ESCOLA POLITÉCNICA - UNIVERSIDADE DE SÃO PAULO

ESPECTROFOTOMETRIA EM LTCC

Pedro Gonçalves Alves

São Paulo

2010

ESCOLA POLITÉCNICA - UNIVERSIDADE DE SÃO PAULO

ESPECTROFOTOMETRIA EM LTCC

Trabalho de formatura apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do título de Graduação em Engenharia

Pedro Gonçalves Alves

Orientador: Antônio Carlos Seabra

São Paulo

2010

FICHA CATALOGRÁFICA

M2010B

Alves, Pedro Gonçalves
Espectrofotometria em LTCC / P.G. Alves. – São Paulo, 2010.
71 p.

Trabalho de Formatura - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Sistemas Eletrônicos.

1. Espectrofotometria 2. Instrumentação e medidas elétricas I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Sistemas Eletrônicos II. t.

DEDICATÓRIA

Gostaria de dedicar este trabalho aos que mais sofreram com ele, os que estiveram próximos de mim durante a execução. Aos meus familiares e amigos que agüentaram os momentos de mau humor e chatice devido aos problemas que aconteciam aqui ou ali. Àquela força e conversa confortante nas horas em que nada parecia dar certo. Devo lembrar também do Professor Seabra que esteve me ajudando desde o início da escolha do tema, sempre solícito quando precisei de sua força em algum aspecto, e da Zaira, que me ajudou durante toda a parte prática, me fornecendo material pra teste, tirando dúvidas e me auxiliando na execução do projeto. Agradeço a todos que se incluem nas categorias acima não somente durante a execução do trabalho, mas também em todo o período em que estive na POLI, já que este trabalho está sendo minha última atividade na escola e marco da conclusão de uma fase muito importante da minha vida.

RESUMO

Baseado no projeto de Espectrofotometria em LTCC da doutoranda Zaira Mendes do Laboratório de Sistemas Integráveis da Escola Politécnica da USP (LSI), este trabalho procura dar uma continuidade, ou se tornar uma ferramenta ao projeto incial. Através da mudança de alguns parâmetros, como hardware e software utilizado, possibilitará a análise de outras substâncias que não são foco do trabalho original. Como um trabalho de formatura, utilizamos uma placa de testes fornecida pelo Professor Seabra, e parte de uma placa de LTCC fornecida pela Zaira. A partir daí trabalhamos em cima de criar um protótipo que se assemelhasse com o objeto final do trabalho de doutorado, um analisador espetrofotométrico. Como objetivo proposto pelo professor, tínhamos que achar o ponto de inflexão da curva obtida pelo sistema de leitura, e esse resultado será apresentado num display de LCD. Ao final do projeto, percebemos que os objetivos iniciais, eram um pouco ousados, mas apesar disso, procuramos manter o foco nas partes mais importantes, para que pudéssemos concluí-lo e ter algo em mãos que poderá ser utilizado e aprimorado por trabalhos seguintes.

Lista de figuras e tabelas

SUMÁRIO

1. INTRODUÇÃO

1.1. Localização histórica/Importância do trabalho

Este projeto de formatura é baseado no projeto de doutorado com o tema "Espectrofotometria em LTCC". E procura durante seu desenvolvimento modificar alguns parâmetros para obter resultados mais qualificados e uma abrangência maior na utilização do dispositivo para a medição de componentes dissolvidos na água.

Sabe-se que o fósforo é um dos nutrientes essenciais para a vida de plantas e animais, e importante na participação de processos tais como: fotossíntese, transformações energéticas, entre outros. O fósforo é absorvido do solo através das raízes das plantas. E dentro das células, o fósforo tem papel importante no metabolismo e fisiologia das plantas, formando moléculas de ATP, fosfolipídios, ácidos nucléico, ésteres de fosfatos e coenzimas.

O fósforo é encontrado nos solos, rochas sedimentares e em águas naturais ou de descarte. Na água os compostos de fósforo podem ser encontrados de diversas maneiras, como: dissolvidos, partículas, detritos, plantas aquáticas e organismos.

Em águas naturais, os fosfatos, podem ser encontrados principalmente na forma de ortofosfato, mas também podem ser encontradas em diversas outras formas, como pirofosfatos, metafosfatos, polifosfato, fosfito, algum outro estado de oxidação ou como em algum composto orgânico de fósforo. Como já dito, os fosfatos são importantes à vida, mas em quantidades excessivas (devido a algum tipo de contaminação) podem causar o fenômeno conhecido como eutrofização das águas.

Este fenômeno se desenvolve nos lagos devido ao excesso de nutrientes dissolvidos, e o íon fosfato funciona normalmente como nutriente limitante do crescimento de algas: quanto maior o suprimento do íon, mais abundante será o desenvolvimento das algas. Quando a vasta massa de algas morre e começa a se decompor por oxidação, ocorre a diminuição de oxigênio dissolvido na água, o que acaba afetando negativamente a vida neste ambiente.

Dessa forma, o excesso de íons fosfato em águas naturais pode ter um efeito devastador na ecologia aquática, tornando o desenvolvimento de métodos analíticos para a detecção de fosfatos muito importante no monitoramento da poluição ambiental.

O método de espectrofotometria com LTCC também pode ser utilizado para se medir a qualidade da água em estações de tratamento de empresas fornecedoras como a Sabesp, por exemplo. E como dito anteriormente, as modificações visam modificar o sistema para que não só o fósforo seja bem determinado, como também o oxigênio dissolvido entre outras substâncias.

1.2. Objetivos

Aliado com as técnicas de espectrofotometria, utilizamos as técnicas analíticas em sistemas de fluxo (FIA) para analisarmos a mistura de água com os corantes durante os testes e caracterizações do projeto.

Através de modificações no hardware e no software do projeto atual, possibilitar a medição de outros componentes que não apenas de fosfatos.

No hardware atual, devemos nos aplicar no circuito de aquisição de dados (pontos), que será descrito posteriormente, para que possamos obter um maior número de pontos, um sistema mais rápido que tenha uma taxa de amostragem maior do que a do sistema atual (aproximadamente 10 ms). Precisaremos de um fotodetector que meça um sinal com comprimento de medida específico também (880 nm). E teremos também que trocar o microcontrolador atual.

Além da modificação no hardware que já temos no projeto, deveremos ter foco na parte de software também. Desenvolveremos um código-fonte, baseado em outros projetos já orientados pelo Professor Seabra, que trata os dados adquiridos pelo sistema. Pensaremos também em algum software para apresentarmos as curvas obtidas após o tratamento e alisamento dos dados obtidos pelo sistema de aquisição.

O objetivo do projeto inicial foi reforçado e especificado de uma melhor forma através de novas conversas com o Professor Seabra, orientador do mesmo.

Focamos como meta principal o tratamento (via software) dos dados obtidos pelo sistema. No tratamento da curva obtida, especificamos alguns valores para obtenção dos pontos.

O sinal de interesse para o cálculo terá a duração de aproximadamente 30 segundos. Iremos salvar os pontos numa variável do nosso microcontrolador, teremos um vetor de 1875 pontos, cada ponto distante 16 ms um do outro, além disso cada ponto será a média corrida de 16 pontos distantes 100 us uns dos outros. Ou seja, para termos o nosso vetor completo, acabaremos lendo 30 mil pontos do conversor A/D do microcontrolaldor PIC.

De forma mais clara, iremos ler da entrada do microcontrolador 16 pontos com intervalo de 100 us e após a leitura dos pontos, daremos um intervalo de 15 ms segundos, poia a intenção é que cada ponto esteja distante 16 ms um do outro. Após a leitura das 16 amostras, fazemos a média dos pontos e guardamos o ponto resultante no microcontrolador.

Com o sinal em mãos, iremos para o objetivo principal, que é o cálculo do ponto de inflexão. Daí através da primeira e segunda derivadas conseguiremos obtê-lo.

Esse trabalho visa a partir da tese de doutorado da Zaira, encontrar o valor no ponto de inflexão, pois isto irá servir como uma ferramenta útil na continuação deste projeto. Já que é uma forma de leitura e tratamento de dados que pode, através da correlação do valor obtido no ponto de inflexão, otimizar a obtenção da curva de absorbância de uma determinada substância diluída na água.

A interação com hardware foi brevemente discutida com o professor, já que se pretende escolher um modelo de PIC, microcontrolador já utilizado na tese de doutorado, que satisfaça os requisitos de armazenamento do código, e que tenha uma boa capacidade de processamento.

Além disso, poderemos mudar o detector do sinal de entrada, que segundo o professor, o original não se adequaria aos requisitos do sistema descritos acima.

1.3. Conteúdo do trabalho

Este trabalho procura descrever todo o processo desenvolvido durante as disciplinas de Projeto de Formatura I e II.

Nesta primeira parte apresentamos o tema e delineamos os primeiros objetivos propostos.

Na segunda parte buscamos identificar e esclarecer um pouco dos principais conceitos que utilizamos para a realização do trabalho.

Na terceira parte, descrevemos de que forma o trabalho foi realizado, como que, a partir dos conceitos obtidos, chegamos aos objetivos propostos. Ou seja, esta terceira parte descreve a etapa de desenvolvimento e teste de todas as partes do trabalho: software, hardware, montagem, integração das partes, problemas e dificuldades encontrados, entre outros.

Finalmente na última parte do trabalho, temos a conclusão do trabalho, que tem como meta principal relacionar os objetivos iniciais com os resultados obtidos, considerando e citando as principais modificações feitas, devido às dificuldades encontradas, limitações técnicas, ou algum outro motivo pertinente.

2. DESENVOLVIMENTO TEÓRICO

Nesta seção descreveremos as bases teóricas do trabalho. A partir das primeiras definições de objetivo do trabalho, procuramos realizar pesquisas para que pudéssemos delimitar melhor o foco do projeto. Parte dessa pesquisa se baseia no doutorado de Zaira Mendes, ainda nessa fase inicial do projeto, pesquisamos outros assuntos que utilizaríamos no decorrer do projeto, como: programação num dispositivo PIC; algoritmos para achar o ponto de inflexão de uma curva; e como utilizar um display de LCD.

2.1. Princípios da espectrofotometria

Espectrofotometria é um método de análise muito usado em na área de biologia e em investigações físico-químicas. O espectrofotômetro é, basicamente, um instrumento que compara a radiação absorvida ou transmitida por uma solução que contém uma quantidade desconhecida de algum componente (soluto).

Todas as substâncias absorvem luz, sejam no espectro visível ou não. A absorção de radiação, UV, infravermelho ou visível depende da estrutura de uma molécula, e é uma característica de cada substância química.

Uma forma simples de entender isso é observando a cor dos objetos, onde os objetos absorvem alguns comprimentos da luz branca e acabamos enxergando apenas os comprimentos da luz branca que não são absorvidos.

E a espectroscopia no ultravioleta visível (UV-VIS) envolve a espectroscopia de fótons, por isso é chamada espectrofotometria. E utiliza a luz na faixa visível, ultravioleta próximo e infravermelho próximo.

Um método utilizado para caracterizar a concentração de substâncias em uma solução que absorve radiação, de modo quantitativo, é a Lei de *Lambert-Beer* através da seguinte expressão:

$$A = -\log_{10}(I/I_0) = \varepsilon c b$$

onde A é a absorvância medida, I_0 é a intensidade da luz incidente em um dado comprimento de onda, I é a intensidade transmitida pela amostra, b é o caminho óptico percorrido pela amostra (distância que a luz percorre na amostra), □ é a constante de absorvidade molar, e é uma constante característica de cada substância, e finalmente c é a concentração da substância medida.

Para o caso geral da lei de *Lambert-Beer*, a absorção esta ligada linearmente com um dos fatores que não é constante do lado direito da expressão, e no nosso caso pode ser considerado como o caminho óptico b percorrido pelo feixe de luz através da amostra caracterizada. Podemos perceber o comportamento linear da lei através da figura seguinte:

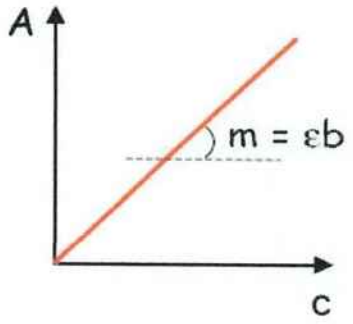

Figura 2-1 Comportamento linear da lei de Lambert-Beer

Como foi dito, existem casos em que a lei não é aplicada adequadamente, existe uma limitação real, que é quando a lei é aplicada a concentrações muito altas, onde a interação entra as moléculas pode acabar influindo na constante de absortividade molar da substância. Há também o desvio químico, onde um analito se dissocia, associa ou reage com um solvente, resultando num produto que tem espectro de absorção diferente.

Outro desvio que pode haver também é o desvio instrumental, para evitar esse desvio, procura-se escolher uma região do espectro onde o □ é constante. Essa propriedade pode ser melhor exemplificada através da figura seguinte:

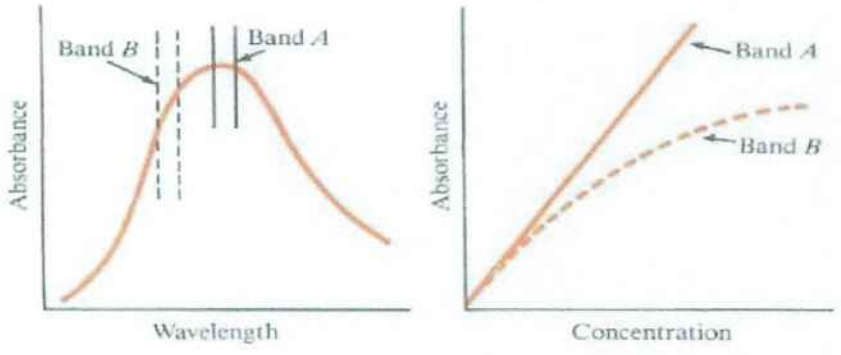

Figura 2-2 Exemplo da região ótima para se caracterizar uma concentração

2.2. Método de determinação do fosfato

As concentrações de fosfatos em amostras de água diferentes, podem variar muito. Desde valores menores do que 0,01 mg L^{-1} em águas extremamente limpas, até valores maiores do que alguns mg L^{-1} em águas poluídas de despejos industriais. Portanto existem diversas maneiras para se caracterizar a presença de fosfatos numa amostra, cada um mais adequado para um caso específico.

Alguns deles são os métodos de análise por injeção em fluxo (FIA) que para determinar a concentração de fosfatos utiliza técnicas com eletrodos de íons seletivos (ISE) e espectrofotometria UC/Vis.

Mas este projeto se baseia na análise através da espectrofotometria de UV/Vis, portanto deixaremos de lado os ISEs.

A espectrofotometria na região do UV/Vis é uma técnica analítica que tem sido bastante usada na determinação e analise de espécies orgânicas e inorgânicas. Em grande parte devido a sua precisão, exatidão, e custo relativamente baixo se comparado a outras técnicas.

A reação que utilizaremos neste projeto para a determinação de fosfatos, envolve a reação do íon ortofosfato com molibdato em meio ácido (ácido ascórbico), produzindo molibdofosfato. Para baixas concentrações de fosfatos, ocorre a redução de Mo(VI) para Mo(V), produzindo o composto azul de molibdênio, que tem uma coloração intensa, fator que melhora a sensibilidade do método.

A seguir temos a reação para a formação do azul de molibdênio:

$$Heptamolibdato + fosfatos \overset{rápida}{\longleftrightarrow} heterpoli\ complexo\ amarelo$$

$$Heterpoli\ complexo\ amarelo + ácido\ ascórbico \overset{lenta}{\longleftrightarrow} azul\ de\ molibdênio$$

A ordem da mistura dos reagentes influencia grandemente no processo, se, inicialmente, a solução de molibdato for misturada com ácido ascórbico (reação rápida), formará molibdato na forma reduzida, e injetando-se os fosfatos neste reativo, ocorrerá a formação rápida do azul de molibdênio (na ordem de segundos). Por outro lado, se os fosfatos forem injetados no fluxo da solução de molibdato, iram formar um produto de coloração amarela (fosfomolibdato), que não é muito reativo, pois possui uma baixa taxa de redução do azul de molibdênio (na ordem de minutos).

O método utilizado, do azul de molibdênio é susceptível a interferências da sílica, mas estas interferências podem ser facilmente eliminadas pela adição do reagente tartarato na solução de molibdato.

2.3. Técnicas analíticas em sistemas de fluxo (FIA)

Sistemas de análise em fluxo são comumente usados na análise de parâmetros ambientais, são usados em diversas áreas, mas especialmente na análise de água, onde esse tipo de análise é particularmente apropriado, devido a sua alta sensibilidade, o que acaba possibilitando baixos níveis de detecção. Os sistemas de análise por injeção em fluxo (FIA), particularmente, têm sido muito utilizados principalmente na mecanização/automatização de análises químicas, já que empregando esse tipo de sistema, é possível a implementação de praticamente todas as etapas envolvidas num processo de análise química, como: amostragem,

separações, diluições, pré-concentrações, adição de reagentes, entre outros. Hansen apresenta um banco de dados muito vasto, com mais de 12000 citações relacionadas à análise de águas utilizando sistemas de análise em fluxo (FIA).

A análise FIA consiste na injeção de um volume conhecido de uma amostra dentro de um fluxo contínuo de solução base ou padrão. Essa amostra atinge um ou mais eletrodos sensores, onde cada um é submetido a um potencial redox característico da espécie que será analisada. Esse detector mede o resultado dessa reação por meio de um sensor eletroquímico e a operação pode ser repetida por diversa s vezes. Portanto, esse sensor discrimina a solução base do analito no fluxo.

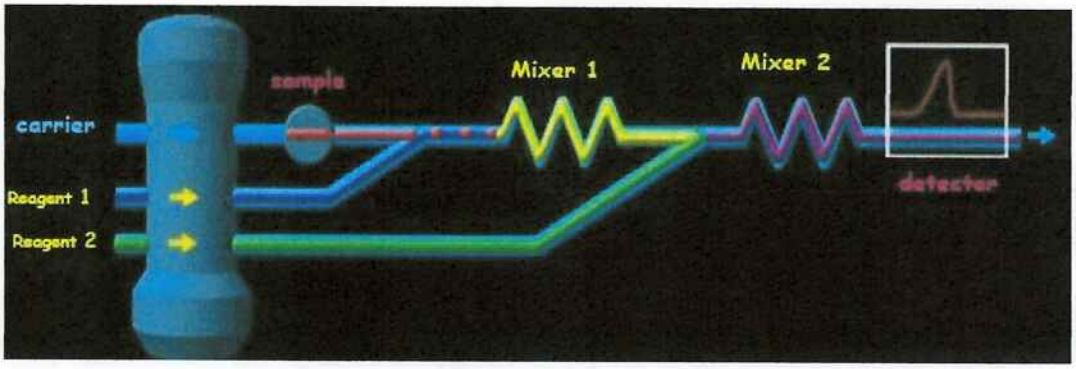

Figura 2-3 Esquema básico do funcionamento da análise FIA

Hoje em dia a miniaturização dos instrumentos analíticos assim como dos sistemas de análise em fluxo (FIAs) que utilizam tecnologias de micro fabricação, são de grande interesse para a química analítica. O maior atrativo disso tudo, é o aumento da demanda gerado pelo baixo custo dos instrumentos utilizados, que são capazes de analisar rapidamente componentes em volumes de amostras extremamente pequenos com um grau de automação consideravelmente alto.

A partir de 1990, o conceito de micro sistemas de análise total (μTAS) foi apresentado, e desde então o desenvolvimento de novos dispositivos está crescendo rapidamente. Através da fabricação de algumas microcâmaras de reação, separação e unidades de detecção no chip, já se obtém um grande aumento na velocidade de análise. Nesse tipo de sistema a diminuição no consumo de amostra e reagente, e a redução na produção de resíduos, assim como a portabilidade do dispositivo, tornam o sistema imensamente atrativo.

Devido às inúmeras possibilidades de aplicações dos dispositivos, os sistemas analíticos micro fabricados estão oferecendo um melhor desempenho em termos de rapidez na resposta e aumento na velocidade da análise. Para o desenvolvimento destes micro sistemas, de análises químicas, diversas tecnologias de fabricação têm sido empregadas, tais como silício, vidro em polímeros (como elastômeros), entre outros. A proposta nesse projeto é, alem de empregar essas tecnologias e materiais, acrescentar o uso de cerâmicas com baixa temperatura de sinterização (LTCC – Low Temperatura Cofired Ceramics) devido as vantagens de micro fabricação que seu uso oferece. No entanto, o uso desses materiais em μTAS requer estudos intensos de desempenho e compatibilidade.

2.4. Cerâmica com baixa temperatura de sinterização

Recentemente, a indústria de cerâmicas eletrônicas desenvolveu uma nova tecnologia de substratos com aplicações em microeletrônica: substratos maleáveis que podem ser associados como um único corpo formando uma estrutura de

múltiplas camadas. Este sistema de filmes espessos permite a deposição de filmes condutores, resistivos e dielétricos em seu corpo.

As firas de cerâmica verde, da tecnologia LTCC, são vitro-cerâmicas com estruturas micro-granulares muito uniformes compostas essencialmente de alumina (Al_2O_3) e óxido de silício (SiO_2). Solventes, plastificantes e um ligante (binder) compõem a parte orgânica deste material, os quais conferem à fita LTCC a maleabilidade e flexibilidade, tornando-a facilmente trabalhável. A temperatura necessária para o enrijecimento dessas fitas é 850°C, que é uma temperatura considerada baixa (se comparada às temperaturas de sinterização dos demais materiais cerâmicos, tais como alumina(1400°C)) porque depende somente do amolecimento da parte vítrea. Estas fitas são geradas pelo método "Doctor Blade", obtendo-se espessuras de 100 a 400μm.

A tecnologia permite a geração de estruturas mecânicas (orifícios, vigas, pontes, canais, cavidades) em tamanhos intermediários, na faixa de 100μm a centenas de micrometros de maneira simples, já que as fitas de LTCC em estado "verde" (antes da sinterização) são facilmente processadas. Após o processamento individual das fitas, estas são laminadas obtendo-se um sistema multicamada. Quando este laminado é sinterizado, gera um corpo rígido com a aplicação desejada (canais, sensores, válvulas, entre outros).

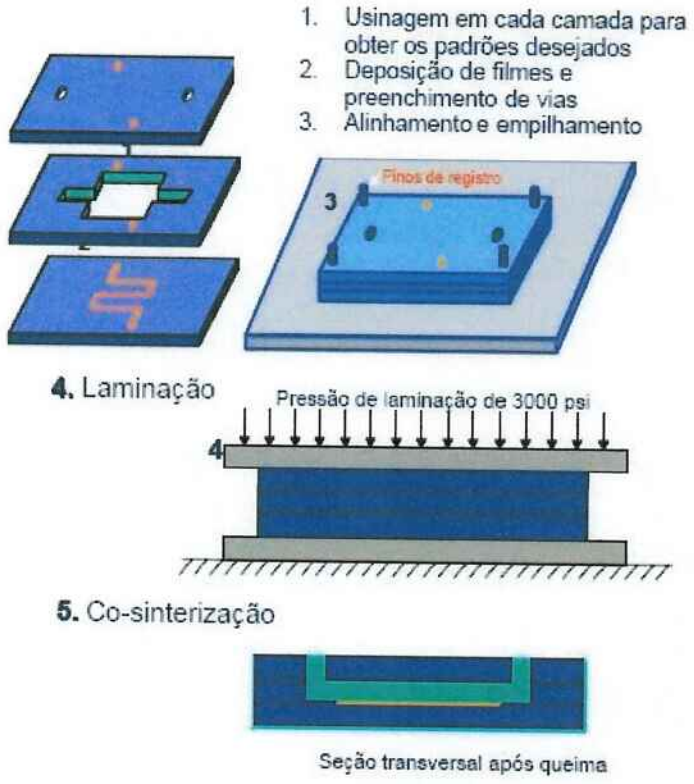

Figura 2-4 Processo de fabricação de uma cavidade com um aquecedor, usando a tecnologia LTCC de fitas cerâmicas verdes

As principais vantagens da tecnologia LTCC de fitas cerâmicas são:

- Simplicidade de usinagem da fita para tamanhos característicos de 100μm até unidades de milímetros;
- Métodos de produção em larga escala podem ser imediatamente aplicados;
- Propriedades termo-físicas do LTCC, como a condutividade térmica, podem ser modificadas;
- Fitas com propriedades específicas, como maior permeabilidade magnética, podem ser obtidas;

- Facilidade da realização de conexões entre camadas (elétricas ou fluídicas);
- Possibilidade de implantação de componentes passivos enterrados (tais como resistores e capacitores construídos com pastas resistivas, por exemplo);
- Fácil integração com circuitos eletrônicos;
- Sistemas com aproximadamente 50 camadas;
- Possibilidade de fabricação de dispositivos auto-encapsulados;
- Técnicas de fabricação simples, baratas e ecologicamente corretas.

2.5. Ponto de Inflexão

Baseado nos objetivos que discutimos com o professor, temos que um dos principais focos do trabalho é achar o ponto de inflexão de uma curva, através da derivação da mesma. A seguir explicaremos rapidamente o significado deste ponto na curva e exemplificaremos alguns casos.

A designação do ponto de inflexão está usualmente associada a uma mudança do sentido da concavidade (para cima ou para baixo) do gráfico de uma função à esquerda e à direita desse ponto, isto é, uma função f tem uma inflexão para $x = a$, ou no ponto $(a,f(a))$, se no ponto $(a,f(a))$ se verifica a mudança do sentido de concavidade do seu gráfico.

Assim, a função f representada no gráfico seguinte, tem uma inflexão para $x = a$:

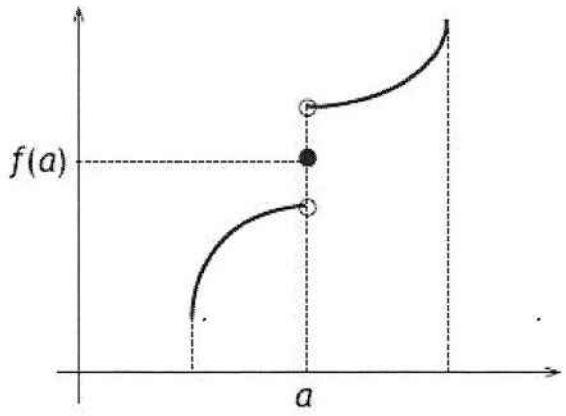

Figura 2-5 Função com ponto de inflexão em $x=a$

A designação do ponto de inflexão é geralmente reservada para os pontos onde a função é contínua.

Temos duas definições:

Definição 1: seja I um intervalo aberto, a um ponto de I, e f uma função contínua em I. A função tem um ponto de inflexão para $x = a$, ou no ponto $(a,f(a))$, se existe $\varepsilon > 0$ tal que o gráfico de f tem a concavidade voltada para cima (para baixo) em $]\, a - \varepsilon,\, a\,[$ e voltada para baixo (para cima) em $]\, a,\, a + \varepsilon\,[$.

Nas figuras seguintes ilustram-se pontos de inflexão, de acordo com a definição anterior:

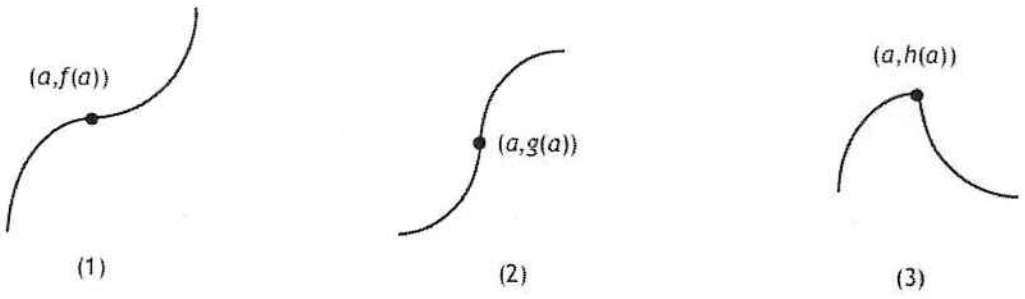

Figura 2-6 Exemplos de curvas com mudanças na concavidade

É importante salientar, que um grande número de autores define ponto de inflexão apenas para funções com derivada (finita ou infinita) nesse ponto. Assim, não consideram que exista ponto de inflexão no caso (3). Da observação dos exemplos (1) e (2) parece que a existência de inflexão está associada à posição relativa entre a tangente no ponto (a tracejado na figura) e o gráfico da função, à sua direita e à sua esquerda.

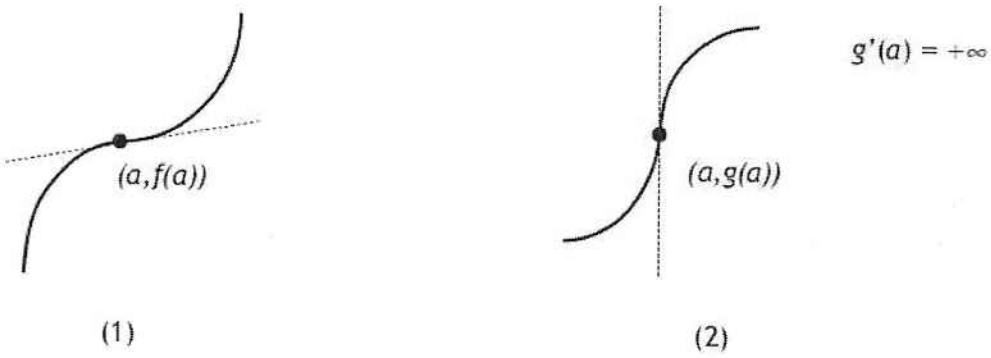

Figura 2-7 Exemplo da tangente no ponto de inflexão das curvas

Temos a partir da observação das curvas acima a seguinte definição:

Definição 2: seja I um intervalo aberto de IR, $a \in I$, $f: I \rightarrow IR$ uma função com derivada no ponto a. Designe-se por t a tangente em $(a,f(a))$.

(i) Se f tem derivada finita em a, o ponto $(a,f(a))$ é um ponto de inflexão se em $]\, a - \varepsilon,\, a\,[$ o gráfico de f está acima (abaixo) de t e em $]\, a,\, a + \varepsilon\,[$ o gráfico de f está abaixo (acima) de t.

(ii) Se f tem derivada infinita em a, o ponto $(a,f(a))$ é um ponto de inflexão se em $]\, a - \varepsilon,\, a\,[$ o gráfico de f está à direita (à esquerda) de t e em $]\, a,\, a + \varepsilon\,[$ o gráfico de f está à esquerda (à direita) de t.

Para as definições 1 e 2 encontram-se exemplos em que existe ponto de inflexão de acordo com uma delas e não existe ponto de inflexão de acordo com a outra.

Das definições, nenhuma é "mais verdadeira" que a outra. Elas podem coincidir em algumas situações, mas são definições diferentes, e podem, dependendo da definição que for usada, conduzir a conclusões diferentes. Portanto numa análise desse tipo é essencial escolher qual definição será adotada.

Vale ressaltar que para o nosso caso neste trabalho, o trecho do sinal que queremos analisar se assemelha com a curva (2) nos dois exemplos dados acima. Logo, não temos que nos alongar em discussões mais filosóficas sobre o que é ou não ponto de inflexão para uma ou outra definição. Este esclarecimento já basta como base teórica para utilizarmos o conceito no projeto.

2.6. Programação com PIC

O PIC é um circuito integrado produzido pela Microchip Technology Inc, que pertence a categoria dos microcontroladores, ou seja, um componente integrado que

em um único dispositivo contem todos os circuitos necessários para realizar um completo sistema digital programável. O PIC pode ser visto externamente como um circuito integrado TTL ou CMOS normal, mas internamente dispõe de todos os dispositivos típicos de um sistema microprocessado, ou seja: Uma CPU (Central Processor Unit ou Unidade de Processamento Central) e sua finalidade é interpretar as instruções de programa; Uma memória PROM (Programmable Read Only Memory ou Memória Programável Somente para Leitura) na qual irá memorizar de maneira permanente as instruções do programa; Uma memória RAM (Random Access Memory ou Memória de Acesso Aleatório) utilizada para memorizar as variáveis utilizadas pelo programa; Uma série de LINHAS de I/O (entrada e saída) para controlar dispositivos externos ou receber pulsos de sensores, chaves, etc.; Uma serie de dispositivos auxiliares ao funcionamento, ou seja, gerador de *clock*, *bus*, contador, etc. A presença de todos estes dispositivos em um espaço extremamente pequeno, da ao projetista ampla gama de trabalho e enorme vantagem em usar um sistema microprocessado, onde em pouco tempo e com poucos componentes externos podemos fazer o que seria oneroso fazer com circuitos tradicionais. O PIC esta disponível em uma ampla gama de modelos para melhor adaptar-se as exigências de projetos específicos, diferenciando-se pelo numero de linha de I/O e pelo conteúdo do dispositivo. Inicia-se com modelo pequeno identificado pela sigla PIC12Cxx dotado de 8 pinos, até chegar a modelos maiores com sigla PIC18Cxx dotados de 40 pinos.

2.7. Display LCD

O display de LCD é largamente utilizado em diversos aparelhos eletro-eletrônicos com a finalidade de mostrar resultados preliminares ou informações que auxiliem no manejo do aparelho.

Para colocá-lo em funcionamento, primeiro precisamos configurá-lo, ou seja, precisamos dizer ao display como vamos transferir os dados para ele (8 ou 4 bits), quantas linhas vamos utilizar, se a mensagem deve ficar fixa ou rolar, se a escrita será da esquerda para direita ou da direita para esquerda, ou seja, todas essas configurações são necessárias antes de escrever qualquer mensagem. Os manuais dos displays normalmente trazem. Mas veremos adiante que para o projeto, saber como mandar um comando para o display de LCD será irrelevante. Portanto basta sabermos como ele funciona, e para que serve cada pino.

Outro detalhe importante quando se trabalha com este tipo de display, é a temporização. Devemos ter um cuidado especial com este ponto, pois uma temporização equivocada inviabilizará o funcionamento do mesmo (não danificará, mas também não funcionará).

Sendo assim, vamos conhecer a pinagem do display LCD 16x2:

- Pinos de dados: **D7** - **D6** -... - **D1** - **D0** (8bits)- os pinos de dados são usados para enviar as palavras de configurações e os dados (caracteres).
- Pinos de controle: **EN** (6), **RS** (4), **R/W** (5) - o pino **EN** informa ao display de LCD quando o dado está pronto para ser lido. O pino **RS** é usado para

diferenciar se a palavra que foi enviada ao LCD é de configuração ou caractere.

- Pinos de alimentação: **VCC** (2) e **GND** (1).
- Pino de controle de contraste: **VO** (3) - este pino permite alterar o contraste do display.
- Pinos de iluminação do fundo - *backlight*: **A** (16), **K** (15) - nem todos os displays possuem iluminação de fundo.

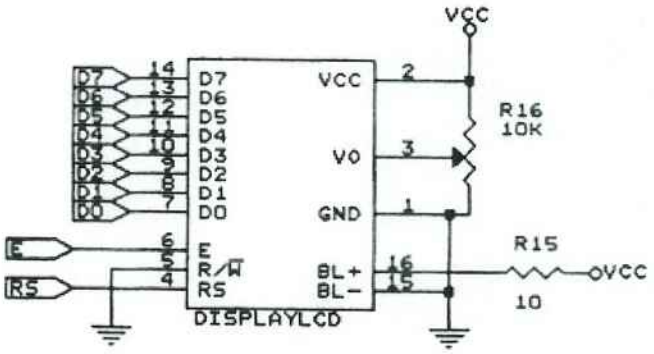

Figura 2-8 Esquema elétrico do display de LCD

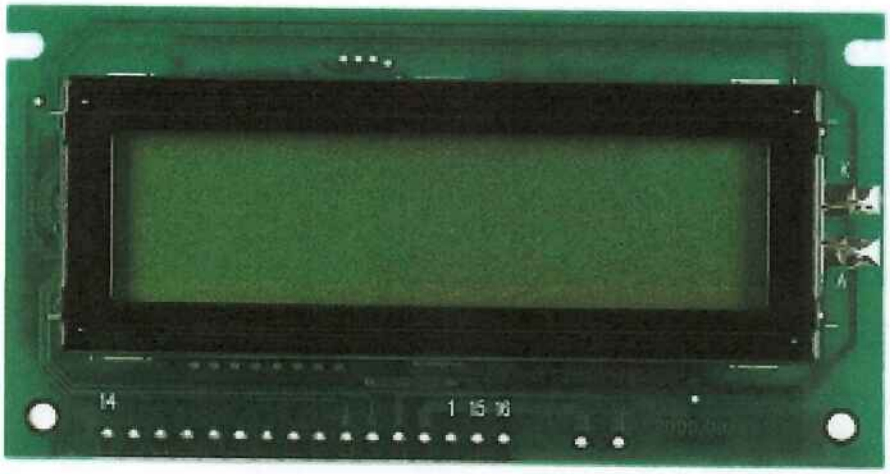

Figura 2-9 Exemplo do display de LCD utilizado

3. DESENVOLVIMENTO PRÁTICO

Esta seção descreverá todos os passos envolvidos na disciplina Projeto de Formatura II, que consiste no desenvolvimento prático do projeto propriamente dito. Separaremos esta parte prática em várias etapas, da maneira como o trabalho realmente foi feito, as etapas foram:

- Desenvolvimento e testes do código (no PC)
- Reuniões de acompanhamento com o professor orientador
- Incorporação da placa de testes fornecida pelo professor
- Caracterização do sistema fluídico com o conjunto de leitura (fotodiodo-led)
- Incorporação do display de LCD ao ambiente do projeto
- Transposição do código do PC para o microcontrolador PIC
- Integração do sistema fluídico e leitura com o sistema de tratamento PIC
- Testes finais de funcionamento

3.1. Desenvolvimento e testes do código

Baseado na necessidade (e objetivo) do projeto, já descrita anteriormente, na breve teoria pesquisada, e logicamente, no conhecimento de cálculo adquirido desde o primeiro ano do curso de engenharia, utilizamos conceitos de cálculo para calcular a primeira e a segunda derivadas no sinal obtido, a partir dos pontos adquiridos pelo sistema (microcontrolador). De posse da segunda derivada, simplesmente encontramos o(s) ponto(s) onde a segunda derivada se iguala a zero, e ali temos o ponto de inflexão da curva.

Tendo essa idéia em mente, queremos escrever um programa que trata essa curva obtida. A idéia foi discutida com o professor Seabra, sobre os tempos de aquisição do sinal na entrada do microcontrolador (que será exposto posteriormente), além disso, temos queremos fazer uma media corrida dos pontos da curva para o alisamento dos dados, daí sim fazemos as derivadas e obtemos o ponto requerido.

Como iremos implementar o código num microcontrolador PIC, o código completo deverá ser escrito em linguagem C, para posteriormente ser compilado e "traduzido" no ambiente do programa **MPLAB** com o auxílio da ferramenta **CCS C Compiler** .

O primeiro passo no código foi implementar um algoritmo para o cálculo das derivadas, utilizamos uma forma simples de derivação numérica, achamos um delta entre dois pontos com uma distância de duas unidades no eixo "tempo" e dividimos por essa distancia. Para a primeira derivada calculamos 9 pontos auxiliares em torno de um ponto principal e a partir desses pontos calculados da primeira derivada, calculamos 7 pontos da segunda derivada. Utilizando os pontos calculados da segunda derivada, fazemos uma média com os 5 pontos centrais e obtemos uma estimativa de um ponto para a nossa segunda derivada. Esse ponto calculado que será posteriormente testado, e no caso de ser zero, será considerado o ponto de inflexão.

Com o algoritmo de derivação, partimos para a estratégia que utilizaremos no código propriamente dito. Desta forma, o *main()* do programa será basicamente dividido nos seguintes estados:

- **INICIAR**: para o nosso código de simulação, este estado apenas escreve um aviso na saída e chama o estado seguinte que será **NORMAL**;

```
printf("\n ESTADO INICIAR\n----------\n");
 i8rotina = NORMAL;
```

- **NORMAL**: que lê a entrada do sinal obtido (inicialmente um arquivo texto que usaremos para a simulação computacional, e posteriormente os pontos obtidos a partir do microcontrolador PIC), guarda num variável do programa, que é um *array* com o número de pontos necessário para tal, e faz a média dos pontos, guardando-os novamente no *array*. Por último, este estado chama o próximo estado, que será **CALCULAR**;

```
printf("\n ESTADO NORMAL\n----------\n");
i16idx  = 0;
while(i16idx < ARRAY_MAX)
{
  ler_video(i16idx);
  media_video(i16idx);
  i16idx++;
}
i8rotina = CALCULAR;
grava_arquivo("C:\\Users\\Pedro\\Trabalho Formatura\\simulacao\\vetor_media.txt");
```

- **CALCULAR**: este estado, talvez o mais importante, ou principal, do nosso programa, calcula a primeira e a segunda derivadas ponto a ponto e vai buscando o possível ponto de inflexão. Se achar o ponto de inflexão, mostra o valor na saída correspondente, zera o contador do vetor dos pontos e vai para o estado **REINICIAR**;

```
printf("\n ESTADO CALCULAR\n----------\n");
 //pela analise no matlab par 30000 o pto de inflexao deve estar entre 7500
 // para 1875 proximo de 468.75
 int iInflxLimInf=0;
 int iInflxLimSup=0;
 for ( int i16idxDer = 5; !bAchouPtoInflx && i16idxDer <= ARRAY_MAX-5;
i16idxDer++)
 {
    calcula_derivadas(i16idxDer);
    if(!iInflxLimInf && !iInflxLimSup && si16segundaDerPosterior == 0)
    {
       iInflxLimInf = i16idxDer;
       printf("\n----------\n Ponto de Inflexao Inicial em %d\n\n",i16idxDer);
    }
    if(iInflxLimInf && !iInflxLimSup && si16segundaDerPosterior !=0)
    {
       iInflxLimSup = i16idxDer;
       bAchouPtoInflx = FALSE;
       printf("\n Ponto de Inflexao  Final em %d\n",i16idxDer);
       bAchouPtoInflx = TRUE;
    }
    if(bAchouPtoInflx)
    {
       printf("\n Ponto de Inflexao  em %d\n",iInflxLimInf + (iInflxLimSup-
iInflxLimInf)/2);
     }
 }
 i16idx = 0;
 bAchouPtoInflx = FALSE;
 system("PAUSE");
 i8rotina = REINICIAR;
```

- **REINICIAR**: este que é também o estado ***default*** será também um estado de aguardo no microcontrolador, ele zera o passo de leitura e chama o estado **INICIAR**, posteriormente explicaremos por que este estado é um estado de aguardo.

```
printf("\n ESTADO REINICIAR\n----------\n");
```

```
i16passo=0;
i8rotina = INICIAR;//inicia
system("PAUSE");
```

Explicados e mostrados os principais pontos do nosso código de simulação computacional, tivemos que criar um sinal de entrada que pudesse ser utilizado pelo nosso programa.

A estratégia foi utilizar o ambiente do **MATLAB**, com a ferramenta do Simulink, para criar um sinal semelhante ao sinal que teremos no sistema real. Este sinal é valido no sentido de que tem um período próximo ao que queremos analisar do sinal real (cerca de 30 segundos), e tem um formato semelhante ao sinal real, algo parecido com a "mistura" de meio período de um seno e uma função degrau.

Abaixo temos o esquema do gerador do sinal que utilizamos, que nos mostra também o resultado das derivadas:

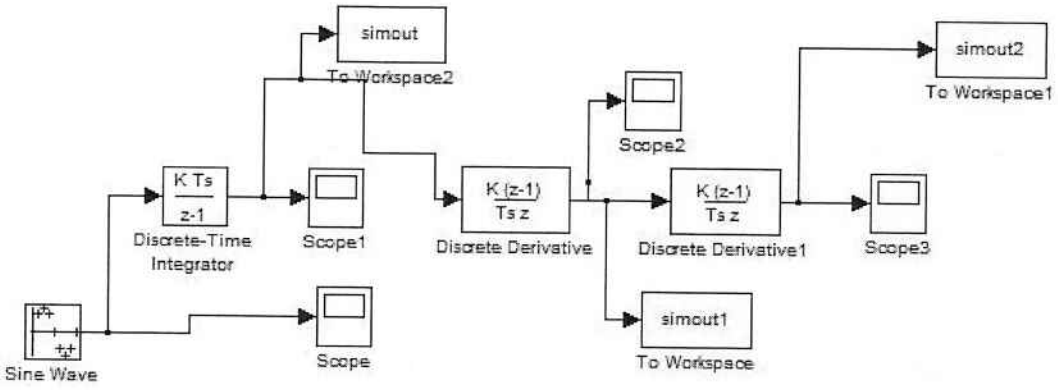

Figura 3-1 Gerador do sinal para a simulação do sistema

Criamos um seno discreto e integramos (discretamente também) esse sinal para obtermos o sinal que utilizaremos na simulação. Com o sinal criado, gravamos o vetor correspondente num arquivo *.txt* que será utilizado como entrada do nosso programa de simulação. O interessante de se utilizar o **MATLAB**, é que após a criação do sinal, no **Simulink**, fazemos as duas derivadas, e através da observação das curvas obtidas podemos ter a previsão do resultado esperado para o ponto de inflexão da curva. A criação dos sinais pelo **Simulink**, e os respectivos gráficos estão descritos abaixo, bem como seus pontos notáveis:

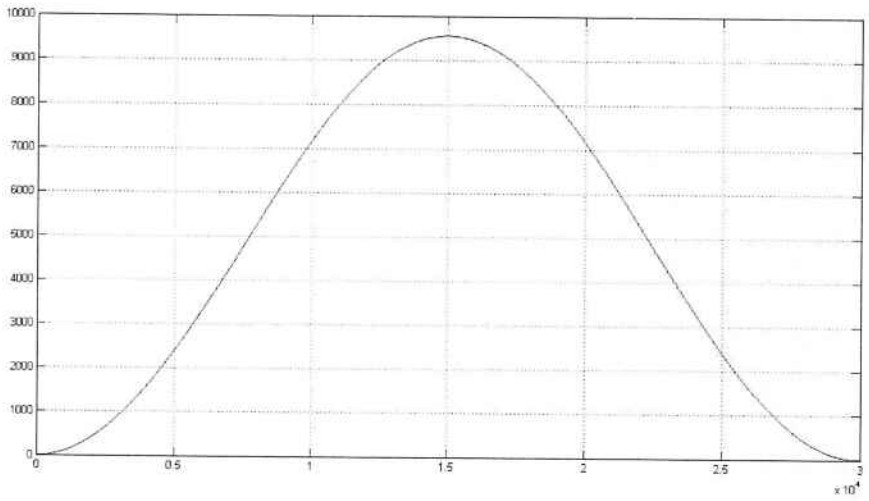

Figura 3-2 Simulação do sinal de entrada

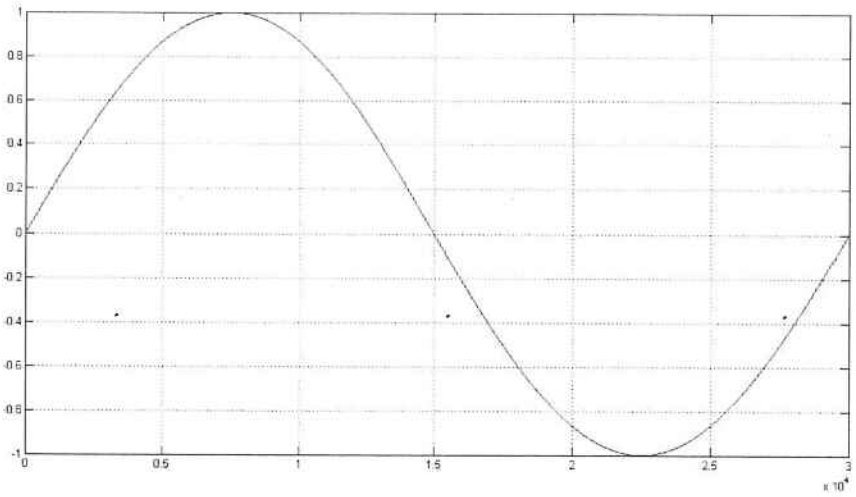

Figura 3-3 Primeira derivada do sinal simulado

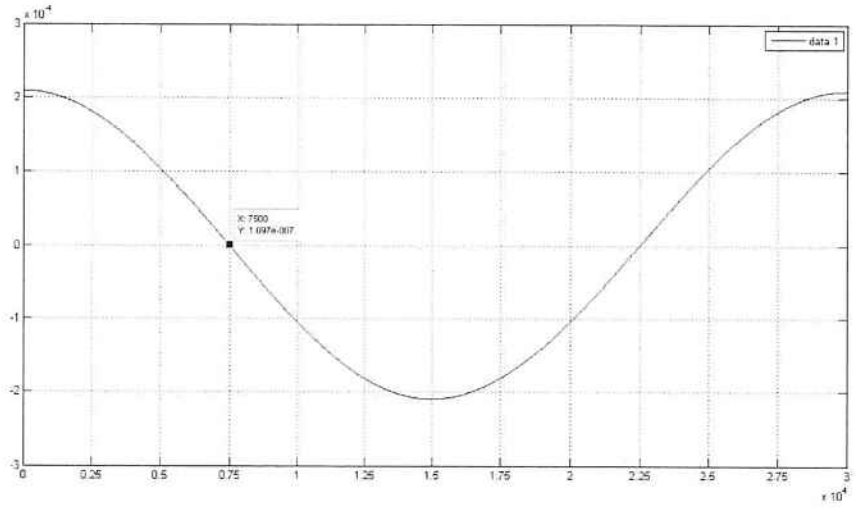

Figura 3-4 Segunda derivada do sinal de simulação com marcação do ponto de inflexão

Como podemos ver no gráfico da curva da segunda derivada do sinal que utilizaremos de entrada, temos o primeiro (justamente este ponto que procuramos) ponto de inflexão em 7500. Com isso em mãos, testamos o programa para ver se seu resultado se aproxima do valor esperado.

No trecho do programa que usamos para simulação, é mostrado apenas o índice do ponto (no vetor) onde ocorre a inflexão, porém para o programa que será colocado no microcontrolador é requisito do projeto mostrar o valor associado ao ponto, para posteriormente ser correlacionado com a absorbância e um perfil de curva que se adéqüe com este valor. Vale dizer que, dada a resolução dos pontos (principalmente no PIC que é de 10 bits) que são inteiros, buscamos encontrar o ponto médio entre os pontos do vetor que possuem a segunda derivada igual a zero. Esse detalhe pode ser visto no trecho de código do estado **CALCULAR**, que se encontra acima.

Por fim, realizando os testes do programa de simulação com o vetor, obtivemos sucesso, e encontramos um ponto muito próximo do esperado para a curva criada para simulação. Abaixo o resultado do programa, que é bem simples e retorna apenas o necessário para a verificação dos cálculos:

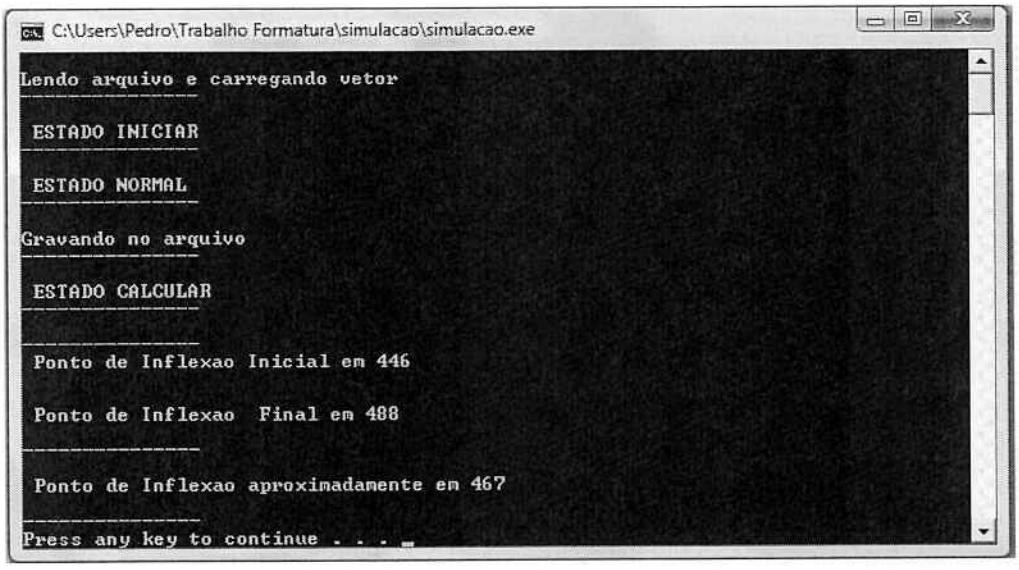

Figura 3-5 *Screenshot* com resultado da simulação do programa

Portanto temos acima a validação do código, com respeito aos cálculos do ponto de inflexão.

3.2. Reuniões de acompanhamento

Durante o semestre, em conversas, tanto com o Professor Seabra quanto com a doutoranda Zaira Mendes, pudemos definir em qual direção seguir com o trabalho.

Confirmamos os valores de intervalo na leitura dos dados pelo PIC com o professor. E em conversas com a Zaira, definimos que para resultado final do trabalho, basta que apresentemos o valor da voltagem (que é a grandeza obtida da leitura) no ponto de inflexão da curva formada pelos pontos obtidos da leitura. Lembrando que esta curva varia 0 e 5V. A partir do ponto de inflexão que achamos, ela o relacionaria com outros valores de acordo com a solução utilizada e poderia encontrar o valor da concentração da substância na água.

Outro ponto que definimos com o professor no início da fase de testes que será descrita a seguir (seção que descreve a placa de testes), foi que para otimizar nosso sistema, deveríamos apresentar o valor (do ponto de inflexão) num display de LCD 16x2. Dessa forma, não teríamos que ficar lendo valores num PC ou em um notebook via conexão serial, o que faz com que o sistema, ainda que em fase de protótipo, fique mais autônomo.

Após os encontros, o professor forneceu uma placa, que era usado por outros grupos do LSI para outros fins, para testarmos o PIC com entrada de dados de testes junto com alguma descrição esquemática que será descrita a seguir, e a Zaira me

auxiliou com a parte de captura de dados e o sistema fluídico, que serão descritos nas próximas seções também.

3.3. Placa de Testes

O Professor Seabra forneceu uma placa de testes para que, de maneira rápida, tivéssemos um ambiente quase que totalmente preparado para testar os PICs. Descrevendo de maneira genérica, a placa possui encaixe para um PIC de 40 pinos, alimentações, *clock* para o PIC, entradas analógicas, led (um na placa mais duas saídas), saídas COM e outros tipos de conexões. Temos também na placa os pinos preparados para utilizarmos o display de LCD que será importante na apresentação do resultado final do trabalho.

Devido ao fato da placa ser genérica ela pode ser usada para diversas funções. Sendo assim, muitos dos módulos não precisaram ser utilizados na realização do projeto. Portanto descreveremos, com o auxílio dos esquemáticos, apenas as partes utilizadas, e de que forma foram utilizadas. Os módulos são descritos a seguir:

- **Comunicação:** neste módulo temos o conjunto de pinos (**COM1-1**, **COM1-2** e **COM1-3**) para a porta **COM1** e o conjunto de pinos (**COM2-1**, **COM2-2** e **COM2-3**) para a porta **COM2**. Os pinos **TX1**, **TX2**, **RX1** e **RX2** são, respectivamente, pinos de saídas e entradas do PIC que são transmitidos para as portas **COM1** e **COM2**. Este módulo, inicialmente, era de algum interesse, até decidirmos que seria melhor obter as saídas no display de LCD ao invés de utilizar as portas **COM**. Portanto na seqüência do trabalho, acabamos não utilizando as portas **COM.**

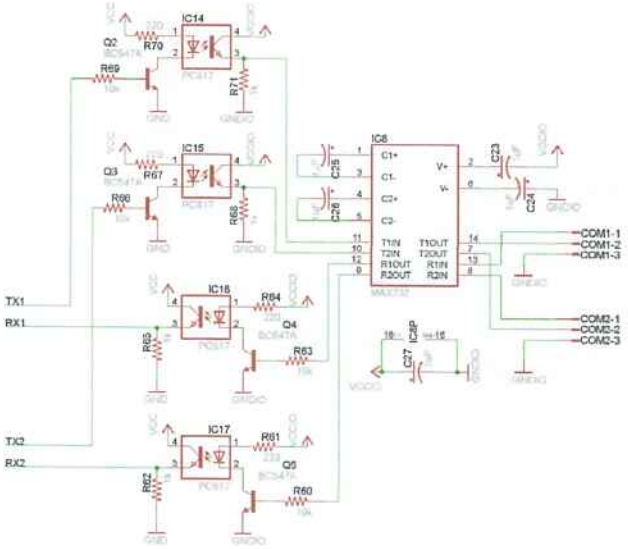

Figura 3-6 Módulo de Comunicação

- **Fontes:** neste módulo temos as alimentações de toda a placa. As saídas, que podem ser acompanhadas nos outros esquemáticos, são **+15V**, **-15V**, **VCC** e **VCCIO** que são alimentadas pelos conjuntos de pinos **+-24VDC**, **+-12VDC** e **+-12VDC-IO**. Claro que temos os pinos de terra também, mas estes podem ser compartilhados. O pino de **VCCIO** alimenta as portas de serial (**COM1** e **COM2**), **+-15V** alimentam os AmpOps que são usados nas entradas de sinal analógico (que serão descritas posteriormente), **VCC** alimenta basicamente todos os outros componentes e os pinos **LCDBL** (**A** e **K**) são pinos que normalmente são usados para alimentar o *backlight* do display de LCD, mas veremos adiante no esquemático de montagem do display de LCD, que utilizamos estes pinos para alimentá-lo, e um outro **GND** e **VCC** para acender o *backlight*.

  Tivemos certa dificuldade com este módulo, por que diferente do que temos nos esquemáticos, não conseguimos alimentá-los com as tensões nominais. Ao ligarmos fontes com as tensões nominais nestes pinos, observávamos que alguns CIs aqueciam muito. O professor informou que os reguladores de tensão poderiam não estar funcionando de maneira correta. Desta forma, tivemos cuidado de abaixar as alimentações e ir subindo a voltagem aos poucos para não danificar o circuito, até acharmos um ponto em que o circuito se comportava de maneira estável. No geral, as alimentações ficaram em torno de +-4V e +-5V para **+-24VDC** e 9V para **+-12VDC** e **+-12VDC-IO**.

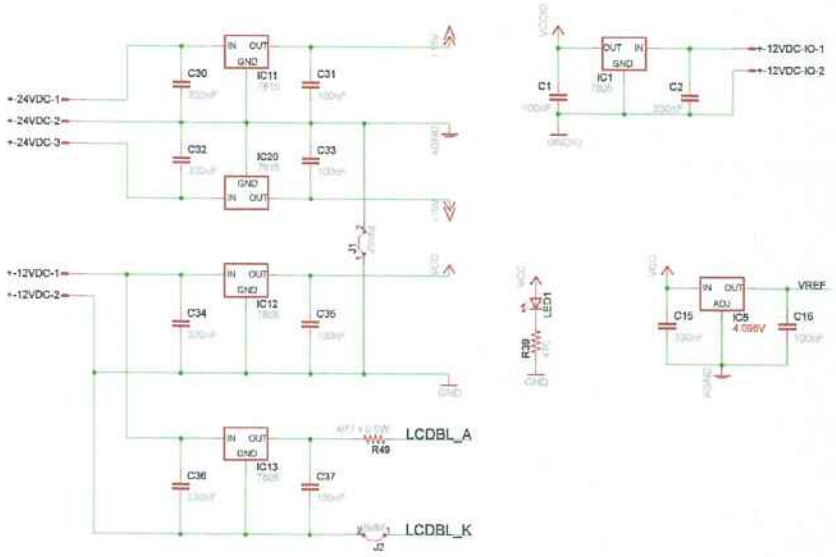

Figura 3-7 Módulo das Fontes

- **LEDs e Buzzer:** este módulo é de menor importância ao trabalho. Na placa temos um LED amarelo que está conectado aos pinos **LED-1** e **LED-2** e fica acesso sempre que o sistema está funcionando e alimentado, e os pinos **LED_0**, **LED_1**, **LED_2** e **BUZ** são pinos que podem ser controlados por saídas do PIC. Durante o desenvolvimento e testes do programa no PIC acoplado a placa, o *buzzer* chegou a ser usado, pelo pino **BUZ** ligado ao PIC, como confirmação de linhas que o PIC passava, resultados obtidos, entre outros. Ou seja, utilizamos *buzzer* como uma espécie de *debug* de placa, lógico que de maneira mais simplificada.

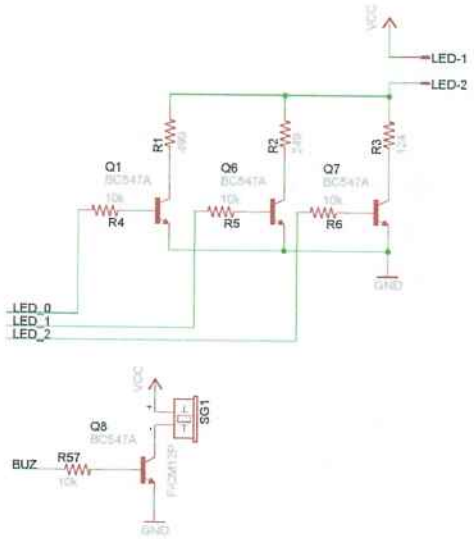

Figura 3-8 Nódulo com os Leds e Buzzer

- **Temperatura:** inicialmente este módulo era a opção para entrarmos com o sinal analógico lido do fotodiodo, acabamos durante o projeto decidindo por escolher o módulo do vídeo, que é bem semelhante a este e já havia sido usado para o mesmo fim. Este módulo é bem simples também, o pino **TEMP-1** é a entrada do sinal, que é regulado pelo AmpOp e em seguida repassado aos próximos módulos.

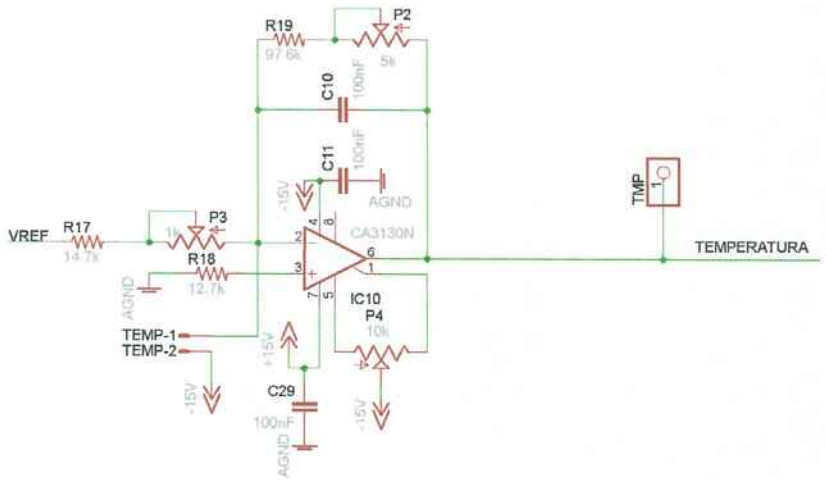

Figura 3-9 Módulo de Temperatura

- **Vídeo:** este módulo foi escolhido para conectarmos nosso sinal analógico vindo do fotodiodo. **ENT_VIDEO** é o pino de entrada, que passa pelo AmpOp e é repassado ao PIC com o nome de **VIDEO**. Este pino de vídeo já foi usado para entradas analógicas em outros projetos, vimos através de códigos que já foram usados com essa placa.

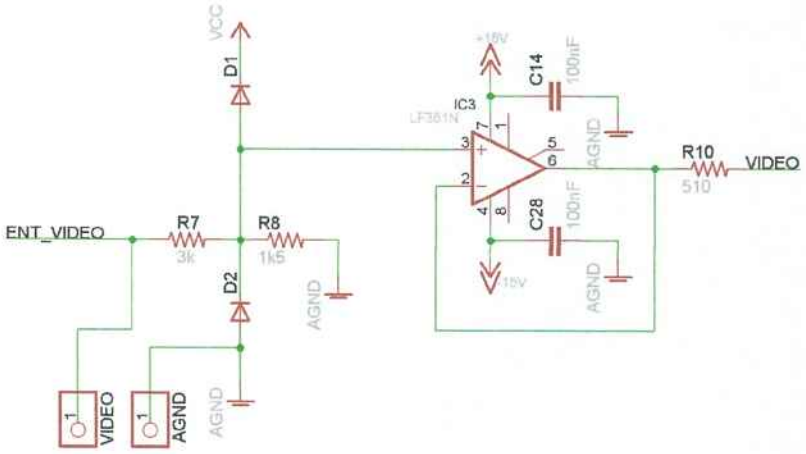

Figura 3-10 Módulo de Vídeo

- **Processamento:** este é, logicamente, o modulo mais importante para o trabalho, já que possui o microcontrolador PIC, as entradas analógicas, os pinos de saída para o display, portas seriais, leds's, buzzer, entre outros. Nas seções anteriores alguns dos pinos importantes para o trabalho já foram conhecidos, como o PIC concentra todas as entradas e saídas, estes pinos serão importantes aqui também. Além das entradas e saídas que já conhecemos como, **VIDEO**, **TX**, **RX**, **BUZ** e **LCDBL**, também utilizaremos o botão **RST**. As entradas e saídas no micrcontrolador, bem como suas funções serão descritas posteriormente nas próximas seções. Vale lembrar que os pinos **RB0** até **RB7** controlarão o display.

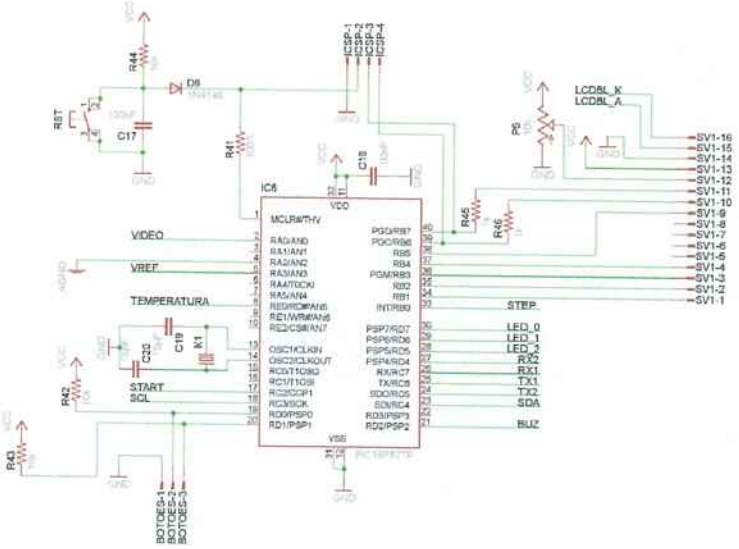

Figura 3-11 Módulo de Processamento

3.4. Caracterização do sistema fluídico com o conjunto de leitura (fotodiodo-led)

Aqui temos a integração de dois sistemas. O sistema fluídico que conta com os potes de água pura, água com colorante, um misturador mecânico ligado aos micro-tubos e uma micro bomba elétrica. Este sistema está ligado ao sistema de leitura, que possui placa e vai para descarte. A parte "física" mais importante do sistema está no líquido passando pelo caminho óptico, por que enquanto a solução passa ali, o led vermelho é projetado pela solução e a radiação resultante é recebida do outro lado do caminho óptico pelo fotodiodo. Explicando o fenômeno de maneira genérica, temos o seguinte, quando a solução de água pura passa pelo sistema, o fotodiodo transmite para o PIC um sinal (voltagem) praticamente constante, já quando a solução com uma mistura passa pelo sistema, temos uma modificação na

voltagem lida, conseqüentemente, os pontos lidos formam a curva que queremos obter no PIC, algo parecido com a curva abaixo:

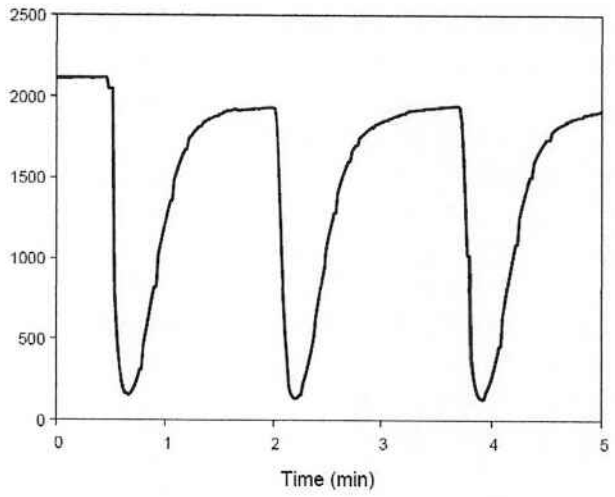

Figura 3-12 Exemplo da curva de leitura do fotodiodo

A seguir temos a descrição de cada parte do sistema fluídico e de leitura:

Figura 3-13 Potes de água pura e água com corante azul

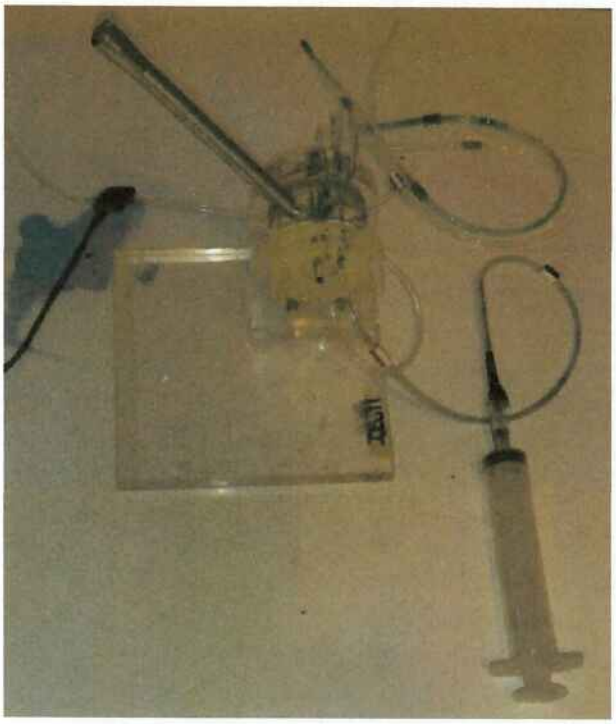

Figura 3-14 Sistema mecânico de mistura

Figura 3-15 Microbomba utilizada no sistema

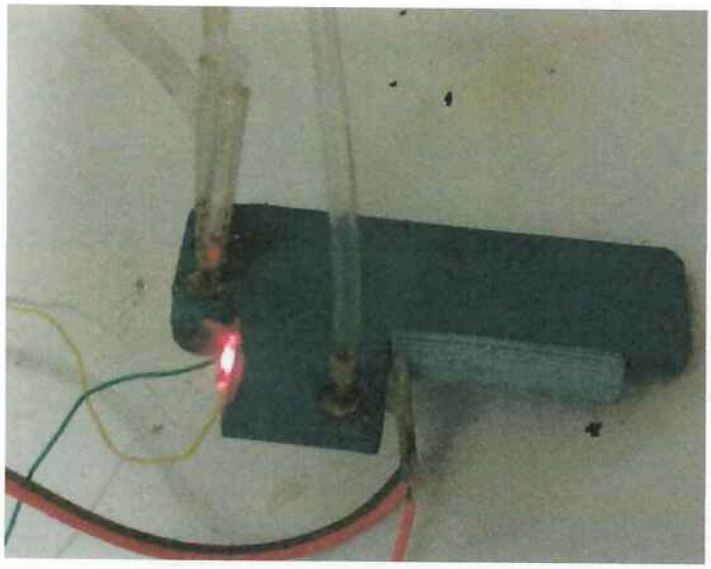

Figura 3-16 Placa de LTCC com sistema de leitura de dados

O diagrama abaixo ilustra de forma mas simplificada como funciona o sistema de aquisição dos dados através da placa de LTCC e o subseqüente envio do sinal para o microcontrolador PIC.

3.5. Incorporação do display de LCD ao ambiente do projeto

Como requisito do projeto, deveríamos usar o display de LCD para apresentarmos resultados e como ferramenta para acompanhar o funcionamento do sistema. Sendo assim, foi adquirido um display de LCD 16x2 para ser acoplado à placa de testes. A placa de testes já possuía um barramento de 16 pinos para se acoplar ao display. Num primeiro momento pensamos que o barramento já estivesse completamente preparado para se acoplar o display, mas analisando com calma, quais pinos e quais valores estávamos recebendo no barramento, foi necessário fazer-se uma pequena adaptação. A seguir temos parte do esquemático referente ao PIC e à conexão com o display de LCD e uma tabela com o pino no PIC e seu número correspondente, o nome dos pinos do barramento na placa e as conexões correspondentes no display, bem como suas funções.

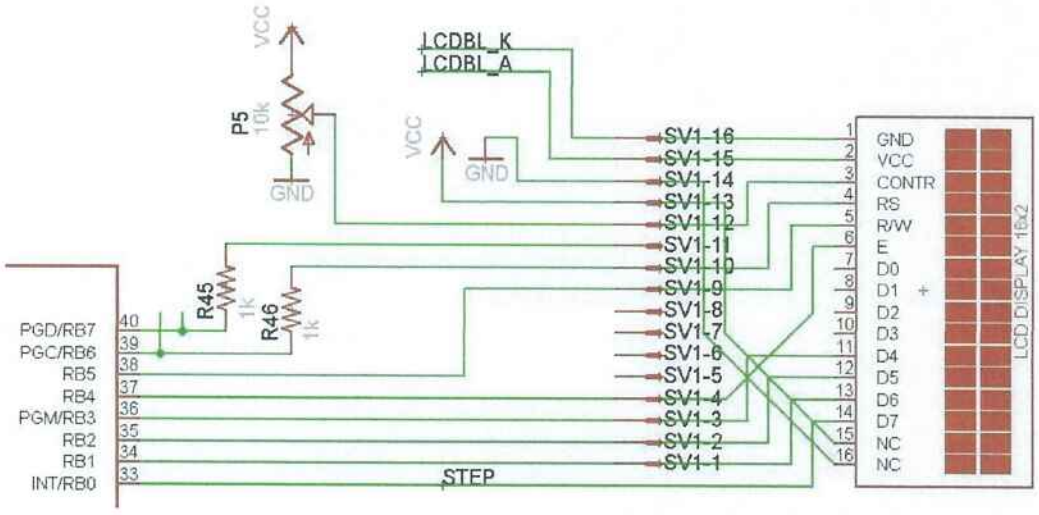

Figura 3-17 Montagem Display de LCD com placa de testes

Origem	# PIC	Nome na placa	Display	# Display	Função no display
LCDBL_K	-	SV1-16	GND	1	Pino de alimentação
LCDBL_A	-	SV1-15	VCC	2	Pino de alimentação
GND	-	SV1-12	VO	3	Controle de contraste do display
RB6	39	SV1-10	RS	4	Diferencia se palavra enviada é de comando ou caractere
RB5	38	SV1-9	R/W	5	Diferencia entre leitura e escrita no LCD
RB4	37	SV1-4	E	6	Informa ao LCD quando o dado está pronto para ser lido
-	-	-	D0	7	Usado para enviar palavra de configuração ou dados (caractere)
-	-	-	D1	8	Configuração e dados
-	-	-	D2	9	Configuração e dados
-	-	-	D3	10	Configuração e dados
RB3	36	SV1-3	D4	11	Configuração e dados
RB2	35	SV1-2	D5	12	Configuração e dados
RB1	34	SV1-1	D6	13	Configuração e dados
RB0	33	STEP	D7	14	Configuração e dados
VCC	-	SV1-13	LED+	15	Pino de alimentação *backlight*
GND	-	SV1-14	LED-	16	Pino de alimentação *backlight*

Tabela 1

A tabela acima resume praticamente todo o trabalho da incorporação do display de LCD ao sistema.

Algo que pode ser notado tanto pela tabela quanto pelo esquemático foi a utilização do pino de **STEP** como um pino de dados no display. Este pino tinha um uso completamente diferente em outros projetos, porém quisemos aproveitar todo o **PORTB** do PIC para utilizá-lo no display, desta forma aproveitamos uma peculiaridade do código fonte do PIC que define todos os pinos **RB**s como saída, mais sobre isso será explicado na próxima seção. Além disso, procuramos utilizar uma estrutura de software que já havia sido escrita para facilitar o controle do display pelo PIC, mexemos o mínimo possível nesta estrutura de código, isto será explicado melhor na próxima seção.

O primeiro passo após ligar as conexões do display de LCD na placa, é rodar um programa de teste no PIC, que seja simples, apenas para apresentar alguma mensagem no visor. Neste momento ao conectarmos o primeiro display, perdeu-se muito tempo pra tentar fazer o display funcionar, até descobriu-se que o display estava com problema. O problema foi "facilmente" solucionado com um display de LCD novo.

3.6. Transposição do código do PC para o microcontrolador PIC

Para o funcionamento do programa em C que escrevemos para PC, no PIC, precisamos fazer algumas mudanças importantes. A maioria delas são algumas

definições e configurações específicas para o perfeito funcionamento do PIC, e umas poucas mudanças estruturais, em decorrência da mudança de ambiente.

As principais definições estão no quadro abaixo:

```
#include <18f4620.h>
#device adc=10
#use delay(clock=20000000)
#fuses HS,NOWDT,NOPROTECT,PUT,NOBROWNOUT,NOLVP,NOMCLR
#use rs232(baud=57600, bits=8, xmit=PIN_C6,rcv=PIN_C7, stream=COM1)
#include <stdlib.h>
#use fixed_io(b_outputs=pin_b7, pin_b6,pin_b5,pin_b4,pin_b3,pin_b2,pin_b1,pin_b0)
#use fixed_io(c_outputs=pin_c7, pin_c6,pin_c5,pin_c4,pin_c3,pin_c2,pin_c1,pin_c0)
#use fixed_io(d_outputs=pin_d2)
#use fast_io(e)
#include <input.c>
#include <stdlib.h>
#include "mod_lcd.c"
```

Olhando o quadro acima, podemos observar os *includes* que temos agora para o funciomento no PIC, como o *header* do PIC 18F4620, que é o modelo de PIC que utilizamos no projeto. Temos alguns *includes* comuns ao código no PC, mas o último *include* (*mod_lcd.c*) é de grande importância para o controle e funcionamento do diaplay de LCD. Este conjunto de funções de controle do display de LCD já tinha sido preparado anteriormente por grupos do professor Seabra. No PC quando queremos "imprimir" um valor ou mensagem na tela, basta escrevermos *printf* e dentro dos parênteses colocarmos nossa mensagem. Para escrever algo no display, basicamente muda que ao utilizarmos uma função que adicionamos de *mod_lcd.c*, então antes de digitar a mensagem, precisamos escrever *lcd_escreve*.

Os *fuses* são configurações de bits bem específicos do PIC, que muitas o próprio compilador seta, nem precisa ser declarado no código-fonte. Além disso temos o *use rs232* que é a configuração para utilizarmos as portas seriais. Esta

configuração continuou no código, apesar de termos abandonado a idéia de utilizar estas portas durante o andamento projeto.

Por último temos as configurações dos pinos, com entrada ou saída. A diretiva *use fixed_io( )* que utilizamos para os **PORT**s **B**, **C** e **D** fixam estes determinados pinos como saída. Já a diretiva *use fast_io( )* necessita de um outro comando durante a execução do programa, por que essa definição permite a modificação de determinado pino para saída ou entrada durante a própria execução do código, essa definição é feita através do comando *set_tris_x( )*.

Outra importante modificação, esta estrutural no código, foi a adição da seguinte linha ao estado **REINICIAR**:

```
while(input(pin_e3));
```

Este trecho é o equivalente a dizer: "enquanto o botão **RST** não for pressionado, continua nesta linha". O *pin_e3*, é o primeiro pino do PIC e esta ligado diretamente ao botão **RST**, o primeiro estado do programa foi setado como o estado **REINCIAR**, e sempre após a execução de um ciclo de medidas, o programa volta para este estado. Portanto, o botão passou a ser utilizado como o *start* das nossas medidas.

3.7. Integração do sistema fluídico e leitura com o sistema de tratamento PIC

Com as etapas anteriores concluídas, esta se tornou muito trivial. Já que para fazermos a integração dos dois sistemas, bastou compartilhar alguma das

alimentações para que alimentássemos tanto o *led* quanto o fotodiodo, e ligássemos a saída com o sinal de leitura do fotodiodo no pino que convencionamos para a leitura analógica do PIC, que no caso é o pino **VIDEO**.

É importante relembrar que nas inicializações do PIC feitas no inicio do programa, temos que setar as portas corretas para que sejam feitas as leituras analógicas. No caso deste sistema basta se torna simples, uma vez que temos apenas uma entrada analógica.

3.8. Testes finais de funcionamento

O algoritmo para achar o ponto de inflexão em uma curva já estava testado e validado no PC, os pinos de saída comandavam o display de LCD de maneira correta, o fotodiodo conseguia "perceber" as variações de luz com uma modificação da solução, para os testes de funcionamento bastava que o sistema de leitura analógico-digital funcionasse. E funcionou, poucos ajustes foram necessários para fazer com que o sistema tivesse o funcionamento desejado. A leitura foi feita nos tempos desejados e obtivemos um valor esperado para o ponto de inflexão.

Na etapa de caracterização do conjunto fotodiodo-led a maior medição (tensão) que obtivemos do fotodiodo ao receber a luz do led, foi de aproximadamente 0,6 V. Esse não é o caso ideal, uma vez que o dispositivo LTCC que usamos foi algo feito provisoriamente, e era aberto. Desta forma não conseguíamos obter toda a luz do led, pois esta se dispersava, além disso, o fotodiodo poderia receber influências do meio externo. Esperávamos inicialmente uma leitura do fotodiodo de 2 V o que daria uma curva com uma gama maior de pontos durante o teste.

Dito isto, precisamos entender como o PIC realiza a transformação da tensão em sua entrada analógica para um valor inteiro. Como utilizamos o conversor A/D do PIC com 10 bits, as entradas analógicas são transformadas em inteiros que variam de 0 até 1023. Ou seja, para a entrada 0 V no PIC o conversor nos dará o número 0, já o valor máximo, 1023, será obtido quando tivermos na entrada do PIC um valor de tensão igual a VREF, uma tensão de referência que utilizamos como máxima, no capítulo com os módulos da placa de testes o VREF pode ser encontrado no módulo de processamento. Como VREF mantivemos a configuração da placa, que utiliza a tensão de uma das fontes, que no caso estava em 4 V.

Depois de conectado todo o sistema, medimos, com o auxílio do multímetro, tensões entre 0,5 e 0,6 V no fotodiodo. Aplicando o conceito dito acima, deduzimos que as leituras (em alto) do fotodiodo convertidas pelo A/D estejam em torno de 150, e foram justamente estes valores que obtemos durante testes na fase de integração dos sistemas.

Por último, testando o programa completo para encontrar o ponto de inflexão da curva obtida, obtemos diversas medidas que variaram entre 50 e 60, o que é um valor coerente com o que esperamos, já que representa cerca de 40% do valor máximo da curva.

4. CONCLUSÕES

Nesta etapa do projeto, é importante observar 3 pontos. De onde partimos, onde queríamos chegar e aonde chegamos. Dessa maneira, podemos visualizar de forma mais racional o que poderia e o que foi feito.

O ponto de partida para este projeto tinha uma base relativamente sólida, era um terreno preparado, podemos dizer assim, mas em contrapartida, era um terreno pouco explorado. Isso significa dizer que tínhamos uma boa base bibliográfica cedida pela doutoranda Zaira, que ajudou a elucidar o objetivo do projeto, mas fora esta bibliografia não tínhamos muitas outras opções de pesquisa, o que se torna um obstáculo, a partir do ponto em que nos distanciamos um pouco do tema central de pesquisa da Zaira. Esta dificuldade acabou sendo suprida por algumas conversas com o professor Seabra e a própria Zaira, que pouco a pouco esclareciam alguns pontos de dúvida e ajudaram numa visão mais clara da solução dos problemas.

O objetivo inicial de Projeto de Formatura I parece algo muito distante, tanto em tempo quanto em resultados alcançados. Inicialmente, colocamos como objetivo criar uma plaquinha em LTCC semelhante a da doutoranda Zaira para que fizéssemos os processamentos necessários para acharmos a concentração de solução e que mostrássemos tanto a curva quanto os resultados numa interface de um software.

Olhando para este objetivo inicial e para onde chegamos, realmente vemos uma distância muito grande. Talvez o objetivo inicial era pretensioso demais para um

projeto de formatura, quem sabe poderia ser continuado em um mestrado, ou talvez todas as tarefas dificilmente poderiam ser realizadas por apenas um aluno.

Alguns pontos acabaram sendo modificados pelo próprio professor, que ao longo da caminhada percebeu o que poderia e o que não poderia ser feito. Ou o que facilitaria também na realização do projeto, como a adicionar o display de LCD ao invés de utilizarmos uma conexão serial com um PC. Não conseguimos obter um chip do microcontrolador PIC que escolhemos num encapsulamento SMD, isso também nos limitou um pouco também.

Fato é que, observando-se o objetivo inicial e o resultado final, temos no resultado a essência do objetivo proposto. Ou seja, o principal do trabalho, que é a medição da concentração de uma solução através de técnicas espectrofotométricas, foi feito.

Portanto creio que ainda que tudo o que foi proposto não tenha sido realizado, o que foi realizado pode ser de grande importância para a continuação e desenvolvimento de sistemas semelhantes ao que criamos.

De forma pessoal também, este projeto pôde desenvolver diversas áreas que durante todo o percurso na Escola Politécnica não tinham sido tocados. Foi um projeto muito abrangente com diversos ramos de pesquisa e atuação, o que com certeza cria dificuldades na realização e seu término causa um grato sentimento de realização.

5. REFERÊNCIAS BIBLIOGRÁFICAS

1 Slideshare: <*http://www.slideshare.net/b.cortez/aula-05-espectrofotometria-uv-vis*> **acesso em 20 de julho de 2009.**

2 Wikipedia: <*http://pt.wikipedia.org/wiki/Espectrofotometria*> **acesso em 20 de julho de 2009.**

3 Wikipedia: <*http://pt.wikipedia.org/wiki/Espectroscopia_UV/vis%C3%ADvel*> **acesso em 19 de julho de 2009.**

4 Mendes da Rocha, Zaira. *Espectrofotometria em LTCC.* **Relatório de doutorado, 2007**

5 HANSEN, E.h. *Exploiting Kinetic-Based Flow Injetction Methods for Quantitative Chemical Assays.* **ANAL. CHIM. ACTA, 261 (1992) 125.**

5 HANSEN, E.H. *FIAlab Instruments <http://flowinjection.com/>* **acesso em 15 de julho de 2009.**

6 GUALHARDO C.X., MASINI J.C. *spectrophotometric determinaition of phosphate and silicate by sequential injection using molybdenum blue chemistry.* **ANALYTICA CHIMICA ACTA 417 (2000) 191-200.**

7 Microcontroladores PIC: <http://www2.brazcubas.br/professores1/arquivos/20_franklin/T7037A/Microcontroladores_Pic_-_Apostila.pdf> **acesso em 10 de setembro de 2009.**

7 Uso de Display de LCD: <http://www.ee.pucrs.br/~terroso/html/lcd.html> **acesso em 25 de outubro de 2009.**

6. ANEXOS

Código Fonte implementado no PIC:

```
#include <18f4620.h>
#device adc=10
#use delay(clock=20000000)
#fuses HS,NOWDT,NOPROTECT,PUT,NOBROWNOUT,NOLVP,NOMCLR
#use fast_io(B)
#use rs232(baud=57600, bits=8,  xmit=PIN_C6,rcv=PIN_C7, stream=COM1)
#include <input.c>
#define hi(x)  (*(&x+1))
#include <stdlib.h>
#use fixed_io(b_outputs=pin_b7, pin_b6,pin_b5,pin_b4,pin_b3,pin_b2,pin_b1,pin_b0)
#use fixed_io(c_outputs=pin_c7, pin_c6,pin_c5,pin_c4,pin_c3,pin_c2,pin_c1,pin_c0)
#use fixed_io(d_outputs=pin_d2)
#use fast_io(e)
#zero_ram
#bit t1_overflow=0x0C.0
#define  STRING_SIZE    100          // The number of characters allowed to input
#include <stdlib.h>
#include "mod_lcd.c"


//defines do programa
#define ARRAY_MAX 1875
#define DIODE_MIN 30 //>0
#define DIODE_MAX 510 //<512
#define INCLINACAO_MIN -200//<0; para detectar regiao de interesse;
#define MAX_PONTOS 64//pontos de interpolação
```

```
#define VIDEO_MAX 1020 /

#define MAX_CICLOS 15 //< 64, para ADC=10bits; CICLOS COMPLETOS colhidos do sinal de video


//estados da maquina de coleta de sinal

#define ARMAR   0

#define LER     1

#define ENCERRAR 2


//estados da maquina principal

#define INICIAR  0

#define NORMAL   1

#define CALCULAR 2

#define REINICIAR 3


//defines da placa

#define AJUSTAR_RELOGIO     pin_d0 //botao externo1 para ajuste hora

#define TAMPA_ABERTA        pin_d1 //botao externo2 q identifica tampa aberta

#define START    pin_C2 //sinal start (ENTRADA)

#define LED_0    pin_D7 //para acionamento do LED (SAIDAS)

#define LED_1    pin_D6

#define LED_2    pin_D5

#define BUZ      pin_D2 //para acionamento do SINAL SONORO (BUZZER)

#define CANAL_DADOS 5 //canal AD para sinal analógico de video


//VARIÁVEIS

//globais

BYTE buffer; //armazena byte recebido na interrupcao serial


static int16 i16passo;
```

```
static int8 i8estadoleitura;
static int8 i8ciclovideo;
static int16 i16pulsovideo, i16videomin, i16videomax;
static boolean bfimleitura;

static int16 i16vetorVideo[ARRAY_MAX];

static signed int16 si16primeiraDerPosteriorPosteriorPosteriorPosterior;
static signed int16 si16primeiraDerPosteriorPosteriorPosterior;
static signed int16 si16primeiraDerPosteriorPosterior;
static signed int16 si16primeiraDerPosterior;
static signed int16 si16primeiraDer;
static signed int16 si16primeiraDerAnterior;
static signed int16 si16primeiraDerAnteriorAnterior;
static signed int16 si16primeiraDerAnteriorAnteriorAnterior;
static signed int16 si16primeiraDerAnteriorAnteriorAnteriorAnterior;

static signed int16 si16segundaDerPosteriorPosteriorPosterior;
static signed int16 si16segundaDerPosteriorPosterior;
static signed int16 si16segundaDerPosterior;
static signed int16 si16segundaDer;
static signed int16 si16segundaDerC;
static signed int16 si16segundaDerAnterior;
static signed int16 si16segundaDerAnteriorC;
static signed int16 si16segundaDerAnteriorAnterior;
static signed int16 si16segundaDerAnteriorAnteriorAnterior;

static int8 i8rotina;
```

```
static int16 i16idx   = 0;
static float32 f32porc=0;
static boolean bAchouPtoInflx = FALSE;
static int16 iInflxLimInf=0;
static int16 iInflxLimSup=0;
static int16 i16idxDer;

// tratamento de interrupcoes
//prioridade, da maior para menor
#priority EXT,RDA

#int_default
void default_isr(){}//tratamento padrão de interrupcao não identificada

//#int_EXT
//tratamento de interrupcao externa via pino RB0
//tratamento de interrupcao externa
//coleta e análise de sinal de video, ciclo a ciclo

//void ext_isr()
void le_entrada()
{
  i16passo++;
  switch(i8estadoleitura)
  {
    case ARMAR:
    {
      if(TRUE)
      {
```

```
    i16passo = 0;
    i8estadoleitura = LER;
  }
};
break;
case LER:
{
//coleta video
//normal 10us
delay_us(100); //para centrar a leitura no meio de cada pulso de video
i16pulsovideo = read_adc(); //faz leitura
i16vetorVideo[i16idx*(MAX_CICLOS+1) + i16passo] = i16pulsovideo; //armazena
//acha video max e video min
if(i16pulsovideo < i16videomin)i16videomin=i16pulsovideo;
if(i16pulsovideo > i16videomax)i16videomax=i16pulsovideo;

if(i16passo >= MAX_CICLOS) i8estadoleitura = ENCERRAR;//fim da coleta
};
break;

case ENCERRAR:
{
bfimleitura = TRUE;
};
break;//não faz nada
default:
{
  i8estadoleitura = ENCERRAR;
};
break;
```

```
  }//switch
}//ext_isr

#int_RDA
//interrupcao para recepcao serial de COM1 SOMENTE!
void serial_isr() {
 . buffer = fgetc(COM1);
}

//FUNCOES
//funcoes gerais
//funcao para recepcao de um byte por comunicacao serial
#separate
BYTE bgetc(){
  BYTE resposta;
  resposta = buffer;
  buffer = '0';
  return resposta;
}

void media_dados(int16 idx){
  int16 i16soma = 0;
  int16 i;
  //Calcula Média do vetor de dados acumulado
  for(i=idx ; i <= idx + MAX_CICLOS ; i++)  i16soma = i16soma + i16vetorVideo[i];

  //CALCULA A MÉDIA DOS ÚLTIMOS 16 PONTOS E ARMAZENA NO PRIMEIRO IDX
DAQUELE CICLO
  i16vetorvideo[idx] = i16soma/(MAX_CICLOS+1);
}
```

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

//////////////////////

void calcula_derivadas(int16 i){

  si16primeiraDerPosteriorPosteriorPosteriorPosterior = (int16)(((float)i16vetorVideo[i+5] -
(float)i16vetorVideo[i+3])/2);

  si16primeiraDerPosteriorPosteriorPosterior      = (int16)(((float)i16vetorVideo[i+4] -
(float)i16vetorVideo[i+2])/2);

  si16primeiraDerPosteriorPosterior               = (int16)(((float)i16vetorVideo[i+3] -
(float)i16vetorVideo[i+1])/2);

  si16primeiraDerPosterior                        = (int16)(((float)i16vetorVideo[i+2] -
(float)i16vetorVideo[i])/2);

  si16primeiraDer                                 = (int16)(((float)i16vetorVideo[i+1] -
(float)i16vetorVideo[i-1])/2);

  si16primeiraDerAnterior                         = (int16)(((float)i16vetorVideo[i]   -
(float)i16vetorVideo[i-2])/2);

  si16primeiraDerAnteriorAnterior                 = (int16)(((float)i16vetorVideo[i-1] -
(float)i16vetorVideo[i-3])/2);

  si16primeiraDerAnteriorAnteriorAnterior         = (int16)(((float)i16vetorVideo[i-2] -
(float)i16vetorVideo[i-4])/2);

  si16primeiraDerAnteriorAnteriorAnteriorAnterior = (int16)(((float)i16vetorVideo[i-3] -
(float)i16vetorVideo[i-5])/2);


  si16segundaDerAnteriorC = si16segundaDerC;

  si16segundaDerC         = si16segundaDerPosterior;


  si16segundaDerPosteriorPosteriorPosterior = (si16primeiraDerPosteriorPosteriorPosteriorPosterior
- si16primeiraDerPosteriorPosterior)/2;

  si16segundaDerPosteriorPosterior      = (si16primeiraDerPosteriorPosteriorPosterior      -
si16primeiraDerPosterior)/2;

  si16segundaDerPosterior               = (si16primeiraDerPosteriorPosterior               -
si16primeiraDer)/2;

  si16segundaDer                        = (si16primeiraDerPosterior                        -
si16primeiraDerAnterior)/2;
```

```
   si16segundaDerAnterior              = (si16primeiraDer                    -
si16primeiraDerAnteriorAnterior)/2;

   si16segundaDerAnteriorAnterior      = (si16primeiraDerAnterior            -
si16primeiraDerAnteriorAnteriorAnterior)/2;

   si16segundaDerAnteriorAnteriorAnterior = (si16primeiraDerAnteriorAnterior -
si16primeiraDerAnteriorAnteriorAnteriorAnterior)/2;


   si16segundaDerPosterior = (int16)(((float)si16segundaDerAnterior + (float)si16segundaDer +
(float)si16segundaDerPosterior

                  + (float)si16segundaDerPosteriorPosterior +
(float)si16segundaDerPosteriorPosteriorPosterior)/5);

   si16segundaDerAnterior = si16segundaDerAnteriorC;

   si16segundaDer      = si16segundaDerC;

}

//////////////////////


//////////////////////
//Roda varredura do sinal de video e armazena leituras
//zera variáveis antes de coletar
void ler_dados(void)
{
   //inicia variaveis
   //variavel de controle
   i8ciclovideo = 0;
   bfimleitura = FALSE;
   //variaveis de dados
   i16videomin = 1024;
   i16videomax = 0;

   set_adc_channel(CANAL_DADOS);//canal de leitura do sinal do fotodiodo
   i8estadoleitura = ARMAR;
```

```
  while(!bfimleitura)
  {
    le_entrada();
  };//ao final tem-se as contagens

  disable_interrupts(int_ext);
}//fim ler_dados

//////////////////////
//le video
//compara video max
//incrementa led
//////////////////////

//////////////////////
//////////////////////
void inicia_refra(void){

  int16 i16cont;
  for(i16cont=0; i16cont<ARRAY_MAX; i16cont++)
    i16vetorVideo[i16cont] = 0;

  //configura as portas analogicas para conversao AD
//  setup_ADC_ports (AN0_AN1_AN4_AN5_AN6_AN7_VREF_VREF);//para 18F452
  setup_ADC_ports (AN0_TO_AN5|VREF_VREF );//para 18F4620
  //configura o clock da conversao AD
  setup_adc(ADC_CLOCK_DIV_2);
  //setup_adc( ADC_CLOCK_INTERNAL );

  // serial, inicia com '0'
```

```
  buffer = '0';

  //habilita interrupcao
  enable_interrupts(global);
  //habilita interrupcao de recepcao serial
  enable_interrupts(int_rda);
  ext_int_edge( L_TO_H ); //int EXT ocorrendo na borda de subida do sinal "step" .
  //if(saida_estendida == 1)fprintf(COM2,"\r\n *  Pronto\n");
  //tocaBip();
}

void main()
{
  lcd_ini();
  delay_us(100);
  set_tris_e(0xff);
  inicia_refra();

  i8rotina = REINICIAR;
  i16idx   = 0;
  bAchouPtoInflx = FALSE;
  iInflxLimInf=0;
  iInflxLimSup=0;

  while(TRUE)
  {
     switch(i8rotina)
     {
        case NORMAL:
        {
```

```
        i16idx  = 0;
        printf(lcd_escreve, "\f Lendo");
        printf(lcd_escreve, "\n Entrada...");
        delay_ms(1000);
      while(i16idx < ARRAY_MAX)
        {
        ler_dados();
        media_dados(i16idx);
        delay_ms(15);
        i16idx++;
        }
        i8rotina = CALCULAR;
      };
      break;

      case INICIAR:
      {
        printf(lcd_escreve, "\f Iniciando");
        printf(lcd_escreve, "\n Dispositivo...");
        inicia_refra();
      delay_ms(1000);
        i8rotina = NORMAL;//inicia

      };
    break;
      case CALCULAR:
      {
        //pela analise no matlab par 30000 o pto de inflexao deve estar entre 7500
        // para 1875 proximo de 468.75
        //segundo ponto em 22500
```

```
  //proporcionalmente em 1406,25
  int iInflxLimInf=0;
  int iInflxLimSup=0;

  for ( i16idxDer = 5; !bAchouPtoInflx && i16idxDer <= ARRAY_MAX-5; i16idxDer++)
  {
  //printf("\n***passouCALIBRAR %d ***\n", bAchouPtoInflx);
  f32porc = (i16idxDer*100)/1875;
  printf(lcd_escreve, "\f Calculando %f ", f32porc);
  printf(lcd_escreve, "\n derivadas parciais");
  calcula_derivadas(i16idxDer);

  if(!iInflxLimInf && !iInflxLimSup && si16segundaDerPosterior == 0)
  {
    iInflxLimInf = i16idxDer;
    //printf("\n\n ponto de inflexao inicial em %d\n\n",i16idxDer);
  }

  if(iInflxLimInf && !iInflxLimSup && si16segundaDerPosterior !=0)
  {
    iInflxLimSup = i16idxDer;
    bAchouPtoInflx = FALSE;
    //printf("\n\n ponto de inflexao final em %d\n\n",i16idxDer);
    bAchouPtoInflx = TRUE;
  }
  if(bAchouPtoInflx)
  {
    printf(lcd_escreve, "\f Valor no ponto");
    printf(lcd_escreve, "\n --- %d ---",iInflxLimInf + (iInflxLimSup-iInflxLimInf)/2);
    delay_ms(10000);
```

```
            }
            else if (i16idxDer == ARRAY_MAX-5)
            {
              printf(lcd_escreve, "\f Valor no ponto");
              printf(lcd_escreve, "\nNnao achou valor");
              delay_ms(1000);
            }
            }
             i16idx = 0;
             bAchouPtoInflx = FALSE;
             i8rotina = REINICIAR;
          };
          break;
          case REINICIAR:
          default:
          {
          printf(lcd_escreve, "\f Aperte o botao");
            printf(lcd_escreve, "\n para iniciar.");
          while(input(pin_e3));
          fprintf(COM1, "testando enviando dados para pc");
          delay_ms(500);
            //printf("\n MAQUINA DE ESTADO default\n");
            i16passo=0;
            i8rotina = INICIAR;//inicia
            //system("PAUSE");
          };
          break;
        }//NORMAL
      }// while true*/
}
```