

**UNIVERSIDADE LUTERANA DO BRASIL**  
**CENTRO DE TECNOLOGIA E INFORMÁTICA**  
**CURSO DE ENGENHARIA ELÉTRICA**

**DESENVOLVIMENTO DE UM SISTEMA DE  
ELETROESTIMULAÇÃO TÁTIL PARA AUXÍLIO À  
LOCOMOÇÃO DE DEFICIENTES VISUAIS**

**Marcelo Wiggers Borges**

**Canoas, Julho de 2007**

**MARCELO WIGGERS BORGES**

**DESENVOLVIMENTO DE UM SISTEMA DE  
ELETROESTIMULAÇÃO TÁTIL PARA AUXÍLIO À  
LOCOMOÇÃO DE DEFICIENTES VISUAIS**

**Trabalho de Conclusão de Curso  
apresentado como requisito para a  
obtenção do grau de Engenheiro  
Eletricista pela Universidade Luterana  
do Brasil – Campus Canoas - Curso de  
Engenharia Elétrica.**

Orientador: Prof. Dr. Alexandre Balbinot

Canoas, Julho de 2007.

**MARCELO WIGGERS BORGES**

**DESENVOLVIMENTO DE UM SISTEMA DE  
ELETROESTIMULAÇÃO TÁTIL PARA AUXÍLIO À  
LOCOMOÇÃO DE DEFICIENTES VISUAIS**

**BANCA AVALIADORA:**

Profa. M.E. Carla Diniz Lopes \_\_\_\_\_

Prof. Dr. Valner João Brusamarello \_\_\_\_\_

**ORIENTADOR:**

Prof. Dr. Alexandre Balbinot \_\_\_\_\_

Trabalho apresentado e aprovado em:

## **Agradecimentos**

Ao meu professor orientador, Alexandre Balbinot, pela inestimável ajuda no empréstimo de referências bibliográficas, pelas sugestões ao longo do desenvolvimento do presente estudo, e acima de tudo pelo incentivo e confiança no meu trabalho.

A família por sempre estar presente nos momentos mais difíceis.

Aos amigos por compreender a ausência que se iniciou juntamente ao meu ingresso na engenharia.

E não vos conformei a este mundo, mas transformai-vos  
pela renovação da vossa mente.

*Romanos 12:2*

## RESUMO

O objetivo desse trabalho visa adquirir imagens selecionadas com o uso de uma câmera *webcam* e fazer o reconhecimento das bordas, a fim de iniciar os estudos de substituição sensorial voltado para pessoas portadoras de deficiência visual, através de eletroestimulação cutânea. Para tanto, optou-se por utilizar o *software* Matlab® como plataforma para efetuar os métodos de processamento da imagem digital e fazê-lo comunicar-se com um microcontrolador da família PIC, que é o responsável por controlar a estimulação. A imagem adquirida tem sua resolução diminuída para 64 *pixels* e é enviada serialmente através de um circuito FTDI para a placa UCP, onde o microcontrolador redireciona os dados para as placas de chaveamento. Nas placas são conectados os terminais de uma cinta de eletrodos ordenados de forma matricial, desenvolvida para ser alojada na região abdominal, de modo a representar o resultado do tratamento da imagem utilizando estimulação por corrente elétrica. Com os ensaios realizados no sistema foi possível ajustar a intensidade do sinal para níveis seguros sem comprometer a estimulação, e formas simples puderam ser representadas nos eletrodos, sendo restringidas apenas pela técnica de reconhecimento das bordas para a baixa resolução de 8x8.

Palavras-chave: Tecnologia Assistiva, processamento, comunicação serial, Matlab®, eletroestimulação

## ABSTRACT

The purpose of this work consist in acquiring selected images from a webcam and recognize the edges, in order to start the sensorial substitution studies, benefiting visual deficient people through cutaneous electrical stimulation. Such development is implemented in Matlab® software, as a platform to realize the digital image processing and to make it communicates with the microcontroller PIC that is responsible for controlling the stimulation. The acquired image has its resolution lowed to 64 pixels so it is sending by serial data transmission across a FTDI circuit to the UCP board, where the microcontroller redirects the information to the switchboards. On boards are connected the electrodes that are disposed in matrix on a belt, that was developed to be arranged in the abdominal region in way to represent the result of image processing, using electrical current in stimulation. In previous tests were possible to adjust the signal intensity to safe levels without compromising the stimulation so, simple forms could be represented in the electrodes, being restricted only by the edges recognition technique to the low resolution of 8x8.

**Key-words:** Assistive technology, image processing, serial communication, Matlab®, electrical stimulation.

## SUMÁRIO

<b>1</b>	<b>INTRODUÇÃO .....</b>	<b>9</b>
<b>2</b>	<b>REVISÃO BIBLIOGRÁFICA.....</b>	<b>11</b>
2.1	REPRESENTAÇÃO DE IMAGENS DIGITAIS .....	11
2.2	FORMATOS DE IMAGENS .....	13
2.2.1	Imagem Binária .....	13
2.2.2	Imagem RGB .....	14
2.2.1	Imagem em Tons de Cinza .....	16
2.3	AQUISIÇÃO DE IMAGENS .....	16
2.3.1	Introdução aos dispositivos de Tubo de Raios Catódicos - TRC .....	17
2.3.2	Introdução aos dispositivos <i>Charge Coupled Device</i> - CCD .....	19
2.4	PASSOS FUNDAMENTAIS EM PROCESSAMENTO DE IMAGENS.....	22
2.5	TECNOLOGIA ASSISTIVA - TA.....	26
<b>3</b>	<b>APARATO EXPERIMENTAL .....</b>	<b>28</b>
3.1	AQUISIÇÃO DA IMAGEM DIGITAL .....	29
3.1.1	Reconhecimento da <i>Webcam</i> .....	30
3.1.2	Criando Objeto de Entrada de Vídeo.....	32
3.1.3	Script M-File.....	35



3.2	PROCESSAMENTO DA IMAGEM DIGITAL .....	36
3.2.1	Conversão entre tipos de imagem.....	37
3.2.2	Detecção das Bordas.....	42
3.3	COMUNICAÇÃO DO PC COM O MICROCONTROLADOR .....	44
3.3.1	Reconhecimento da Serial pelo Matlab®.....	47
3.3.2	Convertendo a Matriz binária para Caractere.....	48
3.4	CONFECÇÃO DO <i>HARDWARE</i> PARA ELETROESTIMULAÇÃO.....	50
3.4.1	Placa UCP.....	50
3.4.2	Placas de Chaveamento .....	60
3.4.3	Matriz de Monitoramento .....	65
3.4.4	Matriz de Eletrodos .....	66
<b>4</b>	<b>RESULTADOS E DISCUSÕES .....</b>	<b>71</b>
4.1	ENSAIOS DE CAPTURA E CHAVEAMENTO .....	72
4.1.1	Ensaio de figuras circulares .....	73
4.1.2	Ensaio de colunas .....	76
4.2	SINAL USADO PARA ELETROESTIMULAÇÃO .....	81
<b>5</b>	<b>CONCLUSÕES .....</b>	<b>84</b>
<b>6</b>	<b>SUGESTÕES E MELHORIAS .....</b>	<b>86</b>
<b>7</b>	<b>REFERÊNCIAS BIBLIOGRÁFICAS.....</b>	<b>88</b>

## LISTA DE FIGURAS

Figura 2.1 – Convenção dos eixos para representação de imagens digitais.....	12
Figura 2.2 – Imagem Digital e sua matriz representativa.....	14
Figura 2.3 – Processo aditivo das cores primárias.....	15
Figura 2.4 – Tubo de raios catódicos de tecnologia vidicon.....	18
Figura 2.5 – Sequência de varredura do feixe de elétrons no TRC.....	18
Figura 2.6 – Matriz CCD de 512 x 512 <i>pixels</i> .....	22
Figura 2.7 – Passos fundamentais em processamento de imagens.....	25
Figura 3.1 – Diagrama de blocos do sistema desenvolvido.....	28
Figura 3.2 – Câmera <i>webcam</i> GoTec 3810.....	30
Figura 3.3 – Espaço de trabalho do Matlab®.....	33
Figura 3.4 – <i>Preview</i> do Matlab®.....	35
Figura 3.5 – Imagem RGB.....	39
Figura 3.6 – Imagem Convertida para <i>Grayscale</i> .....	39
Figura 3.7 – Ponto de limiarização de uma imagem grayscale.....	41
Figura 3.8 – Imagem convertida para binário.....	42

Figura 3.9 – Circuito de conversão USB para RS-232.....	45
Figura 3.10 – Detalhe do <i>chip</i> FT232R e da entrada USB.....	46
Figura 3.11 – Placa UCP em detalhe.....	51
Figura 3.12 – Circuito do PIC18F4620.....	52
Figura 3.13 – Representação dos pinos de saída da placa UCP.....	53
Figura 3.14 – Fluxograma do <i>software</i> do PIC para a identificação dos dados.....	58
Figura 3.15 – Circuito para eletroestimulação.....	62
Figura 3.16 – Circuito da três fontes reguladas por placa de chaveamento.....	63
Figura 3.17 – Placa de chaveamento.....	64
Figura 3.18 – Matriz de monitoramento.....	65
Figura 3.19 – Circuito da matriz de monitoramento.....	66
Figura 3.20 – Eletrodo de ECG.....	67
Figura 3.21 – Cinta com a matriz de eletrodos vista frontal.....	68
Figura 3.22 – Matriz de eletrodos vista de trás.....	69
Figura 4.1 – Visão do sistema.....	72
Figura 4.2 – Imagem do círculo adquirido.....	73
Figura 4.3 – Imagem do círculo convertido para <i>grayscale</i> .....	74
Figura 4.4 – Imagem do círculo convertido para binário.....	74
Figura 4.5 – Resultado do círculo em detalhe.....	75
Figura 4.6 – Visão geral do experimento com o círculo.....	75
Figura 4.7 – Processamento de uma coluna vertical.....	76
Figura 4.8 – Representação de uma reta vertical na matriz de monitoramento.....	77
Figura 4.9 – Processamento de uma coluna horizontal.....	78
Figura 4.10 – Representação de uma reta horizontal na matriz de monitoramento.....	78

Figura 4.11 – Processamento de uma coluna interrompida na metade da tela.....	79
Figura 4.12 – Representação de uma coluna interrompida na matriz de led`s.....	80
Figura 4.13 – Ensaio com a cinta de eletrodos.....	82
Figura 4.14 – Sinal da estimulação.....	83

## LISTA DE ABREVIATURAS E SIGLAS

PPDV – Pessoas Portadoras de Deficiência Visual

PC – *Personal Computer* – Computador Pessoal.

PDI – *Digital Image Processing* – Processamento Digital de Imagem.

CCD – *Charge Coupled Devices* – Dispositivo de Carga Acoplado.

ADA – *American With Disabilities Act*

USB – *Universal Serial Bus* – Barramento Universal Serial.

TA – Tecnologia Assistiva.

SIPO – *Serial Input Parallel Output* – Entrada Serial, Saída Paralela.

## 1 INTRODUÇÃO

A dificuldade em montar uma representação mental de ambientes desconhecidos é um dos problemas enfrentados por pessoas portadores de deficiência visual (PPDV), dificultando a sua mobilidade. Segundo Pereira (2005), existem instituições que utilizam técnicas especiais de locomoção baseado em treinamento e busca exaustiva do ambiente que estão voltados para o auxílio dessas pessoas, mas a relação de dependência a terceiros permanece inalterada.

Neste trabalho, foi desenvolvido um sistema que busca uma maior independência de PPDV. Esse sistema captura uma imagem que se encontra a frente do usuário, e repassa as bordas de objetos em primeiro plano para o usuário através de excitação elétrica na região do abdômen, fazendo uso de uma cinta configurada com uma matriz de eletrodos.

O sistema somestésico (responsável pelo tato) possui uma densidade de receptores nervosos e representação cortical muito inferior ao olho humano, tornando-o mais limitado no reconhecimento de contornos (Kandel, Schwartz e Jessel, 2002). Para aguçar o estímulo e

diminuir os efeitos dessa limitação, nesse trabalho as imagens capturadas são primeiramente processadas digitalmente por um PC, utilizando o do *software* Matlab®. Esse programa executa as rotinas de processamento digital com técnicas não lineares e rotinas de filtros espaciais, de modo a diminuir a sua resolução e posteriormente fazer o reconhecimento dos contornos.

O objetivo principal desse trabalho é, após a aquisição da imagem, efetuar o processamento digital para converter seu formato e detectar bordas de figuras simples como círculos e retas e realizar a estimulação cutânea na região de abdômen. A eletroestimulação tátil é feita usando-se uma cinta flexível confeccionada para esse fim, onde eletrodos são arranjados matricialmente e interligados a circuitos de chaveamento.

Uma representação análoga à imagem captada pela câmera será estimulada na cinta de eletrodos de maneira que uma PPDV possa, depois de passar por um período de treinamento, fazer o reconhecimento de contornos de objetos a sua frente. Testes devem ser feitos para encontrar os melhores níveis de tensão a ser aplicada ao estímulo.

A tecnologia assistiva desenvolvida neste trabalho não pretende substituir a visão humana, mas busca auxiliar a vida do PPDV através de um método seguro, por não ser necessária intervenção cirúrgica e prático por ser portátil e poder ser retirado para a prática de outras atividades ou no descanso.

## 2 REVISÃO BIBLIOGRÁFICA

### 2.1 REPRESENTAÇÃO DE IMAGENS DIGITAIS

Segundo Gonzalez e Woods (2002), o termo imagem monocromática refere-se à função bidimensional de intensidade de luz  $f(x, y)$ , onde os argumentos indicam as coordenadas espaciais, e o valor de  $f$  em qualquer ponto  $(x, y)$  representa o brilho (tons de cinza) da imagem naquele ponto. Uma imagem digital adota a mesma função  $f(x, y)$  como modelo de representação, porém de maneira discretizada tanto em coordenadas espaciais quanto em brilho. A Figura 3.1 mostra a convenção mais adotada por autores acerca dos eixos.



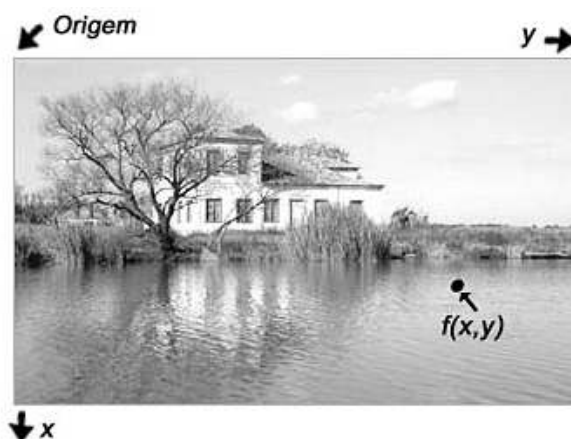


Figura 2.1 – Convenção dos eixos para representação de imagens digitais.

Fonte: Gonzalez e Woods, 2000.

Para que qualquer computador possa interpretar uma imagem, a mesma deve ser primeiro armazenada de forma satisfatória para que possa ser manipulada por um programa computacional. O modo mais prático de fazer isto é dividir a imagem em uma coleção de valores discretos (e normalmente pequenos) que são conhecidos como *elementos da imagem* ou *elementos da figura*. Geralmente a imagem é dividida em uma grade retangular, de forma que cada elemento é um retângulo pequeno. Uma vez feito isso, para cada um é determinado um valor que representa a sua tonalidade. É assumido que o elemento inteiro tem a mesma tonalidade, e assim qualquer variação de cor que existiu dentro dessa área antes da imagem ser discretizada está perdida. Porém, se a área de cada elemento for muito pequena, então a natureza discreta da imagem não é frequentemente visível ao olho humano (Gonzalez e Woods, 2000).

Pode-se compreender, então, uma imagem digital como uma matriz onde os argumentos  $x$  e  $y$  indicam os índices de linha e coluna respectivamente, identificando assim o endereço de um ponto da imagem. Do Inglês a expressão *picture elements*, cuja abreviação *pixels* é largamente adotada, inclusive em posteriores explicações ao longo dessa documentação.

## 2.2 FORMATOS DE IMAGENS

### 2.2.1 Imagem Binária

Imagens binárias são imagens cujos *pixels* tem apenas dois possíveis valores de intensidade. Normalmente são exibidos como preto e branco. Numericamente, os dois valores são freqüentemente 0 para preto e 1 para branco.

Imagens binárias são produzidas freqüentemente através de filtros “*thresholding*”, tendo como entrada imagens em tons de cinza “*grayscale*” ou RGB para separar um objeto em primeiro plano do fundo da imagem. O objeto de interesse normalmente tem seus *pixels* definidos em um valor não-zero de modo a tornar suas formas na cor branca, enquanto o fundo fica em 0 para ser exibido na tonalidade preta. A polaridade pode ser invertida de acordo com a necessidade. Como por exemplo, a Figura 2.2.

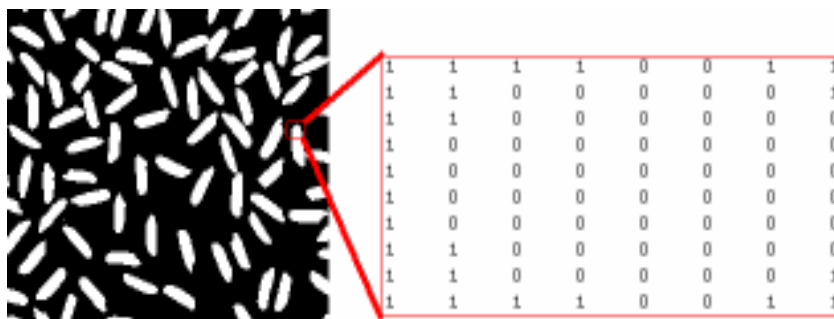


Figura 2.2 – Imagem Digital e sua matriz representativa.

Fonte: *Image Acquisition Toolbox* do Matlab®.

### 2.2.2 Imagem RGB

Uma cor percebida pelo olho humano pode ser definida por uma combinação linear de três cores primárias: vermelho (*red*), verde (*green*) e azul (*blue*). Estas três cores formam a base para o *RGB-colourspace*. Por este motivo, o modelo RGB é um sistema aditivo de cor, onde uma determinada cor é obtida somando-se diferentes proporções das cores primárias, emitidas por fontes de luz. O modelo aditivo de cor tem como característica a geração do preto quando não há nenhuma fonte de luz é emitida.

Se duas cores primárias forem somadas em iguais proporções haverá o surgimento das chamadas cores secundárias. Assim, a fonte de luz vermelha somada com a fonte de luz verde em igual proporção produz luz amarela; a fonte de luz verde com a fonte de luz azul produz

luz ciano e a fonte de luz vermelha com a fonte de luz azul produz luz magenta. A Figura 2.3 ilustra esse processo.



Figura 2.3 – Processo aditivo das cores primárias.

Fonte: Azevedo e Conci, 2003.

Cada elemento de cor da matriz RGB pode ser definido por um vetor no espaço de cor tridimensional. A intensidade é determinada pelo comprimento do vetor, e a cor atual pelos dois ângulos que descrevem a orientação do vetor no espaço de cor.

Uma imagem RGB também pode ser transformada em outros sistemas de coordenadas que poderiam ser mais úteis para algumas aplicações. Uma base comum para o espaço de cor é o IHS. Neste sistema de coordenada, uma tonalidade é descrita com sua intensidade, cor (comprimento de onda comum) e saturação (a quantia de branco).

O sistema de cores RGB foi tema da presente revisão bibliográfica por ser o sistema considerado pelo *software* Matlab® em um primeiro momento, após a aquisição da imagem pela câmera.

### **2.2.1 Imagem em Tons de Cinza**

A grande razão do destaque conferido as imagens em tons de cinza, conhecida por *grayscale* ou *graylevel*, diferenciando-as de qualquer outro tipo de imagem de cor é que menos informação precisa ser fornecida para cada *pixel*. Na realidade, uma imagem em *grayscale* tem seu componente vermelho, verde e azul com a mesma intensidade no espaço RGB, e assim só é necessário especificar um único valor de intensidade por *pixel*, ao invés das três intensidades em uma imagem de cor.

## **2.3 AQUISIÇÃO DE IMAGENS**

A etapa de aquisição de imagem consiste na captura de uma representação da informação visual, que deve ser fidedigna ao evento e em condições de ser processada por um computador. Normalmente essas representações são montadas a partir de fontes de radiação como calor, raios-X e microondas, e absorvidas por sensores sensíveis a esse tipo de radiação (Gonzalez e Woods, 2000).

Os sensores de imagem atuam como transdutores por converter a informação visual incidente em sinais elétricos que constituem a representação da imagem. Existem duas tecnologias que se destacam no mercado utilizadas em equipamentos de captura de imagens: Tubo de raios catódicos (TRC) e *Charge Coupled Device* (CCD).

### **2.3.1 Introdução aos dispositivos de Tubo de Raios Catódicos - TRC**

O princípio de funcionamento de um (TRC) consiste basicamente em um tubo de vidro com um canhão de elétrons acoplado em uma das extremidades, grades para ajuste dos feixes de elétrons, bobinas para a orientação eletromagnética das partículas e uma placa feita de material fotoemissor ou fotocondutor localizada no extremo oposto do canhão.

A variação da tecnologia de vídeo da câmera TRC é definida pela escolha da característica do material da citada placa; a tecnologia *Orticon* usa o material fotoemissor enquanto a *Vidicon* e *Pumblicon* o fotocondutor (Beê, R.T, 1999). Para elucidar o funcionamento básico do dispositivo TRC foi utilizado como exemplo o tubo Vidicon, esquematizado na Figura 2.4.

A placa fotocondutora do dispositivo recebe a radiação de luz oriunda do objeto cuja imagem deve ser captada, devido às suas propriedades sofre variação de resistência elétrica em toda a sua extensão de maneira proporcional as variações de incidência de luz, formando uma imagem resistiva do objeto alvo. A Figura 2.5 mostra a sequência de varredura de um feixe de elétrons.

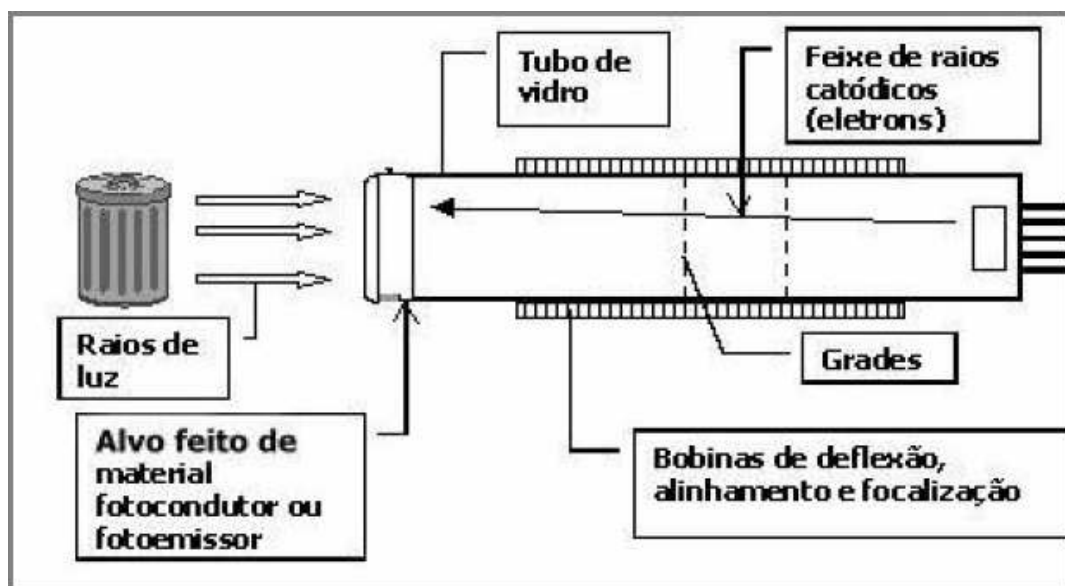


Figura 2.4 – Tubo de raios catódicos de tecnologia vidicon.

Fonte: Beê, R.T, 1999.

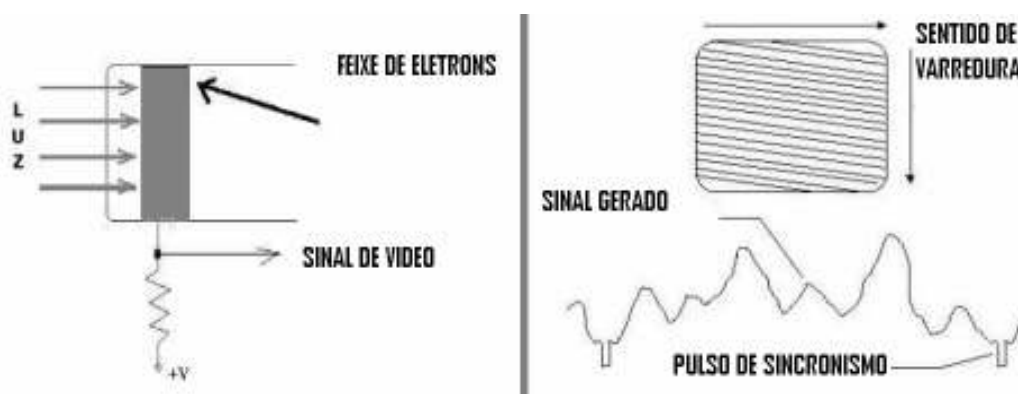


Figura 2.5 – Sequência de varredura do feixe de elétrons no TRC.

Fonte: Beê, R.T, 1999.

A outra face da placa é varrida por feixes de elétrons que divide sua extensão em 525 linhas horizontais, estabelecendo uma corrente elétrica para cada ponto encontrado sobre ela em função de sua resistência elétrica. Por fim, o desejado sinal de vídeo será a composição final dos sinais de tensão elétrica obtidos pela queda de potencial da corrente sobre um resistor de carga.

### **2.3.2 Introdução aos dispositivos *Charge Coupled Device* - CCD**

A tecnologia CCD foi desenvolvida em 1970 nos laboratórios Bell por Boyle e Smith, e é um dispositivo semicondutor capaz de converter sinais de luz visível em sinais elétrico, registrando a imagem focada por uma câmera de vídeo sem a necessidade de utilizar feixes de elétrons.

No princípio o CCD foi desenvolvido para ser utilizado como um *chip* de memória e não de imagem. Mas no decorrer da evolução dos semicondutores houve a descoberta de melhores tecnologias para o armazenamento de dados como, na época, o EEPROM (Electrically Erasable Programmable Read Only Memory), fazendo com que esses chips se tornassem ultrapassados para essa aplicação.

Devido à facilidade em transferência de cargas elétricas, começaram a ser aplicados para o registro de imagens formadas em uma matriz de pixels. Ligados a essa matriz (formada de pequenas regiões chamadas *fotossítios*), os chips poderiam efetuar com extrema velocidade a



transferência individual das cargas concentradas em cada elemento, efetuando a leitura indireta da imagem projetada pelas lentes de uma câmera. Comprovado o excelente desempenho, passou a ser empregado nas câmeras de vídeo no lugar dos antigos tubos de imagem.

### ***2.3.2.1 Princípio de funcionamento***

Os semicondutores têm seus elétrons segmentados em camadas chamadas de bandas, e para cada banda há elétrons que possuem certa energia característica. A banda mais baixa é conhecida como banda de valência e é onde a maioria dos elétrons encontra-se, a de maior energia é chamada banda de condução. Com a ação da absorção de fótons, os elétrons de baixa energia podem ser excitados e saltar para bandas de maior energia, deixando lacunas que passam a funcionar praticamente como cargas positivas.

O desejado é o “salto” para a banda de condução, o que só ocorre com um ganho entre 1,5 e 5 eV; com energia maior que 5 eV há o surgimento de múltiplos pares de elétrons/lacunas e esse processo torna-se cada vez mais intenso até os 10 keV de energia, quando o Silício (Si) praticamente não sofre mais interação com os fótons, devido ao curto comprimento de onda da luz nessa faixa de energia. Os elétrons excitados pela energia incidente tendem a decair a banda de valência após algum período de tempo, o que no CCD não acontece devido ao uso de um campo elétrico (Felber, P, 2002).

Em uma câmera, uma lente converge à radiação luminosa para uma determinada região conhecida como *focal*, onde existe um CI que possui um CCD. Uma imagem fotográfica é formada por distintas áreas, claras e escuras, e ao se projetar sobre a região focal, este receberá mais fótons, outros menos, outros quase nenhuma, de acordo com o desenho da imagem.

Como o CCD é formado por uma matriz de fotossítios (cada um deles é capaz de manifestar o fenômeno do “salto” entre bandas), o padrão da luz que o atinge forma um padrão correspondente de cargas por elemento. Se, em dado instante, cada fotossítio tiver a intensidade de corrente que está gerando medida e anotada, e existir um dispositivo que recebendo determinada intensidade de corrente brilhe de acordo com esta intensidade, será possível reproduzir a imagem em um aparelho contendo milhares destes dispositivos, dispostos da mesma maneira que os fotossítios no CCD. Cabe observar que os CCD's para fins específicos de captação de imagens em infravermelho são confeccionados de Germânio (Ge) e para as demais câmeras o semiconductor é o Silício (Si). A Figura 2.6 mostra o diagrama de uma típica matriz CCD de 512 x 512 *pixels*.

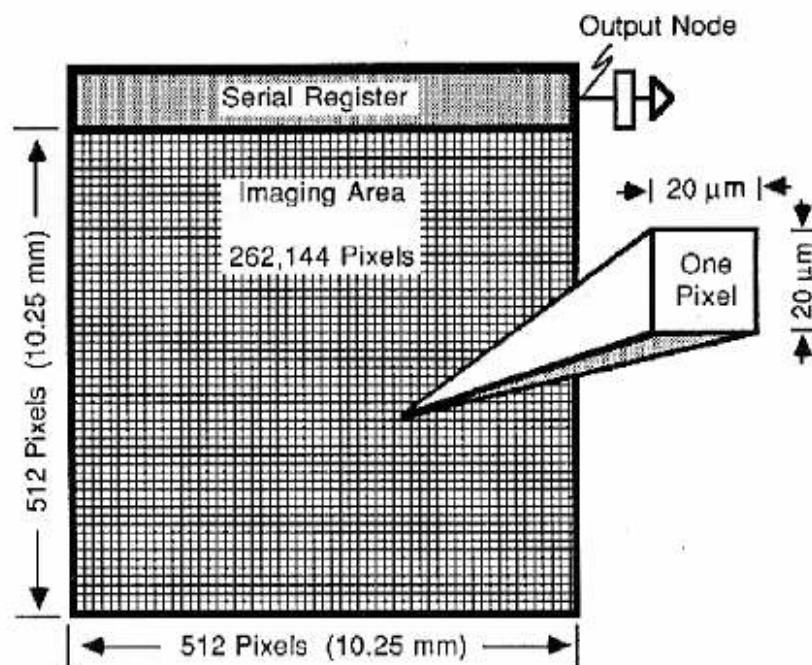


Figura 2.6 – Matriz CCD de 512 x 512 *pixels*.

Fonte: Howell, 2007.

## 2.4 PASSOS FUNDAMENTAIS EM PROCESSAMENTO DE IMAGENS

O processamento de imagens digitais abrange uma ampla escala de *hardware* e fundamentos teóricos (Gonzalez e Woods, 2000). Os passos fundamentais necessários para executar a tarefa são descritos como aquisição da imagem, pré-processamento, segmentação, representação e descrição, reconhecimento e interpretação; todos interligados a uma base de conhecimento.

Para a etapa de aquisição, faz-se necessário um sensor para registrar a imagem inteira do domínio do problema e, posteriormente, ser efetuada a digitalização do sinal produzido. Se a saída do dispositivo de aquisição não se encontra na forma digital, um conversor analógico-digital deve ser implementado para fazer a digitalização (Ribeiro, 2002).

Depois de adquirir a imagem digital, a próxima etapa é a de pré-processamento. O objetivo chave desse passo é melhorar a imagem para que fique mais apropriada para uma aplicação específica do que a imagem original, diminuindo as chances de erro. O pré-processamento compreende uma grande variedade de técnicas, como o realce de contraste, remoção de ruídos e isolamento de regiões cuja textura indique a probabilidade de obter dados não necessários ao processo.

A segmentação subdivide uma imagem em suas partes ou objetos constituintes. O nível até o qual essa subdivisão deve se realizada, assim como a técnica utilizada, depende do problema que está sendo resolvido (Gonzalez e Woods, 2000). Algoritmos de segmentação permitem achar diferenças entre dois ou mais objetos, e distinguir as partículas umas das outras e do fundo da imagem. Esta distinção permitirá ao programa interpretar *pixels* contíguos e agrupá-los em regiões.

Os algoritmos de segmentação para imagens monocromáticas são geralmente baseados em uma das seguintes propriedades básicas de valores de níveis de cinza: descontinuidade e similaridade. Na descontinuidade a abordagem é particionar a imagem baseado em mudanças

bruscas nos níveis de cinza. As principais áreas de interesse são as detecções de pontos isolados, detecção de linhas e bordas na imagem. Na similaridade, as principais abordagens baseiam-se em limiarização e crescimento de regiões (Gonzalez e Woods, 2000).

As rotinas de segmentação costumam ser as mais difíceis no processamento de imagens digitais, mas se bem desenvolvidas favorecem substancialmente a solução do problema. Por outro lado, se forem mal formuladas quase sempre asseveram falhas no processamento.

O próximo estágio do processamento diz respeito à representação e descrição da imagem. A saída da etapa de segmentação é tipicamente formada por dados na forma de *pixels*, que correspondem à fronteira de uma região, bem como todos os pontos em seu interior. Assim, a representação é a parte responsável pela tomada de decisão entre a exibição por fronteira, onde o foco são as características da forma externa como concavidade, pontos de inflexão ou cantos, ou por região, onde o interesse se concentra em propriedades internas como, por exemplo, a textura da imagem (Gonzalez e Woods, 2000).

Ligado à representação, a descrição corresponde ao método a ser utilizado para descrever os dados, de modo que as características de interesse sejam enfatizadas. A descrição procura encontrar características particulares, podendo ser usadas para a discriminação entre as classes de objetos.

A etapa final, conhecida como reconhecimento e interpretação, é a responsável por rotular o objeto discriminado no passo anterior e fazer a sua associação, com o uso dos descritores. Em um processo para a identificação de caracteres, o reconhecimento se destina a fazer uso das informações passadas pelo bloco anterior e atribuir o rótulo, como o dígito (9). Já a interpretação analisa dados como posicionamento dentro do quadro em foco ou o número da sequência de caracteres para interpretar, por exemplo, se trata de um código postal.

Por fim, todos os passos que constituem o processamento de imagem podem estar interligados a uma base de conhecimento, conforme pode ser visto na Figura 2.7. A base de conhecimento se destina a encurtar o processo através de troca de informações entre os blocos. Isso é realizado quando se tem um conhecimento prévio de objeto em estudo, passando informações sobre setores prioritários para análise, ou mesmo cancelando etapas de processos futuros que considerar desnecessário.

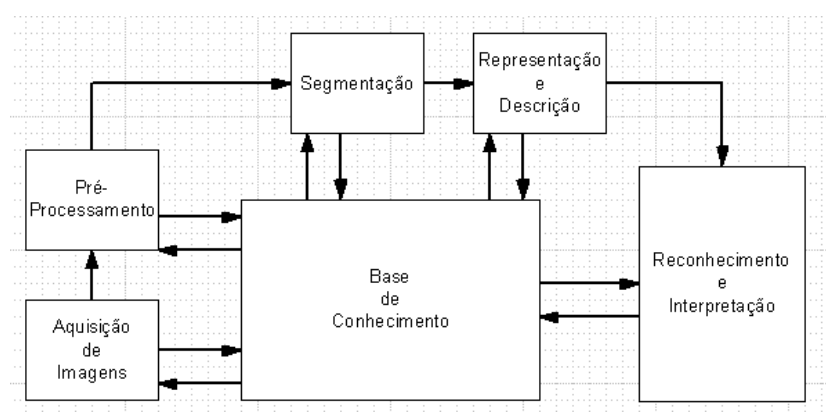


Figura 2.7 – Passos fundamentais em processamento de imagens.

## 2.5 TECNOLOGIA ASSISTIVA - TA

Tecnologia assistiva (TA) é um termo ainda novo, utilizado para identificar todo o arsenal de recursos e serviços que contribuem para proporcionar ou ampliar habilidades funcionais de pessoas com deficiência e consequentemente promover vida independente e inclusão. (Rita Bersch, 2005).

Em um sentido amplo, a evolução tecnológica caminha na direção de tornar a vida mais fácil. Talheres, canetas, computadores, controle remoto, celulares são uma amostra da interminável lista de recursos já assimilados pela sociedade. A TA é definida segundo conceito da *American With Disabilities Act* (ADA) como uma ampla gama de equipamentos, serviços, estratégias e práticas concebidas e aplicadas para minimizar os problemas funcionais encontrados pelos indivíduos com deficiências. (Cook e Hussey, 1995).

A termo *Assistive Technology* foi criado em 1988 nos EUA com um conjunto de leis que regula os direitos dos cidadãos americanos e tem por objetivo proporcionar aos portadores de deficiência maior independência, qualidade de vida e inclusão social, através da ampliação de sua comunicação, mobilidade, controle de seu ambiente e a possibilidade de exercer atividades e aprendizado com maior facilidade.

A tecnologia assistiva é dividida em diversas categorias, sendo uma em especial citada neste capítulo pela co-relação com o tema deste trabalho; trata-se do auxílio para cegos ou

peessoas com visão sub-normal. Nesta categoria entra os equipamentos desenvolvidos visando à independência completa ou parcial das pessoas com deficiência visual na realização de tarefas como: consultar o relógio, digitar em teclados, identificação de cores nas peças de vestuário e ter mobilidade em termos gerais.



### 3 APARATO EXPERIMENTAL

Para facilitar a compreensão do sistema implementado, a Figura 3.1 apresenta um diagrama de blocos resumido contendo os quatro blocos gerais do sistema desenvolvido.

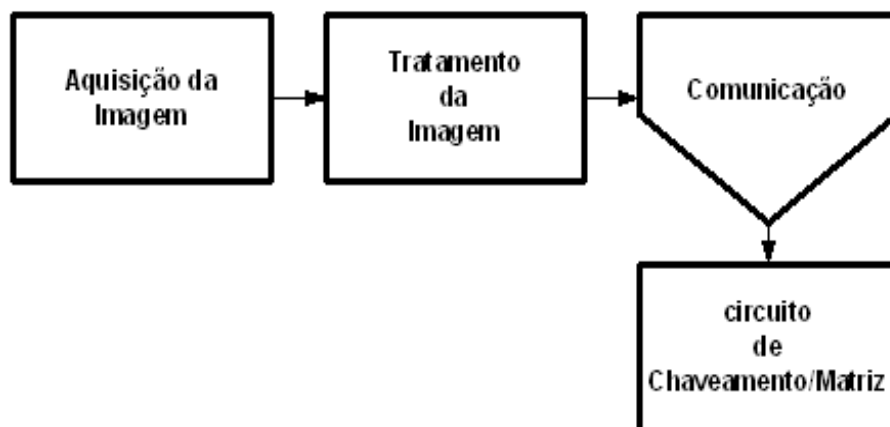


Figura 3.1 – Diagrama de blocos do sistema desenvolvido.

Esse diagrama foi composto por se tratar de técnicas distintas de atuação dentro do estudo proposto, sendo trabalhadas inclusive de forma individual, eventualmente, ao longo do período de desenvolvimento, e posteriormente interligado e efetuado os devidos ajustes.

### **3.1 AQUISIÇÃO DA IMAGEM DIGITAL**

A primeira etapa do projeto consiste em adquirir uma imagem (nesse trabalho foram utilizadas imagens controladas) que se encontra a frente de uma pessoa com deficiência visual, fazendo-se uso de um câmera *webcam*. A escolha da *webcam* surgiu devido a sua portabilidade e de não haver a necessidade de uma resolução mais requintada da imagem (não é o objetivo deste trabalho trabalhar com imagens detalhadas).

Foi utilizado o modelo 3810 da Leadership GoTec para o experimento; sua resolução é de 0,3 Megapixel e o tamanho da imagem de 352 x 288, com conexão USB. A Figura 3.2 ilustra a câmera escolhida. A câmera foi conectada a um PC que possui instalado o *software* Matlab® (*Matrix Laboratory*) versão 7.2.0 que foi utilizado como plataforma para a etapa de aquisição da imagem e também no processamento e na comunicação com o circuito externo.



Figura 3.2 – Câmera *webcam* GoTec 3810.

### 3.1.1 Reconhecimento da *Webcam*

No Matlab® existe uma família de pacotes de aplicativos destinados a resolver problemas específicos nas mais diversas áreas de atuação e são chamados de *toolboxes* onde são encontradas as funções correspondentes.

Apesar de a câmera *webcam* vir acompanhada de seu *software* específico de captura, optou-se de não usá-lo, e sim fazer a aquisição da imagem pelo próprio Matlab®, que oferece esse suporte através do pacote *Image Acquisition Toolbox*. Essa escolha se deu pelo fato de o posterior tratamento da imagem também ser realizado pelo mesmo *software*.

O *Image Acquisition Toolbox* é uma coleção de funções que estendem a capacidade do Matlab® para apoiar uma gama extensiva de operações de aquisição de imagem. Incluem a identificação de periféricos de aquisição baseados na tecnologia *Universal Serial Bus* (USB),

a capacidade de visualizar um *preview* do vídeo em tempo real, configuração de funções de chamada (*callback*) que ocorrem quando certos eventos acontecem, além de permitir que os dados ou imagens sejam apresentados no *workspace*, entre outras diversas funções.

O primeiro comando a ser digitado é a função ***imaqhwinf***, utilizada para avaliar se existe *hardware* de aquisição de imagem disponível. Ao digitar essa sintaxe o Matlab® retorna algumas informações gerais como o nome do *toolbox* em que a função está inserida, versão do *toolbox* e versão do Matlab®.

Mas a informação de interesse é o nome do adaptador. O adaptador é um *software* que o *toolbox* usa para se comunicar com o *driver* do dispositivo de aquisição de imagem. Existe uma lista grande de adaptadores registrados, muitos específicos para determinados fabricantes. A *webcam* em particular conectada ao PC gerou a seguinte resposta:

***InstalledAdaptors: {'coreco' 'winvideo'}***

Nesse caso, foram identificados dois adaptadores passíveis de uso para esse dispositivo: coreco e winvideo. O adaptador ‘winvideo’ por conter o padrão de vídeo para o Windows e possuir o padrão amplamente utilizado para câmeras IEEE 1394 (FireWire) foi o selecionado. Próximo comando é novamente a função ***imaqhwinfo*** seguido entre parênteses do nome do adaptador:

***>> imaqhwinfo('winvideo')***

Novamente é retornada uma série de informações, mas a de interesse é o número do equipamento que corresponde a ao código que o adaptador atribui para identificar o dispositivo e permitir a comunicação. A resposta obtida foi:

***DeviceIDs: {[1]}***

Agora de posse dessas duas informações, é possível obter informações do Matlab® acerca da câmera em particular, através do comando:

***>> imaqhwinfo('winvideo',1)***

Com a resposta obtida após entrar com esse comando, tem-se a certeza do pleno reconhecimento do dispositivo pelo *toolbox*, pois o *driver* repassa para o Matlab® o nome da *webcam* ('*SoC PC-Camera*') e seu formato padrão ('*RGB24\_352x288*').

### **3.1.2 Criando Objeto de Entrada de Vídeo**

Um objeto de entrada de vídeo é necessário ser criado, pois é através dele que o *toolbox* representa a conexão entre o Matlab® e o dispositivo de aquisição. Para sua criação faz-se uso dos dados recebidos anteriormente do comando ***imaqhwinfo***, na seção sobre o reconhecimento da *webcam*.

A função pronta que cria esse objeto é a ***videoinput***, que assim como a anterior, também pertence à *Image Acquisition Toolbox*. O comando a ser digitado é:

```
>>vid = videoinput ('winvideo', 1, 'RGB24_352x288');
```

A sintaxe define que após a função ***videoinput*** ser digitada, coloca-se entre parênteses, na ordem: o nome do adaptador, número do dispositivo e formato padrão. Deve-se ter o cuidado de colocar o nome do adaptador e o formato padrão entre apóstrofes, por se tratar de dados na forma de caracteres.

O objeto de entrada de vídeo deve ser armazenado a uma variável, por isso foi atribuído a *vid* (nome adotado para o objeto de vídeo nesse estudo). Caso o programador não atribua o objeto a uma variável o Matlab®, faz isso automaticamente, o que não é recomendado, pois ele usa sempre o mesmo nome para esses casos (ans); da próxima vez que uma atribuição automática ocorrer será gravado na mesma variável e o objeto será perdido.

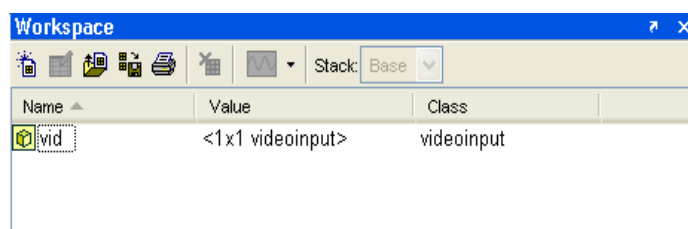


Figura 3.3 – Espaço de trabalho do Matlab®.

Na Figura 3.3 se visualiza a janela *workspace* do Matlab®, onde é possível acompanhar as variáveis criadas ao longo do programa, bem como suas características de classe e valor. A descrição sucinta do objeto de entrada de vídeo pode ser acessada através da função *get*. Essa função é muito usada dentro do *software* e não é exclusiva do *toolbox* de aquisição de imagem. Ela repassa todas as características de um objeto criado com seus valores *default*, valores que podem ser alterados, se necessário, com a função *set*.

```
>> get(vid)
```

A função *preview* digitada com o nome do objeto de vídeo criado entre parênteses abre uma janela de tamanho reduzido no ambiente Matlab®, passando a mostrar na tela a imagem em tempo real adquirida pela *webcam*. Com a função *preview*, o vídeo não está sendo gravado em nenhuma locação de memória do PC, que é o objetivo, pois a informação a ser tratada para a continuação do processo é apenas a foto que dela será retirada, conforme será mostrado. A Figura 3.4 mostra a janela do *preview* após a chamada do comando.

```
>> preview(vid)
```

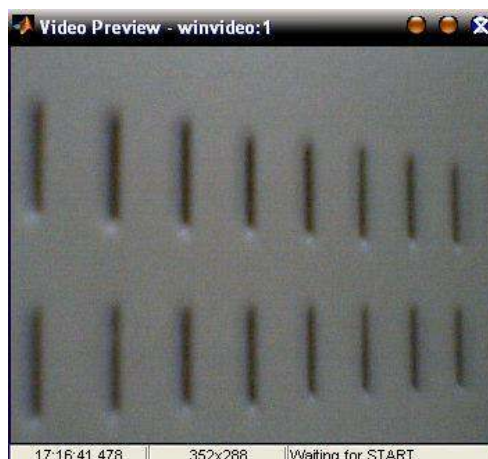


Figura 3.4 – *Preview* do Matlab®.

### 3.1.3 Script M-File

A maneira mais simples de se fazer um programa em Matlab® é criar um arquivo texto com a lista de comandos desejados. Os comandos são os mesmos que usados no *Command Window* e tem a mesma sintaxe. Um programa escrito assim é chamado *script* e toda vez que for chamado efetua a lista dos comandos como se eles fossem entrados sequencialmente via teclado.

Esse arquivo pode ser criado com qualquer editor de texto e deve ter a extensão *.m*, para chamá-lo basta entrar com o nome do arquivo no *Command Window* do Matlab®. O arquivo *script* deve estar no diretório corrente ou no *path* do Matlab®, que é a lista dos diretórios onde os arquivos são procurados caso não o ache no diretório corrente.



Os arquivos *script* são úteis quando se deseja efetuar uma sequência de comandos com muita frequência, como é o caso do objetivo deste trabalho, em que é almejado efetuar todo o processo de aquisição e tratamento da imagem para posterior envio dos dados à matriz de eletrodos, efetuando a eletroestimulação. Esse processo precisa ser rápido e sem a necessidade da digitação dos comandos um a um.

Uma das ferramentas disponíveis no Matlab® é um editor de textos específico para programas .m, o M-File Editor. Embora seja possível editar um arquivo .m em qualquer editor de textos, o editor do Matlab® possui características que facilitam muito o trabalho do programador, como o fato de enfatizar as palavras chaves com cores diferentes, verificar balanço de parênteses em expressões e prover uma interface amigável para o uso do *debug*.

### 3.2 PROCESSAMENTO DA IMAGEM DIGITAL

Depois da imagem já adquirida pelo Matlab®, começa a etapa de processamento da imagem digital. Antes do comando de captura da foto para o processamento, foi atribuído um novo valor para a especificação ***TriggerFrameDelay***, especificação essa pertencente ao objeto de entrada de vídeo. O seu valor *default* é zero, o que significa que o primeiro *frame* que ocorrer depois do gatilho será o capturado.

Entende-se por “gatilho” o comando recebido pelo Matlab® para a captura da imagem. Como nesse trabalho todo o *software* desenvolvido na plataforma Matlab® é chamado através de um script, como já exposto, o tempo entre as funções *preview* e *getsnapshot* (essa última responsável pelo gatilho) será muito curto e a foto adquirida ficaria totalmente escurecida; isso ocorre porque as câmeras levam alguns instantes de tempo para ajustar o foco após entrarem em funcionamento. Então à especificação *TriggerFrameDelay* foi atribuído o valor sessenta (60). Esse valor gera um *delay* de 2 segundos após o comando para registrar a foto.

```
>> set(vid, 'TriggerFrameDelay', 60);
```

Na sequência, o próximo comando efetua o registro da imagem, e grava na variável *fcam*:

```
>> fcam = getsnapshot(vid);
```

### 3.2.1 Conversão entre tipos de imagem

Assim como na etapa de aquisição, no processamento da imagem digital também foi utilizado um *toolbox* específico, chamado de *Image Processing Toolbox*, que possui uma coleção extensa de operações. Como algumas funções do Matlab® trabalham especificamente com algum tipo de imagem, torna-se necessário fazer conversões entre tipos para poder utilizar de seus recursos. Nesse trabalho é necessário fazer um reconhecimento das bordas de uma imagem de resolução 8 x 8, para assim poder gerar sua representação análoga na matriz

de eletrodos. (Essa resolução é relativamente baixa, mas foi à escolha em função da extensão do *hardware* para estimulação).

### 3.2.1.1 *RGB para Grayscale*

A primeira função de conversão utilizada é a ***rgb2gray*** que faz a conversão direta da imagem RGB para tons de cinza. Dentro do algoritmo dessa função, como descrito no *Image Processing Toolbox*, há a conversão dos valores RGB para coordenadas *National Television Systems Committee* (NTSC), que força os valores do matiz e saturação para zero e converte novamente para RGB.

Para mostrar os passos de processamento será usada uma imagem como modelo, chamada ‘ananas.jpg’.

```
>>ananas = imread('ananas.jpg');
```

Com esse comando o Matlab® lê a imagem com o nome especificado e salva na variável que foi atribuída. O resultado do próximo comando abre uma janela com a ilustração da imagem.

```
>>figure: imshow(ananas);
```



Figura 3.5 – Imagem RGB.

Fonte: <http://www.seeyhoo.com>.

Após o comando de conversão ***rgb2gray***, pode-se verificar o resultado do processo.

```
>>ananas_cinza = rgb2gray(ananas)
```



Figura 3.6 – Imagem Convertida para *Grayscale*.

### 3.2.1.2 *Grayscale para imagem Binária*

Para se chegar a uma imagem binária, uma das técnicas mais utilizadas é a de detecção de limiares, frequentemente aplicada nas rotinas de processamento de imagem devido a sua eficiência e praticidade. Conhecida como limiarização ou *thresholding*, o método agrupa diferentes objetos e regiões da imagem conforme os níveis de intensidade luminosa entre eles, através de um histograma. A técnica é analisar a similaridade dos níveis de cinza da imagem extraindo os objetos de interesse através da seleção de um limiar  $T$  que separa os agrupamentos de níveis de cinza.

A operação de teste que envolve a função  $T$  é apresentada como:

$$T = T[x, y, p(x, y), f(x, y)]$$

Onde  $f(x, y)$  representa a intensidade luminosa de um ponto e  $p(x, y)$  representa uma propriedade local deste ponto na imagem. Já uma imagem limiarizada  $g(x, y)$  é definida como:

$$g(x, y) = \begin{cases} 1 & \text{se } f(x, y) > T \\ 0 & \text{se } f(x, y) \leq T \end{cases}$$

Onde  $f(x, y)$  corresponde ao nível de cinza do ponto, os *pixels* rotulados com 1 correspondem aos objetos e os *pixels* rotulados com 0 correspondem ao fundo, e  $T$  é um valor

de tom de cinza pré-definido denominado limiar. Em uma imagem com 256 tons de cinza, se o valor do limiar  $T$  foi definido em 100, as tonalidades entre zero e esse valor será considerado o fundo da imagem (será atribuído o valor binário 0 – cor preta); os que forem acima de 100 serão tratados como objetos da imagem (será atribuído o valor binário 1 – cor branca). A Figura 3.7 representa um histograma, onde o limiar  $T$  foi atribuído a um valor entre duas áreas de maior precipitação de tons de cinza.

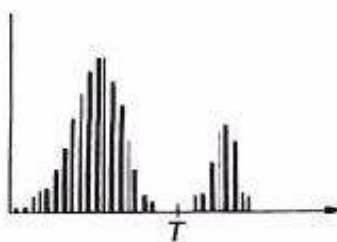


Figura 3.7 – Ponto de limiarização de uma imagem grayscale.

Fonte: Gonzalez e Woods, 2000.

O método de limiarização foi o método escolhido para transformar a imagem para binário. O Matlab® possui dentro do *Image Processing Toolbox* uma função para se encontrar o fator limiar  $T$ .

```
>>T = graythresh(ananas_cinza);
```

A função *graythresh* determina o melhor limiar para a imagem em questão e atribui à variável T. Tendo o limiar em mãos, extraído da imagem *grayscale*, pode-se converter a imagem RGB direto para a imagem binária através do comando *im2bw*.

```
>>ananas_binaria = im2bw(ananas, T);
```

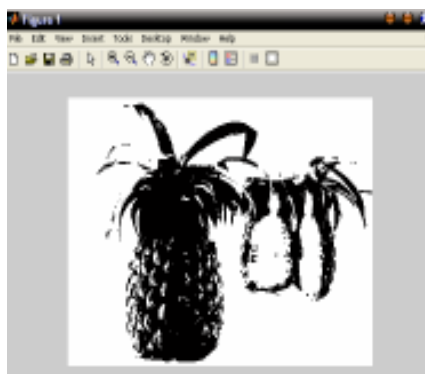


Figura 3.8 – Imagem convertida para binário.

### 3.2.2 Detecção das Bordas

Dentre as técnicas de segmentação de imagens como reconhecimento de linhas e reconhecimento de pontos, a detecção de bordas é de longe a mais eficiente para detecção de descontinuidade em tons de cinza. (Gonzalez e Woods, 2000).

Para esse estudo, antes de realizar a detecção das bordas foi necessário diminuir a resolução da imagem, visto a resolução original adquirida pela *webcam* ser de 352 x 288 e o protótipo desenvolvido para a eletroestimulação utilizar uma resolução de 8 x 8 em função do custo de sua implementação. Para chegar a essa resolução foi utilizado um comando do Matlab® chamado *imresize*. Basta digitar essa função e posteriormente, entre parênteses, digitar o nome da variável que contem a imagem a ter sua resolução reduzida, juntamente com a dimensão almejada. Deve-se atribuir logicamente esse comando a uma nova variável, como segue a sintaxe a seguir.

```
>> f8x8 = imresize(foto, [8 8]);
```

Após a adaptação da imagem à resolução 8 x 8 desejada, começa a etapa de detecção de borda. O Matlab® oferece a função *edge* para a detecção de bordas, e com uma sintaxe parecida é possível se utilizar de seis métodos distintos: *Sobel*, *Prewitt*, *Canny*, *Zerocross*, *Log* e *Roberts*.

Todos os métodos disponíveis pelo Matlab® foram testados ao longo dos diversos dias de ensaios, mas devido à baixa resolução da imagem, o único que apresentou uma representação aceitável foi o método de Canny. O processo de detecção de bordas de Canny baseia-se em dois critérios básicos de desempenho, os critérios de detecção e localização. Estes critérios estão sujeitos ainda a um terceiro, conhecido como injunção de resposta múltipla, que força o processo a detectar uma única borda onde existe somente uma borda. (Do Vale, 2002).



```
>>fborda = edge(foto, 'canny');
```

Este comando efetua a detecção da borda pelo método de Canny.

### 3.3 COMUNICAÇÃO DO PC COM O MICROCONTROLADOR

Como resultado da etapa de processamento da imagem digital é obtido uma matriz binária de resolução 8 x 8 que representa analogamente a imagem original adquirida pela *webcam*. Essa matriz binária gerada pelo Matlab® é a informação que deve ser enviada para um *hardware* externo contendo um microcontrolador PIC18F4620 da Microship, montado em uma placa padrão.

O microcontrolador, de posse dessa informação, executará o *software* gravado em sua memória e fará a partição da matriz, de modo a enviar um vetor de 8 bits para cada uma das oito placas de chaveamento. Esse processo de chaveamento será explicado posteriormente.

Fez-se necessário criar uma porta serial “virtual” para a comunicação do Matlab® com o *hardware* externo, pois o PC usado para a pesquisa é um *notebook* Acer modelo 5040 que possui apenas portas USB como interface para periféricos. Foi pesquisada a alternativa de fazer a comunicação direta pela interface USB, mas as funções encontradas no Matlab® para

esse tipo de comunicação, como a função *visa*, se restringem a periféricos comerciais que já possuem um protocolo de comunicação reconhecida por ele.

Para criar essa serial “virtual” foi utilizado o CI FT232R da FTDI Chip que gera uma interface de USB para UART. Esse dispositivo vem incorporado de melhorias em relação a sua versão anterior, tais como: um oscilador interno calibrado, E2PROM interna, filtro na alimentação e detector de elevação de tensão de alimentação com gerador de *reset*.

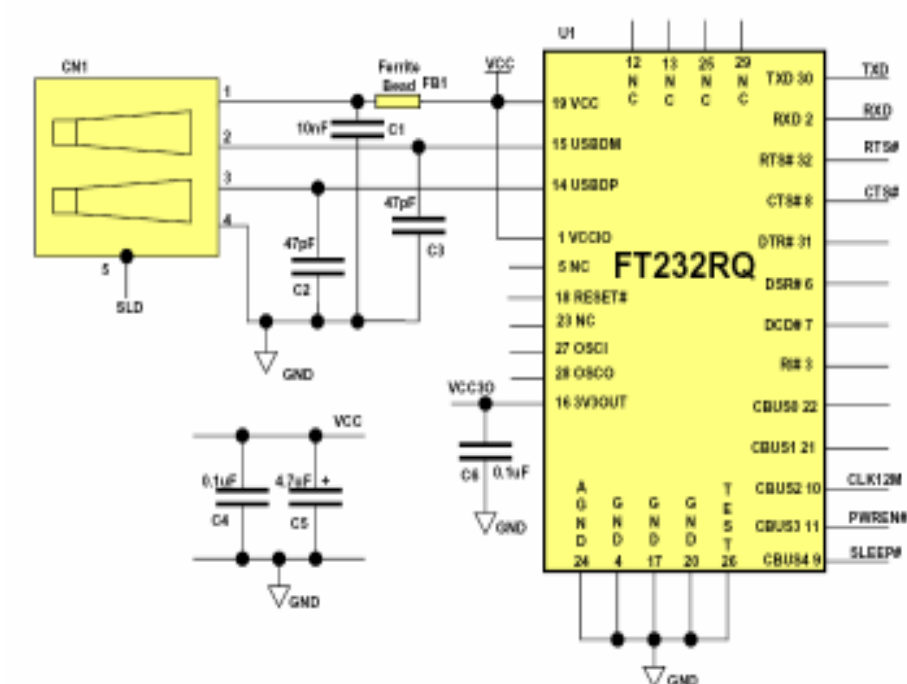


Figura 3.9 – Circuito de conversão USB para RS-232.

Fonte: *Datasheet* do FT232R

O circuito recomendado pelo fabricante, conforme ilustra a figura 3.9, foi montado na mesma placa onde se encontra o PIC. Também foi adaptada uma porta USB2 fêmea na placa e interligado com o FT232R através dos pinos USBDP e USBDM. Pinos 15 e 16 respectivamente. A Figura 3.10 mostra em detalhe as ligações do *chip* FT232R, juntamente com a porta USB2.

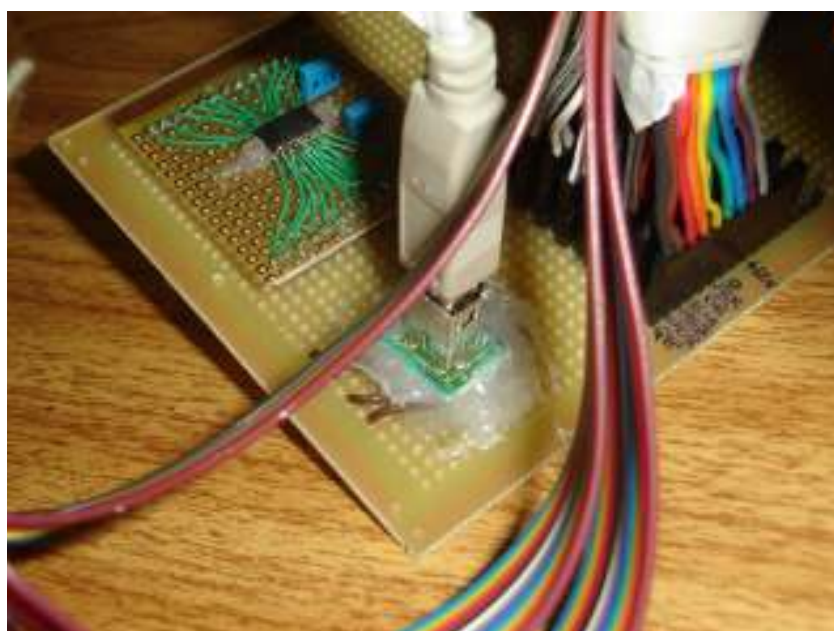


Figura 3.10 – Detalhe do *chip* FT232R e da entrada USB.

O FT232R possui dois pinos denominados de CBUS1 e CBUS0 (pinos 22 e 23) que existem para se fazer o acompanhamento visual dos estados de TXD e RXD. Coloca-se entre cada um deles e a fonte de alimentação um led acompanhado de seu resistor limitador de

corrente. Quando a comunicação é aberta e informações são enviadas do PC para o seu *buffer*, acende-se o led, mostrando que a comunicação serial está ocorrendo sem problemas.

Após o término da montagem do circuito FTDI, para constatar se o mesmo foi reconhecido pelo sistema operacional, faz-se a conexão entre ele e o PC e abre-se o “Gerenciadores de dispositivos”, localizado no painel de controle do Windows. A interface funcionou como esperado e o sistema operacional identificou a nova serial como ‘COM4’.

### 3.3.1 Reconhecimento da Serial pelo Matlab®

Tendo o conhecimento do nome atribuído pelo sistema operacional para a porta serial “virtual” (no caso específico desse trabalho foi a COM4) o Matlab® pode acessar essa porta para enviar ou receber dados por ela.

```
>>s4 = serial('COM4');
```

Com esse comando o Matlab® cria um objeto de comunicação serial associado ao canal físico ‘COM4’ e atribui esse objeto a uma variável, escolhida como sendo “s4”. No próximo comando são desabilitados *DataTerminalReady* e *RequestToSend*; isso porque o *software* foi implementado para ter comunicação serial assíncrona, sem controle de fluxo. Por se tratar de um protótipo não era o foco no momento.

```
>>set(s4, 'DataTerminalReady', 'off', 'RequestToSend', 'off');
```

O objeto de vídeo foi criado e configurado. Para se abrir a comunicação com o objeto de vídeo deve-se utilizar a função *fopen*. Pode-se verificar quando a comunicação foi aberta porque os led's associados ao chip FT232R nos pinos UBUS1 e UBUS0 acendem quando isso ocorre. Depois de abrir a comunicação, o FT232R está pronto para receber a informação do Matlab® e armazenar em seu *buffer*. A sintaxe utilizada nesse estudo para abrir a comunicação serial no Matlab® foi:

```
>>fopen(s4);
```

### 3.3.2 Convertendo a Matriz binária para Caractere

Depois de a imagem passar pelo processo de conversão de formato, diminuição da resolução e detecção de borda, uma matriz binária de 64 elementos é gerada e armazenada em uma variável. Essa matriz representa o contorno do objeto captado pela *webcam*, onde os *pixels* que representam a borda têm o valor 1 e tonalidade branca, enquanto o fundo tem valor zero e tonalidade preta.

Essa variável precisa ser enviada para o microcontrolador externo, pois ele é o responsável por controlar o *hardware* que irá efetuar a eletroestimulação tátil. Para fazer a

transmissão do dado de forma confiável para o microcontrolador, optou-se em transformá-lo previamente em um vetor de caracteres; a informação dessa forma continua a ser formada por valores de '0' e '1', porém no padrão ASCII. Os comandos mostrados na sequência primeiro fazem à conversão da matriz binária em *double* e depois o resultado é transformado em um vetor de 64 caracteres em ASCII.

```
>>M = double(matriz_binária);
```

```
>>m = char(reshape(M+48, 64, 1)');
```

A função ***reshape*** redimensiona a matriz 8 x 8 *double* em um vetor de 64 x 1, e a função ***char*** converte seus elementos para caracteres. Assim a informação está pronta para ser enviada. O comando ***fprintf*** é usado para enviar o dado ***m*** do processo pela porta ***s4*** também especificada entre parênteses.

```
>>fprintf(s4, m);
```

No anexo B está adicionado todo o script das rotinas acessadas pelo Matlab®, incluindo as de aquisição, processamento e de comunicação.

### **3.4 CONFECÇÃO DO *HARDWARE* PARA ELETROESTIMULAÇÃO**

A confecção do *hardware* de chaveamento foi à etapa de maior dedicação e esforço ao longo do período de implementação deste trabalho. Uma série de aparatos tiveram de ser projetados e confeccionados para poder dar corpo ao projeto e posterior comprovação de sua viabilidade. Será detalhada cada segmentação do *hardware* no desenvolver dessa monografia, mas para uma visão geral pode ser listada como: Placa UCP, placas de chaveamento, matriz de monitoramento e matriz de eletrodos.

#### **3.4.1 Placa UCP**

Essa é a placa mãe do projeto. Nela fica situado o microcontrolador PIC18F4620 da Microchip, utilizado para armazenar e particionar o vetor de caracteres enviado pelo Matlab®, de forma a atuar sobre as oito placas de chaveamento. Nela também fica localizada a porta USB por onde entram os dados vindos do PC e todo o circuito de conversão para RS-232.

A entrada de força de todo o sistema também fica alocada na placa UCP. Terminais soldados à placa recebem alimentação externa oriunda de uma bateria Chumbo-Ácida de 12 V e 7 Ah. A tensão da bateria alimenta o circuito regulador de tensão de 5 V formado por um LM7805, responsável por alimentar o microcontrolador e o chip FT232R.

Essa placa possui oito fileiras de pinos, ordenados para a conexão dos *flats cables*, utilizados para fazer a comunicação com as placas de chaveamento. De cada um desses conectores saem nove fios, na sequência: QH-, SRCLR, SRCLK, RCLK, OE, SER, GND, reserva, +12 V. Na Figura 3.11 é apresentada a placa UCP: na parte superior aparece os terminais de alimentação e o circuito regulador de tensão, no centro o PIC e as fileiras de conectores para a comunicação com as placas de chaveamento, e na parte inferior o circuito FTDI e a conexão USB2.

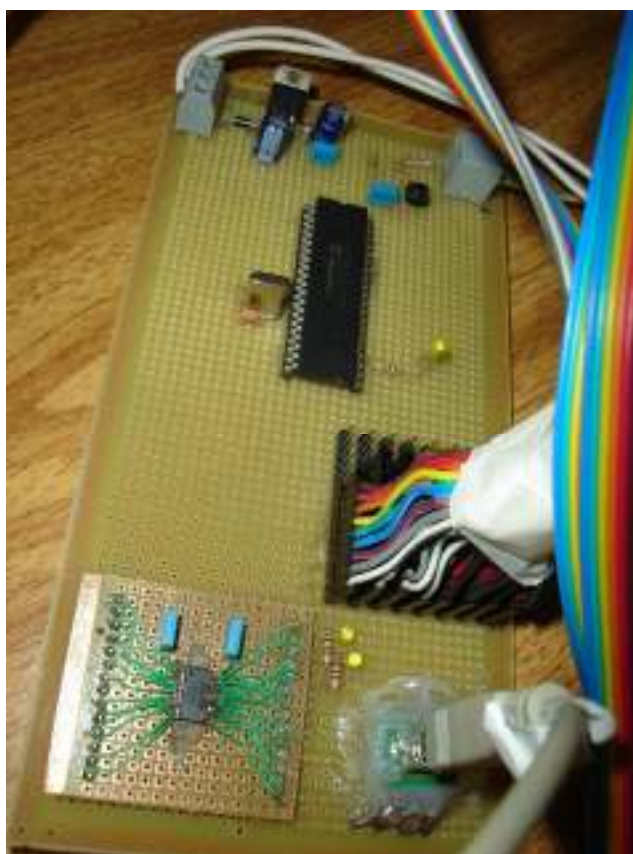


Figura 3.11 – Placa USB em detalhe.



Os seis primeiros correspondem a pinos do *Shift Register*, que são comandados pelo microcontrolador. Tem-se, ainda, o GND e o + 12 V que são as mesmas linhas vindas da bateria e vão alimentar também todo o circuito das placas de chaveamento. Ainda foi levada uma linha reserva para cada placa para o caso de uma eventual necessidade de alteração do projeto. A Figura 3.12 apresenta o esquemático do PIC18F4620.

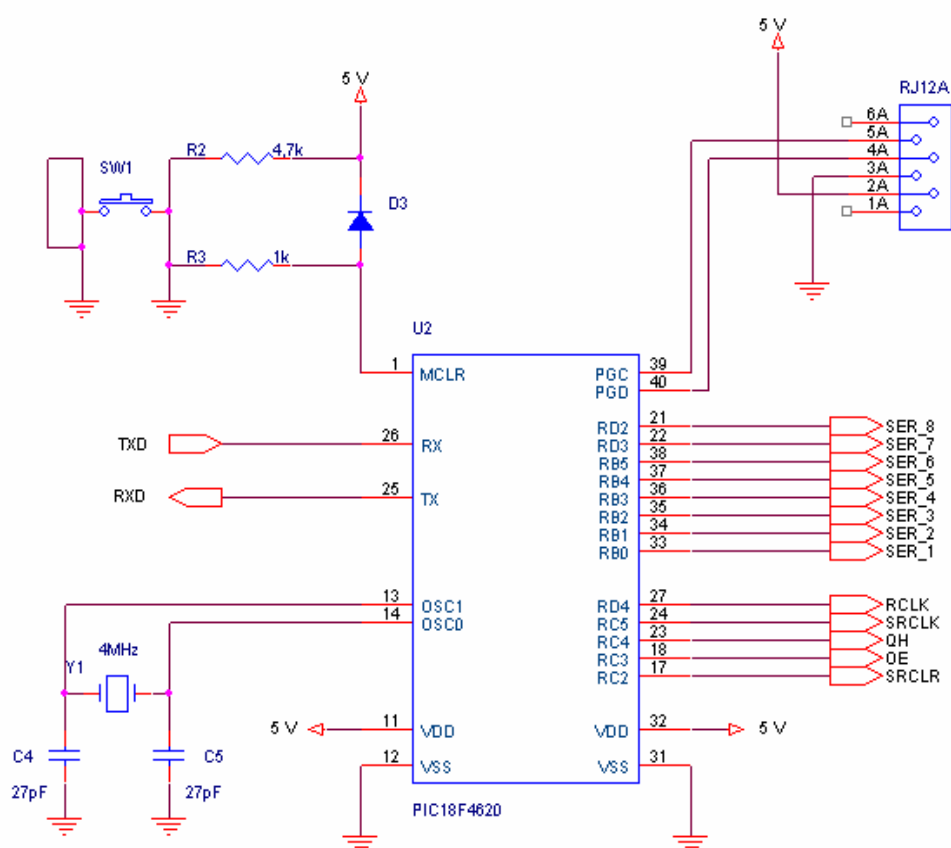


Figura 3.12 – Circuito do PIC18F4620

Pode-se ver na Figura 3.12 que foi instalado na placa um conector RJ12. Através dele é gravado o *software* do microcontrolador de maneira *on board*. Para isso, adquiriu-se um gravador intitulado ICD2, licenciado pela Microchip. A Figura 3.13 mostra o desenho dos pinos de saída da placa.

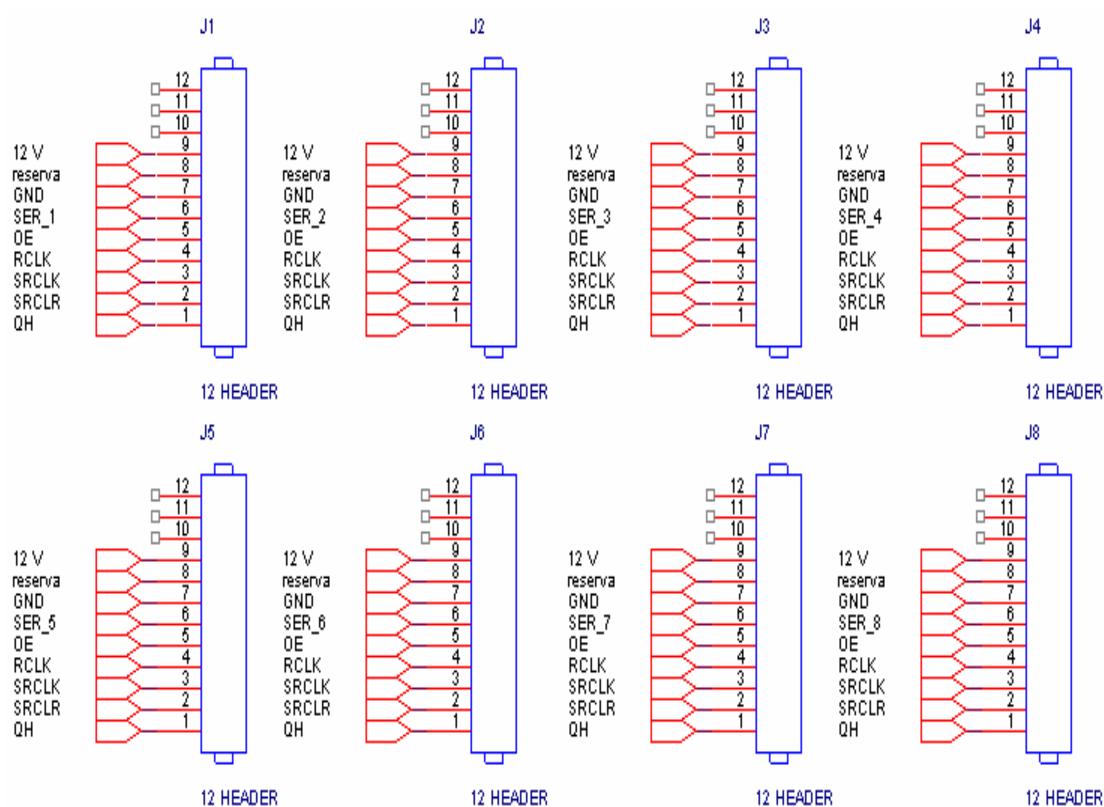


Figura 3.13 – Representação dos pinos de saída da placa UCP.

Analisando o esquemático dos conectores, percebe-se que somente os pinos SER\_X variam entre as fileiras, os demais são ligados em comum a todos. O motivo será detalhado na seção 3.4.2 que fala das placas de chaveamento. O esquemático é apresentado novamente nos anexos.

#### **3.4.1.1 Rotinas do Microcontrolador**

Nessa seção é explicado com detalhes a lógica do programa escrito na linguagem C para ser utilizado no microcontrolador PIC18F4620. O editor usado para escrever o programa foi o MPLAB IDE versão 7.50, sendo ele um ambiente integrado para estudo e desenvolvimento com a família PIC de microcontroladores. Sua principal característica é a total integração de seus módulos com o ambiente windows.

A primeira coisa a se fazer ao programar um PIC é acertar os bits de configuração (*configuration bits*). A sintaxe utilizada pelo compilador C 18 para essa função é:

***#pragma config 'registrador' = 'valor'***

Nessas configurações é possível ajustar opções como o tipo de clock a ser utilizado, a existência ou não de *watchdog*, a habilitação ou não de pinos de A/D, etc. Depois do *configuration bits*, outra obrigatoriedade de se trabalhar com o PIC é configurar o sentido das

portas de I/O, ou seja, se serão utilizadas como entrada ou saída. Isso é feito através dos registradores TRISx. No caso desta aplicação apenas o TRISC foi configurado para possuir um pino de entrada, sendo esse pino a entrada RX do EUSART. Nos demais só foram usados como *output*. Segue como ficou a sintaxe dos registradores TRISx.

```
TRISA = 0b00000000;  
TRISB = 0b00000000;  
TRISC = 0b10000000;  
TRISD = 0b00000000;
```

Como visto, ao atribuir o valor zero para um bit desses registradores está se dizendo que ele é uma porta de saída. O número binário 0b10000000 atribuído ao TRISC diz que o bit 7 desse registrador (RC7) é de entrada e foi atribuído com o valor 1.

Como o PIC deve efetuar uma comunicação serial com o PC para receber o dado tratado do Matlab®, foi necessário configurá-lo para isso. O 18F4620 possui um modulo EUSART (*Enhanced Universal Synchronous Asynchronous*). A sintaxe de configuração da EUSART que foi utilizada é mostrada a seguir juntamente com o comentário de seu significado.

```
/* TXSTA: TRANSMIT STATUS AND CONTROL REGISTER */  
  
TXSTAbits.CSRC = 0; //Tanto faz (Don't care) para o modo "Asynchronous"  
  
TXSTAbits.TX9 = 0; //Habilita 8-bits de transmissão  
  
TXSTAbits.TXEN = 1; //Habilita Transmissão Serial  
  
TXSTAbits.SYNC = 0; //Habilita o modo "Asynchronous"
```

```

TXSTAbits.SENDB = 0; // "Sync Break transmission completed"

TXSTAbits.BRGH = 0; //Habilita Low Speed mode para o Baud Rate Selected Bit

/* RCSTA: RECEIVE STATUS AND CONTROL REGISTER */

RCSTAbits.SPEN = 1; //Habilita Porta Serial

RCSTAbits.RX9 = 0; //Habilita 8-bits de transmissão

RCSTAbits.CREN = 1; //Habilita o para receber continuamente

/* BAUDCON: BAUD RATE CONTROL REGISTER */

BAUDCONbits.BRG16 = 0;

BAUDCONbits.ABDEN = 0;

SPBRG = 24;

```

Com isso termina a etapa de configuração e começa o algoritmo aplicado ao projeto propriamente dito. A primeira rotina executada pelo *software* é responsável por receber o vetor de caracteres do Matlab®, via serial, e gravar em uma variável. Essa variável precisa ser do tipo matriz de caracteres e com locação para 64 *bytes*.

```

Int indice = 0;
Char DATA;
char eletrodo [64];

for (indice=0; indice<=63; indice++)
{
    while (!PIR1bits.RCIF);
    DATA = RCREG;
    eletrodo[indice] = DATA;
    PIR1bits.RCIF = 0;
}

```

A rotina executa o bloco de comando em um laço *for* por 64 vezes; para cada interação, ela espera o registrador ***PIR1bits.RCIF*** ficar em nível alto para sair do comando *while*. Esse registrador é um *flag* que indica que a *buffer* da RX recebeu um *byte*. Depois disso se lê o dado da *buffer* e repassa seu valor para a variável DATA do tipo *char*. Depois atribui o valor da variável DATA para o vetor *eletrodo[índice]*.

No primeiro laço DATA será guardado em *eletrodo[0]*, e assim sucessivamente até a todos os elementos do *eletrodo* serem atribuídos com os valores do vetor enviado pelo Matlab®. No término dessa operação a “imagem” adquirida pela *webcam* já está armazenada no microcontrolador e pronta para ser redistribuída pelo PIC para os circuitos de chaveamento.

De posse da matriz, foi utilizado novamente um comando *for* com um laço de 64 vezes. Dentro de seu bloco existe um comando *switch* que faz a comparação do índice do laço *for* com diversos ***case`s*** em cascata, sendo esses intercalados de oito em oito.

Essa rotina, deve-se ao fato de que o Matlab®, ao utilizar a função ***reshape*** para redimensionar a matriz *double* 8 x 8 para *char* 64 x 1 durante a etapa de processamento de imagem, ordenar os caracteres por colunas; ou seja, após o último elemento da primeira coluna ser ordenado vem o primeiro elemento da segunda coluna, e assim por diante.

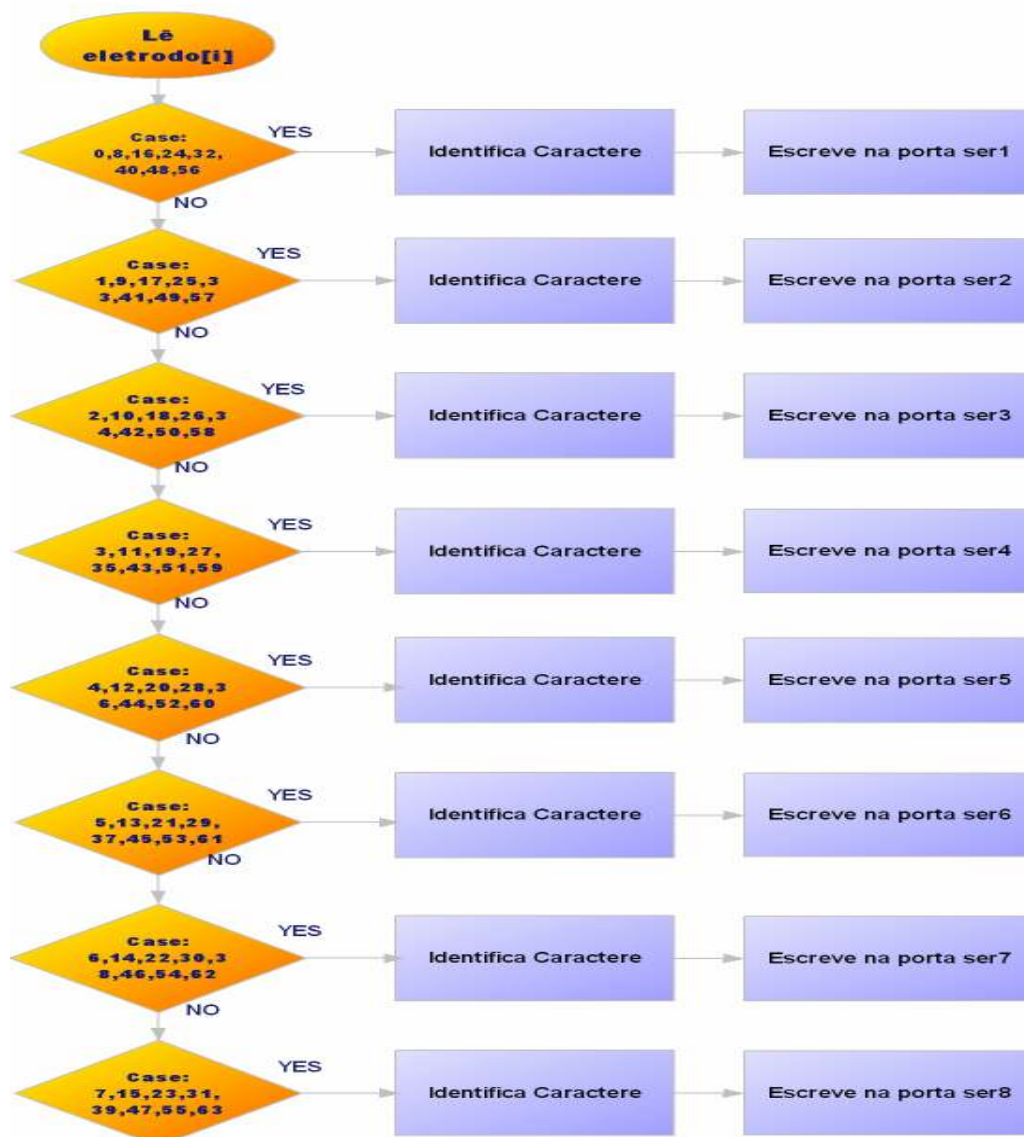


Figura 3.14 – Fluxograma do *software* do PIC para a identificação dos dados.

A Figura 3.14 é um fluxograma resumido da rotina de identificação dos dados da variável *matriz* e controle da informação a ser enviada a entrada serial dos SIPOs. Se o índice do laço *for* for igual a 0, 8, 16, 24, 32, 40, 48 ou 56 (condições dos *case`s* em cascata), significa que pertence a primeira linha da matriz, então o valor do eletrodo(i) deve ser jogado em **PORTBbits.RC0**, por ele estar ligado fisicamente a entrada serial (SER) do registrador de deslocamento da primeira placa de chaveamento, e responsável por configurar a primeira linha da matriz de eletrodos. Porém, o dado de eletrodo(i) não é jogado diretamente em **PORTBbits.RC0**; se essa condição acontecer, entra um comando *if* para reconhecer se é o caractere '0' ou '1' para então jogar o bit correto na porta.

Depois de sair do *break*, *i* é incrementado e o processo continua, até que a última linha tenha seu bit configurado pelo pino SER, então, dentro desse bloco de comando, um pulso de *clock* é dado em comum a todos os oito registradores de deslocamento, e o segundo bit pode começar a ser atribuído.

Esse algoritmo não será inserido no corpo dessa monografia por ser muito extenso e repetitivo, limitando-se apenas a explicar seu princípio de funcionamento. O *software* completo encontra-se no anexo B.

Depois que os 8 bits dos oito registradores de deslocamento forem configurados, o *software* sai desse comando *for* e entra em um comando *while*, gerando um *clock* indefinidamente na frequência de 400 Hz no pino **PORTCbits.RC3** que é interligado aos



pinos de *enable* dos registradores de deslocamento. Isso faz com que um chaveamento nessa frequência ocorra no primário do transformador, fazendo o transformador elevar o trem de pulsos.

Os secundários desses transformadores são conectados no par de eletrodos, que por sua vez serão os responsáveis pela eletroestimulação. O motivo da escolha dos transformadores para a estimulação consta na seção 3.4.4 (Matriz de eletrodos).

### 3.4.2 Placas de Chaveamento

Foram construídas oito placas de chaveamento idênticas, cada qual é responsável por receber as informações pertinentes a uma das linhas que compõem a matriz a ser representada na eletroestimulação. As placas foram nomeadas de linha\_1 até linha\_8.

Cada placa de chaveamento recebe da placa UCP um chicote contendo nove vias, conforme já mencionado. Dois desses cabos são a alimentação vinda da bateria e vão servir para alimentar as três fontes reguladas existentes; sendo dois circuitos reguladores de tensão de 5 v utilizando o LM7805 e um circuito regulador de tensão ajustável utilizando o LM317.

No projeto optou-se por adicionar um *shift Register* (registrador de deslocamento) de 8 bits do tipo SIPO (*Serial input – parallel Output*) em cada placa. Esse registrador tem o seu

pino de entrada serial (SER) conectado ao chicote. O *software* se responsabiliza por enviar a cada um dos registradores de deslocamento, através de pinos distintos do PIC, o dado de 8 bits a ser carregado em seus flip-flop`s internos. Esse *byte* deve conter a configuração análoga da linha a qual a placa representa.

Para se fazer o deslocamento dos bits no interior dos registradores é necessário inserir um *clock* no pino (RCLK); o algoritmo implementado possibilita utilizar um único pino do PIC para o *clock*, sendo ele comum a todos os registradores. Foi utilizada a porta ***PORTBbits.RC5*** do microcontrolador para essa função.

Conforme especificação do fabricante pode-se ligar em curto os dois pinos de *clock* do registrador (RCLK e SRCLK), desde que se efetue um *clock* adicional após o final de deslocamento. Essa medida foi adotada, visto não haver necessidade de um segundo pulso de *clock* para a aplicação desse projeto.

Como se trata de um SIPO, cada um dos 8 bits que entraram serialmente serão representados em 8 saídas distintas, de forma paralela. Assim como cada placa de chaveamento corresponde a uma linha da matriz, cada saída do SIPO será igual ao estado de um elemento dessa linha. Existem, em cada placa de chaveamento, oito circuitos iguais responsáveis por fazer a eletroestimulação, e são como mostrados na Figura 3.15.

Como já exposto, existe três fontes reguladoras de tensão por placa de chaveamento. O primeiro reguladores de tensão de 5 V alimenta o *shift register*, a entrada do foto-acoplador e uma das linhas da matriz de monitoramento; sendo o positivo dessa primeira fonte nomeado de +5 Vcc. A outra fonte de 5 V chamada de +5 Vdd é conectada ao coletor do foto-acoplador, e a saída do regulador ajustável vai ao dreno do mosfet.

O motivo de duas fontes de 5 V é separar a alimentação dos CTs da alimentação do chaveamento do mosfet, por medida de proteção aos componentes.

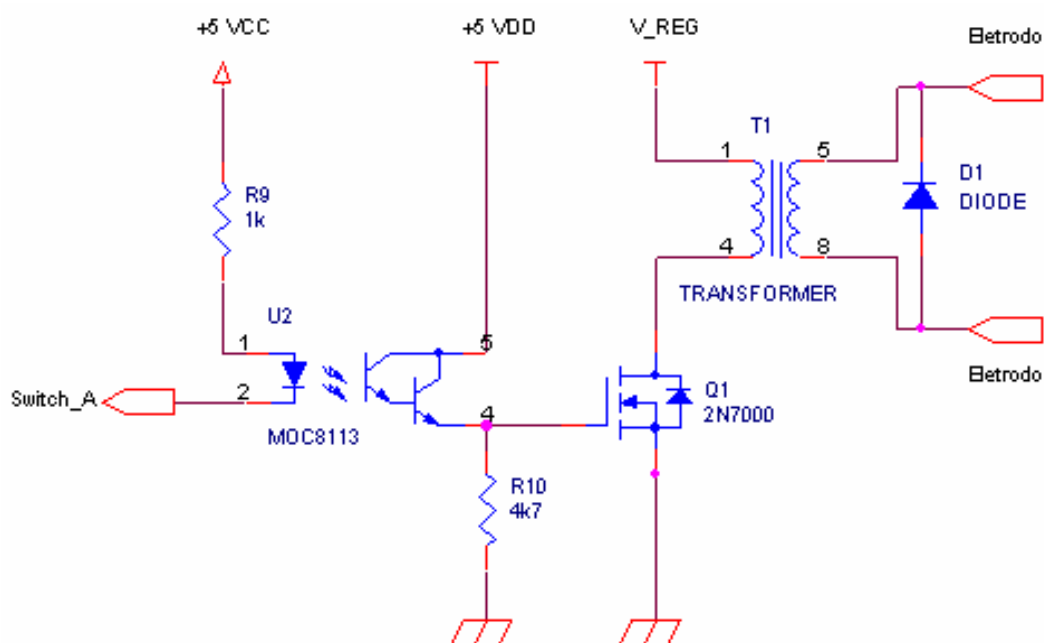


Figura 3.15 – Circuito para eletroestimulação.

Cada saída do *Shift Register* é ligada a um circuito de eletroestimulação através do catodo de um foto-acoplador, de modo que se a saída estiver em nível alto o MOC8113 (foto-acoplador utilizado) permanece desligado, porque seu anodo esta em +5 Vcc.

No caso da saída do *shift register* ser nível baixo, o diodo interno do MOC8113 liga e seu *darlington* conduz, aplicando uma queda de tensão de aproximadamente 3,3 v sobre o resistor de  $4k7\Omega$  ligado a seu emissor. No catálogo, o fabricante do mosfet 2N7000 garante que o mesmo conduz a partir de 2 V; com o chaveamento do mosfet fecha-se o circuito e a tensão da fonte ajustável (V\_REG) chega ao primário do transformador.

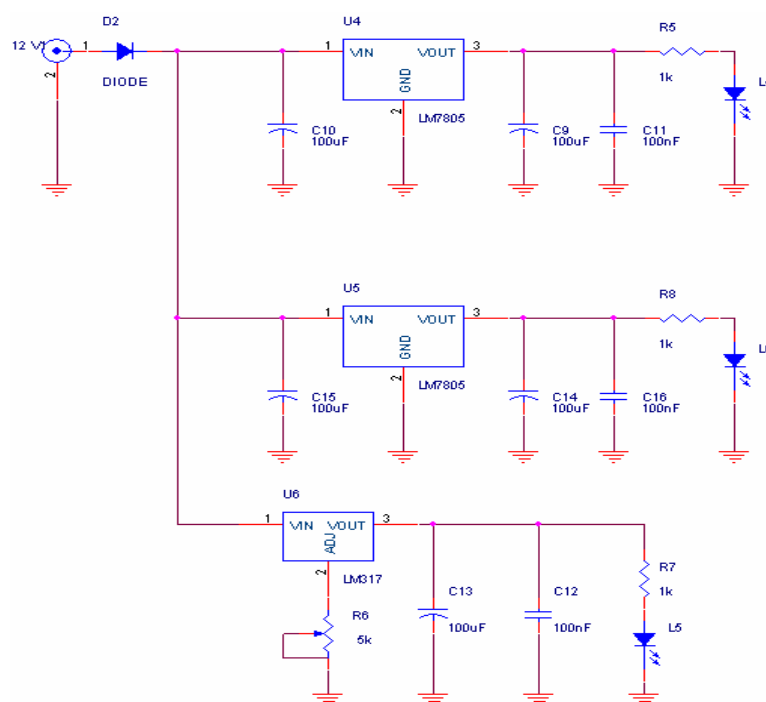


Figura 3.16 – Circuito das três fontes reguladas por placa de chaveamento.

A Figura 3.16 mostra o circuito das três fontes reguladas existentes por placa de chaveamento. Para gerar o chaveamento em 400 Hz do mosfet, o *software* gera um clock especial com essa frequência no pino **PORTCbits.RC5** do PIC. Esse clock é ligado em comum ao pino OE de todos os registradores de deslocamento, onde OE é o pino de *enable* dos registradores. Dessa forma, simultaneamente, todos os SIPO's habilitam e desabilitam seus pinos de saída nessa frequência.

Por fim, existe o pino SRCLR do registrador de deslocamento, utilizado quando se quer “limpar” todos os bits de saída. O *software* criado habilita o **PORTCbits.RC2** em nível alto logo no início do algoritmo e assim permanece inalterado. Um eventual descuido na sua utilização pode causar problemas na matriz. As Figuras 3.17 mostra o *layout* da placa.

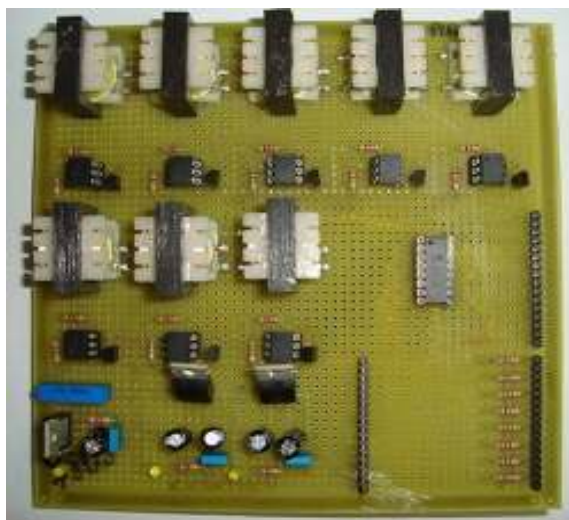


Figura 3.17 – Placa de Chaveamento.

### 3.4.3 Matriz de Monitoramento

Apesar do objetivo principal do projeto ser a representação da “imagem” na forma de eletroestimulação na matriz de eletrodos, foi montado uma matriz de led`s para o monitoramento visual do funcionamento do sistema. Ela foi confeccionada com *flat cable* e possui *plug`s* nas extremidades, de modo que pode ser removida quando necessária. A Figura 3.18 ilustra a matriz de led`s em detalhe.

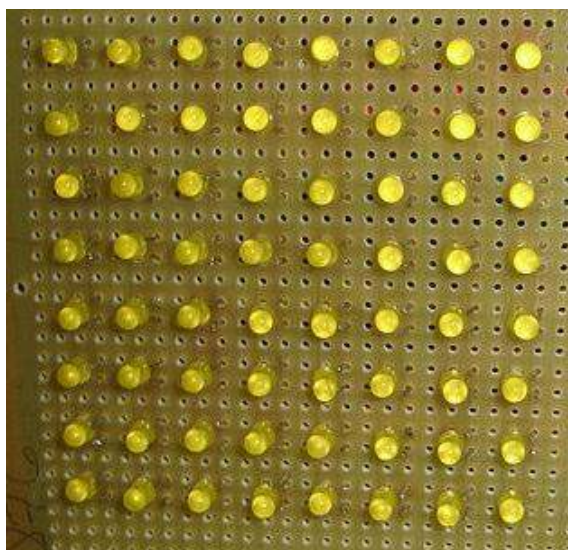


Figura 3.18 – Matriz de monitoramento.

O catodo dos led`s está conectada juntamente com o catodo do foto-acoplador, de modo que quando a saída do *shift register* for colocada em nível baixo, o led acenderá, acusando que

aquele circuito de chaveamento está aplicando tensão no primário do transformador. A Figura 3.19 mostra o esquemático da matriz de monitoramento.

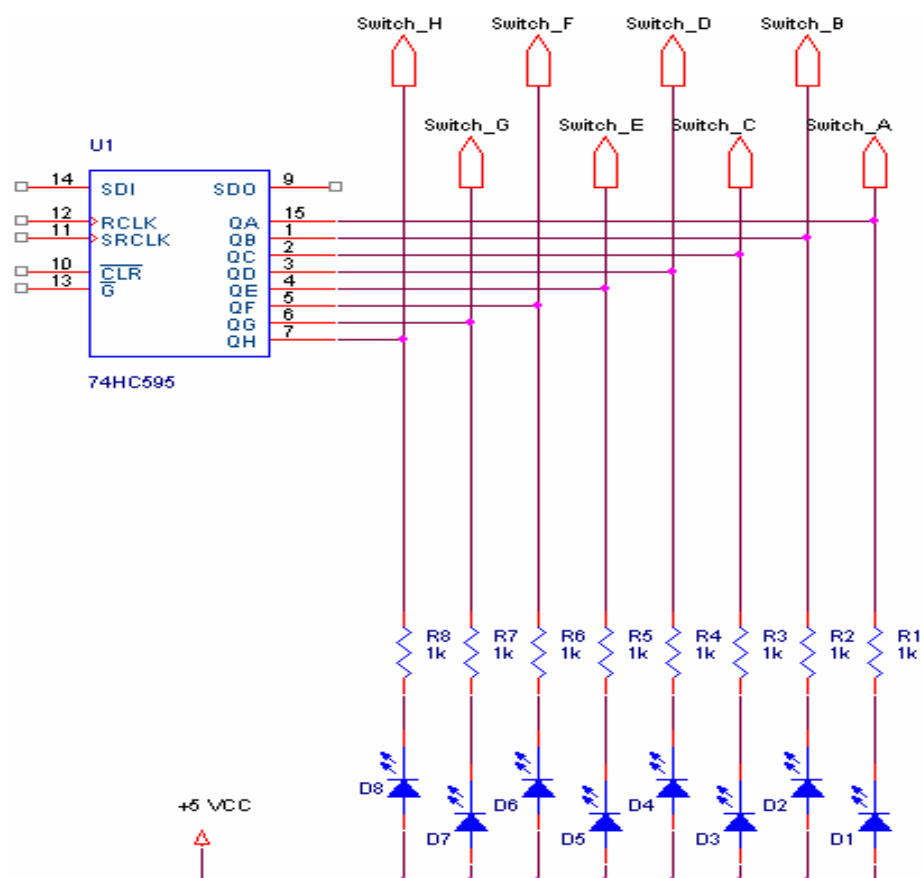


Figura 3.19 – Circuito da matriz de monitoramento.

#### 3.4.4 Matriz de Eletrodos

A matriz de eletrodos precisava ser implementada sob uma superfície flexível para poder aderir por inteiro à superfície do abdômen, devido às grandes proporções da cinta (300 x 200

mm). Por haver a necessidade de se utilizar gel condutor, também é importante o material ser impermeável.

Por esses motivos se optou por utilizar uma chapa de raio-x para o protótipo. Nela foram afixados 128 eletrodos de ECG (Eletrocardiograma), cuja área de contato é revestida de Ag/AgCl (Cloreto de Prata) e possuem impedância parecida com a da pele. Na Figura 3.20 aparece o eletrodo da Kobme utilizado na montagem.



Figura 3.20 – Eletrodo de ECG.

Fonte: <http://www.cirurgicapassos.com.br>

Um par de eletrodos se faz necessário para representar um *pixel* da imagem a ser representada, pois atuam no projeto com um dipolo. A corrente que entra em um eletrodo sai pelo seu par; e esse pequeno trecho da derme sofre a sensação de estímulo.



Segundo a lei dos nós de kirchoff, a soma algébrica das correntes que entram em um nó é nula. Então, para garantir que a corrente que entra por um eletrodo obrigatoriamente saia pelo seu par, utilizou-se os transformadores. O secundário de cada transformador foi conectado a um dipolo, e sua propriedade de isolamento galvânica não permite que haja fuga de correntes entre dipolos e acabe por “borrar” a estimulação e, conseqüentemente prejudicar na identificação das formas. A Figura 3.21 ilustra a cinta de eletrodos.



Figura 3.21 – Cinta com a matriz de eletrodos vista frontal.

A lamina da matriz de eletrodos foi costurada a uma cinta de material de boa elasticidade, podendo ser presa em volta do corpo, na região abdominal. Foram fixadas tiras de velcro na lamina e no tecido, na parte de trás da cinta, com o propósito de poder ser removida a matriz

sempre que necessário, como no caso de manutenção. Na Figura 3.22 tem-se a visão traseira dos eletrodos.



Figura 3.22 – Matriz de eletrodos vista de trás.

A conexão dos fios com os eletrodos teve de ser apenas por contato, na forma de ilhós, por eles não suportarem a temperatura elevada necessária para se efetuar a solda. A fixação foi dada sob pressão da extremidade do fio em torno dos eletrodos, apesar do método, o resultado foi satisfatório e houveram poucas falhas durante o processo, sendo resolvidos em uma segunda vistoria. Para cada linha da matriz foi usado um *flat cable* de 2 m de comprimento, para haver espaço de locomoção nos ensaios. Na outra extremidade, foram presos a

conectores fêmea, de modo a serem ligados as placas de chaveamento de maneira prática; mesmo método adotado na matriz de monitoramento e nos cabos de comunicação entre placa UCP e placas de chaveamento.

## 4 RESULTADOS E DISCUSÕES

Depois de concluída a etapa de projeto e montagem do *hardware* e programado os *softwares* do Matlab® e do microcontrolador, começaram os ensaios. O primeiro passo foi à verificação do funcionamento das placas. Devido o tamanho do *hardware* implementado e o fato dos circuitos terem sido montados na sua totalidade em placas padrão comerciais, era de esperar que alguns *bug`s* ocorressem. Foram corrigidos problemas como solda fria e pequenos curtos entre terminais dos componentes.

Após os reparos, foi testado individualmente cada circuito de chaveamento utilizando o gerador de funções, aplicando-se uma onda quadrada de zero a 5 V na entrada do fotoacoplador para verificar o chaveamento do mosfet. Depois de garantido o funcionamento dos mesmos, passou-se para a etapa de testes dos registradores de deslocamento.

Os registradores de deslocamento foram testados conectando-se todo o sistema e abrindo comunicação com o Matlab®. Foi enviado pela serial um vetor de caracteres de forma direta, ou seja, sem o uso da rotina completa de aquisição e tratamento de imagem. Esse vetor pré-determinado serviu para visualizar seu “reflexo” na matriz de monitoramento. A Figura 4.1 mostra todo o aparato conectado nessa fase de teste.



Figura 4.1 – Visão do sistema.

#### **4.1 ENSAIOS DE CAPTURA E CHAVEAMENTO**

Verificado que a rotina de configuração dos registradores de deslocamento funciona, foi possível começar os ensaios do sistema do modo que foi idealizado. Para isso foram

selecionadas imagens comportadas para serem adquiridas pela *webcam*. Foram feitos diversos ensaios do sistema, sendo alguns apresentados nessa documentação.

#### 4.1.1 Ensaio de figuras circulares

O primeiro ensaio apresentado será o da captura de uma circunferência. O modelo foi aproximado da câmera e o *script inicio* criado no Matlab® foi digitado em seu *prompt*. Como já descrito, todos os algoritmos são executados sequencialmente e após o término do tratamento da imagem, o dado é enviado ao PIC. A Figura 4.2 mostra a imagem RGB (original) adquirida, a Figura 4.3 é a convertida para *grayscale*, e a Figura 4.4 é sua conversão para binária.

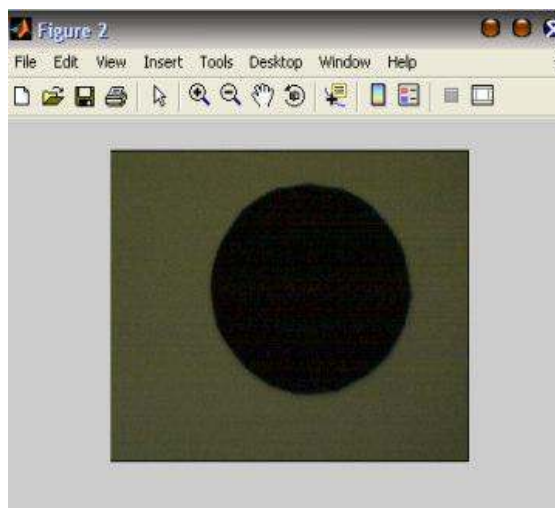


Figura 4.2 – Imagem do círculo adquirido (RGB).

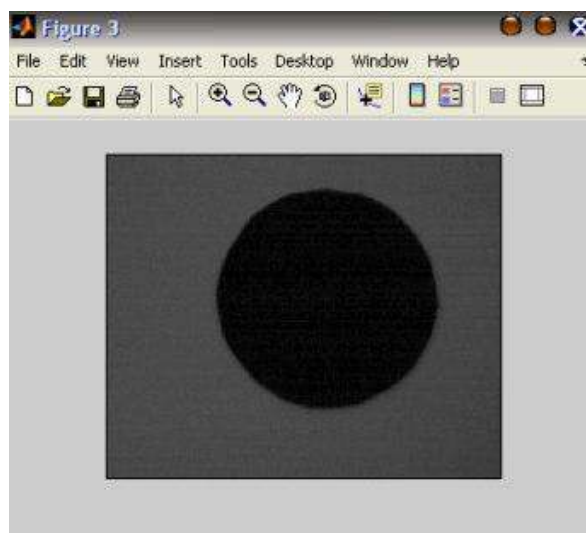


Figura 4.3 – Imagem do círculo convertido para *grayscale*.

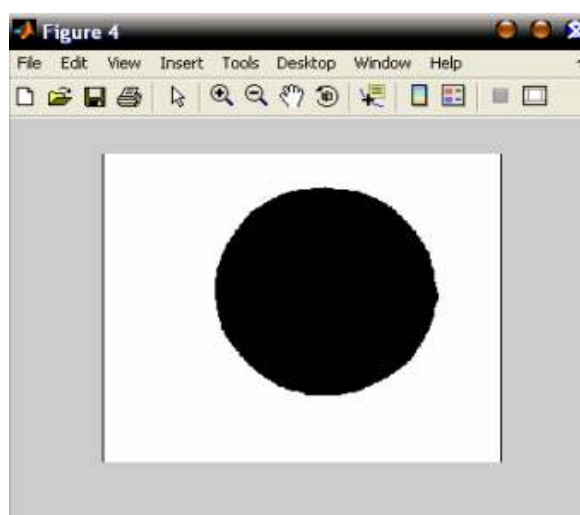


Figura 4.4 – Imagem do círculo convertido para binário.

A Figura 4.5 mostra em detalhes a matriz de monitoramento com o resultado do ensaio sendo representado e a Figura 4.6 ilustra a tela do PC juntamente com a matriz durante o processo experimental.



Figura 4.5 – Resultado do círculo em detalhe.

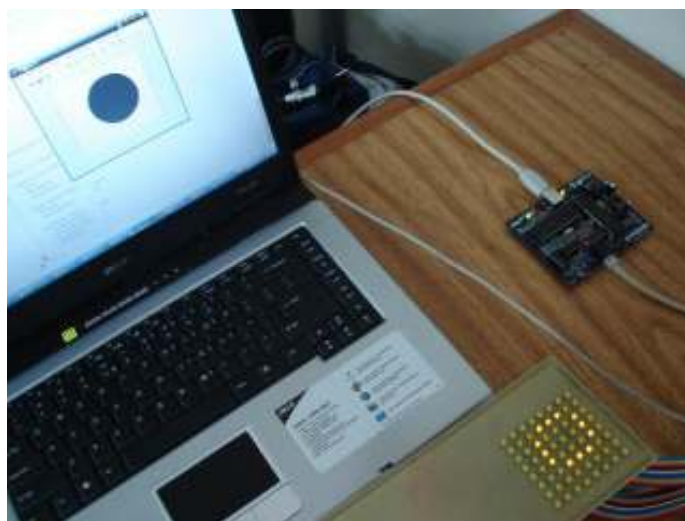


Figura 4.6 – Visão geral do experimento com o círculo.



O ensaio para detecção de bordas em figuras circulares mostrou-se muito eficiente nesse processo; quase a totalidade dos ensaios apresentou o resultado esperado, mesmo havendo deslocamento do círculo para os lados no enquadramento ou pequenas variações no seu raio, desde que respeitando a resolução do protótipo.

#### 4.1.2 Ensaio de colunas

Foi realizada uma bateria de ensaios de reconhecimento de colunas, com variações na direção. Colunas verticais, horizontais e inclinadas foram utilizadas e chegou-se a alguns resultados. Na Figura 4.7 aparece o processo de uma coluna vertical e na Figura 4.8 sua representação visual nos led's.



Figura 4.7 – Processamento de uma coluna vertical.

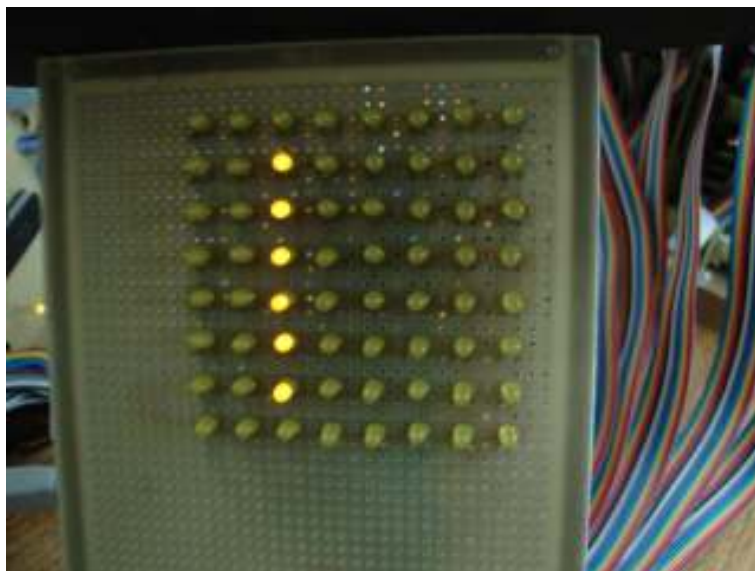


Figura 4.8 – Representação de uma reta vertical na matriz de monitoramento.

Esse ensaio mostrou que colunas também podem ser representadas de maneira satisfatória e com boa aproximação no método de *canny*. Os objetos representados podem ter sua orientação estimada, como visto na Figura 4.8, onde a coluna foi representada a esquerda da matriz, de modo análogo à coluna captada mostrada na Figura 4.7.

Mas algumas observações são válidas: mesmo a coluna atravessando a tela de captura de extremo a extremo, a primeira e última linha não é representada como visto na Figura 4.8. Isso ocorre porque a técnica visa reconhecer as bordas da imagem, mas quando ela ultrapassa os limites do *frame*, o seu “corte” não é considerado fronteira do objeto. Outro ponto a ser destacado no tratamento de colunas é a inconstância de sua representação, como será

exemplificado no próximo ensaio. A Figura 4.9 mostra uma coluna horizontal sendo tratada e a Figura 4.10 sua representação na matriz de monitoramento.



Figura 4.9 – Processamento de uma coluna horizontal.

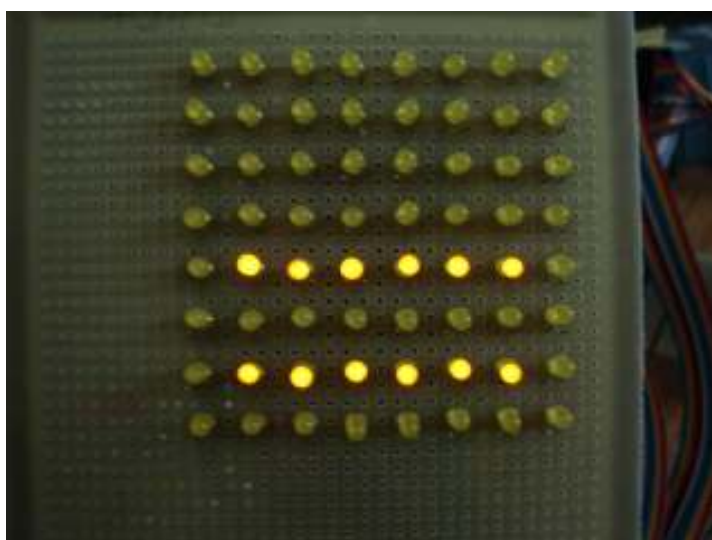


Figura 4.10 – Representação de uma reta horizontal na matriz de monitoramento.

Nesse ensaio se verifica o aparecimento de duas retas. Isso é compreensível e esperado, pois uma coluna possui espessura, e duas bordas são identificadas. O problema é não haver uma regularidade, pois, imagens de colunas idênticas (entende-se aqui para os padrões da visão humana) podem apresentar uma borda (como o ensaio anterior) ou duas, e se torna mais um problema a ser resolvido em futuros estudos voltados ao melhoramento desse trabalho, pois a PPDV precisa de um padrão definido para poder usufruir do método. Vale destacar que as duas observações acerca da identificação de colunas ocorre para qualquer direção.

Na Figura 4.10 pode-se ver que apesar de aparecerem duas bordas, nenhuma teve representação na primeira e última coluna da matriz. A Figura 4.11 mostra uma representação de coluna na vertical cujo término se deu no meio da tela.

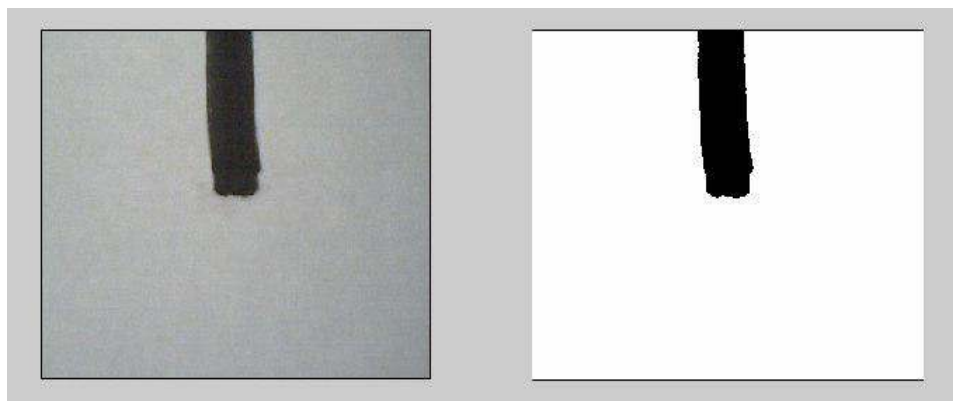


Figura 4.11 – Processamento de uma coluna interrompida na metade da tela.

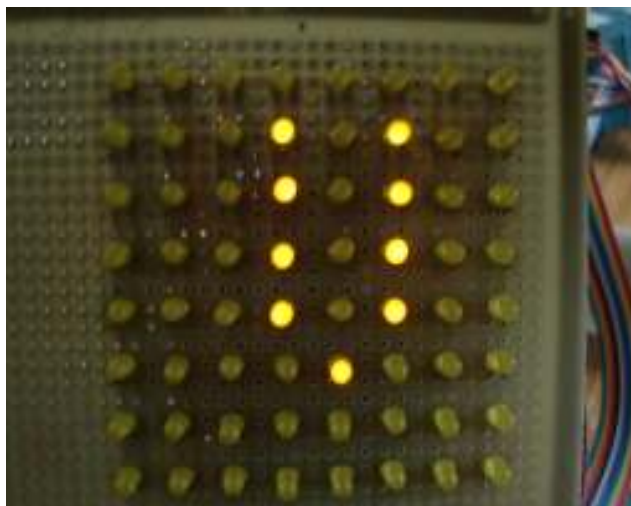


Figura 4.12 – Representação de uma coluna interrompida na matriz de led`s.

A Figura 4.12 mostra que quando a coluna não corta a área de captura da imagem por inteiro, é possível detectar uma terceira borda, que representa a parte onde a coluna é interrompida. Essa representação se deu pelo *bit* localizado entre as linha maiores. Com esse ensaio é possível perceber que o sistema segue uma lógica entre as representações

Essas imagens comportadas foram usadas para testar o funcionamento do protótipo e demonstrar que é viável de ser implementado. Ele funcionou de maneira satisfatória dentro de suas limitações de *hardware*, principalmente pela resolução de 8 x 8 *pixels* utilizada, devendo apenas ser investido em uma tecnologia mais sofisticada.

## 4.2 SINAL USADO PARA ELETROESTIMULAÇÃO

Seguindo sugestões de alguns pesquisadores (K.A.Kaczmarek, 1991) e (T.P.Way, K.E.Barner, 1997), a forma de onda escolhida para efetuar a estimulação foi um trem de pulsos. Para isso se optou por utilizar de uma fonte de tensão regulada, podendo excursionar a saída de 1,2 a 12 V com a regulação de resistores variáveis, de forma a poder ajustar o nível da estimulação.

Essa fonte é ligada ao circuito de chaveamento de forma a injetar os pulsos de tensão DC no primário de um transformador monofásico de relação 9/50V fabricado sob encomenda. O uso dos transformadores se deu para testar um sistema onde não houvesse fuga de corrente entre diferentes pares de eletrodos, buscando a criação de uma “imagem” formada por estimulações pontuais na derme. Com os transformadores usados dessa forma existe também uma isolamento galvânica de usuário com relação a fonte de alimentação, sendo uma forma mais segura para o usuário.

Foram montados 64 circuitos de chaveamento, um para cada elemento da matriz 8 x 8. Os pulsos são controlados pelo microcontrolador através de rotinas de *delay* e são facilmente alterados conforme a necessidade do ensaio. Os pulsos foram testados empiricamente sob diversos valores de frequência e *duty cycle*. Os ensaios desse método realizado com o uso da cinta de eletrodos constatou que na frequência de 400Hz é onde o usuário sente maior

estimulação com menos corrente elétrica. Na figura 4.13 aparece o uso da cinta durante os ensaios.



Figura 4.13 – Ensaio com a cinta de eletrodos.

Nessa frequência observou-se que a estimulação dá a sensação de uma pequena vibração na derme, mas o uso de gel condutor se faz necessário, pois sem ele existe uma sensação de “pontada” e acaba sendo desconfortável se for usado por um período maior de tempo. Chegou-se ao valor de tensão de 2,0 V no regulador de tensão utilizando pulsos de 400 Hz, e esse vem sendo o valor adotado até o momento para os ensaios.

A Figura 4.14 mostra o gráfico gerado pelo osciloscópio ligado ao secundário do transformador, representando assim a tensão aplicada nos eletrodos.

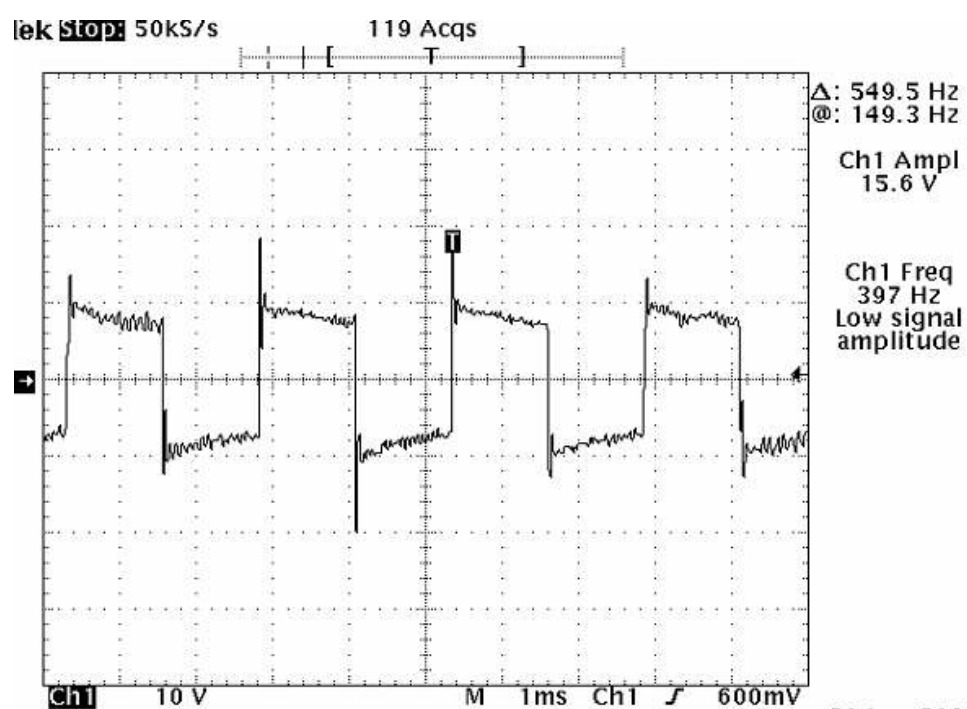


Figura 4.14 – Sinal da esimulação.



## 5 CONCLUSÕES

Neste trabalho foi feito um protótipo inspirado em técnicas de substituição sensorial (SS) e tecnologia assistiva, voltado para o auxílio a pessoas portadoras de deficiência visual. Foi proposta a substituição sensorial da visão pelo tato, fazendo com que imagens comportadas fossem captadas, tratadas digitalmente para detectar seus contornos e reduzir resolução, e posteriormente passadas ao usuário através de eletroestimulação tátil no abdômen, fazendo uso de uma cinta de eletrodos confeccionada para esse fim.

O sistema desenvolvido utiliza-se de uma câmera *webcam* para efetuar a aquisição da imagem diretamente ao *software* Matlab®. Depois de a imagem ser adquirida, ela passa por uma etapa de processamento, onde é convertida de formato e têm sua resolução reduzida para 64 *pixels*, para então se efetuar o reconhecimento das bordas utilizando-se o método de canny, já incluso no *Image Acquisition Toolbox* do Matlab®. Como resultado desse processo se obtém uma matriz que representa, de forma análoga, o contorno da imagem adquirida. Essa

matriz é enviada serialmente para um microcontrolador da família PIC, sendo esse responsável por redistribuir o dado entre as placas de chaveamento que iram estimular os eletrodos.

Devido o uso de transformadores monofásicos no circuito de chaveamento, sendo seus enrolamentos secundários conectados aos pares de eletrodos, foi possível uma estimulação pontual na derme, aumentando a eficiência na detecção das bordas, além de uma maior segurança do usuário por haver isolamento galvânica entre ele e a fonte de alimentação do sistema. Apesar do estímulo ser elétrico, obteve-se uma sensação de vibração quando são ajustadas adequadamente as formas de onda e a intensidade do sinal. Foi determinado de forma empírica que a melhor faixa de frequência do estímulo é próximo dos 400 Hz, mas não se chegou ainda a certeza das melhores características do sinal da eletroestimulação, devido o sistema não ter sido ensaiado o suficiente com diferentes usuários.

O aparato funcionou como imaginado, porém, não é possível acertar a figura estimulada sem antes passar por um período de treinamento. Ao contrário do que pensa o senso comum, a perda de um sentido não acarreta automaticamente a melhora dos demais. Essa técnica esta enquadrada como as demais técnicas de substituição sensorial, como a leitura em Braille ou a leitura labial, em que se usa um sentido para substituir outro e é necessário prática para ser bem utilizada.

## **6 SUGESTÕES E MELHORIAS**

O protótipo desenvolvido mostrou que com o método adotado, é possível efetuar a eletroestimulação tátil como substituição sensorial com segurança. Mas para uma maior eficiência do sistema na continuidade dos estudos, podem ser listadas algumas sugestões futuras:

- a) Efetuar testes com grupos de deficientes visuais para validar o sistema;
- b) Minimizar o sistema empregando melhores tecnologias, como o uso de componentes SMD;
- c) Aumentar a resolução da matriz, aumentando a eficácia na detecção das bordas e podendo partir para imagens reais de ambientes;

- d) Diminuir o diâmetro dos eletrodos para suportar o aumento da resolução. Montar a matriz de eletrodos em *flexcircuit*;
- e) Após definir as intensidades da estimulação, implementar circuitos para limitar a corrente de excitação, como fontes de corrente, para uma segurança ainda maior do usuário.

## 7 REFERÊNCIAS BIBLIOGRÁFICAS

E.R.KANDEL, J.H.SCHWARTZ, T.M.JESSEL. Principles of neural science. McGrawHill, 4<sup>th</sup> ed. New York: 2002.

GONZALEZ, RAFAEL C.; WOODS, RICHARD E. Processamento de imagens digitais. São Paulo: Edgard Blücher, 2000.

AZEVEDO, E.; CONCI A. Computação Gráfica: Teoria e prática. Rio de Janeiro: Elsevier, 2003.

BEÊ, R.T, TRASEL, R. Células fotocondutivas. Curitiba: 1999.

FELBER, P. A literature study as a project for ECE 545. Illinois Institute of Technology. 2002.

ANTONINO. Estudo preliminar para o desenvolvimento de visão através da sensação tátil, utilizando estimulação eletrocutânea. Tese de Mestrado, UNICAMP, 1997.

DU VALE, GIOVAVI MAIA. Processo de Detecção de Bordas de Canny. Mestrado do Programa de Pós-Graduação em Ciências, 2002

T.P.WAY, K.E.BARNER. Automatic visual to tactile transtation – Part I: Human factors, access methods, and image manipulation. IEEE Transaction on Rehhabilitation bEngineering. V.5, No 1, 1997.

K.A.KACZMAREK Electrotactile and vibrotactile display for sensory substitution systems neutophysiological Basic of a tactile vision-substitution system. IEEE Transactin on Biomedical Engineering, BME – 38, 1991.

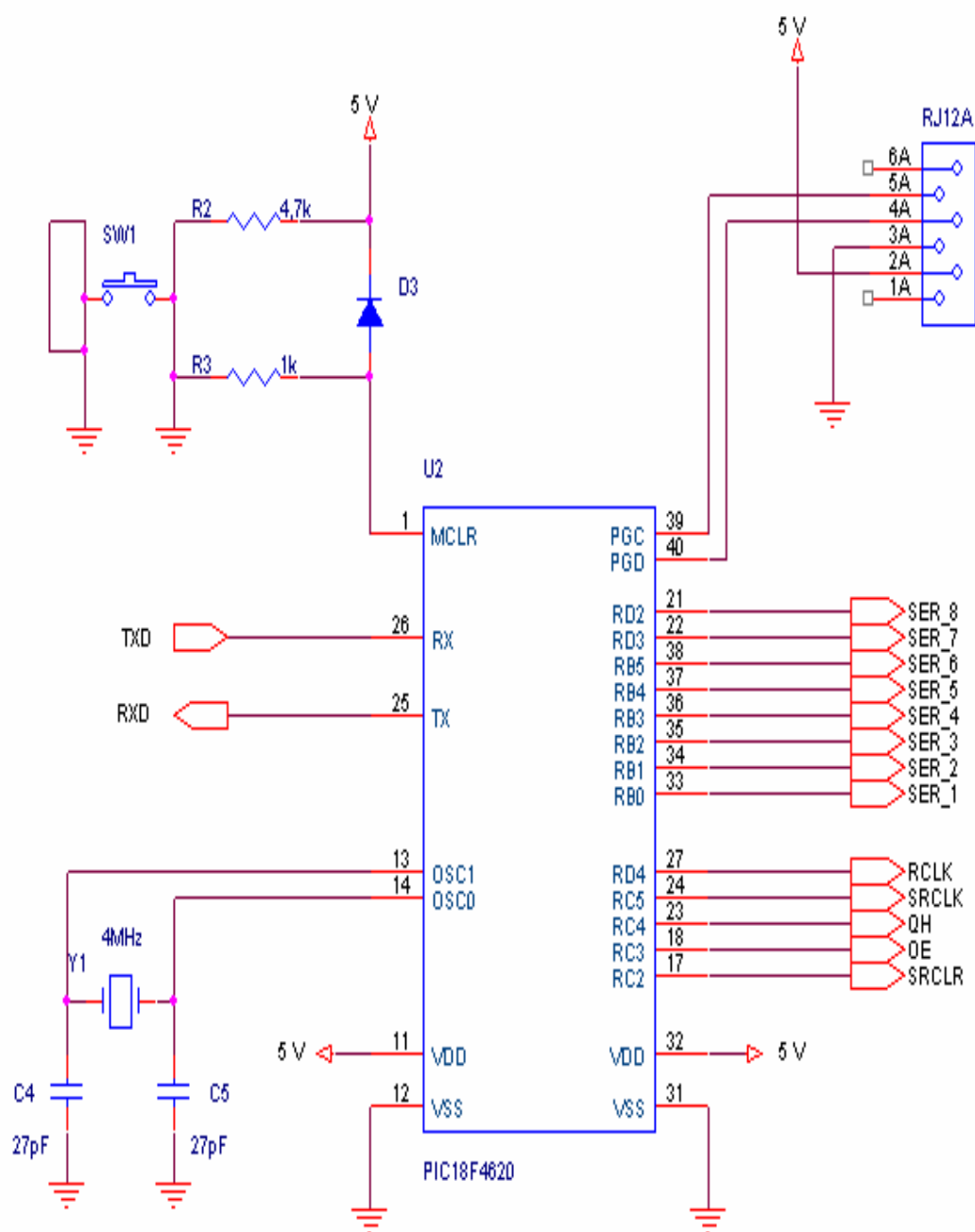
P.NOHAMA. Investigação em procepção artificial. Tese de Doutorado, UNICAMP, 1997.

P.RECH-Y-RITA, K.A.KACZMAREK, M.E.TYLER, M.GARCIA-LARA. Form perception with a 49-point electrotactile stimulus array on the tongue: P technical note. J.rehad. Res. Dev, Vol.35, 1998.

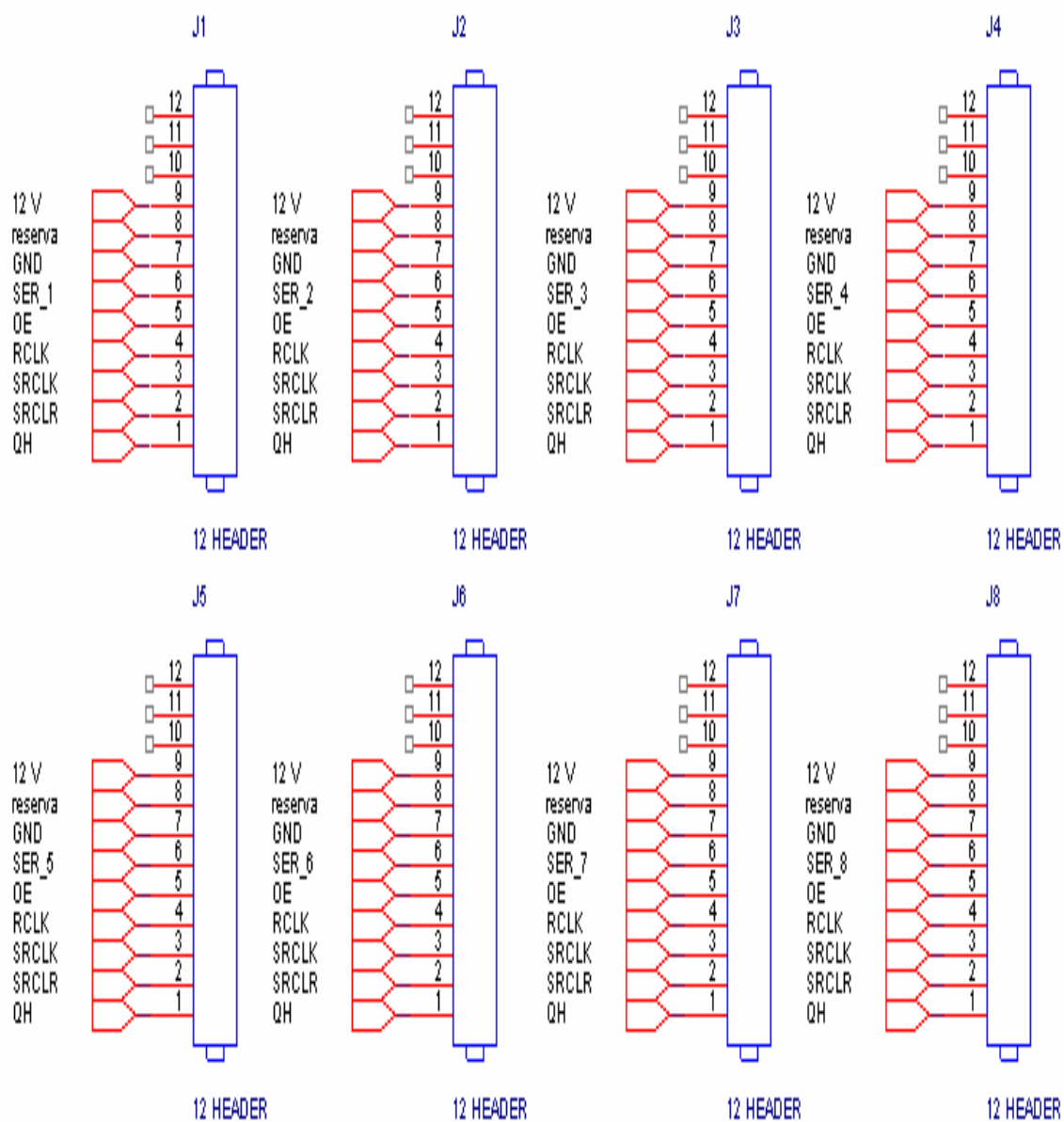
T.MAUCHER, M.KARLHEINZ, J.SCHEMMEL. Än interative tactile graphics display. Sixth international symposium on signal processing and its applications, ISSPA 2001.

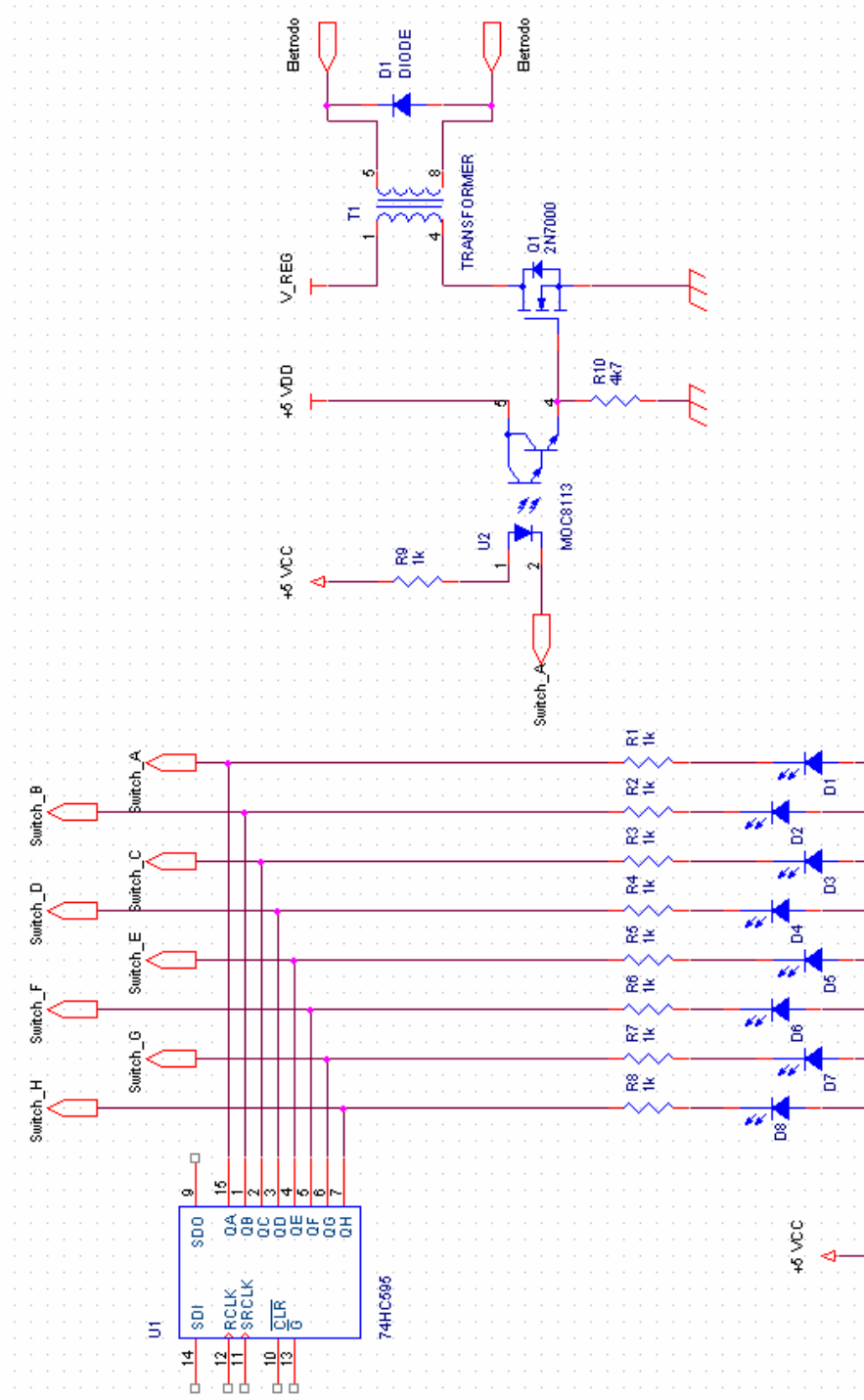
RIBEIRO, Bruno B. Um estudo de técnicas para o desenvolvimento de sistemas de processamento de imagens digitais. Niterói: UFF, 2002.

## ANEXO A









## ANEXO B

### Script *inicio.m* desenvolvido no Matlab

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                Detectando a webcam                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

imaqhwinfo
imaqhwinfo('winvideo')
imaqhwinfo('winvideo',1)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                Criando Objeto de Entrada de Video                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

vid = videoinput('winvideo',1,'RGB24_352x288')
get(vid)
get(getselectedsource(vid))

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                Tirando uma Foto                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
set(vid,'TriggerFrameDelay',60);
preview(vid)
t = timer('TimerFcn',@mycallback, 'Period', 100.0);
fcam = getsnapshot(vid);
figure: imshow(fcam);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                                Transformando a Imagem em Preto e Branco.                                %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

fgray = rgb2gray(fcam);
figure: imshow(fgray);

```

```
threshold = graythresh(fgray);
fpb = im2bw(fcam,threshold);
figure: imshow(fpb);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               Diminuindo a Resolução da Imagem Para 8x8.                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
f8x8 = imresize(fpb, [8 8]);
size(f8x8)
figure: imshow(f8x8);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               Detecção de Borda                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
f8 = double(f8x8);
mcanny = edge(f8, 'canny');
figure: imshow(mcanny);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               Transforma para Caractere                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
M = double(mcanny);
m = fliplr(M);
ascii = char(reshape(m+48, 64, 1)')
figure: imshow(m);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%                               Prepara Comunicação Serial e Envia o Dado                               %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
s4 = serial('COM4')
set(s4,'DataTerminalReady','off','RequestToSend','off');
fopen(s4)
fprintf(s4,ascii)
```

## ANEXO C

### Software desenvolvido no MPLAB para o PIC18F4620

```
/******  
Autor: Marcelo Wiggers Borges  
*****/  
  
#include <p18f4620.h>  
  
/*Set up the configuration bits*/  
  
/*CONFIG1H*/  
#pragma config IESO = OFF  
#pragma config FCMEN = OFF  
#pragma config OSC = HSPLL  
  
/*CONFIG2L*/  
#pragma config BOREN = OFF  
#pragma config BORV = 2  
#pragma config PWRT = OFF
```

```
/*CONFIG2H*/  
#pragma config WDTPS = 32768  
#pragma config WDT = OFF  
  
/*CONFIG3H*/  
#pragma config MCLRE = ON  
#pragma config LPT1OSC = OFF  
#pragma config PBAEN = OFF  
#pragma config CCP2MX = PORTC  
  
/*CONFIG4L*/  
#pragma config DEBUG = OFF  
#pragma config XINST = OFF  
#pragma config LVP = OFF  
#pragma config STVREN = OFF  
  
/*CONFIG5L*/  
#pragma config CP0 = OFF  
#pragma config CP1 = OFF  
#pragma config CP2 = OFF  
#pragma config CP3 = OFF  
  
/*CONFIG5H*/  
#pragma config CPB = OFF  
#pragma config CPD = OFF  
  
/*CONFIG6L*/  
#pragma config WRT0 = OFF  
#pragma config WRT1 = OFF  
#pragma config WRT2 = OFF  
#pragma config WRT3 = OFF  
  
/*CONFIG6H*/  
#pragma config WRTD = OFF  
#pragma config WRTB = OFF  
#pragma config WRTC = OFF  
  
/*CONFIG7L*/  
#pragma config EBTR0 = OFF  
#pragma config EBTR1 = OFF  
#pragma config EBTR2 = OFF  
#pragma config EBTR3 = OFF  
  
/*CONFIG7H*/
```

```
#pragma config EBTRB = OFF
```

```
////////// Função de Atraso 1 //////////
```

```
int delay (void)
{
    int ii;

    for (ii=0; ii<10000; ii++);
}
```

```
//////////
```

```
////////// Função de Atraso 2 //////////
```

```
int delay_2 (void)
{
    int ii;

    for (ii=0; ii<9000; ii++);
}
```

```
//////////
```

```
////////// Função de Atraso 3 //////////
```

```
int delay_3 (void)
{
    int ii;

    for (ii=0; ii<290; ii++);
}
```

```
//////////
```

```
void main (void)
{
```

```
    int j, indice, i, ii;
    char DATA;
```

```
char eletrodo [64];
//int a = 1;
```

```
////////// Parâmetros de Configuração da Serial //////////
/* TXSTA: TRANSMIT STATUS AND CONTROL REGISTER */
```

```
TXSTAbits.CSRC = 0; //Tanto faz (Don't care) para o modo "Asynchronous"
TXSTAbits.TX9 = 0; //Habilita 8-bits de transmissão
TXSTAbits.TXEN = 1; //Habilita Transmissão Serial
TXSTAbits.SYNC = 0; //Habilita o modo "Asynchronous"
TXSTAbits.SENDB = 0; //"Sync Break transmission completed"
TXSTAbits.BRGH = 0; //Habilita "Low Speed mode" para o Baud Rate Selected Bit
//TXSTAbits.TRMT =
//TXSTAbits.TX9D =
```

```
/* RCSTA: RECEIVE STATUS AND CONTROL REGISTER */
```

```
RCSTAbits.SPEN = 1; //Habilita Porta Serial
RCSTAbits.RX9 = 0; //Habilita 8-bits de transmissão
//RCSTAbits.SREN = 0; //Tanto faz (Don't care) para o modo "Asynchronous"
RCSTAbits.CREN = 1; //Habilita o para receber continuamente*****
//RCSTAbits.ADDEN =
//RCSTAbits.FERR =
//RCSTAbits.RX9D =
```

```
/* BAUDCON: BAUD RATE CONTROL REGISTER */
```

```
//BAUDCONbits.ABDVDF =
//BAUDCONbits.RCIDL =
//BAUDCONbits.SCKP =
BAUDCONbits.BRG16 = 0;
BAUDCONbits.ABDEN = 0;
//BAUDCONbits.WUE =
//BAUDCONbits.RXDTP =
//BAUDCONbits.TXCKP =
```

```
SPBRG = 24;
```



```

////////// Habilitação das Portas de I/O //////////
TRISA = 0b00000000;
TRISB = 0b00000000;
TRISC = 0b10000000;
TRISD = 0b00000000;

```

```

PORTCbits.RC3 = 1;

```

```

for (j=0; j<=100; j++)
{
    delay();
}

```

```

PORTCbits.RC2 = 1;
PORTCbits.RC3 = 1;
PORTCbits.RC5 = 0;
PIR1bits.RCIF = 0;

```

```

for (indice=0; indice<=63; indice++)
{
    while (!PIR1bits.RCIF);
    DATA = RCREG;
    eletrodo[indice] = DATA;
    PIR1bits.RCIF = 0;
}

```

```

for (i=0; i<=63; i++)
{
    switch (i)
    {
        case 0:
        case 8:
        case 16:
        case 24:
        case 32:

```

```

case 40:
case 48:
case 56:
if (eletrodo[i] == '0')
{
    PORTBbits.RB0 = 1;
}
else if (eletrodo[i] == '1')
{
    PORTBbits.RB0 = 0;
}

break;
case 1:
case 9:
case 17:
case 25:
case 33:
case 41:
case 49:
case 57:
if (eletrodo[i] == '0')
{
    PORTBbits.RB1 = 1;
}
else if (eletrodo[i] == '1')
{
    PORTBbits.RB1 = 0;
}

break;
case 2:
case 10:
case 18:
case 26:
case 34:
case 42:
case 50:
case 58:
if (eletrodo[i] == '0')
{
    PORTBbits.RB2 = 1;
}
else if (eletrodo[i] == '1')
{

```

```

        PORTBbits.RB2 = 0;
    }
    break;
case 3:
case 11:
case 19:
case 27:
case 35:
case 43:
case 51:
case 59:
if (eletrodo[i] == '0')
{
    PORTBbits.RB3 = 1;
}
else if (eletrodo[i] == '1')
{
    PORTBbits.RB3 = 0;
}
break;
case 4:
case 12:
case 20:
case 28:
case 36:
case 44:
case 52:
case 60:
if (eletrodo[i] == '0')
{
    PORTBbits.RB4 = 1;
}
else if (eletrodo[i] == '1')
{
    PORTBbits.RB4 = 0;
}
break;
case 5:
case 13:
case 21:
case 29:
case 37:
case 45:
case 53:
case 61:

```

```

if (eletrodo[i] == '0')
{
    PORTBbits.RB5 = 1;
}
else if (eletrodo[i] == '1')
{
    PORTBbits.RB5 = 0;
}
case 6:
case 14:
case 22:
case 30:
case 38:
case 46:
case 54:
case 62:
if (eletrodo[i] == '0')
{
    PORTDbits.RD2 = 1;
}
else if (eletrodo[i] == '1')
{
    PORTDbits.RD2 = 0;
}
break;
case 7:
case 15:
case 23:
case 31:
case 39:
case 47:
case 55:
case 63:
if (eletrodo[i] == '0')
{
    PORTDbits.RD3 = 1;
}
else if (eletrodo[i] == '1')
{
    PORTDbits.RD3 = 0;
}
delay_2();
PORTCbits.RC5 = 1;
delay_2();
PORTCbits.RC5 = 0;

```

```
        break;  
    }  
  
}  
  
    delay_2();  
    PORTCbits.RC5 = 1;  
    delay_2();  
    PORTCbits.RC5 = 0;  
  
    while(1) // (frequencia de 400Hz)  
    {  
        PORTCbits.RC3 = 0;  
        delay_3();  
        PORTCbits.RC3 = 1;  
        delay_3();  
    }  
  
}
```