FREDERICO LEONI FRANCO KAWANO
ROGÉRIO EDUARDO SILVA SANTANA

SISTEMA PARA LOCALIZAÇÃO ACÚSTICA

Dissertação apresentada à Escola Politécnica da Universidade de São Paulo para a obtenção de graduação em Engenharia Mecânica.

Área de Concentração:
Engenharia Mecatrônica

Orientador:
Professor Doutor Celso Massatoshi Furukawa

SÃO PAULO
2001

Agradecimentos

À Escola Politécnica da Universidade de São Paulo, que durante cinco anos nos acolheu e preperou para nossa futura vida profissional, e também por ter cedido equipamentos para que este trabalho pudesse ser concluído.

Ao Professor Celso Furukawa, que nos orientou durante todo este ano de trabalho.

Ao aluno de mestrado Frederico Vines Faria de Lima, por todo apoio, colaboração e por sua presença nos momentos difíceis deste trabalho, que não foram poucos.

Ao nosso amigo Antônio Fernando Maiorquim por ceder sua sala, equipamentos e conhecimento para que o nosso protótipo pudesse ser construído e funcionasse.

Aos nossos companheiros e amigos de classe, que nos apoiaram, incentivaram e principalmente, reconheceram nosso trabalho, principalmente aos amigos, Celso Ramos de Sousa, Daniel Olioni Anderson, Eduardo Paissanot, Leonardo Lopes Carnellos e Rodrigo de Deus Reinaldo, que nos emprestaram alguns equipamentos e valorizaram o funcionamento do nosso protótipo.

Por fim, às nossas famílias que nunca deixaram de nos incentivar e apoiar e a Deus que nos deu saúde para poder terminar este longo caminho.

Índice

1. Introdução

O projeto consiste na implementação de um sistema de medição de distância, de um veículo móvel em relação a objeto fixo (chamado de alvo), ambos imersos no mar. Esta medição é feita através da cronometragem do tempo entre a emissão de um sinal acústico pelo veículo móvel e a recepção pelo mesmo de um sinal acústico de resposta emitido pelo alvo. Assim, conhecida a velocidade do som no mar, pode-se calcular a distância entre o veículo móvel e o alvo.

Este sistema a ser implementado tem por finalidade ser um sensor para realizar o controle de posicionamento de um veículo submarino autônomo. Para que se possa realizar este controle de posicionamento, é necessário a existência de três sensores de distância localizados em pontos distintos do veículo móvel. Assim, com a leitura dos três sensores o controlador determina a posição exata do alvo em relação ao veículo móvel.

O esquema geral do projeto está na figura a seguir:

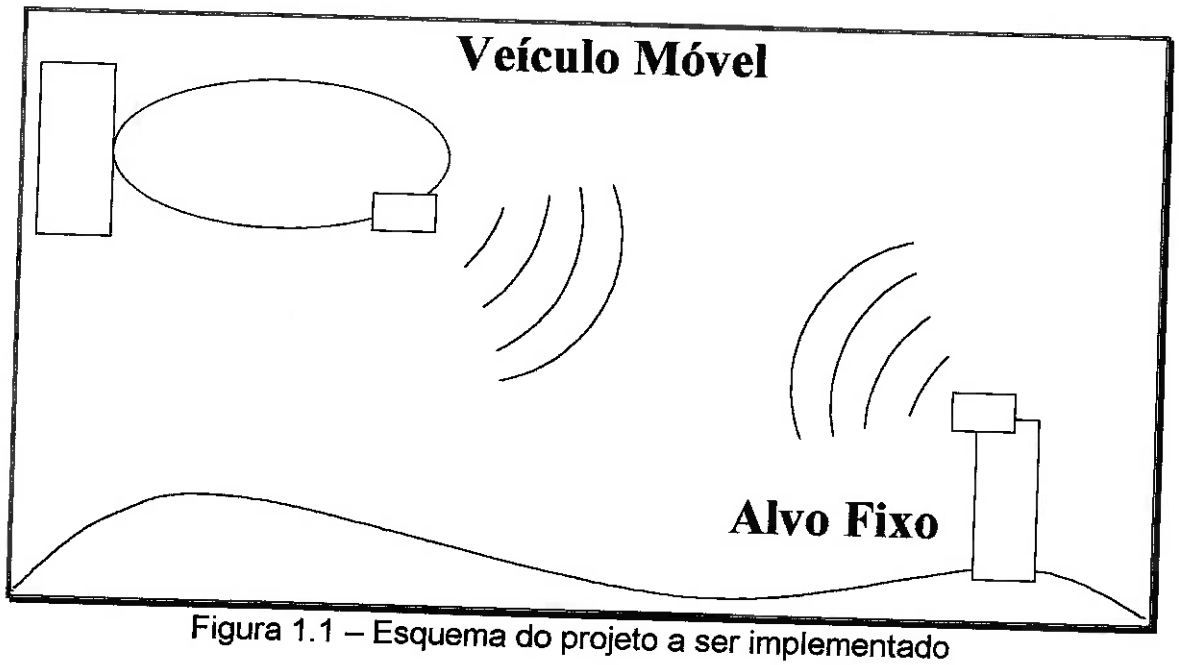


Figura 1.1 – Esquema do projeto a ser implementado

2. Sistema de Medição

O projeto consiste em cronometrar o tempo entre a emissão de uma onda pelo veículo móvel e a recepção por este mesmo dispositivo de uma resposta do veículo fixo, que por sua vez só emite esta resposta assim que excitado pela emissão do veículo móvel. Assim sendo, para que se possa realizar uma boa medição, sem interferência de eventuais ecos, que possam provocar cronometragens equivocadas, cada unidade emite sinais em freqüências distintas, sendo 150 kHz para o veículo móvel e 170 kHz para o alvo fixo.

Para realizar a emissão e recepção de ondas acústicas serão utilizados transdutores piezelétricos, que são capazes de converter uma excitação elétrica em uma onda ou uma excitação mecânica (produzida por uma onda) em diferença de potencial elétrico nos terminais do transdutor.

A fim de produzir a excitação elétrica necessária para a emissão da onda pelo transdutor será implementado um circuito elétrico, bem como um circuito para receber a tensão potencial produzida pelo transdutor. O esquema destes circuitos está ilustrado na figura 2.1.

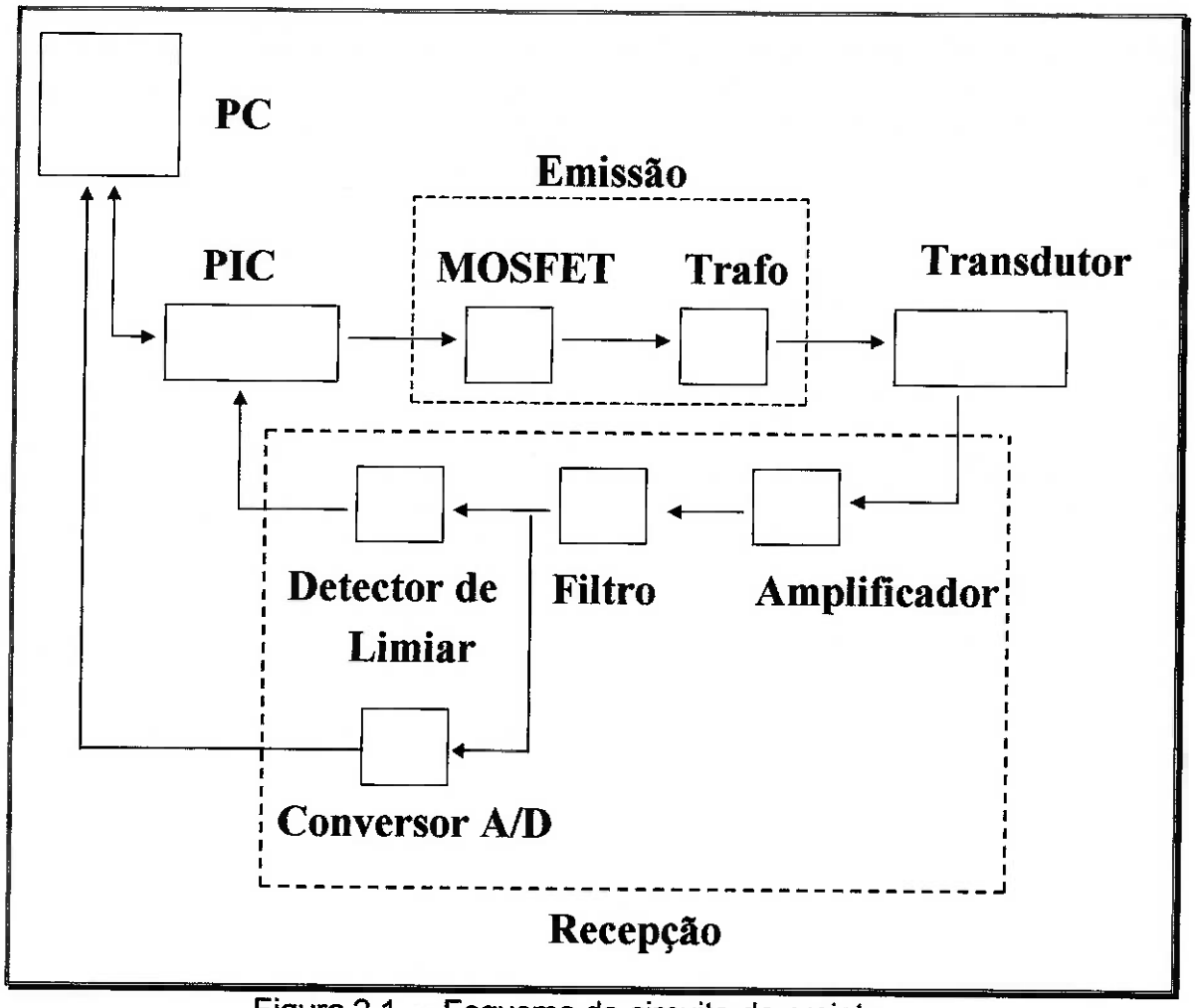


Figura 2.1. – Esquema do circuito do projeto

O circuito de emissão, consiste em uma fonte de tensão, um gerador de pulsos, um dispositivo capaz de chavear a fonte de tensão a partir dos pulsos emitidos pelo gerador e finalmente um transformador capaz de adequar a tensão fornecida com a tensão admissível no transdutor. Convém explicar que cada circuito de emissão, tanto do veículo móvel como o veículo fixo, possuem seus respectivos geradores de pulsos que possibilitam que estes emitam sinais em freqüências distintas.

Já o circuito de recepção consiste num amplificador para que o sinal (tensão) produzido pelo transdutor seja amplificado, um filtro para se eliminar o ruído de alta freqüência e, finalmente, para tratar o sinal, o circuito pode realizá-lo de duas formas: uma através de um dispositivo analógico, que nada mais é que um detector de limiar, cuja função é informar a chegada do sinal de

resposta. E a outra maneira, através de um conversor A/D, que digitalizará a onda de saída do filtro, para que esta seja analisada por software.

No veículo móvel será utilizado um microcontrolador PIC, que no caso de uma recepção analógica tem a função de produzir os pulsos, cronometrar o tempo entre a emissão do mesmo e a recepção da resposta do veículo fixo, diferenciando esta resposta de ecos da sua própria emissão, e posterior realização do cálculo da distância e envio deste valor para o PC. Já para o caso de tratamento de sinal digital, será utilizado um mesmo controlador PIC, porém com a função de gerar os pulsos para a emissão, habilitar a leitura do sinal digitalizado pela placa A/D e manter a comunicação entre este veículo e um PC que seria o seu controlador central.

No alvo, também será utilizado um microcontrolador PIC a fim de emitir uma onda de resposta, segundo uma freqüência desejada (170 KHz), a partir do recebimento da onda emitida pelo veículo móvel, e como no sistema analógico do veículo móvel, diferenciando este sinal esperado de eventuais ecos que possam ser detectados. Convém mencionar que o circuito do alvo fixo é idêntico ao circuito da unidade móvel com tratamento analógico, porém sem possuir uma comunicação com o PC.

3. Estudo de Acústica

Para conhecer o comportamento de ondas acústicas no mar, foi realizado um estudo sobre acústica, baseado na referência [2].

Para calcular a distância entre o veículo móvel e o alvo, cronometrando o tempo entre a missão e a recepção de um onda, foi admitido que a onda possui uma velocidade constante durante todo o percurso. Através do estudo realizado, verificou-se que a velocidade da onda no mar depende apenas da pressão, temperatura e salinidade do meio. Considerando que a área de atuação do veículo móvel é pequena, de modo que a variação destes fatores pode ser desconsiderada, assumi-se que a velocidade da onda é constante em todo o percurso da mesma, o que valida o método de calculo da distância empregado.

A existência de perdas na intensidade acústica durante sua propagação no mar, foi outro fator estudado, afim de estimar a intensidade acústica da onda emitida necessária para que o transdutor de recepção possa detectá-la. Verificou-se então que o coeficiente de absorção de intensidade acústica por unidade de distância percorrida depende das condições do meio e da freqüência da onda, sendo que quanto maior a freqüência da onda, maior será o coeficiente de absorção.

4. Estudo de Transdutores

Os estudos para este capítulo foram baseados na referência [1].

4.1. Materiais Piezelétricos

Materiais Piezelétricos são aqueles que tem a propriedade de se deformarem a partir de um campo elétrico aplicado, mudando assim suas dimensões; e de produzirem um diferença de potencial a partir de uma deformação provocada.

A variação do momento do dipolo por unidade (P) de volume é:

$$\Delta P = P \cdot S = e \cdot S \qquad (1)$$

Onde, 'e' é chamado de constante de tensão piezelétrica e S é a deformação do material do piezelétrico na direção do campo elétrico, que foi adotada em z.

O deslocamento elétrico por unidade de área, devido ao campo elétrico aplicado, é dado pela seguinte expressão:

$$D = \varepsilon^S \cdot E + \Delta P \qquad (2)$$

onde ε^S é a constante dielétrica para deformação nula ou constante e E é o campo elétrico aplicado.

A tensão interna resultante, devido ao campo elétrico é dada por:

$$T_E = e \cdot E \qquad (3)$$

A tensão total é dada por:

$$T + T_E \qquad (4)$$

onde T é a tensão aplicada externamente ao material piezelétrico. Pela lei de Hooke:

$$T + T_E = c^E \cdot S \qquad (5)$$

onde c^E é a constante elástica na presença de um campo elétrico constante (E).

Os principais materiais piezelétricos são os material ferroelétricos, como as cerâmicas PZT e alguns cristais. Alguns materiais não ferroelétricos, como o quartzo, podem ser piezelétricos, porém, possuem uma constante de tensão piezelétrica (e) baixa, em relação aos ferroelétricos.

Considerando um deslocamento elétrico nulo, pode-se obter uma nova constante de elasticidade (c^D) definida por:

$$c^D = c^E\left(1 + K^2\right) \qquad (6)$$

onde

$$K^2 = \frac{e^2}{c^E \cdot \varepsilon^S} \qquad (7)$$

E a tensão externa para D = 0, fica:

$$T = c^E\left(1 + \frac{e^2}{c^E \cdot \varepsilon^S}\right) \cdot S = c^D \cdot S \qquad (8)$$

Considerando que a tensão externa (T) for nula obtém-se a constante dielétrica $\left(\varepsilon^T\right)$, que vale:

$$\varepsilon^T = \varepsilon^S\left(1 + K^2\right) \qquad (9)$$

e

$$D = \varepsilon^T \cdot E \qquad (10)$$

4.2. Transdutores

Os transdutores piezelétricos consistem num dispositivo que converte energia mecânica em elétrica, e vice-versa. Como visto anteriormente, o material piezelétrico se deforma ao aplicarmos um campo elétrico ao mesmo, assim utilizando uma fonte de tensão variável, o transdutor produziria uma onda mecânica. O oposto também é válido, produzindo uma diferença de potencial variável entre os terminais do transdutor.

Para que se melhore a potência transmitida do transdutor para o fluido costuma-se utilizar uma camada de "matching", cuja a impedância acústica está entre a do material piezelétrico e a do meio, de modo que se atenue a diferença acústica entre estes materiais.

Para que se atenue a ressonância do material piezelétrico costuma-se utilizar uma camada de material de grande perda acústica na face do material piezelétrico que não está em contato com o meio fluido.

Considerando um transdutor uniforme com seção transversal de muitos comprimentos de onda e os eletrodos posicionados nas faces opostas do material piezelétrico e normais à direção z (figura 4.1), só haverá campo elétrico na direção z e ondas longitudinais nesta mesma direção.

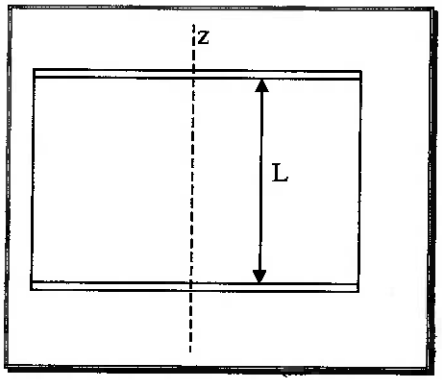


Figura 4.1. – Material piezelétrico e eletrodos

Se mostrar-se o transdutor como sendo uma caixa preta de uma porta elétrica e duas portas mecânicas, definido a força aplicada na superfície do transdutor e a voltagem do circuito elétrico, assim como a velocidade de partícula nesta mesma superfície e a corrente no mesmo circuito (figura 4.2).

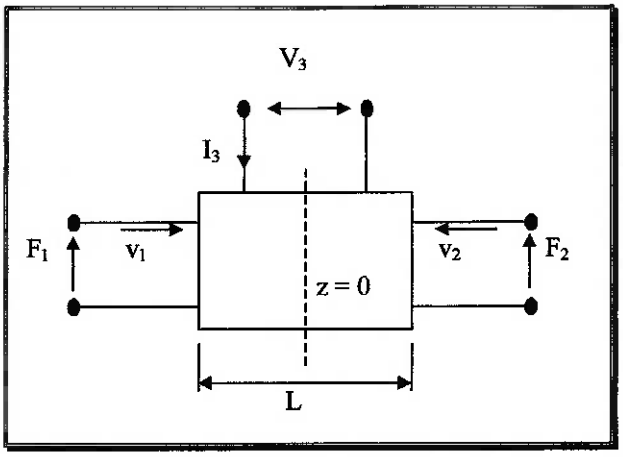


Figura 4.2. – Esquema do transdutor como caixa preta

Assim, as condição de contorno para as forças e as velocidades mostradas acima:

$$\begin{aligned} F_1 &= -AT\left(\frac{-L}{2}\right) \\ F_2 &= -AT\left(\frac{L}{2}\right) \\ v_1 &= v\left(\frac{-L}{2}\right) \\ v_2 &= -v\left(\frac{L}{2}\right) \end{aligned} \qquad (11)$$

Considerando um volume infinitesimal, as grandezas variam em z da seguinte forma:

$$\frac{dT}{dz} = \rho_{m0} \cdot v$$
$$\frac{dv}{dz} = S \qquad (12)$$

Como as grandezas variam harmonicamente (proporcionais a $e^{j\omega t}$):

$$\frac{dT}{dz} = j \cdot \omega \cdot \rho_{m0} \cdot v; \qquad (13)$$

$$\frac{dv}{dz} = j \cdot \omega \cdot S \qquad (14)$$

e

$$I_3 = j \cdot \omega \cdot A \cdot D \qquad (15)$$

onde ρ_{m0} é a densidade mássica do meio de propagação sem perturbação.

A voltagem do circuito é obtida por:

$$V_3 = \int_{-\frac{L}{2}}^{\frac{L}{2}} E dz \qquad (16)$$

Escrevendo a equação da tensão externa em função de c^D e D, obtém-se:

$$T = c^D \cdot S - hD \qquad (17)$$

onde

$$h = \frac{e}{\varepsilon^S} \qquad (18)$$

Utilizando as equações (13) e (15):

$$\frac{d^2v}{dz^2} + \frac{w^2 \cdot \rho_{m0} \cdot v}{c^D} = 0 \qquad (19)$$

cuja solução é:

$$v = v_F \cdot e^{-j\cdot\overline{\beta}_a\cdot z} + v_B \cdot e^{j\cdot\overline{\beta}_a\cdot z} \qquad (20)$$

e

$$T = T_F \cdot e^{-j\cdot\overline{\beta}_a\cdot z} + T_B \cdot e^{j\cdot\overline{\beta}_a\cdot z} - h\cdot D \qquad (21)$$

onde os sub-índices F e B correspondem, respectivamente, as ondas se propagando no sentido positivo e negativo de z; e $\overline{\beta}_a$ é a constante de propagação para D = 0.

Definindo:

$$\begin{aligned} \overline{\beta}_a &= \omega \cdot \left(\frac{\rho_{m0}}{c^D}\right)^{\frac{1}{2}} \\ \overline{Z}_0 &= \left(\rho_{m0} \cdot c^D\right)^{\frac{1}{2}} \\ T_F &= -\overline{Z}_0 \cdot v_F \\ T_B &= \overline{Z}_0 \cdot v_B \end{aligned} \qquad (22)$$

onde $\overline{Z}_0$ é a impedância acústica.

Usando as condições de contorno e a solução da equação diferencial anterior:

$$v = \frac{-v_2 \cdot \operatorname{sen}\left(\overline{\beta}_a\left(z+\frac{L}{2}\right)\right) + v_1 \cdot \operatorname{sen}\left(\overline{\beta}_a\left(\frac{L}{2}-z\right)\right)}{\operatorname{sen}\overline{\beta}_a \cdot L} \qquad (23)$$

substituindo nas equações (11), (14) e (17) obtém-se a seguinte matriz:

$$\begin{bmatrix} F_1 \\ F_2 \\ V_3 \end{bmatrix} = -j \cdot \begin{bmatrix} Z_C \cdot \cot g(\overline{\beta}_a \cdot L) & Z_C \cdot \operatorname{cosec}(\overline{\beta}_a \cdot L) & \frac{h}{\omega} \\ Z_C \cdot \operatorname{cosec}(\overline{\beta}_a \cdot L) & Z_C \cdot \cot g(\overline{\beta}_a \cdot L) & \frac{h}{\omega} \\ \frac{h}{\omega} & \frac{h}{\omega} & \frac{1}{\omega \cdot C_0} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ I_3 \end{bmatrix} \qquad (22)$$

onde

$$C_0 = \frac{\varepsilon_2 \cdot A}{L} \qquad (23)$$

e

$$Z_C = \overline{Z}_0 \cdot A \qquad (24)$$

4.3. Construção de Transdutores

Para este projeto, foram construídos dois transdutores piezelétricos. O esquema de montagem e principais componentes dos dois transdutores construídos estão ilustrados na figura a seguir.

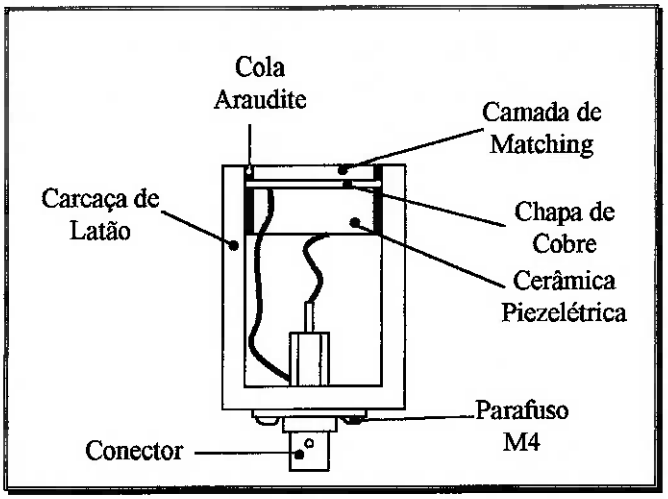


Figura 4.3. – Esquema de construção do transdutor 1

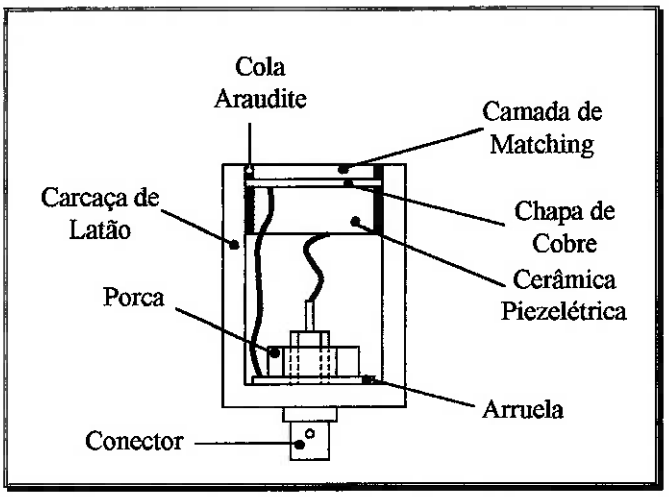


Figura 4.4. – Esquema de construção do transdutor 2

Sobre a construção destas peças é importante ressaltar que carcaça de latão é usinada, e serve como o "terra" do circuito, conectado por um fio soldado à placa de cobre, que serve como eletrodo para a cerâmica piezelétrica (como o discutido no item anterior). Após a usinagem da carcaça, foi feita a soldagem dos fios condutores no conector – cerâmica e terra – cobre, como mostram as figuras anteriores. Após completa esta etapa, a camada de maching, a placa de cobre e a cerâmica piezelétrica são fixadas entre si por uma cola condutora. Após feita esta colagem, o conector foi preso à carcaça, externamente ou internamente, como é o caso dos transdutores 1 e 2

respectivamente. Enfim, o conjunto matching, cobre e cerâmica é colado na carcaça por uma cola epoxi (Araudite). Esta colagem é feita da seguinte maneira: é feita uma aplicação da cola na parte lateral do conjunto e após a isto, este é encaixado na abertura da carcaça, possibilitando a colagem. Após o encaixe o a carcaça é virada de modo que o conjunto colado fique em contato com um plástico, inerte a este tipo de cola, apoiado em uma mesa (por exemplo) para possibilitar que a cola seque de forma que o conjunto cerâmica, cobre e matching fiquem em uma posição correta em relação à carcaça.

Foram, então feitas medidas dos dois transdutores construídos, de modo se verificar a freqüência de ressonância dos mesmos. Assim, as curvas de impedância e fase para os dois transdutores foram respectivamente:

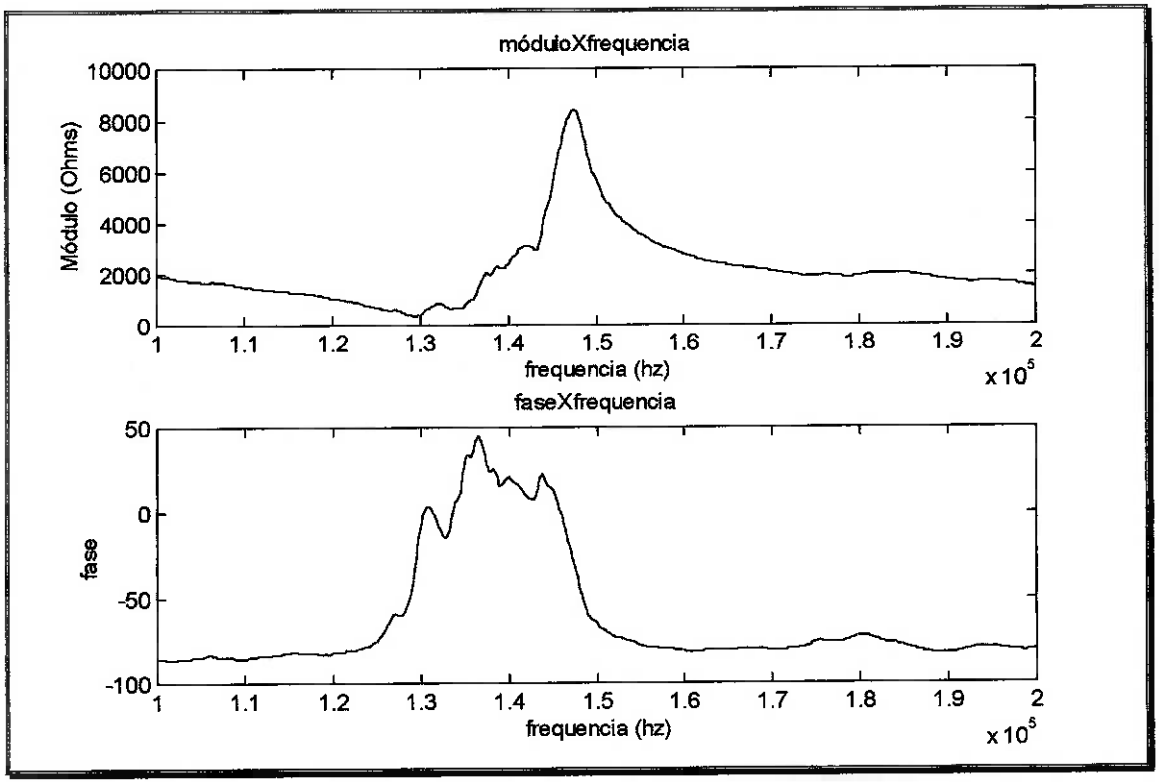


Figura 4.5 - Módulo e Fase do transdutor 1 em função da freqüência

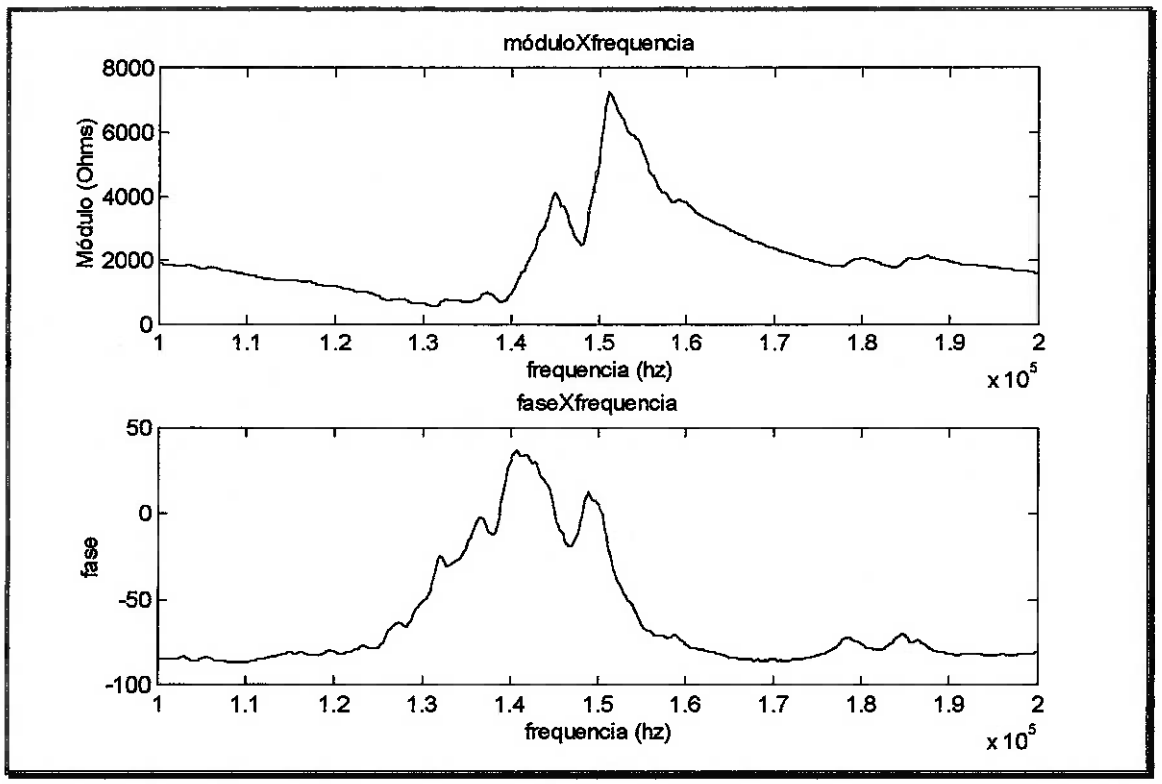


Figura 4.6. - Módulo e fase do transdutor2 em função da freqüência

Pode-se perceber que as curvas encontradas foram muito semelhantes. Verifica-se a ressonância próxima dos 150 kHz, como era de se esperar. Observa-se uma pequena diferença entre os módulos de impedância na ressonância para os dois transdutores, mas para este projeto esta diferença não tem magnitude considerável.

Também foram ensaiados os dois transdutores imersos na água. Assim as curvas foram:

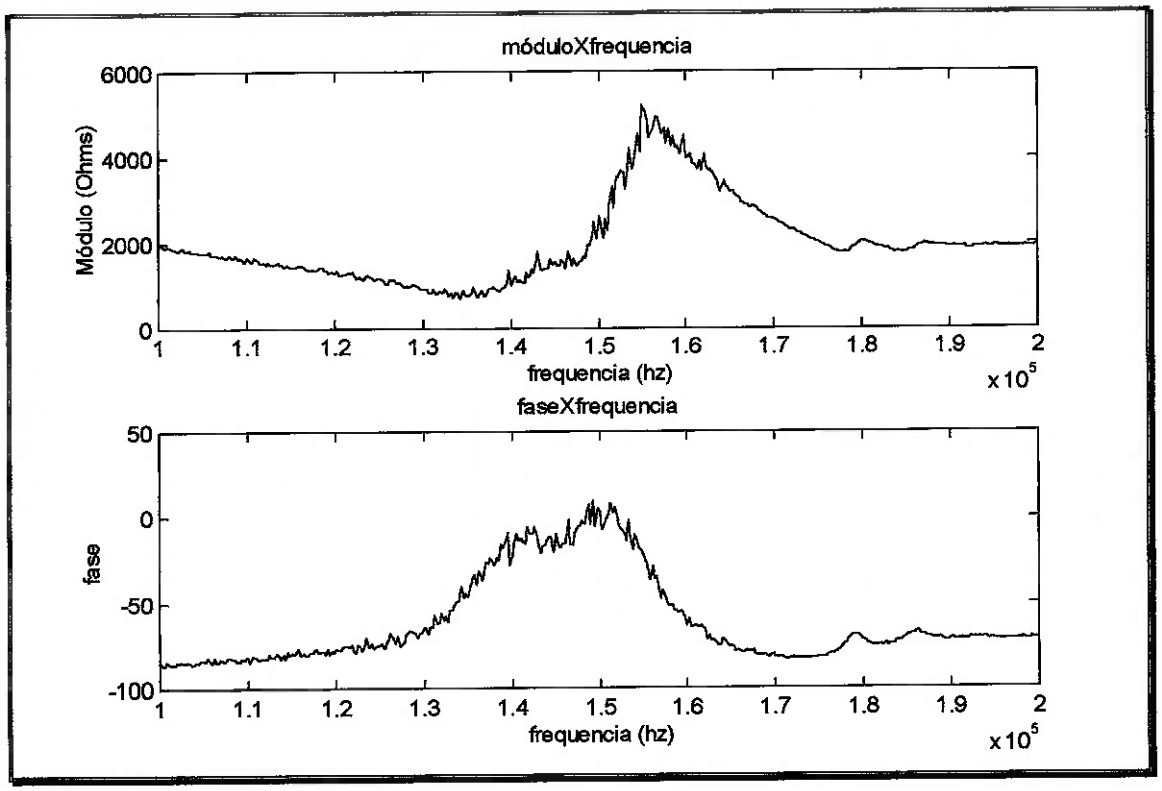


Figura 4.7. - Módulo e fase em função da freqüência do transdutor 1 imerso

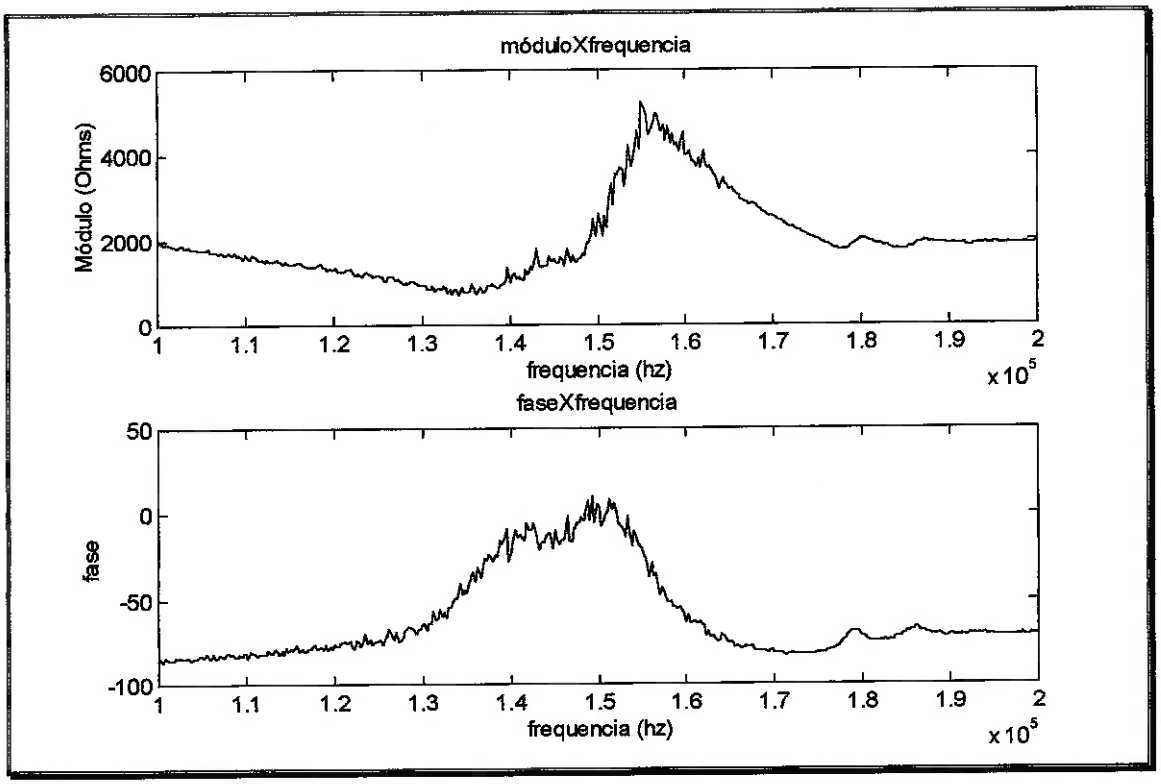


Figura 4.8. - Módulo e fase em função da freqüência para o transdutor 2 imerso

A partir destes gráficos, observa-se que apesar do módulo diminuir para os dois casos, e o sinal se mostrar mais ruidoso, os transdutores mantiveram a freqüência de ressonância na água, o que é de suma importância para este projeto.

4.4. Circuito equivalente dos Transdutores

Para os dois transdutores, no caso deles imersos ou não imersos, foi calculado um circuito equivalente, ou seja um circuito que apresentasse uma resposta semelhante da impedância em função da freqüência. Para o caso do transdutor 1 não imerso (em contato com o ar), o circuito equivalente foi:

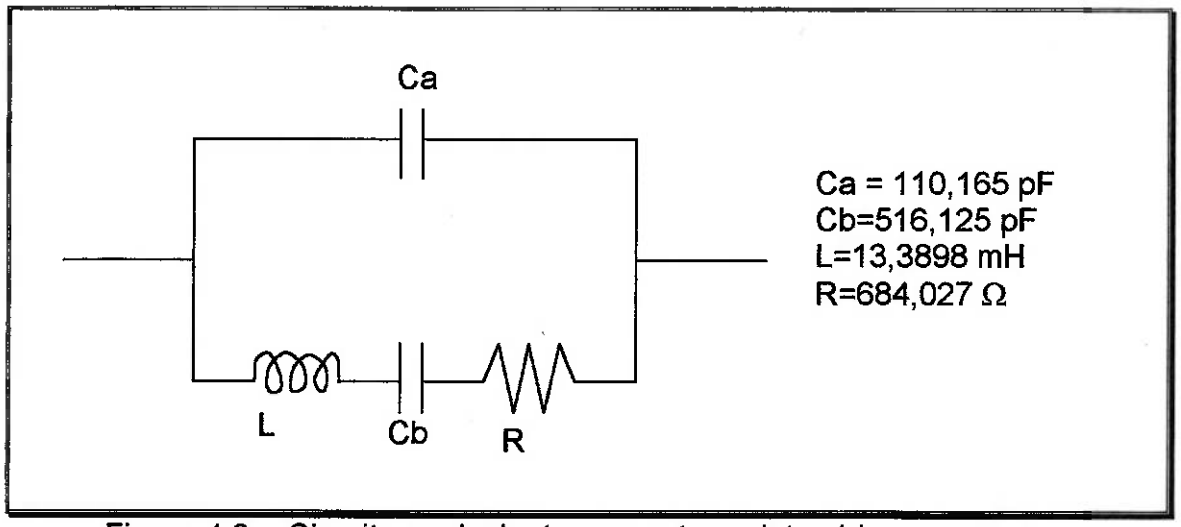


Figura 4.9. - Circuito equivalente para o transdutor 1 imerso no ar

Assim, o respectivo gráfico de módulo e fase em função da freqüência para este circuito é:

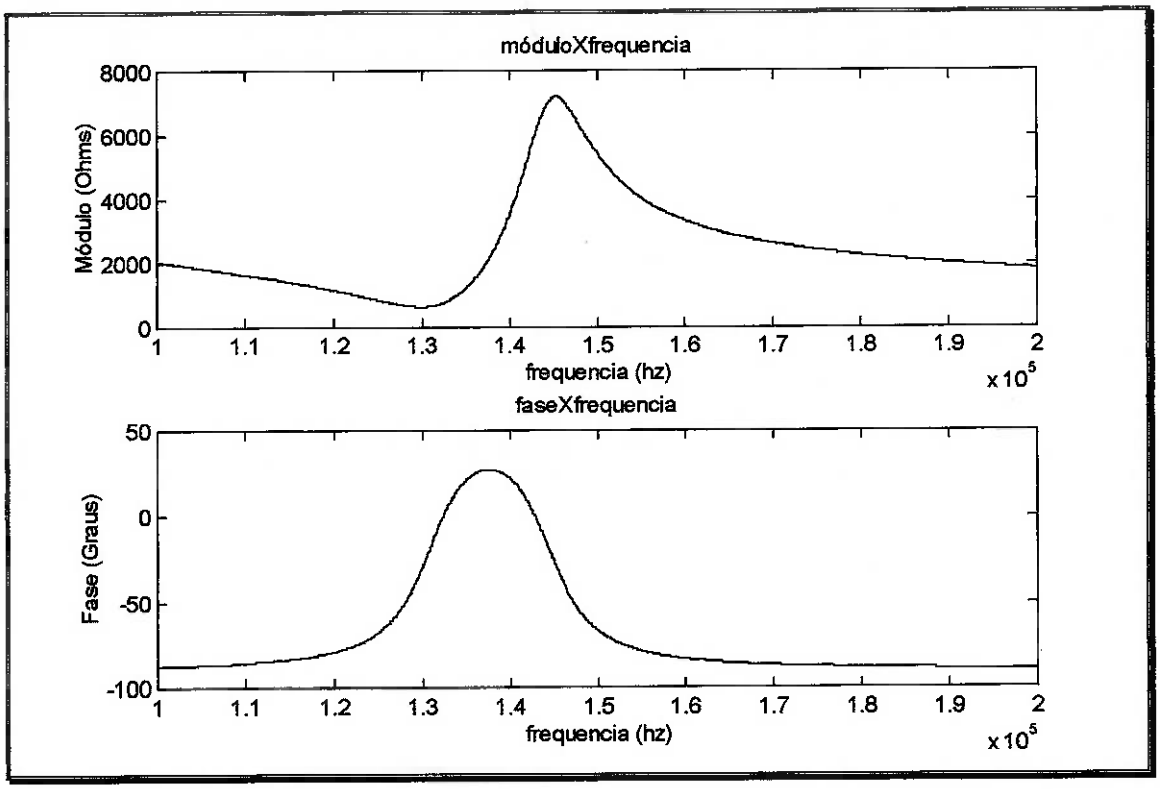


Figura 4.10. - Curvas do circuito equivalente para o transdutor 1 no ar

Já para o mesmo transdutor imerso na água, o circuito equivalente ficou:

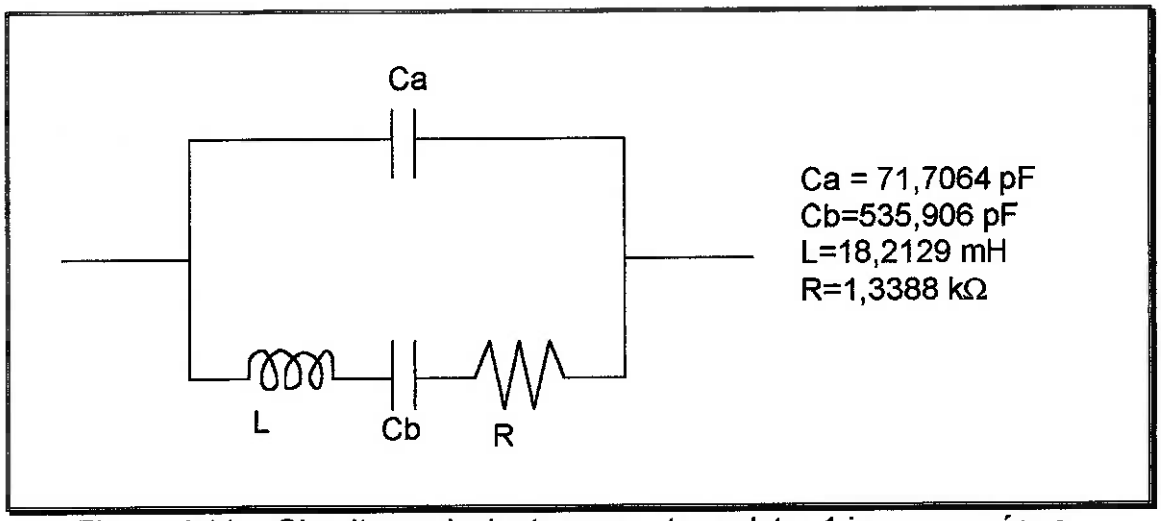


Figura 4.11. - Circuito equivalente para o transdutor 1 imerso na água

E a respectiva curva ficou:

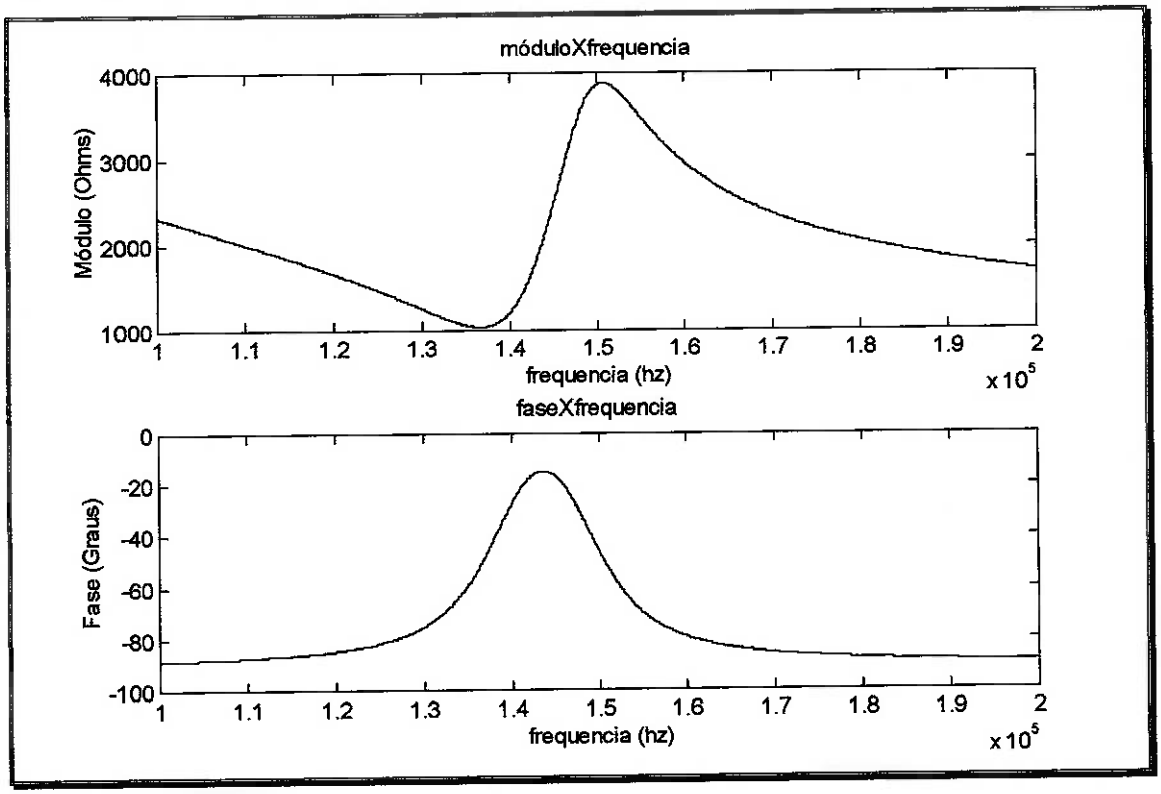


Figura 4.12. - Curvas do circuito equivalente para o transdutor 1 imerso

Verifica-se assim, que para este transdutor, os resultados obtidos referentes aos circuitos equivalente pelo impedômetro foram satisfatórios, pois as curvas entre o real medido e a simulação do circuito equivalente apresentaram as mesmas características.

Já para o transdutor 2, foram calculados os circuitos equivalente deste imerso e não imerso na água da mesma forma, através do impedômetro. Assim sendo, o circuito equivalente para o transdutor 2 não imerso ficou:

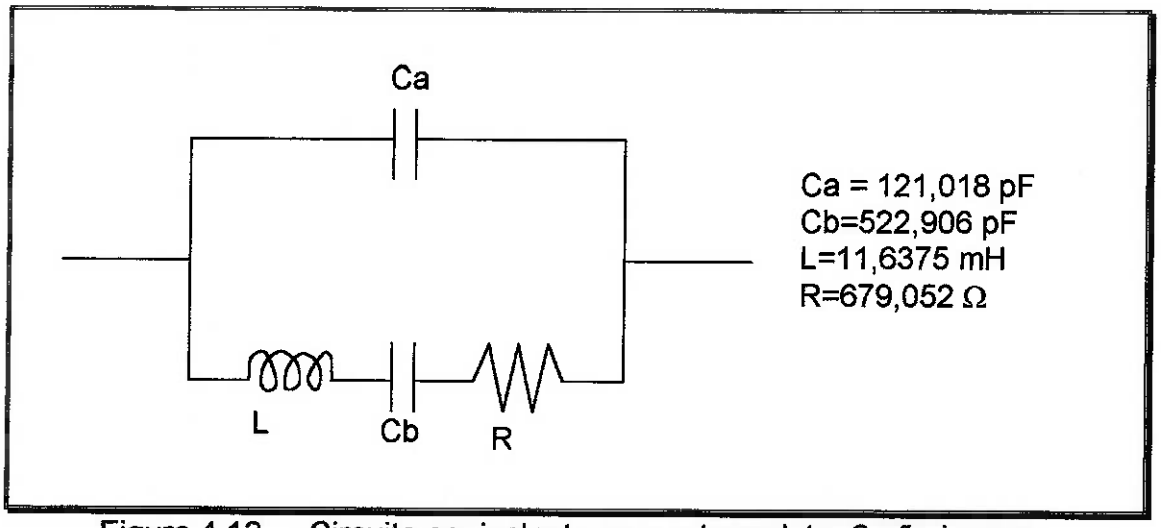


Figura 4.13. - Circuito equivalente para o transdutor 2 não imerso

E a as curvas deste circuito:

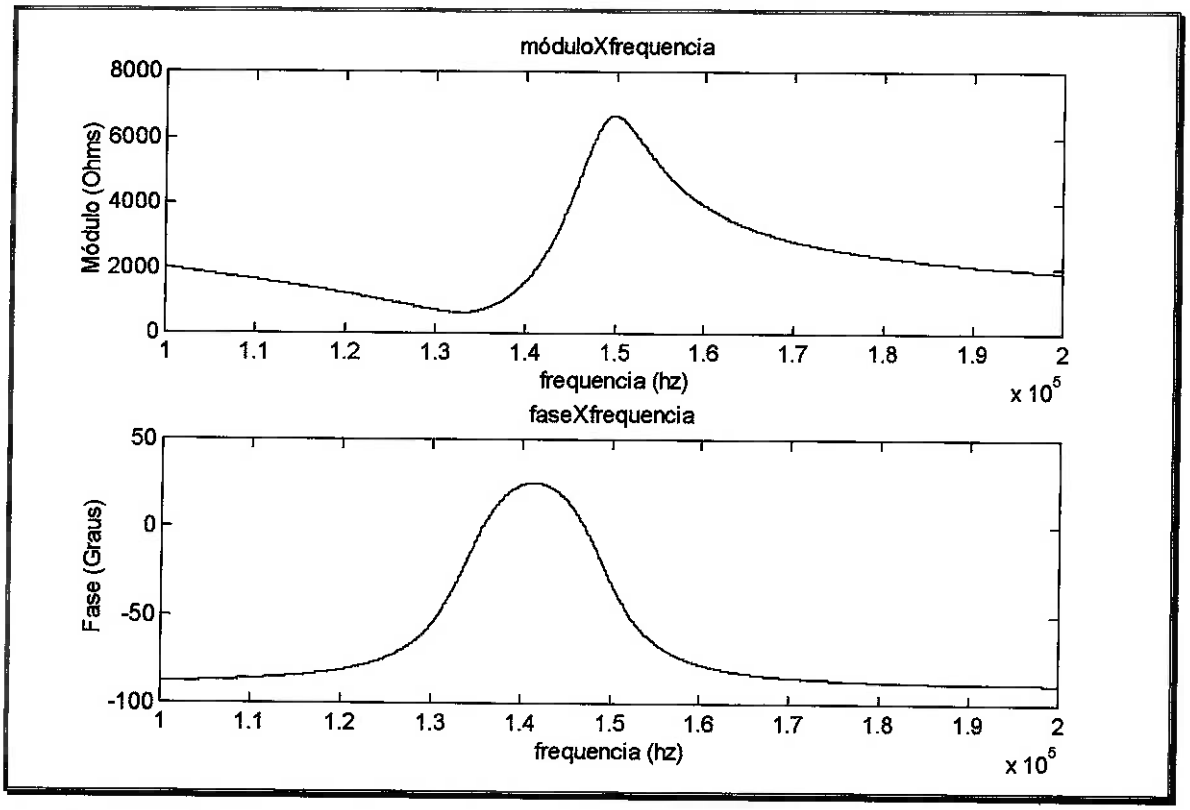


Figura 4.14. - Curvas do circuito equivalente do transdutor 2 não imerso

Já para este transdutor imerso na água, o circuito equivalente ficou:

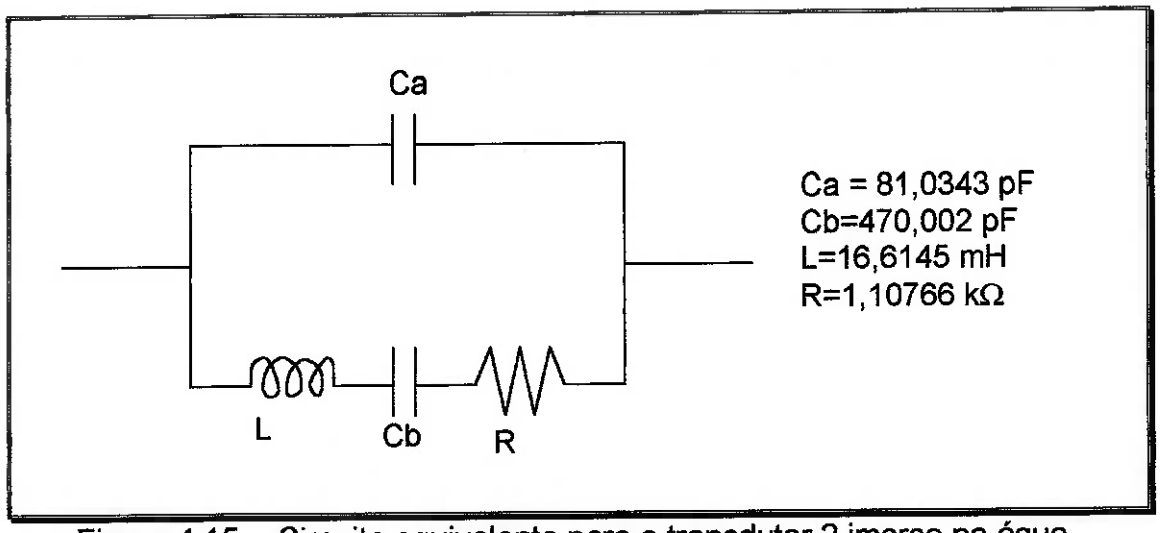


Figura 4.15. - Circuito equivalente para o transdutor 2 imerso na água

E, a curva obtida foi:

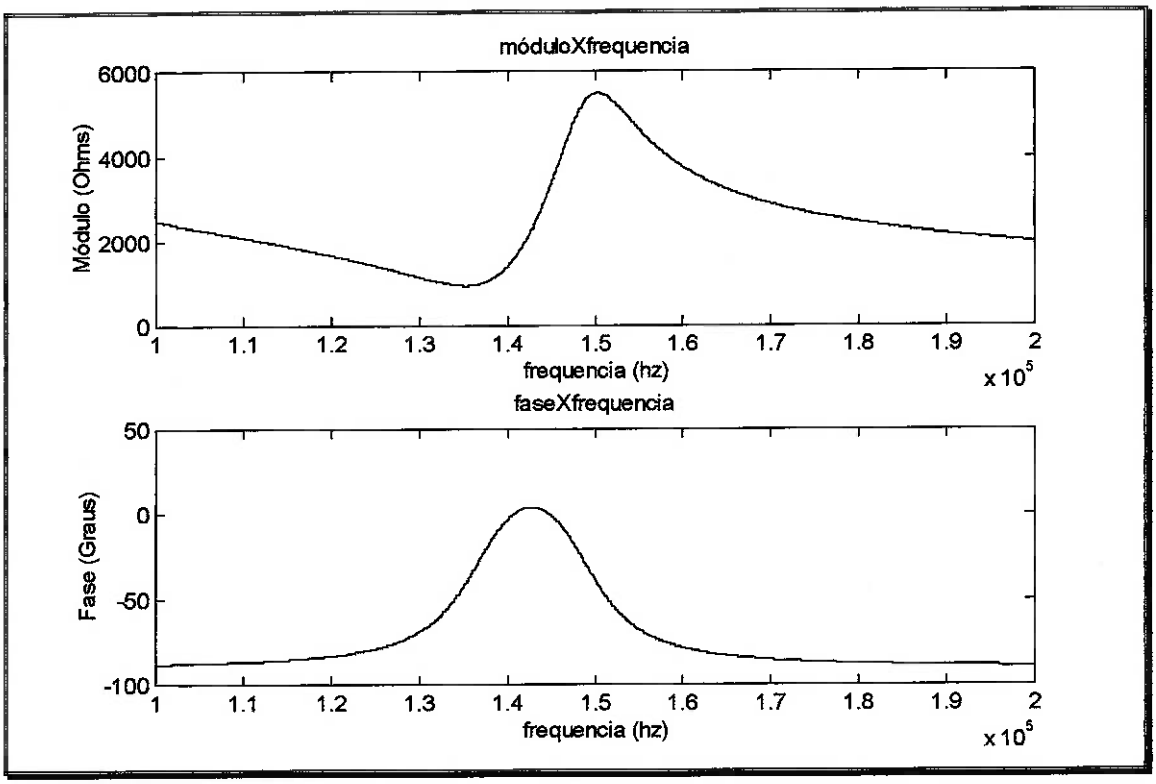


Figura 4.16. - Curvas do circuito equivalente do transdutor 2 imerso na água

Concluindo, para os dois transdutores construídos, foram obtidos bons resultados, referentes a freqüência de ressonância pretendida, para os

mesmos imersos na água. Além disso, foram calculados circuitos equivalentes para estes transdutores tanto imersos como não imersos na água. Estes circuitos, por sua vez, foram testados, onde verificou-se que o circuito equivalente calculado está próximo ao real medido (direto no impedômetro).

5. Estudo do Microcontrolador PIC

Primeiramente deve-se esclarecer que um microcontrolador corresponde a um microprocessador com os seus periféricos típicos (RAM,ROM, Timers, Serial, etc.), todos num só chip. O microcontrolador PIC escolhido foi o PIC16C73B (Microship, EUA), devido ao fato de ser de fácil acesso para alunos e por ser completo para as necessidades do projeto. O estudo, descrito em seguida, foi realizado a partir do datasheet do PIC16C73B [3], obtido através do site oficial da microchip.

5.1 Arquitetura

Os microcontroladores PIC apresentam uma estrutura de máquina interna do tipo Havard, enquanto a maioria dos microcontroladores tradicionais apresenta uma arquitetura tipo Von-Neumann. Na arquitetura tradicional, existe apenas um barramento (bus) interno por onde passam as instruções e os dados. Já na arquitetura tipo Havard existem dois barramentos internos, sendo um de dados e outro de instruções. Para o microcontrolador PIC estudado o barramento de dados é sempre de 8 bits e o de instrução é de 14 bits. Isto permite que enquanto uma instrução está sendo realizada uma outra seja "buscada" da memória. Como o barramento de instruções é maior do que 8 bits, a operações de códigos da instrução já inclui o dado e o local de onde operará, implicando na utilização de apenas uma posição da memória, economizando memória de programa.

5.2 Estruturação Interna.

A figura a seguir ilustra a estruturação interna do PIC16C73B, retirada diretamente do datasheet.

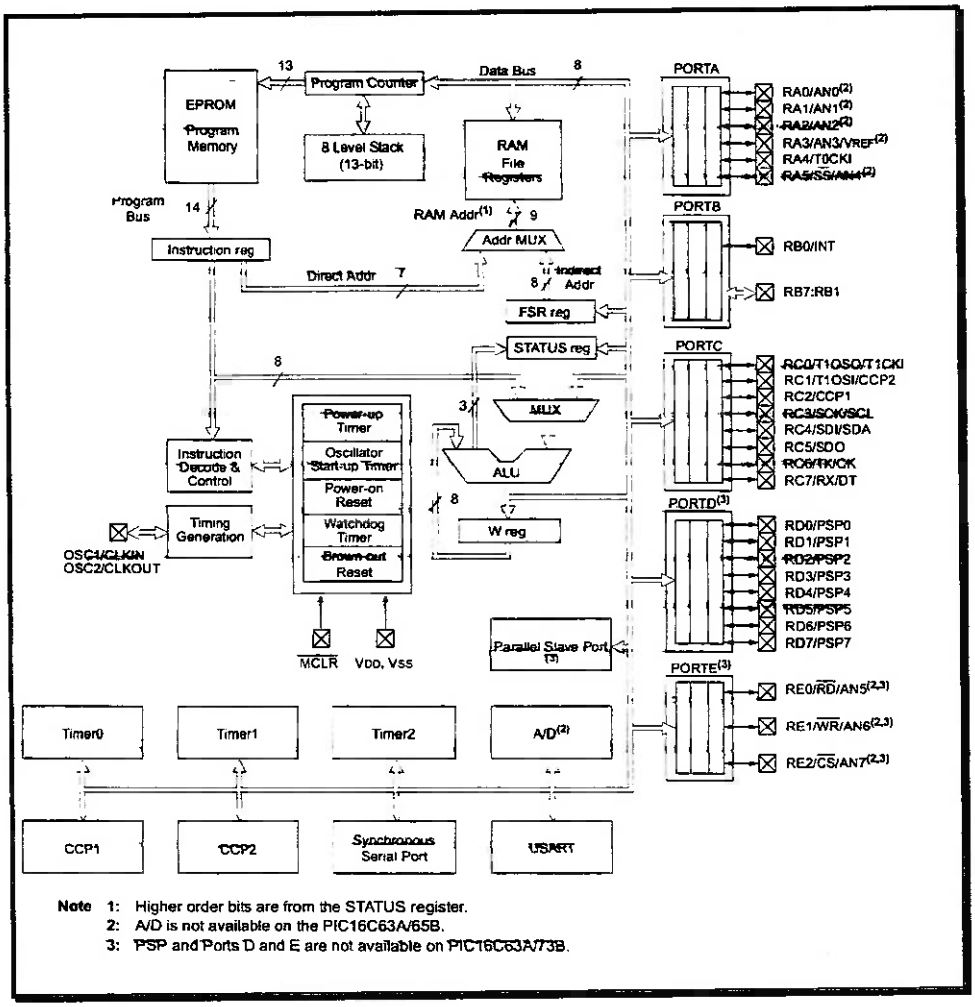


Figura 5.1. – Esquema da arquitetura interna do PIC

A ULA é o componente no qual são realizadas as operações aritméticas entre os valores da memória de dados e o do registrador de trabalho (W), que está diretamente ligado à ULA.

5.3 Periféricos.

Timer0: Um timer de 8 bits, que através do Prescaler, pode ser programado para dividir a freqüência do clock em 1, 2, 4, 8, 16, 32, 64, 128, 256. Sendo que esta freqüência do clock pode ser a da máquina (interna) ou externa. Este timer seta um flag e pode realizar uma interrupção (se esta

estiver habilitada), quando a contagem estoura, ou seja, passa de 0XFF para 0X00.

Timer1: Um timer/contador de 16 bits, pois pode operar como timer ou como contador. Por ser um Timer/Contador de 16 bits possuí dois registradores de 8 bits (TMR1H e TMR1L). No primeiro caso o clock é interno (clock da máquina) e no segundo é externo, ou seja, incrementando o contador a cada borda de subida do clock externo. A freqüência do clock pode ser dividida 1, 2, 4, 8 pelo prescaler. Quando a contagem estoura um flag é setado e pode se realizar uma interrupção.

Timer2: Um timer de 8 bits, que pode ser programado para dividir a freqüência do clock por 1, 4, 16. Este timer tem o período dado pelo registrador de 8 bits PR2, cujo valor default é 0XFF. Ou seja, o timer conta de 0X00 até o valor de PR2 e estoura, quando isto ocorre um flag é setado e pode se realizar uma interrupção.

Capture/Compare/PWM- CCP1 e CCP2: Os módulos CCP podem trabalhar nos modos Capture, Compare e PWM. No primeiro modo, o valor do timer1 é capturado por dois registradores de 8 bits (CCRP1H e CCRP1L) quando ocorre um evento externo no pino CCP1 ou CCP2. Quando ocorrer a "captura" um flag é setado e pode ocorrer interrupção No modo compare o valor do timer1 é comparado constantemente com o valor dos registradores de um dos CCP e, quando os valores forem iguais, um dos pinos CCP é alterado, um flag é setado e pode ocorrer uma interrupção. No modo PWM, pulsos são gerados num dos pinos CCP, sendo que o período e a largura destes são controlados pelo Timer2.

SYNCHRONOUS SERIAL PORT (SSP): Uma interface serial síncrona útil para comunicar com outros periféricos (Serial EEPROMs, Displays, conversores A/D, CIs, etc) ou microcontroladores.

UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART): É uma interface serial que pode operar no modo

assíncrono, utilizado para comunicar com um PC, por exemplo, ou operar no modo síncrono, para se comunicar com outros periféricos.

A/D: Converte uma entrada analógica para um sinal digital de 8 bits. A sua freqüência de amostragem máxima é de aproximadamente 65kHz.

5.4 Memórias do PIC.

Existem dois tipos de memórias existentes no PIC: a memória de programa (EPROM para o PIC estudado) e memória de dados (RAM).

5.4.1 Memória de programa

A memória de programa do PIC16C73B é uma EPROM de 14 bits, ou seja, pode ser gravada várias vezes, já que podem ser apagados por meio de luz ultra violeta. A organização desta memória é ilustrada através da seguinte figura:

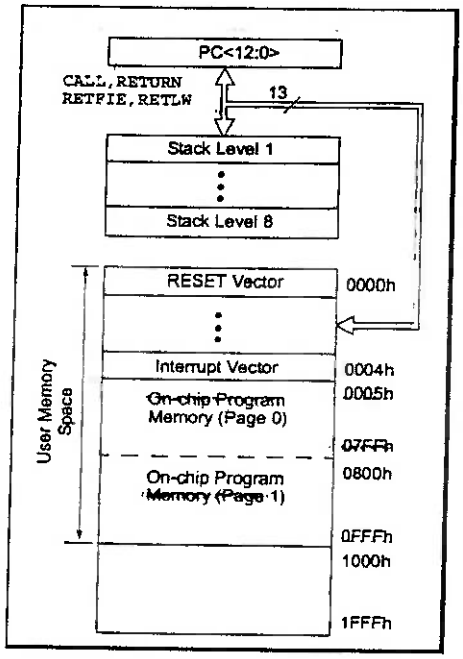


Figura 5.2. – Esquema da memória EPROM

O vetor de reset nada mais é do que o primeiro endereço da memória de programa que será executado quando o PIC começa a rodar, ou seja, quando o mesmo for alimentado ou resetado.

O vetor de interrupção é o endereço (0x04) no qual ocorrerá o início do tratamento de todas as interrupções, ou seja, onde estará as rotinas de todas as interrupções.

A pilha (stack) é um local, separado da memória de programação, no qual serão armazenados os endereços de retorno quando se utilizar instruções de chamada de rotina.

5.4.2 Memória de dados

A memória de dados é a RAM, que é utilizada para guardar todas as variáveis (registradores de uso geral) e registradores especiais utilizados pelo programa. Esta memória armazena dados de 8 bits e é volátil, ou seja, os dados são perdidos quando o PIC é desligado.

Os registradores especiais são aqueles que permitem configurar os periféricos internos do PIC (ports, conversores, timers, seriais, interrupções, etc), conhecer status de operações e portas, dentre outras funções.

Os registradores de uso geral são utilizados para armazenar as variáveis definidas pelo usuário.

A figura a seguir mostra como esta memória está organizada

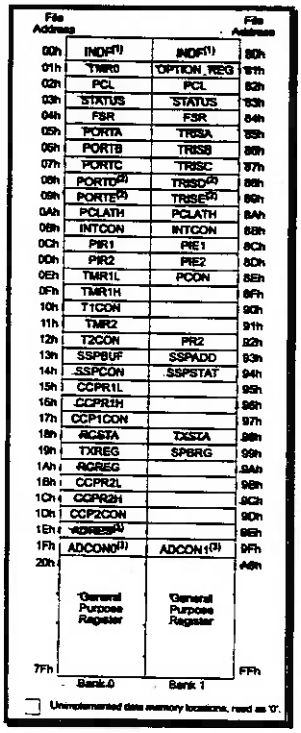


Figura 5.3. – Esquema da memória RAM

5.5 Programação

A linguagem de programação a ser utilizada será a Assembler, pois o sistema de compilação, simulação e gravação a ser utilizado é o software MPlab, que se utiliza desta linguagem.

6. Circuito de emissão

Este circuito é o responsável pelo acionamento do transdutor. Para acioná-lo, o circuito deve fornecer uma voltagem variável e cuja módulo seja suficiente para excitá-lo. Tendo isto em vista foi elaborado o seguinte circuito de emissão:

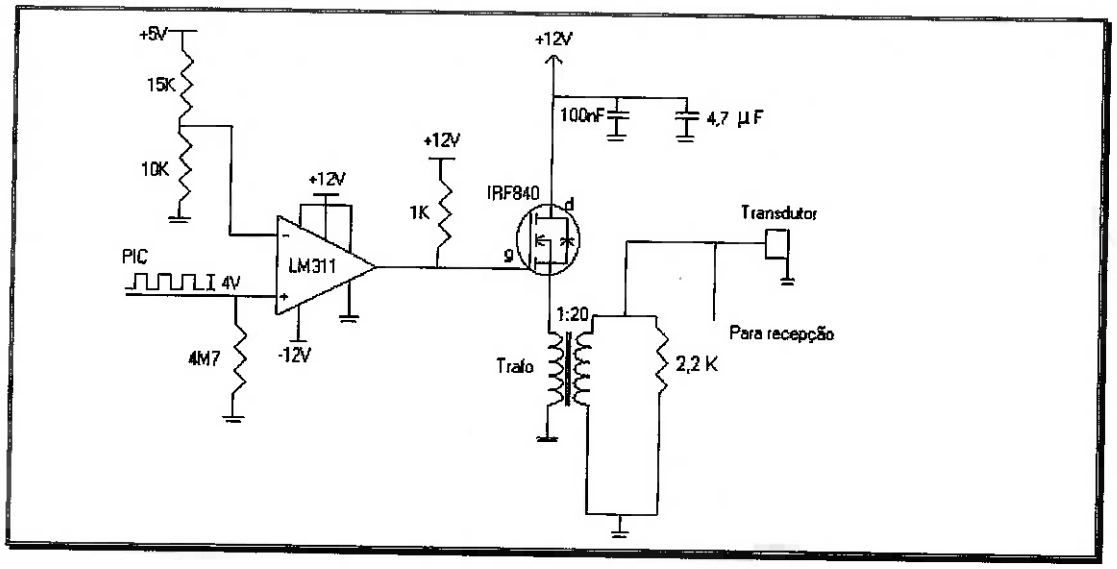


Figura 6.1. – Circuito de emissão

O circuito funciona da seguinte forma: o PIC gera um trem de pulsos de amplitude de no mínimo 4 V, alimentando o terminal positivo do comparador LM311 (Texas Instruments, EUA). O terminal negativo do mesmo (tensão de referência) é alimentado por um divisor de tensão, que fornece 2 V. Quando a tensão no terminal positivo for 4V, portanto, maior que a tensão de referência, o sinal de saída do comparador será de 12 V. Quando a entrada no terminal positivo for zero o sinal de saída do comparador será 0V. Este sinal irá chavear o POWER MOSFET (International Retifier, EUA), chaveando, desta forma, o primário do transformador, propiciando a voltagem variável. O transformador transformará a tensão pico a pico num valor considerável (200V) para poder acionar o transdutor.

Tanto o comparador como o POWER MOSFET estão sendo utilizados para chavear tensão. Porque, então, não usar apenas um deles? Isto se deve ao fato de ao se utilizar apenas o comparador para chavear o primário do transformador, este exigirá uma corrente que o queimará. Agora, se apenas o POWER MOSFET fosse utilizado, o sinal do PIC não seria suficiente para chaveá-lo.

Os capacitores de 4,7μF e 100 nF são usados para estabilizar a fonte em 12 V, já que a mesma possui uma resistência interna e pode ocorrer uma queda momentânea de tensão quando altas correntes são exigidas.

Para o transformador do circuito foi utilizado um núcleo de ferrite modelo RM-6S da fabricante Thornton, o fio do primário de bitola 28 AWG e do secundário de 32 AWG. Sendo que para o veículo móvel foram utilizadas 3 espiras no primário e 60 espiras no secundário e no alvo fixo foram utilizadas 2 espiras no primário e 40 espiras no secundário.

O circuito de emissão do veículo móvel é acionado por um comando do PC, via comunicação serial, ou seja, pelo canal serial. O PC envia um comando para que o PIC comece a geral o trem de pulsos para que a emissão ocorra. Para que esta comunicação se realize é necessário uma conversão de tensões, já que o PC trabalha com voltagens de comunicação do padrão RS-232 e o PIC usa tensões TTL. Para a finalidade de se adequar estas tensões de comunicação foi utilizado o conversor MAX232 (MAXIM, EUA).

O cabo serial utilizado foi de 25 pinos e foi ligado na porta COM1 do PC.

O circuito foi montado com todos os componentes necessários, segundo os datasheets do PIC e do MAX232. O projeto deste circuito se encontra nos anexos A e B, juntamente com o projeto de todo o circuito de emissão.

Já para o alvo fixo, o trem de pulsos do PIC, é iniciado assim que um sinal do veículo móvel for recebido.

7. Acionamento do transdutor

O sinal do acionamento do transdutor do veículo móvel, realizando o teste com este dentro de um tanque de água, e com a voltagem sendo fornecida pelo seu transformador (já descrito no item anterior), porém com 9 espiras no primário e 160 espiras no secundário, é mostrado na figura a seguir:

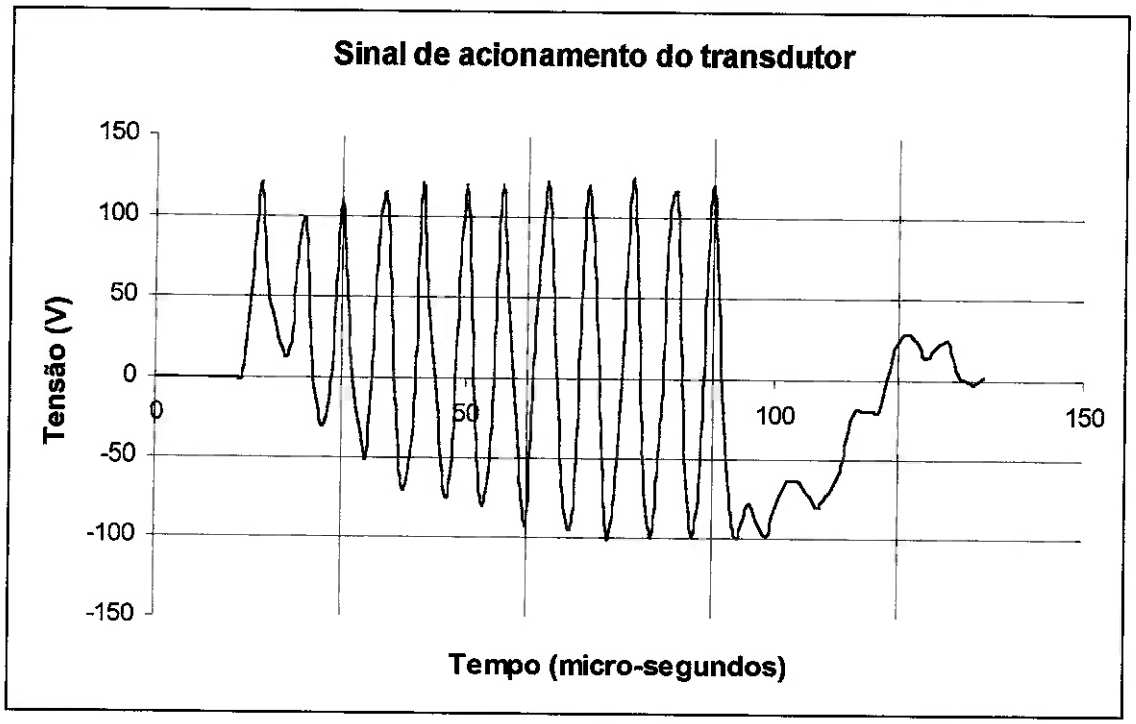


Figura 7.1. – Tensão na entrada do transdutor

Pode-se observar que o sinal apresenta um transiente nos seus instantes iniciais e apresenta um comportamento oscilatório aproximadamente senoidal no restante do sinal. Este fenômeno não é desejável para o acionamento do transdutor.

A causa desse transitório é o não "casamento" entre as partes imaginárias do secundário do transformador e do transdutor, na freqüência de operação, ou seja, a indutância do secundário do transformador não cancela a parte capacitiva do transdutor [6].

Para se resolver este problema, poderia-se adotar duas soluções. Uma seria escolher outro transformador de forma a casar a indutância, a outra seria adicionar em série com o transdutor capacitores ou indutores.

A solução adotada foi utilizar o mesmo núcleo de ferrite já utilizado, calcular o número de espiras necessários para promover o casamento e montar o transformador, de acordo com a referência [6].

7.1. Cálculo do número de espiras para o veículo móvel

O primeiro passo foi calcular o AL para o núcleo de ferrite urilizado (modelo RM-6S), através da seguinte expressão:

$$A_L = \frac{L}{n^2} \qquad (25)$$

onde: L é a indutância do primário ou secundário e n é o número de espiras, também, no primário ou no secundário.

Medindo-se a impedância do secundário do transformador, através do impedômetro, obteve-se:

$$Z_{imaginário} = 34{,}5\ k\Omega \qquad (26)$$

Mas:

$$Z_{imaginário} = \omega \cdot L = 2 \cdot \pi \cdot f \cdot L \qquad (27)$$

onde: f é a freqüência de operação e L é a indutância

Para a freqüência de operação da unidade móvel, que é de 150 kHz, e sabendo que o número de espiras no secundário é 160 (ver item 7), obtém-se:

$$A_L = \frac{L}{n^2} = \frac{34{,}5 \cdot 10^3}{2 \cdot \pi \cdot 150 \cdot 10^3 \cdot 160^2} = 1430 \cdot 10^{-9} \qquad (28)$$

Através do impedômetro, sabe-se que na freqüência de 150 kHz (freqüência de operação) o transdutor possuí uma impedância imaginária de:

$$Zt_{imaginário} = -5\,k\Omega \qquad (29)$$

Portanto, o número de espiras no secundário será:

$$\begin{aligned} &\omega \cdot L_{secundário} = -Zt_{imaginário} = 5 \cdot 10^3 \Rightarrow \\ &\omega \cdot n_{secundário}{}^2 \cdot A_L = 5 \cdot 10^3 \Rightarrow \\ &n_{secundário} = \sqrt{\frac{5 \cdot 10^3}{1430 \cdot 10^{-9} \cdot 2 \cdot \pi \cdot 150 \cdot 10^3}} = 60{,}90 \end{aligned} \qquad (30)$$

Como a relação do transformador é de 1:20, escolhe-se que o secundário terá 60 espiras e o primário 3.

Um novo transformador , com núcleo de ferrite RM-6S, foi construído e montado no circuito de emissão. O sinal de acionamento do transformador fica agora:

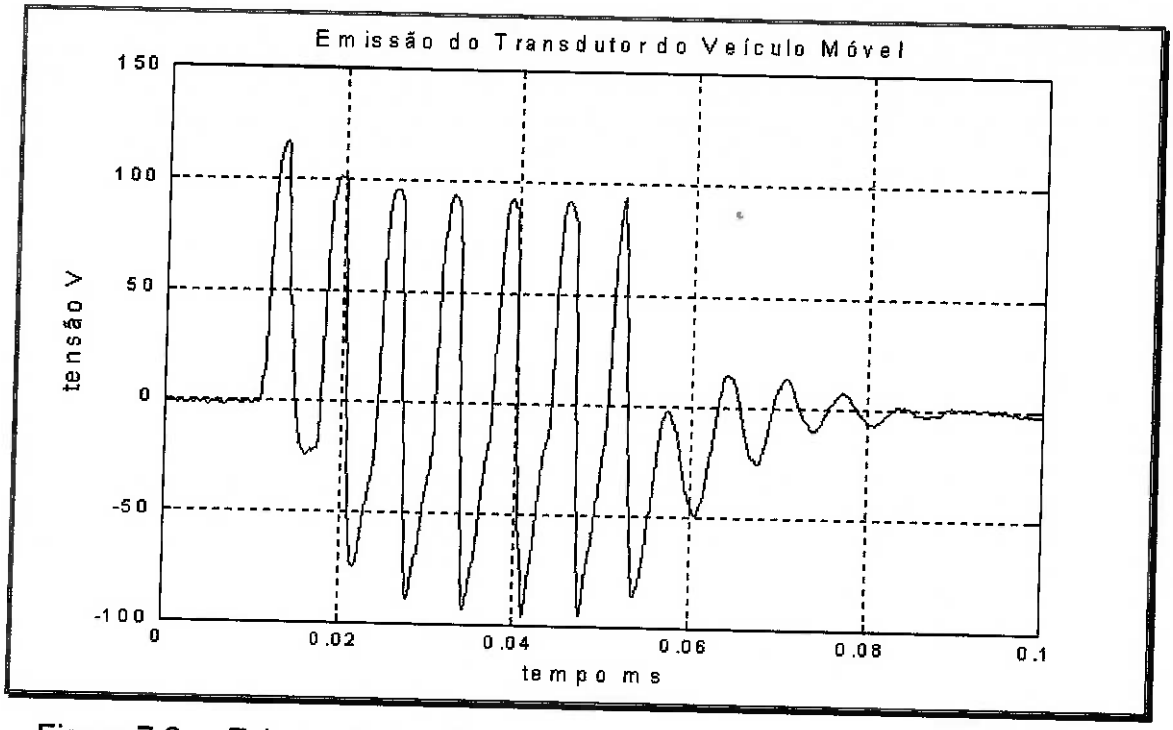


Figura 7.2. – Esboço da tensão na entrada do transdutor do veículo móvel, acionado pelo novo transformador

Assim, verificou-se que o transitório praticamente desapareceu. Analisando a figura anterior observa-se que o transitório se resume a um único ciclo, o que representa uma melhora muito grande, em relação ao transformador utilizado anteriormente.

7.2. Cálculo do número de espiras para o alvo fixo

Para o alvo fixo, foi adquirido um transformador RM-6S cujo AL é igual a $2000 \cdot 10^{-9}$ H.

Agora, medindo-se através do impedômetro, sabe-se que na freqüência de 170 kHz (freqüência de operação) o transdutor possuí uma impedância imaginária de:

$$Zt_{imaginário} = -2{,}6176\ k\Omega \qquad (31)$$

Portanto, o número de espiras no secundário será:

$$\omega \cdot L_{secundário} = -Zt_{imaginário} = 2{,}6176 \cdot 10^3 \Rightarrow$$
$$\omega \cdot n_{secundário}^{\ 2} \cdot A_L = 2{,}6176 \cdot 10^3 \Rightarrow \quad (32)$$
$$n_{secundário} = \sqrt{\frac{2{,}6176 \cdot 10^3}{2000 \cdot 10^{-9} \cdot 2 \cdot \pi \cdot 170 \cdot 10^3}} = 35{,}01$$

Como a relação do transformador é de 1:20, escolhe-se que o secundário terá 40 espiras e o primário 2.

Assim, acurva de excitação do transdutor acionado por este novo transformador fica:

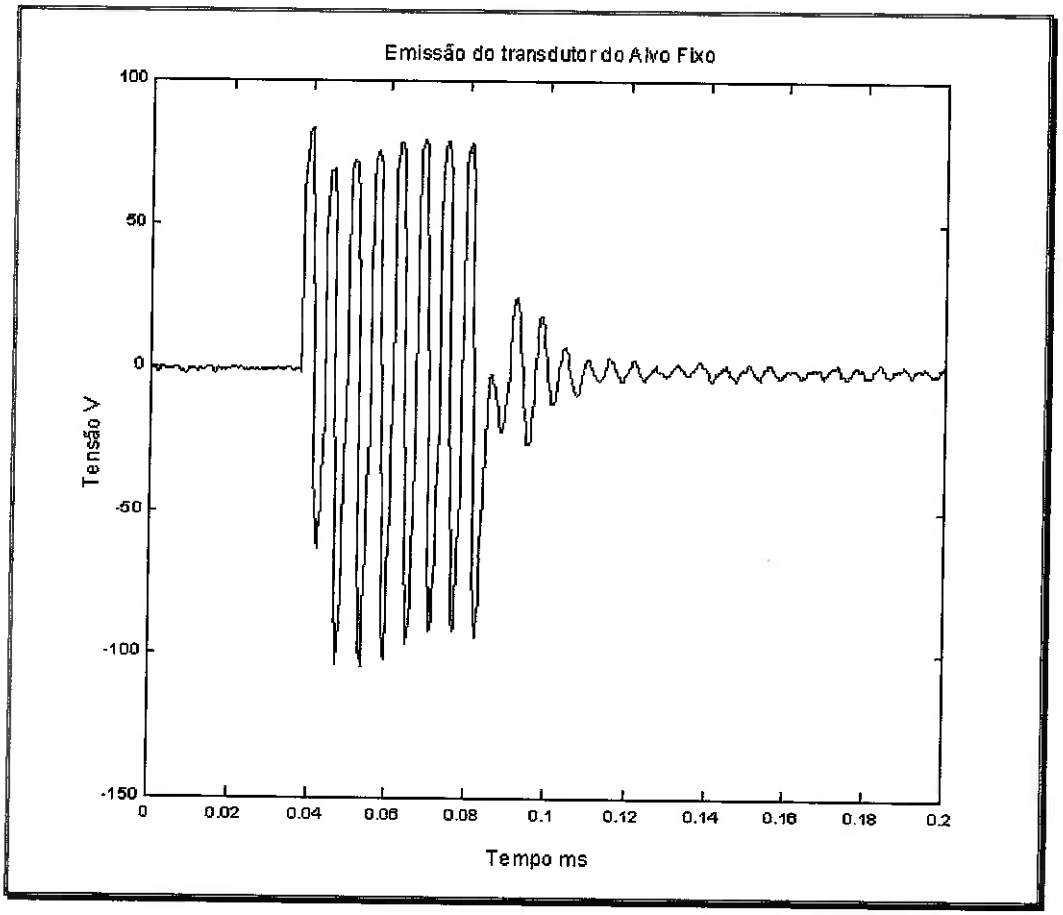


Figura 7.3. – Esboço da tensão na entrada do transdutor do alvo fixo

8. Circuito de recepção

8.1. Esquema Geral

O circuito de recepção é o responsável pela detecção do sinal de resposta, que é uma diferença de potencial (tensão), proveniente dos terminais do transdutor e, de informar ao microcontrolador PIC para parar de cronometrar o tempo e realizar o cálculo da distância, no caso do tratamento de sinal analógico, ou fornecer dados para serem digitalizados pela placa A/D, no caso de tratamento digital do sinal. O circuito de recepção, para ambos os tratamentos de sinal, é ilustrado pela figura a seguir:

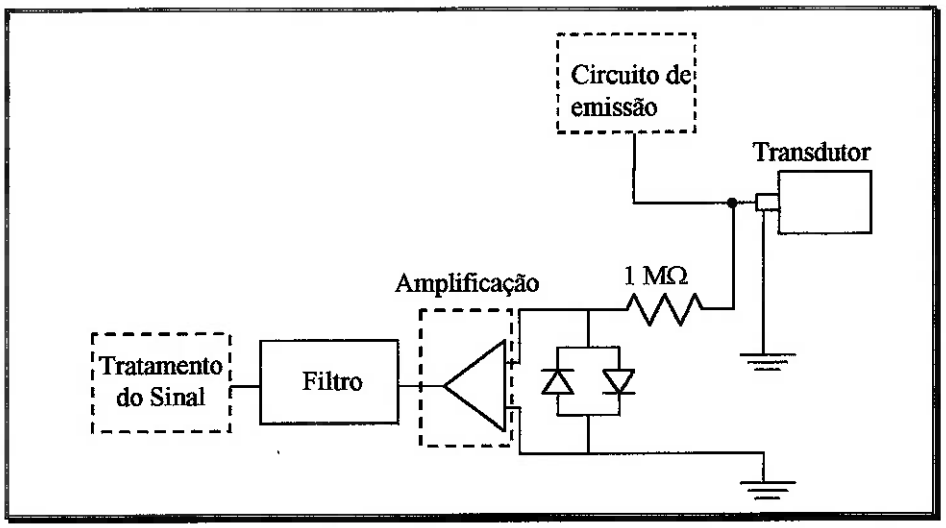


Figura 8.1. – Esquema do Circuito de recepção

O circuito funciona da seguinte forma: sobre os terminais dos diodos cairá toda a tensão de resposta proveniente do transdutor, esta tensão será amplificada, pois esta é da ordem de mV. Em seguida ela será filtrada, a fim de eliminar os sinais de alta freqüência e, finalmente, a tensão proveniente do filtro será tratada para que se possa calcular a distância desejada.

A seguir será explicado individualmente os componentes do circuito de recepção.

8.2 Diodos

Como já explicado, os diodos são os responsáveis por delimitar a tensão de resposta proveniente do transdutor. A idéia da utilização de diodos retificadores é a seguinte: a tensão de resposta é da ordem de mV, não sendo, portanto, maior que a tensão de joelho de um diodo retificador de silício (0,7V). Desta forma, o diodo ficará reversamente polarizado e toda a tensão de resposta cairá sobre o mesmo. Utiliza-se dois diodos em paralelo, pois o sinal de resposta possuí um ciclo positivo e outro negativo.

Por outro lado, na emissão, os diodos conduzirão, fixando a tensão de entrada do amplificador em 0,7V, evitando que o mesmo seja queimado (lembrar que a tensão de excitação do transdutor é da ordem de 200V pico a pico). A utilização de um resistor elétrico de 1MΩ em série aos diodos é para que, durante a emissão, a menor quantidade de corrente seja desviada do transdutor para os diodos, evitando, assim, perdas significativas de potência para o transdutor.

8.3 Amplificação

Para realizar a amplificação optou-se por utilizar um amplificador simples o TL081 (Texas Instruments, EUA). O circuito utilizado foi o amplificador não inversor, como ilustra a figura a seguir.

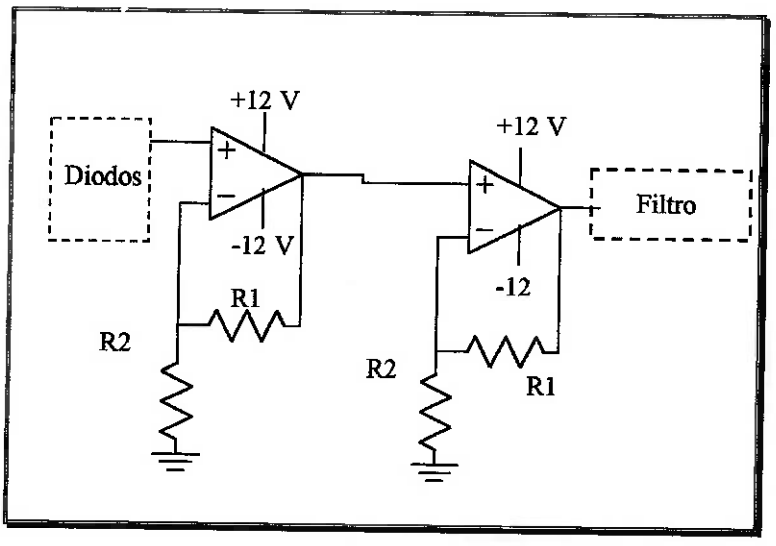


Figura 8.2. – Esquema do Amplificador não inversor.

Através da referência [5], o ganho é dado por:

$$A = 1 + \frac{R_1}{R_2}$$

Como não se sabe exatamente o ganho necessário, ao invés de se soldar diretamente resistores na placa do circuito, foram soldados conectores, a fim de se poder mudar o ganho do amplificador conforme a necessidade, pois devido a perda de intensidade que a onda acústica sofre pela distância percorrida, não se pode saber qual o ganho necessário para que o sinal possa ser percebido pelo detector de limiar no tratamento analógico ou pela digitalização no tratamento digital.

Assim sendo, a idéia seria a imposição de um ganho grande, de modo que possibilitasse o tratamento de sinais a uma distância de algumas dezenas de metros e trabalhasse na saturação do TL081 para pequenas distâncias. Esta idéia porem foi descartada pela distorção no sinal que ocorre com a saturação do amplificador, caso esta ocorra. Este problema é crucial para a medida de distância, pois com a distorção não se pode garantir que a freqüência esperada será satisfeita para o tratamento analógico e que o sinal de referência será observado para o tratamento digital. Assim, é necessário que se ajuste os ganhos dos amplificadores para cada ordem de grandeza que se queira medir.

Outro problema observado na amplificação do sinal de recepção, foi a dificuldade de se obter grandes ganhos (necessários para medições de grandes distâncias) com o tipo de amplificador operacional utilizado. Isto ocorre devido a largura de banda destes componentes, isto é, para a freqüência de excitação utilizada (150 kHz para o veículo móvel e 170 kHz para o alvo fixo) o TL081 não consegue aplicar ao sinal amplificações muito maiores que 26 vezes com muita eficiência. Por isso, optou-se por realizar a amplificação em

duas etapas, para que se possa aplicar amplificações menores que 15 vezes em cada estágio, de modo a se obter uma amplificação final de até 225 vezes.

8.4 Filtro Passa Baixa

Este filtro será o responsável por filtrar os ruídos de alta freqüência tanto para o veículo móvel como para o alvo fixo, portanto, será utilizado um filtro passa baixa. Escolheu-se por utilizar um filtro Butterworth de 4 pólos, apresentando uma atenuação de 80 dB por década, cujo circuito está ilustrado pela figura 8.3.

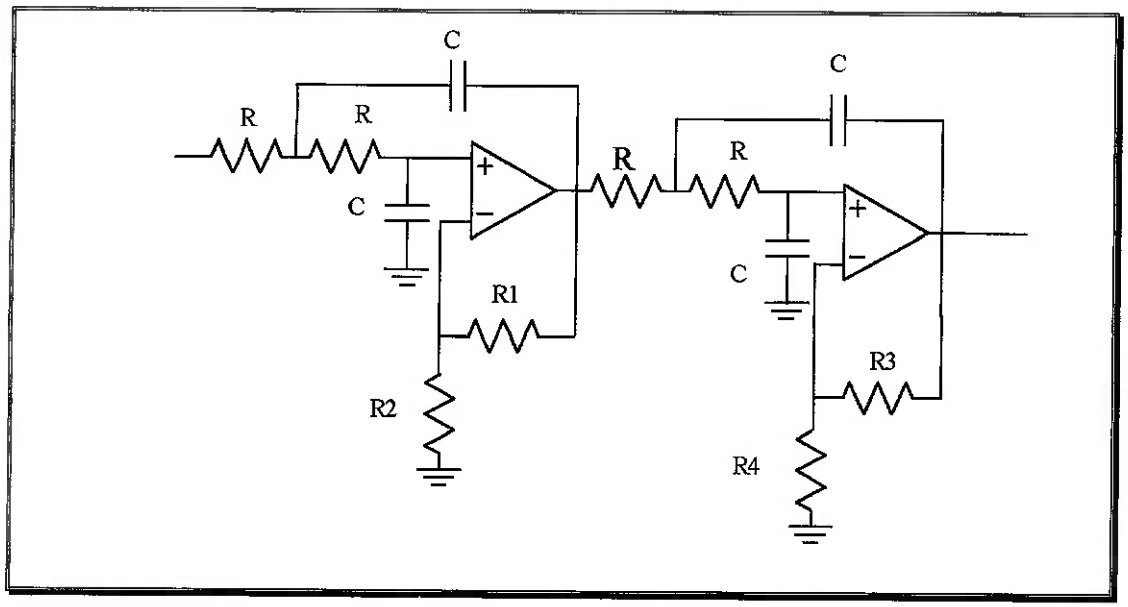


Figura 8.3. – Filtro Butterworth de 4 pólos

Através da tabela de ganhos para filtros Butterworth (ver referência [5]), sabe-se que para o filtro possuir uma resposta maximamente plana o ganho da primeira seção, ou seja, do primeiro filtro deverá ser de A1=1,152 e da segunda seção deverá ser de A2=2,235. Sendo os ganhos são dados por:

$$A_1 = \frac{R_1}{R_2}$$

$$A_2 = \frac{R_3}{R_4}$$

Escolhendo-se R1=1,5 kΩ, R2=10 kΩ, R3=2,7kΩ e R4=2,2 KΩ, obtém-se os seguintes ganhos:

$$A1 = 1 + \frac{1,5}{10} = 1,150$$

$$A2 = 1 + \frac{2,7}{2,2} = 2,227$$

A freqüência de corte é dado por:

$$f_c = \frac{1}{2 \cdot \pi \cdot R \cdot C}$$

Como a faixa de freqüência de operação do transdutor pode variar entre 130 a 170 kHz, escolheu-se que a freqüência de corte do filtro deveria ser em torno de 200 kHz. Escolhendo-se R=390 Ω e C= 2,1 nF, obtém-se:

$$f_c = \frac{1}{2 \cdot \pi \cdot 390 \cdot 2,1 \cdot 10^{-9}} = 194,3 \text{ kHz}$$

A figura 29 ilustra o filtro com os valores dos componentes:

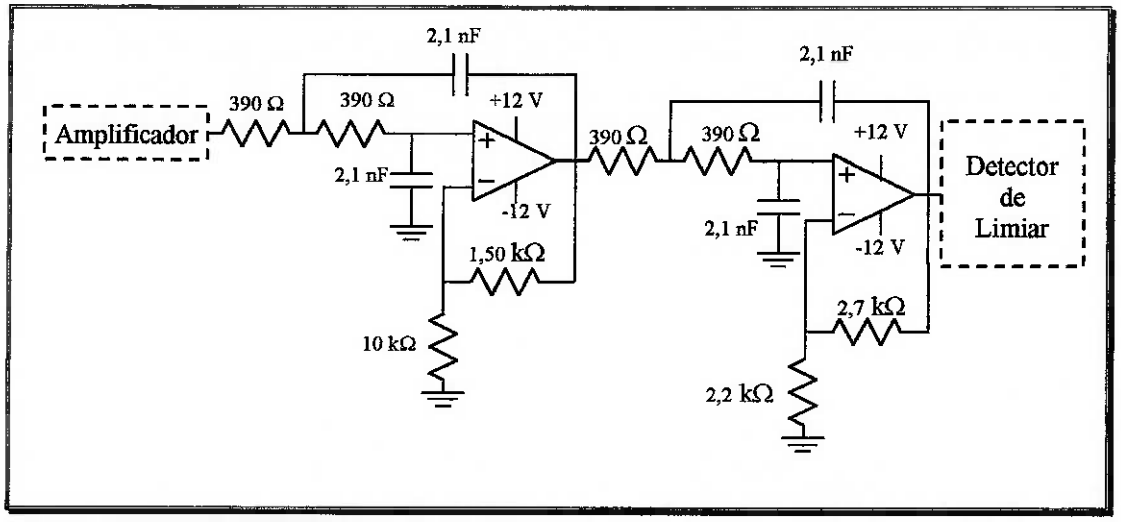


Figura 8.4. – Esquema do Filtro utilizados em ambos os veículos

Levantou-se a curva da resposta em freqüência dos filtro, através de dados experimentais, utilizando-se o gerador de funções. As curvas obtidas para o veículo móvel e o alvo fixo estão ilustradas nas figuras 8.5 e 8.6 respectivamente.

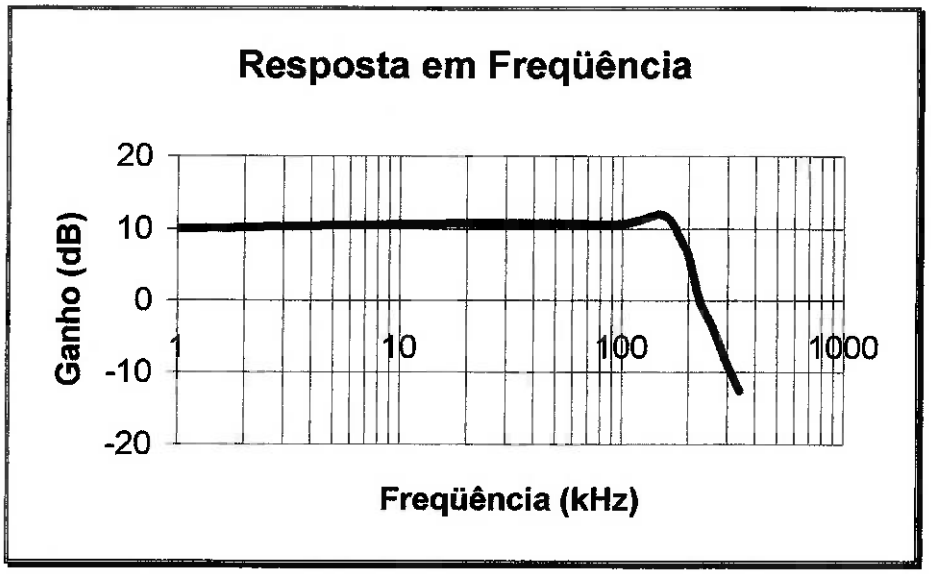


Figura 8.5 - Resposta em freqüência do filtro do veículo móvel

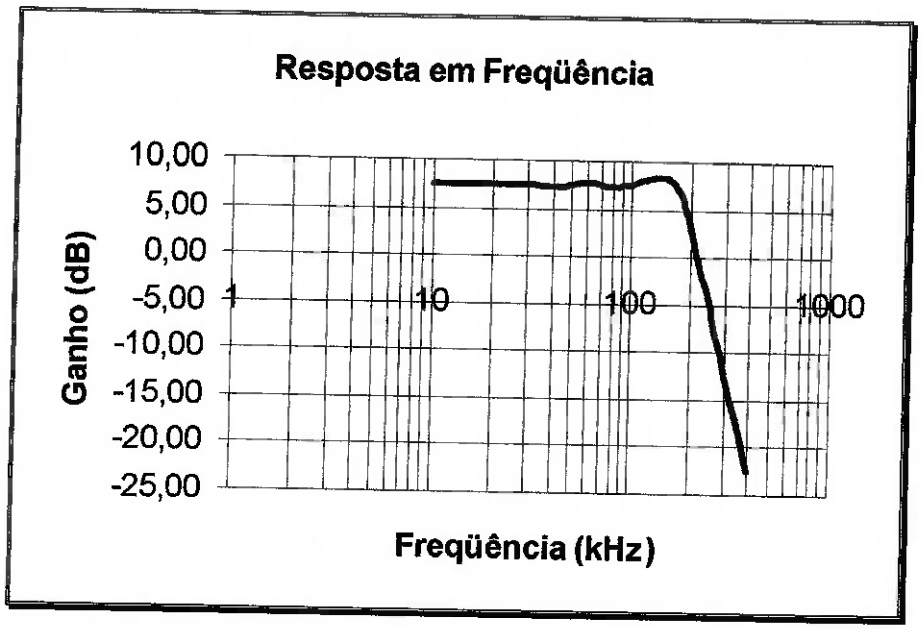


Figura 8.6. - Resposta em freqüência do filtro do alvo fixo

E os dados obtidos foram:

Freqüência	Ganho (dB)
1,00	9,97
10,00	10,58
50,00	10,63
80,00	10,50
100,00	10,58
110,00	10,81
120,00	11,13
130,00	11,43
140,00	11,87
145,00	11,93
150,00	11,95
155,00	11,84
160,00	11,53
165,00	11,27
170,00	10,63
175,00	10,13
180,00	9,25
190,00	7,82
200,00	6,40
225,00	0,00
250,00	-2,97
275,00	-5,85
300,00	-8,87
340,00	-12,58

Tabela 1 Dados obtidos para o filtro do veículo móvel

Freqüência	Ganho (dB)
10,00	7,52
30,00	7,52
40,00	7,25
50,00	7,52
60,00	7,60
70,00	7,43
80,00	7,34
90,00	7,43
100,00	7,52
110,00	7,78
120,00	8,03
130,00	8,20
140,00	8,28
150,00	8,20
155,00	8,12
160,00	7,86
165,00	7,60
170,00	7,15
175,00	6,85
180,00	6,44
185,00	5,80
190,00	4,98
195,00	4,22
200,00	2,92
210,00	1,12
220,00	-0,45
230,00	-2,07
240,00	-3,41
250,00	-4,81
300,00	-12,77
400,00	-22,50

Tabela 2 Dados obtidos para o filtro do alvo fixo

Pelas tabela mostradas acima, percebe-se que a freqüência de corte realmente se encontra por volta 190 kHz para os dois filtros utilizados, como mostravam os cálculos teóricos descritos anteriormente. Assim, conclui-se que os filtros construídos estão de acordo com o projetado.

Outra importante característica desta etapa de filtragem do sinal, é que o sinal ao "passar" pelo filtro sofre um atraso, em relação à saída da etapa de amplificação. Assim, deve-se descontar este atraso no momento de cronometrar o tempo de recebimento da resposta do alvo fixo. O atraso do

na saída do filtro em relação à saída do amplificador está mostrado na figura seguinte:

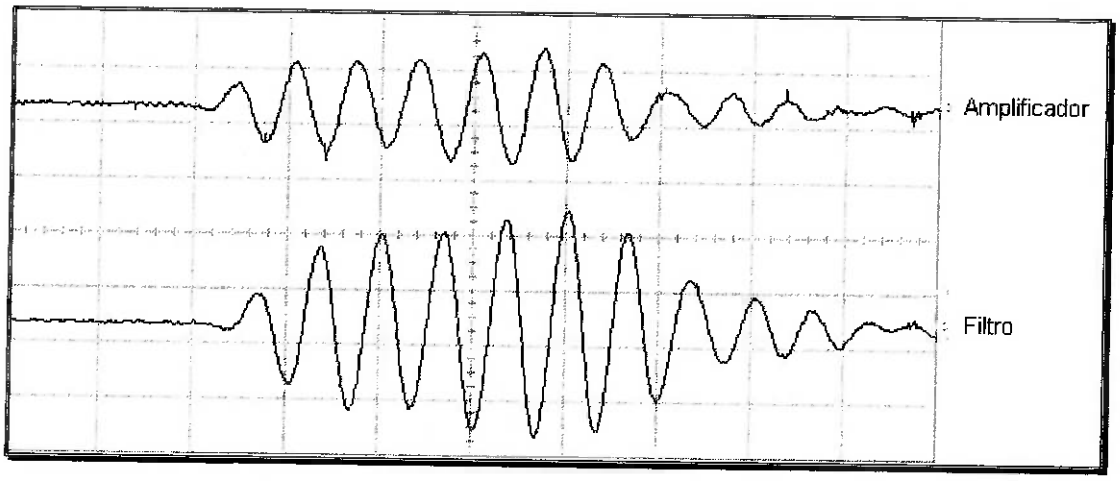


Figura 8.7. - Atraso do sinal na saída do filtro do alvo fixo

8.5 Tratamento do Sinal

8.5.1. Tratamento Analógico (Detector de Limiar)

O detector de limiar só excitará o PIC se a tensão proveniente do filtro for maior que uma tensão mínima estipulada (tensão de referência), isto para evitar que sinais de ruído excitem o mesmo. O circuito do detector de limiar está ilustrado pela figura 8.8.

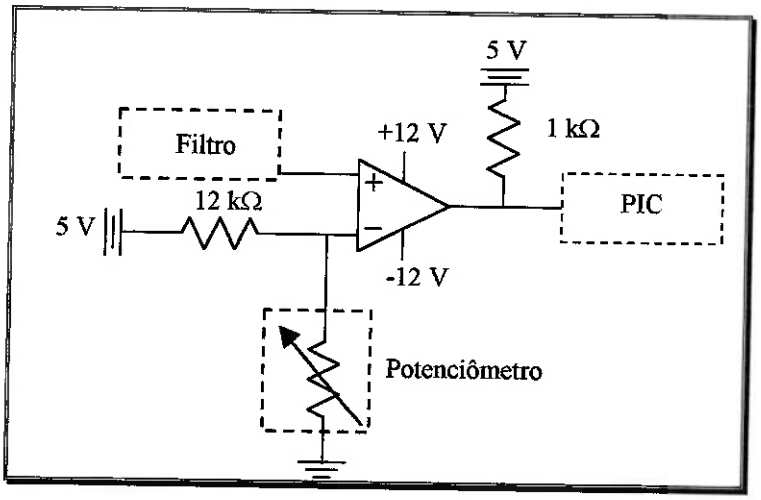


Figura 8.8. – Esquema do detector de Limiar

O circuito funciona da seguinte forma: a tensão proveniente do filtro alimentará o terminal positivo do comparador, já o terminal negativo do mesmo é alimentado pela tensão de referência. Se o primeiro for maior que o segundo a saída do comparador será de 5 V, porém, se for menor a saída será zero.

A tensão de referência é determinada pelo divisor de tensão sendo dada por:

$$V_{ref} = \frac{5}{12 + R_{pot}} \cdot R_{pot}$$

onde R_{pot} é a resistência do potenciômetro em KΩ, que pode variar de 0 a 50.

Optou-se por utilizar o potenciômetro por não se conhecer exatamente qual o valor de tensão que deverá ser utilizado como referência, sendo este a ser determinado experimentalmente.

Assim, se o sinal de recepção, amplificado e filtrado pelas etapas anteriores, excitar o detector de limiar com uma freqüência " f " e amplitude "A", com "A" > "V_{ref}", o comparador de tensão produzirá uma onda quadrada com amplitude de 5 Volts e freqüência " f ".

Então, assim que o sinal excitar o comparador de tensão, este produzirá a onda quadrada, cuja freqüência será analisada pelo PIC, isto para identificar se este sinal é o da resposta do transdutor da outra unidade, ou se é proveniente do próprio eco da emissão desta unidade.

Para melhor entendimento, a figura a seguir ilustra a relação entre a saída do filtro e saída do detector de limiar, mostrando, desta forma, a relação entre as freqüências destes dois sinais. Esta figura foi adquirida pelo circuito do alvo fixo durante um teste de recepção do mesmo:

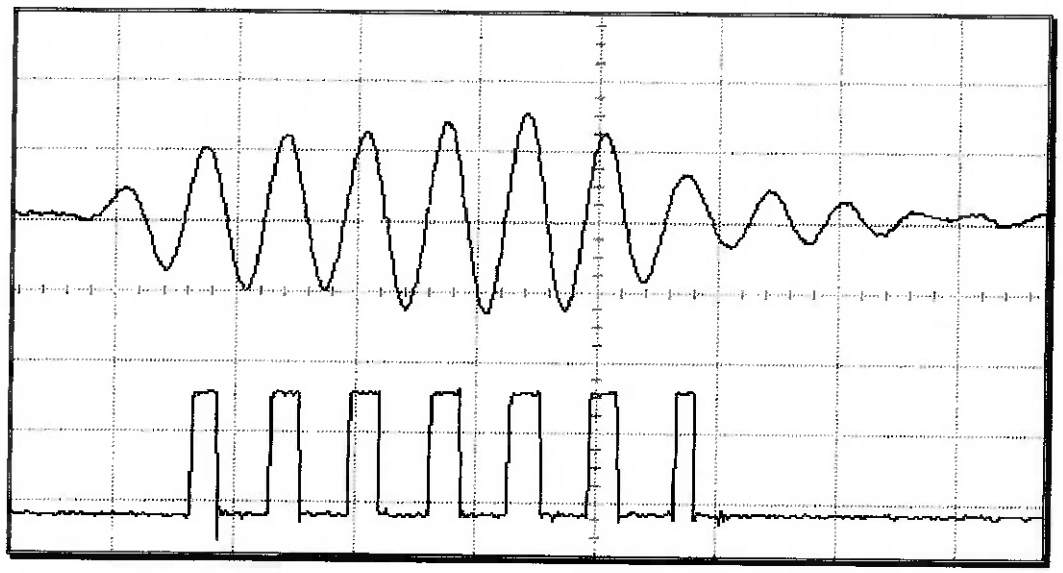


Figura 8.9. - Relação entre os sinais provenientes do filtro e detector de limiar

Um dos grandes problemas do tratamento analógico do sinal, é a não percepção, pelo comparador de tensão, do primeiro pulso do sinal recebido, erro tal que pode causar perdas na precisão da medição de distâncias. Concluindo, há a necessidade de, para uma faixa pequena de distâncias, ajustar o ganho dos amplificadores ou o potenciômetro do detector de limiar, ou ambos, de modo a permitir que o LM311 consiga "perceber" o primeiro pulso do sinal e torne a medição mais correta.

8.5.2. Tratamento Digital (Conversor A/D)

A idéia central, para este tratamento, é a de digitalizar o sinal de saída do filtro, através de um conversor A/D, e realizar um tratamento do mesmo via software a fim de determinar quando o sinal esperado de resposta chegou. Para isto é necessário se obter um sinal de referência, que represente o sinal esperado de resposta, através do qual o software "procurará" onde o sinal de resposta se encontra no sinal digitalizado. A idéia de se utilizar um sinal de referência é valida, pois o formato do sinal de recepção do transdutor sempre será o mesmo, independente da distância entre as unidades, variando apenas a sua amplitude. O sinal de referência será obtido experimentalmente.

Para realizar um teste inicial de todo o projeto, optou-se em realizar o tratamento do sinal digitalizado via PC devido à maior facilidade em desenvolver o software de correlação dos sinais (ver capítulo 10). A figura a seguir ilustra a idéia central do tratamento digital do sinal:

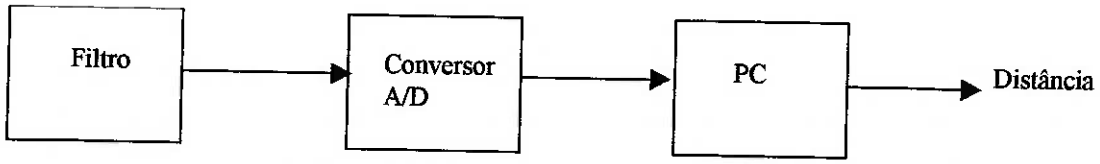


Figura 8.10. – Tratamento digital do sinal

O circuito que realizará a digitalização do sinal da saída do filtro, precisará possuir um conversor analógico digital, uma memória que armazene o sinal digitalizado a ser posteriormente transmitido ao PC e de um clock programável a fim de se poder variar a taxa de amostragem do conversor A/D. O circuito de digitalização é ilustrado na figura a seguir:

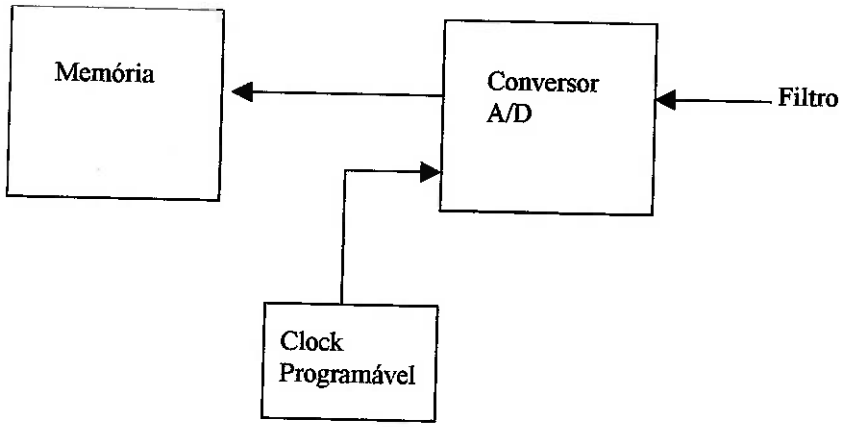


Figura 8.11. – Circuito de digitalização.

O conversor analógico digital escolhido foi o de 12 bits (AD6640) da Analog Devices. O clock deste conversor é fornecido pelo Clock Generator ICD2061A da CYPRESS, cuja freqüência é programável serialmente. Para armazenar os dados digitalizados utilizou-se da memória FIFO CY7C4261 da

CYPRESS de 9 bits e 16 K de memória, portanto, foi necessária a utilização de duas memórias FIFO.

Para controlar o circuito de digitalização escolheu-se utilizar o microcontrolador PIC. Este possuí a função de programar o Clock Generator, de controlar a escrita e a leitura da memória FIFO e enviar os dados para o PC, não possuindo nenhum controle sobre o conversor A/D que funciona o tempo todo. O clock da escrita da FIFO é feito pelo próprio Clock Generator e a habilitação da mesma é realizada pelo PIC. Para realizar a leitura da FIFO é necessário que o PIC habilite a memória para leitura e que produza o clock de leitura. Já para a programação do Clock Generator, o PIC precisa enviar os dados de programação e realizar o clock de leitura destes dados.

Buffers são utilizados para manter a saída das memórias, a fim de permitir que o PIC leia estes dados e os envie serialmente para o PC. O esquema do circuito de digitalização completo é ilustrado pela figura a seguir:

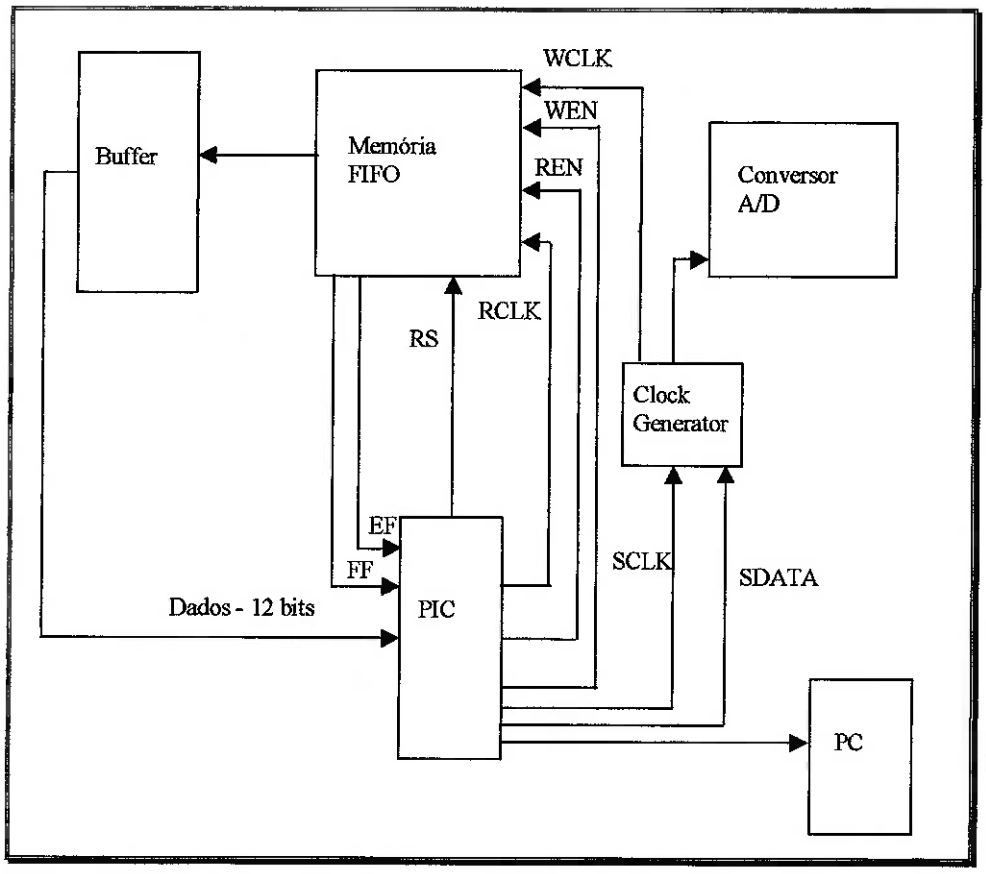


Figura 8.12. – Esquema do circuito de digitalização

Onde :

WCLK: clock da escrita na FIFO;

WEN: habilitação da escrita na FIFO;

REN: habilitação da leitura da FIFO;

RCLCK: clock de leitura da FIFO;

EF: flag que indica se a memória está vazia;

FF: flag que indica se a memória está cheia;

RS: reset da FIFO;

SDATA: envia dados de programação para o Clock Generator;

SCLK: produz o clock de leitura dos dados de programação.

O microcontrolador PIC necessita de 12 pinos de entrada para ler os dados, 2 pinos para ler os flags (EF e FF) da memória, 2 pinos para habilitar leitura e escrita, 1 pino para realizar o clock da leitura, 1 pino para realizar o reset da FIFO, 2 pinos para realizar a comunicação serial com o PC e mais 2 pinos para realizar a programação do clock.

Este PIC além de realizar o controle do circuito de digitalização ele terá que também efetuar a emissão da unidade, ou seja, produzir o trem de pulsos. Desta forma são necessários 23 pinos, porém o PIC 16C73B utilizado possuí apenas 22 pinos I/O (ports A, B e C). Portanto, foi necessário o compartilhamento de pinos, que foi feito através da utilização de dois Buffer e de uma porta inversora, como ilustra a figura a seguir:

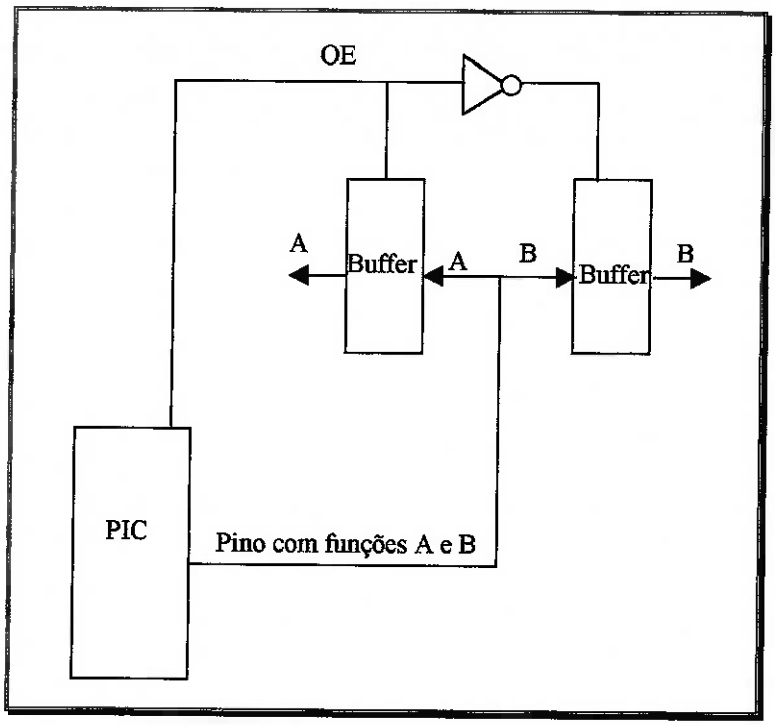


Figura 8.13. – Compartilhamento de pinos do PIC

O compartilhamento funciona da seguinte forma: um pino do PIC, que possui duas funções A e B, alimenta dois Buffer, cujas habilitações são feitas por outro pino (OE), de forma que quando um dos Buffer estiver habilitado o outro estará desabilitado, sendo isto obtido através da utilização da porta inversora.

Os circuitos completos do circuito de digitalização e do circuito responsável pelo seu controle estão no anexo B, bem como a sua conexão com a unidade móvel.

9 Softwares desenvolvidos

Nesta seção será discutida os programas feitos para o PIC e para o PC.

9.1 Programação do PIC

Para a unidade móvel realizaram-se dois tipos de programação, uma para o tratamento digital e outro para o tratamento analógico. Já para a unidade fixa foi realizada apenas a programação para o tratamento analógico. A seguir será explicada a programação dos microcontroladores de cada unidade para os seus diferentes tipos de tratamento.

9.1.1 Programação da unidade móvel com tratamento analógico

O PIC da unidade móvel deve ser capaz de realizar as seguintes operações:

- Inicializar o sistema de emissão, ou seja, deve gerar o trem de pulsos.
- Cronometrar o tempo entre a emissão e o sinal de resposta, que receberá do circuito de recepção.
- Ser capaz de ler o sinal produzido pelo sistema de recepção da unidade móvel e analisar a sua freqüência, a fim de diferenciar o sinal de resposta da unidade fixa com possíveis ecos provenientes da emissão da própria unidade móvel.
- Cálculo da distância, ou seja, ser capaz de realizar a multiplicação entre o tempo cronometrado e a velocidade do som na água.
- Comunicar com o PC (unidade de comando), a fim de receber novas configurações e enviar o valor da distância.

Para cada uma das operações foram escolhidos os periféricos necessários do PIC a fim de realizá-las.

Emissão: Para gerar o trem de pulsos utilizou-se a excitação do pino 2 do PORTB, configurado como saída. O período, a largura do pulso e a duração do trem de pulsos são determinados através de constantes previamente programadas.

Cronometragem: Para realizar a cronometragem utilizou-se o Timer 1 (2 bytes) e quando este estoura, ocorre uma interrupção e um registrador de uso geral de 1 byte (variável) é setado. Portanto, a cronometragem é de 3 bytes, a fim de se obter uma faixa de medida de tempo adequada.

Leitura do sinal de resposta: Para que seja possível a análise da freqüência do sinal de resposta foi utilizado o Timer0 com sua freqüência de clock obtida externamente, ou seja, pelo pino 4 do PORTA, que será a entrada do sinal de resposta.

Multiplicação: Como o PIC não possui instrução de multiplicação, essa operação foi implementada por software através de uma macro que realiza multiplicação de 8 bits por 8 bits. Como a multiplicação é de 24 bits por 24 bits, os números são decompostos em bytes, as operações são feitas e a resposta de 48 bits é composta. Foi decidido usar 3 bytes para armazenar a velocidade do som. Pode ser programada uma velocidade de até 1670 m/s com precisão de 0,1 mm/s.

Comunicação com o PC: Para a comunicação com o PC, foi decidido usar uma comunicação serial assíncrona, portanto, o periférico USART foi utilizado. Foram implementadas macros para a comunicação serial: uma macro para receber dado do PC e armazenar na memória do PIC, uma macro para enviar um byte para o PC e uma macro para enviar o conteúdo de um endereço para o PC.

A listagem do programa implementado para esta função se encontra no anexo E deste trabalho.

9.1.2 Programação da unidade fixa com tratamento analógico

O PIC da unidade fixa deve, ao receber o sinal de recepção, gerar o trem de pulsos. Porém, antes de enviar a resposta, o mesmo deve esperar um certo tempo, isto para evitar situações nas quais a unidade móvel receba a resposta antes de terminar o sinal de emissão, podendo isto ocorrer quando a unidade móvel está muito perto da fixa.

Para realizar a emissão e a análise do sinal de resposta, utilizou-se o mesmo software da unidade móvel com tratamento analógico. Já para realizar o tempo de espera utilizou-se o Timer0 com prescaler de 1:64, ou seja, o tempo de espera para o clock de 20 MHz, é obtido por:

$$t_{espera} = \frac{256 \cdot 16}{\frac{20000000}{4}} = 8{,}19 \cdot 10^{-4}\,s \qquad (33)$$

A listagem do programa implementado para esta função se encontra no anexo F deste trabalho.

9.1.3 Programação da unidade móvel com tratamento digital

O PIC da unidade móvel deve ser capaz de realizar as seguintes operações:

- Programar serialmente o clock da placa A/D.
- Realizar a amostragem do sinal recebido, ou seja, habilitar a escrita na memória FIFO.
- Ser capaz de limpar a FIFO.
- Realizar a emissão, ou seja, a geração do trem de pulsos.
- Realizar a leitura da memória FIFO e transmitir estes dados ao PC.
- Verificar os flags da memória FIFO que indicam se a mesma está cheia (Full Flag) ou vazia (Empty Flag).

- Comunicar com o PC (unidade de comando), a fim de receber novas configurações e enviar o conteúdo da memória FIFO.

Para cada uma das operações foram escolhidos os periféricos necessários do PIC a fim de realizá-las.

Programação do clock: Para este procedimento foram necessários dois pinos, ambos programados como output, um para enviar os dados de programação (pino 0 do PORTA) e outro para ser o clock de leitura destes dados (pino 1 do PORTA).

Escrita na memória FIFO: Como o clock de escrita da FIFO é produzido pelo clock genarator, para realizar a sua escrita é necessária apenas a utilização de um port para habilitar a escrita (pino 2 do PORTA), programado como output.

Limpar a FIFO: Utiliza-se o pino 3 do PORTA como reset da FIFO, sendo programado como output.

Emissão: Para realizá-la utiliza-se o pino 2 do PORTA, programado como output.

Leitura: Para este procedimento foram necessários dois ports, um para habilitar a leitura (pino 0 do PORTA) e outro para realizar o clock de leitura (pino 1 do PORTA), ambos programados como output. Os 12 bits fornecidos pela FIFO são lidos por todos os pinos do PORTB (8 pinos) e pelos pinos 0 a 3 do PORTC, todos programados como input.

Verificar flags: Utiliza-se os pinos 4 e 5 do PORTC, ambos programados como input, para verificarem respectivamente o Empty Flag e o Full Flag da FIFO.

A listagem do programa implementado para esta função se encontra no anexo D deste trabalho.

9.2 Programação do PC

O software utilizado para realizar a interface entre o PC e o PIC foi o Pacific C. A programação da interface do PC com a unidade móvel para os diferentes tipos de tratamentro de sinal, será explicada nos próximos itens.

9.2.1 Programação do PC para tratamento analógico

Este programa realizará a interface entre o PC e o PIC da unidade móvel com tratamento analógico.

O programa do PC é um programa simples. A função dele é receber comandos do usuário, verificar sua validade, enviar os comandos ao PIC e receber respostas. Está preparado para enviar/receber dados das portas COM1, COM2, COM3 e COM4.

O programa do PC e da unidade móvel foram desenvolvidos em conjunto e ambos só funcionam se estiverem rodando ao mesmo tempo e se a comunicação serial estiver ligada corretamente.

Lista de Comandos:

Os comandos reconhecidos pelo PIC são todos enviados pelo PC. A seguir, encontra-se uma lista dos comandos disponíveis ao usuário:

Realizar localização: o PC envia um comando ao PIC. O comando é reconhecido e o microcontrolador gera o trem de pulsos para acionamento do transdutor. A cronometragem do tempo começa e a unidade móvel se prepara para receber o sinal de resposta. Se a resposta da unidade fixa é recebida, a distância é calculada e enviada ao PC, que imprime a resposta na tela

Modo de teste: através desse modo o PIC fica enviando trens de pulsos continuamente, existindo um espaçamento entre eles. Este modo é utilizado para que se possa analisar, por exemplo, o sinal de resposta da unidade fixa.

Reinicializar comunicação: se por algum motivo o PIC é resetado durante a execução do programa da unidade de comando ou o programa da unidade de comando é reiniciado quando o programa do PIC já está rodando,

esse comando permite o reestabelecimento da comunicação entre o PIC e o PC.

Verificar Parâmetros programados: esse comando é enviado ao PIC, que envia os valores programados na sua memória de velocidade, largura de pulso, período dos pulsos e duração do trem de pulsos.

Programar velocidade: esse comando permite que um novo valor de velocidade do som seja programado no PIC. Quando o PIC é inicializado, o valor default de 1500m/s é armazenado na memória. Esse valor pode ser modificado através desse comando.

Programar largura de pulso: esse comando permite que um novo valor de largura de pulso seja programado no PIC.

Programar período dos pulsos: esse comando permite que um novo valor de período seja programado no PIC.

Programar duração do trem de pulsos: esse comando permite que um novo valor de duração do trem de pulsos seja programado no PIC.

Ajuda: esse comando exibe uma lista dos comandos e sua sintaxe.

Finalizar programa: termina a execução do programa da unidade de comando. O programa da unidade móvel, no entanto, permanece rodando.

A listagem do programa implementado para esta função se encontra no anexo G deste trabalho.

9.2.2 Programação do PC para tratamento digital

Este programa realizará a interface entre o PC e o PIC da unidade móvel com tratamento digital.

O programa é similar ao anterior, mudando apenas alguns comandos, que serão comentados a seguir:

Lista de Comandos:

Os comandos reconhecidos pelo PIC são todos enviados pelo PC. A seguir, encontra-se uma lista dos comandos disponíveis ao usuário:

Programar clock: o PC envia a palavra de programação, obtida pelo software BITCALC, e o microcontrolador programa serialmente o clock.

Realizar localização: o PC envia um comando ao PIC. O comando é reconhecido e o microcontrolador gera o trem de pulsos para acionamento do transdutor. Logo em seguida o PIC escreve na memória FIFO até a mesma encher.

Limpar FIFO: permite resetar a memória FIFO.

Ler memória FIFO: permite ler os dados armazenados na memória e transmiti-las para o PC. Convém mencionar, que para não existir erro de leitura por parte do PC, realizou-se o seguinte protocolo: para cada dado enviado pelo PIC o PC responde se recebeu ou não corretamente, se o primeiro ocorrer o PIC envia outro dado, porém, se o segundo ocorrer o PIC envia novamente o mesmo dado enviado anteriormente.

Modo de teste: idem ao programa anterior.

Reinicializar comunicação: idem ao programa anterior.

Verificar Parâmetros programados: esse comando é enviado ao PIC, que retorna os valores programados da freqüência do clock, do status da FIFO (cheia, vazia, não cheia e erro).

Ajuda: esse comando exibe uma lista dos comandos e sua sintaxe.

Finalizar programa: termina a execução do programa da unidade de comando. O programa da unidade móvel, no entanto, permanece rodando.

A listagem do programa implementado para esta função se encontra no anexo H deste trabalho.

10. O Algoritmo da Correlação

Os estudos realizados para entendimento da teoria contida neste capítulo, foram baseados na referência [4].

Este algoritmo tem por finalidade a determinação do início de uma onda de referência, com um determinada freqüência, que se encontra em algum lugar de um sinal amostrado.

O algoritmo se baseia na Transformada de Fourier, cujo conceito está explicado no próximo item.

10.1. Transformada de Fourier para Amostragens Discretas

Nas comuns situações, a transformada de Fourier deve ser aplicada a uma função h(t), amostrada por algum meio de aquisição de dados, como é o caso do tratamento digital realizado neste trabalho. Assim, conclui-se que os pontos amostrados são adquiridos a uma determinada taxa de amostragem (Δ), que deve ser constante entre dois pontos amostrados consecutivos.

Pelo teorema de Nyquist, a freqüência de amostragem (f_c) do sinal deve ser de no mínimo 2 vezes a freqüência do sinal a ser amostrado, para que se possa evitar o aliasing da amostra.

Definidos estes conceitos, supondo que há uma função de amostragem dada por:

$$h_k \equiv h(t_k), \text{ onde } t_k \equiv k\Delta, \text{ e } k = 0,1,2,K\ ,N-1 \qquad (34)$$

a transformada de Fourier para os N pontos amostrados, seria:

$$H_n = \sum_{k=0}^{N-1} h_k \cdot e^{\frac{2\pi i k n}{N}} \qquad (35)$$

onde o índice "n" varia de $-\frac{N}{2}$ até $\frac{N}{2}$, já que a freqüência da transformada, dada por:

$$f_n \equiv \frac{n}{N\Delta}, \qquad (36)$$

deve seguir o teorema de Nyquist, limitando o valor de "n" entre os valores acima especificados.

A partir da equação (35), define-se a transformada inversa de Fourier como:

$$h_k = \frac{1}{N}\sum_{n=0}^{N-1} H_n \cdot e^{-\frac{2\pi i k n}{N}} \qquad (37)$$

10.2. A Transformada Rápida de Fourier (FFT)

A partir da formulação mostrada no item anterior, definiu-se um algoritmo para o cálculo das transformadas de Fourier por meio de computadores. O principal meio de produzir este cálculo rapidamente, é por meio da Transformada Rápida de Fourier.

Este algoritmo pode ser explicado da seguinte forma:

Primeiramente, dividindo-se a equação (35) em dois termos:

$$F_k = \sum_{j=0}^{N-1}\left(e^{-\frac{2\pi i j k}{N}}\right) f_j =$$

$$= \sum_{j=0}^{\frac{N}{2}-1}\left(e^{-\frac{2\pi i k(2j)}{N}}\right) f_{2j} + \sum_{j=0}^{\frac{N}{2}-1}\left(e^{-\frac{2\pi i k(2j-1)}{N}}\right) f_{2j+1} \Rightarrow \qquad (38)$$

$$\Rightarrow F_k = F_k^e + W^k \cdot F_k^o$$

onde F_k^e e F_k^o são as parcelas pares e ímpares e "W" é definido por:

$$W = e^{-\frac{2\pi i}{N}} \qquad (39)$$

Assim, ainda pose-se dividir cada termo da equação (38) por mais dois termos, usando a transformada para $\frac{N}{4}$. A grande vantagem deste método é a possibilidade do cálculo recursivo.

10.3. Correlação Usando FFT

A correlação consiste num método de se comparar duas funções distintas mas representadas por um mesmo tipo de dado. A correlação é verificada, sobrepondo-se as duas funções, com uma delas se "movimentando" sobre a outra.

A correlação entre duas funções g(t) e h(t) pode ser denotada como:

$$Corr(g,h)(t) = Corr(h,g)(-t) \qquad (40)$$

O valor da correlação é maior no instante de tempo onde a função g(t) é próxima de uma cópia de h(t)

A correlação de funções discretas g_k e h_k, ambas com período n, é dada por:

$$Corr(g,h)_j \equiv \sum_{k=0}^{N-1} g_j + kh_k \qquad (41)$$

Esta correlação pode ser escrita em termos da transformada discreta de Fourier, de modo que tem-se:

$$Corr(g,h)_j \equiv G_k \cdot H_k^* \qquad (42)$$

onde G_k e H_k são as transformadas de Fourier de g_i e h_i, e o asterisco denota conjugado da função.

Concluindo, pode-se computar a correlação entre duas amostras, utilizando FFT. O procedimento para isto seria a obtenção da Transformada Rápida das duas funções, a multiplicação da transformada de uma pelo conjugado da transformada da outra e a posterior obtenção da transformada inversa deste produto. Assim, a partir dos valores da correlação obtidos para cada ponto da função discretizada g_i, conclui-se que o ponto com o maior valor da correlação é o ponto onde a sobreposição desta onda com a função h_i obteve melhor resultado.

10.4. Programa Implementado

Para se fazer o tratamento digital do sinal de resposta do alvo fixo, como já explicado anteriormente, foi implementado um programa que realiza a correlação entre o sinal recebido pelo o veículo móvel e o sinal de referência da resposta do alvo fixo. Este programa foi feito pelo software MATLAB, de forma que a determinação do ponto de máximo valor da correlação foi feito a partir de um interpolação por um função do segundo grau dos três pontos de maior valor da correlação. A listagem deste programa esta contida no anexo I deste trabalho.

11. Testes Realizados

Afim de testar os circuitos do veículo móvel e do alvo fixo, bem como analisar a acurácia e precisão dos métodos digital e analógico de tratamento de sinais, vários testes foram realizados, sendo que os projetos dos circuitos utilizados para estes testes (realizados em um tanque com dimensões bem definidas), juntamente como os ganhos utilizados nas etapas de amplificação, estão mostrados nos anexos A, B e C.

A determinação da acurácia do circuito analógico é muito importante, visto que não foi possível realizar o tratamento digital no circuito do alvo fixo. Deste modo, erros como o mostrado na figura 8.9., onde um pulso é "perdido" pelo detector de limiar, podem ocorrer na medição da distância.

11.1 Teste dos circuitos de recepção

Para se realizar este teste, primeiramente, colocou-se a unidade móvel a emitir sinais seguidamente, segundo um intervalo de tempo pré - determinado, de modo a se poder adquirir dados do circuito de recepção do alvo fixo. Assim sendo, os sinais recebidos estão mostrados nas figuras 11.1., 11.2., 11.3. para a saída do segundo estágio da etapa de amplificação, o filtro e o detector de limiar respectivamente:

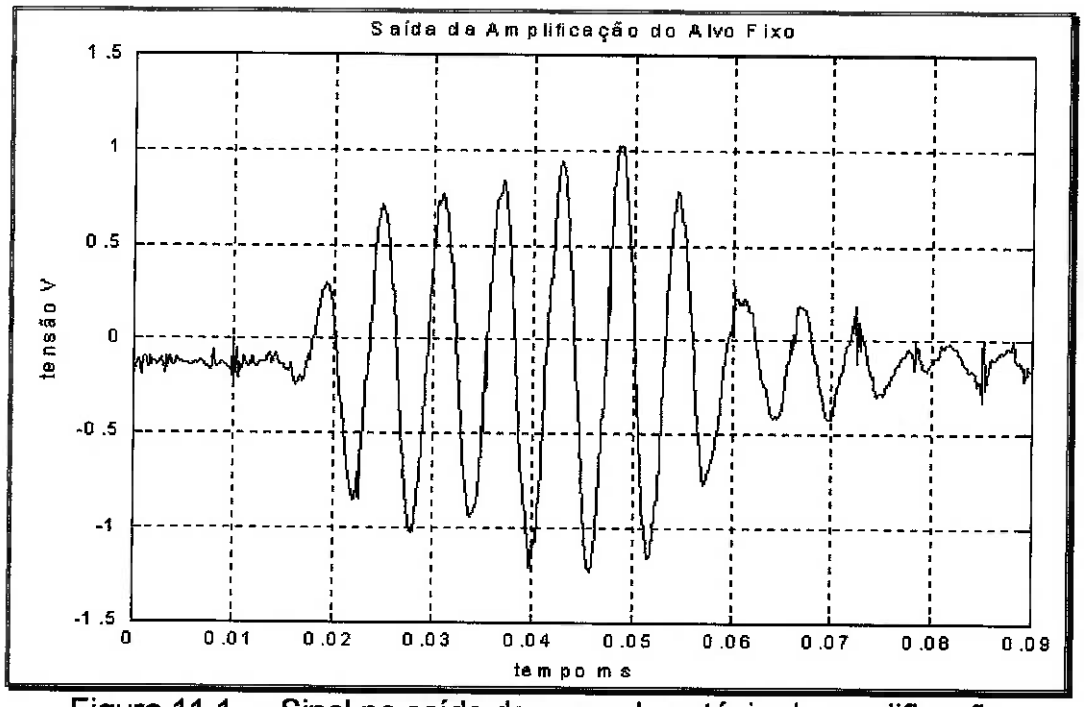


Figura 11.1. - Sinal na saída do segundo estágio de amplificação

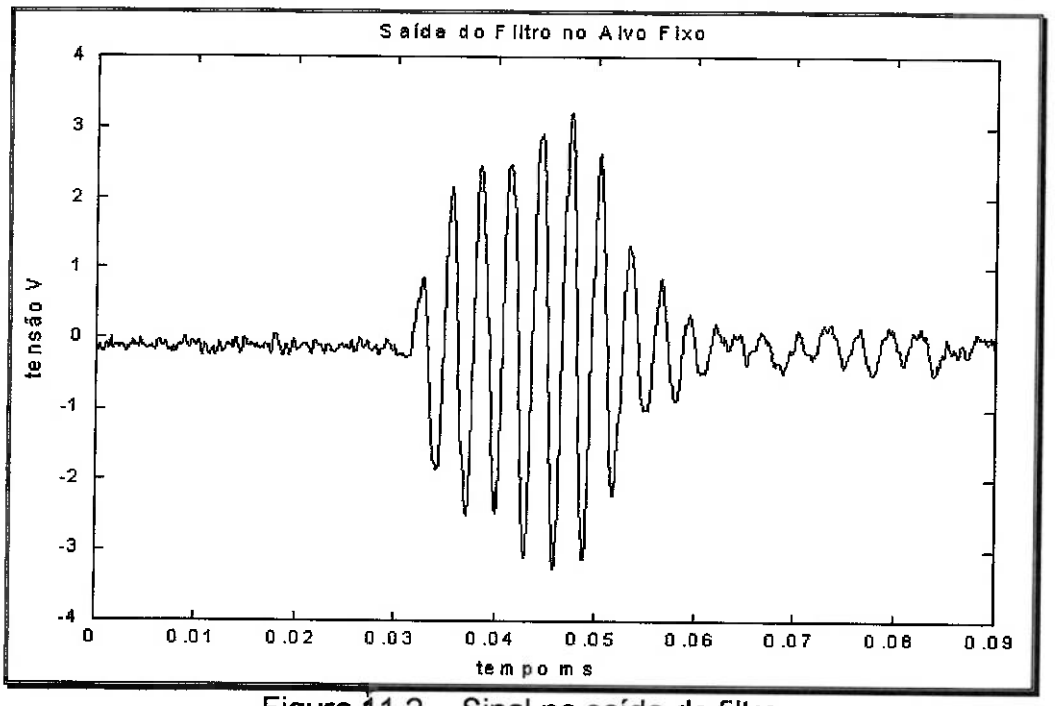


Figura 11.2. - Sinal na saída do filtro

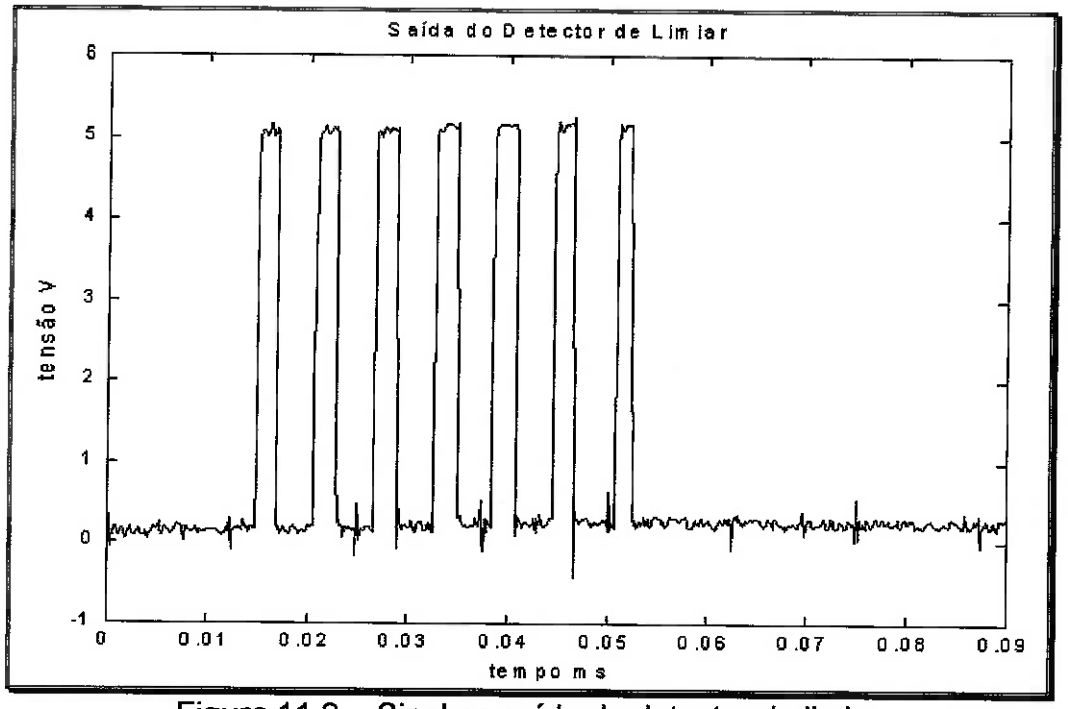


Figura 11.3. - Sinal na saída do detector de limiar

Feito isso, preparou-se o teste para o circuito de recepção do veículo móvel. Para isso, colocou-se um gerador de funções no lugar do comparador de tensão da recepção do alvo fixo, de modo que o circuito perceba a existência de um sinal de resposta à uma freqüência regulada pelo gerador de funções, que estará dentro da faixa esperada pelo controlador deste circuito. Assim sendo, os sinais amostrados no amplificador, filtro e detector de limiar do veículo móvel, estão mostrados nas figuras a seguir:

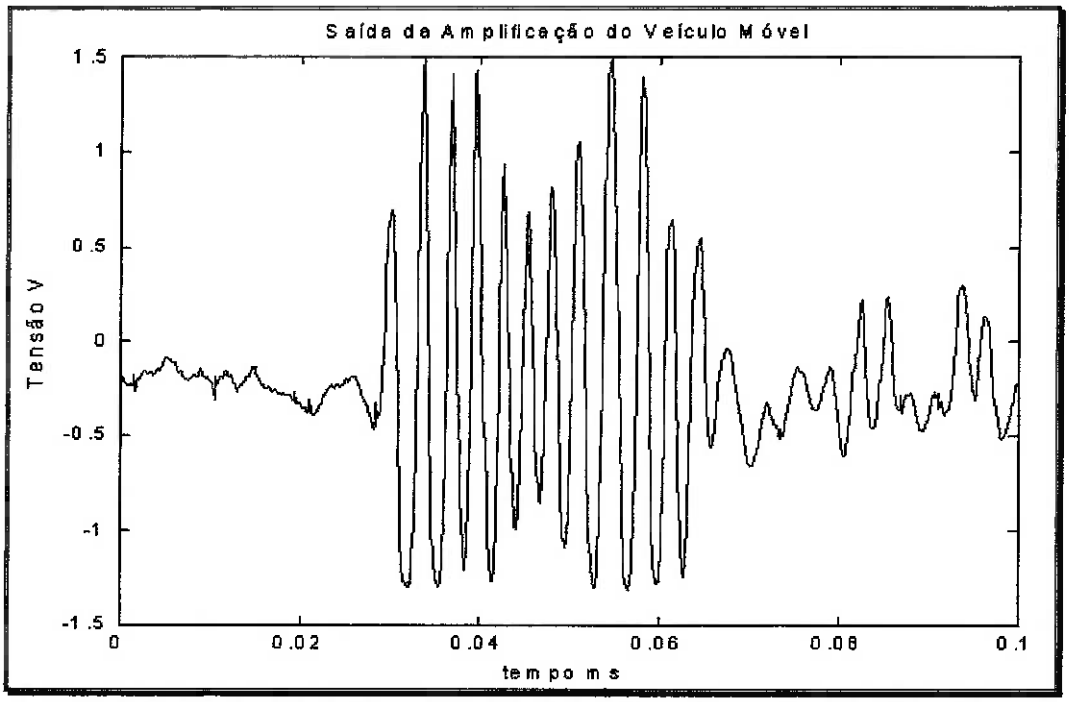


Figura 11.4. - Sinal na saída do segundo estágio de amplificação

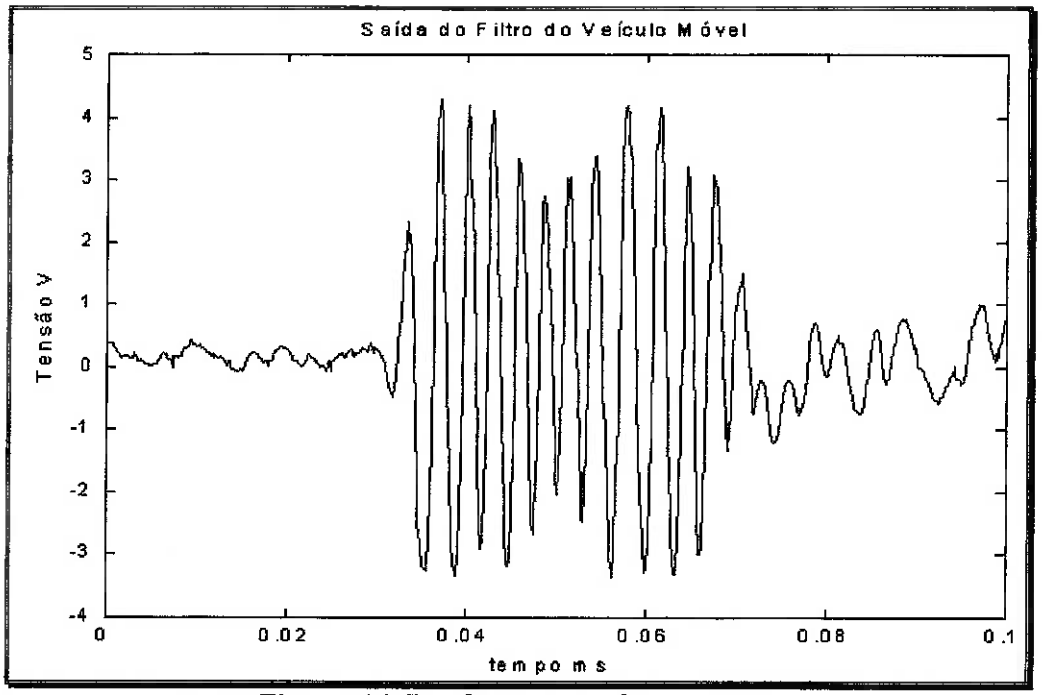


Figura 11.5. - Sinal na saída do filtro

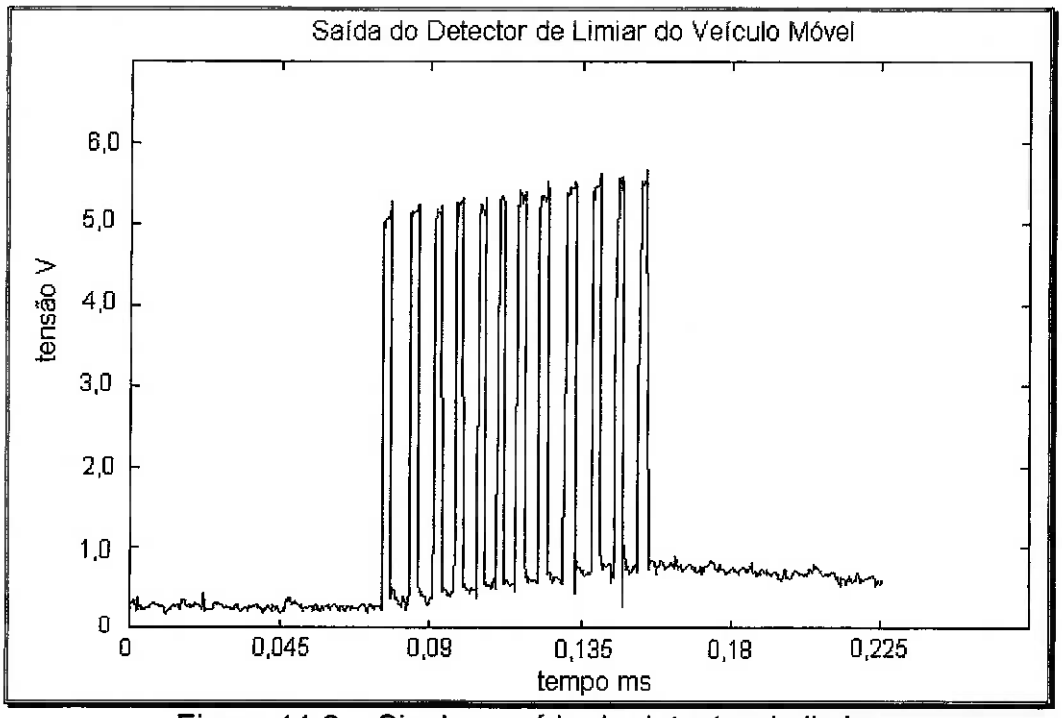


Figura 11.6. - Sinal na saída do detector de limiar

11.2 Testes do tratamento digital

Para averiguar a acurácia e precisão do tratamento de sinal digital no cálculo da distância, realizou-se um experimento, no qual os dois transdutores conectados as suas respectivas placas foram postos submergidos em lados opostos de um reservatório de água de 47 cm de comprimento, 32 cm de largura e 26 cm de profundidade. A disposição do experimento está esquematizada na figura a seguir:

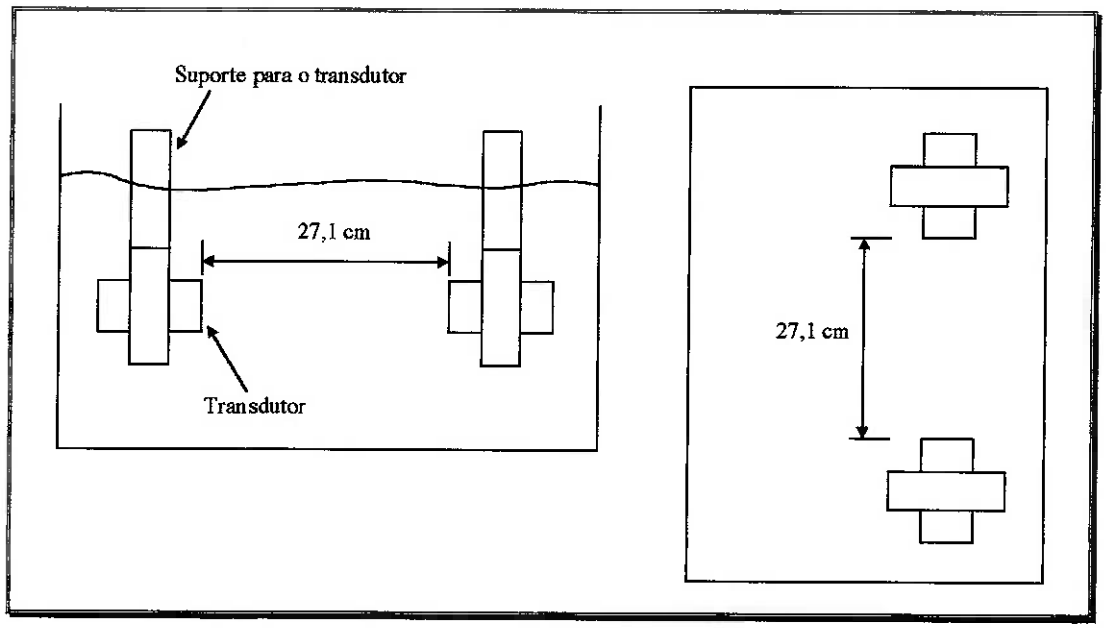


Figura 11.7. - Esquema da disposição dos transdutores no experimento

Convém mencionar que para a realização deste experimento, apenas a unidade móvel possuí tratamento de sinal digital, sendo o tratamento da unidade fixa analógico. Desta forma, para que a determinação da acurácia do tratamento de sinal digital não seja comprometida pelo tratamento analógico, os ganhos dos amplificadores bem como a tensão de referência do detector de limiar foram ajustados, de forma que o erro ilustrado pela figura 8.9 não ocorra.

O primeiro passo para realizar o teste, foi a obtenção de um onda de referência para que se possa realizar o algoritmo da correlação no tratamento do sinal amostrado pelo conversor A/D. Para isto, induziu-se o alvo fixo a emitir o sinal de resposta utilizando-se um gerador de função, que cria uma onda quadrada com freqüência entre 130 kHz e 155 kHz, que nada mais é que a faixa onde estará contida a freqüência do sinal emitido pelo veículo móvel. A unidade móvel ficou amostrando o sinal de resposta, e a partir desta amostragem, selecionou-se experimentalmente, os pontos que representam a recepção deste sinal. Concluindo, este conjunto de pontos serão a referência para que se possa realizar o algoritmo da correlação. O sinal de referência amostrado foi:

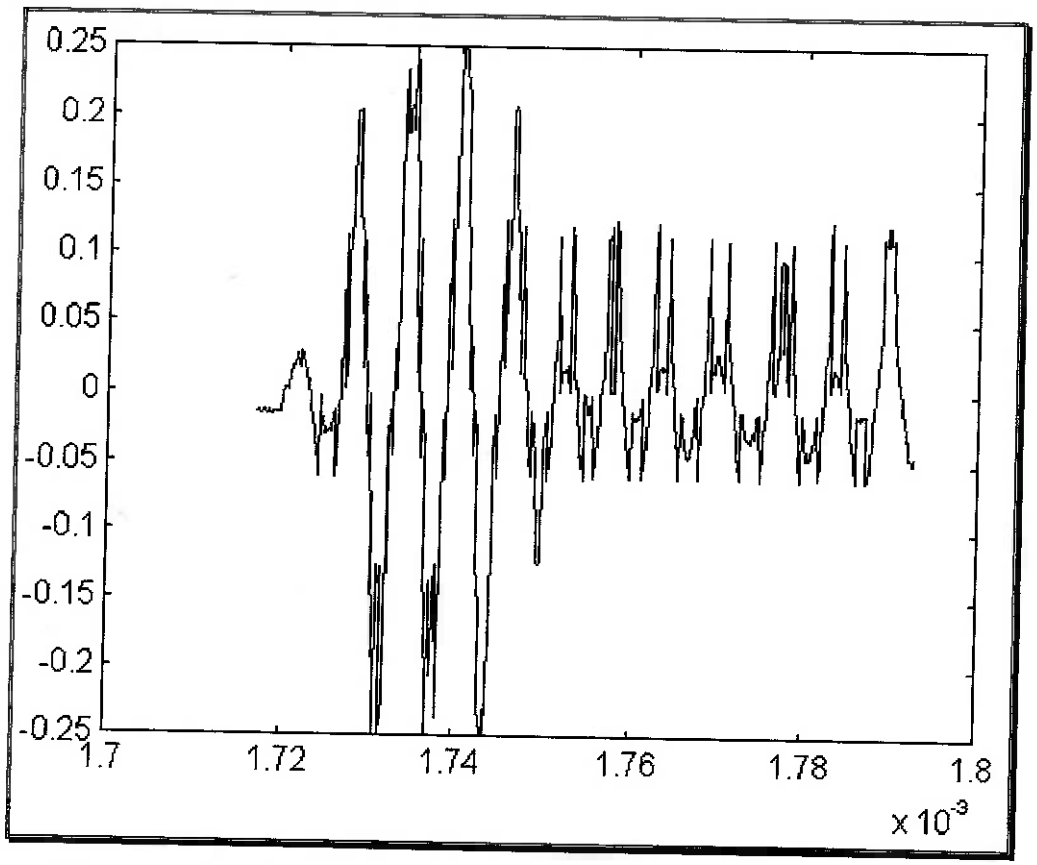


Figura 11.8. – Sinal de referência para o algoritmo da correlação

Feito isso, o próximo passo foi a determinação da velocidade do som na água que preenche o reservatório, submetidas às condições do local. Para isto, realizou-se uma medida de distância, apenas com o intuito de amostrar os sinais de retorno no transdutor do veículo móvel. Assim, o que se observou foi:

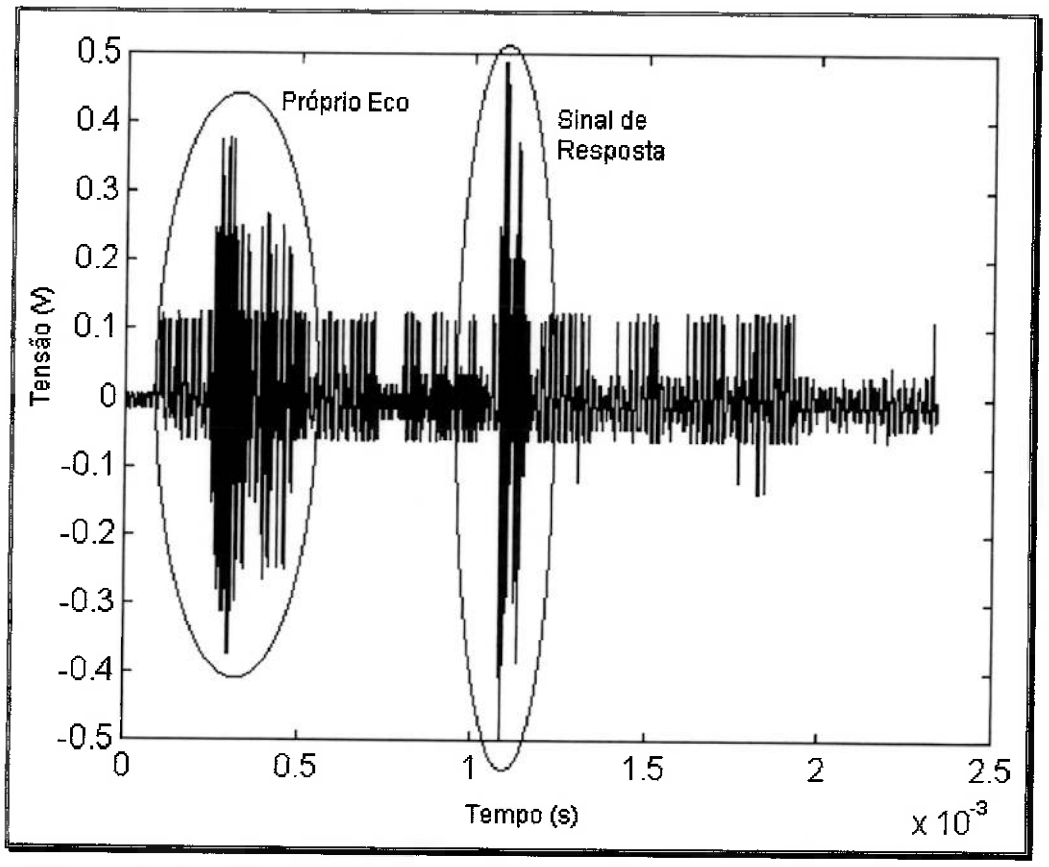


Figura 11.9. - Sinal amostrado no veículo móvel na disposição inicial

Assim, observa-se que antes da resposta emitida pelo alvo fixo, existem ecos, provenientes da própria emissão do sinal do veículo móvel. Observando-se o gráfico, verificou-se que o tempo do início da amostragem e a percepção do primeiro eco (que seria da reflexão da onça acústica no transdutor do alvo fixo) foi de 0,22 ms. Porém, o tempo entre o início da amostragem e o início da emissão pelo veículo móvel é de 0,155 ms (tempo programado), portanto o tempo entre a emissão do sinal e a recepção do próprio eco foi de 0,375 ms. Sabendo-se que a distância entre os transdutores é de vinte e seis centímetros, concluí-se que a velocidade do som na água é de 1400m/s.

Possuindo a velocidade do som na água e o sinal de referência, agora é possível realizar o cálculo da distância a partir dos sinais amostrados. Para a disposição indicada pela figura 11.7, foram realizadas dez amostragens, de modo a verificar a repetibilidade e acurácia do projeto. Os resultados estão contidos na tabela a seguir:

Medida	Distância (m)
1	0,2866
2	0,2688
3	0,2689
4	0,2687
5	0,2691
6	0,2691
7	0,2690
8	0,2690
9	0,2689
10	0,2687
Média	0,2689
Desvio	0,0002

Tabela 3 – Distâncias medidas para o primeiro experimento

Agora, utilizando-se a mesma velocidade do som calculada anteriormente, realizou-se uma nova medida de distância, porém com uma mudança na disposição dos transdutores, que está mostrada na figura a seguir:

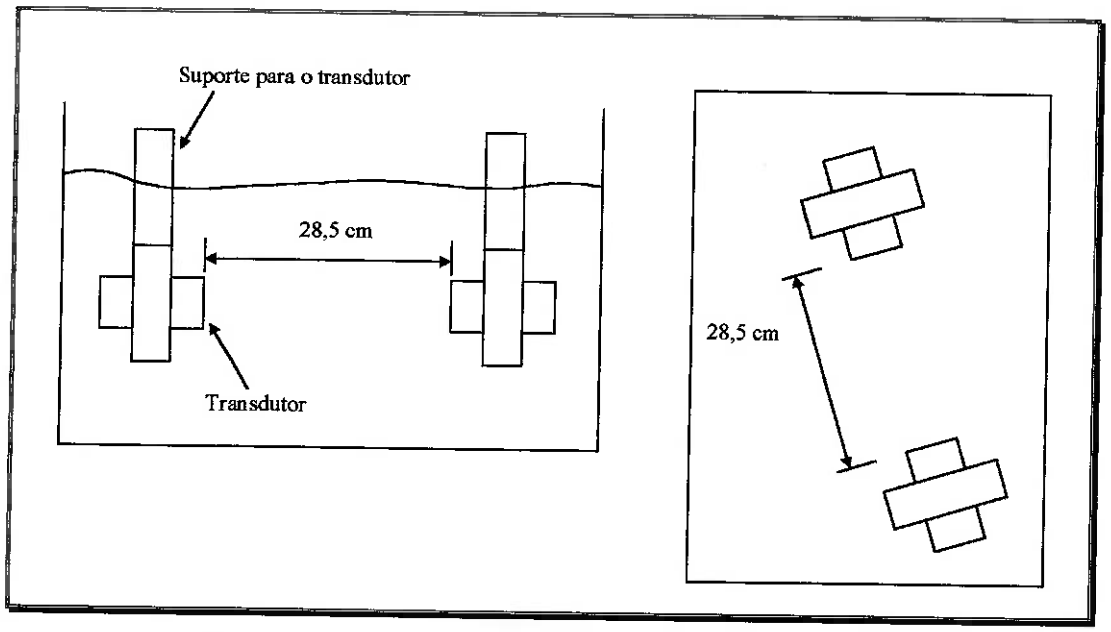


Figura 11.9. - Disposição dos transdutores para o novo experimento

Assim, o sinal amostrado foi o seguinte:

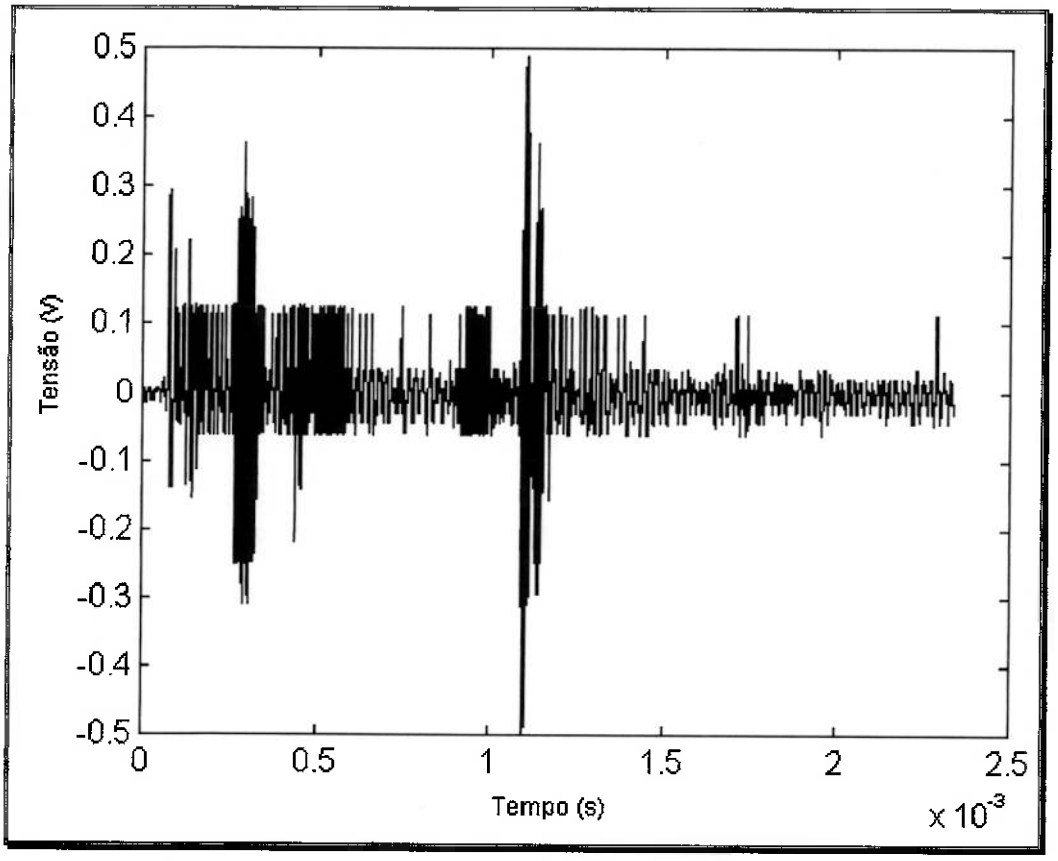


Figura 11.10. - Sinal amostrado no segundo experimento

E aplicando-se o mesmo algoritmo do primeiro experimento, com a mesma onda de referência, obteve-se as seguintes medidas:

Medida	Distância (m)
1	0,2777
2	0,2770
3	0,2774
4	0,2776
5	0,2776
6	0,2778
7	0,2777
8	0,2775
9	0,2773
10	0,2770
Média	0,2774
Desvio	0,0003

Tabela 4 – Distâncias medidas para o segundo experimento

Agora, uma outra medida foi realizada, com os transdutores posicionados segundo a figura a seguir:

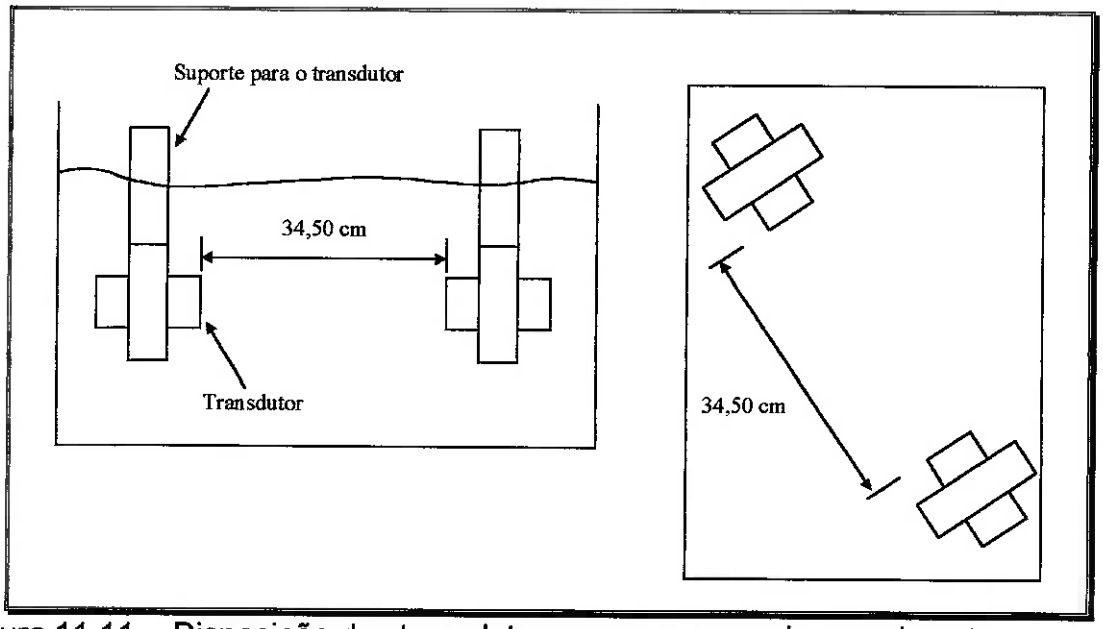


Figura 11.11. - Disposição dos transdutores para o segundo experimento

Assim, o sinal amostrado foi o seguinte:

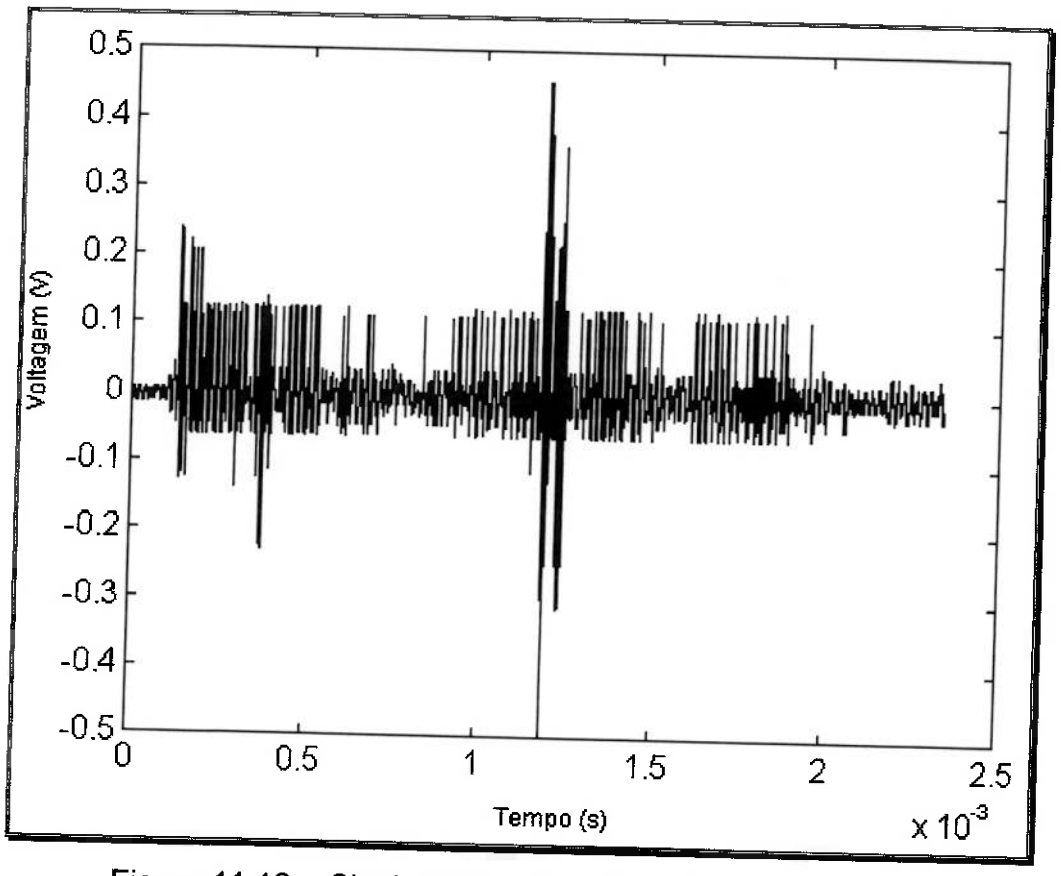


Figura 11.12. - Sinal amostrado no terceiro experimento7

E, aplicando-se o mesmo procedimento, obteve-se os seguintes dados:

Medida	Distância (m)
1	0,3383
2	0,3387
3	0,3391
4	0,3390
5	0,3388
6	0,3388
7	0,3388
8	0,3389
Média	0,3389
Desvio	0,0001

Tabela 5 – Distâncias medidas para o terceiro experimento

Observa-se que a distância calculada, para os experimentos realizados, está muito próxima da real, indicando que o sistema possui uma

boa acurácia. Já o pequeno valor de desvio, na ordem de 2 a 3 décimos de milímetro, indica uma boa repetibilidade para o mesmo.

11.3 Testes do tratamento analógico

Colocando o veículo móvel para realizar a medida de distância pelo modo analógico, foram feitas várias medidas, com os transdutores dispostos segundo a figura a seguir:

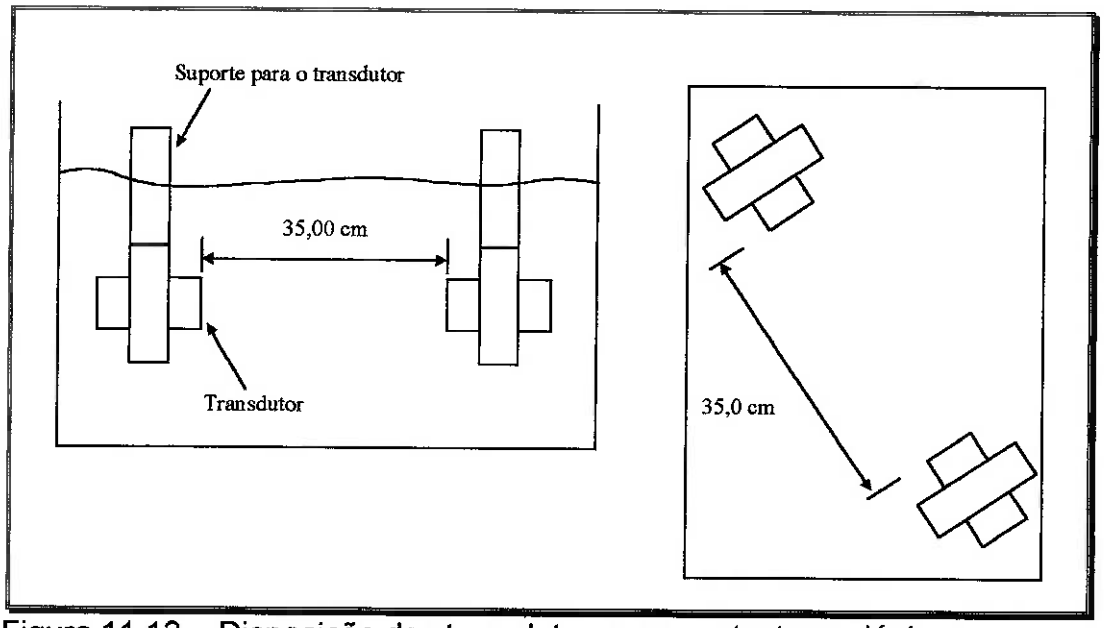


Figura 11.13. - Disposição dos transdutores para o teste analógico

E assim sendo, os dados obtidos foram:

Medida	Saída em Hexadecimal	Distância (m)
1	09e3015c00	0,3963
2	0a90746f80	0,4235
3	09e3015c00	0,3963
4	0b2deac1c0	0,4481
5	09dea41180	0,3956
6	09e2379880	0,3962
7	0b214b4b00	0,4461
8	0bba9b9a80	0,4701
9	0b3411cfc0	0,4491
10	0acd8b43c0	0,4331

Média	0,4254
Desvio	0,0279

Tabela 6 - Distâncias medidas para o teste analógico

Outro teste realizado, foi o seguinte:

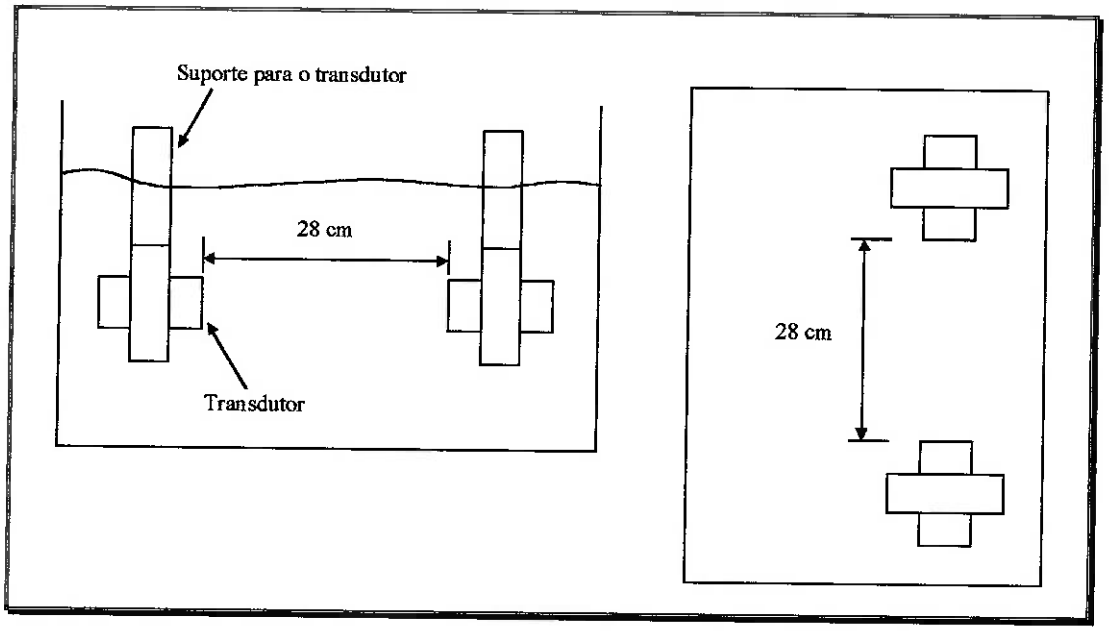


Figura 11.14. – Disposição dos transdutores para o segundo teste analógico

E os resultados obtidos foram:

Medida	Saída em Hexadecimal	Distância (m)
1	0871dc4780	0,3385
2	0813fbafc0	0,3238
3	0813fbafc0	0,3238
4	08e34d2780	0,3563
5	086e2da240	0,3380
6	0870f765c0	0,3383
7	08748aecc0	0,3389
8	088d787f80	0,3428
9	081958fa40	0,3246
10	08756fce80	0,3391

Média	0,3364
Desvio	0,0101

Tabela 7 – Distâncias medidas para o segundo teste analógico

Pelos resultados obtidos, e observações no osciloscópio, observou-se que a verificação da freqüência do sinal realizada pelo PIC pode trazer erros para a medida da distância. Isto ocorre, pela maneira pela qual o PIC faz esta verificação, que consiste na cronometragem do tempo necessário para que o sinal em questão tenha quatro bordas de subida. Como a onda de resposta analisada no circuito do veículo móvel tem um formato ruim (ver figuras 11.4. e 11.5.), é provável que haja distorções na freqüência de alguns trechos do sinal fornecido pelo detector de limiar. Assim, através de observações com o osciloscópio, constatou-se que isto realmente ocorreu para os dois experimentos, acarretando um erro de 3,25 cm, devido a não consideração de cinco bordas de subida (quatro períodos) do sinal recebido. Deste modo, corrigindo-se este erro, os resultados seriam os seguintes:

Medida	Distância (m)
1	0,3638
2	0,3910
3	0,3638
4	0,4156
5	0,3631
6	0,3637
7	0,4136
8	0,4376
9	0,4166
10	0,4006
Média	0,3929
Desvio	0,0279

Tabela 8 – Distâncias corrigidas para o primeiro teste analógico

Medida	Distância (m)
1	0,3060
2	0,2913
3	0,2913
4	0,3238
5	0,3055
6	0,3058
7	0,3064
8	0,3103
9	0,2921
10	0,3066
Média	0,3039
Desvio	0,0101

Tabela 9 – Distâncias corrigidas para o segundo teste analógico

12. Conclusões

Através dos testes realizados e de observações feitas ao longo do projeto do protótipo, chegou-se a várias conclusões:

Primeiramente, para aplicações práticas, há a necessidade de se utilizar um sistema que controle o ganho nas etapas de amplificação dos circuitos de recepção do veículo móvel e do alvo fixo. Isto é preciso devido à necessidade de um ganho mínimo para que o sinal possa excitar tanto o tratamento digital como o analógico e, por outro lado, o ganho não pode ser alto o suficiente para que os amplificadores operacionais atinjam a faixa de saturação. Porém, neste trabalho houve a necessidade de se adequar o ganho manualmente para uma dada distância a ser medida.

Uma outra observação está nas ondas recebidas pelos transdutores dos dois circuitos, que são notavelmente diferentes. Isto ocorre devido a freqüência de excitação dos transdutores. Como se sabe, os dois transdutores foram projetados para funcionar a uma freqüência de 150 kHz, logo, o sinal de recepção do transdutor do veículo móvel, que é excitado a uma freqüência de 170 kHz, apresenta uma forma de onda muito mais distorcida que a do transdutor do alvo fixo, que é excitado por um onda com freqüência de 150 kHz.

Outra constatação importante está relacionada aos problemas em se detectar o sinal corretamente utilizando o comparador de tensão LM311. Isto provém do formato da onda de resposta, recebida pelos circuitos de recepção, que possui um primeiro pulso menor que os demais, de modo que o limiar programado no LM311 possa não detectar este pulso. Caso isto ocorra, existirá um erro na medição do sinal da ordem de 9 mm. Este com certeza é um dos grandes problemas do tratamento analógico.

Um outro problema do tratamento analógico, é a verificação da freqüência recebida pelo PIC dos circuitos. Como o sinal de resposta não tem uma forma adequada, principalmente na recepção do veículo móvel, os primeiros pulsos detectados pelo LM311 podem apresentar uma freqüência

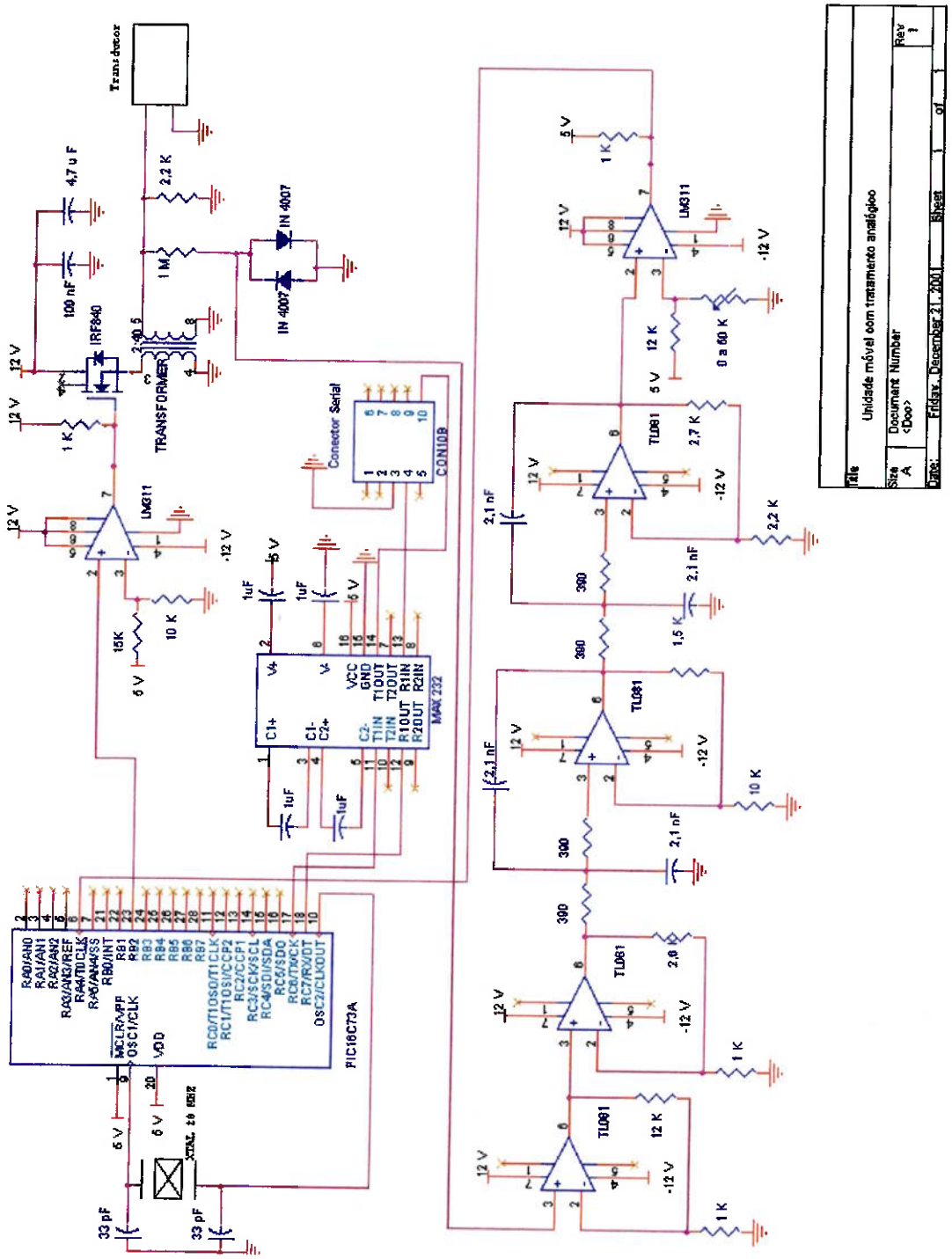
diferente da esperada, fazendo com que o PIC desconsidere cinco ou seis bordas de subida do sinal de resposta, ou seja, quatro ou cinco pulsos (períodos), proporcionando um erro de aproximadamente 3,25 cm na medição da distância.

Com relação ao teste realizado para a unidade móvel possuindo tratamento digital, os grandes erros provavelmente provém do circuito de recepção do alvo fixo, que possui tratamento analógico, podendo apresentar erros de "perdas" de pulsos pelo detector de limiar, da ordem de 9 mm (ver figura 8.9). Os erros de verificação de freqüência neste circuito são menos freqüentes que no veículo móvel, devido a forma da onda recebida, que é visivelmente mais adequada para o tratamento de sinal analógico. Já para a unidade móvel, uma provável fonte de erro é o sinal de referência, que foi determinado a partir de uma aquisição experimental. Este método não garante que o ponto real de início da resposta seja o ponto de início da referência obtida.

Outro problema dos testes realizados com o tratamento digital, foi o tempo de transição dos dados adquiridos pela placa A/D para o PC para que a correlação dos sinais seja efetuada. Porém para aplicações reais, o processamento da correlação deve ser realizado no próprio circuito, com os dados obtidos diretamente do A/D, de modo a se eliminar este tempo devido a comunicação serial.

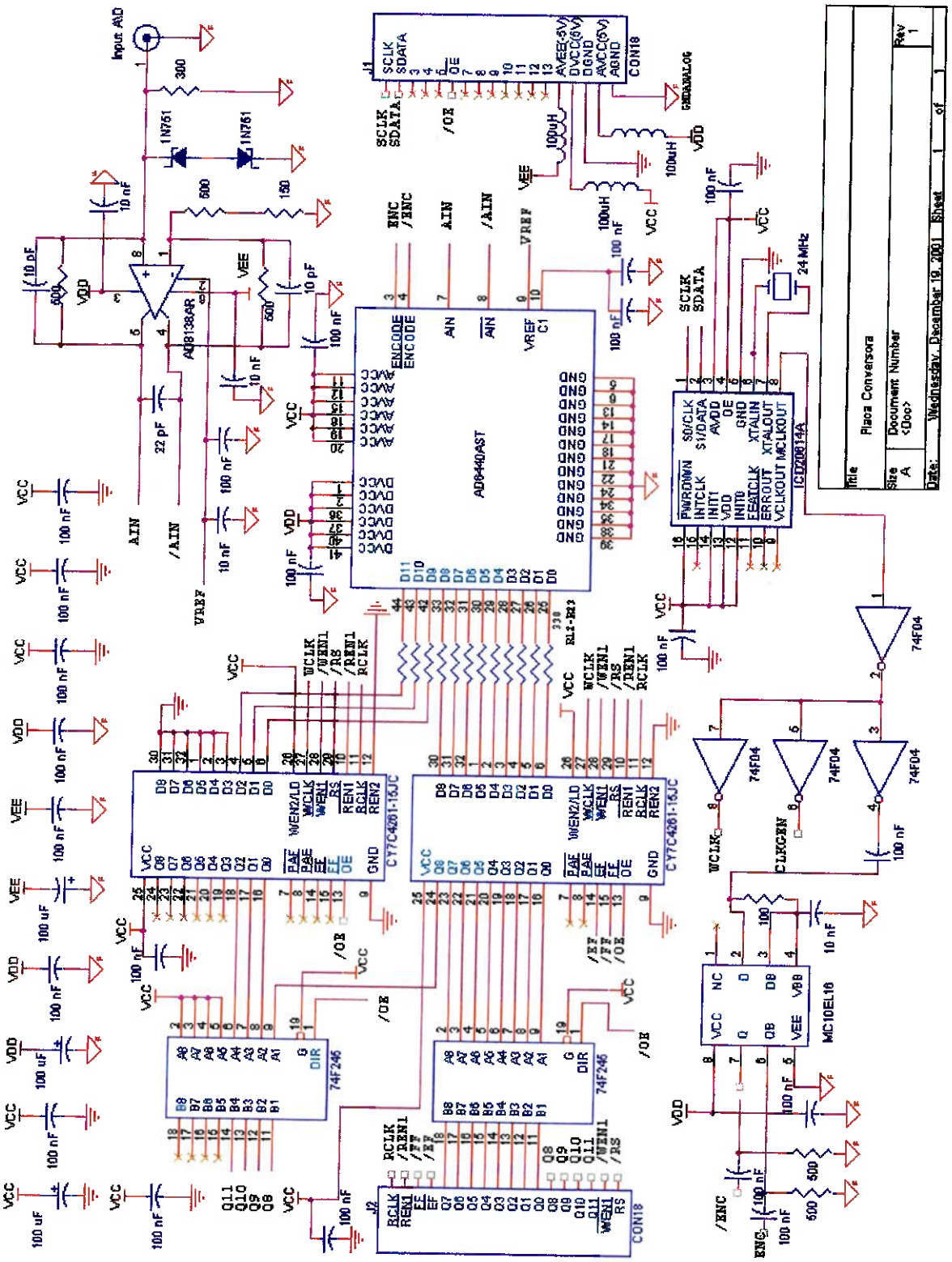
Resumindo, os dados obtidos nos testes foram satisfatórios, apresentando um maior acurácia e precisão no tratamento digital, como já era esperado. Com certeza, se for implementado um método de se realizar o tratamento digital nos circuitos do veículo móvel e do alvo fixo, a acurácia deverá ser bem mais satisfatória.

Anexo A – Projeto do circuito do veículo móvel com tratamento analógico de sinais

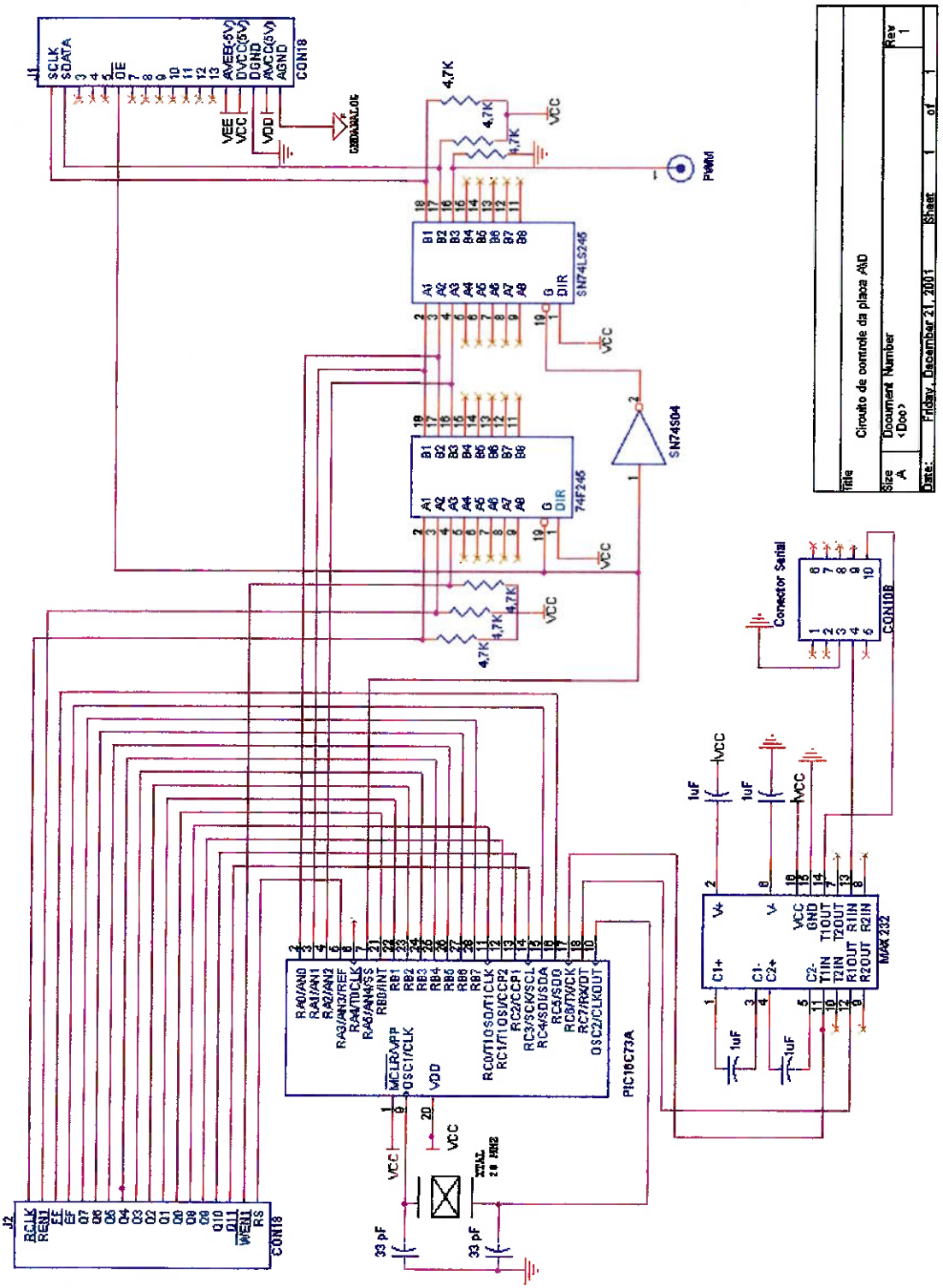


Anexo B – Projeto do circuito do veículo móvel com tratamento digital de sinais

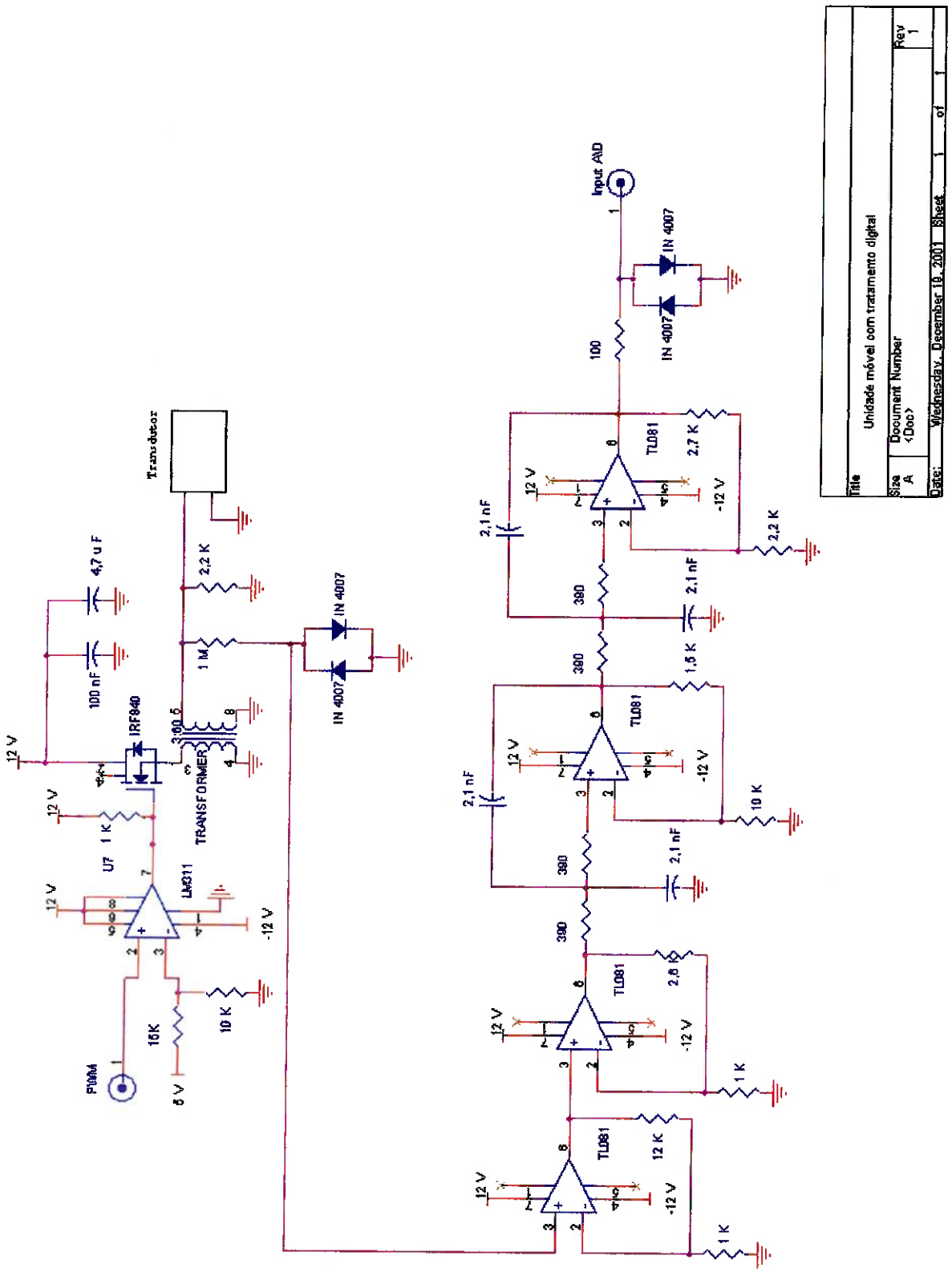
B.1. Placa Digitalizadora



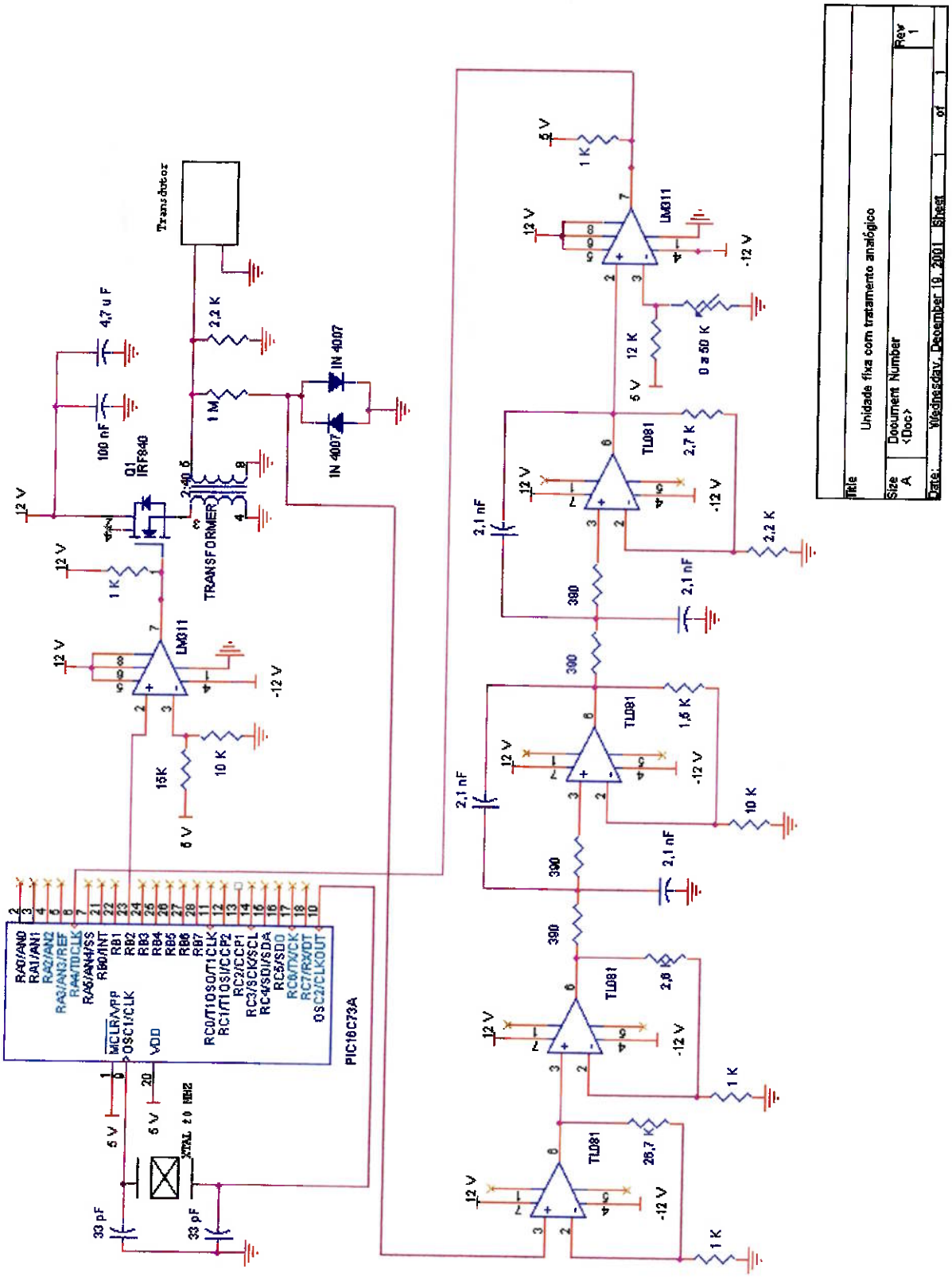
B.2. Controle da Placa Digitalizadora



B.3. Circuito do Veículo Móvel Com Tratamento Digital de Sinais



Anexo C - Projeto do circuito do alvo fixo



Anexo D – Listagem do programa do PIC para o veículo móvel com tratamento digital de sinais

```
; Circuito do PIC com tratamento digital

; O PIC controla a placa de digitalização e a emissão da unidade móvel
; Os dados da memória FIFO são transmitidos via serial para o PC e o cálculo
; da distância feito no mesmo

; Pinos de leitura de dados: PORTB e pinos 0 a 3 do PORTC,
; totalizando 12bits.
; Pino 3 PORTA: reset da FIFO.
; Pino 0 PORTA: pino de dados (Sdata) para programação do
; clock gen e REN1
; (habilitação da leitura da FIFO)
; Pino 1 PORTA: clock (SCLK) para se programar o clock gen.
; Pino 2 PORTA: pino que produzirá o pwm e WEN1 (habilita a escrita da FIFO)
; Pino 5 PORTA: pino de controle do compartilhamento dos 3 pinos anteriores
; Pino 4 PORTC: empty flag (indica se a fifo está vazia)
; Pino 5 PORTC: full flag (indica se a FIFO está cheia)


        list p=16c73b
        include <p16c73b.inc>

;********************* MACROS *****************************************

;*****************************************************************************
;Envia byte para o PC

sndbt   macrobyte

        bsf   STATUS,RP0; Bank 1
        btfss TXSTA,TRMT
        goto  $-1     ; Espera até liberação de TXREG
        bcf   STATUS,RP0; Bank 0
        movlw         byte
        movwf         TXREG  ; Envia byte
        bsf   STATUS,RP0; Bank 1
        btfsc TXSTA,TRMT
        goto  $-1     ; Espera até limpar o flag
        bcf   STATUS,RP0; Bank 0

        endm
;*****************************************************************************
;Envia conteúdo de um registrador para o PC

sndrg   macroregister

        bsf   STATUS,RP0; Bank 1
        btfss TXSTA,TRMT
        goto  $-1     ; Espera até liberação de TXREG
        bcf   STATUS,RP0; Bank 0
        movf  register,0
        movwf         TXREG    ; Envia registrador
        bsf   STATUS,RP0; Bank 1
        btfsc TXSTA,TRMT
        goto  $-1     ; Espera até limpar o flag
        bcf   STATUS,RP0; Bank 0
```

```
        endm
;*************************************************************************
;Recebe byte do PC

rcvbt   macroendereço

        btfss  PIR1,RCIF ; Espera dado
        goto   $-1

        movf  RCREG,0
        movwf       endereço  ; Salva byte recebido no endereço

        endm

;******* Registradores Utilizados - Bank 0 - 20h a 7Fh ********************

        CBLOCK 0X21

            word1   ; byte menos significativo para a programação do clock
            word2
            word3   ; byte mais significativo para a programação do clock
            word    ;utilizada para armazenar cada um dos bytes anteriores
                    ;durante a programação
            unlocked; constante para realizar o destravamento durante a
                    ; programação do clock
            aux     ; utilizada para apontar os endereços de word1, 2 e 3
            bit     ; utilizado na função clock para armazenar o número
                    ; de bits em um byte (8)
            dad     ; utilizada na função clock para armazenar quantos
                    ; bytes são necessários para realizar a programação
            cont    ; func. delay (representa o tempo de atraso)
            comp    ; utilizado para indicar se a programação do clock
                    ; será a default (diferente de 1) ou se será
                    ; definida pelo usuário (iguala a 1)
            lsb     ; armazena o byte menos significativo da
                    ; leitura da fifo
            msb     ; armazena o byte mais significativo da leitura
                    ; da fifo
            TEMP0 ; variável temporária

;Para a geração do trem de pulsos
            th              ; th*3+5=número de ciclos em 1 (PWM)
            tl              ; tl*3+4=número de ciclos em 0 (PWM)
            TRAINL                  ;número de pulsos
            th1             ; utilizados para armazenar valores de
            tl1             ; th, tl e TRAINL durante o pwm
            TRAINL1


        ENDC


;******************** PROGRAMA PRINCIPAL ************************************
        org 0x00  ; Define endereco base no vetor de inicio 0x00
reset
        goto start ; Vai para o inicio do programa

        org 0x04
```

```
start

;************* Programações Default dos timers e módulos CCP ****************

        ; Programação do Timer0 - Controla o tempo de espera entre um trem
        ; de pulso e outro no modo teste

        bcf    STATUS,RP0 ; Bank 0
        bcf    INTCON,T0IE; Desarma interrupção
        bcf    INTCON,T0IF; Limpa flag de overflow
        bsf    STATUS,RP0 ; Bank 1
        movlw       B'11101000'  ; Palavra de programação
        movwf       OPTION_REG ; Programa palavra de controle
                                ; (sem começar contagem)


        ;Programação da PORTA

        bsf STATUS,RP0 ; banco 1
        bcf INTCON,GIE ; desabilita a chave geral das interrupções.

        movlw  B'000000' ; apenas o pino PORTA,4 foi configurado
                         ; como entrada
        movwf TRISA
        movlw  0xFF  ; todos os pinos como entrada
        movwf TRISB
        movlw B'00000111' ; transforma os pinos do PORTA como
                          ; entradas digitais
        movwf ADCON1
        movlw B'10111111' ; apenas o pino de transmissão serial é saída.
        movwf TRISC

;****************** Constantes - valores default ****************************
        bcf    STATUS,RP0 ; Bank 0
        bcf PORTA,2 ; limpa a saída do PORTA pwm (OE=1)
        bsf PORTA,5 ; habilita os pinos 0, 1 e 2 do PORTA para
              ; prog. clock(0 e 1) e pwm (2)
        bsf PORTA,3 ; desativa reset
        bsf PORTA,0 ; desabilita a leitura
        bcf PORTA,1 ; coloca o  SCLK em zero (OE=0).


        movlw       0x04
        movwf       th       ; tempo em high: 3*4+5=17 (número de ciclos)

        movlw       0x04
        movwf       tl       ; tempo em low: 3*4+4=16 (número de ciclos),
                    ; portanto período:33


        movlw       0x08   ;
        movwf       TRAINL          ; número de pulsos desejados

; programa o clock para 7 MHz, utilizando-se um clock de referência de 24 MHz

        movlw 0x96
        movwf word1
```

```
        movlw 0x65
        movwf word2
        movlw 0x64
        movwf word3

        movlw       0x02    ; coloca um valor diferente de 1 em comp,
                            ; necessário para a função clock
        movwf       comp
        call clock ; programa clock com o valor default

        limpar a FIFO
        call limpa



;********************** Inicialização da USART ******************************
        bsf   STATUS,RP0; Bank1
        movlw       0x20    ; Baud rate = 9600 bps
        movwf       SPBRG
        bcf   TXSTA,TX9 ; Transmissão de 8 bits
        bsf   TXSTA,TXEN; Habilita transmissão
        bcf   TXSTA,SYNC; Modo Assíncrono
        bcf   TXSTA,BRGH; Baixa velocidade
        bcf   PIE1,TXIE ; Desarma interrupções de transmissão
        bcf   PIE1,RCIE ; Desarma interrupções de recepção
        bcf   STATUS,RP0; Bank 0
        bsf   RCSTA,SPEN; Habilita porta serial
        bcf   RCSTA,RX9 ; Recepção de 8 bits
        bsf   RCSTA,CREN; Habilita recepção

;******************** Inicializacao da Comunicacao ************************

        bcf   STATUS,RP0; Bank 0

        ;Sequencia de inicio de comunicacao
In_com  bcf   RCSTA,CREN; Desabilita recepção
        bsf   RCSTA,CREN; Habilita recepção
                        ; (P/ evitar travamento da recepcao)

        btfss PIR1,RCIF ; Espera pelo sinal do PC
        goto  $-1       ; Se o sinal não chegou, testa novamente

        movf  RCREG,0       ; Leitura do sinal
        movwf       TEMP0   ; Armazenamento temporário

        movlw       0xAA    ; Sinal de inicializacao esperado
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss STATUS,Z  ; Segue em frente se recebeu o sinal correto
        goto  In_com    ; Se o sinal não é o esperado, continua esperando

        ;Resposta ao PC se recebeu sinal correto (no inicio ou no reinicio)
Recom
        btfss PIR1,TXIF
        goto  $-1       ; Espera até liberação de TXREG
        movlw       0xBB    ; Sinal de resposta
        movwf       TXREG   ; Envia resposta ao PC
```

```
;********************* Loop principal - leitura de comandos ******************
        bcf   STATUS,RP0; Bank 0

Loop    bcf   RCSTA,CREN; Desabilita recepção
        bsf   RCSTA,CREN; Habilita recepção
                          ; (P/ evitar travamento da recepcao)

w_com   btfss PIR1,RCIF ; Loop de espera por um comando
        goto  w_com   ; Se um comando não chegou, testa novamente

        movf  RCREG,0       ; Leitura do comando
        movwf       TEMP0   ; Armazenamento temporário

        movlw       0x47    ; Código ASCII do comando G (realizar localização)
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss STATUS,Z  ; Testa o resultado da operação
        goto  $+3       ; Se o result. não é nulo, verif. o próximo comando
        call  LOCATE        ; Se o resultado é nulo, executa localização
        goto  Loop      ; Volta ao início do Loop e espera novo comando

        movlw       0x56    ; Código ASCII do comando V
                            ; (verificar parametros)
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss STATUS,Z  ; Testa o resultado da operação
        goto  $+3       ; Se o result. não é nulo, verif. o próximo comando
        call  VERIFY        ; Se o resultado é nulo, envia parametros
        goto  Loop      ; Volta ao início do Loop e espera novo comando

        movlw       0x54    ; Código ASCII do comando T (teste de emissão)
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss STATUS,Z  ; Testa o resultado da operação
        goto  $+3       ; Se o result. não é nulo, verif. o próximo comando
        call  TESTE     ; Se o resultado é nulo, inicia teste
        goto  Loop      ; Volta ao início do Loop e espera novo comando


        movlw       0x52    ; Código ASCII do comando R (ler FIFO)
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss STATUS,Z  ; Testa o resultado da operação
        goto  $+3       ; Se o result. não é nulo, verif. o próximo comando
        call  ler       ; Se o resultado é nulo, le fifo
        goto  Loop      ; Volta ao início do Loop e espera novo comando

        movlw       0xAA    ; Código de reinicializacao
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss STATUS,Z  ; Testa o resultado da operação
        goto  $+2       ; Se o result. não é nulo, verif. o próximo comando
        goto  Recom     ; Envia resposta de reinicializacao


        movlw       0x46    ; Código ASCII do comando F (programa clock)
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss STATUS,Z  ; Testa o resultado da operação
        goto  $+5       ; Se o result. não é nulo, verif. o próximo comando
        movlw       0x01
        movwf       comp
        call  clock     ; Se o resultado é nulo, executa programação do clock
        goto  Loop      ; Volta ao início do Loop e espera novo comando
```

```
        movlw   0x4C    ; Código ASCII do comando L (limpar FIFO)
        subwf TEMP0,0   ; Realiza comparação por subtração
        btfss  STATUS,Z ; Testa o resultado da operação
        goto   $+3      ; Se o result. não é nulo, verif. o próximo comando
        call   limpa    ; Se o resultado é nulo, limpar FIFO
        goto   Loop     ; Volta ao início do Loop e espera novo comando

        goto   Loop     ; Volta ao início do Loop e espera novo comando

;******************** ROTINAS DE EXECUÇÃO DOS COMANDOS **********************
LOCATE ; Realiza localização e envia resposta

        ; realiza pwm

        movf  TRAINL,0
        movwf       TRAINL1
        movf  tl,0
        movwf       tl1
        movf  th,0
        movwf       th1


pwm

        bsf    PORTA,2
        decfsz      th1
        goto   $-1
        nop
        movf  tl,0
        movwf       tl1
        movf  th,0
        movwf       th1
        bcf    PORTA,2
        decfsz      tl1
        goto$-1
        nop
        decfsz      TRAINL1
        goto   pwm


; espera a oscilação do transdutor, após o término do trem de pulsos, terminar

        bsf    STATUS,RP0 ; Bank 1
        movlw       B'11000000'  ; Palavra de programação prescaler 1:2
        movwf       OPTION_REG ; Programa palavra de controle
                                ; (sem começar contagem)
        bcf    STATUS,RP0 ; Bank 0
        clrf   TMR0  ; limpa timer
        bcf    INTCON,T0IF  ; limpa bit de estouro

        ;Espera pelo fim do trem e pára os pulsos
        btfss  INTCON,T0IF  ;
        goto   $-1          ; Se Timer0 não estourar, continua no loop

;*****************************************************************************
;                       escreve na FIFO
;*****************************************************************************
        bsf PORTA,2 ;desabilita a escrita
        nop
        bcf PORTA,5 ;habilita para a escrita e leitura
```

```
        call limpa  ; limpa a FIFO antes de enche-la
        bcf PORTA,2 ; habilita a escrita

escrever        ; espera a memoria ficar cheia
        btfsc PORTC,5
        goto escrever

        bsf PORTA,2 ; desabilita a escrita


        bcf PORTA,5 ; habilita o pino RA0 como REN1 e RA1 como RCLK
        bsf PORTA,1 ; sincronização com o RCLK, a fim de que o
                    ; EMPTY FLAG seja setado
        bcf PORTA,1

        bsf PORTA,5 ; habilita para pwm e programação do clock
        nop
        bcf PORTA,2 ; limpa PORTA do pwm

        sndbt 0x50
        sndbt 0x4F
        sndbt 0x4B
        sndbt 0x00

        return
;*****************************************************************************
;                           le a FIFO
;*****************************************************************************

ler

        bsf PORTA,2 ; desabilita a escrita na FIFO.
        bcf PORTA,5 ; habilita o pino RA0 como REN1 e RA1 como
                    ; RCLK e RA2 como WEN1
        bcf PORTA,0 ; habilita a leitura

leitura


        bsf PORTA,1  ; realiza o clock da leitura (RCLK)
        bcf PORTA,1
        movf PORTB,0
        movwf lsb      ; armazena em lsb o byte menos significativo dos dados.
        movf PORTC,0
        movwf msb    ;armazena em msb o byte mais significativo dos dados.
        movlw B'00001111' ; realiza operação and, pois apenas
                          ; os primeiros 4 bits interessam.
        andwf msb,1


; realiza protocolo de comunicação com o PC:
; O PIC envia um dado e o PC verifica se este é valido ou não. Se a primeira opção ocorrer o
PIC envia o próximo dado, se não ele envia
; o dado novamente.

envia
        sndbt 0x4E ; prepara para enviar resposta numérica
        sndbt 0x02
        sndrg lsb
```

```
        sndrg msb

        rcvbt aux  ; byte de teste

        ; Espera sinal de dado válido do PC (0xCC)

        movlw       0xCC     ; Código de acerto
        subwf aux,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   envia     ; Se o result. não é nulo, verif. o próximo comando

        btfsc PORTC,4 ; realiza a leitura até a memoria esvaziar
        goto leitura

        ; desativa leitura


        bsf PORTA,0  ; desabilita a leitura
        bsf PORTA,5  ; habilita a progrmação do clock e pwm.
        bcf PORTA,2

        ; retorna para o loop principal

        return


;*****************************************************************************
;                          Programa clock
;*****************************************************************************


clock

        movlw       0x01     ; verifica se o programação será feita pelo usuário,
                             ; ou se será a default.
        subwf comp,0
        btfss  STATUS,Z
        goto   main
        rcvbt  word1
        rcvbt  word2
        rcvbt  word3


main

        ; configuração das constantes a serem utilizadas

        movlw 0x08
        movwf bit
        movlw 0x03
        movwf dad
        movlw 0x05
        movwf unlocked
        bcf    PORTA,1


;*****************************************************************************
;                                destrava
;*****************************************************************************
```

```
destrava
        bsf PORTA,0
        call delay
        bsf PORTA,1
        call delay
        bcf PORTA,1
        decfsz unlocked,1
        goto destrava

        call delay
        bcf PORTA,0
        call delay
        bsf PORTA,1
        call delay


;*****************************************************************************
;                                Start Bit
;*****************************************************************************

        bsf PORTA,1
        bcf PORTA,0
        call delay
        bcf PORTA,1
        call delay
        bsf PORTA,1
        call delay


;*****************************************************************************
;                            Escreve os dados
;*****************************************************************************

        movlw 0x20
        movwf aux  ; aux: variável que indicará os endereços
                   ; das palavras a serem programada (word1, word2, word3).

dados

        bsf PORTA,1
        call delay
        incf aux,1
        movf aux,0
        movwf FSR
        movf INDF,0
        movwf word

bits
        ;btfss PORTA,4
        ;goto main

        movlw 0x01
        xorwf word,1
        btfsc word,0
        goto $+2
        goto $+3
    bsf PORTA,0
        goto $+2
```

```
            bcf PORTA,0
            call delay
            bcf PORTA,1
            call delay
            btfsc word,0
            goto $+2
            goto $+3
        bcf PORTA,0
            goto $+2
            bsf PORTA,0
            call delay
            bsf PORTA,1
            call delay
            rrf word,1
            decfsz bit,1
            goto $+2
            goto $+2
            goto bits

            movlw 0x08
            movwf bit

            decfsz dad,1
            goto dados

;*****************************************************************************************
;                                        Stop bits
;*****************************************************************************************

            bsf PORTA,1
            bsf PORTA,0
            call delay
            bcf PORTA,1
            call delay
            bsf PORTA,1
            call delay
            bcf PORTA,1
            bcf PORTA,0

            sndbt 0x50    ; código de inicialização
            sndbt 0x4F    ; 'O'
            sndbt 0x4B    ; 'K'
            sndbt 0x00    ; null

            return


;*************************************************************************
;                            limpa a FIFO
;*************************************************************************

limpa
            bcf PORTA,3 ; reseta a fifo
            bsf PORTA,3 ; desativa o reset

            return


;*************************************************************************
VERIFY  ;Envia os parametros programados
```

```
        ; envia a palavra de programação do clock
        sndbt 0x4E ; prepara para enviar resposta numérica
        sndbt 0x03
        sndrg word3
        sndrg word2
        sndrg word1

        ; envia status da FIFO

        btfsc PORTC,4
        goto  verifica
        btfss PORTC,5
        goto erro
        goto vazio

verifica

        btfsc PORTC,5
        goto erro
        goto cheio

erro
        sndbt 0x50    ; código de palavra
        sndbt 0x65    ; 'E'
        sndbt 0x52    ; 'R'
        sndbt 0x52    ; 'R'
        sndbt 0x6F    ; 'O'
        sndbt 0x00    ; 'null'

        goto env_tl

cheio

        sndbt 0x50    ; código de palavra
        sndbt 0x63    ; 'C'
        sndbt 0x48    ; 'H'
        sndbt 0x65    ; 'E'
        sndbt 0x49    ; 'I'
        sndbt 0x6F    ; 'O'
        sndbt 0x00    ; 'null'

        goto env_tl

vazio

        sndbt 0x50    ; código de palavra
        sndbt 0x56    ; 'V'
        sndbt 0x61    ; 'A'
        sndbt 0x5A    ; 'Z'
        sndbt 0x49    ; 'I'
        sndbt 0x6F    ; 'O'
        sndbt 0x00    ; 'null'


        ;Retorna ao Loop principal
        return
;*****************************************************************************
TESTE   ;Realiza teste de emissão
```

```
        sndbt 0x50    ; código de palavra
        sndbt 0x54    ; 'T'
        sndbt 0x65    ; 'e'
        sndbt 0x73    ; 's'
        sndbt 0x74    ; 't'
        sndbt 0x65    ; 'e'
        sndbt 0x20    ; ' '
        sndbt 0x49    ; 'I'
        sndbt 0x6E    ; 'n'
        sndbt 0x69    ; 'i'
        sndbt 0x63    ; 'c'
        sndbt 0x69    ; 'i'
        sndbt 0x61    ; 'a'
        sndbt 0x64    ; 'd'
        sndbt 0x6F    ; 'o'
        sndbt 0x00    ; 'null'


;Programação do Timer0 para medição do período de repetição

tstcon

        bsf    STATUS,RP0; Bank 1
        movlw       0xE5      ; Palavra de programação, prescaler 1:64
        movwf       OPTION_REG ; Programa palavra de controle
        bcf    OPTION_REG,T0CS; Liga Timer0
        bcf    STATUS,RP0; Bank 0

        ;Medição do intervalo
        clrf   TMR0          ; Limpa Timer0
        bcf    INTCON,T0IF   ; Limpa flag de overflow
        btfss  INTCON,T0IF   ;
        goto   $-1           ; Espera Timer0 estourar



        ; realiza pwm

        movf  TRAINL,0
        movwf       TRAINL1
        movf  tl,0
        movwf       tl1
        movf  th,0
        movwf       th1


pwm1

        bsf    PORTA,2
        decfsz      th1
        goto   $-1
        nop
        movf  tl,0
        movwf       tl1
        movf  th,0
        movwf       th1
```

```
        bcf    PORTA,2
        decfsz         tl1
        goto$-1
        nop
        decfsz         TRAINL1
        goto   pwm1



        btfss  PIR1,RCIF ; Testa se recebeu comando de parada
        goto   tstcon    ; Continua no teste se não recebeu nenhum comando

        movf   RCREG,0         ; Leitura do comando
        movwf          TEMP0   ; Armazenamento temporário

        movlw          0x5A    ; Código ASCII do comando Z (Encerrar teste)
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   tstcon    ; Continua com o teste se não recebeu comando
                         ; de parada


        sndbt  0x50    ; código de palavra
        sndbt  0x54    ; 'T'
        sndbt  0x65    ; 'e'
        sndbt  0x73    ; 's'
        sndbt  0x74    ; 't'
        sndbt  0x65    ; 'e'
        sndbt  0x20    ; ' '
        sndbt  0x45    ; 'E'
        sndbt  0x6E    ; 'n'
        sndbt  0x63    ; 'c'
        sndbt  0x65    ; 'e'
        sndbt  0x72    ; 'r'
        sndbt  0x72    ; 'r'
        sndbt  0x61    ; 'a'
        sndbt  0x64    ; 'd'
        sndbt  0x6F    ; 'o'
        sndbt  0x00    ; 'null'

        return         ; Encerra o teste


;*******************************************************************************

; Função para realizar um delay durante a programação do clock

delay
        movlw 0x05
        movwf cont
        decfsz cont,1
        goto $-1

        return


        end
```

Anexo E – Listagem do programa do PIC da unidade móvel para tratamento analógico de sinais

; Programa do PIC da unidade móvel com tratamento analógico

```
; Realiza a emissão, cronometra o tempo até a chegada do sinal de resposta e calcula a distância,
;enviando este valor para o PC.

;Pino de saída do trem de pulsos: RB2 pino 23

;Pino de entrada: RA4 - pino 06
;O PIC analisa a frequência do sinal de entrada deste pino e responde se for o sinal
; de resposta esperado.


        list p=16c73b
        include <p16c73b.inc>

;********************* MACROS *****************************************
;Multiplicação - 1 byte x 1 byte

mpy     macromultplr,multpld,rlow,rhigh,count,temp

    clrf  rlow
    clrf  rhigh
        movf  multplr,0
        movwf       temp
    movlw     0x8
    movwf     count
    movf      multpld,0
    bcf  STATUS, C  ; Clear carry bit
        ; Loop de multiplicação
        rrf     temp,1
    btfsc     STATUS, C
    addwf     rhigh,1
    rrf   rhigh,1
    rrf   rlow,1
    decfsz  count,1
    goto$-6

        endm
;*****************************************************************************
;Envia byte para o PC

sndbt   macrobyte

        bsf    STATUS,RP0; Bank 1
        btfss  TXSTA,TRMT
        goto   $-1     ; Espera até liberação de TXREG
        bcf    STATUS,RP0; Bank 0
        movlw       byte
        movwf       TXREG  ; Envia byte
        bsf    STATUS,RP0; Bank 1
        btfsc  TXSTA,TRMT
        goto   $-1     ; Espera até limpar o flag
        bcf    STATUS,RP0; Bank 0

        endm
```

```
;*****************************************************************************
;Envia conteúdo de um registrador para o PC

sndrg   macroregister

        bsf   STATUS,RP0; Bank 1
        btfss TXSTA,TRMT
        goto  $-1     ; Espera até liberação de TXREG
        bcf   STATUS,RP0; Bank 0
        movf  register,0
        movwf         TXREG    ; Envia registrador
        bsf   STATUS,RP0; Bank 1
        btfsc TXSTA,TRMT
        goto  $-1     ; Espera até limpar o flag
        bcf   STATUS,RP0; Bank 0

        endm

;*****************************************************************************
;Recebe byte do PC

rcvbt   macroendereço

        btfss PIR1,RCIF ; Espera dado
        goto  $-1

        movf  RCREG,0
        movwf         endereço  ; Salva byte recebido no endereço

        endm

;******* Registradores Utilizados - Bank 0 - 20h a 7Fh **********************
;Para o cálculo da distância (3 bytes X 3 bytes)
VSOUND0         equ 0x30 ;
VSOUND1     equ 0x31 ;
VSOUND2         equ 0x32 ;Vel. do som com resolução de 24 bytes (0.1 mm/s)

TIME0       equ 0x40 ;LSB do tempo cron. no Timer1 (resolução de 200 ns)
TIME1       equ 0x41 ;MSB do tempo cron. no Timer1
TIME2       equ 0x42 ;Número de overflows do Timer1

RESP0       equ 0x50
RESP1       equ 0x51
RESP2       equ 0x52
RESP3       equ 0x53
RESP4       equ 0x54
RESP5       equ 0x55 ;Resposta da multiplicação (6 bytes)

X0          equ 0x60
X1          equ 0x61
Y0          equ 0x62
Y1          equ 0x63
Y2          equ 0x64
Z0          equ 0x65
Z1          equ 0x66
Z2          equ 0x67
W0          equ 0x68
W1          equ 0x69
W2          equ 0x6A
```

```
P0          equ 0x6B
P1          equ 0x6C ;Termos da soma final (registradores auxiliares)

TEMP0       equ 0x6D
TEMP1       equ 0x6E
TEMP2       equ 0x6F
TEMP3       equ 0x45
TEMP        equ 0x46
CONT        equ 0x47 ;Variáveis temporárias e contadores


aux         equ 0x73 ;contador de tempo para descobrir a freq. do sinal de entrada

;Para a geração do trem de pulsos

status_temp equ 0x74 ;armazena temporariamente o valor do status (para a inter.)
w_temp      equ 0x75 ;armazena temporariamente o valor do work (para a inter.)

th          equ 0x70; th*3+5=número de ciclos em 1 (PWM)
tl          equ 0x71; tl*3+4=número de ciclos em 0 (PWM)
TRAINL      equ 0x72;número de pulsos
th1         equ 0x73; utilizados para armazenar valores de th, tl e TRAINL durante
tl1         equ 0x76; o pwm.
TRAINL1     equ 0x77

;********** PROGRAMA PRINCIPAL **********
        org 0x00  ; Define endereco base no vetor de inicio 0x00
reset
        goto start ; Vai para o inicio do programa

        ; interrupção do timer1, necesário para incrementar o terceiro byte da cronometragem

        org 0x04  ; inicio da interrupção

        ; salva w e status do programa principal
        movwf w_temp
        swapf STATUS,0
        movwf status_temp


        incf TIME2,1 ; adiciona 1 no byte 3
        btfsc STATUS,C ; se estourar este byte a distância é maior que 100 m (erro)
        goto erro
        bcf PIR1,TMR1IF ; limpa o flag de interrupção do TIMER1
        movlw 0x01   ; verifica se o Timer 0 é maior ou igual a 1
        subwf TMR0,0
        btfss STATUS,C
        goto $+7
        movlw 0x04   ; verifica se o Timer 0 é menor ou igual a 4.
        subwf TMR0,0
        btfsc STATUS,C
        goto $+3


        movlw 0x17   ; compensação de aux devido à ocorrência da interrupção
        addwf aux,1

        ; recupera w e status
```

```
        swapf status_temp,0
        movwf STATUS
        swapf w_temp,1
        swapf w_temp,0

        retfie ; retorna da interrupção


start
;****************** Constantes - valores default ****************************
        bcf    STATUS,RP0 ; Bank 0

        movlw       0x04
        movwf       th      ; tempo em high: 3*4+5=17 (número de ciclos)

        movlw       0x04
        movwf       tl      ; tempo em low: 3*4+4=16 (número de ciclos), portanto
;período:33


        movlw       0x08    ;
        movwf       TRAINL          ; número de pulsos desejados

        movlw       0xC0
        movwf       VSOUND0

        movlw       0xE1
        movwf       VSOUND1

        movlw       0xE4
        movwf       VSOUND2     ; Velocidade do som (1500.0000 m/s)

;************* Programações Default dos PORTS ***************


        ;Programação da PORTA

        bsf    STATUS,RP0; Bank 1
        movlw B'00000111'
        movwf ADCON1 ; habilita o portA como valor digital
        bsf TRISA,4    ; habilita o pino como entrada


;********************** Inicialização da USART ****************************
        bsf    STATUS,RP0; Bank1
        movlw       0x20     ; Baud rate = 9600 bps
        movwf       SPBRG
        bcf    TXSTA,TX9 ; Transmissão de 8 bits
        bsf    TXSTA,TXEN; Habilita transmissão
        bcf    TXSTA,SYNC; Modo Assíncrono
        bcf    TXSTA,BRGH; Baixa velocidade
        bcf    PIE1,TXIE ; Desarma interrupções de transmissão
        bcf    PIE1,RCIE ; Desarma interrupções de recepção
        bcf    STATUS,RP0; Bank 0
        bsf    RCSTA,SPEN; Habilita porta serial
        bcf    RCSTA,RX9 ; Recepção de 8 bits
        bsf    RCSTA,CREN; Habilita recepção

;******************** Inicializacao da Comunicacao ************************
```

```
        bcf    STATUS,RP0; Bank 0

        ;Sequencia de inicio de comunicacao
In_com  bcf    RCSTA,CREN; Desabilita recepção
        bsf    RCSTA,CREN; Habilita recepção (P/ evitar travamento da recepcao)

        btfss  PIR1,RCIF ; Espera pelo sinal do PC
        goto   $-1       ; Se o sinal não chegou, testa novamente

        movf   RCREG,0       ; Leitura do sinal
        movwf        TEMP0   ; Armazenamento temporário

        movlw        0xAA    ; Sinal de inicializacao esperado
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Segue em frente se recebeu o sinal correto
        goto   In_com  ; Se o sinal não é o esperado, continua esperando

        ;Resposta ao PC se recebeu sinal correto (no inicio ou no reinicio)
Recom   btfss  PIR1,TXIF
        goto   $-1     ; Espera até liberação de TXREG
        movlw        0xBB    ; Sinal de resposta
        movwf        TXREG   ; Envia resposta ao PC

;******************** Loop principal - leitura de comandos ******************
        bcf    STATUS,RP0; Bank 0

Loop    bcf    RCSTA,CREN; Desabilita recepção
        bsf    RCSTA,CREN; Habilita recepção (P/ evitar travamento da recepcao)

w_com   btfss  PIR1,RCIF ; Loop de espera por um comando
        goto   w_com   ; Se um comando não chegou, testa novamente

        movf   RCREG,0       ; Leitura do comando
        movwf        TEMP0   ; Armazenamento temporário

        movlw        0x47    ; Código ASCII do comando G (realizar localização)
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   $+3       ; Se o result. não é nulo, verif. o próximo comando
        call   LOCATE          ; Se o resultado é nulo, executa localização
        goto   Loop      ; Volta ao início do Loop e espera novo comando

        movlw        0x53    ; Código ASCII do comando S (verificar parametros)
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   $+3       ; Se o result. não é nulo, verif. o próximo comando
        call   VERIFY          ; Se o resultado é nulo, envia parametros
        goto   Loop      ; Volta ao início do Loop e espera novo comando

        movlw        0x54    ; Código ASCII do comando T (teste de emissão)
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   $+3       ; Se o result. não é nulo, verif. o próximo comando
        call   TESTE   ; Se o resultado é nulo, inicia teste
        goto   Loop      ; Volta ao início do Loop e espera novo comando

        movlw        0x41    ; Código ASCII do comando A (teste do pino)
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   $+3       ; Se o result. não é nulo, verif. o próximo comando
```

```
        call   TST_PIN          ; Se o resultado é nulo, realiza teste
        goto   Loop     ; Volta ao início do Loop e espera novo comando

        movlw          0xAA     ; Código de reinicializacao
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   $+2       ; Se o result. não é nulo, verif. o próximo comando
        goto   Recom   ; Envia resposta de reinicializacao

        movlw          0x56     ; Código ASCII do comando V (prog. velocidade)
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   $+3       ; Se o result. não é nulo, verif. o próximo comando
        call   SET_V   ; Se o resultado é nulo, programa velocidade
        goto   Loop      ; Volta ao início do Loop e espera novo comando


        goto   Loop      ; Volta ao início do Loop e espera novo comando

;********************* ROTINAS DE EXECUÇÃO DOS COMANDOS **********************
LOCATE ; Realiza localização e envia resposta



        ;Programação do Timer1 - Cronometragem do tempo

        bcf    STATUS,RP0 ; Bank 0
        clrf   TMR1L   ; Limpa Timer1
        clrf   TMR1H   ; Limpa Timer1
        clrf   TIME2   ; Limpa o contador de overflows
        bcf    PIR1,TMR1IF; Limpa flag de overflow
        movlw          0x00     ; Palavra de controle
        movwf          T1CON    ; Programa palavra de controle (sem ligar)
        bsf    STATUS,RP0 ; Bank 1
        bcf    INTCON,7   ; desabilita a chave geral das interrupções
        bcf    PIE1,TMR1IE; desabilita interrupção
        bcf    STATUS,RP0 ; Bank 0


        ; realiza pwm

        movf  TRAINL,0
        movwf          TRAINL1
        movf  tl,0
        movwf          tl1
        movf  th,0
        movwf          th1
        bsf    T1CON,0 ; início da cronometragem

pwm

        bsf    PORTA,2
        decfsz         th1
        goto   $-1
        nop
        movf  tl,0
        movwf          tl1
        movf  th,0
        movwf          th1
        bcf    PORTA,2
```

```
            decfsz      tl1
            goto$-1
            nop
            decfsz      TRAINL1
            goto  pwm

            ; espera a oscilação do transdutor, após o término do trem de pulsos, terminar

            bsf   STATUS,RP0 ; Bank 1
            movlw       B'11000000'  ; Palavra de programação prescaler 1:2
            movwf       OPTION_REG  ;  Programa  palavra  de  controle  (sem  começar
;contagem)
            bcf   STATUS,RP0 ; Bank 0
            clrf  TMR0  ; limpa timer
            bcf   INTCON,T0IF  ; limpa bit de estouro

            ;Espera pelo fim do trem e pára os pulsos
            btfss INTCON,T0IF  ;
            goto  $-1                ; Se Timer0 não estourar, continua no loop

            ; Programa Timer 0 para realizar a diferenciação do eco

            bcf   STATUS,RP0  ; Bank 0
            bcf   INTCON,T0IF; Limpa flag de overflow
            bsf   STATUS,RP0 ; Bank 1
            movlw       B'10101000' ; Palavra de programação
            movwf       OPTION_REG ; Programa palavra de controle

            ; Realiza a diferenciação entre o eco e o sinal a ser recebido

            bcf   STATUS,RP0; Bank 0
            movlw 0x00
            movwf aux

inicio

            clrf  TMR0    ; limpa Timer0
            movlw       0xA4
            movwf       aux

espera
            btfss TMR0,0            ; espera o sinal externo ativar o clock
            goto  espera ;
;           através de aux cronometra-se o número de ciclos executados na freq. do PIC, até que
;a 3 borda de subida
;           do clock externo ocorra

            movlw 0x06

            bcf  STATUS,C

verifica
            addwf aux,1
            btfsc STATUS,C
            goto  inicio
            btfss TMR0,2
            goto verifica

            bcf   STATUS,C
            movlw 0xF7  ; verifica se a freq. é menor que 180 khz e maior que 165
```

```
        subwf aux,0
        btfss STATUS,C
        goto inicio


        bcf    T1CON,TMR1ON   ; Termina cronometragem
        movf  TMR1L,0         ; armazena valor menos significativo do timer em TIME0
        movwf         TIME0
        movf  TMR1H,0                 ; armazena valor mais significativo do timer em TIME1
        movwf         TIME1

        ; realização da compensação

        movlw         0x10    ; compensação do segundo byte
        subwf TIME1,1
        movlw         0xD7    ; compensação do primeiro byte
        subwf TIME0,1

        ;Executa multiplicação
;***** A multiplicação é feita pela decomposição de cada número em bytes   *
;***** Os números a serem multiplicados (VSOUND e TIME) possuem 3 bytes cada*
;***** O resultado possui 6 bytes                                          *
;***** Vários testes foram feitos. Multiplicação OK                        *
;***** No de ciclos : ~729 ciclos (145,8 us)                               *
;***** A resposta da multiplicação é em 10e-8 mm e é a distância entre     *
;***** as unidades                                                         *

        mpy   VSOUND2,TIME2,X0,X1,CONT,TEMP

        mpy   VSOUND0,TIME0,P0,P1,CONT,TEMP

        clrf    Y1
        clrf    Y2
        mpy   VSOUND2,TIME1,TEMP0,TEMP1,CONT,TEMP
        mpy   TIME2,VSOUND1,TEMP2,TEMP3,CONT,TEMP
        movf  TEMP0,0
        addwf TEMP2,0
        movwf         Y0
        btfsc  STATUS,C; testa o bit carry
        incf    Y1,1
        movf  TEMP1,0
        addwf TEMP3,0
        btfsc  STATUS,C; testa o bit carry
        incf    Y2,1
        addwf Y1,1

        clrf    W1
        clrf    W2
        mpy   VSOUND1,TIME0,TEMP0,TEMP1,CONT,TEMP
        mpy   VSOUND0,TIME1,TEMP2,TEMP3,CONT,TEMP
        movf  TEMP0,0
        addwf TEMP2,0
        movwf         W0
        btfsc  STATUS,C; testa o bit carry
        incf    W1,1
        movf  TEMP1,0
        addwf TEMP3,0
        btfsc  STATUS,C; testa o bit carry
        incf    W2,1
        addwf W1,1
```

```
clrf    Z1
clrf    Z2
mpy   VSOUND2,TIME0,TEMP0,TEMP1,CONT,TEMP
mpy   VSOUND0,TIME2,TEMP2,TEMP3,CONT,TEMP
movf  TEMP0,0
addwf TEMP2,0
movwf         Z0
btfsc  STATUS,C; testa o bit carry
incf    Z1,1
movf  TEMP1,0
addwf TEMP3,0
btfsc  STATUS,C; testa o bit carry
incf    Z2,1
addwf Z1,1
mpy   VSOUND1,TIME1,TEMP0,TEMP1,CONT,TEMP
movf  TEMP0,0
addwf Z0,1
btfss  STATUS,C; pula se o carry for 1
goto  $+5     ; se o carry=0, não incrementar os outros bytes
movlw         0x01
addwf Z1,1
btfsc  STATUS,C
incf    Z2,1
movf  TEMP1,0
addwf Z1,1
btfsc  STATUS,C
incf    Z2,1

;Soma dos termos encontrados
clrf    RESP2
clrf    RESP3
clrf    RESP4
clrf    RESP5 ; Limpa os registradores de resposta

movf  P0,0
movwf         RESP0 ; Primeiro byte do resultado

movf  W0,0
addwf P1,0
movwf         RESP1 ; Segundo byte do resultado
btfsc  STATUS,C; testa o bit carry
incf    RESP2,1         ; incrementa o terceiro byte

movf  W1,0
addwf Z0,0
addwf RESP2,1         ; terceiro byte do resultado
btfsc  STATUS,C; testa o bit carry
incf    RESP3,1         ; incrementa o quarto byte

movf  W2,0
addwf Z1,0
addwf RESP3,1
btfsc  STATUS,C; testa o bit carry
incf    RESP4,1         ; incrementa o quinto byte
movf  Y0,0
addwf RESP3,1         ; completa a soma do quarto byte
btfsc  STATUS,C; testa o bit carry
incf    RESP4,1         ; incrementa o quinto byte
```

```
            movf  Z2,0
            addwf Y1,0
            addwf RESP4,1
            btfsc  STATUS,C; testa o bit carry
            incf   RESP5,1          ; incrementa o sexto byte
            movf  X0,0
            addwf RESP4,1           ; completa a soma do quinto byte
            btfsc  STATUS,C; testa o bit carry
            incf   RESP5,1          ; incrementa o sexto byte

            movf  X1,0
            addwf RESP5,1
            movf  Y2,0
            addwf RESP5,1           ; completa a soma do sexto byte (operação completa)

            ;Envia resposta via canal serial
            sndbt 0x4E    ; código de número
            sndbt 0x06    ; número de bytes do número
            sndrg RESP0 ; primeiro byte da resposta
            sndrg RESP1 ; segundo byte da resposta
            sndrg RESP2 ; terceiro byte da resposta
            sndrg RESP3 ; quarto byte da resposta
            sndrg RESP4 ; quinto byte da resposta
            sndrg RESP5 ; sexto byte da resposta



            ;Retorna ao Loop principal
            return

            ;Mensagem de erro - resposta não recebida
erro        sndbt 0x50    ; código de palavra
            sndbt 0x45    ; 'E'
            sndbt 0x72    ; 'r'
            sndbt 0x72    ; 'r'
            sndbt 0x6F    ; 'o'
            sndbt 0x00    ; 'null'

            ;Retorna a preparação dos timers
            goto  Loop

;*****************************************************************************
VERIFY  ;Envia velocidade do som programada

            ;Velocidade do som
            sndbt 0x4E
            sndbt 0x03
            sndrg VSOUND2
            sndrg VSOUND1
            sndrg VSOUND0


            ;Retorna ao Loop principal
            return

;*****************************************************************************
TESTE   ;Realiza teste de emissão

            sndbt 0x50    ; código de palavra
            sndbt 0x54    ; 'T'
```

```
        sndbt 0x65    ; 'e'
        sndbt 0x73    ; 's'
        sndbt 0x74    ; 't'
        sndbt 0x65    ; 'e'
        sndbt 0x20    ; ' '
        sndbt 0x49    ; 'I'
        sndbt 0x6E    ; 'n'
        sndbt 0x69    ; 'i'
        sndbt 0x63    ; 'c'
        sndbt 0x69    ; 'i'
        sndbt 0x61    ; 'a'
        sndbt 0x64    ; 'd'
        sndbt 0x6F    ; 'o'
        sndbt 0x00    ; 'null'

; Programação do Timer 0

pgd
        bcf   STATUS,RP0 ; Bank 0
        bcf   INTCON,T0IE; Desarma interrupção
        bcf   INTCON,T0IF; Limpa flag de overflow
        bsf   STATUS,RP0; Bank 1
        movlw       B'11101000'  ; Palavra de programação
        movwf       OPTION_REG ; Programa palavra de controle

; Realiza a diferenciação entre o eco e o sinal a ser recebido

        bcf   STATUS,RP0; Bank 0
        movlw 0x00
        movwf aux

inicio1

        clrf   TMR0    ; limpa Timer0
        movlw       0xA4
        movwf       aux

espera1
        btfss  TMR0,0          ; espera o sinal externo ativar o clock
        goto   espera1  ;

;       através de aux cronometra-se o número de ciclos executados na freq. do PIC, até que
;a 3 borda de subida
;       do clock externo ocorra

        movlw 0x06

        bcf  STATUS,C

verifica1
        addwf aux,1
        btfsc STATUS,C
        goto  inicio1
        btfss TMR0,2
        goto verifica1

        bcf   STATUS,C
        movlw 0xF7   ; verifica se a freq. é menor que 180 khz e maior que 165
```

```
        subwf aux,0
        btfss STATUS,C
        goto inicio1


tstcon  nop
        ;Programação do Timer0 para medição do período de repetição
        bsf    STATUS,RP0; Bank 1
        movlw          0xE5      ; Palavra de programação 1:64
        movwf          OPTION_REG ; Programa palavra de controle
        bcf    OPTION_REG,T0CS; Liga Timer0
        bcf    STATUS,RP0; Bank 0

        ;Medição do intervalo
        clrf   TMR0            ; Limpa Timer0
        bcf    INTCON,T0IF     ; Limpa flag de overflow
        btfss  INTCON,T0IF     ;
        goto   $-1             ; Espera Timer0 estourar


        ; realiza pwm

        movf  TRAINL,0
        movwf          TRAINL1
        movf  tl,0
        movwf          tl1
        movf  th,0
        movwf          th1


pwm1

        bsf    PORTA,2
        decfsz         th1
        goto   $-1
        nop
        movf  tl,0
        movwf          tl1
        movf  th,0
        movwf          th1
        bcf    PORTA,2
        decfsz         tl1
        goto$-1
        nop
        decfsz         TRAINL1
        goto   pwm1

        btfss  PIR1,RCIF ; Testa se recebeu comando de parada
        goto   pgd       ; Continua no teste se não recebeu nenhum comando

        movf  RCREG,0          ; Leitura do comando
        movwf          TEMP0   ; Armazenamento temporário

        movlw          0x5A    ; Código ASCII do comando Z (Encerrar teste)
        subwf TEMP0,0  ; Realiza comparação por subtração
        btfss  STATUS,Z  ; Testa o resultado da operação
        goto   pgd       ; Continua com o teste se não recebeu comando de parada

        sndbt 0x50     ; código de palavra
        sndbt 0x54     ; 'T'
```

```
            sndbt 0x65    ; 'e'
            sndbt 0x73    ; 's'
            sndbt 0x74    ; 't'
            sndbt 0x65    ; 'e'
            sndbt 0x20    ; ' '
            sndbt 0x45    ; 'E'
            sndbt 0x6E    ; 'n'
            sndbt 0x63    ; 'c'
            sndbt 0x65    ; 'e'
            sndbt 0x72    ; 'r'
            sndbt 0x72    ; 'r'
            sndbt 0x61    ; 'a'
            sndbt 0x64    ; 'd'
            sndbt 0x6F    ; 'o'
            sndbt 0x00    ; 'null'

            return        ; Encerra o teste

;*************************************************************************
SET_V   ;Programa nova velocidade do som

            rcvbt  VSOUND0
            rcvbt  VSOUND1
            rcvbt  VSOUND2

            ;Confirmação à unidade de comando
            sndbt 0x50    ; código de palavra
            sndbt 0x4F    ; 'O'
            sndbt 0x4B    ; 'K'
            sndbt 0x00    ; 'null'

            return


;*************************************************************************
            end
```

Anexo F – Programa do PIC para a unidade fixa

;Programa do PIC - sistema de localização - Unidade Fixa

```
;Pino de entrada: RA4 - pino 06
;O PIC analisa a frequência do sinal de entrada deste pino e responde se for o sinal
; de resposta esperado.

;A resposta é composta de um tempo de espera determinado e
;um trem de pulsos com período, largura de pulso e duração
;programados. O tempo de espera é necessário para que a unidade móvel esteja pronta
;para receber o sinal de resposta. Tempo de espera 256*64 ciclos de máquina
;
;Pino de saída do trem de pulsos: RB2 pino 23
;
;Valores de período, largura de pulso e duração do trem devem ser
;  modificados no programa

        list p=16c73b
        include <p16c73b.inc>

;******* Registradores Utilizados - Bank 0 - 20h a 7Fh **********************
;Para a geração do trem de pulsos
TRAINL      equ 0x22 ;número de pulsos
aux         equ 0x21 ;contador de tempo para descobrir a freq. do sinal de entrada

;********************* PROGRAMA PRINCIPAL ***********************************
        org 0x00 ; Define endereco base no vetor de inicio 0x00
reset
        goto start ; Vai para o inicio do programa

        org 0x04   ; Endereco de início do codigo do usuario

start


;************* Programações Default do timer ****************

        ;Programação do Timer0 - Controla o comprimento do trem
        bcf    STATUS,RP0 ; Bank 0
        bcf    INTCON,T0IE; Desarma interrupção
        bcf    INTCON,T0IF; Limpa flag de overflow
        bsf    STATUS,RP0 ; Bank 1
        movlw       B'10101000' ; Palavra de programação
        movwf       OPTION_REG ; Programa palavra de controle


;********************** Loop de Espera ***********************************


        bsf    STATUS,RP0; Bank 1

        bsf TRISA,4    ;  habilita o pino como entrada
        bcf TRISB,2   ; habilita pino como saída

        bcf    STATUS,RP0; Bank 0

inicio
```

```
            clrf    TMR0    ; limpa Timer0
            clrf    aux

    espera
            btfss   TMR0,0          ; espera o sinal externo ativar o clock
            goto    espera  ;



    ;           através de aux cronometra-se o número de ciclos executados na freq. do PIC, até que
    ;a 3 borda de subida
    ;           do clock externo ocorra
            movlw 0x06

            bcf   STATUS,C
    verifica
            addwf aux,f
            btfsc STATUS,C
            goto  inicio
            btfss TMR0,2
            goto verifica

            bcf    STATUS,C
            movlw 0x61    ; verifica se a freq. é menor que 158 khz
            subwf aux,0
            btfss STATUS,C
            goto inicio
            movlw 0x6F    ; verifica se a freq. é maior ou igual a 135 khz
            subwf aux,0
            btfsc STATUS,C
            goto inicio
            call RESPOSTA
            goto start
    ;******************** ROTINA DE EXECUÇÃO DO COMANDO **********************

    ; Espera um tempo determinado (256*64*2e-7 segundos) e envia o sinal de resposta

    RESPOSTA

            ;Para teste de emissão
            bsf    STATUS,RP0; Bank 1
            movlw          B'11100011'   ; Palavra de programação  1:16
            movwf          OPTION_REG ; Programa palavra de controle
            bcf    OPTION_REG,T0CS; Liga Timer0
            bcf    STATUS,RP0; Bank 0

            clrf    TMR0            ; Limpa Timer0
            bcf     INTCON,T0IF     ; Limpa flag de overflow
            btfss   INTCON,T0IF     ;
            goto    $-1             ; Espera Timer0 estourar

            movlw          0x08
            movwf          TRAINL

pwm
            bsf    PORTB,2
            nop
            nop
            nop
            nop
```

```
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        bcf    PORTB,2
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        nop
        decfsz TRAINL,1
        goto   pwm

        ;Retorna ao Loop de espera
        return
;*****************************************************************************
        end
```

Anexo G – Programa de interface com o PIC - Unidade de Comando (PC), para o tratamento analógico de sinal

```
 * A unidade de comando envia comandos ao PIC e espera por
 * confirmações ou respostas.

 * É verificada a validade de cada comando. Sao enviados ao PIC
 * somente comandos validos.
 * A sintaxe de cada comando é a seguinte:
 * (letra do comando)<argumento - se necessario> (sem espaco em branco)

 * Para o comando G (ordem para localização), a resposta esperada
 * é a distância em milimetros entre a unidade móvel e a fixa.
 * Para o comando S (verificar parametros), a resposta esperada
 * uma lista de parametros numa ordem pre-estabelecida.
 * O comando I (inicializacao)realiza a reinicializacao da comunicacao em caso de
 * reset do PIC.
 * O comando T (modo de teste), aciona o modo de teste, isto e',
 * espera por valores de distancia enviados pelo pic regularmente.
 * Para os demais comandos, é esperada a confirmação 'OK'.

 * O comando é enviado para o PIC como uma palavra de ate 32 bits,
 * que consiste de comando (8 bits - c¢digo ASCII) e argumento (ate 24 bits).

 * Quando a maioria dos comandos é executada, uma resposta do PIC e
 * esperada. Se a resposta nao chega num certo intervalo de tempo,
 * o programa julga que ocoreu um erro.

 */

#include<stdio.h>
#include<string.h>
#include<dos.h>
#include<conio.h>

/* DEFINICOES */
#define ESPERA 2600000/* No. limite de iteracoes de espera */

/* VARIAVEIS GLOBAIS */
int PORT;                /* Endereco da porta serial escolhida */

/* ROTINAS */
void InicializaPorta(void)
/* Prepara a porta serial para comunicacao */
{
        outp(PORT + 3 , 0x00); /* Limpa DLAB - Acesso ao IER          */
        outp(PORT + 1 , 0x00); /* Desabilita interrupções             */
        outp(PORT + 3 , 0x80); /* Seta DLAB - Acesso ao Divisor Latch */
        outp(PORT + 0 , 0x0C); /* Set Baud rate - Divisor Latch Low Byte */
                               /*       0x03 =  38,400 BPS            */
                               /*       0x01 = 115,200 BPS            */
                               /*       0x02 =  57,600 BPS            */
                               /*       0x06 =  19,200 BPS            */
                               /*    -> 0x0C =   9,600 BPS            */
                               /*       0x18 =   4,800 BPS            */
                               /*       0x30 =   2,400 BPS            */
        outp(PORT + 1 , 0x00); /* Set Baud rate - Divisor Latch High Byte */
        outp(PORT + 3 , 0x03); /* 8 Bits, Sem paridade, 1 Stop Bit    */
        outp(PORT + 2 , 0xC7); /* FIFO Control Register               */
```

```
        outp(PORT + 4 , 0x0B); /* Seta DTR, RTS, e OUT2                */
}

int Espera_Transmissor(void)
/* Espera o buffer de transmissao esvaziar */
{
        long int i;         /* Contador */

        /* Espera ate que o flag Data Ready esteja setado */
        for(i=0;i<ESPERA && ((inp(PORT+5)/32)%2)==0;i++);

        if(i<ESPERA)
              return 1;/* Transmissor preparado */
        else
              return 0;/* Algum erro ocorreu    */
}

int Espera_Dado(void)
/* Espera que o dado esteja disponivel para leitura da porta */
{
        long int i;         /* Contador */

        for(i=0;i<ESPERA && (inp(PORT+5)%2)==0;i++);

        if (i<ESPERA)
              return 1; /* Dado preparado */
        else
              return 0; /* time out - dado nao chegou */
}

void Recebe_Resposta(void)
{/* Recebe uma resposta do PIC segundo as seguintes regras:
        - Uma resposta em ASCII deve ser precedida pelo codigo 0x50 e deve
        se encerrar com o codigo 0x00
        - Uma resposta numerica deve ser precedida pelo codigo 0x4E e pelo
        numero de bytes da resposta
*/
  int     i;
  int     flagR;
  short   word;
  int     cont;
  short   resposta[30];

  flagR=Espera_Dado();
  if(flagR==0)
    printf("\nNao ha resposta\n");
  else
  {
    word=inp(PORT);
    if(word==0x50)
    {
      for(i=0;word!=0x00&&flagR==1;i++)
      {
            flagR=Espera_Dado();
            word=inp(PORT);
            resposta[i]=word;
          }
          if(flagR==1)
            {
              i=0;
```

```
                printf("\n");
                while (resposta[i]!=0x00)
                {
                    printf("%c",resposta[i]);
                    i++;
                }
                  printf("\n");
              }
            else
              printf("\nFormato dos dados recebidos e invalido\n");
      }/* Do if */
      else
        if(word==0x4E)
        {
              flagR=Espera_Dado();
              if(flagR==0)
                printf("\nFormato dos dados recebidos e invalido\n");
              else
              {
                cont=inp(PORT);
                for(i=0;i<cont&&flagR==1;i++)
                {
                    flagR=Espera_Dado();
                    resposta[i]=inp(PORT);
                }
                if(flagR==0)
                  printf("\nFormato dos dados recebidos e invalido\n");
                else
                {
                    printf("\n");
                    for(i=0;i<cont;i++)
                    {
                       if(resposta[i]<16)
                          printf("0"); /* Correcao da impressao */
                      printf("%x",resposta[i]);
                    }
                    printf("\n");
                }
              }
            }/* Do if */
            else
              printf("\nFormato dos dados recebidos e invalido\n");
   }/* Do else */
}/* Da funcao */

void Envia_Mensagem(char mensagem[])
{/* Envia um vetor de tamanho definido ao PIC.
   O tamanho do vetor esta contido na posicao '0'.
   O PIC deve interpretar os dados.
*/
        int tamanho;
        int i;
        int flagR;

        flagR=1;
        tamanho=mensagem[0];
        i=1;
        while((i<=tamanho)&&(flagR==1))
        {
              flagR=Espera_Transmissor();
```

```
                outp(PORT,mensagem[i]);
                i++;
            }
            if(tamanho<1)
              printf("\nFormato de dados invalido\n");
            else
              if(flagR==0)
                printf("\nProblemas na transmissao de dados. Transmissor nao libera\n");
}

void locate(void)
/* Envia ordem de realizar a localizacao e espera pelo valor da distancia */
{
        char   mensagem[30];

        mensagem[0]=1;   /* Tamanho da mensagem */
        mensagem[1]=0x47; /* Codigo da letra de comando - G */

        Envia_Mensagem(mensagem);

        printf("\nDistancia (em 10e-8mm em hexadecimal): ");
        Recebe_Resposta();
}

void help(void)
{/* Exibe lista de comandos */
        printf("\nLISTA DE COMANDOS:");
        printf("\nComando\t\t\tSintaxe\t\tUnidade\n");
        printf("-----------------------------------------------");
        printf("\nLocalizar\t\tG\t\t-");
        printf("\nProgramar velocidade\tV<velocidade>\t10e-1mm/s");
        printf("\nProgramar periodo\tP<periodo>\tNo. de ciclos");
        printf("\nProgramar largura\tW<largura>\tNo. de ciclos");
        printf("\nProgramar comprimento\tL<comprimento>\tNo. de ciclos");
        printf("\nReiniciar comunicacao\t\t\t-");
        printf("\nLer Pino\t\tA\t\t-");
        printf("\nVerificar parametros\tS\t\t-");
        printf("\nTeste: recepcao-emissao\tT\t\t-");
        printf("\nTeste: emissao\tB\t\t-");
        printf("\nAjuda\t\t\t?\t\t-");
        printf("\nSair do programa\tQ\t\t-");
        printf("\n\nOBSERVAÇÕES:\n");
        printf("1. Não use espaço em branco\n");
        printf("2. Use letras maiúsculas ou min£sculas\n");
        printf("3. Não use ponto flutuante\n");
}

void setvelocity(char comando[])
{/* Programa velocidade do som (em 10e-1mm/s) */
        int     i   ; /* contador */
        long int V   ; /* velocidade do som em 10^(-1)mm/s */
        char    mensagem[30];

        mensagem[0]=4;      /* Tamanho da mensagem */
        for(i=1,V=0;i<strlen(comando);i++)
              V=(V)*10+(comando[i]-48);
        if((V<3000000)||(V>16777215))
              printf("\nArgumento fora da faixa permitida");
        else
        {
```

```
                mensagem[1]=0x56;       /* codigo da letra de comando */
                mensagem[2]=V%(0x100);  /* LSB da velocidade          */
                V=V/(0x100);
                mensagem[3]=V%(0x100);
                V=V/(0x100);
                mensagem[4]=V%(0x100);  /* MSB da velocidade          */

                Envia_Mensagem(mensagem);

                Recebe_Resposta();

        }/* end else */
}




void Verifica_Parametros(void)
{/* Imprime velocidade do som, periodo, largura de pulso e comprimento do
  trem programados no PIC
*/
        char  mensagem[30];

        mensagem[0]=1;          /* Tamanho da mensagem */
        mensagem[1]=0x53; /* Codigo da letra de comando - S */

        Envia_Mensagem(mensagem);

        printf("\nVelocidade: ");
        Recebe_Resposta();


}


void Teste(void)
{/* Controla o modo de teste */
        char  ch;

        outp(PORT,0x54);
        Recebe_Resposta();
        for(;ch!=27;)
        {
           if(kbhit())  ch=getch();
           if(ch==27)
           {
                outp(PORT,0x5a);  /*Mensagem para encerrar o teste*/
                Recebe_Resposta();
           }

        }
}


/****** PROGRAMA PRINCIPAL ******/
void main (void)
{
        /* Vari veis */
        char  comando[10]; /* Comando do usuario */
```

```
int    flag;       /* Indicador geral */
int    p;            /* No. da porta de comuniçaõ */
int    inicio;     /* Codigos de inicio */
int    inicializado;/* Indica inicializacao da comunicacao */
char  ch;

/* Inicializacoes das variaveis */
comando[0]=00;
inicio=0;
ch=0;
inicializado=0;

printf("\n\n Programa de comando do PIC - Comunicacao Serial\n\n");

for(;inicializado==0;) /* Loop de inicializacao */
{
        p=0;
        for(;p!=1 && p!=2 && p!=3 && p!=4;)
        {
                printf("\nPorta de comunicação (1,2,3,4): ");
                scanf("%d",&p);
        }
        if(p==1) PORT=0x3f8;
        if(p==2) PORT=0x2f8;
        if(p==3) PORT=0x3e8;
        if(p==4) PORT=0x2e8;

        printf("\nEndereço: %x\n",PORT);

        InicializaPorta();  /* Preparacao da porta serial */

        /* Inicializacao da comunicacao */
        Espera_Transmissor();
        outp(PORT,0xAA);      /* Envia codigo de inicio */
        flag=Espera_Dado();
        inicio=inp(PORT);
        if(flag==1&&inicio==0xBB)
        {
                printf("\nComunicacao inicializada com sucesso\n");
                inicializado=1;
        }
        else
                printf("\nFalha na inicializacao da comunicacao\n");
}/* Fim do loop de inicializacao */

/* Loop Principal */
while((comando[0]!='q')&&(comando[0]!='Q'))
{
        outp(PORT + 2 , 0xC7);/* Limpa os FIFO's da porta serial */
        flag=0;
        printf("\nReady:) "); /* Prompt */
        scanf("%s",comando);
        if((comando[0]=='G')||(comando[0]=='g'))
        {
                locate(); /* Ordem para localizacao */
                flag=1;   /* Seta flag de passagem */
        }
        if((comando[0]=='S')||(comando[0]=='s'))
        {
                Verifica_Parametros();
```

```
                flag=1;  /* Seta flag de passagem */
        }
        if((comando[0]=='T')||(comando[0]=='t'))
        {
                Teste(); /* Inicia modo de teste */
                flag=1;  /* Seta flag de passagem */
        }

        if((comando[0]=='I')||(comando[0]=='i'))
        {
                Espera_Transmissor();
                outp(PORT,0xAA);     /* Envia codigo de inicio */
                flag=Espera_Dado();
                inicio=inp(PORT);
                if(flag==1&&inicio==0xBB)
                {
                        printf("\nComunicacao reinicializada\n");
                        inicializado=1;
                }
                else
                        printf("\nFalha na comunicacao\n");
                flag=1;  /* Seta flag de passagem */
        }

        if (comando[0]=='?')
        {
                help();     /* Mostra lista de comandos */
                flag=1;
        }

        if((comando[0]=='V')||(comando[0]=='v'))
        {
                setvelocity(comando);/* Programa veloc. do som */
                flag=1;
        }

        if((comando[0]=='Q')||(comando[0]=='q'))
                flag=1;

        if (flag==0) /* Se nao houve nenhum comando valido */
                printf("\nComando inexistente. Digite ? para ajuda\n");

    } /* Fim do while */

} /* Fim do main */
```

Anexo H – Programa de interface com o PIC - Unidade de Comando (PC), para o tratamento digital de sinais

```
/* Programa de testes da placa A/D


 * O PC envia comandos ao PIC e espera por
 * confirmacoes ou respostas.

 * E verificada a validade de cada comando. Sao enviados ao PIC
 * somente comandos validos.

 * Para o comando V (verificar status), a resposta esperada
 * e a palavra de programacao atual e o status das FIFOs.
 * O comando I (inicializacao) realiza a reinicializacao da comunicacao
 * em caso de reset do PIC.
 * Para os demais comandos, e esperada a confirmacao 'OK'.

 * Quando a maioria dos comandos e executada, uma resposta do PIC e
 * esperada. Se a resposta nao chega num certo intervalo de tempo,
 * o programa julga que ocoreu um erro.

 */

#include<stdio.h>
#include<string.h>
#include<dos.h>
#include<conio.h>

/* DEFINICOES */
#define ESPERA 2600000/* No. limite de iteracoes de espera */

/* VARIAVEIS GLOBAIS */
int PORT;                   /* Endereco da porta serial escolhida */

/* ROTINAS */
void InicializaPorta(void)
/* Prepara a porta serial para comunicacao */
{
        outp(PORT + 3 , 0x00); /* Limpa DLAB - Acesso ao IER          */
        outp(PORT + 1 , 0x00); /* Desabilita interrupções             */
        outp(PORT + 3 , 0x80); /* Seta DLAB - Acesso ao Divisor Latch */
        outp(PORT + 0 , 0x0C); /* Set Baud rate - Divisor Latch Low Byte */
                               /*          0x03 =  38,400 BPS         */
                               /*       0x01 = 115,200 BPS            */
                               /*       0x02 =  57,600 BPS            */
                               /*       0x06 =  19,200 BPS            */
                               /*    -> 0x0C =   9,600 BPS            */
                               /*       0x18 =   4,800 BPS            */
                               /*       0x30 =   2,400 BPS            */
        outp(PORT + 1 , 0x00); /* Set Baud rate - Divisor Latch High Byte */
        outp(PORT + 3 , 0x03); /* 8 Bits, Sem paridade, 1 Stop Bit    */
        outp(PORT + 2 , 0xC7); /* FIFO Control Register               */
        outp(PORT + 4 , 0x0B); /* Seta DTR, RTS, e OUT2               */
}

int Espera_Transmissor(void)
/* Espera o buffer de transmissao esvaziar */
{
        long int i;         /* Contador */
```

```
        /* Espera ate que o flag Data Ready esteja setado */
        for(i=0;i<ESPERA && ((inp(PORT+5)/32)%2)==0;i++);

        if(i<ESPERA)
            return 1;/* Transmissor preparado */
        else
            return 0;/* Algum erro ocorreu   */
}

int Espera_Dado(void)
/* Espera que o dado esteja disponivel para leitura da porta */
{
        long int i;         /* Contador */

        for(i=0;i<ESPERA && (inp(PORT+5)%2)==0;i++);

        if (i<ESPERA)
            return 1; /* Dado preparado */
        else
            return 0; /* time out - dado nao chegou */
}

void Recebe_Resposta(void)
{/* Recebe uma resposta do PIC segundo as seguintes regras:
        - Uma resposta em ASCII deve ser precedida pelo codigo 0x50 e deve
        se encerrar com o codigo 0x00
        - Uma resposta numerica deve ser precedida pelo codigo 0x4E e pelo
        numero de bytes da resposta
*/
  int   i;
  int   flagR;
  int   word;
  int   cont;
  int   resposta[30];

  flagR=Espera_Dado();
  if(flagR==0)
    printf("\nNao ha resposta\n");
  else
  {
    word=inp(PORT);
    if(word==0x50)
    {
      for(i=0;word!=0x00&&flagR==1;i++)
      {
          flagR=Espera_Dado();
          word=inp(PORT);
          resposta[i]=word;
      }
      if(flagR==1)
        {
          i=0;
          printf("\n");
          while (resposta[i]!=0x00)
          {
              printf("%c",resposta[i]);
              i++;
          }
           printf("\n");
```

```
            }
          else
            printf("\nFormato dos dados recebidos e invalido\n");
      }/* Do if */
      else
        if(word==0x4E)
        {
            flagR=Espera_Dado();
            if(flagR==0)
              printf("\nFormato dos dados recebidos e invalido\n");
            else
            {
              cont=inp(PORT);
              for(i=0;i<cont&&flagR==1;i++)
              {
                  flagR=Espera_Dado();
                  resposta[i]=inp(PORT);
              }
              if(flagR==0)
                printf("\nFormato dos dados recebidos e invalido\n");
              else
              {
                  printf("\n");
                  for(i=0;i<cont;i++)
                  {
                     if(resposta[i]<16)
                        printf("0"); /* Correcao da impressao */
                    printf("%x",resposta[i]);
                  }
                  printf("\n");
              }
            }
          }/* Do if */
          else
            printf("\nFormato dos dados recebidos e invalido\n");
   }/* Do else */
}/* Da funcao */

void Envia_Mensagem(int mensagem[])
{/* Envia um vetor de tamanho definido ao PIC.
   O tamanho do vetor esta contido na posicao '0'.
   O PIC deve interpretar os dados.
*/
        int tamanho;
        int i;
        int flagR;

        flagR=1;
        tamanho=mensagem[0];
        i=1;
        while((i<=tamanho)&&(flagR==1))
        {
            flagR=Espera_Transmissor();
            outp(PORT,mensagem[i]);
            i++;
        }
        if(tamanho<1)
          printf("\nFormato de dados invalido\n");
        else
          if(flagR==0)
```

```
            printf("\nProblemas na transmissao de dados. Transmissor nao libera\n");
}

void help(void)
{/* Exibe lista de comandos */
        printf("\nLISTA DE COMANDOS:");
        printf("\nComando\t\t\tSintaxe\t\tUnidade\n");
        printf("-----------------------------------------------");
        printf("\nProgramar Frequencia\tF\t\t<Calculado pelo Bitcalc>");
        printf("\nRealizar Localizacao\tG\t\t-");
        printf("\nLimpar FIFOs\t\tL\t\t-");
        printf("\nReinicializacaoL\tI\t\t-");
        printf("\nTeste\tT\t\t-");
        printf("\nRealizar leitura da fifos\tR\t\t-");
        printf("\nVerificar status\tV\t\t-");
        printf("\nAjuda\t\t\t?\t\t-");
        printf("\nSair do programa\tQ\t\t-");
        printf("\n\nOBSERVACOES:\n");
        printf("1. Nao use espaco em branco\n");
        printf("2. Use letras maiusculas ou minusculas\n");
}

void setfrequency(void)
{/* Programa nova frequencia */
        long int word ; /* palavra de programacao */
        int     mensagem[30];

        mensagem[0]=4;                  /* Tamanho da mensagem */
        mensagem[1]=0x46;           /* codigo da letra de comando (F)*/
        printf("\nPalavra de programacao: ");
        scanf("%lx",&word);
        mensagem[2]=word%(0x100);       /* LSB da palavra  */
        word=word/(0x100);
        mensagem[3]=word%(0x100);
        word=word/(0x100);
        mensagem[4]=word;           /* MSB da palavra  */

        Envia_Mensagem(mensagem);

        Recebe_Resposta();
}


void limpar(void)
{/* Limpa as FIFOs*/
        int mensagem[2];
        mensagem[0]=1;              /* Tamanho da mensagem */
        mensagem[1]=0x4C;     /* Codigo da letra de comando - L */
        Envia_Mensagem(mensagem);

}

void locate(void)
/* Envia ordem de realizar a localizacao e espera pelo valor da distancia */
{
        int  mensagem[2];

        mensagem[0]=1;   /* Tamanho da mensagem */
        mensagem[1]=0x47; /* Codigo da letra de comando - G */
```

```
        Envia_Mensagem(mensagem);
        Recebe_Resposta();

}

 void Read(void)
{/* Le as FIFOs e grava num arquivo texto */
        int mensagem[2];
        int    cont;
        int    flag;
        int    flag1;
        int    flag2;
        int    flag3;
        int    lsb;
        int    msb;
        int    word1;
        int    word2;
        int    word;
        FILE *fp;

        mensagem[0]=1;              /* Tamanho da mensagem */
        mensagem[1]=0x52;    /* Codigo da letra de comando - R */
        Envia_Mensagem(mensagem);

       /* Abre o arquivo */
       fp=fopen("dados.txt","w");

       cont=0x00;
       flag=1;
       flag1=1;
       /* Loop de leitura */
       while(cont<=0x5000 && flag1==1)
       {

            flag1=1;
           flag2=1;
            flag3=1;
            while(flag1==1&&flag2==1&&flag3==1)
            {
                    flag1=Espera_Dado();
                    word1=inp(PORT);
                    flag2=Espera_Dado();
                    word2=inp(PORT);
                    if(word1==0x4E && word2==0x02 && flag1==1 && flag2==1)
                            flag3=0;
            }

           if(flag3==0)
           {
            flag=Espera_Dado();
            if(flag==1)
            {
                     lsb=inp(PORT);
            }
            flag=Espera_Dado();
            if(flag==1 && flag1==1 && flag2==1)
            {
                    msb=inp(PORT);
                    word=0x100*msb+lsb;
```

```
                        fprintf(fp,"%d",word);
                    }
                    fprintf(fp,"\n");
                    cont++;
                    Espera_Transmissor();
                    outp(PORT,0xCC);
                  }
                 else
                  {
                    Espera_Transmissor();
                    outp(PORT,0xDD);
                  }
            }
            /* Fecha o arquivo */
            fclose(fp);
  }

  void Teste(void)
  {/* Controla o modo de teste */
            char  ch;

            outp(PORT,0x54);
            Recebe_Resposta();
            for(;ch!=27;)
            {
               if(kbhit())  ch=getch();
               if(ch==27)
               {
                     outp(PORT,0x5a);  /*Mensagem para encerrar o teste*/
                     Recebe_Resposta();
               }

            }
  }

 void verify(void)
 {/* Verifica palavra de programacao e status das FIFOs, valores de tl, th e TRAINL */
            int     mensagem[30];

            mensagem[0]=1;          /* Tamanho da mensagem */
            mensagem[1]=0x56; /* Codigo da letra de comando - V */

            Envia_Mensagem(mensagem);

            printf("\nPalavra de programacao: ");
            Recebe_Resposta();

            printf("\nStatus das FIFOs: ");
            Recebe_Resposta();


 }



/****** PROGRAMA PRINCIPAL ******/
 void main (void)
 {
          /* Variaveis */
          char  comando[10]; /* Comando do usuario */
```

```
int    flag;       /* Indicador geral */
int    p;          /* No. da porta de comunicacao */
int    inicio;     /* Codigos de inicio */
int    inicializado;/* Indica inicializacao da comunicacao */
char   ch;

/* Inicializacoes das variaveis */
comando[0]=0;
inicio=0;
ch=0;
inicializado=0;

printf("\n\n Programa de testes da placa AD\n\n");

for(;inicializado==0;) /* Loop de inicializacao */
{
        p=0;
        for(;p!=1 && p!=2 && p!=3 && p!=4;)
        {
                printf("\nPorta de comunicacao (1,2,3,4): ");
                scanf("%d",&p);
        }
        if(p==1) PORT=0x3f8;
        if(p==2) PORT=0x2f8;
        if(p==3) PORT=0x3e8;
        if(p==4) PORT=0x2e8;

        printf("\nEndereco: %x\n",PORT);

        InicializaPorta();  /* Preparacao da porta serial */

        /* Inicializacao da comunicacao */
        Espera_Transmissor();
        outp(PORT,0xAA);     /* Envia codigo de inicio */
        flag=1;
        inicializado=0;
        while(flag==1&&inicio!=0xBB)
        {
                flag=Espera_Dado();
                inicio=inp(PORT);
        }
        if(flag==1)
        {
                printf("\nComunicacao OK\n");
                inicializado=1;
        }
        else
                printf("\nFalha na comunicacao\n");

}/* Fim do loop de inicializacao */

/* Loop Principal */
while((comando[0]!='q')&&(comando[0]!='Q'))
{
        outp(PORT + 2 , 0xC7);/* Limpa os FIFO's da porta serial */
        flag=0;
        printf("\n>> "); /* Prompt */
        scanf("%s",comando);
        if((comando[0]=='l')||(comando[0]=='i'))
        {
```

```
            Espera_Transmissor();
            outp(PORT,0xAA);     /* Envia codigo de inicio */
            flag=1;
            inicio=0;
            while(flag==1&&inicio!=0xBB)
            {
                  flag=Espera_Dado();
                  inicio=inp(PORT);
            }
            if(flag==1)
            {
                  printf("\nComunicacao reinicializada\n");
                  inicializado=1;
            }
            else
                  printf("\nFalha na comunicacao\n");
          flag=1; /* Seta flag de passagem */
 }
 if((comando[0]=='F')||(comando[0]=='f'))
 {
          setfrequency();
          flag=1; /* Seta flag de passagem */
 }
 if((comando[0]=='R')||(comando[0]=='r'))
 {

          Read();          /* Realiza leitura das FIFOs */
          flag=1; /* Seta flag de passagem */
 }
 if((comando[0]=='V')||(comando[0]=='v'))
 {
          verify();          /* Verifica Status */
          flag=1;
 }
 if((comando[0]=='G')||(comando[0]=='g'))
 {
          locate();          /* Realiza amostragem de dados */
          flag=1;
 }
 if((comando[0]=='L')||(comando[0]=='l'))
 {
          limpar();          /* Limpa FIFOs */
          flag=1;
 }

 if((comando[0]=='T')||(comando[0]=='t'))
 {
          Teste(); /* Inicia modo de teste */
          flag=1;  /* Seta flag de passagem */
 }


 if (comando[0]=='?')
 {
          help(); /* Mostra lista de comandos */
          flag=1;
 }
 if((comando[0]=='Q')||(comando[0]=='q'))
          flag=1;
```

```
            if (flag==0) /* Se nao houve nenhum comando valido */
                printf("\nComando inexistente. Digite ? para ajuda\n");

        } /* Fim do while */

} /* Fim do main */
```

Anexo I – Programa de correlação de sinais utilizando a Transformada Rápida de Fourier (FFT)

```
clear all
close all


tam=1/(7000000); % tempo de amostragem utilizado


% sinal de refererencia

% Cria o sinal de referência a partir do sinal de resposta proveniente
% da unidade móvel,
% cuja aquisição foi realizada através da placa AD.

% carrega o arquivo txt criado pelo software de comunicação serial.

load dados1.txt
sinal=dados1';
[n,m]=size(sinal);
cont=1;

% Inversão do bit mais significativo

for cont=1:m
   if sinal(cont)<2048
   sinal(cont)=sinal(cont)+2048;
   else
   sinal(cont)=sinal(cont)-2048;
   end
    cont=cont+1;
end

% Correção de offset e escala
sinal=sinal/4095*2;
sinal=sinal-1;
ref=sinal(12040:12550); % o sinal de referência é obtido visualmente a
                                    %partir do sinal de resposta
[n,mr]=size(ref);
ts=0:tam:(m-1)*(tam);
tsr=ts(12040:12550);
plot(tsr,ref,'b');

% dados da placa AD

load dados.txt
sinal=dados';
[n,m]=size(sinal);
cont=1;

% Inversão do bit mais significativo
for cont=1:m
   if sinal(cont)<2048
   sinal(cont)=sinal(cont)+2048;
   else
   sinal(cont)=sinal(cont)-2048;
   end
```

```
    cont=cont+1;
end

% Correção de offset e escala
sinal=sinal/4095*2;
sinal=sinal-1;

ts=0:tam:(m-1)*(tam);
figure
plot(ts,sinal,'b');

% realiza correlação

S = fft(sinal, 32*1024); % transformada de fourier do sinal. Usa-se 32 bytes, pois a taxa de
amostragem do sinal
                                % amostrado deve ser pelo menos duas vezes maior que a
freqüência do sinal original.
R = fft(ref,32*1024);% transformada de fourier do sinal de referência
COR=S .* conj(R); % efetua correlação
cor = ifft(COR); % transformada inversa de Fourier
plot(abs(cor))
acor = abs(cor);
[linha,coluna]=size(acor);
maior(3)=acor(1);

% determinação dos três maiores pontos da curva

for i=1:coluna
  if acor(i)>maior(3)
    maior(3)=acor(i);
    ponto(1)=i*tam;
  else
    if acor(i)>maior(2);
      maior(2)=acor(i);
      ponto(2)=i*tam;
    else
      if acor(i)>maior(1);
        maior(1)=acor(i);
        ponto(3)=i*tam;
      end
    end
  end
end

p=polyfit(ponto,maior,2); % realiza a interpolação de segunda ordem
                                % dos três maiores pontos
ponto_corr=-p(2)/(2*p(1));%determinação do máximo da curva interpolada

% cálculo da distância, realizando os descontos necessários devido aos
% atrasos existentes nos microcontroladores das unidades

vsound=1400;
distancia=(vsound*(ponto_corr-(4232-788)*(1/5000000)))/2
```

Referências Bibliográficas

1. Kino, Gordon S. – ***Acoustic waves: Devices, imaging and analog signal processing*** - Prentice-Hall – 1987

2. Kinsler, Lawrence E. [et al] - ***Fundamentals of Acoustics*** - 3rd ed. - New York - Wyley - 1982

3. ***Datasheet da família pic16cXX*** – fornecido pela Microchip Technology - Contém detalhes do uso do PIC e de sua programação.

4. Flannery, B. P.; Press, W. H.; Teukolsky, S. A.; Vetterling, W. T. **Numerical Recipes in C:** The Art of Scientific Computing. 2 ed. Cambridge.

5. Malvino A. P. – **Eletrônica,** Makron Books, 4ª edição, 1997, volumes I e II.

6. Shorrock, G.; Woodward, B. A Multiple Function Sonar Rangefinder for Divers. **Ultrasonics,** p. 16-24, Jan. 1984.

Bibliografia Recomendada

1. ***Manual de instruções do compilador Pacific C*** – download do site da internet www.htsoft.com

2. ***Manual de instruções do MPLAB*** – fornecido pela Microchip Technology

3. De Sousa, D. J. - ***Desbravando o PIC,*** - Erica, 2ª edição, 2000.