FACULDADE DE TECNOLOGIA SENAI CIMATEC
ESPECIALIZAÇÃO EM CONTROLE, AUTOMAÇÃO E ROBÓTICA

TADEU ABREU CERQUEIRA

PLATAFORMA DE COMANDO POR VOZ PARA UMA CADEIRA DE RODAS MOTORIZADA

Salvador
Setembro, 2014

TADEU ABREU CERQUEIRA

PLATAFORMA DE COMANDO POR VOZ PARA UMA CADEIRA DE RODAS MOTORIZADA

Trabalho de conclusão de curso apresentado ao Curso de Especialização em Automação e Controle da Faculdade de Tecnologia SENAI-CIMATEC como parte dos requisitos para a obtenção do título de Especialista em Controle, Automação e Robótica.

Orientador: Prof. Me. Oberdan Rocha Pinheiro

Salvador
Setembro, 2014

TADEU ABREU CERQUEIRA
PLATAFORMA DE COMANDO POR VOZ PARA UMA CADEIRA DE RODAS MOTORIZADA/ TADEU ABREU CERQUEIRA. – Salvador, Setembro, 2014-

81 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Me. Oberdan Rocha Pinheiro

Trabalho de Conclusão de Curso – Faculdade de Tecnologia SENAI CIMATEC, Setembro, 2014.

1. Cadeira de Rodas Motorizada. 2. Reconhecimento de Voz. I. Oberdan Rocha Pinheiro II. Faculdade de Tecnologia SENAI CIMATEC III. Plataforma de comando por voz para uma cadeira de rodas motorizada

CDU 02:141:005.7

TADEU ABREU CERQUEIRA

PLATAFORMA DE COMANDO POR VOZ PARA UMA CADEIRA DE RODAS MOTORIZADA

Trabalho de conclusão de curso apresentado ao Curso de Especialização em Automação e Controle da Faculdade de Tecnologia SENAI-CIMATEC como parte dos requisitos para a obtenção do título de Especialista em Controle, Automação e Robótica.

Aprovado em 11 de setembro de 2014.

Banca Examinadora

Oberdan Rocha Pinheiro - Orientador
Mestre em Modelagem Computacional e Tecnologia Industrial pela Faculdade de Tecnologia SENAI CIMATEC, Salvador-BA, Brasil
Faculdade de Tecnologia SENAI CIMATEC

Emanuel Benício de Almeida Cajueiro
Mestre em Engenharia Industrial pela Universidade Federal da Bahia, Salvador-BA, Brasil
Universidade Federal da Bahia - UFBA

Tito Luís Maia Santos
Doutor em Engenharia de Automação e Sistemas pela Universidade Federal de Santa Catarina, Florianópolis-SC, Brasil
Universidade Federal da Bahia - UFBA

Setembro, 2014

Aos portadores de necessidades físicas especiais e àqueles que acreditam que pequenas ações fazem um mundo melhor.

AGRADECIMENTOS

Agradeço em primeiro lugar à Deus, pela dádiva da vida, à minha mãe Dilma, à minha namorada Laryssa e a todos os meus familiares por todo o apoio dado nos momentos que passei.

Sou grato ao meu orientador Oberdan, por aceitar e acreditar na minha proposta e aos meus colegas e amigos da UFBA e do SENAI - CIMATEC, por me ajudarem sempre quando precisei, com sugestões importantes para conseguir realizar minhas atividades durante todo o curso de especialização.

Agradeço ao ISI em Automação e Controle e ao meu coordenador, Dr. Herman Augusto Lepikson, por todo suporte e apoio que me foi dado para realização deste trabalho.

Sou grato ao SENAI - CIMATEC, por toda estrutura que foi fornecida não só a mim, mas a todos os meus colegas, sendo de importância singular para realização de trabalhos inovadores que venham a fazer com que a Bahia se torne cada vez mais participativa no cenário científico e tecnológico do Brasil e do Mundo.

Faço um agradecimento especial ao Dr. Josemar Rodrigues de Souza, o idealizador deste projeto, por todo apoio e incentivo em ver essa ideia ser concretizada.

A Todos o meu Muito Obrigado!

RESUMO

No presente trabalho desenvolveu-se uma plataforma de comando para controlar por voz os movimentos de uma cadeira de rodas motorizada por usuários tetraplégicos ou paraplégicos com limitações físicas severas nos membros superiores. Implementou-se um circuito de controle para enviar sinais elétricos que ativam os comandos direcionais da cadeira e assim a controla através de um *notebook*. Dessa forma, os comandos são enviados ao circuito de interface por uma porta de comunicação serial do *notebook*, utilizando para isso um *software* programado em linguagem Java. Esse *software* é capaz de reconhecer comandos vocais dados pelo usuário e convertê-los em sinais a serem enviados ao circuito de interface para assim movimentar a cadeira de rodas, além de disponibilizar um servidor local para que outros dispositivos remotos, como celulares *smartphones* ou *Tablets*, possam se comunicar com a cadeira de rodas, abrindo novas perspectivas de trabalhos. Portanto, esse trabalho traz uma possibilidade clara de melhoria na mobilidade, acessibilidade e conforto das pessoas portadoras de necessidades especiais, através de uma solução com um baixo custo de desenvolvimento.

Palavras-chaves: Cadeira de Rodas Motorizada. Reconhecimento de Voz. Linguagem Java.

ABSTRACT

In this work is developed a platform to control the movements of a motorized wheelchair using voice command by quadriplegics users or paraplegics with severe physical limitations in the upper limbs. It is implemented a interface circuit to send electrical signals that activate the directional commands from the chair and controls through a notebook. Thus, commands are sent to an interface circuit with a serial port communication of a notebook, using a software programmed in Java language. This software is able to recognize voice commands given by the user and convert them into signals to be sent to the interface circuit to move the wheelchair, in addition the software providing a local server to other remote device, such as mobile smartphones or tablets, communicate with the wheelchair, opening new prospects for work in the area. Therefore, this work provides a clear opportunity for improvement in mobility, accessibility and comfort of people with special needs, through a solution with a low cost of development.
Key-words: Motorized Wheelchair. Voice Recognition. Java Language.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AP	*Access Point*
API	*Aplication Programing Interface*
BSS	*Basic Service Set*
CI	Circuito Integrado
DAC	*Digital to Analog Converter*
DSP	*Digital Signal Processor*
FPGA	*Field-Programmable Gate Array*
IBGE	Instituto Brasileiro de Geografia e Estatística
IHM	Interface Homem-Máquina
IP	*Internet Protocol*
LAN	*Local Area Network*
LPC	*Linear Predictor Coeficients*
PCI	Placa de Circuito Impresso
PWM	*Pulse-Width Modulation*
RS232	Recommended Standard 232
TCP	*Transmission Control Protocol*
WI-FI	*Wireless-Fidelity*

SUMÁRIO

1 INTRODUÇÃO

Segundo dados do IBGE (2010) - Instituto Brasileiro de Geografia e Estatística - existem aproximadamente 1 milhão de pessoas que usam cadeira de rodas no Brasil. São indivíduos que utilizam a cadeira de rodas por terem algum tipo de deficiência física e devido a isso apresentam grandes dificuldades na realização de tarefas simples do cotidiano, necessitando inclusive que outras pessoas os auxiliem na maioria das vezes (CHIELE; ZERBETTO, 2010).

Devido ao crescente desenvolvimento científico e tecnológico na área de produtos para cadeirantes observado nos últimos anos, as vidas dos diversos portadores de necessidades físicas especiais tem mudado significativamente (ALBRECHT, 2012). Atualmente já existem soluções que conseguem proporcionar aos portadores de necessidades físicas especiais uma maior independência, melhorando consideravelmente a qualidade de vida. Umas dessas soluções, já bem popularizada, é o uso de cadeiras motorizadas, controladas através de um *jostick* acoplado, que visa diminuir o desgaste físico nos membros superiores pelos cadeirantes.

Várias aplicações surgem a partir da cadeira de rodas motorizada com o objetivo de aumentar ainda mais o conforto e a mobilidade dos deficientes físicos. Existe atualmente uma série de trabalhos que fazem, como pretendido neste trabalho, o reconhecimento de voz aplicado ao controle dos movimentos em cadeiras de rodas, como em (CHIELE; ZERBETTO, 2010), (NETO; CASTRO; FELIX, 2010) e (SILVA, 2007). Em ambos os trabalhos a técnica envolvida no reconhecimento da voz é basicamente a mesma, usando redes neurais artificiais.

Vale ressaltar que existem também outros trabalhos mais avançados, como por exemplo visto em (FERREIRA et al., 2007), no qual a cadeira é controlada por sinais cerebrais enviados pelo usuário.

1.1 OBJETIVO

O objetivo principal desse trabalho é desenvolver uma plataforma de comando por voz para controlar os movimentos de uma cadeira de rodas motorizada e facilitar a mobilidade dos portadores de necessidades físicas especiais, principalmente dos tetraplégicos ou paraplégicos com limitações severas de movimento nos membros superiores.

1.1.1 OBJETIVOS ESPECÍFICOS

Para atingir o objetivo proposto implementa-se um circuito de interface que realiza a função de enviar os sinais analógicos de controle ao circuito da cadeira de rodas, sendo utilizado para isso um Arduíno programado e um circuito de interface para gerar as saídas desejadas de acordo com as entradas de dados provenientes do *notebook* que é colocado junto à cadeira. Assim, a ideia é que a partir de comandos por voz, dados pelo usuário ao *notebook*, possam ser enviados sinais de controle para o circuito da cadeira de rodas, que corresponda ao movimento solicitado.

Nesse sentido, toda a comunicação entre o Arduíno e *notebook* é realizada por *bytes* na forma serial, enviados no barramento de interface RS232. Para isso, é implementado um *software* em linguagem Java que acessa a porta de comunicação serial do computador e assim envia os *bytes* correspondentes para o Arduíno, que por sua vez interpreta e envia uma sequência de bits através da porta de saída digital a um circuito de interface. Este circuito converte a sequência de bits provenientes do Arduíno em sinais analógicos de tensão que são enviados ao circuito original da cadeira de rodas para, assim, controlar o movimento. Na Figura 1 pode-se observar como ocorre o fluxo de dados para controlar através da voz os movimentos da cadeira de rodas.

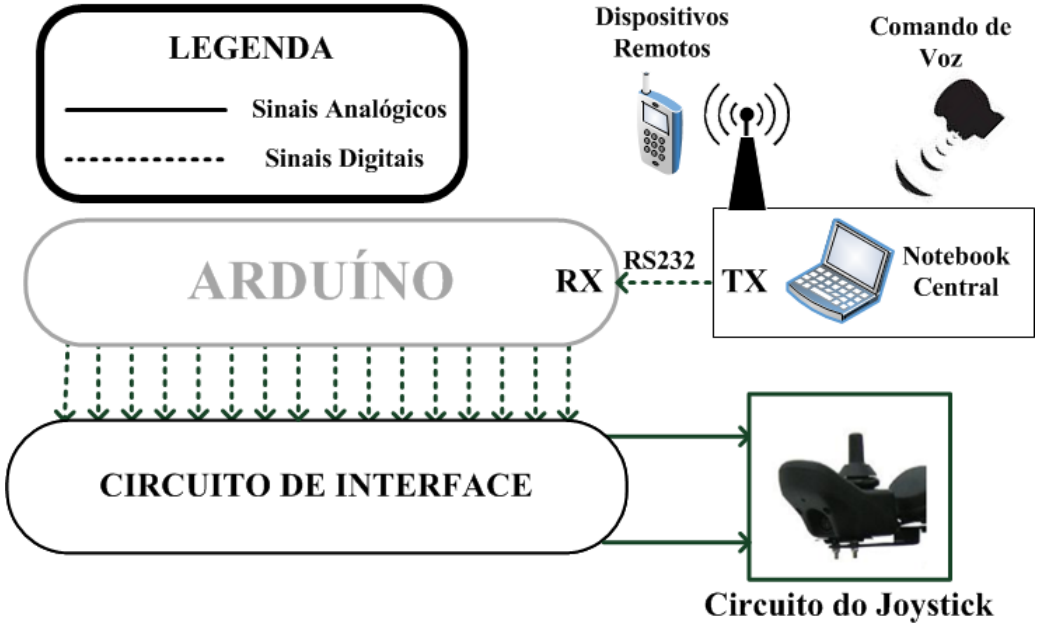

Figura 1 – Esquemático da plataforma de comando por voz para a cadeira de rodas motorizada. Fonte: Autor.

O *software* implementado no *notebook* gera também um servidor na rede local, disponibilizando assim que um outro dispositivo remoto, como *Smartphone* ou *Tablet*, possa também enviar comandos direcionais para a cadeira de rodas, desde que conectados à mesma rede do *notebook*, através de uma conexão WI-FI.

1.2 JUSTIFICATIVA

Tarefas simples do dia-a-dia tornam-se desafios muito severos aos portadores de necessidades físicas especiais, que além de todas as dificuldades acabam também enfrentando discriminação por parte de algumas pessoas da sociedade (ALBRECHT, 2012). Nesse sentido, projetos que melhorem a mobilidade e o conforto reduzem a dificuldade de realizar atividades sem a ajuda de terceiros e aumentam muito a qualidade de vida das milhares de pessoas que usam a cadeira de rodas para se locomover.

O uso já bastante difundido do *joystick* ou similar para controlar uma cadeira de rodas motorizada pode ser impossibilitado para determinados portadores de necessidades físicas especiais, como os tetraplégicos e paraplégicos com limitações severas no movimento dos braços. Dessa forma, a aplicação sugerida neste trabalho irá controlar a cadeira de rodas através de comandos vocais, facilitando a vida desses usuários e garantindo uma maior mobilidade e conforto no movimento.

Adicionalmente, este trabalho oferece a possibilidade de conectar a cadeira de rodas a um *smartphone* ou *Tablet*, visto que o *software* implementado no *notebook* disponibiliza um servidor local que pode enviar e receber informações de um cliente remoto, possibilitando controlar os movimentos da cadeira remotamente. Com isso, novos trabalhos podem ser desenvolvidos visando melhorar ainda mais a mobilidade e interface da plataforma de comando.

Além disso, esse trabalho gera outras perspectivas de implementações futuras na cadeiras de rodas, visto que podem ser desenvolvidas outras técnicas, como reconhecimento de gestos que, juntamente com o sistema desenvolvido, pode proporcionar ainda mais facilidade no transporte dos portadores de necessidades físicas especiais.

1.3 ORGANIZAÇÃO DA MONOGRAFIA

Este trabalho é dividido em sete capítulos. No capítulo 1 é mostrado a problemática abordada, bem como o objetivo do trabalho e a justificativa do mesmo.

No capítulo 2 são abordados aspectos técnicos da Cadeira de Rodas utilizada no projeto, destacando as partes de motorização, baterias e o uso do *joystick*.

No capítulo 3 é explicado como foi desenvolvido o circuito de interface entre o *notebook* e o microcontrolador da cadeira de rodas para controlar os movimentos, explicitando a programação do *firmware* no Arduíno, a comunicação serial com o *notebook*, a conversão digital/analógica e as conexões realizadas com o *joystick* para enviar os sinais analógicos.

No capítulo 4 é explicado como são desenvolvidas as funções do reconhecimento

de voz via *software* em linguagem Java no *notebook*. Além disso, é mostrado como se implementa a comunicação serial em Java e como é criado servidor local que disponibiliza a comunicação com um dispositivo remoto.

No capítulo 5 é explicitado como o foi realizada a implementação do circuito de interface da plataforma de comando por voz em *protoboard* e em PCI (Placa de Circuito Impresso). É mostrado também os resultados dos testes no protótipo final desenvolvido.

Por último, no capítulo 6, são feitas as considerações finais do trabalho e as contribuições do mesmo, além de sugerir algumas atividades futuras de pesquisa para melhorar o projeto.

2 A CADEIRA DE RODAS MOTORIZADA

Neste capítulo serão vistos os aspectos técnicos da cadeira de rodas motorizada da *Freedom Carbon* que foi utilizada para a implementação da plataforma de comando por voz. Serão dadas explicações a respeito das especificações do motor da cadeira, de suas baterias e principalmente o funcionamento do circuito do *Joystick* que controla o movimento através de dois sinais de tensão.

Na Figura 2 pode-se observar, na sua forma original, a cadeira de rodas motorizada que foi utilizada neste trabalho.

Figura 2 – Cadeira de Rodas Motorizada. Fonte: (FREEDOM CARBON, 2012).

2.1 MOTORIZAÇÃO

O sistema de motorização da *Freedom Carbon* é composto basicamente de dois motores, com rotação reversível, de corrente contínua que apresentam uma tensão de operação igual a $24V$ e potência ativa em torno de $400W$ (FREEDOM CARBON, 2012). Dessa forma, como a cadeira tem duas rodas fixas com motores reversíveis, ela atua como um robô diferencial, podendo-se locomover para todas as direções, a depender das velocidades que são aplicadas em cada uma das suas rodas (SIEGWART; NOURBAKHSH, 2004).

O sistema de freio dos motores é do tipo eletromagnético, sendo composto por dois bloqueadores eletromagnéticos que quando não recebem corrente elétrica geram um campo magnético oposto ao movimento de giro do motor, impedindo o movimento. Contrariamente, quando os bloqueadores magnéticos são alimentados por corrente elétrica

liberam os eixos e permitem que o motor gire de acordo com o comando desejado pelo usuário (FREEDOM CARBON, 2012).

Na Figura 3 pode-se observar um desenho do sistema de motorização usado pela *Freedom Carbon*, destacando o motor de corrente contínua, o sistema de redução, o freio eletromagnético, as polias e a alavanca de freio que é acionada de acordo com o acionamento do *Joystick*, para desativar ou ativar o freio eletromagnético.

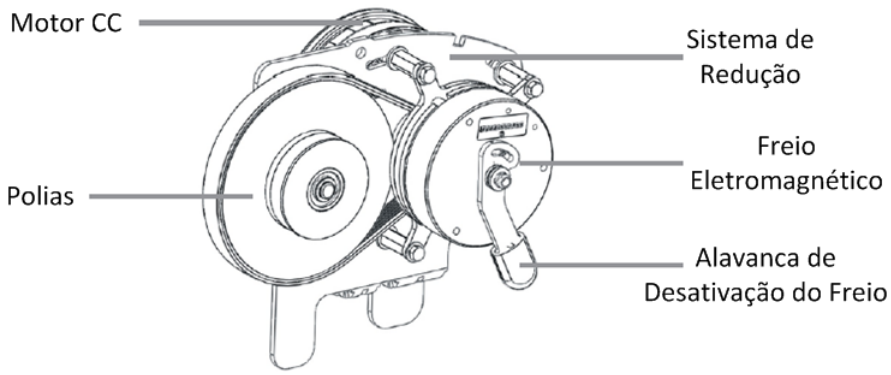

Figura 3 – Esquema Simplificado do Sistema de Motorização da Cadeira de Rodas. Fonte: (FREEDOM CARBON, 2012).

É importante destacar também que todo esse sistema de motorização é protegido por carenagens, de acordo com o que é visto no desenho da Figura 4 a seguir.

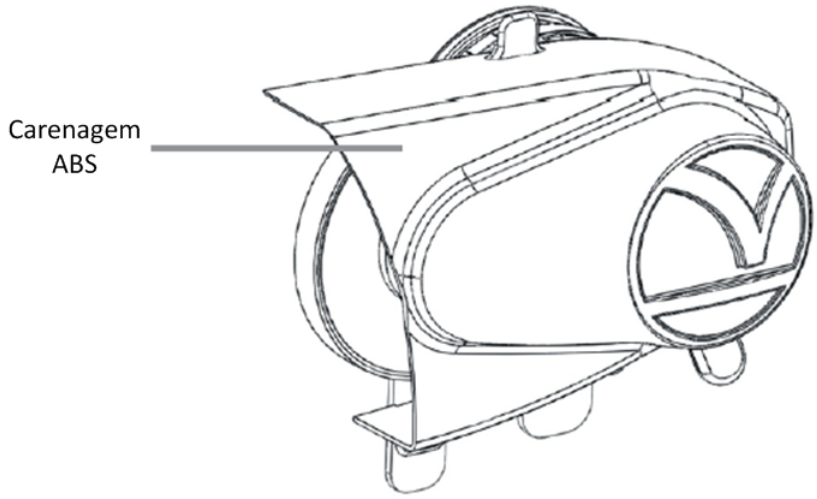

Figura 4 – Desenho da Carenagem para Proteção do Sistema de Motorização da Cadeira. Fonte: (FREEDOM CARBON, 2012).

2.2 BATERIAS

No sistema da cadeira de rodas motorizada da *Freedom Carbon* são usadas duas baterias chumbo-ácidas SM, uma com tensão e corrente nominais de $12V$ e $50A$ e outra com $12V$ e $70A$[1], sendo ambas do modelo denominado SSX (FREEDOM CARBON, 2012). Essas baterias são carregadas através de um carregador com controle de corrente específico da *Freedom Carbon*, com capacidade de $24V$ e $4A$ (FREEDOM CARBON, 2012).

Na Figura 5 pode-se observar o esquema de conexão para carregar as baterias da cadeira de rodas (FREEDOM CARBON, 2012).

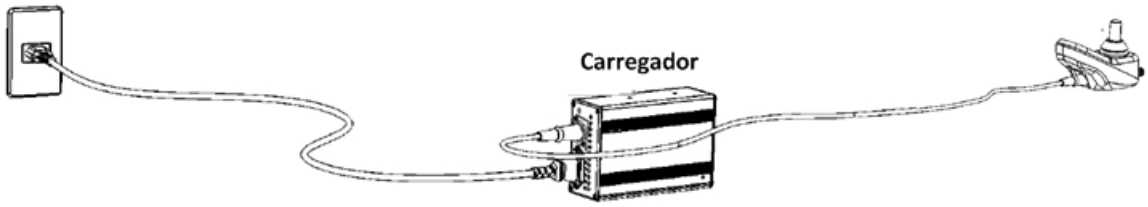

Figura 5 – Carregador de Bateria e Cabos. Fonte: (FREEDOM CARBON, 2012).

A conexão com o *joystick* é realizada através de um conector Canon, como pode ser observado na Figura 6.

Figura 6 – Conectores Canon. Fonte: Autor.

2.3 JOYSTICK

O *joystick* da cadeira de rodas da *Freedom Carbon* envia sinais a um circuito embarcado, composto por um microcontrolador PIC da *Microchip*. Dessa forma, de acordo com o movimento executado na alavanca do *joystick* são enviados dois sinais de tensão ao circuito do microcontrolador, que por sua vez interpreta e envia outros sinais ao sistema de motorização da cadeira para realizar os movimentos desejados.

[1] A bateria de $50A$ alimenta um dos motores e a de $70A$ alimenta o outro motor e também o circuito de potência, por isso a diferença de corrente entre ambas as baterias

Na Figura 7 pode-se observar a foto do circuito do microcontrolador, destacando-se o PIC e barramento onde são enviados os sinais do *joystick* diretamente ao circuito.

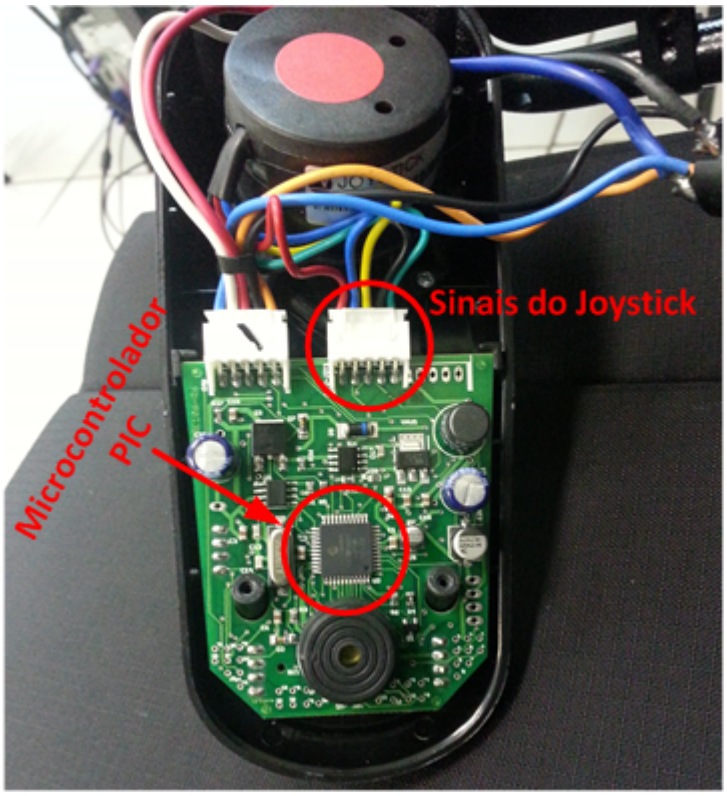

Figura 7 – Circuito do *joystick* visto de cima. Fonte: Autor.

Este trabalho tem a ideia de realizar um circuito que faz a mesma função do *joystick*, mas através de um *notebook* que aplica sinais analógicos de tensão ao circuito do microcontrolador para assim controlar os movimentos da cadeira de rodas. Em resumo, implementa-se um circuito que gera em sua saída os mesmos sinais de tensão que o *joystick* original da cadeira de rodas fornece para o circuito do microcontrolador, de acordo com o movimento que é solicitado.

Para desenvolver o circuito que simula os sinais de tensão do *joystick* original da cadeira de rodas é necessário antes de tudo entender quais os fios responsáveis por mandar os respectivos sinais e os valores específicos de tensão para serem enviados ao circuito do PIC e assim realizar os movimentos na cadeira de rodas.

Na Figura 8 pode-se observar um esquema com os fios de ligação no circuito do microcontrolador, destacando os barramentos provenientes do *joystick* e do conector Canon com seus respectivos valores ou faixas de tensão, medidos por um multímetro.

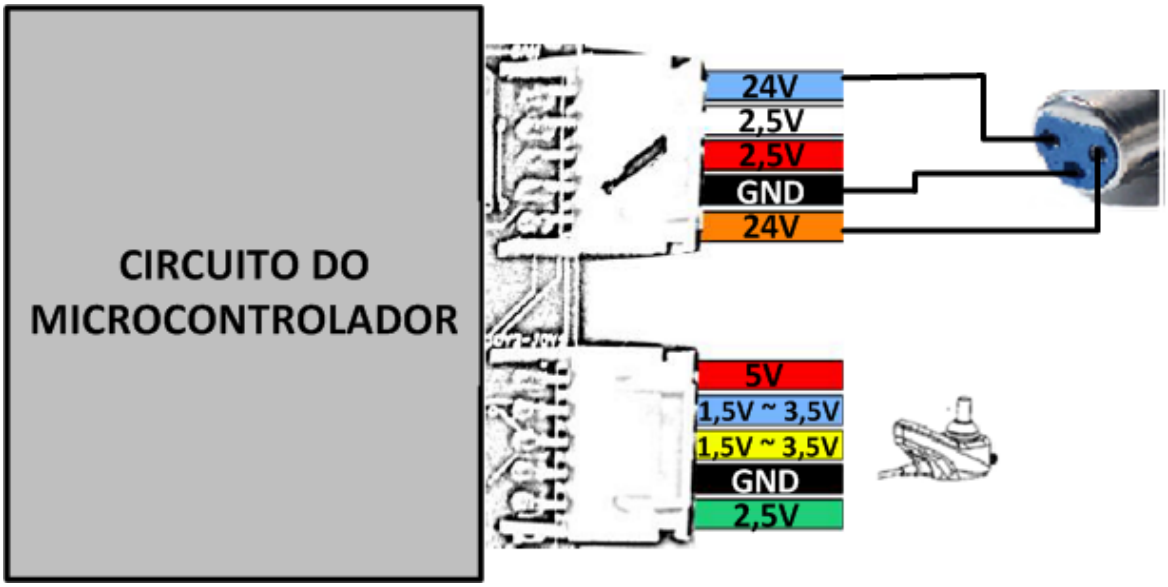

Figura 8 – Esquema de ligação entre o *joystick* e o circuito do microcontrolador. Fonte: Autor.

O fio azul proveniente do *joystick* controla as ações de ir para frente ou ir para trás, enquanto que o fio amarelo controla as ações de ir para esquerda ou direita.

Para cada movimento, foram realizadas medidas nas tensões de cada fio com o auxílio de um multímetro digital. Com isso, pode-se construir a Tabela 1, em que se observa os respectivos valores de tensão dos fios do barramento controlado pelo *joystick* e entende-se de que forma esses valores alteram o movimento da cadeira de rodas.

Tabela 1 – Valores de tensão dos fios do barramento controlado pelo *joystick* e os respectivos movimentos.

	Parado	**Direita**	**Esquerda**	**Frente**	**Ré**
Fio Azul	2,5V	2,5V	2,5V	3,5V	1,5V
Fio Amarelo	2,5V	1,5V	3,5V	2,5V	2,5V

Nota-se que é possível controlar os movimentos da cadeira de rodas apenas variando os valores de tensão aplicados nos fios azul e amarelo localizados no barramento proveniente do *joystick*. Assim, basta implementar um circuito que seja capaz de enviar sinais analógicos a esses pinos, a partir de comandos enviados de um microcomputador, sendo que dependendo da tensão aplicada, a velocidade de cada roda da cadeira varia sua intensidade (os extremos são em $1,5V$ e $3,5V$). No entanto, vale destacar que o circuito do microcontrolador tem muitas proteções contra erros de comando, de forma que o envio de tensões fora das faixas especificadas nos respectivos fios faz com que a cadeira seja travada e nenhum movimento possa ser realizado até que o circuito seja desligado e ligado novamente.

O circuito que simula as ações do *joystick* original deve ser projetado para respeitar

os limites operacionais das tensões a serem enviadas ao circuito do microcontrolador da cadeira de rodas, impedindo que ocorra o travamento do movimento por conta de uma tensão enviada fora da faixa de operação. Com isso, recomenda-se utilizar um circuito de interface que forneça sinais analógicos de tensão a partir de comandos digitais provenientes do *notebook*.

3 O CIRCUITO DE INTERFACE

Neste capítulo será mostrado como foi concebido o circuito de interface que simula as ações do *joystick* original e faz a comunicação entre o *notebook* central, a cadeira de rodas e o Arduíno, para realizar os movimentos desejados. Destacam-se a utilização de conversores digital/analógico para enviar os sinais de tensão ao circuito do microcontrolador da cadeira de rodas.

Serão dadas explicações a respeito da plataforma Arduíno utilizada, sobre como é realizada a comunicação serial e como é feita conversão analógica/digital. Por fim, é ilustrada a lógica implementada para o *firmware* do Arduíno.

3.1 A PLATAFORMA ARDUÍNO

O Arduíno é uma plataforma eletrônica de código aberto que é baseado em *hardware* e *software* de fácil utilização, sendo destinado principalmente para realização de projetos de desenvolvimento (ARDUINO©, 2005).

Com o Arduíno é possível se comunicar pela porta serial de um computador, enviar sinais digitais em paralelo, processar informações e outras funcionalidades. Neste trabalho, é necessário se comunicar com o *notebook* através da porta de comunicação serial, processar o sinal enviado e mandar uma sequência de 16 bits para um outro circuito que converterá essa sequência binária em dois sinais analógicos de tensão.

Viu-se que o Arduíno Mega se encaixa bem nas especificações necessárias, já que o mesmo apresenta mais de 16 pinos para saídas digitais, diferentemente do Arduíno Uno, por exemplo, e entre outros modelos. Na Figura 9 pode-se observar o Arduíno Mega, utilizado no projeto deste trabalho.

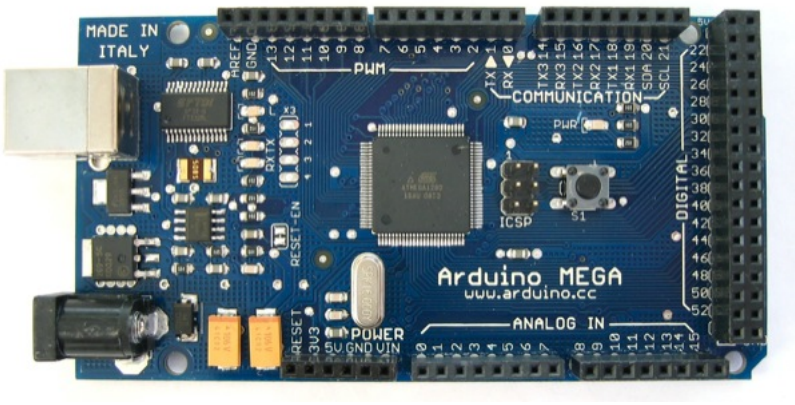

Figura 9 – Arduino Mega ADK. Fonte: (ARDUINO©, 2005).

3.2 COMUNICAÇÃO SERIAL

O circuito prevê a utilização da comunicação serial para trocar informações entre o *notebook* e o Arduíno, de forma que é importante ter o entendimento mais aprofundado sobre como acontece esse envio e recebimento de *bytes* na comunicação serial.

O envio de mensagens digitais prevê a utilização de uma sequência normalmente longa de bits. No entanto, não é tão prático e muito menos econômico realizar a transferência de todos os bits simultaneamente, ou seja, em paralelo. Isso porque para transmissão em paralelo é necessário um fio para cada bit transmitido. Com isso, tendo-se que enviar 8 bits, por exemplo, seria necessário 8 fios que permitissem o envio do sinal de tensão representativo do bit, formando assim uma palavra ou conjunto de bits (CANZIAN, 2002).

Nesse sentido, utiliza-se a transmissão bit-serial, normalmente chamada de transmissão serial, que é o método de comunicação escolhido por diversos periféricos de computadores atualmente. Nessa transmissão a mensagem digital é quebrada em partes menores e transmitida sequencialmente através de um único fio, economizando espaço e reduzindo custos (CANZIAN, 2002). Na Figura 10 pode-se observar uma comparação entre a comunicação paralela e a serial.

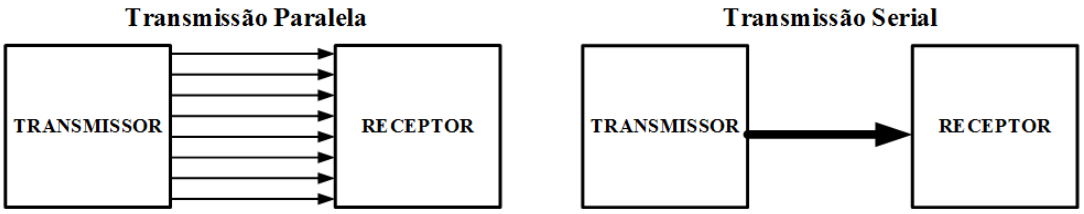

Figura 10 – Comparação entre comunicação serial e paralela. Fonte: Autor.

A comunicação serial pode ser de dois tipos principais: Síncrona e Assíncrona. A comunicação serial síncrona baseia-se em um sinal de *clock* em separado que é associado ao dado. Dessa forma, o sincronismo é feito a cada caractere, através de um *clock* localizado no receptor, cujo período é menor que o período de duração do bit (GONZAGA, 2005).

Já na comunicação serial assíncrona não existe um sincronismo contínuo entre o transmissor e receptor. Portanto, a mensagem é enviada sem a necessidade de um *clock*, como ocorre na comunicação síncrona (GONZAGA, 2005). Assim, o formato do caractere na comunicação assíncrona deve seguir um protocolo para que o receptor saiba em que momento deve começar a ler os dados e em que momento parar. Considerando que se deseja enviar 8 bits de dados, deve haver além dos bits de dados 1 *start* bit ou bit de início, 1 bit de paridade e 1 *stop* bit ou bit de parada (GONZAGA, 2005). A Figura 11 ilustra como são enviados os dados na comunicação serial assíncrona, destacando seus bits de controle.

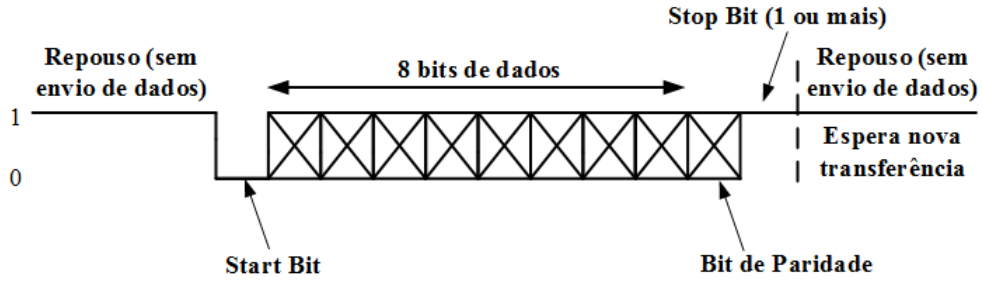

Figura 11 – Formato do caractere para a comunicação serial assíncrona. Fonte: (GONZAGA, 2005).

Existem alguns parâmetros de configuração que devem ser escolhidos para que ocorra a comunicação serial assíncrona: Taxa de Comunicação ou *BAUD RATE*, Bits de dados, Paridade e Bits de Parada.

O *BAUD RATE* consiste na velocidade de transmissão, ou seja, na quantidade de bits enviados ou recebidos por unidade de tempo. Na Tabela 2 pode-se observar as taxas de comunicação mais comuns e o respectivo período de permanência para cada bit.

Tabela 2 – Taxas de comunicação assíncrona mais comuns. Fonte: (GONZAGA, 2005).

Taxa de Comunicação (bits/s)	Tempo de duração do bit
110	$9,1ms$
150	$6,66ms$
300	$3,33ms$
600	$1,66ms$
1200	$833\mu s$
2400	$416\mu s$
4800	$208\mu s$
9600	$104\mu s$
19200	$52\mu s$
57600	**17 μs**

Neste trabalho foi utilizado um *BAUD RATE* de 57600 bits/s para realizar as comunicações seriais entre o computador e o Arduíno.

Os Bits de dados correspondem à quantidade de bits desejada para o dado enviado. Normalmente um dado transmitido através de uma comunicação serial apresenta 8 bits, no entanto existem aplicações que utilizam dados de 12 ou mais bits (CANZIAN, 2002).

Além da comunicação serial, o circuito necessita também converter duas sequências digitais em dois valores respectivos de tensão a serem aplicados em dois fios no circuito do microcontrolador da cadeira de rodas. Dessa forma, a seguir será dado um entendimento de como é feita essa conversão.

3.3 CONVERSÃO DIGITAL/ANALÓGICA

Usando apenas o Arduíno e a comunicação serial do *notebook* não é possível enviar os sinais de tensão dentro dos limites operacionais para o circuito do microcontrolador da cadeira de rodas e simular a ação do *joystick* original. Isso porque o Arduíno fornece apenas tensões de $5V$ (para saída em nível lógico '1') ou $0V$ (para saída em nível lógico '0').

É importante destacar que mesmo utilizando a função PWM no Arduíno, não serão enviados valores de tensão nas faixas específicas para o circuito do microcontrolador da cadeira de rodas, já que na modulação por largura de pulso só são enviados sinais de $5V$ ou $0V$ em uma determinada frequência de *clock*, de tal forma que a média em um período seja o valor de tensão desejado. No entanto, os sinais de $5V$ e $0V$ do sinal PWM são percebidos pelo circuito do microcontrolador da cadeira de rodas, que impossibilita o movimento e trava qualquer tentativa posterior de comandar a cadeira. Dessa forma, é concebido um circuito de interface entre o Arduíno e a cadeira de rodas, que faz a conversão de uma sequência de 16 bits, enviados dos pinos I/O do Arduíno, para dois valores analógicos de tensão que serão aplicados no circuito da cadeira de rodas.

O CI utilizado nesse projeto é o DAC0808, que é um conversor digital/analógico de 8 bits, de maneira que para o projeto são necessários dois desses CIs. De acordo com o fabricante (NATIONAL SEMICONDUCTOR, 1995), existe uma aplicação típica do DAC0808 para que o mesmo funcione como um conversor de 8 bits com saída de tensão entre $0V$ e $5V$. A Figura 12 mostra o circuito que é implementado juntamente com o DAC0808 para que o mesmo realize a conversão de 8 bits em um sinal analógico.

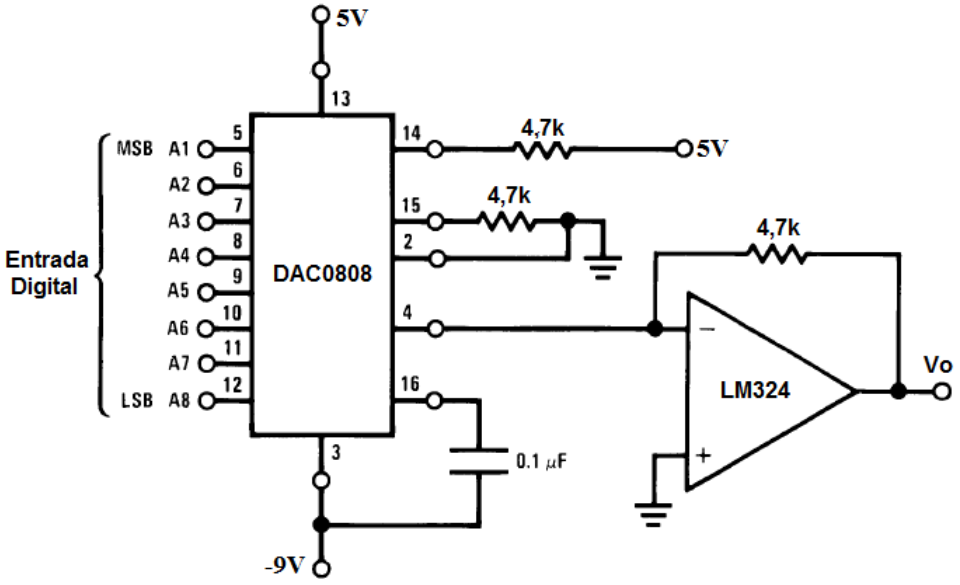

Figura 12 – Aplicação utilizada para o DAC0808 no sistema de controle. Fonte: (NATIONAL SEMICONDUCTOR, 1995).

Note que há a utilização de uma fonte de tensão igual a $-9V$ para alimentar o circuito do conversor. Dessa forma, optou-se por utilizar uma bateria de $9V$ com polaridade invertida para fornecer essa tensão no circuito. Além disso, é utilizado o LM324 na saída do DAC para amplificar o sinal.

Para encontrar a tensão de saída V_0 do circuito conversor realiza-se o seguinte cálculo (NATIONAL SEMICONDUCTOR, 1995):

$$V_0 = 5. \left(\frac{A1}{2} + \frac{A2}{4} + \cdots + \frac{A8}{256} \right) \tag{3.1}$$

Sendo que os bits de $A1$ até $A8$ são provenientes da porta de saída digital do Arduíno, assumindo cada um deles valores '1' para $5V$ ou '0' para $0V$. Dentro da programação do Arduíno o cálculo da Equação 3.1 é levado em consideração para fornecer a sequência correta de bits e aplicar a tensão desejada no circuito do microcontrolador da cadeira de rodas.

Como é necessário aplicar dois sinais analógicos de tensão[1], são implementados dois circuitos idênticos ao que foi ilustrado na Figura 12, sendo que um envia a tensão ao fio azul para comandar os movimentos para frente e para trás, enquanto o outro DAC aplica as tensões no fio amarelo para controlar o movimento da direita e esquerda. Com isso, tem-se a interface completa entre o Arduíno, que envia 16 bits nas saídas digitais, e os DACs, que paralelamente convertem a sequência binária em respectivos sinais analógicos de tensão para serem aplicados na cadeira de rodas e realizar o movimento desejado.

3.4 O FIRMWARE E AS SIMULAÇÕES DO CIRCUITO

Antes de implementar o circuito de interface fisicamente foram realizadas simulações para verificar se o comportamento atende aos requisitos desejados. Primeiramente foi necessário implementar o *firmware* do Arduíno em linguagem C para interagir com o circuito dos DACs. Nesse caso, o objetivo do *firmware* é ler caracteres vindos do *notebook* e para cada tipo de caractere lido enviar uma respectiva sequência de bits ao circuito dos DACs.

Seguindo o propósito do projeto, a ideia é movimentar a cadeira de rodas para direita, esquerda, frente e para trás. Logo, são quatro movimentos possíveis, além do comando de parar qualquer um desses movimentos, ou seja, cinco comandos que podem ser executados pelo usuário. Com isso, deve-se utilizar cinco caracteres, cada um representando uma ação: parar, frente, ré, direita e esquerda.

[1] Aplica-se a tensão entre $1,5V$ e $3,5V$ nos fios azul e amarelo do barramento proveniente do *joystick* original.

Escolheu-se os caracteres de '0' a '4', que representam na tabela ASCII os decimais de 48 a 52. Dessa forma, o *notebook* envia para o Arduíno apenas esses cinco caracteres pela porta serial. Na Tabela 3 pode-se observar os caracteres escolhidos, seu valor equivalente na tabela ASCII e o comando representativo de cada um deles dentro do circuito de interface.

Tabela 3 – Caracteres escolhidos para comunicação com o Arduíno e execução dos movimentos na cadeira de rodas.

Caractere	Valor decimal na Tabela ASCII	Função de Movimento da Cadeira
'0'	48	Parar o Movimento
'1'	49	Ir para Frente
'2'	50	Ir para Trás
'3'	51	Virar para Direita
'4'	52	Virar para Esquerda

De acordo com o caractere lido pelo Arduíno são enviadas duas sequências de 8 bits cada, representando as duas tensões analógicas que deverão ser aplicadas no circuito da cadeira de rodas. Na Figura 13 pode ser observado o fluxograma simplificado da lógica do *firmware* desenvolvido para o Arduíno.

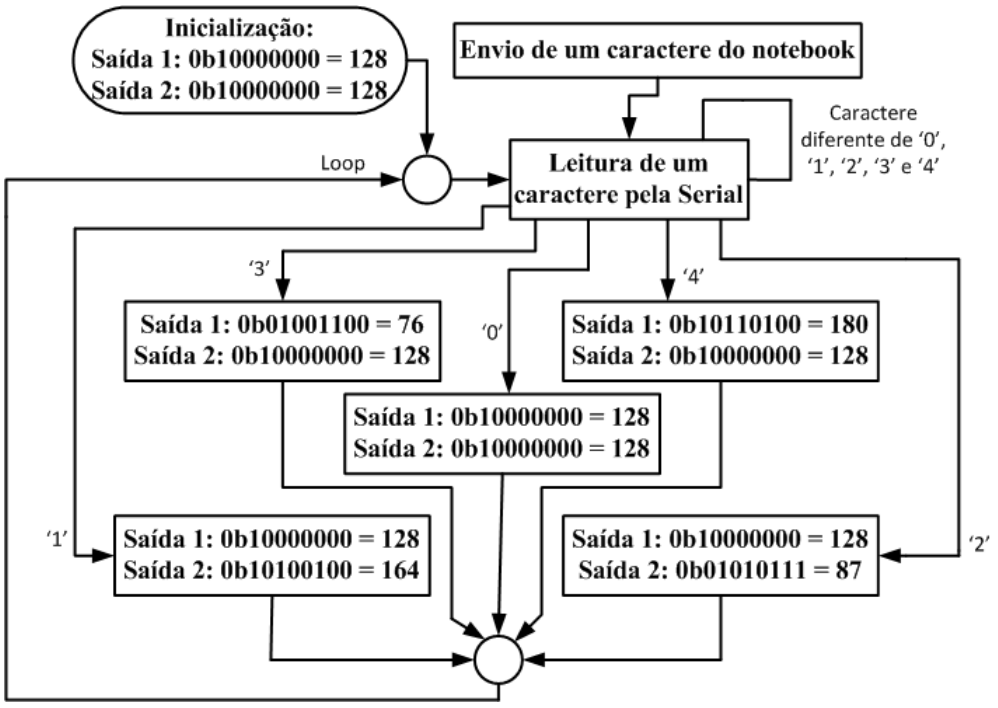

Figura 13 – Fluxograma simplificado do *firmware* implementado em Arduíno. Fonte: Autor.

No Apêndice A deste trabalho pode ser observado todo o código, desenvolvido em linguagem C, para o *firmware* do Arduíno.

A partir do *firmware* implementado para o Arduíno se comunicar com o *notebook* e enviar as sequências binárias ao circuito dos DACs, realiza-se a simulação do circuito de interface que envolve o Arduíno e os DACs. Na Figura 14 pode-se observar o esquemático construído no Proteus para a simulação do circuito de controle.

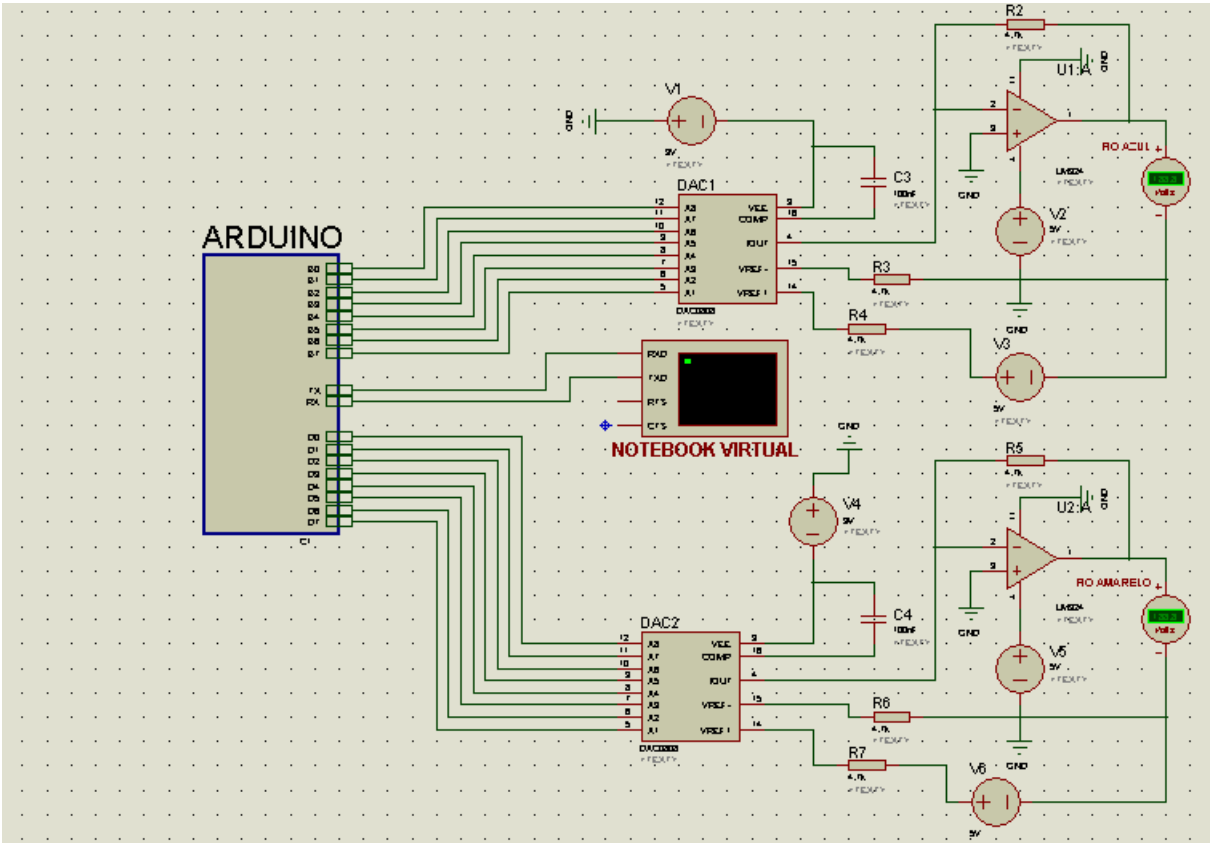

Figura 14 – Esquemático da Simulação realizada em Proteus para o circuito de controle. Fonte: Autor.

Viu-se na simulação que o circuito funcionou da forma como era previsto, enviando sinais analógicos nas saídas de acordo com o comando que era enviado do *notebook* virtual.

4 TÉCNICAS UTILIZADAS

Neste capítulo serão abordadas as técnicas que foram utilizadas no envio e leitura de dados pela porta serial do *notebook*, no reconhecimento dos comandos de voz dados pelo usuário e também nas técnicas de comunicação entre um cliente e servidor local para linguagem Java.

4.1 COMUNICAÇÃO SERIAL EM JAVA

Neste trabalho é desenvolvida uma aplicação na plataforma aberta Java para fazer a comunicação serial entre o *notebook* e o Arduíno, de acordo com os comandos de voz dados pelo usuário. Portanto, é necessário o entendimento do processo realizado para a implementação da comunicação serial em Java.

A linguagem Java possibilita o acesso às portas de comunicação serial. Dessa forma, através de uma API de comunicação serial fornecida gratuitamente, pode-se reconhecer as portas existentes na máquina e que estão disponíveis para serem abertas. A vantagem de utilizar essa API vem do fato de ser orientada a objeto, ser rapidamente assimilável e possuir ótima performance (GOMES, 2007).

O método *getPortIdentifiers* da linguagem Java retorna uma estrutura enumerada das portas disponíveis. Com isso, pode-se encontrar e abrir a porta COM disponível. Por fim, criam-se os fluxos de entrada e saída da porta serial aberta para poder enviar e receber os *bytes* através da mesma. A Figura 15 ilustra o fluxograma de um simples programa em Java para utilização de uma porta serial.

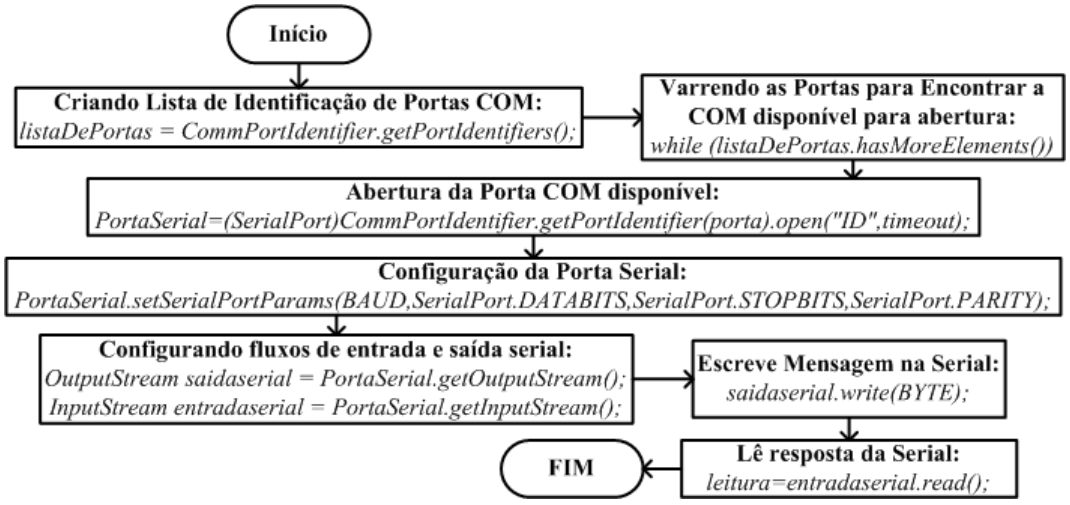

Figura 15 – Fluxograma simplificado para comunicação serial em Java. Fonte: Autor.

4.2 RECONHECIMENTO DE VOZ

Ao longo dos anos, várias pesquisas foram introduzidas e aperfeiçoadas no campo da comunicação através da fala, que resultaram na invenção do megafone, telefone, entre outros (GEVAERT; TSENOV; MLADENOV, 2010).

A constante evolução tecnológica faz do reconhecimento de voz um dos campos mais desafiadores e com diversas aplicações práticas. No final do século 18, Von Kemplen criou uma máquina que sintetizava palavras e frases. Robôs autônomos e sistemas computacionais utilizam esta técnica para navegação e conversão em tempo real de texto em fala, respectivamente. Apesar do bom progresso deste campo, ainda há muitos problemas. Variação do locutor, ruído externo e caráter contínuo do discurso são alguns dos exemplos de variáveis a serem otimizadas (GEVAERT; TSENOV; MLADENOV, 2010).

Alguns trabalhos interessantes com o reconhecimento de voz foram propostos nos últimos anos, como em (VALIATI, 2001) e (BRANDÃO, 2005). As técnicas utilizadas para o problema tem basicamente a mesma ideia, sendo um campo aberto para otimização, processamento de sinais e redes neurais artificiais.

O reconhecimento da voz é inicializado com um estágio de pré-processamento, cuja função é recortar o sinal de áudio, para atenuar os ruídos, extraindo em seguida os parâmetros da fala que são usados no reconhecimento dos padrões (VALIATI, 2001).

A extração das características é a etapa responsável por estimar parâmetros do sinal de áudio. Esses parâmetros são chamados de "características", pois representam, de forma matemática, algum aspecto da voz, ou do trato vocal, e que podem ser utilizados para o reconhecimento de palavras ou do próprio locutor (BRANDÃO, 2005). Diversas características são expressas na literatura, dentre as principais pode-se mencionar: os coeficientes do Preditor Linear (LPC), os coeficientes Cepstrais (pela Transformada de Fourier), coeficientes LPC-Cesptrais, os Mel-Cepstrais, as raízes do filtro Preditor, espectrograma e outros (RIBEIRO, 2013).

Para o reconhecimento das características extraídas, normalmente utilizam-se redes neurais treinadas para detectar os padrões de determinado sinal de áudio processado. A rede neural artificial é um modelo matemático composto a partir da conexão de blocos básicos denominados "neurônios artificiais"(RIBEIRO, 2013). As principais características desse sistema é a capacidade de:

- aprender, a partir de amostras de treinamento;
- generalizar, a partir do conhecimento adquirido;
- realizar um processamento paralelo da informação;
- produzir um mapeamento não-linear das entradas para as saídas.

De maneira geral, pode-se resumir de forma bem simplificada os passos necessários para realizar o reconhecimento de voz, como pode ser observado na Figura 16 a seguir.

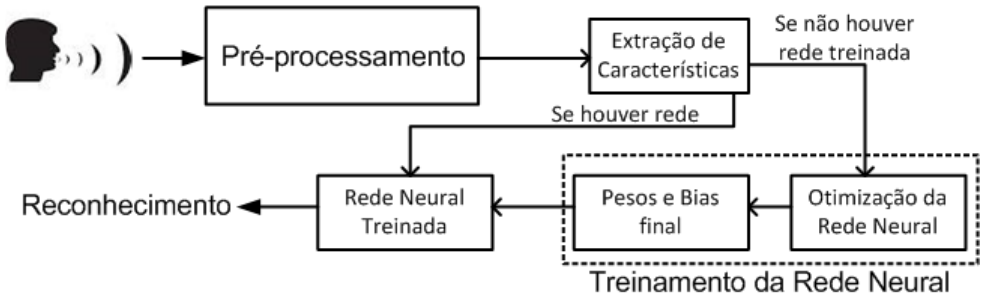

Figura 16 – Diagrama de blocos simplificado dos passos para o reconhecimento de voz. Fonte: Autor.

Já existem plataformas desenvolvidas para realizar o reconhecimento de voz com um alto grau de confiabilidade, como é o caso do *software* IBM *Via Voice* (CHIELE; ZERBETTO, 2010). Com isso, este trabalho utilizou a API (*Application Programming Interface*) *Java Speech* desenvolvida pela IBM, própria para a plataforma Java, também conhecida como IBM *Via Voice*, devido a sua confiabilidade e grande difusão no mercado, além do fato de ser uma plataforma de uso livre para trabalhos acadêmicos.

Usando essa API pode-se aproveitar as vantagens inerentes da plataforma Java, com notável economia de tempo e recursos para desenvolver aplicações de reconhecimento de voz. Vale ressaltar que a implementação da IBM do *Java Speech* está disponível no site da *IBM Alpha Works*[1] para *download* gratuito[2] (IBM, 1998).

Outra grande vantagem de utilizar essa API é que ela fornece a possibilidade de trabalhar com reconhecimento de voz no idioma português, sendo conveniente para um usuário brasileiro ao qual se destina esse trabalho. Dessa forma, basta incorporar a API em uma aplicação Java própria de forma a realizar ações de acordo com o comando de voz que é reconhecido no programa. Para isso, é necessário antes de tudo configurar a aplicação com a voz do usuário que utilizará a cadeira de rodas com comandos de voz. Essa configuração é importante pois a partir disso o programa consegue obter as características inerentes e únicas da voz do cadeirante. Uma vez realizada a configuração da voz do usuário, não será mais necessário fazê-la novamente (IBM, 1998).

Após ter sido realizada toda a configuração da API, deve-se importar a sua biblioteca para o Java através do comando *import javax.speech.**. Com isso, podem ser utilizadas as classes necessárias para o reconhecimento e sintetizador de voz, disponíveis na aplicação da IBM. Resumidamente, a inicialização da API na aplicação Java consiste em quatro passos simples:

[1] http://www.alphaworks.ibm.com/formula/speech

[2] Este software pode ser adquirido e utilizado livremente desde que os fins de uso sejam acadêmicos e não comerciais.

- Primeiramente, é criado o reconhecedor de voz em português usando o método *createRecognizer* da classe do *Java Speech*.
- O segundo passo é alocar a variável usada como reconhecedor de voz com o método *allocate*.
- Em seguida, o terceiro passo consiste em carregar o arquivo de gramática através da classe *loadJSGF*, ou seja, definir quais as palavras que devem ser reconhecidas na aplicação.
- Por último, no quarto passo adiciona-se uma escuta na variável do reconhecedor com os métodos *addResultListener* e *commitChanges*, de forma que ocorra uma interrupção na aplicação toda vez que o usuário fale alguma palavra que esteja dentro do arquivo de gramática previamente configurado.

Como o objetivo é dar comandos direcionais à cadeira, o arquivo de gramática configurado é composto de cinco palavras: "andar", "voltar", "direita", "esquerda"e "parar". Se o usuário não falar nenhuma dessas palavras o programa não realizará qualquer ação, ou seja, não enviará nenhum *byte* pela serial para o Arduíno. No entanto, se um dos cinco comandos for dito pelo usuário, a aplicação em Java reconhecerá a palavra dita, realizando a ação correspondente. Na Figura 17 a seguir pode-se observar o fluxograma simplificado para a aplicação em Java implementada para o reconhecimento dos comandos e a tomada de decisão.

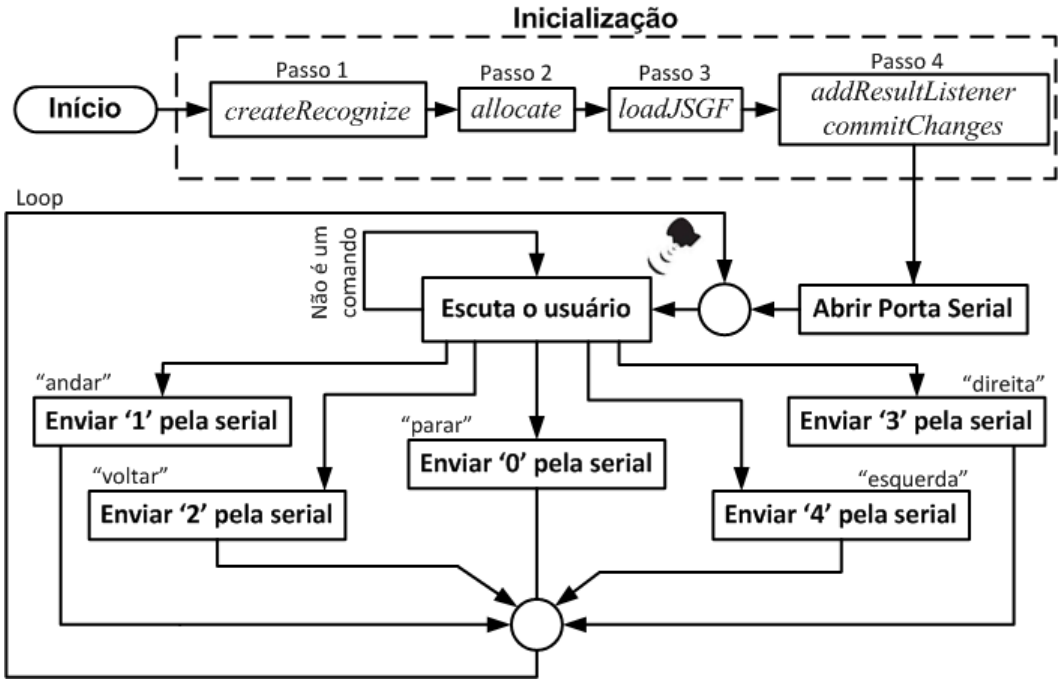

Figura 17 – Fluxograma simplificado para reconhecimento de voz e tomada de decisão na aplicação em Java. Fonte: Autor.

Portanto, cria-se uma simples aplicação em Java para o reconhecimento dos comandos vocais dados pelo usuário da cadeira, sendo enviados os caracteres específicos pela porta serial ao Arduíno previamente programado, para que com os dados enviados seja possível movimentar a cadeira de rodas como desejado. O código desenvolvido para essa aplicação em linguagem Java encontra-se disponível no Apêndice B deste trabalho.

4.3 COMUNICAÇÃO COM SERVIDOR LOCAL EM JAVA

Além do usuário poder controlar a cadeira com comandos de voz, o *software* implementado em Java também disponibiliza um servidor na rede local, para que outros dispositivos móveis, atuando como cliente e conectados em uma rede WI-FI local, possam se comunicar. Essa funcionalidade não é a principal do *software*, no entanto é interessante pois abre perspectivas de trabalhos futuros visando o controle remoto da cadeira de rodas usando *smartphones* ou *Tablets*.

Uma possível utilização de aplicativos que controlam a cadeira remotamente ocorre, por exemplo, quando o usuário está em sua residência e a cadeira encontra-se distante dele, sendo necessário que uma outra pessoa o auxilie para levar a cadeira. Dessa forma, com o controle remoto o próprio usuário poderá enviar os comandos para guiar a cadeira de rodas para próximo dele, através de um *smartphone* ou *Tablet*, garantindo maior independência.

Pode-se definir uma rede local ou LAN como sendo um sistema de comunicação de dados, com ou sem fio, entre computadores ou dispositivos em uma área geográfica delimitada (FOROUZAN, 2008). A Figura 18 ilustra os componentes básicos contidos na arquitetura da LAN sem fio IEEE 802.11 também conhecida como rede WI-FI, utilizadas na conexão de *smartphones* e *Tablets* a uma rede local.

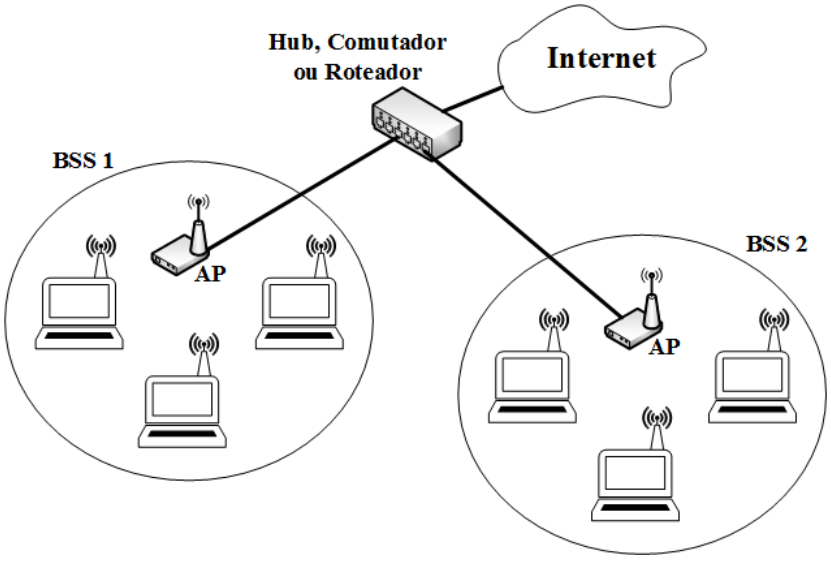

Figura 18 – Arquitetura básica da LAN IEEE 802.11. Fonte: (KUROSE; ROSSA, 2012).

Com relação à arquitetura 802.11 pode-se destacar os seguintes componentes básicos (KUROSE; ROSSA, 2012):

- BSS - Conjunto Básico de Serviço ou *Basic Service Set*;
- AP - Ponto de Acesso ou *Access Point*;
- Hub, comutador ou roteador.

Cada AP é instalado designando-se um Identificador de Conjunto de Serviços ou também denominado SSID (*Service Set Identifier*), composto por uma ou duas palavras à escolha do administrador. Além disso, o administrador deve também configurar o canal do AP de modo que não haja sobreposição nas frequências entre canais. (KUROSE; ROSSA, 2012).

Neste trabalho, o *software* desenvolvido em Java possibilita a conexão com um cliente através do protocolo TCP/IP, orientado à conexão. Assim, o servidor no *notebook* espera um dispositivo móvel (*smartphone* ou *Tablet*) se conectar a ele. Quando for estabelecida a conexão entre o servidor e cliente, ocorrem as transferências de dados através do *socket* de comunicação. A Figura 19 ilustra como o *socket* é usado dentro da comunicação entre dois computadores através de uma rede local.

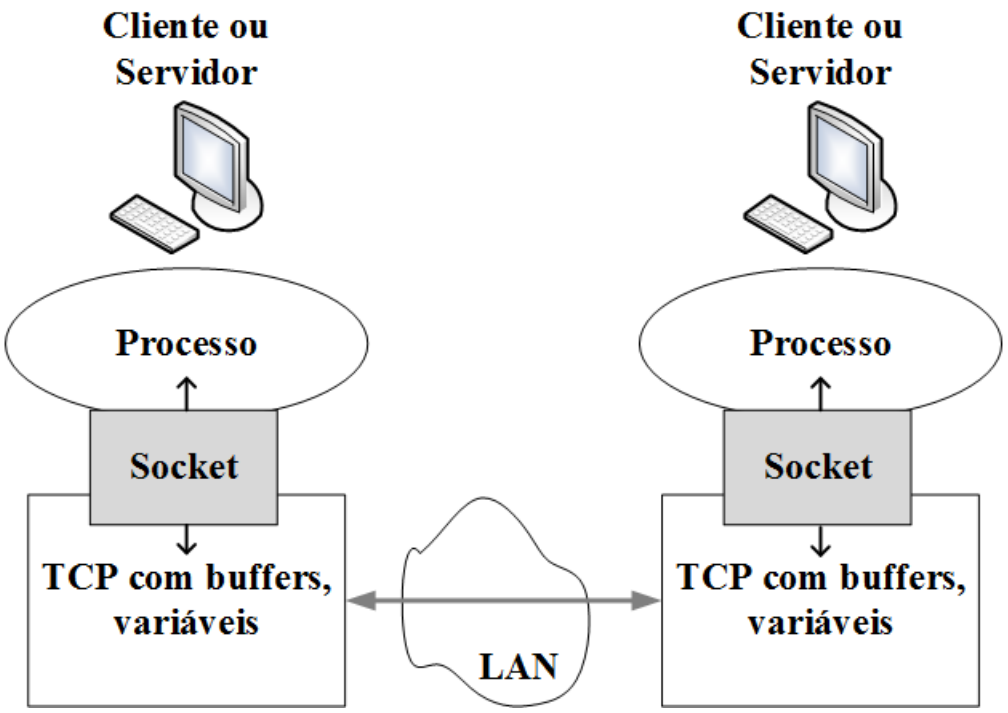

Figura 19 – Utilização do *socket* entre duas máquinas em uma LAN. Fonte: (KUROSE; ROSSA, 2012).

A linguagem Java apresenta classes com vários métodos para implementar aplicações privadas de conexão TCP/IP do tipo cliente-servidor (DEITEL; DEITEL, 2010). Essas

classes são a *java.io* (que contém métodos para cadeias de entrada e saída) e a classe *java.net* que tem métodos de suporte para rede, incluindo principalmente *Socket* e *ServerSocket* (DEITEL; DEITEL, 2010). Na Figura 20 a seguir pode-se observar um fluxograma simplificado das aplicações cliente-servidor em Java ocorrendo concomitantemente.

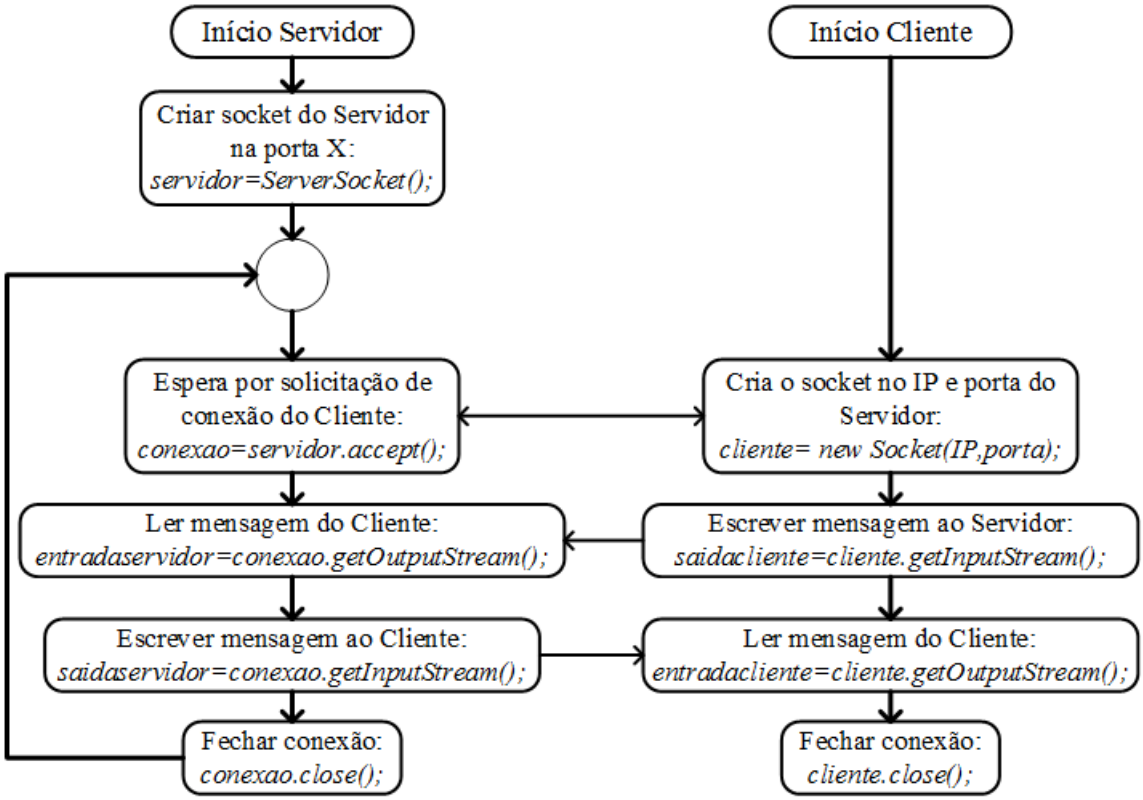

Figura 20 – Fluxograma simplificado de uma aplicação cliente-servidor em Java. Fonte: Autor.

No Apêndice C deste trabalho é mostrado o desenvolvimento de um exemplo de aplicativo para dispositivos com sistema *Android*, que possibilita o controle da cadeira de rodas remotamente através de um *smartphone* ou *Tablet*.

5 IMPLEMENTAÇÃO E RESULTADOS

Neste capítulo será visto como o foi realizada a implementação do circuito de interface da plataforma de comando por voz em *protoboard* e em PCI (Placa de Circuito Impresso), evidenciando aspectos técnicos dos esquemáticos e *layouts* e mostrando também como o circuito desenvolvido foi colocado na estrutura da cadeira de rodas para que fossem realizados os testes com o protótipo final posteriormente.

Serão mostrados os esquemáticos dos circuitos de conversão e amplificação do sinal, os *layouts* dos mesmos, as ligações dos respectivos subcircuitos com a implementação final na cadeira de rodas, os resultados obtidos para os testes do protótipo desenvolvido e por fim uma análise de custos do projeto.

5.1 ESQUEMÁTICO DO CIRCUITO

O esquemático que é desenvolvido neste trabalho consiste no circuito de interface entre o Arduíno e microcontrolador da cadeira de rodas. Dessa forma, é realizado um circuito composto por conversores digital/analógico e um amplificador operacional, necessário para o amplificação do sinal de conversão. Na Figura 21 tem-se o diagrama de blocos simplificado do circuito que é confeccionado no projeto e suas ligações entre a cadeira de rodas e o Arduíno.

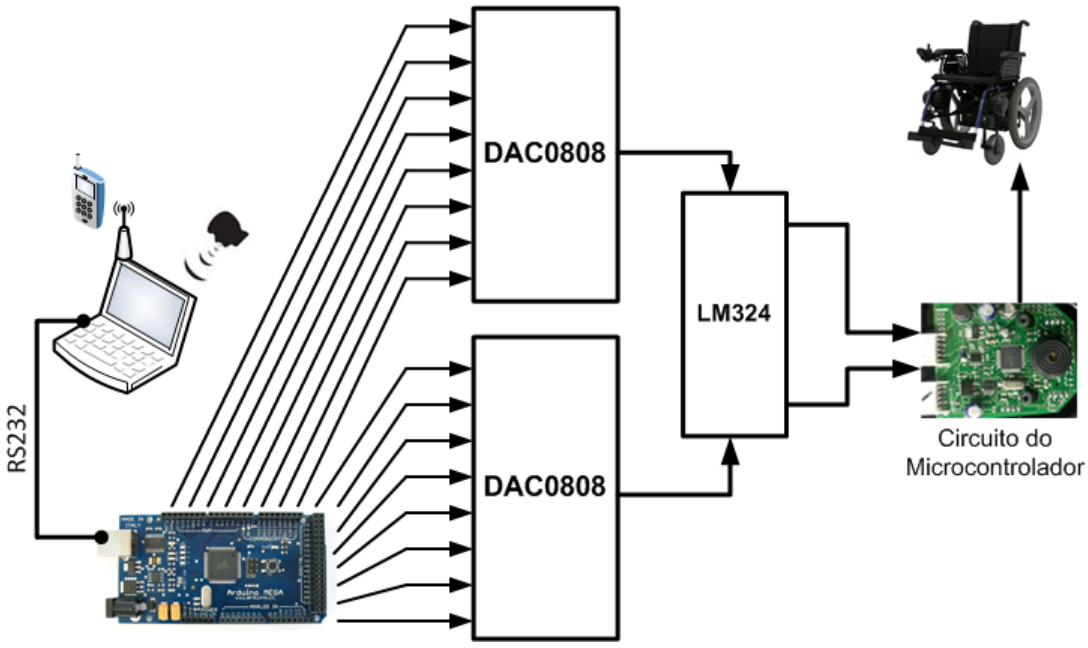

Figura 21 – Diagrama de blocos do circuito de interface entre a cadeira de rodas e o Arduíno. Fonte: Autor.

Nota-se que o circuito de interface atua realizando a conversão dos sinais digitais do Arduíno em valores analógicos colocados no circuito do microcontrolador da cadeira de rodas.

Com propósitos de simplificação, são separados três circuitos menores: dois iguais para os DACs e um para o amplificador LM324. Assim, foram feitos os esquemáticos dos circuitos menores, e posteriormente conectou-se um ao outro.

Os circuitos dos DACs, já evidenciados na Figura 12, foram implementados igualmente, visto que a única diferença entre eles está nos pontos de conexão com o Arduíno e o LM324. Na Figura 22 tem-se o esquemático implementado para o circuito dos DACs[1].

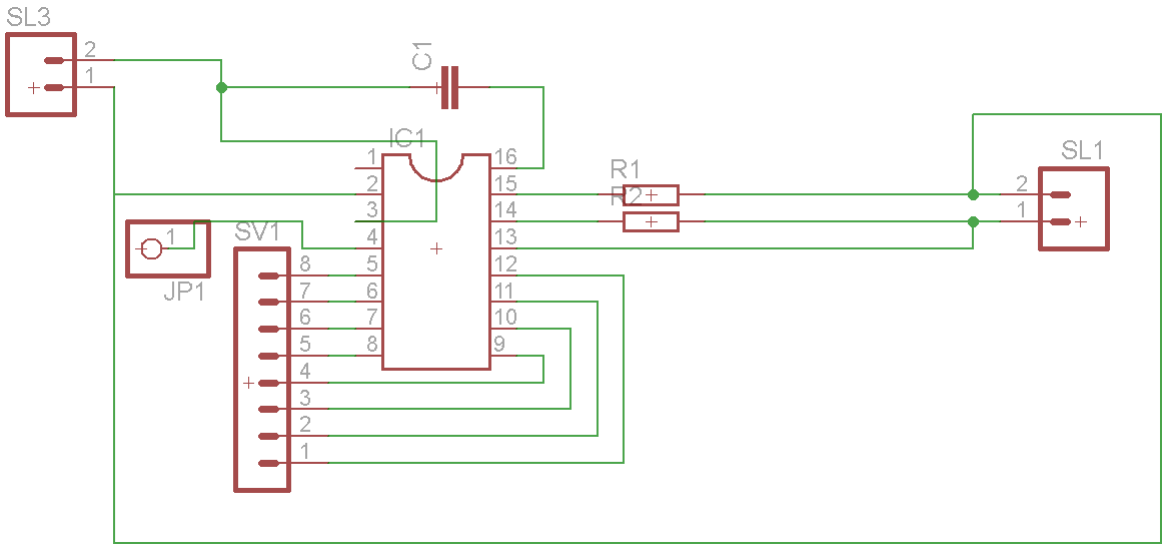

Figura 22 – Esquemático do circuito dos DACs. Fonte: Autor.

Os valores dos resistores e do capacitor já estão evidenciados na Figura 12. O conector SV1 é ligado na saída digital respectiva do Arduíno e o pino JP1 é ligado em uma das entradas do LM324. Por fim, os conectores SL1 e SL3 correspondem às entradas de alimentação de $5V$ e $9V$, respectivamente.

Como também está evidenciado na Figura 12, no circuito do amplificador operacional o LM324 foi colocado para amplificar o sinal de saída dos conversores e mandar para o circuito do microcontrolador da cadeira de rodas. Na Figura 23 a seguir pode-se observar o esquemático implementado para o circuito do LM324.

[1] Todos os esquemáticos foram feitos no *software* de desenvolvimento *Eagle*

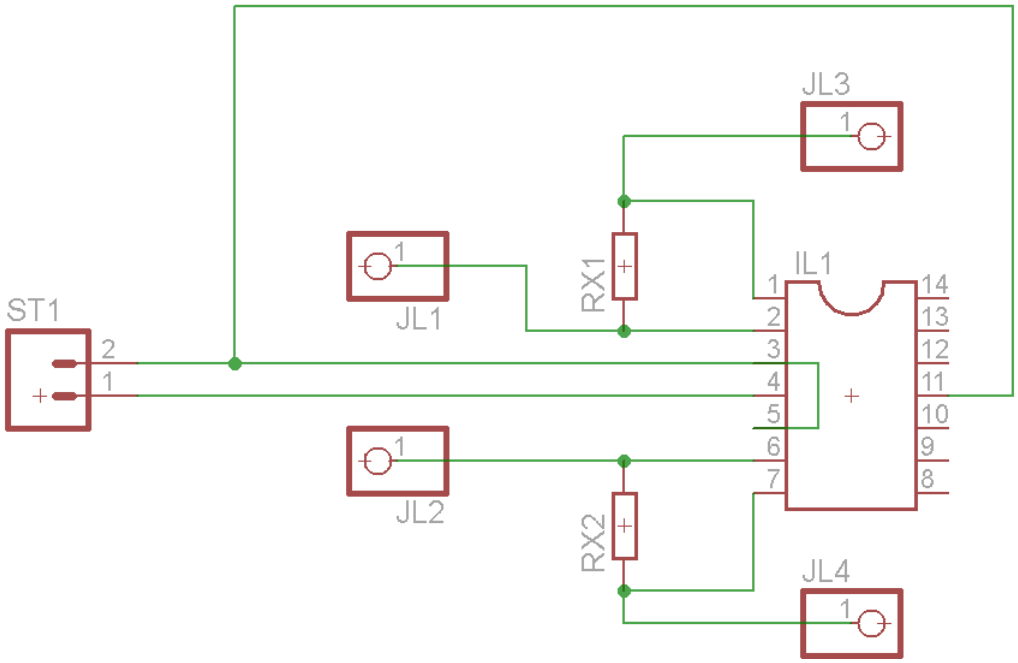

Figura 23 – Esquemático do circuito do amplificador LM324. Fonte: Autor.

Sendo os conectores JL1 e JL2 as entradas, ou seja, saídas dos DACs. Com isso, JL3 e JL4 são as saídas das entradas dos DACs em JL1 e JL2, respectivamente. Por último, o conector ST1 refere-se à alimentação de $5V$ para o circuito do LM324.

Os valores dos resistores para esse esquemático também já estão evidenciados na Figura 12, em que foi seguida a implementação típica para a utilização do DAC0808, de acordo com o fabricante (NATIONAL SEMICONDUCTOR, 1995).

É importante destacar que os circuitos evidenciados nos esquemáticos das Figuras 22 e 23 tem que ter os GNDs conectados ao GND do Arduíno e do circuito do microcontrolador da cadeira de rodas.

5.2 LAYOUT DO CIRCUITO

A partir dos esquemáticos que foram desenvolvidos para os três circuitos e que estão evidenciados nas Figuras 22 e 23, pode-se concebe o *layout* dos circuitos que serão impressos em placas de cobre[2].

Na Figura 24 a seguir pode-se observar o *layout* desenvolvido para os circuitos dos DACs, evidenciando as conexões que são feitas nos seus respectivos pinos.

[2] Todos os *layouts* também foram feitos com o *software* de desenvolvimento *Eagle*

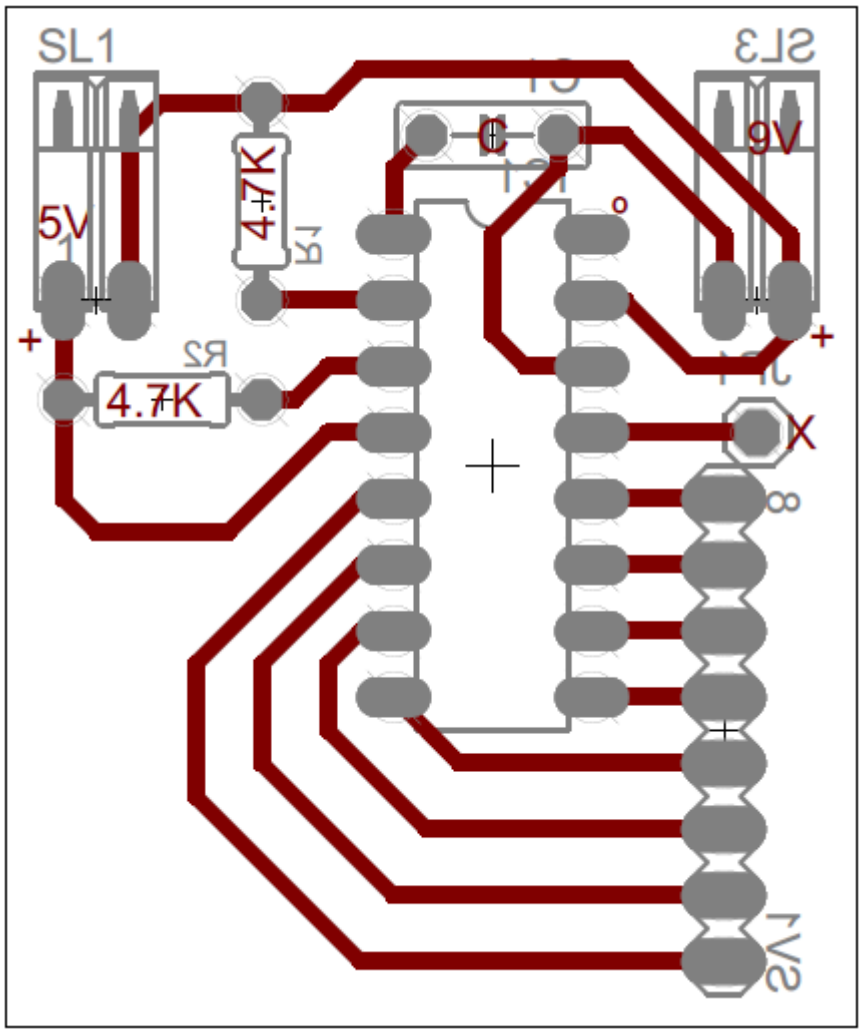

Figura 24 – *Layout* desenvolvido para os circuitos dos DACs. Fonte: Autor.

Na Figura 25 a seguir pode-se observar o *layout* desenvolvido para o circuito do amplificador LM324, que é ligado aos circuitos dos DACs e ao circuito do microcontrolador da cadeira de rodas.

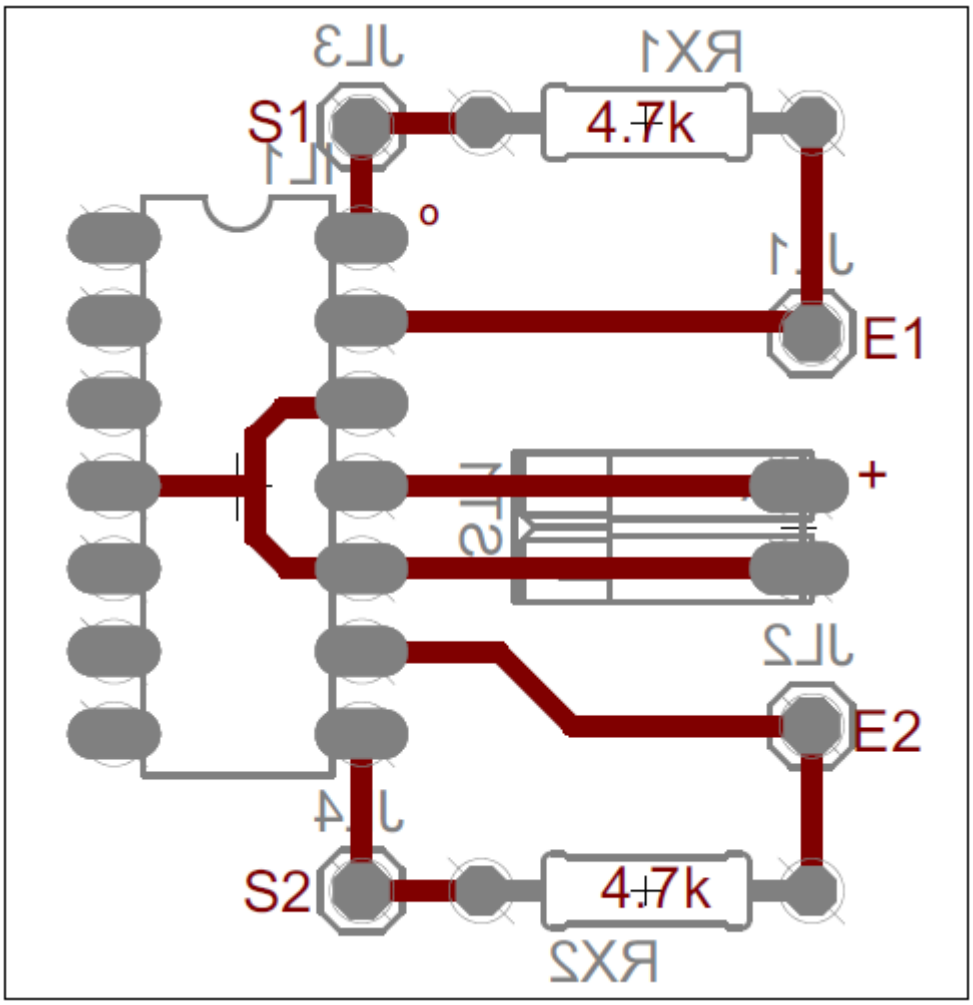

Figura 25 – *Layout* desenvolvido para o circuito do LM324. Fonte: Autor.

A partir dos *layouts* implementados para os três subcircuitos do circuito de interface, pode-se confeccionar a placa de circuito impresso e conectá-la ao Arduíno e ao microcontrolador da cadeira de rodas.

5.3 IMPLEMENTAÇÕES EM PROTOBOARD E EM PCI

Foram realizadas implementações tanto em *protoboard* quanto diretamente com as placas de circuito impresso ou PCI. Primeiramente, foi implementado o circuito em *protoboard*, seguindo exatamente os esquemáticos já evidenciados nas Figuras 22 e 23. Na Figura 26 pode-se observar o circuito implementado em *protoboard* que foi testado junto à cadeira de rodas[3].

[3] O circuito e a *protoboard* estão fixadas no encosto de braço da cadeira de rodas

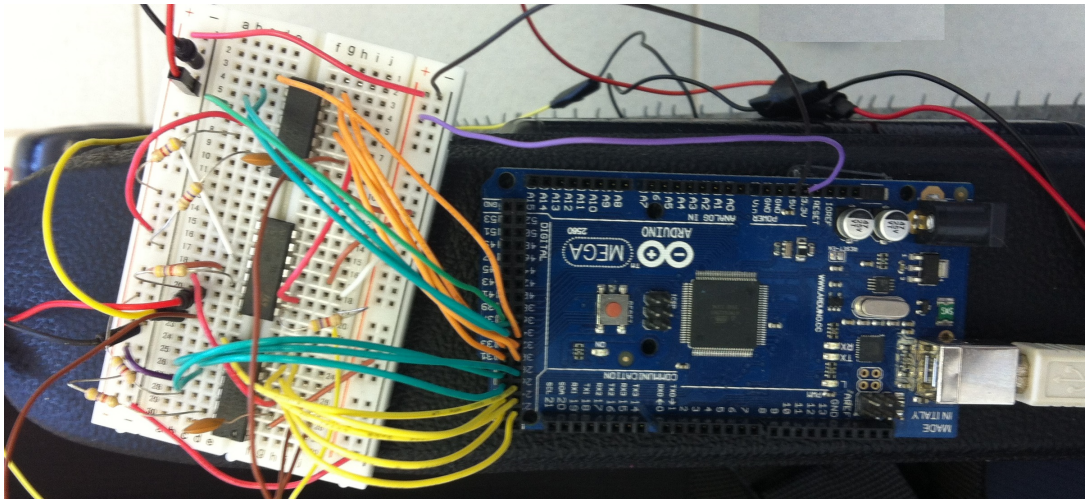

Figura 26 – Circuito de interface em *protoboard* conectado ao Arduíno e à cadeira de rodas. Fonte: Autor.

Validou-se o circuito implementado em *protoboard*, utilizando o sistema completo com todas as conexões, o que possibilitou realizar a confecção das placas de circuito impresso.

Para compreender melhor como são feitas as conexões entre as placas de circuito impresso confeccionadas para o circuito de interface, o Arduíno e o microcontrolador da cadeira de rodas, pode-se observar o esquema ilustrado na Figura 27 a seguir.

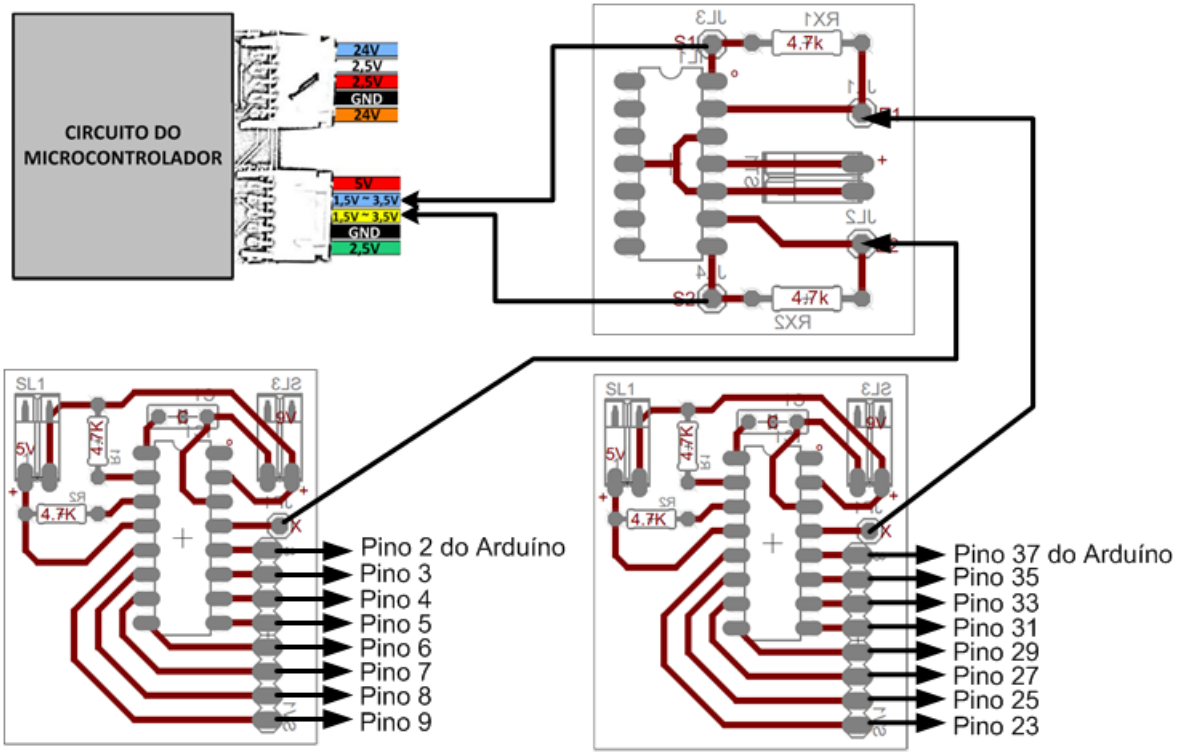

Figura 27 – Esquema de conexões entre o circuito de interface, o Arduíno e o microcontrolador da cadeira de rodas no sistema implementado. Fonte: Autor.

São confeccionadas e conectadas as três placas de circuito impresso, referentes ao circuito de interface. Na Figura 28 pode-se observar as implementações das placas de circuito impresso colocadas junto à cadeira de rodas e conectadas ao Arduíno.

Figura 28 – Placas de circuito impresso conectadas ao Arduíno junto à cadeira de rodas. Fonte: Autor.

Validaram-se as placas de circuito impresso implementadas, utilizando o sistema completo com todas as conexões juntamente com o *software* de reconhecimento de voz implementado em Java.

Por último, para garantir que os circuitos não fossem violados, os mesmos foram colocados em uma caixa de plástico, perfurada para que os fios de comunicação e sinais de controle fossem conectados à cadeira de rodas e ao *notebook*. A caixa com o circuito foi colocada na traseira da cadeira de rodas, para que não ficasse visível ao usuário, garantindo que a estética original da cadeira não fosse alterada significativamente. Na Figura 29 pode-se observar como foi colocada a caixa com o circuito de controle junto à cadeira de rodas e a aparência final com a plataforma implementada.

Figura 29 – Implementação da Caixa com o circuito de controle na cadeira de rodas e a aparência final vista pelo usuário. Fonte: Autor.

5.4 AVALIAÇÃO DOS RESULTADOS

A seguir serão mostrados os resultados obtidos nos testes do protótipo desenvolvido. Dessa forma, será visto o resultado relativo aos comandos de voz enviados à cadeira de rodas. Vale ressaltar que os resultados mostrados neste trabalho evidencia como se comportou a plataforma desenvolvida para as situações de teste que foram criadas.

Para validar o reconhecimento de voz foram feitos testes em dois diferentes ambientes: um fechado, com menos ruído, e outro aberto, com maiores ruídos sonoros. Em cada ambiente foram realizados 50 testes de maneira a observar o número de acertos que relaciona o comando de voz executado com o movimento realizado pela cadeira de rodas. Na Tabela 4 pode-se observar os resultados dos 50 testes realizados para cada comando nos dois ambientes.

Tabela 4 – Resultados dos 50 testes realizados para o comando de voz em ambiente fechado e em ambiente aberto.

Comando	**Acertos em ambiente fechado**	**Acertos em ambiente aberto**
"andar"	48	46
"voltar"	48	45
"direita"	49	47
"esquerda"	49	46
"parar"	49	48

Ocorreram pequenos atrasos no envio do comando para a cadeira de rodas, na ordem de meio segundo. Esses atrasos se devem principalmente ao *hardware*, que apresenta limitações no processamento do sinal e atrasos no envio de caracteres pela porta serial.

Foram notadas limitações de movimento da cadeira de rodas, que em alguns momentos não conseguia ir totalmente em linha reta, devido principalmente ao piso e a

outras condições dos pneus. Vale frisar também que em alguns momentos a cadeira tinha dificuldades de iniciar o movimento partindo do repouso, necessitando de mais tempo para realizar o comando solicitado.

Outra limitação notada é que com a plataforma desenvolvida ainda podem ocorrer acidentes caso o usuário esqueça de comandar a cadeira para interromper o movimento quando estiver perto de escadas ou próximo à vidraças, por exemplo. Essas limitações podem ser revertidas se forem colocados sensores na cadeira de rodas, para perceber obstáculos ou desfiladeiros, de maneira a garantir uma maior segurança em sua utilização por parte do usuário.

Vale destacar também que a plataforma de comando desenvolvida apresenta uma limitação em sua autonomia, pois necessita de uma pilha de $9V$ para operar, obrigando que o usuário troque eventualmente a bateria quando esta estiver com carga baixa.

5.5 ANÁLISE DE CUSTOS

A Tabela 5 mostra os custos de cada componente eletrônico usado no desenvolvimento do circuito de interface conectado ao Arduíno junto à cadeira de rodas, baseando-se nos valores atuais de mercado (DIGI-KEY CORPORATION, 1995).

Tabela 5 – Custos dos componentes do circuito de Interface entre Arduíno e microcontrolador da cadeira de rodas (2014).

Especificação	**Custo por unidade**	**Quantidade**	**Total**
DAC0808	US$ 1,96	2	US$ 3,92
LM324	US$ 0,49	1	US$ 0,49
Cap. $100nF$; Cerâmico	US$ 0,37	2	US$ 0,74
Res. $4,7k\Omega$; 1/4W	US$ 0,08	6	US$ 0,48
Bateria 9V	US$ 2,40	1	US$ 2,40
			US$ 5,63

Pode-se observar que os componentes do circuito de interface desenvolvido apresenta um baixo custo, de apenas US$ 5,63. Além dos componentes, é necessário também o *notebook*, Arduíno e as placas de circuito ou também chamadas de placas de fenolite, na qual o circuito de interface é confeccionado.

O Arduíno Mega tem um custo de aproximadamente US$ 52,10 (ARDUINO©, 2005) e as três placas de fenolite custam aproximadamente US$ 2,50 (DIGI-KEY CORPORATION, 1995).

Com relação ao *notebook*, é suposto que o usuário já possua um *notebook* próprio onde será colocado o *software* de reconhecimento de voz, que é livre de custos. Com isso,

o custo total da plataforma de controle de movimentos por comandos de voz será a soma do Arduíno com os componentes do circuito de interface e placas de fenolite[4].

Portanto, o custo total para o projeto será de US$ 60,23, que ainda é baixo se comparado ao custo total da cadeira de rodas motorizada[5], ou seja, a implementação dessa plataforma de controle por voz para a cadeira de rodas não constitui em um custo elevado de material ou *software*.

[4] Não foram acrescidos os valores dos fios, conectores e materiais de apoio usados na confecção das placas

[5] A cadeira usada neste trabalho tem um custo de quase R$ 9.000,00

6 CONSIDERAÇÕES FINAIS

Com esse trabalho foram gerados resultados satisfatórios na área de sistemas embarcados, programação orientada a objeto e reconhecimento de padrões. Dessa forma, nota-se que esses resultados proporcionam uma maior facilidade na mobilidade e maior conforto aos indivíduos portadores de necessidades especiais, principalmente para os tetraplégicos e para os paraplégicos com limitações severas de movimento nos braços.

O trabalho mostrou uma metodologia visando a construção de uma plataforma para controlar os movimentos de uma cadeira de rodas motorizada, especificamente da *Freedom Carbon*, através dos comandos de voz do cadeirante. Assim, usando a metodologia explicada neste trabalho, é possível reproduzir um resultado bem semelhante ao que foi obtido.

Outra característica importante deste trabalho é que ele foi todo desenvolvido usando plataformas e *softwares* de código aberto como o Arduíno e Java. Dessa forma, são reduzidos os custos de licença, que juntamente com o baixo custo de *hardware* fazem o projeto ter um baixo custo se comparado ao valor total de uma cadeira de rodas motorizada. Isso viabiliza ainda mais a incorporação dessa plataforma nas cadeiras de rodas motorizadas atuais que usam apenas o *joystick* para locomoção.

Vale destacar que o sistema desenvolvido apresentou algumas limitações não previstas, principalmente no tempo de resposta dos comandos e na autonomia da bateria, já que é necessário utilizar uma bateria de $9V$ na alimentação do circuito de interface. No entanto, essas limitações podem ser contornadas em atividades futuras visando o melhoramento da plataforma de comando desenvolvida neste trabalho.

6.1 ATIVIDADES FUTURAS DE PESQUISA

A partir deste trabalho podem ser realizadas outras atividades futuras que complementam o que já foi desenvolvido. Primeiramente serão sugeridas algumas atividades que podem ser feitas de modo a melhorar a plataforma que foi implementada, contornando os problemas de autonomia da bateria e tempo de resposta dos comandos enviados.

Para contornar o problema dos atrasos nos tempos de resposta para os comandos fornecidos, é sugerido que se realize o desenvolvimento de um *hardware* com maior capacidade de processamento. Neste caso, pode-se substituir o Arduíno por um microcontrolador dedicado e de boa capacidade como o MSP ou PIC. Outra opção também é usar um DSP, FPGA ou *Raspberry Pi* no lugar do Arduíno e do *notebook*.

Com relação ao problema de autonomia da bateria, esta pode ser substituída por um circuito conversor DC-DC. Como exemplo, tem-se o circuito integrado TC7660, que converte uma tensão de $5V$ em $-5V$ para poder alimentar os DACs. Com isso, o sistema teria uma maior autonomia, sem necessitar que fosse feita a troca periódica da bateria no circuito de interface. Para o trabalho desenvolvido isso não foi implementado principalmente devido à dificuldade de encontrar esses conversores no mercado nacional.

Para dispensar o uso da bateria, pode-se também substituir o circuito de interface que foi implementado por um filtro passa-baixas passivo composto por resistores e capacitores. Dessa forma, aplica-se na entrada do circuito do filtro um sinal PWM proveniente do Arduíno, para que sejam atenuadas as componentes de frequência mais altas, obtendo na saída um sinal composto aproximadamente da componente DC do sinal de entrada, que é a tensão desejada para aplicar no circuito do microcontrolador da cadeira de rodas.

É possível também dispensar o uso do *notebook* para fazer a captação e o reconhecimento de voz. Já existem módulos compatíveis com o Arduíno que realizam o reconhecimento dos comandos de voz. Um desses módulos, disponíveis para aquisição, é o *EasyVR Speech Recgnition* cuja versão mais atual é a 2.0, com suporte para reconhecer voz na língua portuguesa. Vale ressaltar que os custos desses módulos são bem menores que o custo de um *notebook*, o que viabiliza economicamente a sua utilização.

Pode-se usar também a plataforma implementada, incluindo principalmente o circuito de interface com o Arduíno programado, para desenvolver novos trabalhos em reconhecimento de padrões para outros tipos além da voz. Assim, pode ser implementado, por exemplo, o reconhecimento de gestos com a cabeça ou com os olhos para usuários com limitações de movimento ainda mais severas.

Um outro desdobramento para o trabalho está em incorporar sensores à cadeira de rodas, como *encoders*, câmeras e entre outros, para implementar uma cadeira de rodas capaz de tomar decisões de forma autônoma, desviando de obstáculos e evitando avançar em desfiladeiros ou escadas, o que evita acidentes e garante uma maior segurança aos usuários.

De maneira geral, o trabalho desenvolvido fornece uma gama de possibilidades de melhorias tanto em *hardware*, *software*, sensoriamento e visão computacional, objetivando sempre a melhoria na qualidade de vida dos portadores de necessidades físicas especiais, dando-lhes maior autonomia, mobilidade e conforto.

REFERÊNCIAS

ALBRECHT, B. L. *CONTROLE DE UMA CADEIRA DE RODAS MOTORIZADA ATRAVÉS DE ELETROMIOGRAFIA EM UMA PLATAFORMA EMBARCADA*. Trabalho final de Graduação — Universidade Federal do Rio Grande do Sul - UFRGS, 2012. Citado nas páginas 14 e 16.

ARDUINO©. 2005. Disponível em: <http://www.arduino.cc/>. Citado nas páginas 24 e 46.

BRANDÃO, A. S. *REDES NEURAIS ARTIFICIAIS APLICADAS AO RECONHECIMENTO DE COMANDOS DE VOZ*. Trabalho final de Graduação — Universidade Federal de Viçosa, 2005. Citado na página 32.

CANZIAN, E. *COMUNICAÇÃO SERIAL - RS232*. 2002. Disponível em: <http://www.professores.aedb.br/arlei/AEDB/Arquivos/rs232.pdf>. Citado nas páginas 25 e 26.

CHIELE, M. R.; ZERBETTO, A. DESENVOLVIMENTO DE UMA CADEIRA DE RODAS CONTROLADA POR VOZ. *XVIII Congresso Brasileiro de Automática. Bonito-MS*, 2010. Citado nas páginas 14 e 33.

DEITEL, P.; DEITEL, H. *JAVA, COMO PROGRAMAR*. [S.l.]: Prentice-Hall, 2010. Citado nas páginas 36 e 37.

DIGI-KEY CORPORATION. 1995. Disponível em: <http://www.digikey.com/>. Citado na página 46.

FERREIRA, A. et al. *CADEIRA DE RODAS ROBÓTICA COM INTERFACE DE COMUNICAÇÃO POR PDA COMANDADA POR SINAIS CEREBRAIS*. 2007. Disponível em: <http://fei.edu.br/sbai/SBAI2007/docs%5C30619_1.pdf>. Citado na página 14.

FOROUZAN, B. A. *PROTOCOLO TCP/IP*. [S.l.]: McGraw-Hill, 2008. Citado na página 35.

FREEDOM CARBON. *MANUAL DO PROPRIETÁRIO*. 2012. Disponível em: <http://www.freedom.ind.br/arquivos/produto/manual_br/cadeiras_motorizadas.pdf>. Citado nas páginas 18, 19 e 20.

GEVAERT, W.; TSENOV, G.; MLADENOV, V. NEURAL NETWORKS USED FOR SPEECH RECOGNITION. *Journal of Automatic Control. University of Belgrade*, 2010. Citado na página 32.

GOMES, D. V. *COMUNICAÇÃO SERIAL UTILIZANDO A API DA SUN*. 2007. Disponível em: <https://t71.googlecode.com/files/JavaCommAPI.pdf>. Citado na página 31.

GONZAGA, A. *COMUNICAÇÃO SERIAL*. 2005. Disponível em: <http://iris.sel.eesc.usp.br/sel337/serial.pdf>. Citado nas páginas 25 e 26.

IBM. *IBM RELEASES FIRST JAVA SPEECH API IMPLEMENTATION*. 1998. Disponível em: <https://www-03.ibm.com/press/us/en/pressrelease/2413.wss>. Citado na página 33.

KUROSE, J. F.; ROSSA, K. W. *REDES DE COMPUTADORES E A INTERNET*. [S.l.]: Pearson, 2012. Citado nas páginas 35 e 36.

NATIONAL SEMICONDUCTOR. *DATASHEET DAC0808*. 1995. Disponível em: <http://pdf.datasheetcatalog.com/datasheet/nationalsemiconduc-tor/DS005687.PDF>. Citado nas páginas 27, 28 e 40.

NETO, J. A. O.; CASTRO, M. A. A.; FELIX, L. B. RECONHECIMENTO DE COMANDOS DE VOZ PARA O ACIONAMENTO DE CADEIRA DE RODAS. *XVIII Congresso Brasileiro de Automática. Bonito-MS*, 2010. Citado na página 14.

RIBEIRO, L. S. C. *ESTUDO COMPARATIVO DE TOPOLOGIAS PARA UM SISTEMA NEURAL DE CLASSIFICAÇÃO DE GÊNEROS MUSICAIS*. Trabalho final de Graduação — Universidade Federal da Bahia - UFBA, 2013. Citado na página 32.

SIEGWART, R.; NOURBAKHSH, I. R. *INTRODUCTION TO AUTONOMOUS MOBILE ROBOTS*. [S.l.]: Massachusetts Institute of Technology, 2004. Citado na página 18.

SILVA, R. L. *DESENVOLVIMENTO DE UMA INTERFACE HOMEM-MÁQUINA APLICADA A UMA CADEIRA DE RODAS ROBÓTICA POR MEIO DE PDA*. Dissertação de Mestrado — Universidade Federal do Espírito Santo, 2007. Citado na página 14.

VALIATI, J. F. *RECONHECIMENTO DE VOZ PARA COMANDOS DE DIRECIONAMENTO POR MEIO DE REDES NEURAIS*. 2001. Disponível em: <www2.dem.inpe.br/ijar/RecVozRNeurais.doc>. Citado na página 32.

APÊNDICE A – CÓDIGO DO FIRMWARE DO ARDUÍNO

A seguir, pode-se observar o código desenvolvido em linguagem C para o *firmware* programado no Arduíno:

```
void setup() {
Serial.setTimeout((long)10000000);
Serial.begin(57600);

//Porta 1:
pinMode(37,OUTPUT); //bit mais significativo
pinMode(35,OUTPUT);
pinMode(33,OUTPUT);
pinMode(31,OUTPUT);
pinMode(29,OUTPUT);
pinMode(27,OUTPUT);
pinMode(25,OUTPUT);
pinMode(23,OUTPUT); //bit menos significativo

//Porta 2:
pinMode(2,OUTPUT); //bit mais significativo
pinMode(3,OUTPUT);
pinMode(4,OUTPUT);
pinMode(5,OUTPUT);
pinMode(6,OUTPUT);
pinMode(7,OUTPUT);
pinMode(8,OUTPUT);
pinMode(9,OUTPUT); //bit menos significativo

}

void loop() {

  int tensao11, tensao12, tensao21, tensao22;
  int referencia1, referencia2, referencia;
  int entrada;
```

```
//Porta 1:
digitalWrite(37, HIGH);
digitalWrite(35, LOW);
digitalWrite(33, LOW);
digitalWrite(31, LOW);
digitalWrite(29, LOW);    // = 128
digitalWrite(27, LOW);
digitalWrite(25, LOW);
digitalWrite(23, LOW);

//Porta 2:
digitalWrite(2, HIGH);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
digitalWrite(5, LOW);    // = 128
digitalWrite(6, LOW);
digitalWrite(7, LOW);
digitalWrite(8, LOW);
digitalWrite(9, LOW);

while(1){

  entrada=-1; //valor default da entrada

  while(entrada!=0&&entrada!=1&&entrada!=2&&
                    entrada!=3&&entrada!=4){

    entrada=Serial.read()-48;

  }

  if (entrada==0){ //Parar

      digitalWrite(2, HIGH);
      digitalWrite(3, LOW);
      digitalWrite(4, LOW);
      digitalWrite(5, LOW);
      digitalWrite(6, LOW);
```

```
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        digitalWrite(9, LOW);

        digitalWrite(37, HIGH);
        digitalWrite(35, LOW);
        digitalWrite(33, LOW);
        digitalWrite(31, LOW);
        digitalWrite(29, LOW);
        digitalWrite(27, LOW);
        digitalWrite(25, LOW);
        digitalWrite(23, LOW);
    }

    if (entrada==1) { //Frente

        digitalWrite(2, HIGH);
        digitalWrite(3, LOW);
        digitalWrite(4, LOW);
        digitalWrite(5, LOW);
        digitalWrite(6, LOW);
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        digitalWrite(9, LOW);

        digitalWrite(37, HIGH);
        digitalWrite(35, LOW);
        digitalWrite(33, HIGH);
        digitalWrite(31, LOW);
        digitalWrite(29, LOW);      // = 3.2V
        digitalWrite(27, HIGH);
        digitalWrite(25, LOW);
        digitalWrite(23, LOW);
    }

    if (entrada==2){ //Voltar

        digitalWrite(2, HIGH);
        digitalWrite(3, LOW);
```

```
        digitalWrite(4, LOW);
        digitalWrite(5, LOW);
        digitalWrite(6, LOW);
        digitalWrite(7, LOW);
        digitalWrite(8, LOW);
        digitalWrite(9, LOW);

        digitalWrite(37, LOW);
        digitalWrite(35, HIGH);
        digitalWrite(33, LOW);
        digitalWrite(31, HIGH);
        digitalWrite(29, LOW);          // =1.7 V
        digitalWrite(27, HIGH);
        digitalWrite(25, HIGH);
        digitalWrite(23, HIGH);

    }

    if (entrada==3){ //Direita

        digitalWrite(2, LOW);
        digitalWrite(3, HIGH);
        digitalWrite(4, LOW);
        digitalWrite(5, LOW);
        digitalWrite(6, HIGH);      // = 1.48V
        digitalWrite(7, HIGH);
        digitalWrite(8, LOW);
        digitalWrite(9, LOW);

        digitalWrite(37, HIGH);
        digitalWrite(35, LOW);
        digitalWrite(33, LOW);
        digitalWrite(31, LOW);
        digitalWrite(29, LOW);
        digitalWrite(27, LOW);
        digitalWrite(25, LOW);
        digitalWrite(23, LOW);

    }
```

```
    if (entrada==4){ //Esquerda

        digitalWrite(2, HIGH);
        digitalWrite(3, LOW);
        digitalWrite(4, HIGH);
        digitalWrite(5, HIGH);          // = 3.51V
        digitalWrite(6, LOW);
        digitalWrite(7, HIGH);
        digitalWrite(8, LOW);
        digitalWrite(9, LOW);

        digitalWrite(37, HIGH);
        digitalWrite(35, LOW);
        digitalWrite(33, LOW);
        digitalWrite(31, LOW);
        digitalWrite(29, LOW);
        digitalWrite(27, LOW);
        digitalWrite(25, LOW);
        digitalWrite(23, LOW);

    }
  }
}
```

APÊNDICE B – CÓDIGO DO PROGRAMA EM JAVA PARA RECONHECIMENTO DE VOZ

A seguir, pode-se observar o código desenvolvido em Java para o *software* do *notebook* realizar o reconhecimento de voz e criação do servidor local:

```
package reconhecendo;

import gnu.io.*;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Enumeration;
import java.util.Locale;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.speech.AudioException;

import javax.speech.Central;
import javax.speech.EngineException;
import javax.speech.EngineModeDesc;
import javax.speech.EngineStateError;
import javax.speech.recognition.GrammarException;
import javax.speech.recognition.Recognizer;
import javax.speech.recognition.Result;
import javax.speech.recognition.ResultAdapter;
import javax.speech.recognition.ResultEvent;
import javax.speech.recognition.ResultToken;
import javax.speech.recognition.RuleGrammar;

public class Reconhecendo extends ResultAdapter {
```

```
public static Recognizer rec;
private static int estadoandroid;
private String url;
private String gramatica;
private Command command;
static OutputStream saidaserial;
Sintetizador vp = new Sintetizador();

static ServerSocket servidor;
static Socket socket;
static int statusservidor = 0;
private static OutputStream saidaserver;
private static InputStream entradaserver;
static int cliente, comando;

public Reconhecendo(String url, String gram) {
    this.url = url;
    gramatica = gram;
}

public void setCommand(Command com) {
    command = com;
}

public Command getCommand() {
    return command;
}

public static void Falar(String textoparafalar) {

    String texto = " <JSML> "
            + "<BREAK MSECS=\"300\"/>"
            + "<PROS PITCH=\"90\" RANGE=\"50\" RATE=\"150\">"
            + textoparafalar
            + "</PROS>"
            + " </JSML> ";
    Sintetizador.speak(texto);

}
```

```
    private static void esperar() throws InterruptedException {

        Thread.sleep(150);

    }

    @Override
    public void resultAccepted(ResultEvent e) {
        Result r = (Result) (e.getSource());
        ResultToken tokens[] = r.getBestTokens();
        String frase = "";
        for (int i = 0; i < tokens.length; i++) {
            frase = frase.concat(tokens[i].getSpokenText()
                    + " ");
        }

        if (frase.equals("direita ")) {

            estadoandroid = 3;

            //envia '3'
            try {
                saidaserial.write('3');

            } catch (IOException ex) {
                System.out.println("Erro Serial: " +
                                        ex.toString());
            }

            Falar("Indo para direita.");

        }

        if (frase.equals("esquerda ")) {

            estadoandroid = 4;

            //envia '4'
```

```
            try {
                saidaserial.write('4');

            } catch (IOException ex) {
                System.out.println("Erro Serial: " +
                                        ex.toString());
            }

            Falar("Indo para esquerda.");
        }

        if (frase.equals("andar ")) {

            estadoandroid = 1;

            //envia '1'
            try {
                saidaserial.write('1');

            } catch (IOException ex) {
                System.out.println("Erro Serial: " +
                                        ex.toString());
            }

            Falar("Indo para frente.");
        }

        if (frase.equals("voltar ")) {

            estadoandroid = 2;

            //envia '2'
            try {
                saidaserial.write('2');

            } catch (IOException ex) {
                System.out.println("Erro Serial: " +
                                        ex.toString());
            }
```

```
        Falar("Indo para traz.");
    }

    if (frase.equals("parar ")) {

        estadoandroid = 0;

        //envia '0'
        try {
            saidaserial.write('0');

        } catch (IOException ex) {
            System.out.println("Erro Serial: " +
                                    ex.toString());
        }

        Falar("Movimento parado.");
    }

    if (command != null) {
        command.commandSaid(frase);
    }

    // Deallocate the recognizer and exit
    if ((tokens[0].getSpokenText()).
                            equals("terminar")) {

        try {
            rec.deallocate();
            Falar("Saindo do programa.");
            try {
                Thread.sleep(2000);
            } catch (InterruptedException ex) {
                Logger.getLogger(Reconhecendo.
                    class.getName()).
                    log(Level.SEVERE, null, ex);
            }
            System.exit(0);
```

```
            } catch (EngineException |
                        EngineStateError exp) {
                System.out.println("Erro: " +
                                exp.toString());
            }
        }
    }

    public void iniciar() {
        try {
            // Create a recognizer that supports Portuguese.
            rec = Central.createRecognizer(new EngineModeDesc(
                                        new Locale("pt", "BR")));
            System.out.println("passou 1: " + rec);
            // Start up the recognizer
            rec.allocate();
            System.out.println("passou 2:");
            // Load the grammar from a file, and enable it
            FileReader reader = new
                        FileReader("D:\\gramatica.gram");

            RuleGrammar gram = rec.loadJSGF(reader);
            //RuleGrammar gram = rec.loadJSGF(new
                        java.net.URL(url), gramatica);
            gram.setEnabled(true);
            System.out.println("passou 3:");
            // Add the listener to get results
            rec.addResultListener(this);

            // Commit the grammar
            rec.commitChanges();
            System.out.println("passou 4:");
            // Request focus and start listening
            rec.requestFocus();
            rec.resume();
        } catch (IllegalArgumentException | EngineException |
                        SecurityException | EngineStateError |
                        GrammarException | IOException |
                        AudioException e) {
```

```
            System.out.println("Reconhecedor - Error 1: "
                    + e.toString());
        }
    }

    public static void main(String args[]) throws
                        PortInUseException,
                        UnsupportedCommOperationException,
                        IOException, InterruptedException {

        String porta = null;
        CommPortIdentifier ips;
        Enumeration listaDePortas;
        SerialPort PortaSerial = null;

        Reconhecendo reconhecedor = new Reconhecendo(null,
                                        "gramatica.gram");
        reconhecedor.iniciar();

        listaDePortas = CommPortIdentifier.getPortIdentifiers();

        while (listaDePortas.hasMoreElements() &&
                                        porta == null) {
            ips = (CommPortIdentifier)
                        listaDePortas.nextElement();

            porta = ips.getName();

            if (porta.charAt(0) != 'C' ||
                        porta.charAt(1) != 'O' ||
                        porta.charAt(2) != 'M') {

                porta = null;
            }
        }

        Janela janelacontrole = new Janela(0);

        janelacontrole.setVisible(true);
```

```
try {

    PortaSerial = (SerialPort) CommPortIdentifier.
                getPortIdentifier(porta).
                open("USB-SERIAL", 1000);

    PortaSerial.setSerialPortParams(57600,
                SerialPort.DATABITS_8,
                SerialPort.STOPBITS_1,
                SerialPort.PARITY_NONE);

    PortaSerial.setFlowControlMode(SerialPort.
                                    FLOWCONTROL_NONE);

} catch (NoSuchPortException ex) {
    System.out.println("Erro ao conectar Serial: " +
                                    ex.toString());

    janelacontrole.setVisible(false);
    Janela janelaerro = new Janela(1);
    janelaerro.setVisible(true);
    Thread.sleep(3000);
    System.exit(0);
}

saidaserial = PortaSerial.getOutputStream();

saidaserial.flush();

while (true) {

    if (statusservidor == 0) {
        try {
            servidor = new ServerSocket(1200);
            servidor.setSoTimeout(200);
            socket = servidor.accept();
```

```
            statusservidor = 1;

        } catch (IOException ex) {
            if (statusservidor == 0) {
                servidor.close();
            }
        }
    }

    if (statusservidor == 1) {

        saidaserver = socket.getOutputStream();

        entradaserver = socket.getInputStream();

        statusservidor = 2;
    }

    if (statusservidor == 2) {

        cliente = entradaserver.read();

        esperar();

        if (cliente == '1') {

            if (estadoandroid == 0) { //Para
                saidaserver.write('0');
            }

            if (estadoandroid == 1) { //Frente
                saidaserver.write('1');
            }

            if (estadoandroid == 2) { //Voltar
                saidaserver.write('2');
            }

            if (estadoandroid == 3) { //Direita
```

```
                saidaserver.write('3');
            }

            if (estadoandroid == 4) { //Esquerda
                saidaserver.write('4');
            }

        }

        if (cliente == '2') {

            comando = entradaserver.read();

            if (comando == '0') { //Para
                saidaserial.write('0');
                estadoandroid=0;
            }

            if (comando == '1') { //Frente
                saidaserial.write('1');
                estadoandroid=1;
            }

            if (comando == '2') { //Voltar
                saidaserial.write('2');
                estadoandroid=2;
            }

            if (comando == '3') { //Direita
                saidaserial.write('3');
                estadoandroid=3;
            }

            if (comando == '4') { //Esquerda
                saidaserial.write('4');
                estadoandroid=4;
            }

        }
```

```
                if (cliente == '3') {
                    socket.close();
                    statusservidor = 0;
                }
            }
        }
    }
}
```

APÊNDICE C – DESENVOLVIMENTO DE APLICATIVO PARA O CONTROLE REMOTO DA CADEIRA DE RODAS

Neste apêndice será mostrado o desenvolvimento de um aplicativo para dispositivos móveis com sistema *Android* que, através de uma rede WI-FI local, realiza a comunicação com o *notebook* para que o usuário possa também controlar a cadeira de rodas remotamente. Naturalmente que no desenvolvimento de aplicativos para dispositivos com sistema *Android* usa-se a linguagem Java.

Para a aplicação desenvolvida, o cliente solicita ao servidor, enviando o caractere '1', o estado no qual se encontra a cadeira de rodas, ou seja, qual o movimento ela está executando ou se ela está parada. Dessa forma, o servidor deve enviar os caracteres de '0' a '4' para informar o estado ao cliente.

Para o cliente enviar um comando direcional à cadeira de rodas, deve-se acionar um dos botões da janela de interface do aplicativo. Com isso, ao acionar um desses botões de interface, primeiramente envia-se '2' ao servidor, para o mesmo saber que o cliente quer realizar um comando e não a leitura do estado da cadeira. Em seguida, o cliente envia ao servidor os caracteres de '0' a '4' pela rede WI-FI local, informando o comando correspondente do botão acionado. Assim, o *software* do *notebook* ao ler este comando irá enviar ao Arduíno o caractere correspondente para executar o movimento na cadeira de rodas.

O *software* no *notebook* funciona como uma espécie de ponte de comunicação entre o dispositivo móvel e o circuito de interface com o Arduíno. Na Figura 30 pode-se observar o fluxograma simplificado da aplicação desenvolvida para dispositivos móveis com sistema *Android*, no intuito de controlar a cadeira de rodas remotamente.

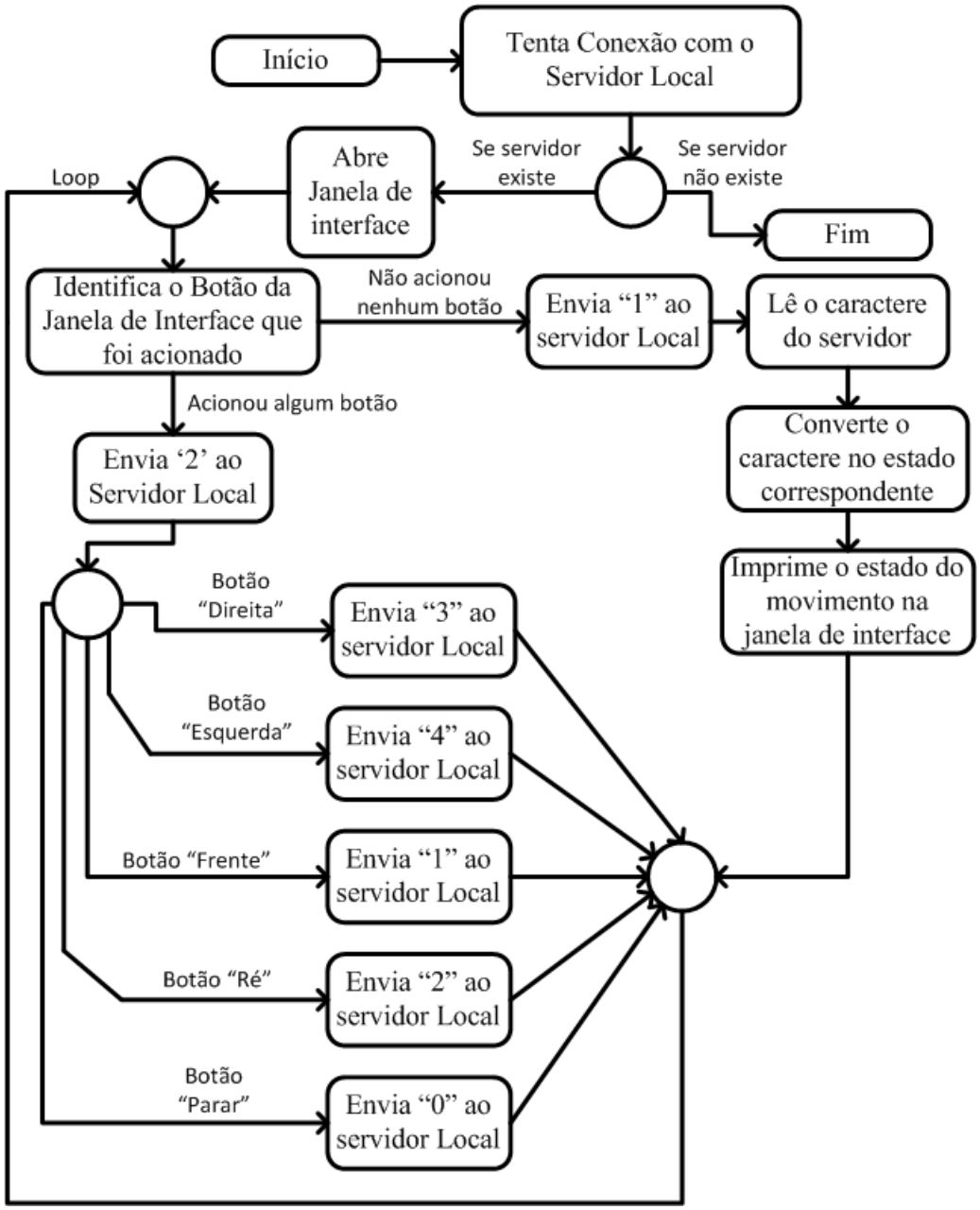

Figura 30 – Fluxograma simplificado da aplicação desenvolvida para dispositivos móveis com sistema *Android*. Fonte: Autor.

Desenvolve-se uma IHM para a aplicação em sistema *Android*, de maneira a possibilitar fácil utilização por parte do usuário. Com isso, a ideia é que através do acionamento de simples botões direcionais na janela de interface do aplicativo, o usuário escolha qual a direção desejada para o movimento. Na Figura 31 pode-se observar a tela de interface do aplicativo desenvolvido para os dispositivos móveis com sistema *Android*.

Figura 31 – Tela de Interface do Aplicativo para dispositivos móveis com sistema *Android*. Fonte: Autor.

Quando for acionado algum movimento com a cadeira de rodas, o texto que está acima da figura da cadeira na janela de interface mudará, evidenciando para o usuário em qual estado de movimentação a cadeira se encontra.

Para testar essa outra possível funcionalidade da plataforma, mandando comandos para a cadeira de rodas remotamente através de um dispositivo móvel, foi realizada uma metodologia de testes parecida com o que foi feito para obter os resultados com relação aos comandos de voz. Dessa forma, testou-se o acionamento das cinco ações possíveis: "andar", "voltar", "direita", "esquerda"e "parar". Como observado na Figura 31, tem-se uma tela de interface com o usuário para possibilitar que o mesmo controle a cadeira de rodas pelo *smartphone* ou *Tablet* apenas clicando em um dos cinco botões da tela, de acordo com a ação de movimento desejada.

Os resultados do movimento da cadeira de rodas correspondeu sempre ao botão clicado na tela de interface do aplicativo, como já era previsto. No entanto, essa possibilidade de movimentação tem a limitação de ter que utilizar uma rede WI-FI local em que o *notebook* também esteja conectado. Além disso, os tempos de resposta para os

comandos enviados do dispositivo móvel são um pouco mais lentos que os tempos dados pelos comandos vocais, já que para o dispositivo móvel existe, além do tempo de resposta do *hardware*, o tempo de envio pela rede até chegar ao *notebook*. Constatou-se uma demora de aproximadamente 1 segundo entre clicar no botão do aplicativo e a cadeira obedecer o comando enviado.

A seguir, pode-se observar o código desenvolvido em Java para o exemplo de aplicativo desenvolvido:

```
package com.exercice.controlecadeira;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

import android.os.Bundle;
import android.app.Activity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;

public class CadeiraAndroid extends Activity {

        Socket socket = null;
        DataOutputStream dadosdesaida = null;
        DataInputStream dadosdeentrada = null;
        private static String Ip;
        private static int conectado=0, estado=-1;
        private static int comando;

        TextView infoTela1;
        EditText IP;
        Button Conectar;
        Button FecharTela1;

        Button FecharTela2;
        Button BotaoDIREITA;
        Button BotaoESQUERDA;
```

```
Button BotaoFRENTE;
Button BotaoRE;
Button BotaoPARAR;
TextView InfoMovimento;

Runnable execucao = new Runnable() {
    @Override
    public void run() {
        while(true){
            if (estado==-1&&conectado==1){
                    try {
                            socket = new Socket(Ip, 1200);
                        } catch
                         (UnknownHostException e) {
                                e.printStackTrace();
                        } catch (IOException e) {
                                e.printStackTrace();
                        }

                    try {
                                dadosdesaida = new
                                DataOutputStream
                                (socket.
                                getOutputStream());

                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                    try {
                                dadosdeentrada = new
                                DataInputStream
                                (socket.
                                getInputStream());

                        } catch (IOException e) {
                                e.printStackTrace();
                        }
                    estado=0;
                }
```

```
if (estado==0){

        try {
              esperar();
                } catch (
                  InterruptedException e) {
                      e.printStackTrace();
                }

        try {
                   dadosdesaida.write('1');
                }  catch (IOException e) {
                      e.printStackTrace();
                }

        try {
              comando=dadosdeentrada.read();
                }  catch (IOException e) {
                      e.printStackTrace();
                }

        if (comando=='0'){ //Para
              InfoMovimento.
              post(new Runnable(){

              @Override
              public void run() {
              InfoMovimento.
              setText
              ("A cadeira esta PARADA");

              }
        });

        }

        if (comando=='1'){ //Frente
              InfoMovimento.
```

```
                post(new Runnable(){

                @Override
                public void run() {
                InfoMovimento.
                setText
                ("Movimento para FRENTE");

                }
        });
        }

        if (comando=='2'){ //Voltar
                InfoMovimento.
                post(new Runnable(){

                @Override
                public void run() {
                InfoMovimento.
                setText("Voltando");
                }
        });
        }

        if (comando=='3'){ //Direita
                InfoMovimento.
                post(new Runnable(){

                @Override
                public void run() {
                InfoMovimento.
                setText
                ("Indo para a DIREITA");
                }
        });
        }

        if (comando=='4'){ //Esquerda
                InfoMovimento.
```

```
                    post(new Runnable(){

                    @Override
                    public void run() {
                    InfoMovimento.
                    setText
                    ("Indo para a ESQUERDA");
                    }
                });
                }

        }

        if (estado!=0&&estado!=-2&&estado!=-1){

                try {
                    esperar();
                } catch (
                    InterruptedException e) {
                        e.printStackTrace();
                }

                try {
                    dadosdesaida.write('2');
                } catch (IOException e) {
                        e.printStackTrace();
                }

                try {
                    esperar();
                } catch (
                    InterruptedException e) {

                    e.printStackTrace();
                }

                if (estado==1){ // Direita
                    try {
                    dadosdesaida.write('3');
```

```
        } catch
          (IOException e) {
          e.printStackTrace();
        }
    }

    if (estado==2){ // Esquerda

          try {
                dadosdesaida.
                write('4');
                } catch (
                IOException e) {
                e.printStackTrace();
          }
    }

    if (estado==3){ // Frente
          try {
                dadosdesaida.
                write('1');
                } catch (
                IOException e) {
                e.printStackTrace();
                }
    }

    if (estado==4){ // Voltar
          try {
                dadosdesaida.
                write('2');
                } catch (
                IOException e) {
                e.printStackTrace();
                }
    }

    if (estado==5){ // Parar
          try {
```

```
                                        dadosdesaida.
                                        write('0');
                                        } catch (
                                        IOException e) {
                                        e.printStackTrace();
                                        }
                        }

                        estado=0;
                }

                if (estado==-2){

                        try {
                                        esperar();
                                } catch (
                                  InterruptedException e) {
                                        e.printStackTrace();
                                }

                        try {
                                        dadosdesaida.
                                        write('3');
                                } catch (IOException e) {
                                        e.printStackTrace();
                                }
                        System.exit(0);
                }

        }
    }
};

private static void esperar()throws
                InterruptedException{

        Thread.sleep(150);

}
```

```
Button.OnClickListener acaobotaofecharprograma1 =
                            new Button.OnClickListener(){
@Override
        public void onClick(View arg0) {
        System.exit(0); //encerra o aplicativo
        }
};

Button.OnClickListener acaobotaofecharprograma2 =
                            new Button.OnClickListener(){
@Override
        public void onClick(View arg0) {
                 estado=-2;
        }
};

Button.OnClickListener acaobotaoconectar =
                        new Button.OnClickListener(){
        @Override
        public void onClick(View arg0) {
                Ip=IP.getText().toString();
                conectado=1;

                setContentView(R.layout.activity_controle);

                InfoMovimento=(TextView)findViewById(
                            R.id.InformacaoMovimento);

                FecharTela2=(Button)findViewById(
                            R.id.fecharprograma2);
                FecharTela2.setOnClickListener(
                            acaobotaofecharprograma2);

                BotaoDIREITA=(Button)findViewById(
                            R.id.botaodireita);
                BotaoDIREITA.setOnClickListener(
                            acaobotaodireita);
```

```
                BotaoESQUERDA=(Button)findViewById(
                        R.id.botaoesquerda);
                BotaoESQUERDA.setOnClickListener(
                        acaobotaoesquerda);

                BotaoFRENTE=(Button)findViewById(
                        R.id.botaofrente);
                BotaoFRENTE.setOnClickListener(
                        acaobotaofrente);

                BotaoRE=(Button)findViewById(
                        R.id.botaore);
                BotaoRE.setOnClickListener(
                        acaobotaore);

                BotaoPARAR=(Button)findViewById(
                        R.id.botaopara);
                BotaoPARAR.setOnClickListener(
                        acaobotaopara);

        }
};

Button.OnClickListener acaobotaodireita =
                new Button.OnClickListener(){
@Override
        public void onClick(View arg0) {
                estado=1;
        }
};

Button.OnClickListener acaobotaoesquerda =
                new Button.OnClickListener(){
@Override
        public void onClick(View arg0) {
                estado=2;
        }
};
```

```
Button.OnClickListener acaobotaofrente =
                    new Button.OnClickListener(){
@Override
        public void onClick(View arg0) {
                estado=3;
        }
};

Button.OnClickListener acaobotaore =
                    new Button.OnClickListener(){
@Override
        public void onClick(View arg0) {
                estado=4;
        }
};

Button.OnClickListener acaobotaopara =
                    new Button.OnClickListener(){
@Override
        public void onClick(View arg0) {
                estado=5;
        }
};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.activity_inicio);

            FecharTela1 = (Button)findViewById(
                            R.id.fecharprograma);
            FecharTela1.setOnClickListener(
                    acaobotaofecharprograma1);

            IP = (EditText)findViewById(R.id.ip);
            IP.setText("10.0.2.2");

            Conectar = (Button)findViewById(
                            R.id.conectar);
```

```
                Conectar.setOnClickListener(
                                acaobotaoconectar);

                Thread Execucao = new Thread(execucao);
                Execucao.start();
        }

}
```