ANDRÉ DONIZETE DE SOUZA PINTO
HILTON APARECIDO DA SILVA
HUGO MACEDO OSAWA

HANDTALKS!

Comunicação entre Deficientes Fono-Auditivos e Leigos

Monografia apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção de grau de Bacharel em Engenharia Elétrica com ênfase em Computação sob a orientação do Prof. Dr. Jorge Kinoshita.

SÃO PAULO
2005

AUTORIZAMOS A REPRODUÇÃO E DIVULGAÇÃO TOTAL OU PARCIAL DESTE TRABALHO, POR QUALQUER MEIO CONVENCIONAL OU ELETRÔNICO, PARA FINS DE PESQUISA, DESDE QUE CITADA A FONTE.

Catalogação na Publicação
Escola Politécnica da Universidade de São Paulo

Osawa, Hugo Macedo; Pinto, André D. Souza; Silva, Hilton Aparecido.
HANDTALKS! Comunicação entre Deficientes Fono-Auditivos e Leigos / Hugo Macedo Osawa; André D. Souza Pinto; Hilton Aparecido da Silva; orientador Jorge Kinoshita.
– São Paulo, 2005
120 f.

Projeto de Formatura (Graduação em Engenharia Elétrica da Computação) – Escola Politécnica da Universidade de São Paulo

FOLHA DE APROVAÇÃO

Monografia apresentado à Escola Politécnica da Universidade de São Paulo para a obtenção de grau de Bacharel em Engenharia Elétrica com ênfase em Computação

Banca Examinadora

Profº Dr.: _____

Instituição: _____ Assinatura: _____

Profº Dr.: _____

Instituição: _____ Assinatura: _____

Profº Dr.: _____

Instituição: _____ Assinatura: _____

Profº Dr.: _____

Instituição: _____ Assinatura: _____

Profº Dr.: _____

Instituição: _____ Assinatura: _____

Profº Dr.: _____

Instituição: _____ Assinatura: _____

Dedicamos a nossos pais, companheiras, professores e amigos pelos apoios e incentivos constantes.

Agradecemos ao nosso orientador por acreditar, incentivar e apoiar nosso projeto.

"Ninguém comete erro maior do que não fazer nada porque só pode fazer um pouco".

Edmund Burke

"Quem mal lê, mal ouve, mal fala, mal vê".

Monteiro Lobato

RESUMO

Osawa, H. M.; Pinto, A. D. S.; Silva, H. A. **HandTalks! Comunicação entre Deficientes Fono-Auditivos e Leigos. 2005. 120f. Projeto de Formatura – Escola Politécnica, Universidade de São Paulo, São Paulo, 2005.**

O "HandTalks!" tem como objetivo principal permitir que pessoas com deficiência fono-auditiva comuniquem-se com pessoas normais (poucos não-deficientes conhecem a linguagem surdo-muda) explorando a grande adaptabilidade do corpo humano em situações de limitações físicas.

O aparelho é baseado em um sistema simples de conversão de movimentos dos dedos das mãos, e das próprias mãos, em palavras sonoras ou textuais (exibidas em displays para *feedback* ao usuário do sistema) através de uma luva com sensores ópticos e de contato.

Como a complexidade dos gestos e movimentos da linguagem é alta, o "HandTalks!" inicialmente faria apenas a conversão do alfabeto da Linguagem Brasileira de Sinais (LIBRAS).

Palavras-Chave: Surdo. Mudo. Deficiência. Luva. Sensores. HandTalks. Gestos.

ABSTRACT

Osawa, H. M.; Pinto, A. D. S.; Silva, H. A. **HandTalks! Communication between Deaf People and Laics. 2005. 120f. Graduation Project – Escola Politécnica, Universidade de São Paulo, São Paulo, 2005.**

"HandTalks!" has, as main objective, to allow deaf people to talk to non-deaf people (few non-deaf people knows deaf language) exploring the huge adaptability of human body to physical limitations.
Device is based on a simple system that converts finger's movement into sound words or texts (showed in a display as a feedback to deaf user) through a glove with optical sensors and contacts.
As deaf gestures and movements are quite complex, HandTalks!, initially, will concentrate on translate the alphabet of Signals Brazilian Language (Linguagem Brasileira de Sinais, LIBRAS)

Keywords: Deaf. Deficient. Glove. Sensor. HandTalks. Gesture.

Lista de Ilustrações

Lista de Tabelas

Definições, Siglas e Abreviaturas

LIBRAS: Língua Brasileira de Sinais, é a língua de sinal oficial utilizada no Brasil.

GUI: *Grafical User Interface* ou Interface Gráfica com o Usuário, é a parte visual de um software, exibido como uma janela.

pyMedia: Biblioteca multiplataforma responsável pela reprodução de arquivos de áudio de diversos formatos.

pySerial: Biblioteca multiplataforma utilizada para comunicação serial (RS-232).

Python: Linguagem de programação interpretada, interativa, dinamicamente tipada e orientada a objetos. É a linguagem utilizada para o desenvolvimento do software HandTalks!.

Usuário Emissor: Usuário que, vestindo a luva HandTalks!, comunica-se através do alfabeto LIBRAS. Este usuário é também responsável pelo ajuste do software HandTalks!.

Usuário Receptor: Usuário que recebe as informações geradas pelo software HandTalks!.

wxGlade: Ferramenta para construção de GUI que utiliza a biblioteca wxPython.

wxPython: Biblioteca multiplataforma para o desenvolvimento de interfaces gráficas que adquirem a aparência nativa do ambiente *desktop* onde o sistema é executado.

Sumário

0. Revisões

02/12/2005 – Versão Final.

06/07/2005 – Proposta Final de Implementação

20/05/2005 – Especificação Completa.

16/05/2005 – Versão Inicial Incompleta.

1. Introdução

1.1 Objetivos

Portadores de deficiência fono-auditiva são, em muitos aspectos, marginalizados pela sociedade, não tendo a possibilidade de um convívio social pleno. Diversos esforços têm sido aplicados na tentativa de reduzir esse distanciamento, incluindo um recente decreto governamental para regulamentação em lei da Língua Brasileira de Sinais (LIBRAS), como relata o trecho abaixo:

LEI N.º 10.436 de 24 de abril de 2002

Dispõe sobre a Língua Brasileira de Sinais - Libras e dá outras providências.

O PRESIDENTE DA REPÚBLICA

Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte Lei:

Art. 1º É reconhecida como meio legal de comunicação e expressão a Língua Brasileira de Sinais - Libras e outros recursos de expressão a ela associados.

Parágrafo único. Entende-se como Língua Brasileira de Sinais - Libras a forma de comunicação e expressão, em que o sistema lingüístico de natureza visual-motora, com estrutura gramatical própria, constituem um sistema lingüístico de transmissão de idéias e fatos, oriundos de comunidades de pessoas surdas do Brasil.

Art. 2º Deve ser garantido, por parte do poder público em geral e empresas concessionárias de serviços públicos, formas institucionalizadas de apoiar o uso e difusão da Língua Brasileira de Sinais - Libras como meio de comunicação objetiva e de utilização corrente das comunidades surdas do Brasil.

Art. 3º As instituições públicas e empresas concessionárias de serviços públicos de assistência à saúde devem garantir atendimento e tratamento adequado aos portadores de deficiência auditiva, de acordo com as normas legais em vigor.

Art. 4º O sistema educacional federal e os sistemas e ducacionais estaduais, municipais e do Distrito Federal devem garantir a inclusão nos cursos de formação de Educação Especial, de Fonoaudiologia e de Magistério, em seus níveis médio e superior, do ensino da Língua Brasileira de Sinais - Libras, como parte integrante dos Parâmetros Curriculares Nacionais - PCNs, conforme legislação vigente.

> Parágrafo único. A Língua Brasileira de Sinais - Libras não poderá substituir a modalidade escrita da língua portuguesa.
> Art. 5º Esta Lei entra em vigor na data de sua publicação.
> Brasília, 24 de abril de 2002; 181º da Independência e 114º da República.
> FERNANDO HENRIQUE CARDOSO
> Paulo Renato Souza
>
> *(Extraído de http://portal.mec.gov.br/seesp/arquivos/pdf/lei10436.pdf)*

O projeto do Handtalks tem por objetivo permitir a comunicação entre deficientes fono-auditivos e leigos através de uma linguagem de sinais padronizada traduzida por uma luva com sensores que detectam movimentos da mão, já que são poucos os não-deficientes que conhecem a linguagem de sinais e a comunicação entre essas pessoas é feita na forma de gestos não padronizados ou textos escritos. A comunicação de pessoas normais para deficientes é feita através da leitura de lábios por aqueles que apresentam a limitação auditiva.

Deve-se ressaltar que devido à complexidade da linguagem de libras (que envolve não apenas o movimento dos dedos como também das mãos, braços em vários graus de liberdade e também do uso de expressão facial) e a falta de tempo para a completa implementação do projeto, o mesmo se limitará a fazer uso de um sub-grupo dos sinais, incluindo letras do alfabeto, números e, dependendo da viabilidade, alguns sinais não padronizados para palavras inteiras.

O objetivo principal do projeto é ser um ponto de partida para outros projetos que completem e complementem a idéia de se criar um tradutor que seja usual.

1.2 *Motivação*

Como são poucos os não-deficientes que conhecem a linguagem surdo-muda e a comunicação entre essas pessoas (surdo ↔ leigo) é feita na forma de gestos não padronizados ou textos escritos. Há uma demanda, principalmente na área da educação por intérpretes que conheçam a LIBRAS para permitir o acesso de pessoas com necessidades especiais ao ensino, desde o fundamental até o superior. No trecho abaixo, Myrna Salerno Monteiro, uma batalhadora pelos direitos dos surdos descreve a situação de abandono que esses indivíduos sofrem:

> **"As dificuldades de conseguir intérpretes nas universidades brasileiras**
> Este é um depoimento que objetiva comentar um pouco sobre as dificuldades dos Surdos brasileiros já formados em Lingüística, Pedagogia e outras áreas que estudam e/ou trabalham nas Universidades Brasileiras Governamentais ou não, mas, não somente nessas universidades, como também nas Escolas Municipais e Escolas Estaduais no Ensino Fundamental e Médio que necessitam dos intérpretes capacitados em LIBRAS para o acompanhamento nas aulas.
> Eu, até hoje, me sinto angustiada, pois vejo tantas reclamações dos Surdos brasileiros ' com a falta de intérpretes, principalmente da Educação. (...)'
> *(Extraído de http://www.feneis.org.br/Educacao/artigos_pesquisas/dificuldades_interpretes.htm)*

A comunicação de pessoas normais para deficientes é feita através da leitura de lábios por aqueles que apresentam a limitação auditiva.

Explorando a grande adaptabilidade do corpo humano em situações de limitações físicas, o "Hand Talks!" baseia-se em um sistema simples de conversão de movimentos dos dedos das mãos, e das próprias mãos, em palavras sonoras ou textuais (exibidas em *displays*), já que, depois de algum tempo utilizando o

equipamento, o usuário familiarizar-se-á ao mesmo, como com um controle de *video game*.

A idéia principal desse projeto é implementar um "tradutor" para o alfabeto surdo-mudo, por razões de simplicidade, descrevendo posteriormente o que poderia ser feito para estender esse tradutor para interpretar as palavras da linguagem (gestos completos).

1.3 Organização

No capítulo 1 trata-se de uma breve introdução ao cenário que se desenrolará no documento inteiro, destacando os objetivos e motivações do projeto apresentado nesta monografia.

O capítulo 2 aborda os principais aspectos conceituais, linguagens escolhidas que estruturam o projeto. Nele estão descritas as tecnologias e fundamentos sobre qual o sistema foi concebido e implementado.

No capítulo 3 serão mostradas as especificações funcionais básicas do sistema e quais são os requisitos funcionais e não-funcionais, os casos de uso do sistema, as classes elaboradas e a interface com o usuário no caso do software.

O capítulo 4 refere-se como foi o planejamento para implementar o projeto descrito no capítulo anterior, bem como a metodologia utilizada para a elaboração do plano.

O Capítulo 5 descreve, com detalhes, o projeto, como ele evoluiu de um estágio primário para a solução definitiva, descrevendo cada módulo do sistema e como estes se comunicam.

No capítulo 6 veremos como o sistema foi testado e validado. Como procedemos nos testes, os critérios e resultados para avaliar a consistência do HandTalks!.

O capítulo 7 se dedica a analisar os resultados obtidos, perspectivas futuras e os riscos do projeto.

O capítulo 8 se concentra nas considerações finas que o grupo fez a respeito do projeto, seu desenvolvimento e conclusão.

No capítulo 9 encontraremos a Bibliografia consultada e utilizada nessa monografia, bem como referências a páginas da internet e afins.

O capítulo 10 apresenta os Anexos à monografia, que se tratam de documentos, esquemas e tabelas que detalham algum aspecto do projeto.

No capítulo 11 contém Apêndices à monografia, que são documentos complementares, de autoria de outros autores, que permite dar uma base do cenário sobre o qual o projeto foi desenvolvido.

2. Aspectos Conceituais

2.1 Linguagem utilizada

A linguagem a ser utilizada para o desenvolvimento desse sistema precisa ter duas características fundamentais: ser multiplataforma e permitir um desenvolvimento rápido. A python então foi escolhida por se enquadrar nesses moldes. É uma linguagem muito apropriada ao desenvolvimento de protótipos por possuir uma sintaxe clara, possuir tipos de alto nível e ser de fácil aprendizado.

Além disso, a python possui uma larga quantidade de biblioteca para as mais diversas funcionalidades, sendo que grande parte desses módulos também é multiplataforma. Tem-se, dessa maneira, um ambiente de desenvolvimento extremamente flexível.

2.2 Componentes Utilizados

Os componentes discretos, digitais, microcontrolador, LEDs, Photo Transistor, placas, fios e cabos empregados nesse projeto são os de uso mais comuns em projetos de eletrônica, sendo facilmente encontrados em lojas especializadas, de baixo custo e de confiabilidade razoável.

3. Especificação Funcional

A luva será composta de um módulo de detecção de movimentos (posição dos dedos), composto de sensores que irão enviar sinais eletrônicos dependendo da posição dos dedos da mão. Esse módulo de detecção enviará ao módulo seguinte, chamado de módulo de conversão, os sinais analógicos que ele detectou.

O módulo de conversão, por sua vez, irá transformar convenientemente os sinais analógicos em sinais digitais, passando esses sinais ao módulo de tradução.

O módulo de tradução pesquisará em sua base de dados a resposta ao estímulo na entrada, tendo com chave os sinais digitais do módulo anterior. Tendo encontrado a resposta (sonora e/ou textual), o este módulo envia ao módulo de execução para gerar a saída.

O esquema seguinte ilustra bem essa parte:

Figura 3.1: Diagrama em Blocos e Esquema Funcional do Sistema

3.1 Detalhamento

A fim de simplificar o projeto para permitir seu desenvolvimento no tempo hábil da disciplina, foi estipulado que apenas o alfabeto LIBRAS (Língua Brasileira de Sinais) deveria ser interpretado pelo sistema. O hardware e software foram, então, projetados para tal tarefa, mas sempre deixando a possibilidade de se criar uma expansão para que toda a língua brasileira de sinais pudesse ser traduzida (modularidade).

3.1.1 Hardware

O conjunto de hardware do projeto é composto por sensores e conversores além da luva em si. Para a conversão do sinal analógico em digital será usado um pequeno microcontrolador como o PIC 16F877 e, se necessário, multiplexadores analógicos e digitais. Esse microcontrolador já possui conversores A/D e interface USART.

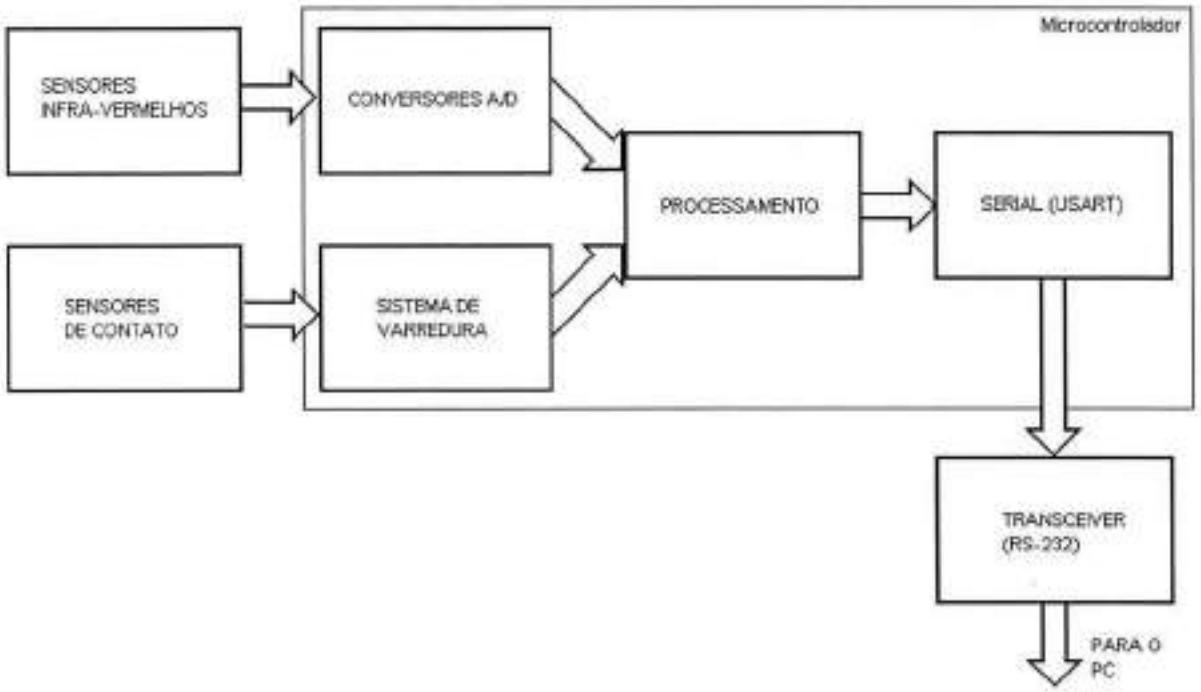


Figura 3.2: Diagrama em Blocos do Hardware (Detector + Conversor)

Para se determinar quais sensores seriam necessários para a captação do alfabeto libras foi feita uma análise do sinal utilizado em cada letra e número, partindo do pressuposto de que seriam necessários ao menos um sensor de dobra ou par de sensores ópticos para cada dedo. Esses sinais foram então comparados,

os possíveis conflitos foram agrupados e uma proposta de solução foi elaborada para eliminar os conflitos (ver anexo 10.3 e 10.9).

Com base nas soluções propostas, os sensores mínimos necessários para a detecção do alfabeto são os seguintes:

3.1.1.1 Sensores de Dobra

- 5 sensores de dobra (um para cada dedo)
- 2 sensores de posição angular:
  - Eixo longitudinal (para conflito entre 1, 6 e 9)
  - Eixo transversal
- 4 sensores de contato:
  - Polegar (parte interna e ponta) com:
    - Indicador (lateral externa)
    - Médio (ponta)
    - Médio (Base)
  - Contato entre Indicador e Médio;
- 1 sensor adicional de velocidade angular (ou de *tilt*), que captaria o "tranco" final da mão quando um sinal acaba de ser executado.

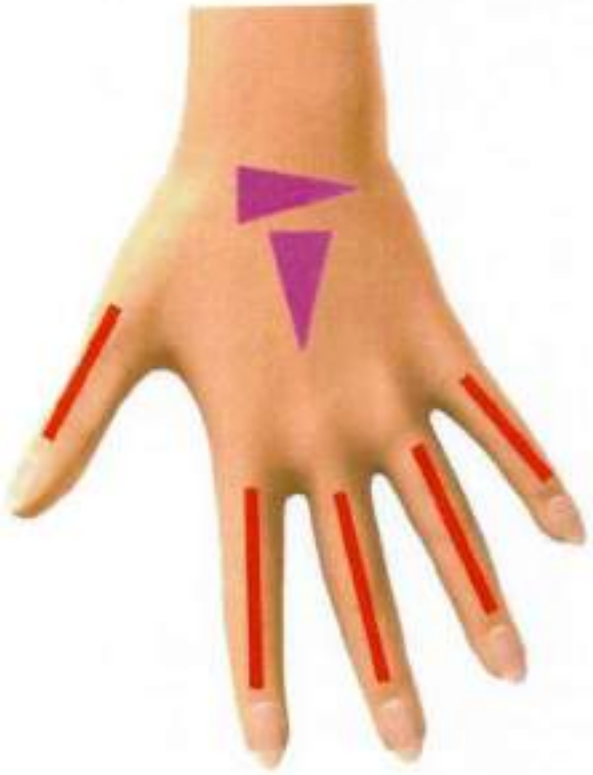


Figura 3.3.1: Local dos Sensores de Dobra na Luva

Figura 3.3.2: Local dos Contatos na Luva

3.1.1.2 Sensores Ópticos

- 5 pares de sensores/emissor infravermelho (um para cada dedo)
  - 4 sensores de contato:
    - Polegar (parte interna e ponta) com:
      - Indicador (lateral externa)
      - Médio (ponta)
      - Médio (Base)
    - Contato entre Indicador e Médio;

O projeto do sensor óptico, escolha desse grupo pode ser conferido nos Anexos 10.4, 10.5 e o circuito elétrico no Anexo 10.10.

As etapas de tradução e execução serão executadas num microcomputador PC com interface serial RS-232 e placa de som. Num segundo instante essas etapas poderão ser transferidas a um microcontrolador de médio porte, como o eZ80Acclaim, conectado a um CI de processamento de som e um autofalante.

3.1.2 Software

O microcontrolador de pequeno porte conterá um software que ficará monitorando o tranco da mão. Quando este sinal for detectado, ele envia o estado atual dos demais sensores pela interface serial. Ele também poderá enviar os estados dos sensores ao receber um pedido do computador.

Do lado do computador, haverá um software que receberá os dados da luva e, através de uma rede neural, fará a tradução dos sinais obtidos. Em seguida, o símbolo ou palavra obtido é exibido na forma de texto ou som.

O papel do software no projeto é o de fazer a tradução dos sinais recebidos através da luva HandTalks! do usuário emissor, de acordo com o alfabeto LIBRAS, e reproduzir a letra ao usuário receptor.

3.1.2.1 Descrição Geral

O software HandTalks! faz parte do projeto HandTalks! que possui, além do software, uma luva com sensores que capta a posição dos dedos e o contato entre eles.

Além da interação com a luva, o software faz interface com dois tipos de usuário. O usuário emissor é responsável por um ajuste inicial da luva no software e o usuário receptor recebe a tradução do sinal feito pelo emissor sem qualquer interação com o software.

A letra traduzida deverá ser reproduzida na forma de áudio e texto, além de permanecer armazenada para que se obtenha palavras e frases completas, que serão apenas exibidas na forma de texto. Alguns sinais adicionais ao LIBRAS serão desenvolvidos para a obtenção de um espaço em branco, para a exclusão da última letra reconhecida e para a sinalização de fim de parágrafo.

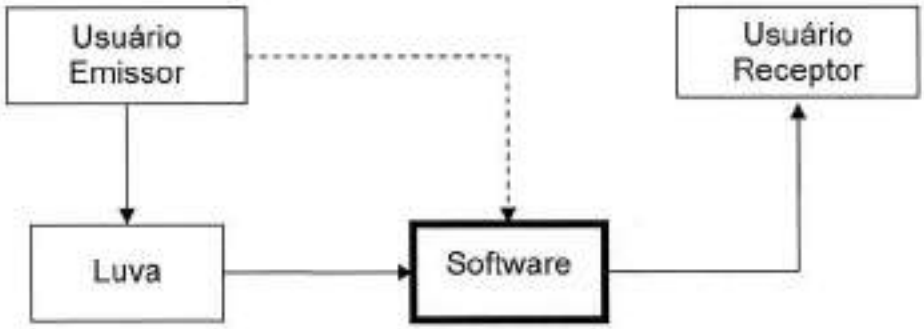


Figura 3.4: Diagrama Básico do Funcionamento do HandTalks!

3.1.3 Protocolos

A comunicação serial RS-232 será feita através de um protocolo que será desenvolvido numa das etapas do projeto. Esse protocolo terá comandos para a obtenção de dados e suas respostas deverão informar qual o sensor em questão e seu valor atual. Quando a resposta é gerada de maneira automática (tranco da mão), essa informação também deverá ser passada, junto com a intensidade do tranco.

4. Planejamento e Metodologia

O plano de implementação para a realização do projeto será baseado na divisão do projeto em 4 grandes módulos:

4.1Módulo 1: Detector

Como o detector envolve apenas componentes eletrônicos discretos, essa parte envolve apenas a elaboração do circuito eletrônico, ajustando as faixas de tensão requeridas aos próximos módulos.

4.2Módulo 2: Conversor

O conversor já envolve codificação do sinal analógico em sinal digital, devendo então estabelecer as condições necessárias para obter-se um sinal digital compatível com os protocolos de transmissão serial (RS-232).

Esta fase envolve apenas o desenvolvimento de um circuito eletrônico e da programação de um microcontrolador de pequeno porte, no caso o Z8. Ele possui um módulo de conversão AD que pode ser multiplexado para até quatro entradas.

No nosso projeto utilizaremos um esquema de amostragem por varredura para os vários sensores, sendo que os valores serão transmitidos com a ocorrência de um evento disparador (o tranco, um intervalo de tempo, ou outro, conforme a conveniência).

4.3Módulos 3 e 4: Tradutor e Executor

Fase onde já estamos recebendo os sinais digitais da interface-homem máquina (luva com sensores). Devemos, então, a partir dos sinais de entrada, pesquisar na memória o endereço através de uma rede neural, que a partir da entrada gerada pela luva, irá encontrar o endereço correspondente à entrada, enviando esse endereço ao módulo de execução que irá reproduzir o som e/ou exibir em um *display* a letra/palavra correspondente.

4.3.1 Requisitos Específicos

A seguir apresentam-se os diversos modelos utilizados na Análise Orientada a Objeto (OOA – Object-Oriented Analysis) empregada.

4.3.1.1 Modelo de Casos de Uso

Atores

O único ator presente nos casos de uso é o usuário emissor, pois ele é responsável pelo ajuste da luva e funcionamento do sistema como um todo, além de executar os sinais que serão traduzidos.

Casos de Uso

- Configuração da Comunicação
- Ajuste da Luva
- Ativação da Comunicação

4.3.2 Descrição dos Casos de Uso

4.3.2.1 Configuração da Comunicação

Descrição: Este caso de uso descreve o processo de configuração da comunicação serial.

Pré-condição: Software em execução, com a comunicação parada.

Evento iniciador: Solicitação de configuração de comunicação (botão)

Atores: Usuário Emissor

Seqüência de eventos:

1. O usuário emissor clica em "Configurar Comunicação".
2. O sistema exibe uma janela com as opções possíveis de comunicação serial, que são as seguintes:
    a. Porta: "COMn" em sistemas Windows, "/dev/ttySn" em sistemas Linux;
    b. Velocidade: taxas padrão de 2400 à 115200 baud.
    c. Bits de Dados: 5, 6, 7 ou 8;
    d. Paridade: Nenhuma, Par ou Ímpar;
    e. Bits de Parada: 1 ou 2;
    f. Controle de Fluxo: RTS/CTS ou Xon/Xoff;
3. O usuário escolhe cada opção, de acordo com a configuração do sistema, e clica em OK.

Pós-condição: A comunicação serial fica configurada, permitindo que seja ativada.

Extensões:

1. O usuário clica no botão "Cancelar"
2. O sistema ignora qualquer alteração na configuração da comunicação e volta o foco à janela principal.

4.3.2.2 Ajuste da Luva

Descrição: Este caso de uso descreve como é feito o ajuste da luva no software.

Pré-condição: Comunicação configurada.

Evento iniciador: Solicitação de ajuste da luva (botão)

Atores: Usuário Emissor

Seqüência de eventos:

1. O usuário emissor clica em "Ajustar Luva".
2. O sistema exibe um assistente com três páginas, para ajuste da mão flexionada, relaxada e esticada, respectivamente. Em todas as páginas haverá um botão "Efetuar Leitura" (além dos botões padrões de um assistente) para capturar o valor atual dos sensores na página especifica.

3. O usuário navega pelo assistente, clicando em "Efetuar Leitura" em cada uma das páginas para calibrar a luva.
4. O usuário clica em "Finalizar" na última página do assistente.

Pós-condição: O software está ajustado ao usuário específico. Esse ajuste pode ser armazenado para posterior recuperação.

Extensões:

1. O usuário clica no botão "Cancelar".
    a. O sistema ignora qualquer alteração na o ajuste da luva e volta o foco à janela principal.
2. O usuário não efetua o ajuste em todas as páginas do assistente.
    a. O sistema não disponibiliza o botão "Finalizar" na última página.

4.3.2.3 Ativação da Comunicação

Descrição: Este caso de uso descreve o processo de ativação da comunicação serial.

Pré-condição: Comunicação configurada e software ajustado à luva.

Evento iniciador: Solicitação de início de comunicação (botão)

Atores: Usuário Emissor

Seqüência de eventos:

1. O usuário emissor clica em "Iniciar Comunicação".
2. O sistema passa a captar os dados da luva e traduzi-los, de forma autônoma.

Pós-condição: O sistema está pronto para ser utilizado pelo usuário receptor.

Extensões:

1. O sistema não consegue se comunicar com a luva.
    a. O sistema exibe uma mensagem de erro e continua sem comunicação com a luva.

4.3.3 Modelo de Classes

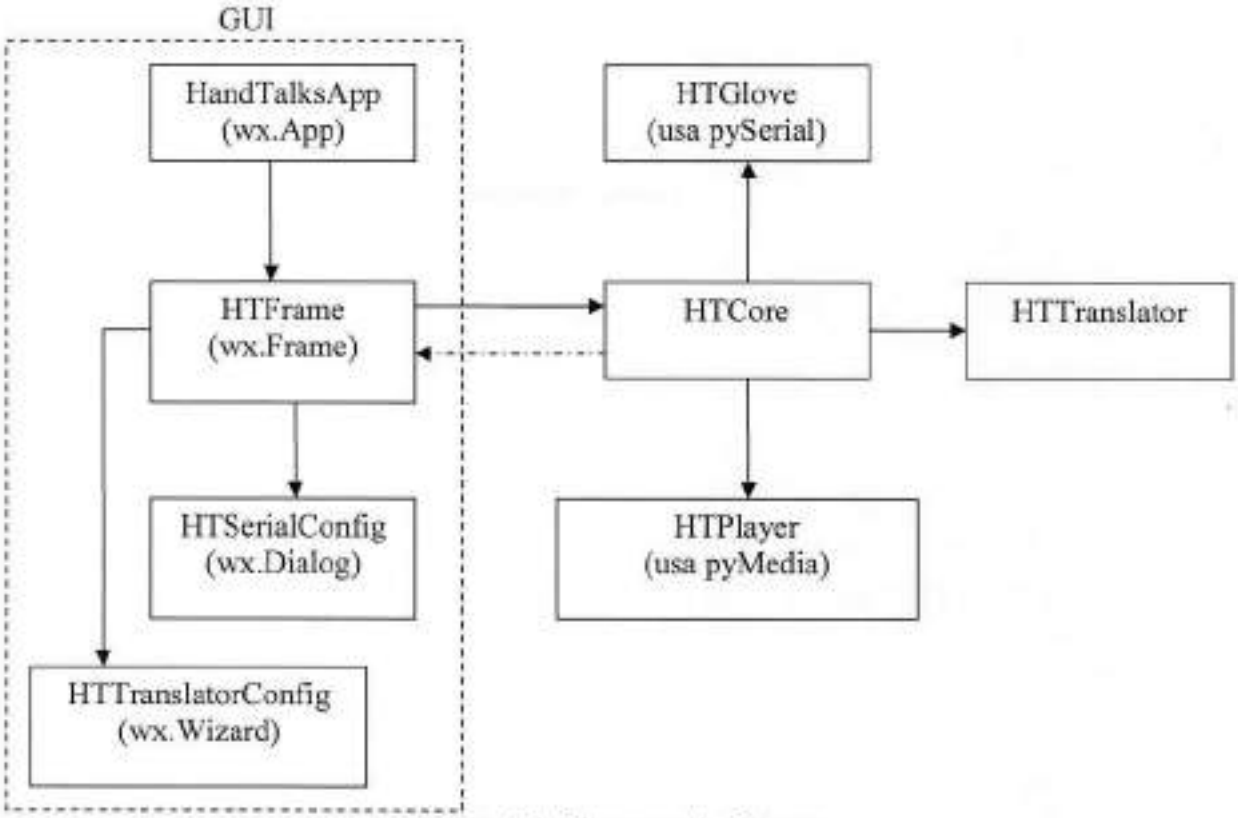


Figura 4.1: Diagrama de Classes

HTCore é a classe que coordena todas as operações do software deixando HandTalksApp, HTFrame, HTSerialConfig e HTTranslatorConfig responsáveis apenas pela interface com o usuário. Desse modo a interface do programa pode ser facilmente substituída por outra.

HTGlove é responsável por estabelecer a comunicação com a luva. Os dados recebidos são preparados e passados para HTTranslator, que faz a tradução e o histórico das letras recebidas. A letra traduzida é passada à HTPlayer, que reproduz o som apropriado, e à HTFrame (através de um evento), que a exibe na tela junto com o histórico.

4.3.4 Requisitos Não Funcionais

Os únicos requisitos não-funcionais aos quais o sistema deve garantir um nível satisfatório são:

4.3.4.1 Usabilidade

O sistema deve ser de fácil utilização.

4.3.4.2 Confiabilidade

A tradução efetuada deve ser confiável, ocorrendo o mínimo possível de erros.

4.3.4.3 Portabilidade

O sistema deve ser executado nas mais variadas plataformas, minimamente em Windows e Linux.

4.4Divisão de Tarefas

Toda a equipe se empenhará em todas as fases, compartilhando as tarefas e e não dividindo o trabalho. Mas conseqüentemente algumas tarefas serão realizadas individualmente. Por isso, o maior responsável por cada módulo é uma separação apenas de quem irá desempenhar a maior parte das tarefas de um dado módulo.

1) Pesquisa de Componentes e Preços
2) Pesquisa Libras
3) Elaboração do Diagrama Elétrico do Detector
4) Implementação do Detector
5) Teste do Detector
6) Elaboração do Diagrama Eletrônico do Conversor
7) Pesquisa do Protocolo de Comunicação
8) Implementação do Conversor
9) Teste do Conversor
10) Integração Detector ↔ Conversor
11) Teste da Integração Detector ↔ Conversor
12) Pesquisa Machine Learning
13) Projeto do Tradutor (Requisitos, diagramas etc)
14) Implementação do Tradutor
15) Teste do Tradutor
16) Integração Detector+Conversor ↔ Tradutor
17) Teste da Integração Detector+Conversor ↔ Tradutor
18) Relatório Final e Apresentação

Concorrentemente com cada tarefa, será elaborado um documento relacionado à mesma.

Tarefa	*André*	*Hilton*	*Hugo*
1	Sensores Ópticos	Sensores Resistivos	Luvas Comerciais
2	X	X	X
3			X
4	X	X	X
5	X		
6		X	
7	X		
8	X	X	X
9		X	X
10		X	X
11	X	X	
12	X	X	X
13	X	X	X
14	X	X	X
15	X		X
16		X	X
17	X		X
18	X	X	X

Tabela 1: Divisão de Tarefas

Um gráfico detalhado e auto-explicativo da divisão de tarefas no projeto pode ser visto no Anexo 11.1.

5. Projeto e Implementação

5.1Protótipo

Como os componentes deveriam ser importados, o grupo resolveu desenvolver um protótipo utilizando sensores ópticos simplificados para exemplificar o funcionamento básico da luva.

Na tarefa de prototipação, desenvolveu-se um modelo simples de detector, utilizando um emissor e um receptor óptico. O diagrama elétrico abaixo sugere o funcionamento do mesmo:

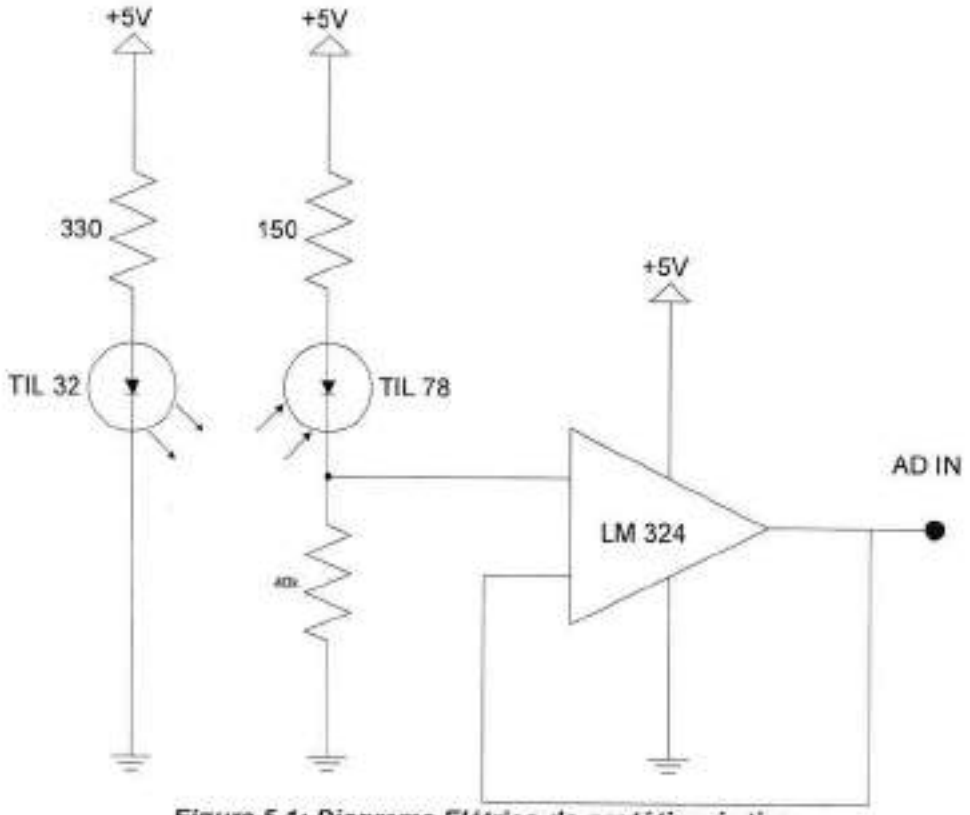


Figura 5.1: Diagrama Elétrico do protótipo óptico

5.2Detector

O módulo do detector foi implementado com uma luva, com sensores acoplados que serão descritos detalhadamente a seguir.

5.2.1 Luva

Foi escolhida uma luva de golfe para ser utilizada, pois apresenta o melhor custo benefício em relação à durabilidade, flexibilidade e justabilidade.

O tecido é composto, tendo partes em lycra, um material sintético, bem justo e flexível e couro nas partes que se necessita de maior durabilidade.

Na luva, foram costurados 10 suportes para LED's, que irão comportar 5 pares de photo diodos e LED infravermelhos, que são os sensores de dobra do dedo.

5.2.2 Sensores

Basicamente formado por pares de Sensor Infravermelho e Emissor Infravermelho e pontos de contato.

No par sensor/emissor foram utilizados os componentes TIL 32 e TIL 78.

5.2.2.1

5.2.2.2 Polarização do TIL 32

I = (Vcc - Vd)/RI

I = (5 - 0,4)/150

I = 30 mA

Figura 5.2.1: Polarização do TIL 32

5.2.2.3 Foto transistor TIL 78

Os valores dos resistores foram escolhidos para adequar os valores de tensão de acordo com a curva de transferência de intensidade percebida por corrente de coletor.

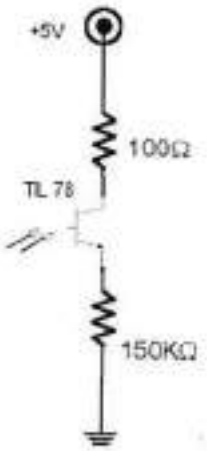


Figura 5.2.2: Polarização do TIL 78

5.2.2.4 Proteção

Para limitação forçada dos níveis de tensão nas entradas dos canais AD, utilizamos amplificadores operacionais LM324 em configuração seguidor. Desta forma, com o amplificador operacional alimentado em 5V, a tensão não ultrapassa 3,8V ~ 4,0V.

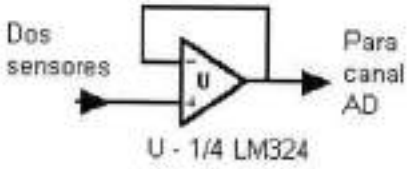


Figura 5.3: Circuito de Proteção

Diodos limitadores também poderiam ser utilizados na entrada dos amplificadores, principalmente se o sinal de entrada fosse oriundo de outro circuito.

Temos assim a curva de transferência do circuito de proteção:

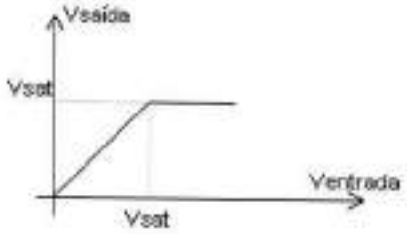


Figura 5.4: Curva de Transferência do Circuito de Proteção

A tensão satura a partir do valor Vsat na entrada do amplificador operacional.

A saída produzida nos amplificadores entra diretamente nas entradas dos canais AD do microcontrolador.

Como contatos, foram utilizados fios condutores no formato em malha, como o desenho a seguir sugere:

Figura 5.5: Malha de Contato

A resistência desse contato, ponto a ponto é de aproximadamente 1 ohm.

São, ao todo, seis contatos distribuídos como na figura 4b.

Figura 5.6: Posições dos Contatos

A luva será calibrada para cada usuário por software, os três estados básicos da luva serão:

5.2.2.5 Relaxado

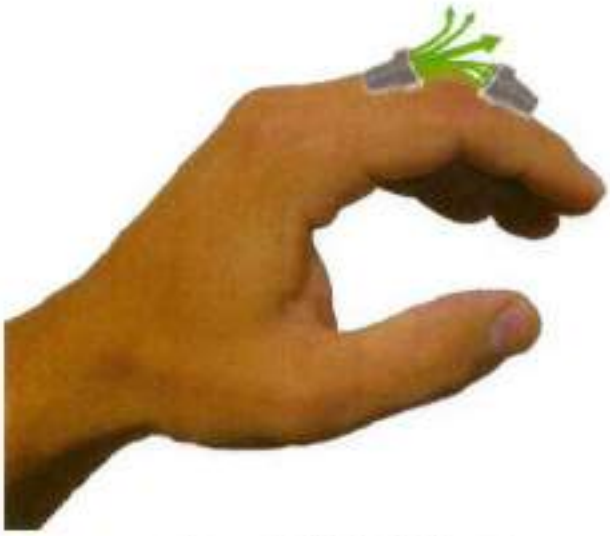


Figura 5.7.1: Mão Relaxada

Permite uma passagem de luva média, pois a distancia não é tão próxima e a angulação não é exatamente reta, Há um pequeno ângulo.

5.2.2.6 *Esticado*

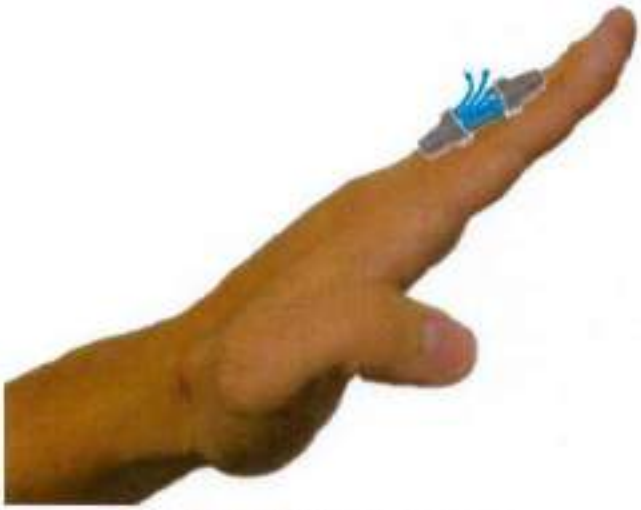


Figura 5.7.2: Mão Esticada

Permite uma grande passagem de luz, pois a distância entre o sensor e o emissor é a mais curta possível, e o não há "ângulo" entre suas linhas de emissão/recepção.

5.2.2.7 *Contraído*

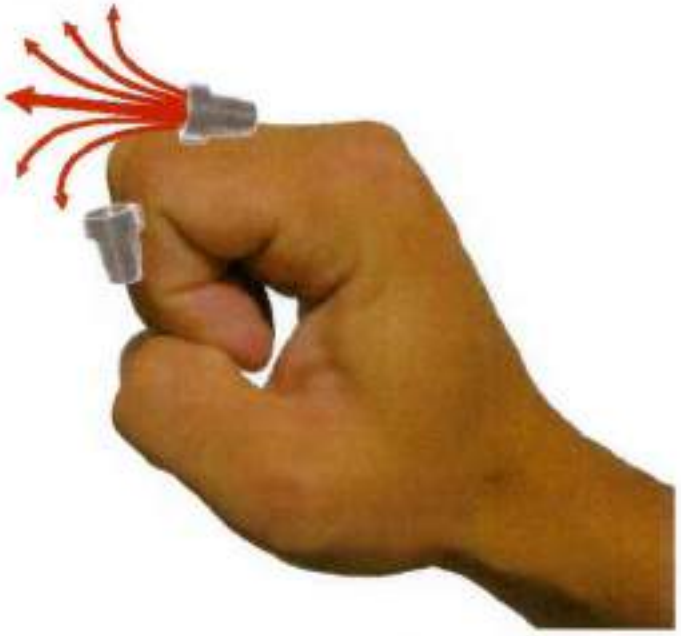


Figura 5.7.3: Mão Contraída

Possui a mais baixa incidência de luz, pois o ângulo é aproximadamente reto entre as linhas de "visada" e a incidência de luz é residual.

Há estados intermediários que serão tratados por software segundo a lógica fuzzy.

5.3Conversor

Inicialmente, iniciamos nossos testes utilizando o kit microcontrolador Z8 da Encore com encapsulamento de 8 pinos, que possui incluso 4 canais AD (com compartilhamento de pinos) e serial RS-232. No entanto o kit se mostrou insuficiente para cobrir os requisitos mínimos para a implementação da luva. São necessários 5 canais AD no mínimo (um para cada dedo, relacionados às dobras) e 4 entradas e 4 saídas para fazer o mapeamento dos contatos.

Foi decidido implementar a parte principal da luva com o microcontrolador PIC 16F877 da Microchip, que possui 8 canais AD, USART e 5 ports para IO, 4 de 8 bits e 1 port com 3 bits (compartilhados com demais recursos). No entanto seria necessário montar a interface de comunicação serial RS-232, considerando que o microcontrolador trabalha com níveis TTL.

Para a captação dos sinais, foi proposto um sistema de varredura utilizando-se uma das portas de 8 bits do microcontrolador, dispondo os contatos de uma forma matricial.

Para que a limitação quanto a contatos de uma mesma linha/coluna feitos simultaneamente não ocorra, é implementado um esquema de varredura.

No caso está sendo utilizado o portB, já que pode-se configurar os pinos como saída ou como entrada e com resistores internos de pull-up .

Utilizamos os 8 bits do portB fazendo a seguinte configuração:

- bits menos significativos como entradas fixas (numerar);

- bits mais significativos como entradas ou saídas (alternadas);

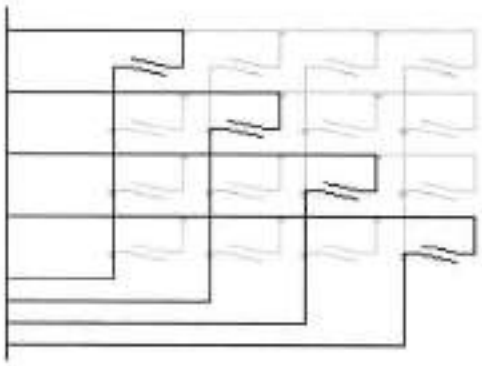


Figura 5.8: Malha de Contatos mapeada na serial

Para o mapeamento proposto, apenas os contatos contínuos representados na figura estão implementados, sendo que uma possível ampliação futura poderá ser feita.

Fazemos o seguinte procedimento:

- Todos os pinos são configurados como entrada e com pull-up interno;
- O 1.o pino é configurado como saída;
- Joga-se nível '0' neste pino de saída;
- Lê-se os pinos de entrada verificando quais as entradas em pull-up foram a '0';
- Repete-se o procedimento, trocando-se o pino a ser configurado como saída.

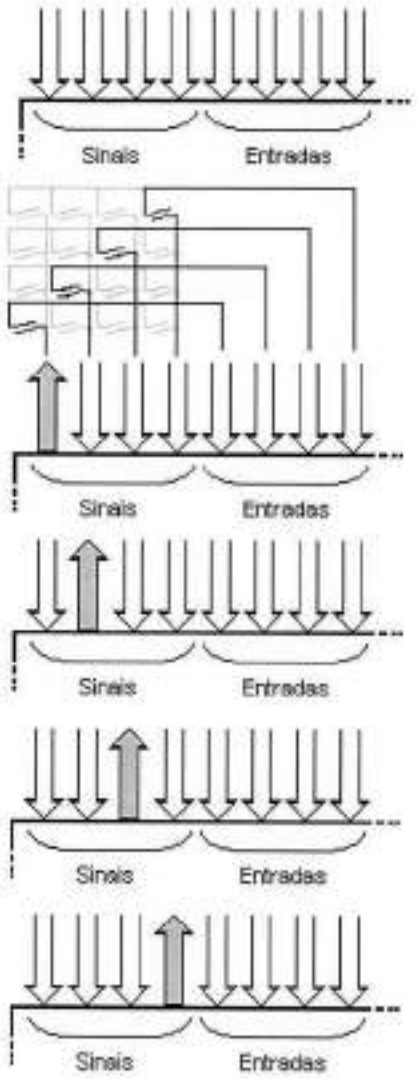


Figura 5.9: Procedimento para a Leitura da Malha de Contatos

5.3.1 Mapeamento

Dobras (ch0 a ch4)			***Contatos (ctcs)***		
Polegar (P)	channel 0	(ch0)	Sinais:		
Indicador (I)	channel 1	(ch1)	PortB	IO: 7 a 4	A a D
Médio (M)	channel 2	(ch2)	Entradas:		
Anelar (A)	channel 3	(ch3)	PortB	IO: 3 a 0	0 a 3
Mínimo (N)	channel 4	(ch4)			

Tabela 5.1: Mapeamento dos Sensores nas Portas Seriais

Os dados são convertidos para decimais ASCII e enviados via RS-232 em um trem de caracteres, que é a string com dados capturados do mapeamento.

Formato: ***$ch0#ch1#ch2#ch3#ch4#ctcs##***

Em que:

- ***$:*** caractere de start.
- ***ch0 a ch4:*** 4 dígitos cada. Decimais ASCII, oriundos dos canais (channel i) AD's.
- ***#:*** caractere de separação de campos.
- ***ctcs:*** 1 dígito. Hexadecimal convertido em ASCII, oriundos dos contatos.
- ***##:*** finalização da string.

5.3.2 Serial

O microcontrolador PIC 16F877 possui uma USART implementada. Para a comunicação entre o protótipo e o computador onde está o módulo de tradução, utiliza-se um transceiver compatível com o modelo MAX232, (14C232) compatibilizando os níveis TTL oriundos do microcontrolador com os níveis RS-232 do PC. Detalhes da implementação da serial podem ser vistos no item Recursos Utilizados e Registradores.

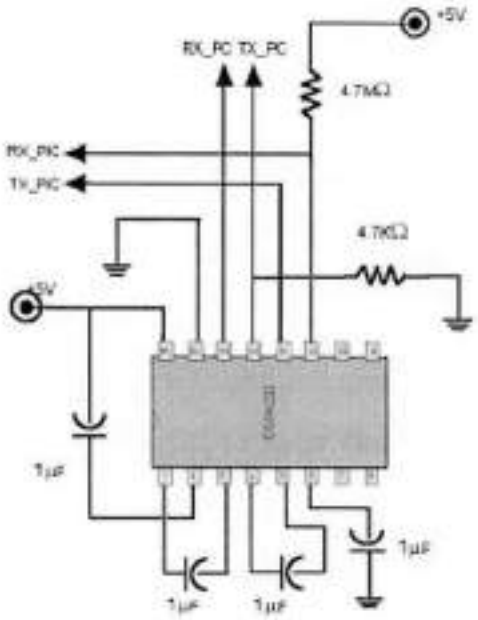


Figura 5.10: Ligações da Serial do PIC (Transceiver)

5.3.3 Fluxograma

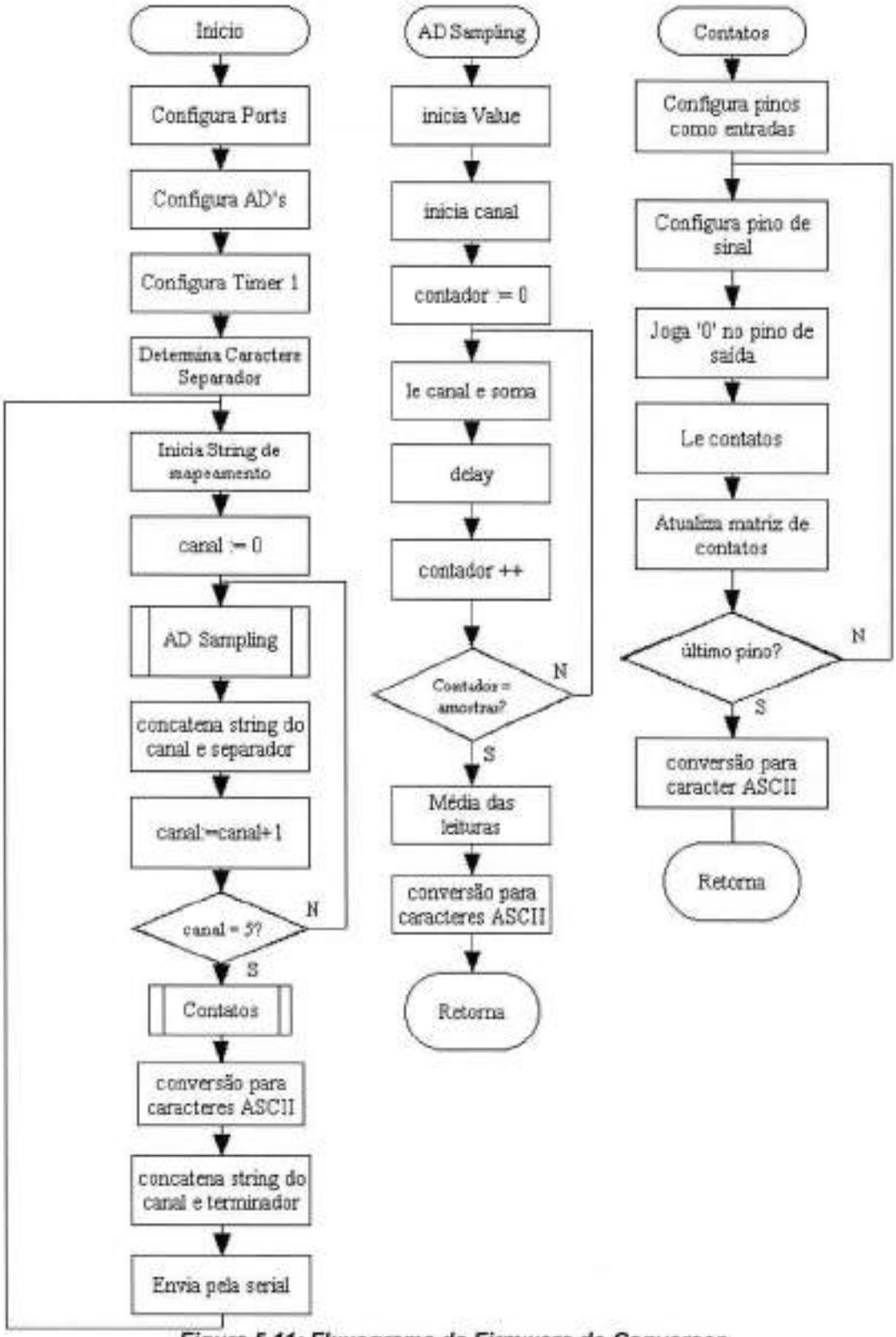


Figura 5.11: Fluxograma do Firmware do Conversor

5.3.4 Recursos utilizados e registradores

5.3.4.1 AD e Registradores:

É necessária a configuração de uma série de registradores para o uso dos canais de conversão AD: ADRESH (Resultado High); ADRESL (Resultado Low); ADCON0 (Controle 0); e ADCON1 (Controle 1).

ADRESH e ADRESL

ADRESH e ADRESL são os registradores de resultado da conversão A/D, formando juntos, um registrador de 10 bits. O resultado poderá ser alinhado à esquerda ou à direita conforme bits de controle em ADCON1.

ADCON0 - controle do conversor AD:

bit 7-6 ADCS1:ADCS0: A/D clock de conversão

00 = FOSC/2

01 = FOSC/8

10 = FOSC/32

11 = FRC (do módulo oscilador RC interno do A/D)

No caso, configurado em 11.

bit 5-3 CHS2:CHS0: seleção de Canal

000 = channel 0, (RA0/AN0)

001 = channel 1, (RA1/AN1)

010 = channel 2, (RA2/AN2)

011 = channel 3, (RA3/AN3)

100 = channel 4, (RA5/AN4)

101 = channel 5, (RE0/AN5)(1)

110 = channel 6, (RE1/AN6)(1)

111 = channel 7, (RE2/AN7)(1)

Iniciado em 000, sendo trocado no decorrer da varredura. (0 a 4, 000 ~ 100)

bit 2 GO/DONE: Status de Conversão A/D

Se o ADON = 1:

1 = A/D conversão em progresso (impondo '1', a conversão A/D se inicia)

0 = A/D conversão não está em progresso (automaticamente vai a '0' no fim de uma conversão A/D)

bit 1 Não implementado: Lido como '0'

bit 0 ADON: A/D On

1 = módulo de conversão A/D está operando

0 = módulo de conversão A/D desligado

ADCON1 - Configuração do AD

bit 7 ADFM: A/D Formato do Resultado

1 = Alinhamento à direita. 6 bits mais significativos do ADRESH lidos como '0'.

0 = Alinhamento à esquerda. 6 bits menos significativos do ADRESL lidos '0'.

bit 6-4: (não implementado) lido como '0'

bit 3-0 PCFG3:PCFG0: Configuração de Port A/D:

PCFG3: PCFG0	AN7[1] RE2	AN6[1] RE1	AN5[1] RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs[2]
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A – entrada analógica D – I/O digital

Tabela 5.2: Configuração dos pinos de AD

Neste caso, é utilizada a configuração 0010 (canais AN0 a AN4, equivalendo ao que foi chamado de ch0 a ch4, respectivamente).

5.3.4.2 USART e Registradores:

Para utilização da USART, deve ser considerada a configuração do registrador TXSTA, que é o registrador de controle e status da transmissão.

TXSTA:

bit 7 CSRC: Seleção de fonte de clock

Asynchronous mode: Irrelevante

Synchronous mode:

1 = Master mode (clock gerado internamento a partir do BRG)

0 = Slave mode (clock de fonte externa)

bit 6 TX9: Habilitação de transmissão do nono bit

1 = transmissão de 9-bit

0 = transmissão de 8-bit

bit 5 TXEN: Habilitação de transmissão

1 = Transmissão habilitada

0 = Transmissão desabilitada

SREN/CREN sobreescreve TXEN no SYNC mode.

bit 4 SYNC: Seleção de Modo da USART

1 = Synchronous mode

0 = Asynchronous mode

Configurado em modo assincrono.

bit 3: (não implementado) Lido como '0'.

bit 2 BRGH: Geração de Baud Rate

Asynchronous mode:

1 = High speed Baud Rate = FOSC/(16(X+1))

0 = Low speed Baud Rate = FOSC/(64(X+1))

Synchronous mode:

Não usado neste modo

bit 1 TRMT: Status do TSR (Transmit Shift Register Status)

1 = TSR vazio

0 = TSR cheio

bit 0 TX9D: nono bit de Dado (pode ser o bit de paridade)

5.4 Tradutor e Executor

Por se tratar de um produto que tem a proposta de se tornar pervasivo em versões futuras, ele não depende nem interage com outros sistemas, ou seja, funciona de forma independente.

5.4.1 Tradutor Fuzzy

A lógica fuzzy foi definida utilizando-se uma variável de entrada com três estados para cada dedo:

Figura 5.12: Gráfico Fuzzy dos Três Estados

As letras são previamente tabeladas de acordo com os três estados possíveis de cada dedo, e de acordo com os estados dos contatos, como no trecho de código seguinte:

```
'B': {'fingers': [CONTRACTED, STRAINED, STRAINED, STRAINED, STRAINED],
      'contacts': [False, False, True, False]},
```

Cada estado é definido no processo de calibração, onde o usuário emissor ajusta o software à sua mão. O estado relaxado era inicialmente triangular e também definido na calibração, porém para contornar o problema dos sensores infra-vermelho serem muito instáveis (ver Resultados Obtidos), esse estado foi transformado num trapézio, definido de acordo com a posição da mão contraída:

```
x1 = self.contracted[finger]
x2 = x1 + 20
x3 = x2 + 40
x4 = self.strained[finger]
```

Para uma certa posição da mão, o processo de tradução qualifica cada dedo em cada um dos três estados, especificando a porcentagem destes estados para cada valor lido, conforme a figura 5.12. Um exemplo poderia ser:

	Polegar	Indicador	Médio	Anelar	Mínimo
Contraído	0%	78%	98%	100%	93%
Relaxado	20%	12%	2%	0%	7%
Esticado	80%	0%	0%	0%	0%

A tabela das letras é, então, filtrada de acordo com a informação dos contatos. Assim, apenas as letras com contatos semelhantes são processadas. Cada letra é avaliada e uma porcentagem final é dada indicando a proximidade da letra com o dado de entrada. Exemplo:

	Polegar	Indicador	Médio	Anelar	Mínimo	**Final**
Letra A	80%	78%	98%	100%	93%	**90%**

No último passo não é feita a *defuzzyficação*, apenas a letra que obtiver uma maior proximidade é retornada como resultado da tradução, desde que acima de 70% e distante 5% da "segunda colocada". Se essas condições não forem satisfeitas nenhuma letra é retornada, indicando estado de repouso.

5.4.2 Interfaces de Usuário

A interface com o usuário deve ser o mais fácil e intuitiva possível. Por essa razão, alguns aspectos foram levados em consideração:

- Utilização tanto de menus quanto de barra de ferramentas.
- Uso de ajudas rápidas:
    - Curtas: surgem quando o cursor do mouse fica parado sobre um componente.
    - Longa: exibida na barra de estado.
- Quando possível, utilizar assistentes de configuração.

A janela principal deve seguir, portanto, o seguinte protótipo:

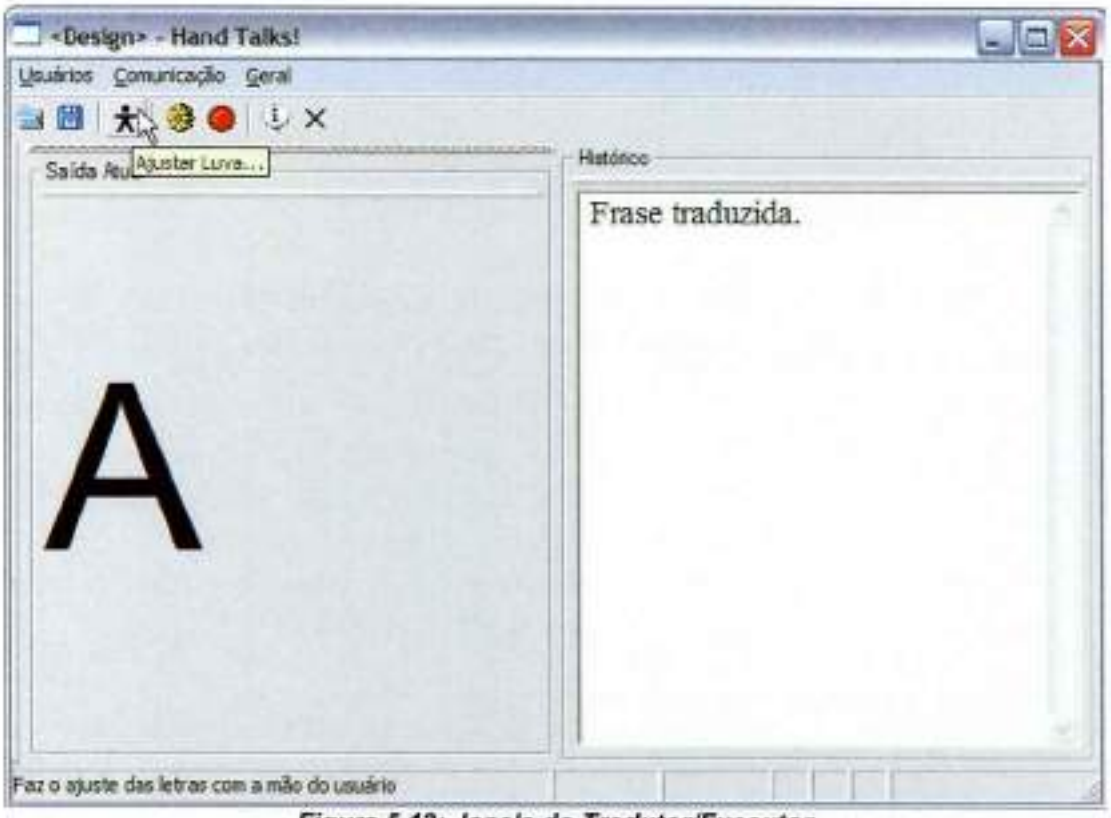


Figura 5.13: Janela do Tradutor/Executor

A janela de configuração da porta serial terá a seguinte aparência:

Figura 5.14: Janela de Configuração da Comunicação Serial

A janela de ajuste da luva ao software será na forma de um assistente com 3 páginas: para ajuste da mão flexionada, relaxada e esticada; e se baseará no seguinte modelo:

Figura 5.15: Janela Protótipo de Calibração

5.4.3 Interfaces de Hardware

Para receber os dados dos sensores da luva, o software se conecta com esta através da porta serial (RS-232) do PC.

5.4.4 Interfaces com Software

O sistema deverá ser construído de forma a ser multiplataforma e, para isso, não apenas a linguagem utilizada precisa desta característica, mas as eventuais bibliotecas também.

A python possui os seguintes módulos que serão utilizados pelo software HandTalks!:

5.4.4.1 pyMedia

Biblioteca para reprodução de mídias nos mais diversos formatos (mp3, ogg e wav, entre outros);

5.4.4.2 pySerial

Biblioteca para acesso á porta serial. Utiliza a porta COMn: em ambiente windows e o arquivo /dev/ttySn em ambiente Linux;

5.4.4.3 wxPython

Biblioteca que gera GUIs com a aparência nativa da plataforma onde o sistema esteja sendo executado (Windows ou Linux)

5.4.5 Interfaces de Comunicação

Para a comunicação entre o software e a luva foi criado um protocolo bem simples. Há apenas um comando que a luva manda continuamente ao software, em texto ASCII:

```
$<polegar>#<indicador>#<médio>#<anelar>#<mínimo>#<contatos>##
```

O valor do sensor de dobra dos dedos varia de 0 à 255 e o campo <contatos> é um mapa de bits com os contatos possíveis.

5.4.6 Operação

A partir da janela principal do programa é possível executar as três operações previstas:

5.4.6.1 Configuração da comunicação:

Ajusta os detalhes da comunicação serial através de uma janela de configuração.

5.4.6.2 Ajuste ao usuário da luva (usuário emissor)

Cria os ajustes de um usuário emissor através de um assistente (com opção de armazenamento em disco), ou abre ajustes de um usuário emissor já armazenados.

5.4.6.3 Ativação da comunicação

Após ter a comunicação e o usuário emissor configurados, basta acessar a opção de ativação da comunicação para que o usuário receptor comece a receber as traduções de forma autônoma.

5.4.7 Funções do Software

O sistema possui duas funções principais: receber os sinais da luva e traduzir os sinais em som e texto.

A recepção do sinal inclui a decodificação dos dados de acordo com o protocolo estabelecido e a adequação desses dados para a utilização na lógica de tradução.

Para fazer a tradução dos sinais, será utilizado um algoritmo de lógica nebulosa (fuzzy) que retornará a letra correspondente à configuração de entrada e a precisão desta informação. Se esta precisão for muito baixa a tradução será ignorada, caso contrário a letra correspondente será exibida na tela e será reproduzido um som que "lê" a letra. Vale ressaltar que deverão ser feitos testes para levantar qual o valor ideal da precisão para que informação seja considerada válida.

Após a tradução, a letra recebida é adicionada à um histórico das letras recebidas, onde poderão ser formadas frases completas. Alguns símbolos adicionais serão criados para criar pontuação e deixar essa exibição visualmente mais agradável.

5.4.8 Características dos Usuários

Dois usuários são previstos para utilizar o sistema. O primeiro é o usuário emissor, que vestirá a luva e gerará os sinais a serem traduzidos e, portanto, será o portador de deficiência fono-auditiva. Não é necessário que este usuário tenha experiência no uso de computadores, mas caso ele não tenha tido nenhum contato com computadores anteriormente será necessário um pequeno treinamento sobre como utilizar a interface. Será necessário que esse usuário tenha, também, conhecimento sobre a língua portuguesa, que talvez não seja comum à todos os conhecedores da língua LIBRAS.

O segundo usuário é o receptor, que não precisa ter conhecimento algum de informática pois o sistema funciona de forma autônoma efetuando constantemente a tradução dos sinais recebidos. A este usuário só é necessário o conhecimento sobre a língua portuguesa.

5.4.9 Restrições

O sistema possui duas sérias restrições por ser ainda um protótipo ou uma idéia inicial do que poderia ser um produto vendável.

A primeira restrição é quanto à tradução apenas do alfabeto LIBRAS, e não de toda a língua. Isso inviabiliza o uso do produto por parte dos deficientes já que, em geral, os sinais feitos não são o do alfabeto, mas outros que representam palavras e ações.

A utilização de um micro-computador PC para a tradução dos sinais é a segunda restrição do sistema, pois impede o uso pervasivo do mesmo. Para uma aplicação real, seria necessário o uso de um microcontrolador portátil que executasse a função do PC e se comunicasse com a luva sem o uso de fios (por bluetooth, por exemplo).

5.4.10 Hipóteses e Dependências

O software HandTalks! depende da luva estar projetada, construída e funcionando, para que os testes possam ser feitos. Isso inclui o hardware da luva (sensores e contatos), bem como o software do microcontrolador PIC que captará os dados e os enviará ao PC.

5.4.11 Versões futuras

Para versões futuras do sistema espera-se que este seja pervasivo, ou seja, seja executado num microcontrolador com display de cristal líquido e autofalante embutidos. Dessa forma o portador de deficiência pode se comunicar com outras pessoas em qualquer lugar.

Outro recurso importante seria a tradução de toda a língua de sinais, não apenas o alfabeto. Esta é uma tarefa que exige grandes mudanças tanto no hardware como no software, mas o sistema desenvolvido pode ser usado como ponto de partida e estímulos iniciais.

5.4.12 Critérios de Aceitação

Por não se tratar ainda de um produto viável a ser utilizado pelos deficientes fono-auditivos, o teste de aceitação será feito pelos próprios integrantes do grupo e o critério para sua aceitação será a correta tradução do sinal efetuado e a correta detecção da mudança de sinal.

6. Testes e Avaliação

6.1Detector

O teste do detector se baseia em verificar se os sinais dos sensores variam significativamente para que se possam identificar os três estados principais de um par de sensores: Esticado, Relaxado e Contraído.

Na posição de Esticado, ou seja, com a maior intensidade possível de sinal, obtivemos um sinal de amplitude média de 3,7 Volts, com desvio de aproximadamente 6%, como pode ser mostrado no gráfico abaixo:

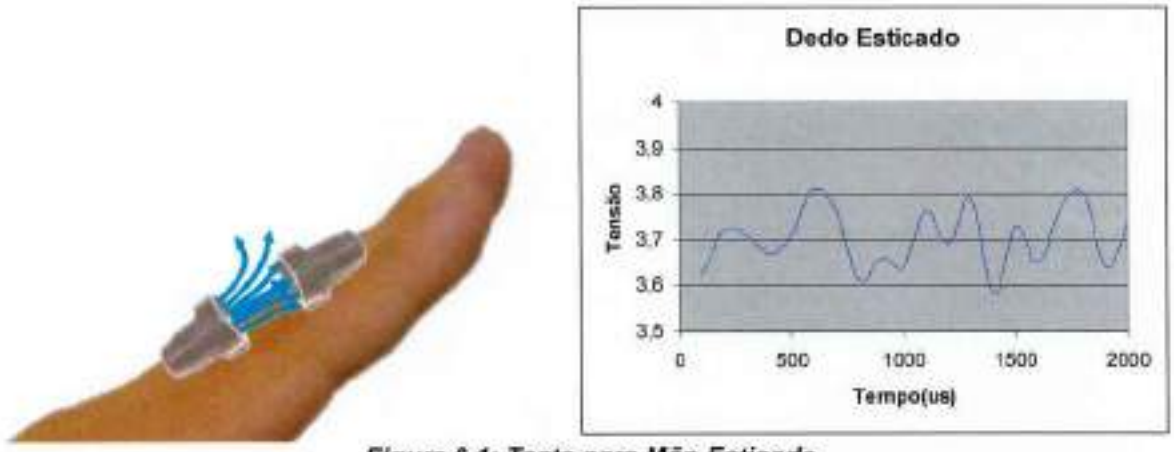


Figura 6.1: Teste para Mão Esticada

Na posição de Relaxado, sem que os dedos estejam tensionados, no estado natural de uma mão, o valor médio alcançado foi de 1,5 Volts, com uma variação de 33%.

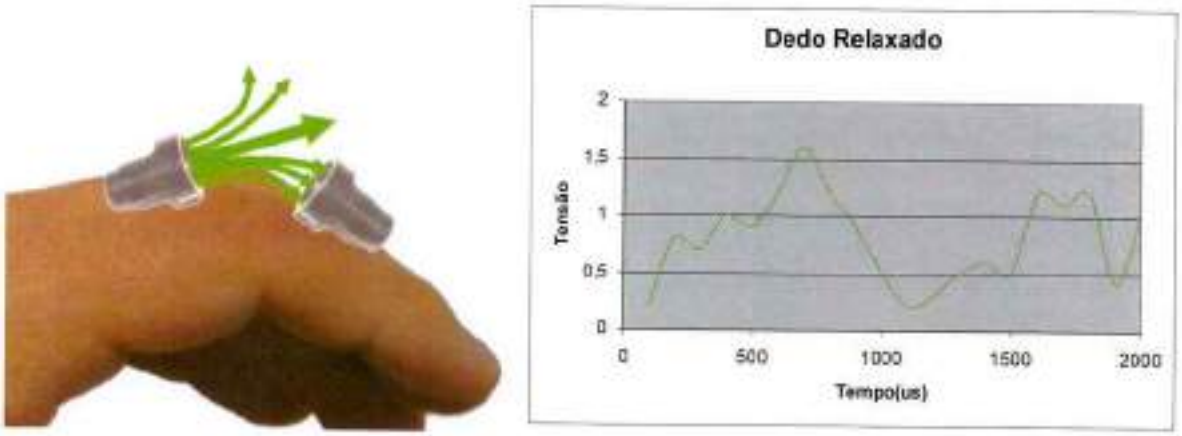


Figura 6.2: Teste para Mão Relaxada

Na posição de Contraído, o valor médio medido é de 0,05 Volts, com variação de 3%

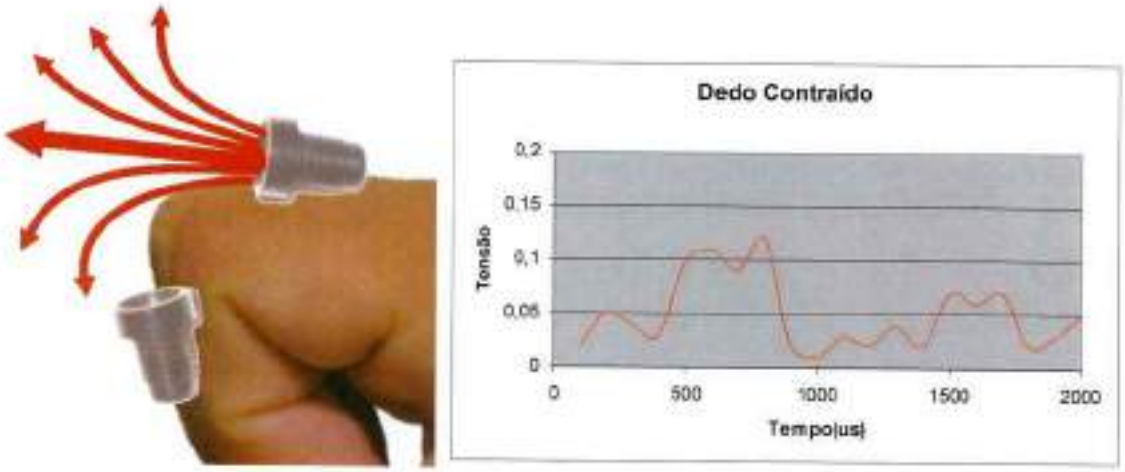


Figura 6.3: Teste para Mão Contraída

6.2Conversor

No conversor, testa-se a resolução e a faixa de valores obtida através dos sinais dos sensores.

Nessa fase, testou-se isoladamente o Conversor A/D, variando a tensão da faixa 0 Volts até 3,8 Volts, obtendo em média a conversão para os valores descritos na tabela abaixo:

Tensão Analógica	*Sinal Digital*
0V	001
1,5V	075
3,8V	138 e 175

Tabela 6.1: Teste de Conversão AD

6.3Detector + Conversor

Fazendo o acoplamento direto entre os dois primeiros subsistemas testados, podemos testar qual será a faixa de valores digitais convertidos.

A tabela a seguir mostra qual é a faixa de valores convertidos para cada estado estudado.

Estado	Média	Mínimo	Máximo
Esticado	3,7V	3,58V	3,81V
Relaxado	1,52V	1,41V	1,65V
Contraído	0,2V	0,02V	0,35V

Tabela 6.2: Valores do Sensor Óptico para os Estados do Dedo

6.4Tradutor e Executor

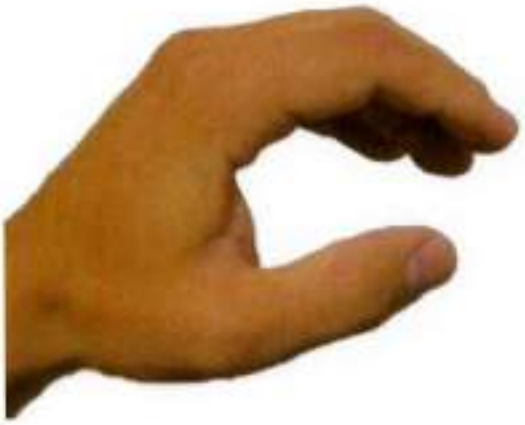
Como essa parte já envolve processamento de dados digitais (software), o teste foi realizado colocando como entrada um trem de valores a serem "traduzidos" e executados.

6.5HandTalks!

Integrando todas as partes testadas individualmente ou duas a duas, podemos testar facilmente o produto completo, executando, como teste a calibração dos estados para os gestos de uma pessoa.

Para isso, a pessoa entra no estado de calibração,

- executa o movimento de mão relaxada

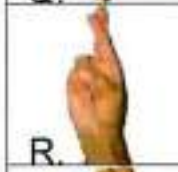


- executa o movimento de mão contraída

- executa o movimento de mão esticada.

Utilizando nossa base de dados instalada que prevê o seguinte mapeamento para cada letra:

Letra	Polegar	Indicador	Médio	Anelar	Mínimo	Contato	Contato
 A.	160 (esticado)	1 (contraído)	2 (contraído)	2 (contraído)	1 (contraído)	**D3, C2**	**1D, 2C, 3E, 4F**
 B.	1 (contraído)	140 (esticado)	146 (esticado)	157 (esticado)	170 (esticado)	**C2**	**3E, 4F**
 C.	60 (relaxado)	70 (relaxado)	64 (relaxado)	66 (relaxado)	72 (relaxado)	**C2**	**3E, 4F**
 D.	60 (relaxado)	157 (esticado)	72 (relaxado)	80 (relaxado)	64 (relaxado)	**B0**	**1C**
 E.	1 (contraído)	2 (contraído)	2 (contraído)	1 (contraído)	1 (contraído)	**C2**	**3E, 4F**
 F.	72 (relaxado)	80 (relaxado)	157 (esticado)	160 (esticado)	162 (esticado)	**D3**	**1D, 4F**
 G.	75 (relaxado)	162 (esticado)	2 (contraído)	1 (contraído)	1 (contraído)	**D3**	**1D, 2C**
 H.	93 (relaxado)	170 (esticado)	141 (esticado)	2 (contraído)	1 (contraído)	**B2**	**1E**

 I.	2 (contraído)	1 (contraído)	3 (contraído)	1 (contraído)	153 (esticado)	**C2**	**2C, 3E, 4F**
 J.	1 (contraído)	3 (contraído)	1 (contraído)	1 (contraído)	159 (esticado)	**C2**	**2C, 3E, 4F**
 K.	64 (relaxado)	162 (esticado)	158 (esticado)	1 (contraído)	1 (contraído)	**B2**	**1E**
 L.	158 (esticado)	172 (esticado)	1 (contraído)	3 (contraído)	1 (contraído)		**2C**
 M.	1 (contraído)	158 (esticado)	172 (esticado)	162 (esticado)	2 (contraído)	**C2**	**3E, 4F**
 N.	1 (contraído)	170 (esticado)	155 (esticado)	3 (contraído)	1 (contraído)	**C2**	**3E**
 O.	60 (relaxado)	70 (relaxado)	64 (relaxado)	66 (relaxado)	72 (relaxado)	**B0**	**1B ^ 1C**
 P.	64 (relaxado)	162 (esticado)	158 (esticado)	1 (contraído)	1 (contraído)	**B2**	**1E**
 Q.	75 (relaxado)	162 (esticado)	2 (contraído)	1 (contraído)	1 (contraído)	**D3,**	**1D, 2C**
 R.	2 (contraído)	160 (esticado)	153 (esticado)	1 (contraído)	1 (contraído)		
 S.	1 (contraído)	2 (contraído)	2 (contraído)	1 (contraído)	1 (contraído)	**C2**	**2C, 3E, 4F**
 T.	72 (relaxado)	80 (relaxado)	157 (esticado)	160 (esticado)	162 (esticado)		**4F**

U. 	2 (contraído)	160 (esticado)	153 (esticado)	1 (contraído)	1 (contraído)	**C2**	**3E**
V. 	1 (contraído)	162 (esticado)	158 (esticado)	1 (contraído)	1 (contraído)		
W. 	2 (contraído)	161 (esticado)	153 (esticado)	158 (esticado)	1 (contraído)		
X. 	1 (contraído)	2 (contraído)	2 (contraído)	1 (contraído)	1 (contraído)		**2C, 4F**
Y. 	155 (esticado)	1 (contraído)	3 (contraído)	1 (contraído)	153 (esticado)	**C2**	**2C, 3E, 4F**
Z. 	1 (contraído)	153 (esticado)	2 (contraído)	1 (contraído)	1 (contraído)	**C2**	**2C, 3E, 4F**

Tabela 6.3: Mapeamento dos Valores Médios Convertidos AD de cada Letra

7. Resultados Obtidos

A primeira etapa do projeto foi a tentativa de conseguir os recursos necessários, onde o grupo teve que reorientar suas escolhas devido a certas limitações e falta de apoio financeiro e de recursos a um projeto social, sem fins lucrativos. O grupo conseguiu reorganizar seu cronograma de atividades e replanejar todo o projeto para se adaptar à nova realidade e conseguir encaminhar o projeto.

Alguns componentes o próprio grupo tratou de adquirir particularmente, porém, como os custos de importação em pequenas quantidades é proibitivo, dado o alto valor do frete, o grupo enfrenta diversos problemas de ordem financeira para suportar o desenvolvimento do projeto, mesmo que sua produção em escala tenha um valor baixo. O grupo ainda aguarda certas respostas de propostas efetuadas a algumas empresas que poderiam se interessar pelo projeto, patrocinando o mesmo, ou pelo menos, os componentes.

Quanto à parte técnica do projeto, devido à dificuldade referente à aquisição de uma luva comercial pelo preço ou pela disponibilidade, foi decidido que a luva seria montada pelo grupo. A maior restrição que pode haver para sua implementação se refere ao custo dos sensores, já que alguns esquemas viáveis de montagem já foram propostos.

O resultado final obtido foi muito satisfatório. Tanto o módulo conversor como o módulo tradutor e executor funcionaram com sucesso. Apenas o módulo detector

não apresentou o funcionamento desejado, apesar de ter funcionado bem em termos de protótipo, tendo basicamente três problemas.

O primeiro problema foi o fato não conseguirmos adquirir todos os sensores necessários para resolver os conflitos de tradução. Para não deixar nenhuma letra sem detecção, contornamos esse problema alterando levemente os sinais de letras conflitantes. Por exemplo, para a letra J não conflitar com a letra I, consideramos que o dedo mínimo na primeira deve ficar relaxado, enquanto que na segunda deve ficar esticado. As letras alteradas foram J, N, P e Q.

Outro problema está nos sensores adotados. A utilização de infra-vermelho para a captação da dobra dos dedos não se mostrou muito eficiente pois não é capaz de detectar com suavidade a dobra. Para minimizar esse problema a variável relaxado da lógica *fuzzy* teve sua largura aumentada, deixando contraído e esticado apenas nas extremidades. Com isso ficou possível, porém ainda difícil, a tradução das letras C, R, S e X, além dos sinais que foram alterados devido ao primeiro problema.

O último refere-se ao posicionamente incorreto de alguns sensores de contato no momento da confecção da luva, o que impossibilitou a tradução das letras D e O.

Assim, concluímos que o projeto conseguiu traduzir com facilidade 61,5% das letras, traduziu com dificuldade 30,8% e não traduziu apenas 7,7%.

O projeto obteve êxito para os planos do grupo, pois funcionou adequadamente, com algumas restrições técnicas devido a orçamento e tempo hábil para a conclusão do mesmo. Permitiu ao grupo criar uma luva com sensores, cuja aplicabilidade se estende as várias esferas, não se restringindo apenas ao propósito desse projeto, podendo tornar-se, com algumas melhorias, uma ferramenta de realidade virtual de baixo custo, ou um dispositivo para manipulação de controles.

8. Considerações Finais

A princípio o grupo teve algumas dificuldades referentes aos próprios elementos que o constituem. O aluno Hugo Macedo Osawa foi o último elemento a compor o grupo, passando por um período de incerteza quanto sua integração efetiva, sendo que alguns aspectos do projeto e tomadas de decisão foram realizados nesta fase. O aluno assumiu que teve uma participação menor na primeira etapa de elaboração da especificação e se comprometeu a intensificar sua contribuição para realização do projeto.

Na parte do usuário, depois de muito verificar-se junto a especialistas e a entidades competentes, o produto, inicialmente gera muitas expectativas, que são, logo de início, frustradas, pois o usuário tem a errônea impressão de que esse projeto irá prontamente ajuda-lo a se comunicar. Depois sabendo do escopo, e do objetivo primário dele, o usuário á acha que o produto de nada servirá.

O que vale frisar é que o objetivo principal do projeto é ser um ponto de partida para algo que não foi feito no Brasil, uma iniciativa para estimular e conscientizar a necessidade de desenvolver mais projetos na área social. Isso é tão importante que recebemos uma Menção Honrosa na categoria Valor Social dentre os projetos de formatura das turmas de 2005.

9. Bibliografia

9.1 Componentes

Aplicações de um Bend Sensor

http://www.flexpoint.com/applications_v2.html

Diversos Tipos de Sensores.

http://generalrobotics.org/ppp/sensor.ppt

Preço do Bend Sensor

http://www.imagesco.com/catalog/flex/FlexSensors.html

Projetos com Bend Sensor Também..

http://www.makingthings.com/projects/cookbook/piezosensor.htm

RS do Brasil

http://www.rsdobrasil.com.br/index2.htm

The CyberShoe - Calçado do MIT com vários Sensores

http://www.media.mit.edu/physics/publications/papers/98.11.IDAT98_Shoe.pdf

Uniaxial Pattern: Micro-Measurements Strain Gages- Sensor de tensão unidirecional

http://www.vishay.com/brands/measurements_group/strain_gages/mmua.htm

9.2 LIBRAS

Língua Brasileira de Sinais

http://www.libras.org.br/

9.3 Luvas Caseiras

Cheap VR - cap. 2 - Usa tubo com led IR na ponta

http://www.phoenixgarage.org/projects/homevr/cheapvr/chpvr002.txt

Mellott's VR Page

http://www.geocities.com/mellott124/

9.4 Luvas Comerciais

P5 Glove

Da mesma fabricante da Powerglove do Nintendo

http://www.vrealities.com/P5.html

10. Anexos

Nesta seção estarão inclusos documentos essenciais ao projeto, tais como mapeamentos dos sensores, o estudo da Língua Brasileira de Sinais (LIBRAS), como o grupo organizou as tarefas e as alocou no tempo e alguns códigos-fonte fundamentais.

10.1 Divisão de Tarefas

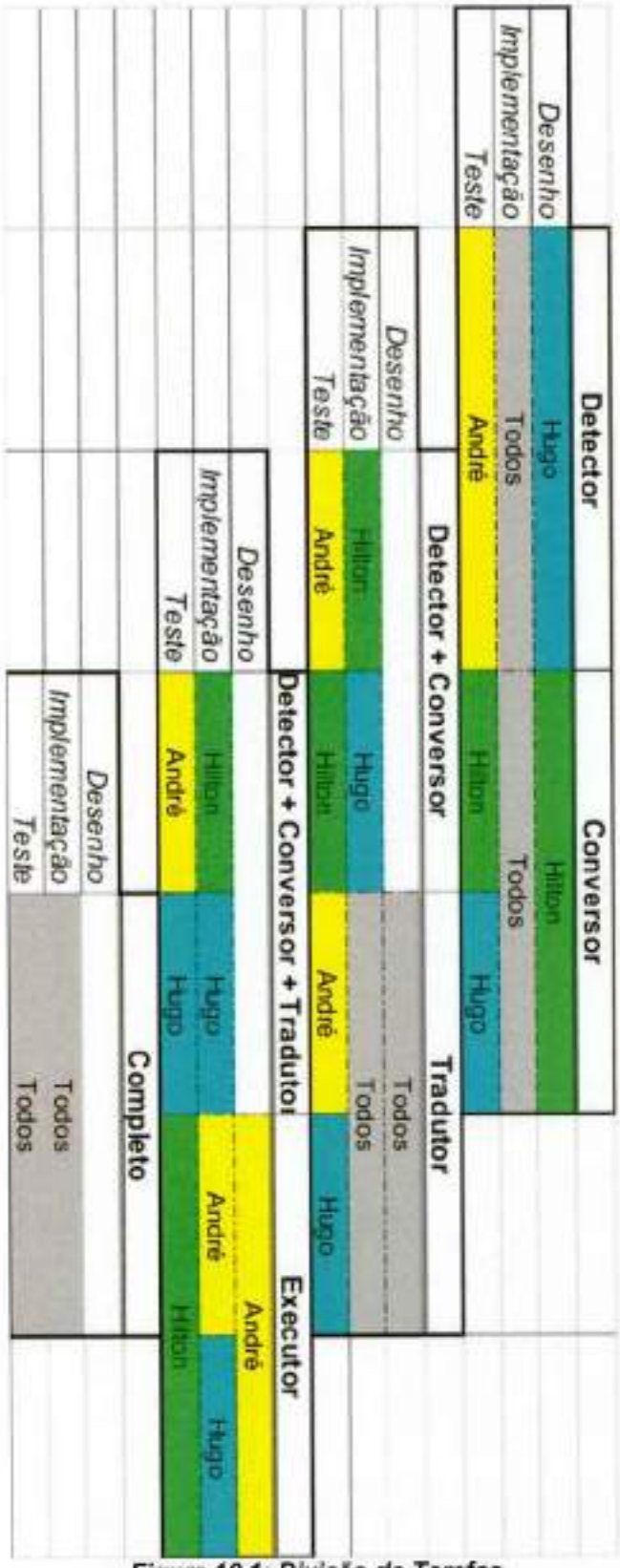


Figura 10.1: Divisão de Tarefas

10.2 Cronograma

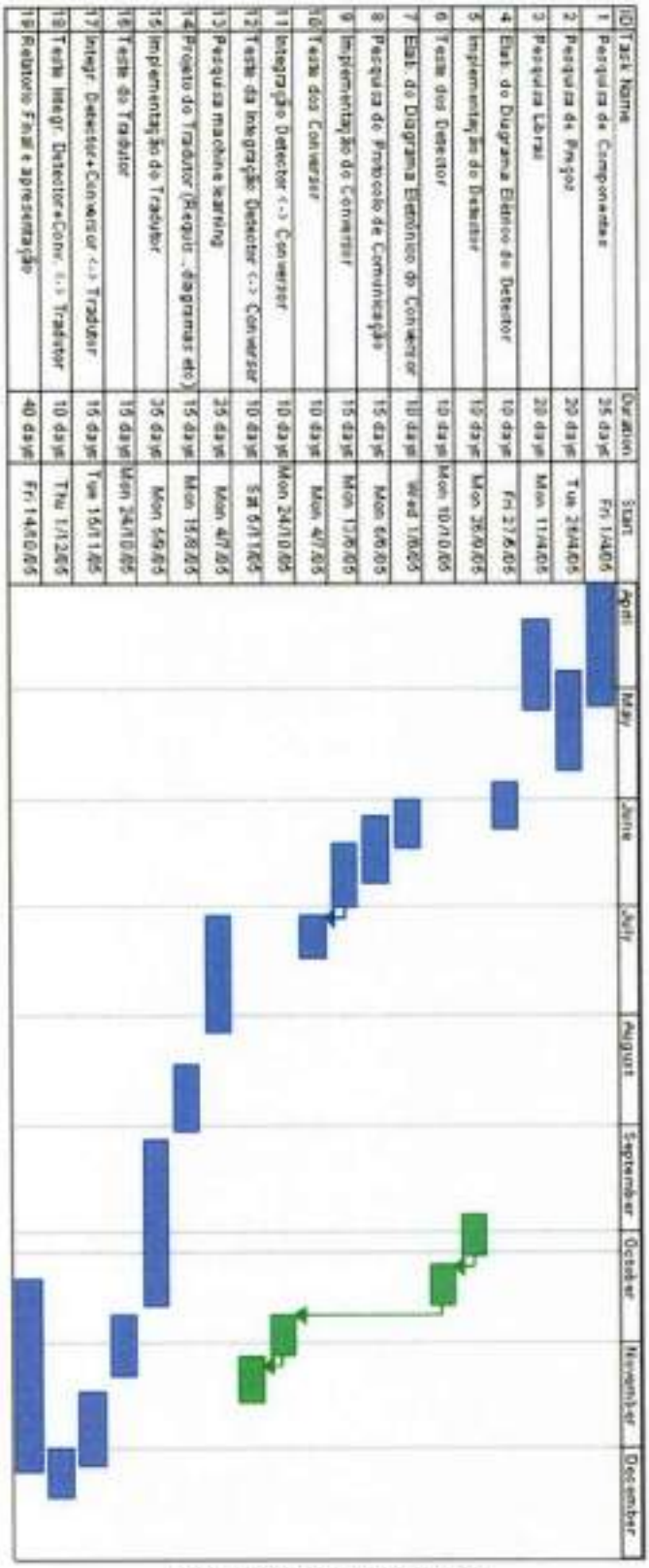


Figura 10.2: Cronograma

10.3 Tabela de Conflitos dos Sensores Básicos

Estado dos Dedos	Conflitos	Possível Solução do Conflito
• Todos os dedos semi-dobrados • Contato entre ponta do polegar e ponta do indicador e/ou médio	 (Zero)  (Letra O)	• Contextual (semântica)
• Polegar esticado • Demais dedos dobrados		• Detectar sentido do eixo da mão
• Polegar e indicador esticados • Demais dedos dobrados		• Detectar sentido do eixo da mão • Detectar contado do polegar com indicador
• Indicador, médio e anular esticados • Demais dedos dobrados		• Abertura do indicador e médio • Detectar sentido do eixo da mão
• Polegar dobrado • Demais dedos esticados		
• Indicador esticado ¾ • Demais dedos dobrados		• Médio semi-sobrado

Estado dos Dedos	*Conflitos*	*Possível Solução do Conflito*
• Todos os dedos dobrados		• Contextual (semântica)
• Polegar junto ao indicador • Demais dedos dobrados		
• Polegar Dobrado • Demais dedos esticados		
• Todos os dedos semi-dobrados (½)		• Detector de giro
• Indicador esticado • Demais dedos semi-dobrados • Contato entre as pontas do polegar e médio		
• Todos os dedos quase dobrados (¾)		

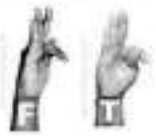
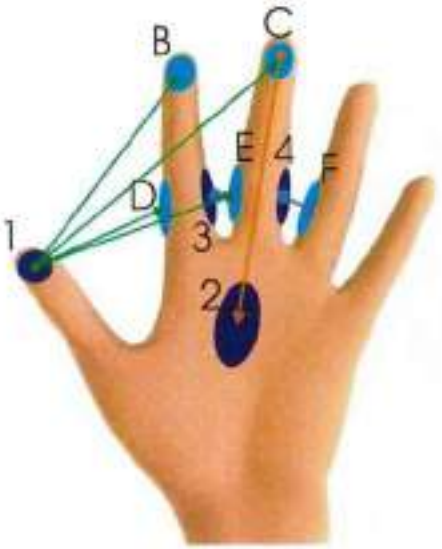
Estado dos Dedos	*Conflitos*	*Possível Solução do Conflito*
• Mínimo, anular, médio e polegar esticados • Indicador semi-dobrado • Contato da parte interna do polegar com parte externa do indicador		• Contato entre polegar e indicador
• Indicador e médio esticados • Demais dedos dobrados		• Abertura do indicador e médio • Contado do polegar c/ indicador • Contato entre indicador e médio • Detectar sentido do eixo da mão
• Mínimo esticado • Demais dedos dobrados		• Detectar sentido do eixo da mão
• Polegar e mínimo esticados • Demais dedos dobrados		
• Indicador esticado • Demais dedos dobrados		

Tabela 10.1: Conflitos Gerais de Tradução e Possíveis Soluções

10.4 Mapeamento dos Contatos da Luva



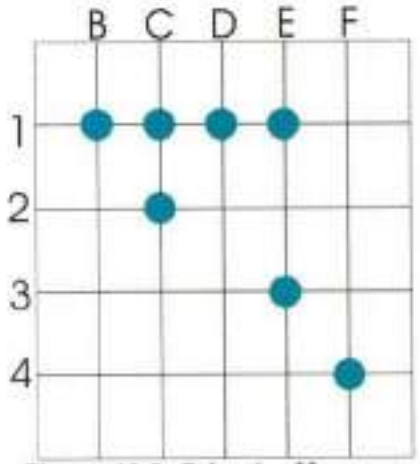
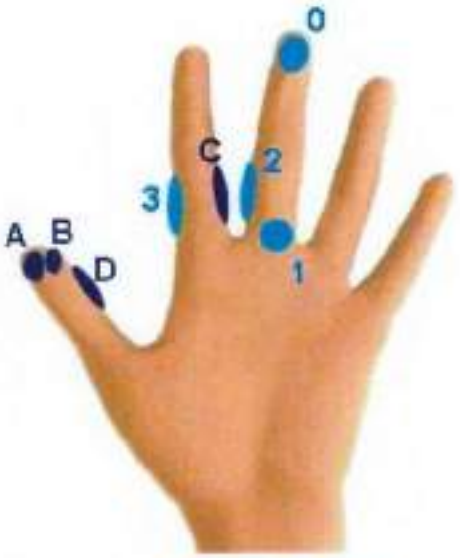


Figura 10.3: Primeiro Mapeamento de Contatos



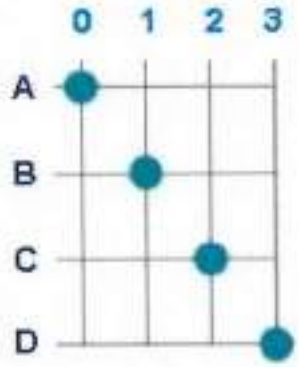


Figura 10.4: Segundo Mapeamento de Contatos

10.5 Mapeamento dos Sensores Resistivos e Ópticos

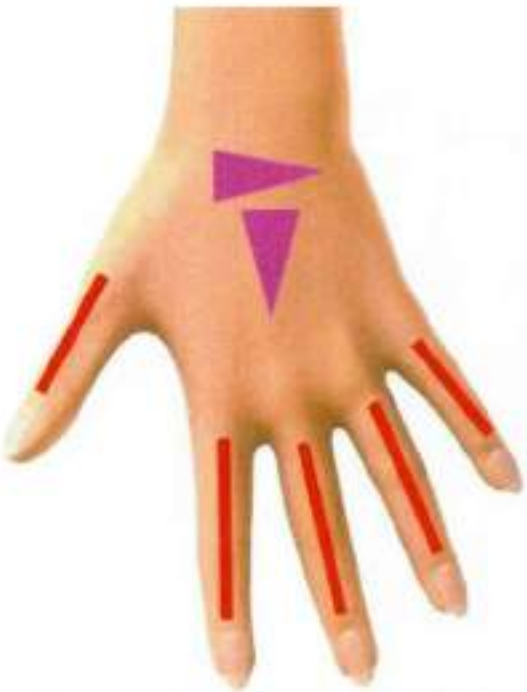


Figura 10.5: Primeiro Mapeamento de Sensores Resistivos (Flex)

Figura 10.6: Segundo Mapeamento de Sensores Ópticos

10.6 Tabela de Preço dos Sensores Pesquisados

O grupo pesquisou componentes que provavelmente seriam utilizados no projeto, tais como sensores "strain gauge", ou *flex sensors*, principais componentes da luva e seus custos. O projeto foi revisto e os Sensores Ópticos foram escolhidos

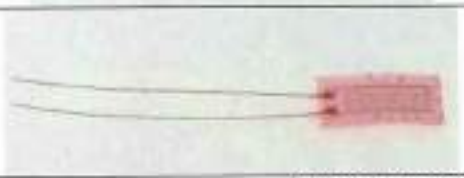
	Sensor	Preço
	Bend (Flex) Sensor	US$10.00
	Photo Emitter	US$1.27
	Photo Detector	US$0.77
	Tilt Sensor	US$2.80
	Angular Position Sensor	US$17.30
	Strain Gauge	US$3.49

Tabela 10.2: Preços dos Componentes

10.7 Tabela de Custos do Projeto

Item	Preço	Quantidade	Subtotal
Soquete torneado 16 pinos	R$ 0,75	1	R$ 0,75
Soquete torneado 40 pinos	R$ 1,80	1	R$ 1,80
Soquete torneado 14 pinos	R$ 0,65	2	R$ 1,30
Terminal molex	R$ 0,05	40	R$ 2,00
Conector molex macho (2 vias)	R$ 0,15	20	R$ 3,00
Conector molex fêmea (2 vias)	R$ 0,10	10	R$ 1,00
Conector macho (4 vias)	R$ 0,20	2	R$ 0,40
Conector fêmea (4 vias)	R$ 0,20	2	R$ 0,40
Terminal	R$ 0,05	8	R$ 0,40
CI PIC 16F877	R$ 39,00	1	R$ 39,00
CI MAX 232	R$ 3,00	1	R$ 3,00
CI LM 324	R$ 0,50	2	R$ 1,00
Capacitor Eletrolítico 1uFx16V	R$ 0,15	4	R$ 0,60
Cristal 4 MHz	R$ 1,00	1	R$ 1,00
TIL 78	R$ 1,50	5	R$ 7,50
TIL 32	R$ 0,30	5	R$ 1,50
Conector DB 9 fêmea	R$ 0,50	1	R$ 0,50
Placa universal dupla face (furo metalizado)	R$ 20,00	1	R$ 20,00
Resistores	R$ 0,05	13	R$ 0,65
Fio flexível (par)	R$ 0,40	6	R$ 2,40
Fio jumper	R$ 0,88	3	R$ 2,64
Luva	R$ 40,00	1	R$ 40,00
		Total	**R$ 130,84**

Tabela 10.3: Custo do Projeto

10.8 Tabela de Mapeamento dos Sensores para cada letra

Letra	Sensores
A	
	P 1D, 2C, 3E, 4F
B	
	IMAN 3E, 4F
C	
	PIMAN 3E, 4F
D	
	I 1C
E	
	3E, 4F
F	
	PIMAN 1D, 4F
G	
	PI 1D, 2C
H	
	PIM 1E
I	
	N 2C, 3E, 4F

Letra	Sensores
J	
	N 2C, 3E, 4F
K	
	PIM 1E
L	
	PI 2C
M	
	IMA 3E, 4F
N	
	IM 3E
O	
	1B ^ 1C
P	
	PIM 1E
Q	
	PI 1D, 2C
R	
	IM

Letra	Sensores
S	
	2C, 3E, 4F
T	
	PIMAN 4F
U	
	IM 3E
V	
	IM
W	
	IMA
X	
	2C, 4F
Y	
	PN 2C, 3E, 4F
Z	
	I 2C, 3E, 4F

Tabela 10.4: Mapeamento de Sensores para Cada Letra

10.9 Tabela de Conflitos

Sensor de Luz	Letra	Contatos	Solução
PIMAN	C	3E, 4F	
	F	1D, 4F	
	T	4F	
IMAN	B	3E, 4F	
PIM	H	1E	Parado
	K	1E	Movimento para Cima
	P	1E	Inclinação da Mão
IMA	M	3E, 4F	
	W		
PI	G	1D, 2C	Sensor Inclinação (Para Cima)
	L	2C	
	Q	1D, 2C	Sensor Inclinação (Para Baixo)
IM	N	3E	Sensor Óptico (Pulso Dobrado)
	R		Dedo Médio sobre Indicador
	U	3E	Sensor Óptico (Pulso Reto)
	V		Dedos Separados
PN	Y	2C, 3E, 4F	
P	A	1D, 2C, 3E, 4F	
I	D	1C	
	Z	2C, 3E, 4F	
N	I	2C, 3E, 4F	Dedo mínimo reto
	J	2C, 3E, 4F	Dedo mínimo curvo (dobrado)
	E	3E, 4F	
	O	1B ^ 1C	
	S	2C, 3E, 4F	
	X	2C, 4F	

Tabela 10.5: Conflitos da solução entre Letras e Possíveis Soluções

10.10 Circuito Elétrico

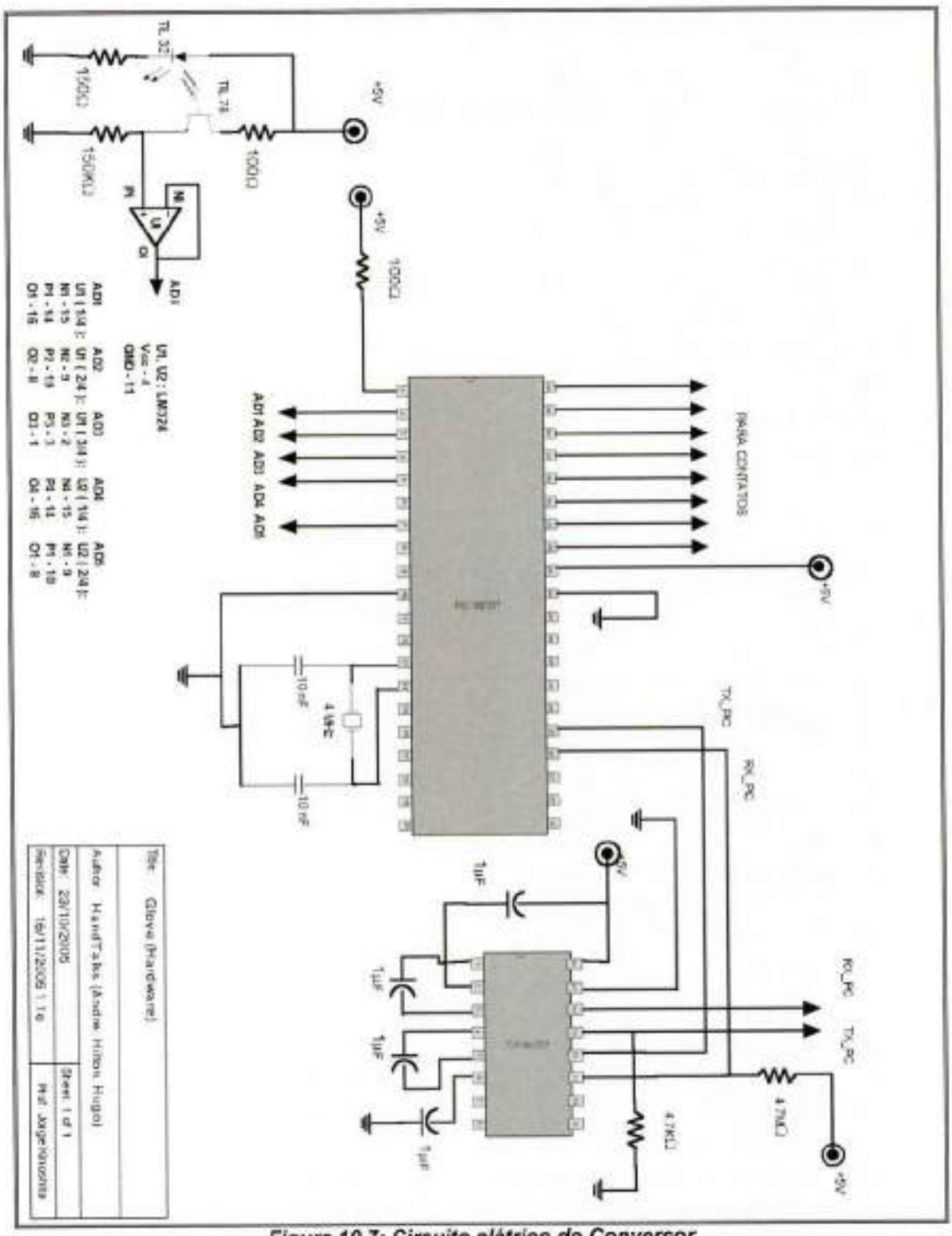


Figura 10.7: Circuito elétrico do Conversor

10.11 Códigos-Fonte Fundamentais

10.11.1 PIC

10.11.1.1 Arquivo projeto1b.h

```
#include <16F877.h>
#device adc=8
#use delay(clock=4000000)
#fuses NOWDT,XT, NOPUT, NOPROTECT, NOBROWNOUT, NOLVP, NOCPD, NOWRT, NODEBUG
#use rs232(baud=19200,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

10.11.1.2 Arquivo projeto_luva.c

```
#include "C:\Arquivos de programas\PICC\teste_proj\projeto1b.h"
  #include <ctype.h>
  #include <math.h>
  #include <stdio.h>
  #include <stddef.h>
  #include <stdlib.h>
  #include <stdlibm.h>
  #include <string.h>

/////////////////////////////////////////////////
Mapeamento:

Dedos:

Dobra

Polegar      - channel 0
Indicador    - channel 1
Medio        - channel 2
Anelar       - channel 3
Minimo       - channel 4


Contatos:

 - Port B


Os dados são convertidos para decimais ASC e enviados em um tren de caracteres:

char str_all_mapping[x]: String com dados capturados do mapeamento. E' enviada
                         via RS232

formato: $ch1#ch2#ch3#ch4#ch5#ctca##

- ch1 a ch5: 4 digitos cada (decimais ASCII, oriundos dos AD's)
- ctca: 1 digito (decimal ASCII, oriundo dos contatos)

/////////////////////////////////////////////////*/

#byte   PORTB  =  6
#define ALL_OUT 0 // port definido como saida
#define ALL_IN  0xff // port definido como entrada
#define n_samples 3 // numero de amostras p/ validacao de um valor AD
```

```
 int x;
 char contatos[4]; // variaveis para leitura dos contatos
 char contatos_aux[4];
 char char_contatos; // variavel para auxilio a conversao dos contatos

/*****************************************************************
*                       Funcao para leitura conversao em decimais
*
*****************************************************************/

void bin_to_decimal( long int c, char *buffer) {
        buffer[3] = (c%10) + '0';
        c/=10;
        buffer[2] = (c%10) + '0';
        c/=10;
        buffer[1] = (c%10) + '0';
        c/=10;
        buffer[0]= (c%10) + '0';
        buffer[4] ='\0';

}

/*****************************************************************
*                       Funcao para leitura dos contatos
*
*****************************************************************/

void contacts_deboucing(){
        char i;
        char trisbp[4] = {0x7F,0xBF,0xDF,0xEF}; //uma linha como saida em cada
iter.
   #pragma use fast_io(B)
   #pragma use fast_io(D)
        for (i=0;i<4;i++){
                set_tris_b(trisbp[i]); //troca qual linha que sera a saida
      output_b(0x00);
      contatos_aux[i] = input_b();
      contatos_aux[i] = contatos_aux[i] & 0x0F; //
      output_d(contatos_aux[i]);
      output_b(0xFF);
                set_tris_b(0xFF); //antes de chavear, faz todos como entrada.
        }
      char_contatos=0;
        for (i=0;i<4;i++){
            if(!bit_test(contatos_aux[i],3-i)){ // bit_test e bit_set: LSbit
               bit_set(char_contatos,i);
            }
        }
      if((char_contatos>=0)&&(char_contatos<=9)){
         char_contatos=char_contatos+'0';
      }
      else if((char_contatos>9)&&(char_contatos<=15)){
         char_contatos=char_contatos+'7';
      }
}

/*****************************************************************
*                       Funcao para leitura dos canais AD
*
*****************************************************************/

void AD_sampling(char channel, char *channel_string){
```

```
   long int value_aux, value;
   int i;
   value_aux = 0;

   set_adc_channel(channel);
   for(i=0; i<n_samples; ++i) { // 5 amostras p/ pegar a media
      value = Read_ADC();
      value_aux += value;
      delay_us(20);
   }
   value_aux = value_aux / n_samples; // e' feita a media
   bin_to_decimal(value_aux, channel_string); // e converte-se para decimal
}


/*****************************************************************
*                                                        Principal
*
*****************************************************************/

void main()
{
   char channel; // canal AD
   char str_terminator[3];
   char str_channel[5]; // string auxiliar p/ leitura de um canal AD
   unsigned char contacts; // caractere que mapeia os contatos
   char str_contacts[2];
   unsigned char contacts_aux;
   char str_all_mapping[50]; // string que contatena todos os canais e o
                             // caractere de contatos para envio via RS232

   //configuracao dos portas
   set_tris_b(ALL_IN);
   port_b_pullups(TRUE);

   //configuracao do AD
   setup_adc_ports(AN0_AN1_AN2_AN3_AN4);
   setup_adc(ADC_CLOCK_INTERNAL);

   //configuracao de recursos de comunicacao
   setup_psp(PSP_DISABLED);
   setup_spi(FALSE);

   //configuracao dos timers
   setup_timer_0(RTCC_INTERNAL|RTCC_DIV_1);
   setup_timer_1(T1_DISABLED);
   setup_timer_2(T2_DISABLED,0,1);


   //definicao do caracter de fim de campo
   strcpy(str_terminator, "#");
   while(1){
      str_all_mapping[0]='\0';
      strcpy(str_all_mapping, "0"); // inicio da string
          char_contatos=0;
      for(channel = 0; channel < 5; channel++){ // varredura dos canais
                AD_sampling(channel, str_channel);
         strcat(str_all_mapping, str_channel);
         strcat(str_all_mapping, str_terminator); //concatenacao da string
      }
      contacts_deboucing(); //leitura dos contatos
      str_contacts[0]=char_contatos;
```

```
        str_contacts[1]='\0';
        strcat(str_all_mapping, str_contacts);
        strcat(str_all_mapping, str_terminator);
        strcat(str_all_mapping, str_terminator); //finalizacao da string
        printf("%s", str_all_mapping);
        delay_ms(2);
    }
}
```

10.11.2 PC

10.11.2.1 Arquivo HTTranslator.py

```
#!/usr/bin/env python
# -*- coding: latin-1 -*-

class HTTranslator (object):
    """ Classe responsável pela tradução de um dado sinal da luva.

    O sinal corresponde a duas tuplas, uma com os valores dos sensores
    AD de cada dedo, e outra com os valores booleanos dos contatos.

    fingers = [polegar, indicador, médio, anelar, mínimo]
    contacts = [polegar_indicador,
                polegar_medio_ponta, polegar_medio_base,
                indicador_medio]
    """

    # Constantes da Lógica Fuzzy
    INPUT_VARIABLES = 3
    CONTRACTED, RELAXED, STRAINED = range(INPUT_VARIABLES)

    # Mapeamento das letras
    MAPPING = {
              'A': {'fingers': [STRAINED, CONTRACTED, CONTRACTED, CONTRACTED,
CONTRACTED],
                    'contacts': [False, False, True, True]},
              'B': {'fingers': [CONTRACTED, STRAINED, STRAINED, STRAINED, STRAINED],
                    'contacts': [False, False, True, False]},
              'C': {'fingers': [STRAINED, RELAXED, RELAXED, RELAXED, RELAXED],
                    'contacts': [False, False, True, False]},
              'D': {'fingers': [STRAINED, STRAINED, RELAXED, RELAXED, RELAXED],
                    'contacts': [True, False, False, False]},
              'E': {'fingers': [CONTRACTED, CONTRACTED, CONTRACTED, CONTRACTED,
CONTRACTED],
                    'contacts': [False, False, True, False]},
              'F': {'fingers': [STRAINED, RELAXED, STRAINED, STRAINED, STRAINED],
                    'contacts': [False, False, False, True]},
              'G': {'fingers': [RELAXED, STRAINED, CONTRACTED, CONTRACTED,
CONTRACTED],
                    'contacts': [False, False, False, True]},
              'H': {'fingers': [STRAINED, STRAINED, STRAINED, CONTRACTED,
CONTRACTED],
                    'contacts': [False, True, True, False]},
              'I': {'fingers': [CONTRACTED, CONTRACTED, CONTRACTED, CONTRACTED,
STRAINED],
                    'contacts': [False, False, True, False]},
              'J': {'fingers': [CONTRACTED, CONTRACTED, CONTRACTED, CONTRACTED,
RELAXED],
                    'contacts': [False, False, True, False]},
```

```
            'F': {'fingers': [STRAINED, STRAINED, STRAINED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, True, False, False]},
            'L': {'fingers': [STRAINED, STRAINED, CONTRACTED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, False, False]},
            'M': {'fingers': [CONTRACTED, STRAINED, STRAINED, STRAINED,
CONTRACTED],
                  'contacts': [False, False, True, False]},
            'N': {'fingers': [CONTRACTED, STRAINED, STRAINED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, True, True]},
            'O': {'fingers': [RELAXED, RELAXED, RELAXED, RELAXED, RELAXED],
                  'contacts': [True, False, True, False]},
            'P': {'fingers': [STRAINED, STRAINED, RELAXED, CONTRACTED, CONTRACTED],
                  'contacts': [False, True, False, False]},
            'Q': {'fingers': [RELAXED, RELAXED, CONTRACTED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, False, True]},
            'R': {'fingers': [RELAXED, STRAINED, RELAXED, CONTRACTED, CONTRACTED],
                  'contacts': [False, False, True, False]},
            'S': {'fingers': [RELAXED, CONTRACTED, CONTRACTED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, True, False]},
            'T': {'fingers': [RELAXED, RELAXED, STRAINED, STRAINED, STRAINED],
                  'contacts': [False, False, False, False]},
            'U': {'fingers': [CONTRACTED, STRAINED, STRAINED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, True, False]},
            'V': {'fingers': [CONTRACTED, STRAINED, STRAINED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, False, False]},
            'W': {'fingers': [RELAXED, STRAINED, STRAINED, STRAINED, CONTRACTED],
                  'contacts': [False, False, False, False]},
            'X': {'fingers': [RELAXED, RELAXED, CONTRACTED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, False, False]},
            'Y': {'fingers': [STRAINED, CONTRACTED, CONTRACTED, CONTRACTED,
STRAINED],
                  'contacts': [False, False, True, False]},
            'Z': {'fingers': [RELAXED, STRAINED, CONTRACTED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, False, False]},
            'VI': {'fingers': [RELAXED, CONTRACTED, CONTRACTED, STRAINED,
STRAINED],
                  'contacts': [False, False, True, False]},
            'PT': {'fingers': [STRAINED, STRAINED, STRAINED, STRAINED, STRAINED],
                  'contacts': [True, False, True, False]},
            'EX': {'fingers': [STRAINED, STRAINED, STRAINED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, True, False]},
            'IN': {'fingers': [STRAINED, STRAINED, STRAINED, CONTRACTED,
CONTRACTED],
                  'contacts': [False, False, False, False]},
            'SP': {'fingers': [RELAXED, STRAINED, STRAINED, STRAINED, STRAINED],
                  'contacts': [False, False, True, True]},
            'BE': {'fingers': [STRAINED, STRAINED, STRAINED, STRAINED, STRAINED],
                  'contacts': [False, False, True, False]},
            'CR': {'fingers': [STRAINED, STRAINED, CONTRACTED, CONTRACTED,
STRAINED],
                  'contacts': [False, False, False, False]}
    } # MAPPING
```

```
    def __init__ (self):
        """ Método de inicialização dos atributos
        """

        # Ajuste da lógica fuzzy
        self.strained = None     # esticado
        self.relaxed = None      # relaxado
        self.contracted = None   # contraído

        # Valor atual dos sensores
        self.fingers = None
        self.contacts = None

        # Zera pesos de entrada
        self.__u_in = [ [0.0 for finger in range(5)]
                        for input in range (self.INPUT_VARIABLES) ]

        # Saída
        self.__result = None

    # __init__()


    def getResult (self):
        """ Retorna o resultado (letra) da última tradução efetuada.
        """
        return self.__result

    def getUIn (self):
        """ Retorna pesos de entrada da última tradução efetuada.
        """
        return self.__u_in

    result = property (getResult, None, None, "Resultado (letra) da última
tradução")
    u_in = property (getUIn, None, None, "Pesos de entrada da última tradução")


    def adjust (self, strained=None, relaxed=None, contracted=None):
        """ Ajusta os patamares da lógica fuzzy.
        Cada parâmetro é uma tupla com cinco valores, um para cada dedo
        """

        # TODO: Verificar consistência dos valores
        if strained is not None:
            self.strained = tuple ([ float(x) for x in strained ])
        if relaxed is not None:
            self.relaxed = tuple ([ float(x) for x in relaxed ])
        if contracted is not None:
            self.contracted = tuple ([ float(x) for x in contracted ])

    # adjust()


    def translate (self, fingers=None, contacts=None):
        """ Método que efetivamente faz a tradução do sinal de entrada numa
        letra, utilizando lógica fuzzy.
        """

        if fingers is not None:
```

```
            self.fingers = fingers
        if contacts is not None:
            self.contacts = contacts

#         print "Dedos:", self.fingers
#         print "Contatos:", self.contacts

        # Calcula o peso de cada variável de entrada (contraido etc.)
        for finger in range(5):
            x1 = self.contracted[finger]
            x2 = x1 + 20
            x3 = x2 + 40
            x4 = self.strained[finger]

            if self.fingers[finger] <= x1:
                self.__u_in[self.CONTRACTED][finger] = 1.0
                self.__u_in[self.RELAXED]   [finger] = 0.0
                self.__u_in[self.STRAINED]  [finger] = 0.0

            elif x1 < self.fingers[finger] <= x2:
                self.__u_in[self.CONTRACTED][finger] =((self.fingers[finger] -x2) /
                                                       (x1                      - x2))
                self.__u_in[self.RELAXED]   [finger] = 1.0 -
self.__u_in[self.CONTRACTED][finger]
                self.__u_in[self.STRAINED]  [finger] = 0.0

            elif x2 < self.fingers[finger] <= x3:
                self.__u_in[self.CONTRACTED][finger] = 0.0
                self.__u_in[self.RELAXED]   [finger] = 1.0
                self.__u_in[self.STRAINED]  [finger] = 0.0

            elif x3 < self.fingers[finger] <= x4:
                self.__u_in[self.CONTRACTED][finger] = 0.0
                self.__u_in[self.RELAXED]   [finger] =((self.fingers[finger] -x4) /
                                                       (x3                      - x4))
                self.__u_in[self.STRAINED]  [finger] = 1.0 -
self.__u_in[self.RELAXED][finger]

            elif x4 < self.fingers[finger]:
                self.__u_in[self.CONTRACTED][finger] = 0.0
                self.__u_in[self.RELAXED]   [finger] = 0.0
                self.__u_in[self.STRAINED]  [finger] = 1.0


        # Apenas exibe
#         print u"Pesos de entrada:"
#         NOMES = [u'Contraido', 'Relaxado', 'Esticado']
#         for inputs in range (self.INPUT_VARIABLES):
#             print "%s:" % NOMES[inputs],
#             for valor in self.__u_in[inputs]:
#                 print "\t%d%%" % (valor*100),
#             print
#         print

#         print u"Pesos de saida:"

        # Calcula o peso de cada variável de saida (letras)
        u_out = {}
        max_u_out = [0.0, 0.0]
        out_letter = [None, None]
        for letter in self.MAPPING:
            # Ignora letras que não coincidiram pelos contatos
```

```
            if self.MAPPING[letter]['contacts'] == self.contacts:
#                print "%s:\t" % letter,
                u_out [letter] = 0.0

                for finger in range(5):
                    finger_input = self.MAPPING[letter]['fingers'][finger]
                    finger_weight = self.__u_in [finger_input][finger]
#                    print "\t%d%%" % (finger_weight*100),
                    u_out [letter] += finger_weight

                u_out [letter] /= 5.0
#                print "\t=%d%%" % (u_out[letter]*100)

                if u_out [letter] > max_u_out[0]:
                    max_u_out[1] = max_u_out[0]
                    max_u_out[0] = u_out [letter]
                    out_letter[1] = out_letter[0]
                    out_letter[0] = letter
                elif u_out [letter] > max_u_out[1]:
                    max_u_out[1] = u_out [letter]
                    out_letter[1] = letter


        # Chance maior de 70%
        if max_u_out[0] > 0.7:
            # Checa se deu empate em 5%
            advantage = (max_u_out[0] - max_u_out[1]) / max_u_out[0]
            if advantage <= 0.05:
#                print u"Empate entre %s e %s" % tuple(out_letter)
                out_letter[0] = None
#            else:
#                print u"Maximo: %s, %d%%" % (out_letter[0], max_u_out[0]*100)
        else:
            out_letter[0] = None
#            print u"Nenhuma letra"

#        print

        self.__result = out_letter[0]
        return self.__result
    # translate()


# Código de teste da classe HTTranslator
if __name__ == "__main__":
    contraido = [1, 2, 1, 1, 2]
    relaxado = [45, 30, 42, 58, 41]
    esticado = [165, 170, 172, 168, 171]

    tradutor = HTTranslator ()
    tradutor.adjust (esticado, relaxado, contraido)

    print "\n** letra A **"
    dedos = [150, 4, 6, 3, 1]
    contatos = [False, False, True, True]
    tradutor.translate (dedos, contatos)
```

11. Apêndices

Neste capítulo encontram-se materiais complementares para o entendimento de detalhes de escolhas no projeto, tais como especificações de componentes e textos gerais.

11.1 Datasheet PIC 16F87X

MICROCHIP

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz

Pin Diagram

PDIP

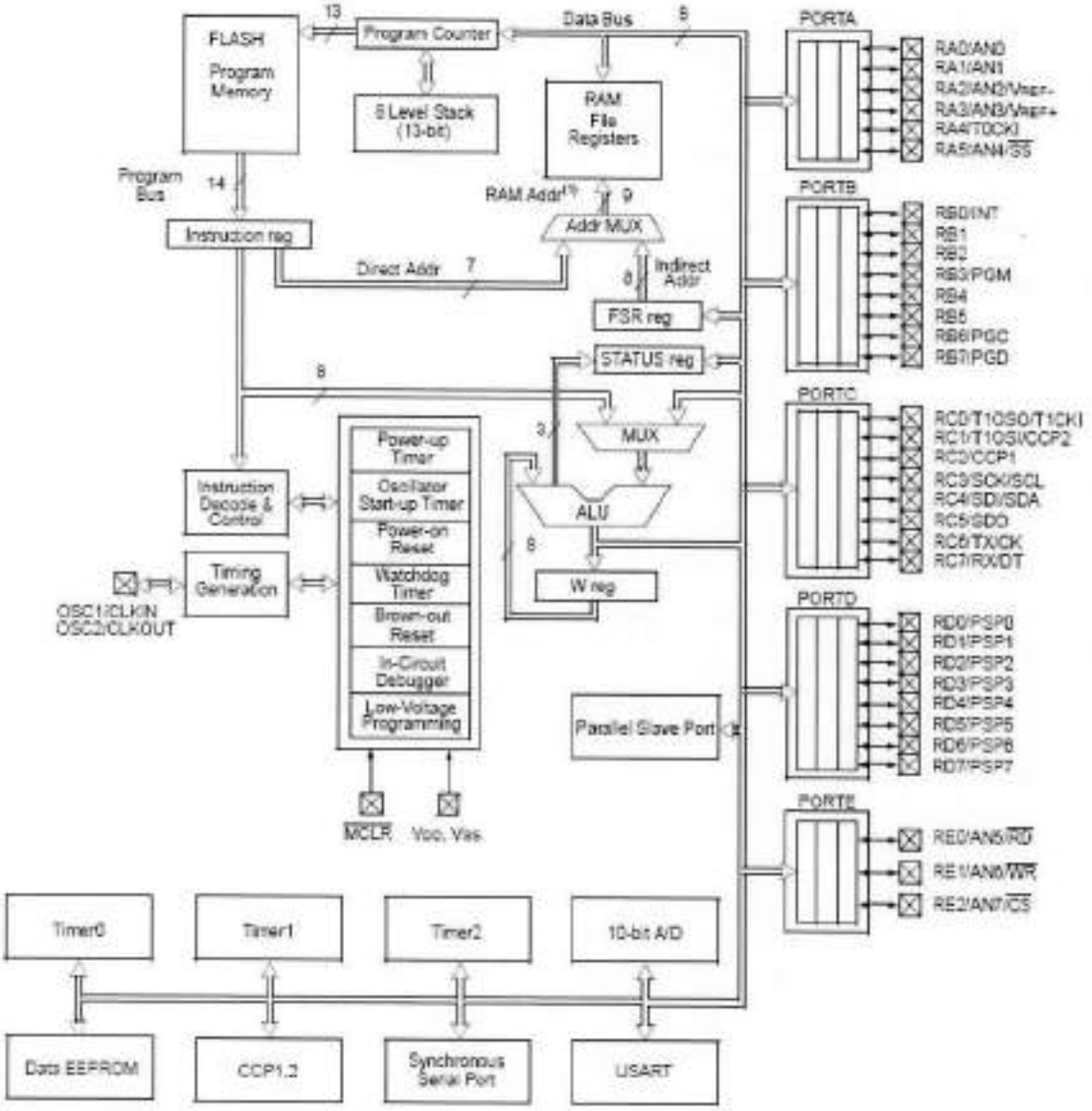
Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I²C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

Key Features PICmicro™ Mid-Range Reference Manual (DS33023)	PIC16F873	PIC16F874	PIC16F876	PIC16F877
Operating Frequency	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz	DC - 20 MHz
RESETS (and Delays)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)	POR, BOR (PWRT, OST)
FLASH Program Memory (14-bit words)	4K	4K	8K	8K
Data Memory (bytes)	192	192	368	368
EEPROM Data Memory	128	128	256	256
Interrupts	13	14	13	14
I/O Ports	Ports A,B,C	Ports A,B,C,D,E	Ports A,B,C	Ports A,B,C,D,E
Timers	3	3	3	3
Capture/Compare/PWM Modules	2	2	2	2
Serial Communications	MSSP, USART	MSSP, USART	MSSP, USART	MSSP, USART
Parallel Communications	—	PSP	—	PSP
10-bit Analog-to-Digital Module	5 input channels	8 input channels	5 input channels	8 input channels
Instruction Set	35 instructions	35 instructions	35 instructions	35 instructions

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes



Note 1: Higher order bits are from the STATUS register.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/ counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	PORTC is a bi-directional I/O port. RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/ output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	PORTE is a bi-directional I/O port. RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
Vss	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
VDD	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

11.2 Datasheet DS14C232

National Semiconductor

February 1996

DS14C232
Low Power +5V Powered TIA/EIA-232 Dual Driver/Receiver

General Description

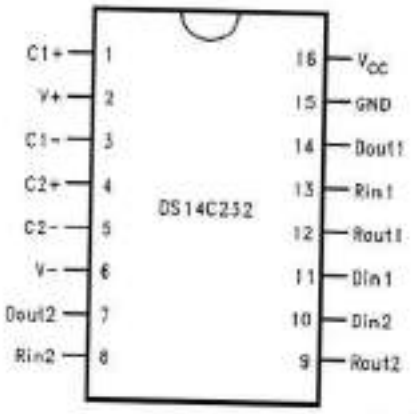
The DS14C232 is a low power dual driver/receiver featuring an onboard DC to DC converter, eliminating the need for ±12V power supplies. The device only requires a +5V power supply. I_{CC} is specified at 3.0 mA maximum, making the device ideal for battery and power conscious applications. The drivers' slew rate is set internally and the receivers feature internal noise filtering, eliminating the need for external slew rate and filter capacitors. The device is designed to interface data terminal equipment (DTE) with data circuit-terminating equipment (DCE). The driver inputs and receiver outputs are TTL and CMOS compatible. DS14C232C driver outputs and receiver inputs meet TIA/EIA-232-E (RS-232) and CCITT V.28 standards.

Features

- Pin compatible with industry standard MAX232, LT1081, ICL232 and TSC232
- Single +5V power supply
- Low power—I_{CC} 3.0 mA maximum
- DS14C232C meets TIA/EIA-232-E (RS-232) and CCITT V.28 standards
- CMOS technology
- Receiver Noise Filter
- Package efficiency—2 drivers and 2 receivers
- Available in Plastic DIP, Narrow and Wide SOIC packages
- TIA/EIA-232 compatible extended temperature range options:

DS14C232T	−40°C to +85°C
DS14C232	−55°C to +125°C

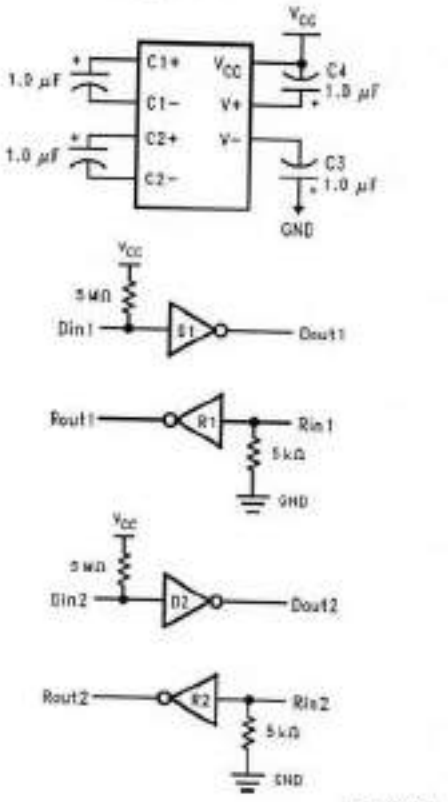
Connection Diagrams



TL/F/10744–1

Order Number DS14C232CN, DS14C232TN, DS14C232CM, DS14C232TM, DS14C232CWM or DS14C232TWM
See NS Package Number N16A, M16A or M16B

Functional Diagram



TL/F/10744–2

11.3Datasheet LM324

National Semiconductor

December 1994

LM124/LM224/LM324/LM2902 Low Power Quad Operational Amplifiers

General Description

The LM124 series consists of four independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

Application areas include transducer amplifiers, DC gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM124 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional ±15V power supplies.

Unique Characteristics

- In the linear mode the input common-mode voltage range includes ground and the output voltage can also swing to ground, even though operated from only a single power supply voltage
- The unity gain cross frequency is temperature compensated
- The input bias current is also temperature compensated

Advantages

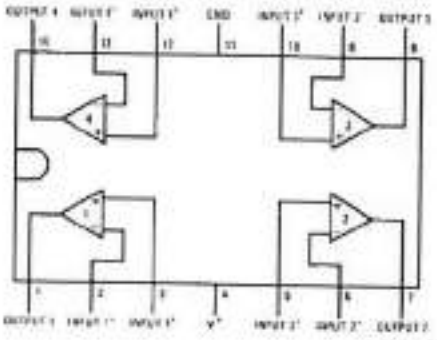
- Eliminates need for dual supplies
- Four internally compensated op amps in a single package
- Allows directly sensing near GND and V_{OUT} also goes to GND
- Compatible with all forms of logic
- Power drain suitable for battery operation

Features

- Internally frequency compensated for unity gain
- Large DC voltage gain 100 dB
- Wide bandwidth (unity gain) 1 MHz (temperature compensated)
- Wide power supply range:
  - Single supply 3V to 32V
  - or dual supplies ±1.5V to ±16V
- Very low supply current drain (700 μA)—essentially independent of supply voltage
- Low input biasing current 45 nA (temperature compensated)
- Low input offset voltage 2 mV and offset current 5 nA
- Input common-mode voltage range includes ground
- Differential input voltage range equal to the power supply voltage
- Large output voltage swing 0V to V+ − 1.5V

Connection Diagram

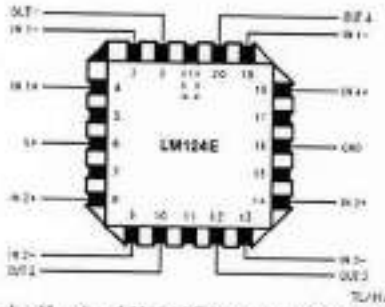
Dual-In-Line Package



TL/H/3239–1

Top View

Order Number LM124J, LM124AJ, LM124J/883, LM124AJ/883*, LM224J, LM224AJ, LM324J, LM324M, LM324AM, LM2902M, LM324N, LM324AN or LM2902N**
See NS Package Number J14A, M14A or N14A



TL/H/3239–32

Order Number LM124AE/883 or LM124E/883
See NS Package Number E20A

TL/H/3239–33

Order Number LM124AW/883 or LM124W/883
See NS Package Number W14B

*LM124A available per JM38510/11006
**LM124 available per JM38510/11005