* possibilitando a utilização destes recursos pelo aplicativo desenvolvido.

Outro recurso disponível no aparelho e utilizado pela aplicação é o acelerômetro. “Os acelerômetros são sensores que permitem medir forças de

aceleração, inclinação, rotação, vibração, colisão e aceleração da gravidade” (SILVA VIEIRA, 2011). A ideia para este recurso seria a possibilidade de alternar o modo de controle do aplicativo entre a forma convencional utilizando botões e esta, através da inclinação do aparelho para representar os movimentos do robô.

A grande quantidade de possibilidades para o projeto tornou a ideia mais viva para sua concepção e após algumas semanas de investigação sobre o tema já foi possível estabelecer conexões simples entre Android e Arduino. O tema deste projeto torna-se muito relevante para as áreas abrangidas pelo Android, Arduino e Robótica, levando em consideração o fato de que este pode ser parte de um projeto maior futuramente.

3.3 Diagrama de casos de uso

A interação do usuário com o aplicativo se dá pelo simples acesso ao menu de opções de comunicação, seguido do estabelecimento de uma conexão com o Arduino, completados estes dois passos, o usuário estará apto a realizar o controle através do envio de comandos do Android para o robô móvel.

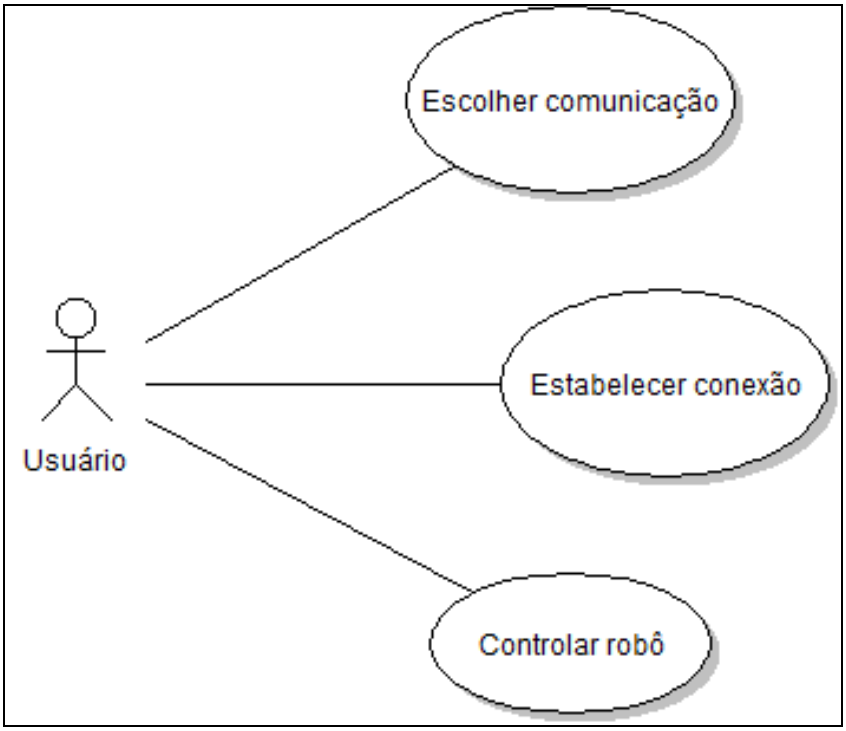


Figura 3.3.1 - Diagrama de Casos de Uso

Fonte: o Autor

Capítulo 4

IMPLEMENTAÇÃO

Neste capítulo são detalhados os processos do desenvolvimento do aplicativo Android e do software embarcado no Arduino, desde os primeiros testes de comunicação até a sua versão atual com possibilidade de múltiplas formas de conexão.

4.1 Implementação Android

4.1.1 Cenários

No desenvolvimento do aplicativo foi utilizada a linguagem de programação Java com o kit de desenvolvimento de software para Android OS. Todos os recursos de conexão são acessados através do toque na tela, com exceção dos menus, que podem ser exibidos através da tecla nativa de menu do aparelho. Os recursos de controle podem ser enviados também através do sensor Acelerômetro do aparelhos através da inclinação do mesmo para representar os movimentos que o robô deve executar.

4.1.2 Cenário do Menu

Ao abrir o aplicativo, o usuário irá se deparar com uma tela para escolha do modo de comunicação que irá utilizar para se conectar ao robô, podendo optar pela conexão via *Bluetooth* ou *Wi-Fi*. As formas de comunicação estão representadas pelo seu símbolo usual e as telas de controle em ambas são bem semelhantes, observando os requisitos necessários para cada tipo de conexão.

A **Figura 4.1** representa a tela inicial do aplicativo no emulador Android:

Figura 4.1.1 - Tela inicial do aplicativo

Fonte: o Autor

4.1.3 Cenário de Conexão Bluetooth

Para que uma conexão *Bluetooth* seja estabelecida entre Android e Arduino, é necessário que seja incluso no código o endereço físico do dispositivo que receberá a conexão, neste caso o módulo *Bluetooth* que estará conectado ao Arduino. Após a seleção do modo *Bluetooth* para conexão, irá aparecer a tela representada pela **Figura 4.2** onde o usuário poderá enviar comandos para o Arduino através do toque na tela.

Figura 4.1.2 - Modo de comunicação via *Bluetooth*

Fonte: o Autor

Com esta tela já é possível perceber que a interface está voltada para o controle direcional do robô, envolvendo os movimentos em todas as direções. Para que a conexão seja estabelecida o usuário deverá utilizar a tecla de Menu do aparelho. Será mostrado um menu com algumas opções na parte de baixo da tela conforme mostrado na **Figura 4.3**, a opção que deve ser escolhida inicialmente é “Abrir Conexão”. Ao escolher esta opção o aplicativo irá acionar o mecanismo de intenção de conexão no endereço físico pré-definido na programação, ou seja, o endereço MAC do módulo *Bluetooth*.

O módulo *Bluetooth* possui um *LED* indicativo de conexão, que pisca alternadamente quando está sem conexão ativa. Para constatar que a conexão foi estabelecida com êxito, o usuário poderá conferir na parte inferior da tela do dispositivo um texto com a palavra “Conectado!” ou ainda verificar se o *LED* do módulo Bluetooth no Arduino está com o *LED* aceso ininterruptamente.

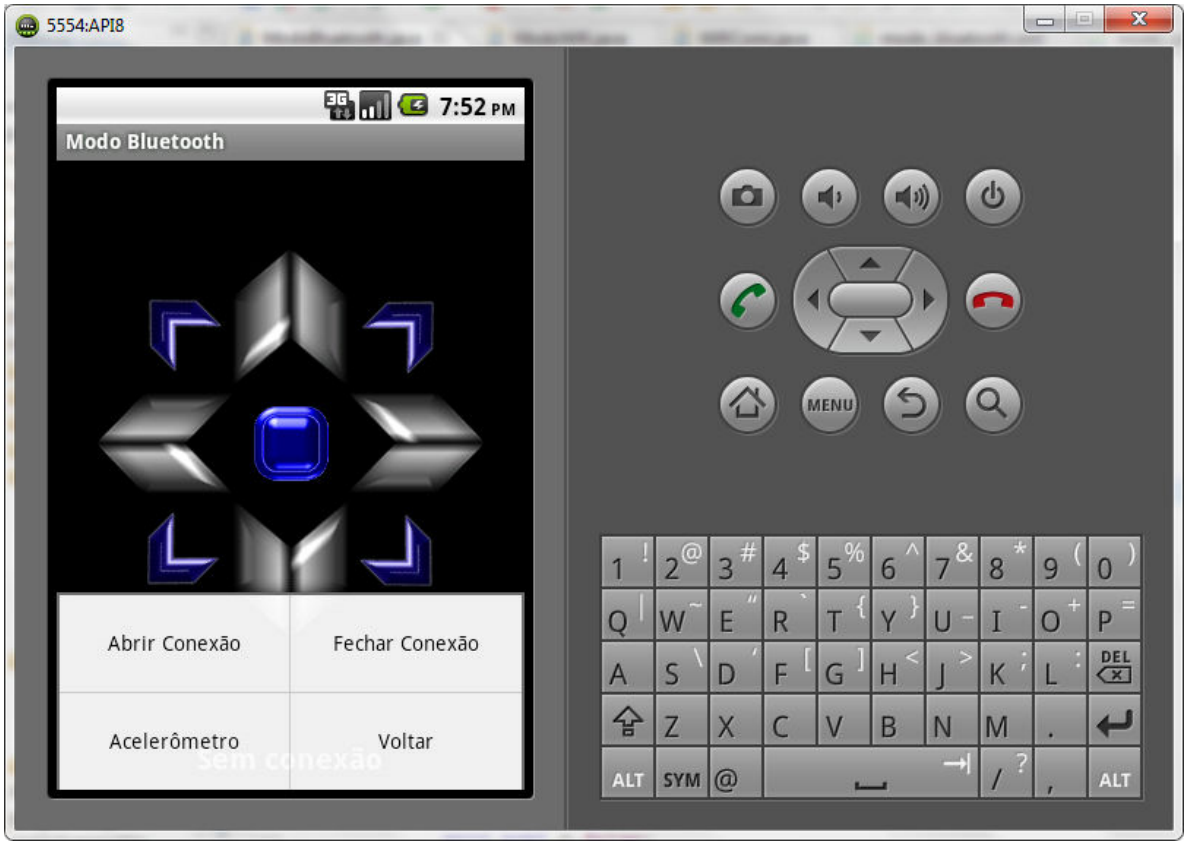


Figura 4.1.3 - Menu de opções no Modo *Bluetooth*

Fonte: o Autor

Após o devido estabelecimento da conexão, é possível enviar comandos ao Arduino através do toque nos botões de direção disponíveis na tela. Para utilizar o controle através da inclinação do aparelho, basta acionar o Menu e selecionar a opção “Acelerômetro”, fazendo isto todos os comandos que antes eram enviados pelos botões passam a ser executados de acordo com a posição do aparelho. Os botões não ficam inativos, porém o envio dos comandos emitidos pelos botões ocorrem uma vez a cada toque, já os comandos do Acelerômetro são enviados num fluxo constante, por isso sempre irão sobrepor os comandos dos botões.

Para enviar o comando que representa o “*Stop*” (botão azul central), por exemplo, o aparelho deve ficar na posição horizontal. Os dados obtidos pelo sensor são avaliados e há uma tolerância de inclinação para este comando devido à dificuldade de manter o aparelho exatamente na posição em que os eixos X e Y estejam com o valor “0”.

A **Tabela 4.1** mostra como os valores são interpretados pelo aparelho e em seguida enviados ao Arduino:

Valor do Sensor			
Eixo X	**Eixo Y**	**Estado**	**Comando Enviado**
< -4	> -4 e < 4	FRENTE	'2'
> -4 e < 4	> 4	ESQUERDA	'4'
> -4 e < 4	> -4 e < 4	PARADO	'5'
> -4 e < 4	< -4	DIREITA	'6'
> 4	> -4 e < 4	RÉ	'8'

Tabela 4.1 - Comandos enviados utilizando Acelerômetro

Fonte: o Autor

Esta forma de comandos possibilita o controle direcional do robô com sinais digitais fixos, mas podem ser transmitidos numa faixa de valores determinada de acordo o valor obtido no sensor para que assim seja possível controlar a intensidade de energia que o Arduino deve repassar a um motor de corrente contínua para controlar sua velocidade, por exemplo.

4.1.4 Cenário de Conexão Wi-Fi

No modo de comunicação via *Wi-Fi* a interface de usuário é bem semelhante à anterior, com um componente a mais no topo da tela para que o usuário possa inserir o endereço IP do servidor que irá receber o sinal. Neste caso, é dispensado o endereço físico do módulo de recepção no Arduino para que a conexão seja estabelecida. A **Figura 4.4** mostra a interface gráfica do cenário. Ao inserir o IP, basta tocar no *Toggle Button* ao lado para iniciar a conexão, é importante ressaltar que o servidor já deve estar aguardando a conexão no momento do toque no botão para que esta possa ser estabelecida com sucesso, caso contrário o aplicativo irá exibir uma mensagem com o nome do erro encontrado, neste caso será a falha na conexão pelo fato de que o servidor não estava aguardando conexões.

Quando uma conexão é estabelecida com o servidor, da mesma forma que acontece no Modo *Bluetooth*, uma mensagem informa a conexão, o diferencial

neste caso é que também é possível verificar uma linha verde no *Toggle Button*, logo abaixo do texto interno ao botão.

Após a conexão, o modo de controle funciona de forma idêntica ao do Modo *Bluetooth*. Neste modo, os comandos são enviados via Sockets pela rede. O servidor foi testado com a utilização de um computador rodando o código que deverá ser executado pelo *Shield Wi-Fi* ligado ao Arduino para que os comandos sejam encaminhados aos atuadores do robô.

Figura 4.1.4 - Modo de comunicação via *Wi-Fi*

Fonte: o Autor

Os valores enviados neste modo são os mesmos do Modo Bluetooth que estão representados na **Tabela 4.1**, tanto no modo de controle por botões quanto no modo de controle por acelerômetro. Quando o controle por acelerômetro está ativo, os comandos são enviados num fluxo constante com um intervalo de meio segundo de um para o outro, ou seja, com o aparelho na posição horizontal o comando “PARADO” estará sendo enviado ao servidor duas vezes a cada segundo.

Ao inclinar o aparelho, o sensor irá indentificar o valor das variáveis X e Y e irá atualizar o comando que deve ser enviado, substituindo o valor anterior pelo atual. Diferente do Modo *Bluetooth*, o menu do Modo *Wi-Fi* terá apenas dois itens: “Acelerômetro” e “Voltar”. As funções de Abrir Conexão e Fechar Conexão presentes no Modo *Bluetooth*, foram implementadas aqui para serem executadas de acordo com o estado do botão presente ao lado da área de endereço IP do Servidor.

Na **Figura 4.5** pode ser observada a tela do aplicativo no momento em que é pressionada a tecla de Menu do aparelho. A tecla “Voltar”, tanto no Modo *Bluetooth* quanto no Modo *Wi-Fi*, encerram a atividade atual e abrem a tela inicial do aplicativo onde constam os botões para escolha do modo de comunicação que será adotado na conexão com o Arduino, portanto não é possível sair de um modo para outro sem antes passar pela tela inicial. A **Figura 4.6** mostra um esquema do fluxo de navegação no aplicativo.

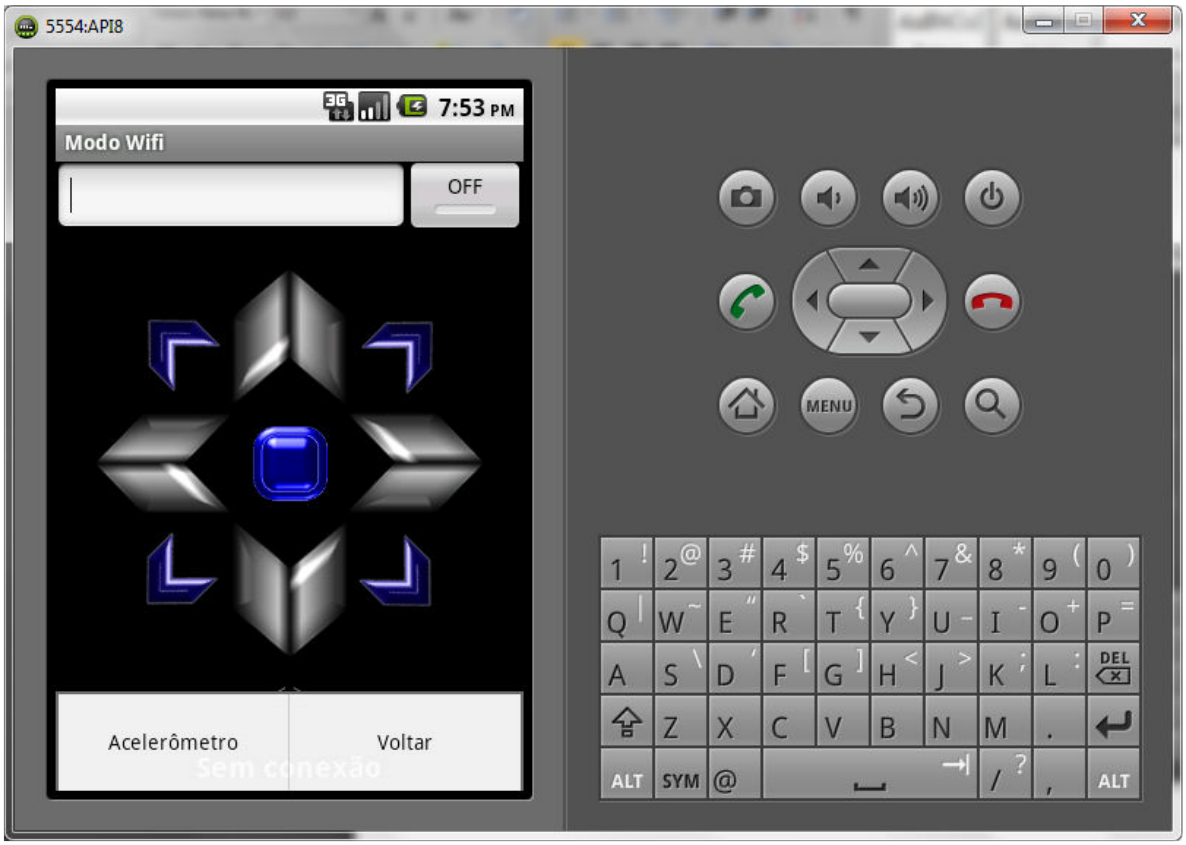


Figura 4.1.5 - Menu de opções no Modo *Wi-Fi*

Fonte: o Autor

A comunicação entre Android e Arduino implementada no aplicativo trata especificamente da finalidade de controlar um robô móvel terrestre, seja ele com rodas, pernas ou outros membros, manipulando a direção que este deve seguir de acordo com os comandos transmitidos na conexão. Porém, outros tipos de projetos em Arduino podem se utilizar deste aplicativo para estabelecer uma forma de controle, basta que os comandos sejam aplicados a determinada tarefa implementada no Arduino.

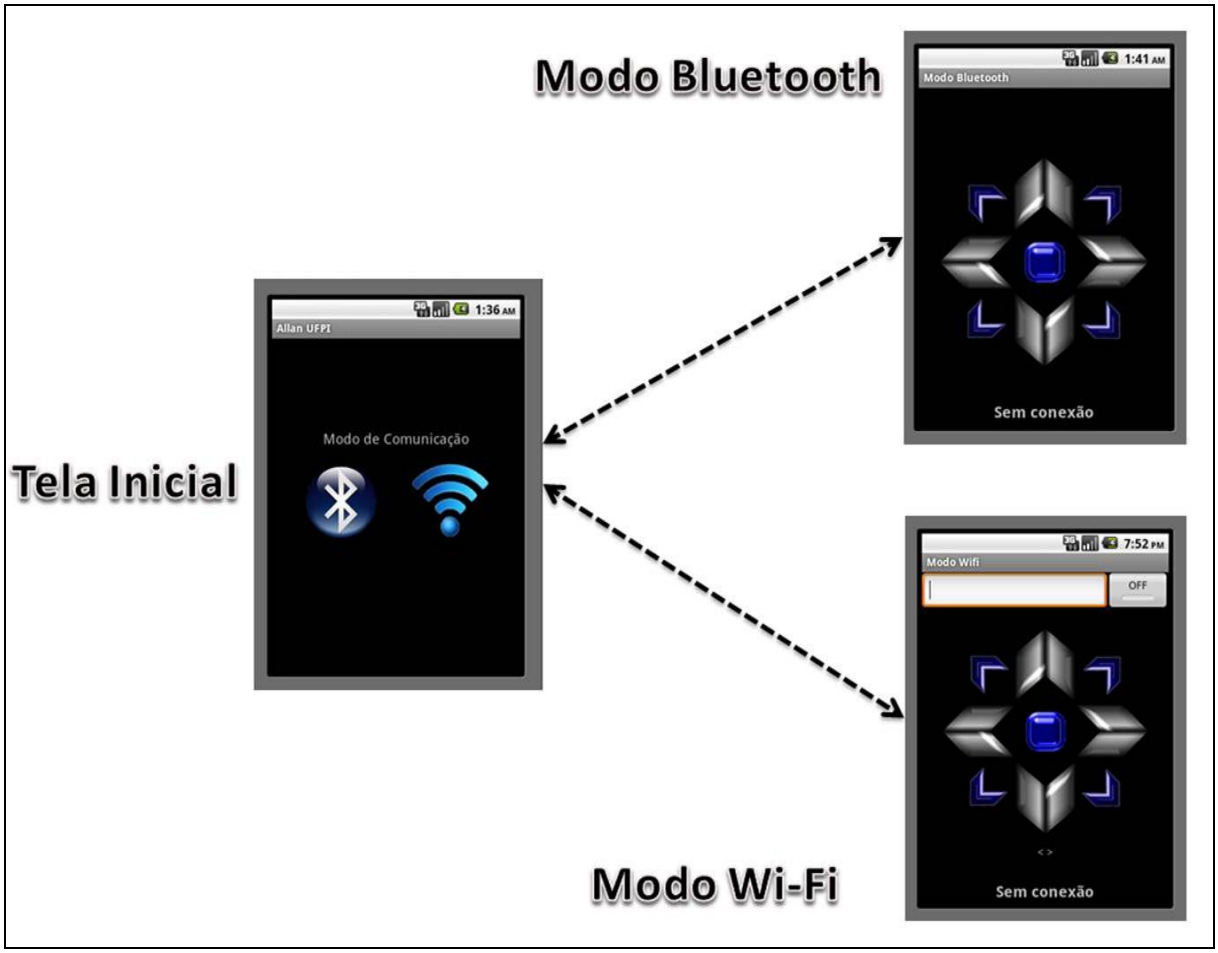


Figura 4.1.6 - Fluxo de navegação no Aplicativo

Fonte: o Autor

4.2 Implementação Arduino

O código embarcado na placa Arduino é bem mais simples e menos complexo que o código da aplicação Android. Para o Arduino fazer o gerenciamento entre comandos recebidos e ações a ser executadas basta um simples comando para recepção dos dados (entrada), com devida identificação do conteúdo, seguido

de um envio de comando (saída) através de suas portas analógicas ou digitais para os componentes do robô, no caso deste projeto, para os motores de locomoção e direção.

A comunicação via Bluetooth no Arduino tem idêntica forma de atuação de uma comunicação serial, portanto, o código que manipula transações utilizando um cabo serial entre um computador e o Arduino também é capaz de manipular transações de um módulo *Bluetooth* sem nenhuma alteração em sua estrutura. A única diferença que será encontrada nesta mudança será a utilização dos pinos RX/TX (recepção/transmissão) no Arduino quando este estiver trabalhando com a comunicação *Bluetooth*. Geralmente estes pinos encontram-se nos pinos digitais zero e um, respectivamente.

4.2.1 Recebendo dados do Android via *Bluetooth*

Para que o Arduino possa receber os dados enviados pela aplicação, basta utilizar o software Arduino e implementar a função ***setup*()** com a chamada de início da comunicação serial, da seguinte forma: ***Serial.begin*(9600);**. O valor passado no parâmetro (9600) representa a taxa de transferência da comunicação que será iniciada, neste caso, esta taxa está ajustada para o valor em que opera o módulo *Bluetooth*, ou seja, a comunicação serial foi iniciada com uma taxa de transferência de 9600 bits por segundo.

Ainda na função inicial, é necessário definir como pinos de saída, os pinos que serão responsáveis por enviar os comandos recebidos do Android para os motores do robô: ***pinMode*(9,*OUTPUT*);**. Esta função serve para definir o pino que será utilizado pela aplicação e seu modo de operação, podendo este ser de entrada ou saída. No caso apresentado, a declaração indica que o pino 9 terá o modo de operação definido como pino de saída, assim ele poderá apenas enviar energia.

Definida a função ***setup*()**, a função **loop()** poderá ser codificada. O código é bem simples e objetivo: ***if*(*Serial.available*() > 0) *char* c = *Serial.read*();**. Esta condição está testando se há alguma conexão serial disponível no momento. O método "*available*" retorna um inteiro indicando a quantidade de conexões. Após esta declaração, dentro das chaves do "***if*"** são declaras as rotinas que o Arduino deve executar. A variável **"c"** exemplificada, estará armazenando o comando que foi

recebido pela comunicação serial enquanto houver comunicação disponível, tendo seu valor atualizado a cada nova interação.

Para interagir com a aplicação Android, o código neste ponto será apenas de verificação do valor recebido seguido de alguma ação representativa do comando em questão, exemplificando: para realizar a ação de parar o robô, a aplicação envia o caractere **'5'**; a variável **"c"** irá armazenar este valor; através de uma verificação do valor contido em **"c"**, o comando pode ser executado pelo Arduino: ***if*(c == '5') PARADO();**. Neste caso, a função **"PARADO()"** está declarada e implementada em outro trecho do programa e está apenas sendo chamada. Seu conteúdo apenas envia o valor **"*LOW*"** para os dois pinos de controle de cada motor, onde estão conectados os fios de positivo e negativo, fazendo com que o motor fique parado por falta de energia para movimentá-lo.

Com este fluxo, o processo de comunicação entre Android e Arduino torna-se simples e eficiente, podendo operar uma unidade robótica com objetividade. Sintetizando, o código Arduino para controlar (parar) um motor ficará assim:

```
int pinoMotor1A = 9;
int pinoMotor1B = 10;

void setup(){
  Serial.begin(9600);
  pinMode(pinoMotor1A, OUTPUT);
  pinMode(pinoMotor1B, OUTPUT);
  }

void loop(){
  if(Serial.available() > 0){
    char c = Serial.read();
    if(c == '5') PARADO();
  }
}

void PARADO(){
  digitalWrite(pinoMotor1A, LOW);
  digitalWrite(pinoMotor1B, LOW);
}
```

Com este código embarcado na placa, basta conectar os fios de um motor CC nas portas digitais 9 e 10 do Arduino. Para fazer com que a função **"PARADO()"** gire um motor quando executada, basta que um dos parâmetros **"*LOW*"** seja

substituído por **"*HIGH*"**. Este código pode ser testado também através do recurso Serial Monitor do software Arduino, com uso do cabo USB entre o computador e a placa. Vale lembrar que o módulo Bluetooth utiliza a mesma forma de comunicação do cabo serial, portanto se o cabo e o módulo estiverem conectados ao Arduino simultaneamente, não será possível fazer o *upload* do código para a placa através do cabo serial, como também não será possível enviar comandos através do Serial Monitor. Caso ambos estejam conectados, uma solução rápida é a interrupção da energia do módulo desconectando o seu fio positivo. No **Apêndice B** consta o esquema de ligação do robô em *protoboard*.

Capítulo 5

TESTES REALIZADOS

Os primeiros testes de comunicação deste projeto foram realizados com o objetivo de acender um *LED* através da aplicação Android. O código utilizado no Arduino foi implementado para que, ao receber um determinado sinal através do módulo *Bluetooth*, a placa enviasse energia para que o *LED* fosse aceso.

Após algumas semanas de pesquisa o objetivo inicial foi alcançado, possibilitando assim uma nova fase de desenvolvimento. Esta nova fase trouxe como desafio a implementação da aplicação em Android, com uma interface amigável e funcional.

5.1 Interface gráfica

A primeira versão do aplicativo tinha um objetivo apenas funcional, buscando apenas estabelecer um padrão de comunicação com a definição de botões específicos para cada comando que seria enviado para a placa de hardware do robô. Os botões foram inseridos na área correspondente à tela do *smartphone* sem considerar uma melhor disposição de *layout* para o usuário.

Nesta etapa, surgiram alguns problemas de estabilidade do aplicativo durante a navegação entre as telas de controle e menus. Outro problema detectado foi a perda de comunicação utilizando o acelerômetro devido à inatividade de recursos visuais, ou seja, após certo tempo controlando o robô com o acelerômetro, a tela se apagava e a comunicação era perdida. Algumas pesquisas sobre os eventos ocorridos revelaram a necessidade de utilização de funções nativas do Android para a manutenção da usabilidade da aplicação.

Estabelecidos os comandos e botões correspondentes, a parte funcional já em funcionamento passou a ser migrada para uma interface amigável e ao mesmo tempo confortável para que o usuário tivesse a liberdade de utilizar os controles sem a necessidade de conferir a tela do aplicativo a cada novo botão que desejasse utilizar.

Após o estabelecimento de um padrão visual para o aplicativo, foi desenvolvida a interface em forma de setas direcionais para representar os movimentos de locomoção do robô.

Concluída a aplicação Android e o código Arduino, foi montado um robô móvel utilizando uma base pronta, apenas conectando os motores aos pinos do Arduino com o intermédio de um chip L293D (ponte H) para realizar a movimentação dos dois motores em ambos os sentidos.

Com todos os componentes em funcionamento, foram realizados vários testes de comunicação e controle, ajustando a aplicação Android e o código Arduino conforme eram percebidas as deficiências do projeto. Em sua versão final, o aplicativo tem uma conexão estável com o Arduino.

5.2 Teste de comunicação *Bluetooth*

Nesta versão, um teste bastante surpreendente foi o teste de alcance do controle via *Bluetooth*, que demonstrou que esta forma de comunicação pode ser utilizada em maior escala para este tipo de atuação. O módulo *Bluetooth* utilizado no Arduino neste projeto é um dispositivo Classe um, modelo AUBTM-22 adquirido no site da Tato (www.tato.ind.br), que possui alcance de até cem metros.

O teste foi realizado em ambiente aberto na Universidade Federal do Piauí, Campus Senador Helvídio Nunes de Barros em Picos. O trajeto do robô foi partir do local onde estava o controle afastando-se até o limite onde foi possível trafegar com sua estrutura. O alcance constatado neste teste foi de aproximadamente 50 metros, onde foi enviado o comando para o robô fazer a curva e retornar. O mesmo encerrou seu trajeto no mesmo local de onde partiu, com o usuário portador do controle no mesmo local em que a conexão foi iniciada.

5.3 Teste de comunicação *Wi-Fi*

Utilizando o modo Wi-Fi, é possível estabelecer a comunicação de duas formas: *ad-hoc*, conectando o Android diretamente ao shield Wi-Fi, ou infraestrutura, utilizando algum ponto de acesso (roteador) como mediador da conexão. O teste realizado foi utilizado utilizando o próprio roteador do laboratório, que faz a distribuição de sinal de internet sem fio para os usuários do mesmo.

O teste de comunicação Wi-Fi foi realizado no próprio laboratório de desenvolvimento. Devido à falta do shield necessário para a comunicação neste modo, foi implementado um código de um servidor *socket* em Java para simular a comunicação entre Android e Arduino com esta tecnologia. O código foi executado no Eclipse IDE rodando em um *notebook* com sistema operacional Windows.

O código tinha a função apenas de receber o comando emitido pelo aplicativo Android e mostrar este comando na tela do computador para que fosse constatado o recebimento do mesmo. O teste foi satisfatório e revelou grande eficiência do aplicativo com esta forma de comunicação, tanto na utilização de botões quanto no controle por acelerômetro.

Capítulo 6

CONSIDERAÇÕES FINAIS

Neste trabalho foram abordados temas relevantes da tecnologia alcançada nos últimos anos. A grande massa de dispositivos Android possibilita a integração dos usuários com aplicativos capazes de padronizar as formas de comunicação e compartilhamento de dados na rede mundial de computadores. O aplicativo construído neste trabalho é de grande importância para o estudo da comunicação entre o Android e o Arduino.

A sofisticação da robótica cresce cada vez mais e a plataforma Arduino vem destacando-se como uma das melhores ferramentas de prototipação na área. A facilidade de produção proporciona rápido avanço no conhecimento para principiantes neste universo tão diversificado da tecnologia.

Algumas dificuldades foram enfrentadas no início do projeto especificamente no que tange ao desenvolvimento da aplicação em Android, como os erros de execução no momento de navegação entre as telas do aplicativo, erros de conexão constantes, entre outras dificuldades que impediram um fluxo constante e crescente de evolução do projeto, porém, todos estes problemas foram sanados e a aplicação mostra-se confiável quanto ao seu objetivo.

O desafio de desenvolver um aplicativo que possibilita duas formas de comunicação distintas com a plataforma Arduino foi superado com este trabalho. Sua estrutura mostra-se alinhada aos objetivos do projeto com grande possibilidade de expansão futura. A implementação foi realizada de forma gradativa e modularizada, sendo testada após a implantação de cada novo componente no projeto, tornando sua credibilidade ainda maior.

Os testes de estabilidade da conexão e alcance da comunicação comprovam que o aplicativo pode ser utilizado com segurança em novos projetos da área. O tempo de resposta do robô ao comando enviado pelo *smartphone* é quase imediato. A dirigibilidade através do acelerômetro pode ser melhorada de acordo com a prática do usuário, o aperfeiçoamento poderá ser mais lento quando este não está habituado com este tipo de recurso.

Como trabalhos futuros supõe-se a implementação de um banco de dados para a aplicação para armazenar:

- Dados das conexões para estatísticas de estabilidade;
- Configurações personalizadas do usuário.

Supõe-se ainda a criação de um menu de configurações para que o usuário possa:

- Inserir, mover ou excluir botões da interface gráfica, possibilitando assim uma personalização do aplicativo para outras finalidades além do controle de robôs móveis;
- Configurar os comandos que cada botão deve enviar;
- Configurar a frequência de envio dos comandos;
- Configurar o envio de faixas de valores de acordo com o valor do sensor Acelerômetro para trabalhar com os pinos analógicos do Arduino (PWM).

REFERÊNCIAS

ABLESON, Frank. Publicação de artigos científicos. **Introdução ao desenvolvimento do Android**, mar. 2013. Disponível em: <http://www.ibm.com/developerworks/br/library/os-android-devel/>. Acesso em: 27 mar. 2013

ANDROID, Disponível em: < http://developer.android.com/guide/basics/what-is-android.html >. Acesso em: mar. 2013

ARDUINO, Disponível em: <http://arduino.cc/>. Acesso em: 29 mar. 2013

ATTROT, Wesley**. Aplicações da Robótica no Ensino de Ciência da Computação. Um Estudo de Caso.** Monografia (Bacharelado em Ciência da Computação) - Universidade Estadual de Londrina, Londrina, 2002. In: **ZANELATTO**, Marcelo Stuani. **Robótica Educacional nos Cursos de Ciência da Computação,** 2004**.** Disponível em: <http://www2.dc.uel.br/nourau/document/?view=81>. Acesso em 29 mar. 2013

CANO, Carlos E. V. **Técnica de navegação de um robô móvel baseado em um sistema de visão para integrá-lo a uma célula de manufatura**. Disponível em: <http://vsites.unb.br/ft/enm/sistmec/site/admin/publicacoes/villa.pdf>. Acesso em: 12 mai. 2012

FERNANDES, Anita Maria da Rocha; **LAURINDO**, Rafael Daniel. Publicação de artigos científicos**. Sistema de Controle Baseado em Telefonia Celular**. Disponível em: < http://www.aedb.br/seget/artigos11/13214145.pdf>. Acesso em: 27 mar. 2013

GARGENTA, Marko. Learning Android. Sebastopol: O'Reilly Media, 2011. In: **SILVA**, Luciano Édipo Pereira da. **Utilização da plataforma Android no desenvolvimento de um aplicativo para o cálculo do Balanço Hídrico Climatológico**, 2009. Disponível em: <http://bhcmovel.googlecode.com/files/TCC%20-%20Final.pdf>. Acesso em 15 mar. 2013

GUERRA, S.C.S. **Sistema de Navegação para Veículo Autônomo Utilizando Lógica Difusa**. 128 p. Dissertação (Mestrado). Escola de Engenharia de São Carlos, Universidade de São Paulo, São Carlos, 2009. In: **VIEIRA**, Manoel Alexandre. **Sistema de telemetria para robôs móveis**, 2011. Trabalho de Conclusão de Curso. Universidade Federal do Vale do São Francisco, Juazeiro, 2011. Disponível em: <http://www.univasf.edu.br/~ccomp/monografias/monografia_3.pdf>. Acesso em 22 mar. 2013

LECHETA, R. R. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK**. São Paulo: Novatec, 2009. In: **SILVA**, Luciano Édipo Pereira da. **Utilização da plataforma Android no desenvolvimento de um aplicativo para o cálculo do Balanço Hídrico Climatológico**, 2009. Disponível em: <http://bhcmovel.googlecode.com/files/TCC%20-%20Final.pdf>. Acesso em 15 mar. 2013

MARTINS, AGENOR. **O Que é Robótica**. São Paulo: Editora Brasiliense, 1993. In: **ZANELATTO**, Marcelo Stuani. **Robótica Educacional nos Cursos de Ciência da Computação,** 2004**.** Disponível em: <http://www2.dc.uel.br/nourau/document/?view=81>. Acesso em 29 mar. 2013

SILVA VIEIRA, JOSE CARLOS. **Plataforma Móvel Aérea QuadRotor**. Escola de Engenharia, Universidade do Minho. Tese de mestrado.

STEELE, J.; **TO**, N. **The Android Developer's Cookbook Building Applications with the Android SDK**. Boston: Pearson Education, 2011. In: **SILVA**, Luciano Édipo Pereira da. **Utilização da plataforma Android no desenvolvimento de um aplicativo para o cálculo do Balanço Hídrico Climatológico**, 2009. Disponível em: <http://bhcmovel.googlecode.com/files/TCC%20-%20Final.pdf>. Acesso em 15 mar. 2013

APÊNDICES

Apêndice A – Algoritmo Android - Comunicação Bluetooth

```
BluetoothAdapter meuAdaptador;
BluetoothSocket socket;
BluetoothDevice dispositivo;
OutputStream saida;
InputStream entrada;
Thread xthread;
byte[] conteudo;
int posConteudo;
int cont;
volatile boolean trabParado;
public boolean btConectado = false;
public boolean ativ_acel = false;
private SensorManager sensorMgr;
public float eixoX, eixoY;
private long lastUpdate = -1;

public void parear() {//Parear Android com módulo Arduino
     meuAdaptador = BluetoothAdapter.getDefaultAdapter();
     if (!meuAdaptador.isEnabled()) {
          Intent enableBluetooth = new Intent(
                    BluetoothAdapter.ACTION_REQUEST_ENABLE);
          startActivityForResult(enableBluetooth, 0);
     }
     if (meuAdaptador == null) {
          txInforme.setText("Adaptador bluetooth
indisponivel.");
     }
          Set<BluetoothDevice> pareados =
meuAdaptador.getBondedDevices();
     if (pareados.size() > 0) {
          for (BluetoothDevice temp : pareados) {
               if (temp.getName().equals("AUBTM-22")) {
                    dispositivo = temp;
                    break;
               }
          }
     }
     txInforme.setText("Dispositivo não encontrado!");
}

public void infoStatus() {//Manipulador de mensagens
     final Handler handler = new Handler();
     final byte delimiter = 127;
     trabParado = false;
     posConteudo = 0;
     conteudo = new byte[1024];
```

```
     xthread = new Thread(new Runnable() {
     public void run() {
          while (!Thread.currentThread().isInterrupted() &&
!trabParado) {
               try {
                    int bytesAvailable = entrada.available();
                    if (bytesAvailable > 0) {
                         byte[] packetBytes = new
byte[bytesAvailable];
                         entrada.read(packetBytes);
                              for (int i = 0; i <
bytesAvailable; i++) {
                                   byte b = packetBytes[i];
                                   if (b == delimiter) {
                                        byte[] encodedBytes =
new byte[posConteudo];

     System.arraycopy(conteudo, 0, encodedBytes,0,
encodedBytes.length);
                                        final String data = new
String( encodedBytes, "US-ASCII");
                                        posConteudo = 0;
                                        handler.post(new
Runnable() {
                                             public void run()
{

     txInforme.setText(data);
                                             }
                                        });
                                   } else {
                                        conteudo[posConteudo++]
= b;
                                   }
                              }
                         }
                    } catch (IOException ex) {
                         trabParado = true;
                    }
               }
          }
     });
          xthread.start();
}

public void frente() throws IOException {//Comando “Frente”
     saida.write('2');//Enviando o caractere ‘2’ ao Arduino
}

public void esquerda() throws IOException {
     saida.write('4'); );
```

```
}

public void parado() throws IOException {
     saida.write('5');
}

public void direita() throws IOException {
          saida.write('6');
}

public void re() throws IOException {
          saida.write('8');
          txInforme.setText("Ré");
}

public void abrirConexao() throws IOException {//Conectar
     UUID uuid = UUID.fromString("00001101-0000-1000-8000-
00805f9b34fb");
     socket =
dispositivo.createRfcommSocketToServiceRecord(uuid);
     socket.connect();
     saida = socket.getOutputStream();
     entrada = socket.getInputStream();
          infoStatus();
     btConectado = true;
     }
     txInforme.setText("Conexão Aberta");
}

public void fecharConexao() throws IOException {//Desconectar
          ativ_acel = false;
          btConectado = false;
          trabParado = true;
          saida.close();
          entrada.close();
          socket.close();
          txInforme.setText("Conexão Encerrada!");
}
```

Apêndice B - Algoritmo Android – Comunicação Wi-Fi

```
String server;
Socket socket;
DataOutputStream os;
boolean ativ_acel, conectado = false;
private SensorManager sensorMgr;
private long lastUpdate = -1;

public void wfrente() {
     try {
          servidor = (EditText) findViewById(R.id.ipservidor);
          server = servidor.getText().toString();
          socket = new Socket(server, 7776);
          os = new DataOutputStream(socket.getOutputStream());
          os.writeUTF("2");//Enviando comando ao Arduino
          os.flush();
          os.close();
          socket.close();
     } catch (Exception e) {
          Toast.makeText(ModoWifi.this, "Erro :" +
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
}
}

public void wesquerda() {
     try {
          servidor = (EditText) findViewById(R.id.ipservidor);
          server = servidor.getText().toString();
          socket = new Socket(server, 7776);
          os = new DataOutputStream(socket.getOutputStream());
          os.writeUTF("4");
          os.flush();
          os.close();
          socket.close();
     } catch (Exception e) {
          Toast.makeText(ModoWifi.this, "Erro :" +
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
     }
}

public void wparado() {
     try {
          servidor = (EditText) findViewById(R.id.ipservidor);
          server = servidor.getText().toString();
          socket = new Socket(server, 7776);
          os = new DataOutputStream(socket.getOutputStream());
```

```
            os.writeUTF("5");
            os.flush();
            os.close();
            socket.close();
        } catch (Exception e) {
            Toast.makeText(ModoWifi.this, "Erro :" +
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
        }
}

public void wdireita() {
        try {
            servidor = (EditText) findViewById(R.id.ipservidor);
            server = servidor.getText().toString();
            socket = new Socket(server, 7776);
            os = new DataOutputStream(socket.getOutputStream());
            os.writeUTF("6");
            os.flush();
            os.close();
            socket.close();
        } catch (Exception e) {
            Toast.makeText(ModoWifi.this, "Erro :" +
e.getLocalizedMessage(),Toast.LENGTH_SHORT).show();
        }
}

public void wre() {
        try {
            servidor = (EditText) findViewById(R.id.ipservidor);
            server = servidor.getText().toString();
            socket = new Socket(server, 7776);
            os = new DataOutputStream(socket.getOutputStream());
            os.writeUTF("8");
            os.flush();
            os.close();
            socket.close();
        } catch (Exception e) {
            Toast.makeText(ModoWifi.this, "Erro :" +
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
        }
}

public void abrirConexao() {//Conectar
        try {
            servidor = (EditText) findViewById(R.id.ipservidor);
            server = servidor.getText().toString();
            socket = new Socket(server, 7776);//Servidor,porta
            os = new DataOutputStream(socket.getOutputStream());
        } catch (Exception e) {
            conectado = false;
```

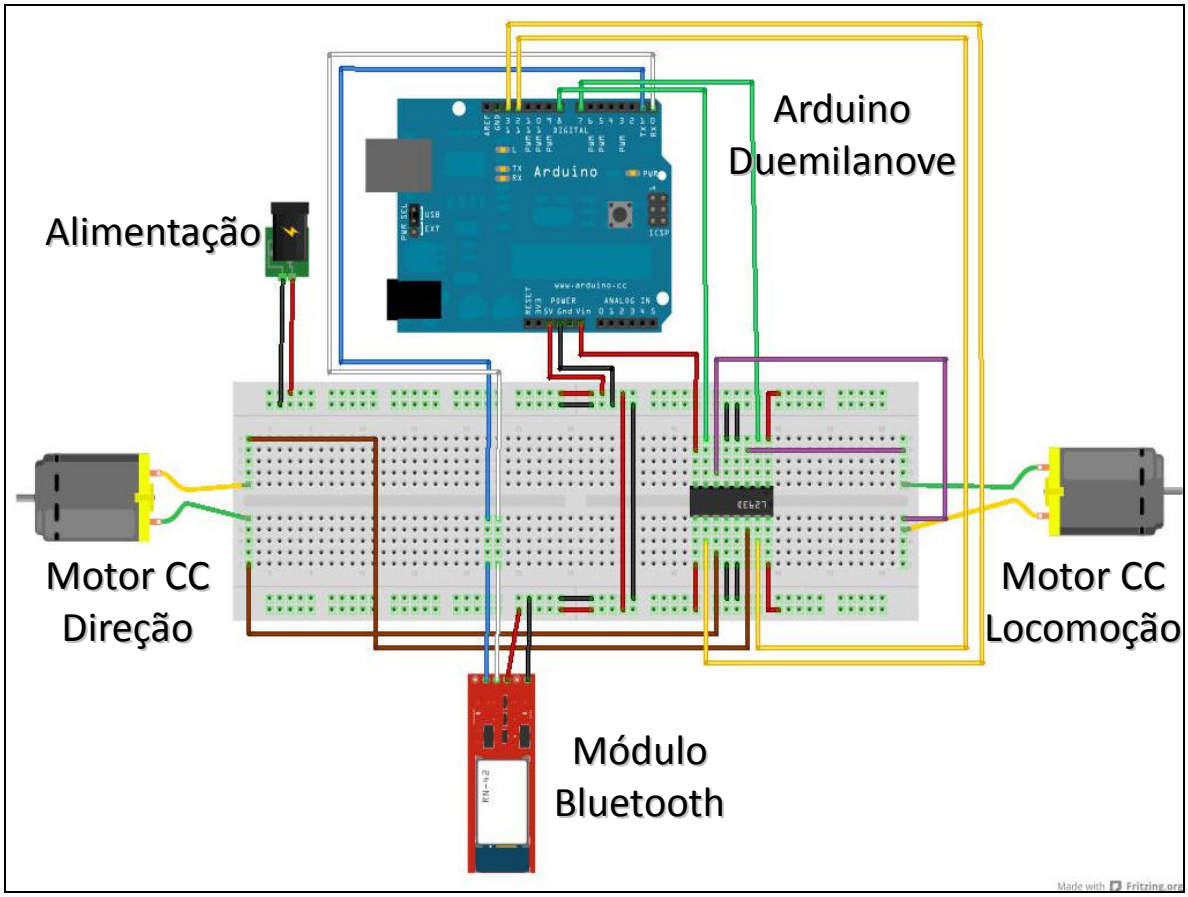
```
          Toast.makeText(ModoWifi.this, "Erro :" +
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
     }
     conectado = true;
}

public void fecharConexao() {//Desconectar
     try {
          os.flush();
          os.close();
          socket.close();
          conectado = false;
     } catch (Exception e) {
          Toast.makeText(ModoWifi.this, "Erro :" +
e.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
     }
}
```

Apêndice C – Código Arduino

```
int m1f=7, m1t=8;//Pinos motor 1
int m2f=12, m2t=13;//Pinos motor 2

void setup(){
  pinMode(m1f,OUTPUT);//Definindo como pinos de saída
  pinMode(m1t,OUTPUT);
  pinMode(m2f,OUTPUT);
  pinMode(m2t,OUTPUT);
  Serial.begin(9600);//Iniciando comunicação Serial
}

void loop(){
  char c = Serial.read();//Caracteres recebidos
  if(Serial.available() > 0){
     if(c == '2')frente();
     if(c == '4')esquerda();
     if(c == '5')parado();
     if(c == '6')direita();
     if(c == '8')re();
  }
}

void alinhar(){
  digitalWrite(m2f, 0);//Enviando impulsos elétricos
  digitalWrite(m2t, 0);//pelas portas definidas
}

void frente(){
  digitalWrite(m1f, 1);
  digitalWrite(m1t, 0);
  alinhar();
}

void esquerda(){
  digitalWrite(m2f, 1);
  digitalWrite(m2t, 0);
}

void parado(){
  digitalWrite(m1f, 0);
  digitalWrite(m1t, 0);
  digitalWrite(m2f, 0);
  digitalWrite(m2t, 0);
}

void direita(){
  digitalWrite(m2f, 0);
```

```
  digitalWrite(m2t, 1);
}

void re(){
  digitalWrite(m1t, 1);
  digitalWrite(m1f, 0);
}
```

Apêndice D – Esquema em protoboard



Apêndice E – Circuito do Robô Móvel