Desenvolvimento de uma aplicação embarcada em celular visando controle de robô via Wi-Fi

Bruno Henrique Andrade Cruz[1]
Josué Fernandes Dall Agnese[1]
Bruno José Fagundes[1]
Marcelo Teixeira Bastos[1]
Rolf Fred Molz[1]
Jacques Nelson Corleta Schreiber[1]

Resumo: Este artigo apresenta um sistema que permite o controle de um robô a partir de um celular, usando conexão Wi-Fi e sensores de movimento. O sistema, que foi construído para a plataforma Google Android, é capaz de captar as informações dos movimentos executados sobre o celular, mapeá-los e gerar comandos de movimentação para o robô, por meio de seu protocolo. Também é possível a recepção de dados coletados pelo robô.

Palavras-chave: Android. Wi-Fi. Wifibot.

Abstract: *This paper presents a system that allows you to control a robot from a phone using Wi-Fi and motion sensors. The system, which was built for the Google Android platform, is able to capture the information of the movements performed on the cell phone, map them, and generate commands to drive the robot, from its protocol. It is also possible to receive data collected by the robot.*

Keywords: *Android. Wi-Fi. Wifibot Lab.*

1 Introdução

A crescente evolução dos dispositivos móveis, impulsionada pelo modo como os fabricantes vêm trazendo aparelhos cada vez mais completos, tem aberto um novo mercado de aplicações para celular. A presença de GPS (Sistema de Posicionamento Global), conexão Wi-Fi, acelerômetros (sensores de movimento) e aumento na capacidade de processamento fazem com que esses aparelhos estejam aptos a portar aplicações mais robustas e prover novas funcionalidades.

Com esse aumento da capacidade dos celulares, tornou-se viável a construção de um projeto para o desenvolvimento de uma aplicação embarcada que permite o controle de um robô via Wi-Fi.

Com a nova geração de celulares, alguns dispositivos foram considerados como alternativa para uso na aplicação. Aparelhos como Apple Iphone, Nokia N95, Nokia N85, Nokia N97 e Samsung GT I7500 foram estudados. No entanto, para o desenvolvimento deste projeto, o aparelho Openmoko Neo Freerunner foi utilizado por ter como principal característica o "hardware aberto" e preencher os requisitos para viabilização da aplicação. Dentre esses requisitos, pode-se citar a conexão Wi-Fi, sensores de movimento (acelerômetros) e capacidade de portar um sistema operacional a escolha do usuário.

[1] Curso de Ciência da Computação, Universidade de Santa Cruz do Sul - UNISC , Av. Independência, 2293, Bairro Universitário - Santa Cruz do Sul (RS) - Brasil
mineirobruno@yahoo.com.br, {josofd, bj.fagundes}@gmail.com, wattz@hotmail.com, {rolf, jacques}@unisc.br

doi: 10.5335/rbca.2011.005

Aparelhos mais robustos requerem sistemas operacionais mais complexos e com capacidade de gerenciar os novos componentes do seu hardware. Assim, alguns sistemas operacionais surgiram para sanar este ponto, dentre os quais se podem citar: Symbian OS, Iphone OS, Google Android e Windows Mobile.

No entanto, o sistema operacional Google Android também se mostrou o mais indicado para ser trabalhado, pelo fato de que o sistema foi construído visando prover maior flexibilidade ao programador e seu código-fonte é de livre acesso para a comunidade de desenvolvedores. Também é necessário frisar a possibilidade de instalar tal sistema em aparelhos que não o portam nativamente, aumentando a relação dos dispositivos aptos a receber o aplicativo.

A partir desse fato, em que os dispositivos acima citados preencheram os requisitos técnicos, pode ser construído um sistema que provê a comunicação de um celular modelo Openmoko Neo Freerunner, portando o sistema operacional Google Android, com um robô modelo Wifibot Lab visando ao controle do mesmo. O sistema tornou-se distinto dos demais projetos por permitir conexão via Wi-Fi, diferentemente dos outros que usam conexão Bluetooth e RF.

Esse sistema fornece a capacidade de controle do robô, independentemente da distância em que o dispositivo móvel se encontra do robô. Tal aplicação pode ser usada em casos em que é necessária a execução de tarefas em ambientes hostis. Assim, o robô pode ser enviado para tal tarefa, removendo qualquer perigo ao usuário. Tarefas de vigilância também são possibilidades fornecidas pelo sistema, pois o robô usado no projeto está munido de uma câmera de vídeo que pode ser acessada via IP.

As próximas seções deste artigo visam expor informações tanto sobre os componentes do projeto como sobre seu desenvolvimento. Na seção 2 serão expostas as informações sobre o robô Wifibot Lab; a seção 3 é responsável por descrever sucintamente o sistema operacional Google Android, assim como as particularidades sobre seu desenvolvimento; a seção 4 aborda o ambiente construído, explicando suas funcionalidades e detalhes técnicos; a seção 5 descreve as conclusões obtidas após o término do trabalho.

2 Wifibot Lab

O robô Wifibot Lab foi desenvolvido pela empresa francesa Wifibot e pode ser encontrado em <http://www.wifibot.com/page5.php>, com o intuito de ajudar em projetos para novos sistemas [1]. A ideia inicial, segundo o fabricante, é de que o robô auxilie no aprendizado sobre robótica e viabilize protótipos de novos sistemas.

O modelo Wifibot Lab mostrou-se a escolha mais coerente, pelo fato de que o robô apresenta seu protocolo de comunicação divulgado para seus consumidores e atende a todos os requisitos do projeto.

Na Figura 1 apresenta-se a estrutura básica do robô. O Wifibot Lab é composto por quatro rodas motrizes, câmera VGA (Video Graphics Array), sensores infravermelhos, motor, bateria, placa controladora e *CPU*. Visando à atualização e integração com outros dispositivos, o Wifibot Lab permite que alterações sejam feitas no seu hardware, onde dispositivos podem ser substituídos ou adicionados ao kit.

Fonte: Manual Wifibot LAB.

Figura 1. Robô Wifibot Lab

O robô porta, como padrão, um processador Intel Atom N270 1.6Ghz. No entanto, esse processador está passível de substituição, como, por exemplo, por um Intel Core 2 Duo. Outras customizações, como acréscimo da memória RAM, inclusão de sensores ultrassônicos e de um braço robótico, também são possíveis.

A alimentação do Wifibot Lab é fornecida por uma bateria íon de Lítio de 7,2V.

2.1. Comunicação

A conexão de rede com o robô pode ser estabelecida de duas formas: Infraestrutura ou Ad-Hoc.

No ambiente Infraestrutura, o robô e o dispositivo móvel se conectam a um ou mais pontos de acesso Wi-Fi, e esses agem como intermediadores de mensagens entre os dispositivos conectados a eles. A vantagem principal desse modo de conexão é a possibilidade de permitir que o usuário se afaste do robô por uma distância indeterminada, já que a ligação entre os roteadores pode ser feita até mesmo pela internet.

Assim, é possível criar uma situação onde o usuário que porta um dispositivo móvel e se encontra em um local que fornece conexão com a internet poderá controlar o seu robô independentemente de onde o mesmo se encontra.

A segunda forma de conexão refere-se ao ambiente Ad-Hoc, onde o dispositivo móvel se conecta diretamente ao robô via Wi-Fi. Neste ambiente, tem-se maior simplicidade de conexão, removendo a necessidade do uso de pontos de acesso Wi-Fi. No entanto, a remoção dos pontos de acesso traz consigo a limitação de distância entre dispositivo móvel e robô.

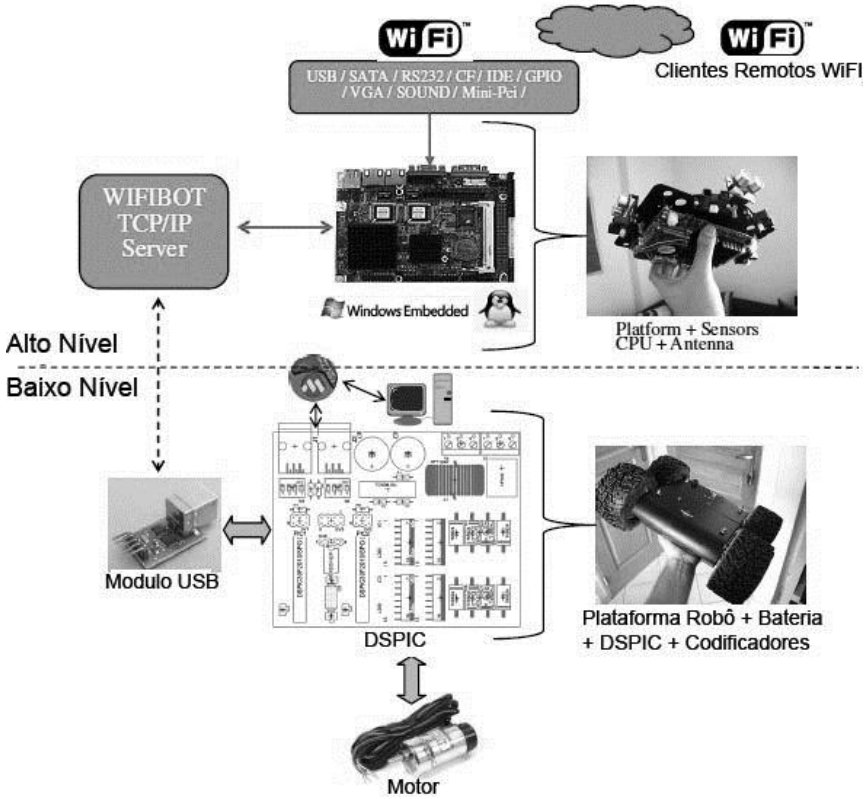
O protocolo de comunicação entre o celular e o dispositivo é um padrão de formato de mensagens, estabelecido para que o robô entenda as mensagens recebidas do dispositivo e vice-versa. Este protocolo é definido pelo robô, e a aplicação foi desenvolvida respeitando suas especificações. Mais detalhes sobre este protocolo serão abordados na seção 4, onde é relatado o processo de comunicação entre o software e o robô.

2.2. Arquitetura do sistema Wifibot Lab

A arquitetura do Wifibot Lab apresentada na Figura 2 é composta por duas partes: alto nível e baixo nível.

A parte de alto nível é composta pelos sensores integrados ao aparelho, CPU e qualquer outro dispositivo que tenha sido integrado pelo usuário ao robô, como sensor ultrassônico e braço robótico. O usuário tem total acesso a este nível via software embarcado, ou considerar o robô como um periférico de rede e enviar os comandos remotamente a ele.

A parte de baixo nível é composta por um controlador ICD2, que é capaz de controlar a placa de motor. Este nível não é acessível ao usuário, a menos que o mesmo tenha um depurador ICD2 para que possa se conectar ao microcontrolador. Essa troca de dados se deve por meio de frames RS232.



Fonte: Manual Wifibot Lab.

Figura 2. Arquitetura do sistema Wifibot

3 Google Android OS

Estudos mostram que hoje em dia mais de três bilhões de pessoas possuem um celular, e isso é quase metade da população mundial. [2] Tais dados demonstram que o mercado de celulares tem se tornado um forte nicho a ser trabalhado, o que tem gerado uma corrida entre as empresas que atuam no ramo de dispositivos móveis.

Em meio a essa disputa pelo mercado de celulares, surgiu a OHA (Open Hanset Alliance), que é uma aliança entre as maiores empresas do ramo de telefonia, liderada pela Google. Dentre seus integrantes podem-se citar: HTC, LG, Motorola, Samsung, Sony Ericsson, Toshiba, Sprint Nextel, China Mobile, T-Mobile, Asus, Intel, Garmin e outras mais. [3]

Do fato de que tais fabricantes disputam pelo mercado da venda de celulares, e não sistemas operacionais, surgiu a intenção de criar uma plataforma de desenvolvimento de código aberto, que seja única e flexível. A partir deste ponto, foi criado o projeto Android.

O Android é uma plataforma de desenvolvimento para dispositivos móveis, baseada no sistema operacional Linux e que vem aquecendo o mercado de telefonia. Tal plataforma visa fornecer um sistema único, que poderá ser instalado nos dispositivos fornecidos por qualquer um dos fabricantes pertencentes ao consórcio.

A arquitetura do sistema Android implementa os conceitos de integração e flexibilidade. Isso quer dizer que os aplicativos fornecidos nativamente com o sistema podem ser substituídos por aplicações customizadas, ou mesmo interagir com outras aplicações que não são nativas do sistema. Este fato se torna muito interessante para os fabricantes, pois, levando em conta que o código-fonte é de livre acesso à comunidade, pode-se acessar este código e inserir alguma forma de marketing na interface com o usuário.

Com relação ao desenvolvimento de aplicações, o Android vem angariando muitos simpatizantes dentre os desenvolvedores para dispositivos móveis, em razão de muitos fatores favoráveis à plataforma. Talvez o maior dos fatores seja o fato de que a Google vem lançando concursos de aplicativos para plataforma Android. Esses concursos oferecem prêmios atrativos para desenvolvedores do mundo inteiro, considerando que no Android Developer Challenger, que foi o maior concurso na área, o investimento da Google foi U$ 10 milhões em prêmios, e as 20 melhores aplicações receberam U$ 275.000,00.

No entanto, o grande benefício desses concursos para a comunidade de usuários foi a imensa quantidade de aplicações construídas durante o processo de avaliação e o fato de que esses códigos-fonte, em sua maioria, foram marcados como código livre por seus criadores.

3.1. Máquina virtual Dalvik

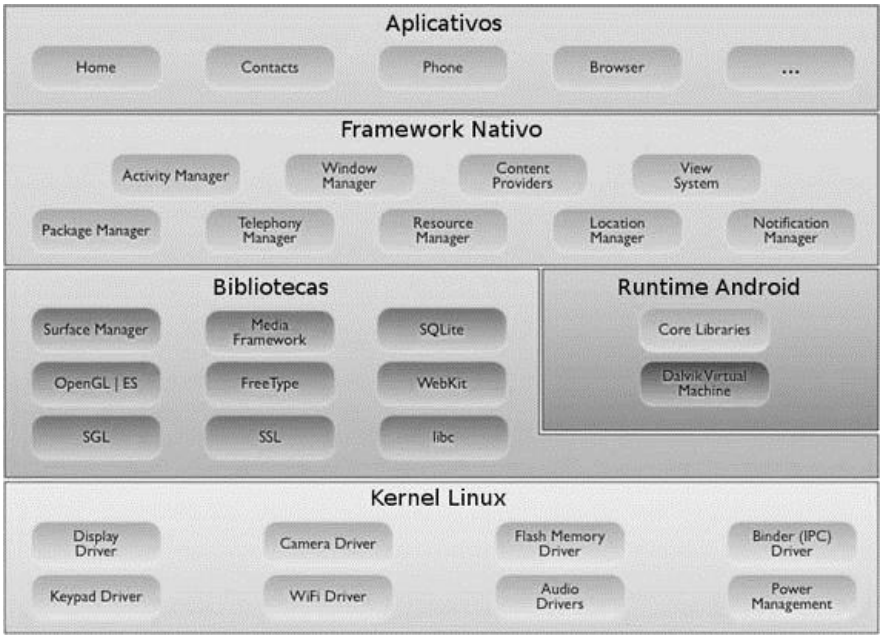
A plataforma Google Android permite o desenvolvimento de aplicativos na linguagem Java. No entanto, não existe uma JVM (Java Virtual Machine) inclusa no sistema, pois o Android tem sua própria máquina virtual, que é otimizada para execução de aplicativos móveis.

Tal máquina virtual foi construída pelos engenheiros da Google, visando a um consumo mínimo de memória e isolamento de processos. Ela permite que as aplicações escritas em linguagem Java sejam executadas normalmente no ambiente Android.

Quando o código Java é compilado para plataforma Android, é feita uma conversão de bytecod (.class) para Dalvik Executable (.dex) e, logo após, é criado um arquivo Android Package File (.apk), composto pelo arquivo .dex e por outros arquivos dependentes, como imagens e arquivos XML (Extensible Markup Language). No entanto, o desenvolvedor não precisa se preocupar com essas conversões, pois a IDE (Integrated Development Environment) se encarrega deste serviço.

3.2. Arquitetura do sistema Android

A arquitetura deste sistema, como pode ser vista na Figura 3, é dividida nos seguintes componentes: Aplicativos, Framework Nativo, Bibliotecas, Runtime Android, Kernel Linux. [2]



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

Figura 3. Arquitetura sistema Android

A camada de aplicativos é composta pelo conjunto de aplicativos nativos do sistema, dentre os quais se podem citar: cliente de e-mail, calendário, mapas, browser e internet, despertador, jogos, e outros.

A camada de framework é fornecida para que o desenvolvedor possa construir aplicativos, aproveitando os recursos que o sistema oferece. Este framework permite que o desenvolvedor tenha o mesmo acesso ao sistema que os aplicativos da camada de aplicativos possuem.

O sistema Android possui uma gama de bibliotecas C/C++ usadas pelos componentes do sistema. Tais bibliotecas são acessadas pelo programador via Java, através do framework do sistema. [4]

O Android Runtime permite que cada processo rode sua própria instância da máquina virtual.

A camada do Kernel é baseada em um Kernel Linux versão 2.6. Esta camada também atua como responsável pela abstração entre o hardware e os aplicativos.

Para construção de softwares que funcionem nesta arquitetura, a Google disponibilizou um pacote de utilidades chamado Android SDK

3.3. Android SDK – Software Development Kit

O Android SDK é um pacote de utilidades necessário para programação na plataforma Android. Ele é composto por um emulador de aplicações, API para linguagem Java e diversas ferramentas utilitárias que visam facilitar o processo de desenvolvimento. [5]

A instalação do SDK é simples e rápida, bastando descompactar o arquivo baixado e executar o instalador contido dentro dele. A partir deste ponto a instalação se torna automática.

O QEMU (emulador contido no SDK do Android) é uma ferramenta que permite a execução de um aplicativo Android em um computador convencional. Este emulador exibe na tela a imagem de um celular, como ilustrado na Figura 4, e assim o usuário consegue ter uma base de como a aplicação será exibida na tela de um dispositivo móvel.

É possível executar a troca de layout do emulador, caso o desenvolvedor queira trabalhar com uma imagem real do dispositivo no qual a aplicação será instalada. Isso se faz pelo uso de Skins das interfaces.

Fonte: Android SDK Resources.

Figura 4. Emulador do Android

Além da possibilidade de o emulador ser executado isoladamente, por linha de comando, existe também a possibilidade de integrá-lo à IDE de desenvolvimento. Assim, a própria IDE poderá executá-lo quando for necessário. Isso se torna possível com a instalação do plugin correspondente à IDE escolhida pelo desenvolvedor.

Atualmente existem duas IDEs disponíveis para plataforma Android: o Netbeans e o Eclipse. No entanto, somente o Eclipse será abordado neste documento, já que este ambiente é o único homologado pela Google.

4 Apresentação do sistema

A motivação inicial para este projeto é a construção de um sistema que possa auxiliar em tarefas que necessitem de informações sobre um determinado ambiente e que exerçam operações sobre o mesmo. Um exemplo seriam os controles de vigilância e tarefas de transporte.

Com o robô sendo controlado por um celular, este pode se movimentar pelo ambiente buscando informações de todos os ângulos. Tais informações seriam providas por meio de uma câmera instalada no equipamento, ou pelos sensores presentes no mesmo. Também existe a possibilidade de execução de tarefa de transporte pré-programada, ou até mesmo ativada remotamente pelo software presente no celular.

As tarefas acima citadas como exemplo, quando feitas manualmente, tornam-se caras e, algumas vezes, perigosas para o usuário. Com a queda nos preços de celulares mais modernos e os robôs com um custo mais acessível, utilizaram-se de ferramentas de desenvolvimento opensource para fabricação de um sistema com custo reduzido.

O objetivo é apresentar uma aplicação que estabeleça a comunicação entre um robô e um celular, gerando, assim, uma forma remota de controle entre tais dispositivos, promovida via Wi-Fi, o que fornece a possibilidade de se afastar do aparelho controlado.

O sistema instalado no dispositivo móvel recebe os dados fornecidos pelos sensores de movimento e mapeia-os para gerar comandos válidos para o robô. A partir desse ponto, o dispositivo móvel envia os comandos para o Wifibot via rede sem fio, instruindo-o a executar os movimentos requisitados pelo usuário.

A Figura 5 divide o processo de controle em três etapas para melhor entendimento do sistema.

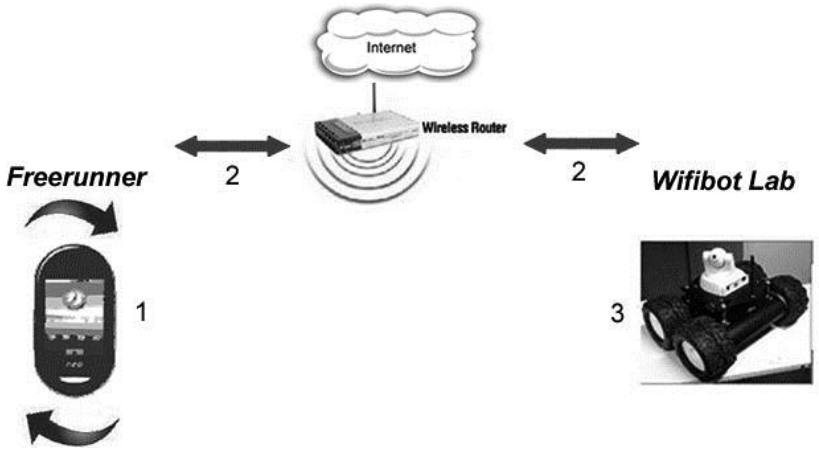


Figura 5. Arquitetura do sistema para movimentação do robô

Na primeira etapa, o usuário inclina o dispositivo móvel para um dos três eixos possíveis, com o intuito de movimentar o robô. O framework do Android coleta esses dados utilizando os sensores de movimento e os repassa para o aplicativo instalado. A partir dos dados coletados, o aplicativo gera comandos para serem enviados ao robô, baseados no protocolo padronizado pelo próprio Wifibot Lab.

Na segunda etapa, o sistema envia a mensagem contendo comandos gerados na etapa anterior por meio de uma conexão sem fio. O destino da mensagem é fixado pelo endereço IP (protocolo de internet) do robô, que possui um dispositivo de rede para que possa se conectar na rede.

No exemplo da Figura 5 a conexão está sendo provida por um roteador, que, assim, faz o repasse das mensagens entre os dispositivos conectados. No entanto, esta mesma conexão poderia ser possível via AD-HOC, como explicado na seção 2.1, e assim as mensagens estariam trafegando diretamente entre robô e celular.

A conexão entre os dispositivos não é diferenciada pelo sistema, e isso estabelece uma independência de modo de conexão. Assim, o usuário é capaz de usar o sistema da mesma forma, independentemente de a comunicação estar sendo provida via Infraestrutura ou Ad-Hoc.

Na terceira etapa, o robô recebe a mensagem contendo os comandos e reproduz os movimentos ordenados pelo usuário.

4.1. Acelerômetros

Os comandos para movimentação transmitidos ao robô são traduções dos movimentos que o usuário exerce sobre o dispositivo móvel, ou seja, quando o usuário deseja que o robô se mova, basta que incline o celular para o lado desejado.

Este meio de interpretação da vontade do usuário é possível graças aos acelerômetros presentes no celular. Tais acelerômetros são sensores integrados ao dispositivo que têm a capacidade de coletar dados sobre a inclinação em que o celular se encontra.

Esses sensores, quando presentes em dispositivos móveis, são compostos por duas superfícies em contato. Quando o aparelho é inclinado, essas superfícies sofrem uma força mecânica, que gera uma carga elétrica proporcional à força exercida; a partir desse ponto os dados são coletados e repassados ao sistema operacional.

Os acelerômetros fornecem dados que podem variar de -10 a 10, os quais se encontram organizados em três eixos, sendo eles: X, Y e Z.

O framework do Android faz o intermédio entre sensor e aplicação, fornecendo ao programador acesso aos dados, de forma simples e eficiente. A cada movimentação do celular, o aplicativo é alertado por um evento, que fornece os dados dos três eixos.

Dessa forma, para cada movimento exercido sobre o dispositivo móvel o celular envia uma mensagem ao robô, solicitando que ele se mexa para o lado desejado, o que possibilita que o usuário controle o celular de forma simples e precisa.

4.2. Protocolo de comunicação

O protocolo de comunicação para comandar o robô consiste no envio e recebimento de um vetor de caracteres. Cada caractere é responsável por uma função específica no sistema.

No entanto, existe mais de uma versão deste protocolo, as quais podem ser interpretadas pelo robô. Esta coleção permite que o usuário decida qual a forma mais conveniente de se comunicar com o robô, e a escolha é baseada no ambiente a ser desenvolvido.

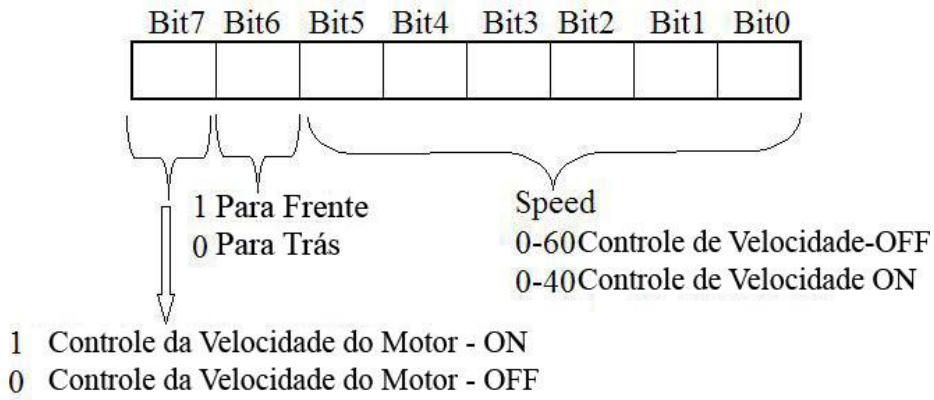
Para que o robô entenda qual versão foi escolhida para comunicação, abre canais para recebimento das informações em portas diferentes. Cada um desses canais, chamados sockets, recebe as informações da mesma forma. No entanto, o robô interpreta as informações diferentemente, usando do protocolo ao qual este canal está ligado.

Assim, o robô interpreta que um vetor de caracteres fornecido pela porta "X" deve ser processado de acordo com o protocolo correspondente àquela porta.

De acordo com os manuais de usuário, o protocolo considerado mais "simples" para comunicação com o robô é o que está referenciando à porta 1500 via TCP (Transmission Control Protocol). Este protocolo tem o nome de Protocolo Geral, pois se encarrega de funções básicas do sistema. São essas funções: o envio de comandos para movimentação do robô e o recebimento de algumas informações dos sensores instalados.

No envio, o vetor tem dois caracteres, cada um com oito bits. Os caracteres enviados devem conter informações sobre a movimentação das rodas. O primeiro caractere, chamado de "comg", fica responsável pelo controle das rodas esquerdas, e o segundo, chamado de "comd", pelas rodas direitas.

Na Figura 6 têm-se as informações contidas em cada caractere de envio.



Fonte: Manual Wifibot Lab.

Figura 6: Informações dos caracteres de envio

O controle de velocidade é independente para rodas esquerdas e direitas. Assim, para que o robô faça uma conversão, basta enviar comandos que contenham informações de velocidades diferentes para cada roda.

Um exemplo de envio para o robô é um vetor, onde o "comg" tem o valor de "11001010" e o "comd", o valor de "11101000". Neste exemplo, as duas rodas se movem para frente, mas com velocidades diferentes, visto que a roda esquerda está com velocidade "10" e a roda direita, com velocidade "40". Com esses valores, o robô faz uma conversão para esquerda, pois a roda direita se movimenta mais rápido do que a esquerda.

5 Conclusão

O atual nível de modernidade em que se encontra o ramo de tecnologia móvel, como demonstrado nas seções anteriores, permitiu a construção de um sistema visando fornecer uma maior sensibilidade na interface com o usuário. Isso foi possível graças aos sensores de movimento e conexões Wi-Fi.

Existe a possibilidade de uma futura melhoria no aplicativo, adicionando ao software a capacidade de entrar em modo automático. Para isso, o robô enviaria fotos captadas pela câmera e o sistema usaria algoritmos de reconhecimento de padrões para poder decidir qual o comando correto a ser enviado para o robô. Dessa forma, o robô poderia ser guiado buscando objetos nos ambientes, ou, mesmo, para impedi-lo de colidir com qualquer objeto.

Atualmente alguns sistemas de controle de robôs por celular usam conexões Bluetooth. Isso impõe que o dispositivo controlador não possa se afastar a mais de 100 metros do robô. Tal limitação faz com que o usuário não possa usá-lo para alguns casos, como, por exemplo, os de vigilância. Dentre esses sistemas, pode-se citar o Robô Pen, que foi projetado pelo engenheiro japonês Akasawa.

Outros robôs encontrados no mercado podem ser controlados via Wi-Fi, no entanto se faz necessário um software para comunicação com o mesmo. Essa aplicação necessita ser instalada em um computador, o que remove a portabilidade do usuário. É possível citar dentre estes o robô Erector Spykee que está sendo fabricado pela empresa Erector.

Referências

[1] Wifibot Enterprise. *Wifitbot Lab Documents*. Disponível em: <http://www.wifibot.com/page5.php>. Acesso em: 8 jun. 2010.

[2] LECHETA, R. *Google Android, Plataforma para Desenvolvimento de Aplicativos Móveis*. Editado por Rubens Prates, Novatec Editora Ltda. 2009.

[3] OPEN HANSET ALLIANCE. *Informações sobre o consórcio Android*. Disponível em: <http://www.openhandsetalliance.com/oha_overview.html>. Acesso em: 27 mai. 2010.

[4] GOOGLE. *Portal de desenvolvedores para Android.* Disponível em: <http://developer.android.com/index.html>. Acesso em: 8 jun. 2010.

[5] GOOGLE. *Software Development kit*. Disponível em: <http://developer.android.com/sdk/index.html>**.** Acesso em: 8 jun. 2010.