UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA
CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES

MARCELO MACHADO SANTOS
GUSTAVO HENRIQUE DUARTE

PROJETO DE UM OSCILOSCÓPIO DIGITAL PARA SINAIS DE ATÉ 4MHZ COM SUPORTE A FILTROS FIR E IIR EM TEMPO REAL E ANÁLISE ESPECTRAL

TRABALHO DE CONCLUSÃO DE CURSO

CURITIBA
2013

MARCELO MACHADO SANTOS
GUSTAVO HENRIQUE DUARTE

PROJETO DE UM OSCILOSCÓPIO DIGITAL PARA SINAIS DE ATÉ 4MHZ COM SUPORTE A FILTROS FIR E IIR EM TEMPO REAL E ANÁLISE ESPECTRAL

Trabalho de Conclusão de Curso de Graduação, apresentado ao Curso Superior de Tecnologia em Sistemas de Telecomunicações do Departamento Acadêmico de Eletrônica – DAELN – da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de Tecnólogo.

Orientador: Prof. MSc. César Janeczko

CURITIBA
2013

TERMO DE APROVAÇÃO

MARCELO MACHADO SANTOS
GUSTAVO HENRIQUE DUARTE

PROJETO DE UM OSCILOSCÓPIO DIGITAL PARA SINAIS DE ATÉ 4MHZ COM SUPORTE A FILTROS FIR E IIR EM TEMPO REAL E ANÁLISE ESPECTRAL

Este trabalho de conclusão de curso foi apresentado no dia 29 de agosto de 2013, como requisito parcial para obtenção do título de Tecnólogo em Sistemas de Telecomunicações, outorgado pela Universidade Tecnológica Federal do Paraná. Os alunos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Prof. MSc. César Janeczko
Coordenador de Curso
Departamento Acadêmico de Eletrônica

_____
Prof. Esp. Sérgio Moribe
Responsável pela Atividade de Trabalho de Conclusão de Curso
Departamento Acadêmico de Eletrônica

BANCA EXAMINADORA

_____
Prof. Dr. Luís Alberto Lucas
UTFPR

_____
Prof. MSc. João Almeida de Góis
UTFPR

_____
Prof. MSc. César Janeczko
Orientador – UTFPR

“A Folha de Aprovação assinada encontra-se na Coordenação do Curso”

AGRADECIMENTOS

Ao Professor César Janeczko pela orientação indispensável para a conclusão deste trabalho e por dar sentido prático, ao ensinar Processamento Digital de Sinais, aos conhecimentos adquiridos anteriormente nas disciplinas de Sinais e Sistemas e Matemática Aplicada.

Reverenciamos o Professor Luís Alberto Lucas pela dedicação ao ensino da linguagem Matlab dentro do Processamento Digital de Imagens, sem o qual seria muito difícil concluir os algoritmos desenvolvidos para o osciloscópio.

E ao Professor João Almeida de Góis, pelo ensino das técnicas básicas de construção eletrônica aprendidas nas disciplinas Projeto Integrador I e Introdução à Eletrônica, essenciais para viabilizar a manufatura do projeto.

Também, como não poderia deixar de lembrar, agradecemos o profissionalismo desempenhado por todo o corpo de docentes e demais técnicos em assuntos educacionais da UTFPR, que forneceram apoio fundamental para a conclusão deste curso.

RESUMO

SANTOS, Marcelo Machado; DUARTE, Gustavo Henrique. **Projeto de um osciloscópio digital para sinais de até 4MHz com suporte a filtros FIR e IIR em tempo real e análise espectral**. 2013. 83 páginas. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

Neste trabalho é apresentado o projeto de um osciloscópio digital para sinais de até 4MHz com suporte a filtros FIR e IIR em tempo real e análise espectral. Trata-se de um osciloscópio monocanal que pode ser expandido, trazendo as funcionalidades típicas desta categoria de equipamento, com a característica de ter um custo muito baixo, possibilitando sua construção prática e acessível a estudantes de tecnologia, engenharia ou técnicos de eletrônica. O analisador/osciloscópio funciona em uma configuração de instrumento virtual, isto é, emula um equipamento real através de um circuito conectado a um computador pessoal pela sua porta USB. O projeto foi implementado na linguagem Matlab e tem como base um módulo arduino interligado a uma placa de aquisição de dados que permite a conversão analógica para digital a uma taxa de quarenta milhões de amostras por segundo.

Palavras-chave: Matlab. Arduino. Osciloscópio. Conversor A/D. Instrumentação Virtual.

ABSTRACT

SANTOS, Marcelo Machado; DUARTE Gustavo Henrique, **Design of a digital oscilloscope for signals up to 4MHz with support for FIR and IIR filters in real time and spectral analysis**, 2013, 83 pages. Trabalho de Conclusão de Curso (Curso Superior de Tecnologia em Sistemas de Telecomunicações), Departamento Acadêmico de Eletrônica, Universidade Tecnológica Federal do Paraná. Curitiba, 2013.

This paper presents the design of a digital oscilloscope for signals up to 4MHz with support for FIR and IIR filters in real time and spectral analysis. It is a monochannel oscilloscope that can be expanded and brings the features typical of this category of equipment, with the characteristic of having a very low cost, making it possible to build practical and accessible to students of technology, engineering or electronics technicians. The analyzer / oscilloscope works at a setting of virtual instrument, that is, emulates a real equipment through a circuit connected to a personal computer through its USB port. The design was implemented in Matlab language and is based on a module arduino connected to a data acquisition card which allows the analog to digital conversion at a rate of forty million samples per second.

Keywords: Matlab. Arduino. Oscilloscope. A/D converter. Virtual Instrumentation.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

AC	*Alternate Current*
CMOS	*Complementary Metal Oxide Semiconductor Logic*
CRC	*Cyclic Redundancy Check*
DC	*Direct Current*
FET	*Field Effect Transistor*
FIR	*Finite Impulse Response*
FPGA	*Field Programmable Gate Array*
HDLC	*High Level Data Link Control*
IDE	*Integrated Development Environment*
IIR	*Infinite Impulse Response*
MSPS	*Mega Samples Per Second*
SONAR	*Sound Navigation and Ranging*
TTL	*Transistor Transistor Logic*
UART	*Universal Asynchronous Receiver/Transmitter*
USB	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

O curso de tecnologia na área de telecomunicações deixa claro aos seus alunos a importância do osciloscópio como ferramenta fundamental para manutenção e análise de circuitos eletrônicos, sejam eles digitais ou analógicos.

O projeto aqui apresentado é um trabalho de conclusão de curso que tem como finalidade possibilitar que um computador possa emular um osciloscópio digital na tela do monitor de vídeo respectivo, ou mais precisamente, a construção prática e acessível a estudantes de tecnologia, engenharia ou técnicos de eletrônica, de um analisador de espectro e osciloscópio digital para sinais de até 4 MHz com suporte a filtros *Finite Impulse Response* (*FIR*) e *Infinite Impulse Response* (*IIR*), ambos em tempo real.

É um instrumento importante devido às limitações humanas em perceber os fenômenos eletromagnéticos nos domínios do tempo e da frequência. O osciloscópio é, portanto, uma extensão dos sentidos. Com ele é possível ver variações de tensão no decorrer do tempo que seriam impossíveis naturalmente. Os olhos, por exemplo, podem perceber ondas eletromagnéticas de 5×10^{14} Hz da luz vermelha a 7.5×10^{14} Hz da luz violeta (LONGAIR, 2003). O projeto do osciloscópio aqui proposto tornará as baixas frequências, menores que 4 MHz, visíveis para olhos humanos.

Uma comparação grosseira pode ser feita com instrumentos usados em outras áreas científicas. O osciloscópio é útil para o tecnólogo/engenheiro assim como microscópio é relevante ao biólogo. Da mesma forma pode ser comparado ao papel do telescópio para o astrônomo. A diferença básica é que o microscópio permite a visão de fenômenos biológicos de um microuniverso, o telescópio do macrouniverso, já o osciloscópio garante a visualização do universo das variações elétricas.

2 DESENVOLVIMENTO

2.1. PROBLEMA

Os estudantes do curso de tecnologia em telecomunicações cedo ou tarde encontram o seguinte problema, quando matriculados em cadeiras que envolvem aulas de laboratório com uso de osciloscópio: o circuito não funcionou em tempo hábil, isto é, o tempo de aula não foi suficiente para sanar a pane que impedia a montagem de ter o desempenho exigido. Infelizmente para a maioria dos acadêmicos é inviável continuar a tentativa em casa, pois não possuem osciloscópios. O que fazer?

Estas dificuldades são, em graus variados, recorrentes nas seguintes cadeiras do curso de tecnologia em sistemas de telecomunicações da UTFPR (grade 545): introdução à eletrônica, circuitos elétricos, eletrônica analógica, introdução à eletrônica digital, instrumentos e medidas, fundamentos de comunicação, radiopropagação, eletrônica digital, comunicação de dados, antenas, processamento digital de sinais e sistemas microcontrolados. Mesmo as matérias que não fazem uso direto de aulas de laboratório, nem de osciloscópios ou analisadores de espectro, são em muito facilitadas no que tange ao entendimento da dinâmica de funcionamento do circuito ou sistema, caso o aluno possua este equipamento para o estudo domiciliar.

2.2 JUSTIFICATIVA

A justificativa *a priori* para este trabalho é a divulgação de uma solução prática para o problema exposto no item anterior. Se no início do curso de tecnologia, engenharia ou até em outras áreas que envolvam o uso de osciloscópios como a biologia, medicina ou geologia, os estudantes com alguma habilidade em construção eletrônica puderem montar o projeto aqui exposto, o rendimento das aulas certamente será incrementado pelo acréscimo deste utilíssimo instrumento de medida e visualização. Na biologia e medicina são utilizados osciloscópios que mostram formas de onda com frequências muito baixas, inferiores a 1000 Hz, para monitorar sinais biológicos de origem animal ou humana. São exemplos conhecidos os gráficos de tensão para batimentos cardíacos ou as ondas de origem encefálica.

Na geologia os osciloscópios permitem a medição do tempo de retorno do eco de uma explosão artificial para descobrir a distância até a jazida ou objeto de interesse. Esta técnica, a localização por eco, é análoga a empregada no equipamento *sonar (Sound Navigation and Ranging)* dos submarinos.

Também é necessário justificar aqui que o osciloscópio a ser montado não é um equipamento de desempenho superior aos comerciais. Ele traz o essencial que se espera deste instrumento, que é a visualização da forma de onda no tempo, a estimativa de sua frequência e amplitude. Também possibilitará, da mesma forma que os osciloscópios digitais mais elaborados, gerar um gráfico que representa o espectro de frequências do sinal. Infelizmente a fidelidade desta visualização ou medida será inferior ao equivalente comercial, mas não a ponto de impedir seu uso laboratorial, didático ou científico.

2.3 OBJETIVOS

2.3.1 Objetivo Geral

Demonstrar que é possível e economicamente viável a construção de um osciloscópio digital e analisador de espectro para sinais de até 4 MHz com finalidades estudantis, laboratoriais ou técnicas.

2.3.2 Objetivos Específicos

- Permitir o uso de um microcomputador como um osciloscópio digital;
- Ter recursos semelhantes a um osciloscópio digitalizado comercial;
- O custo deve ser bem menor do que o de um equipamento comercial;
- O objetivo de custo é de até um terço do salário mínimo brasileiro;
- Incrementar a didática das aulas de Processamento Digital de Sinais;
- Programar um microcontrolador em linguagem C no ambiente “Arduino IDE”;
- Desenvolver um osciloscópio digital fazendo uso de linguagem C;
- Implementar um osciloscópio e analisador de espectro na linguagem MatLab, para sinais de até 4 MHz, com opção de filtragem digital do sinal de entrada;
- Construir um osciloscópio digital com o apoio da arquitetura aberta “Arduino”;
- Possibilitar que os alunos possam ensaiar circuitos fora da escola.

A previsão de custos para a aquisição dos componentes eletrônicos é listada no quadro 1 a seguir.

MATERIAL	CUSTO UNITÁRIO	QUANTIDADE	VALOR
RESISTORES	R$ 0,20	10	R$ 2,00
PLACAS DE C.I.	R$ 10,00	1	R$ 10,00
CAPACITORES	R$ 0,30	18	R$ 5,40
KIT ARDUINO UNO	R$ 70,00	1	R$ 70,00
CONECTORES	R$ 5,00	2	R$ 10,00
744040N	R$ 4,00	1	R$ 4,00
74LS590N	R$ 3,00	2	R$ 6,00
7805TV	R$ 1,00	1	R$ 1,00
UM61512	R$ 10,00	1	R$ 10,00
74LS00	R$ 1,00	1	R$ 1,00
TLC5540SOP	R$ 12,00	1	R$ 12,00
74HC157N	R$ 3,00	1	R$ 3,00
OPA2604	R$ 12,00	1	R$ 12,00
CHAVE 2x5	R$ 9,00	1	R$ 9,00
OSC 40MHz	R$ 20,00	1	R$ 20,00
KIT DE FIOS/PLUGS	R$12,20	1	R$12,20
TOTAL			**R$ 187,60**

Quadro 1 - Orçamento previsto para o material eletrônico
Fonte: Autoria própria

2.4 MÉTODO DE PESQUISA

O método para implementar este projeto foi fundamentado em periódicos, livros, *datasheets*, sendo a maioria das referências versando sobre a conversão de tensões analógicas para digitais, além da insubstituível orientação buscada com tutores das áreas de conhecimento envolvidas.

O trabalho iniciou com a identificação das tecnologias que poderiam ajudar a cumprir os objetivos anteriormente enumerados. O próximo passo foi compartimentar os blocos com elementos de circuitos que possam ser testados isoladamente. Desta forma, pode ser avaliada a viabilidade de construção com a parte ensaiada, sem o risco de descobrir que este elemento é incompatível com o objetivo final.

Em um segundo passo foi analisada a tecnologia de conversores analógico para digital *Flash* e *Half-Flash*, seu princípio de funcionamento, suas vantagens e desvantagens. Este estudo foi fortemente baseado em publicações *on-line* de

fabricantes dos circuitos integrados comerciais que tornam estas tecnologias aplicáveis na prática.

Em seguida os vários blocos de circuitos foram interligados e ensaiados como um todo. Este teste deve revelar se as partes são compatíveis, permitindo uma simulação mais abrangente do funcionamento do *software* e *hardware* envolvidos. Então os parâmetros de funcionamento podem ser escolhidos para que a operação seja compatível com a tecnologia adotada.

Na última etapa de criação do circuito operacional do osciloscópio foi experimentada a adição de diversos conhecimentos técnicos adquiridos com as disciplinas que tem vínculo com o tratamento de sinais digitais e analógicos, seja no domínio do tempo ou das frequências.

3 FUNDAMENTAÇÃO TEÓRICA

3.1 O CONVERSOR FLASH

O conversor flash, também chamado conversor A/D paralelo, tem seu princípio de funcionamento baseado em amplificadores operacionais, conforme Torres (2006). Este conversor trabalha comparando a tensão de entrada – ou seja, o sinal analógico – com uma tensão de referência, que seria o valor máximo obtido pelo sinal analógico. Por exemplo, se a tensão de referência é de 5 volts, isto significa que o pico do sinal analógico seria de 5 volts. Em um conversor A/D de 8 bits, quando o sinal de entrada atinge os 5 volts, é encontrado um valor numérico de 255, ou 11111111 em binário, na saída do conversor A/D, ou seja, o valor máximo possível.

A tensão de referência é reduzida por uma rede de resistores em série e outros comparadores são adicionados para que a tensão de entrada, normalmente um sinal analógico de tensão variando no tempo, possa ser comparada com outros valores.

Na figura 1 um conversor A/D paralelo de 3 bits é representado. A conversão é feita através de amplificadores operacionais comparadores de tensão.

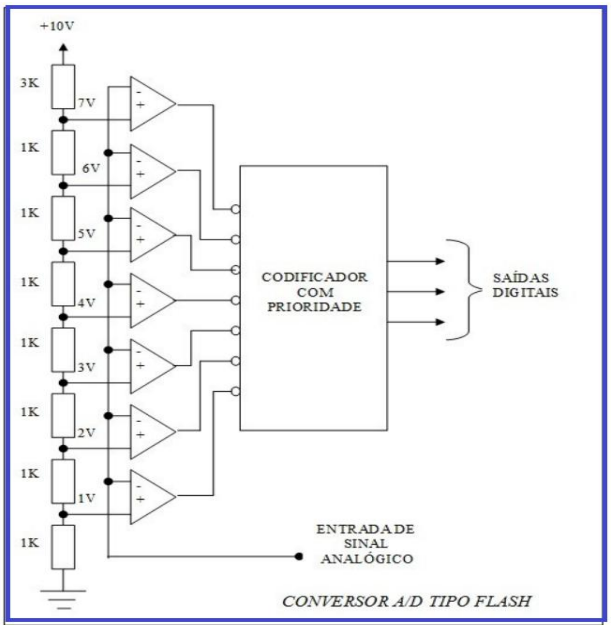


Figura 1 - Conversor Analógico/Digital flash
Fonte: www.clubedohardware.com.br (2012).

O codificador de prioridade pode ser feito através de portas XOR com uma série de diodos e resistores ou através de um único *chip* como o 74HC148, um conhecido codificador de prioridade de 8 para 3 linhas.

Apesar de conversores A/D paralelos usarem um projeto muito simples, eles requerem uma quantidade grande de componentes. O número de comparadores necessários é de (2^n) -1, onde n é o número de *bits* da saída. Para um conversor A/D paralelo de oito bits são necessários 255 comparadores, e para um conversor A/D paralelo de 16 bits são necessários 65.535 comparadores.

3.2 O CONVERSOR TLC5540

Uma tecnologia de compromisso denominada "*Half Flash*" foi adotada no circuito integrado TLC5540, onde o número de comparadores é bem menor. Esta tecnologia pode ser mais bem entendida consultando o *datasheet* deste componente, conforme descrito pela Texas (1995). Na figura 2 o diagrama em bloco funcional do conversor analógico-digital de tecnologia "*Half Flash*" TLC5540 é demonstrado. Esta técnica de conversão se difere da "*Flash*" tradicional por dividir o circuito de comparadores de tensão em três blocos sequenciais, no caso específico deste circuito integrado. O TLC5540 é um conversor de 8 *bits* que opera em taxas de amostragem de até 40 milhões de amostras por segundo. Sua alimentação é de 5 volts. Outra característica relevante para este projeto é o custo muito baixo. É tipicamente empregado para digitalizar sinais de vídeo que ocupam uma banda de até 4 MHz.

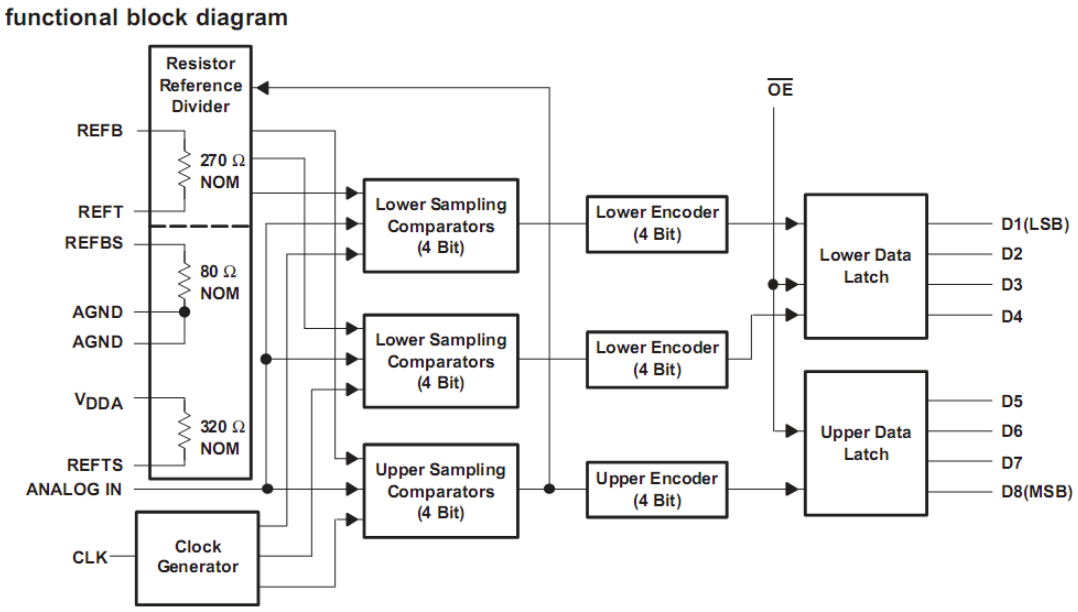


Figura 2 - Conversor A/D "Half Flash" TLC5540
Fonte: www.ti.com (2012).

Devido a esta arquitetura em três blocos sequenciais, um sinal analógico de entrada é digitalizado com a resolução de 8 *bits* após ocorrerem três transições descendentes do relógio na entrada "CLK", conforme representado na figura 2. A consequência disto é que os 8 *bits* de saída sempre estarão atrasados três ciclos de relógio em relação a entrada de sinal. Este atraso é uma desvantagem da tecnologia "half flash" em relação a "flash", mas para a utilização que o osciloscópio é proposto não representa uma deficiência significativa na prática.

3.3 A PLACA ARDUINO

Arduino é o nome dado a uma plataforma de *hardware* livre muito utilizada para finalidades educacionais. O principal arquiteto desta plataforma (BANZI, 2011), afirma que o Arduino é "uma ferramenta para tornar os computadores capazes de detectar e controlar mais do mundo físico do que um computador de mesa". Na prática este circuito é uma pequena placa microcontroladora padronizada onde um código aberto em um ambiente de desenvolvimento integrado permite que o usuário possa facilmente escrever um programa em uma linguagem semelhante ao "C". Este *software* é que vai fazer a placa Arduino controlar o mundo físico, que no caso deste projeto é o TLC5540 e seus componentes associados.

O arduino consiste em uma placa de circuito impresso com entradas e saídas para um microcontrolador da família AVR (ATMEL, 2011) e um *bootloader* que já vem gravado no microcontrolador. Este *bootloader* é um programa gravado na memória *flash* do microcontrolador que permite a fácil gravação do código fonte criado pelo usuário. As versões mais conhecidas do arduino são a Duemilanove (2009) e a UNO (2010), ambas empregadas neste TCC com a denominação genérica “módulo arduino”.

3.4 A PLACA OSCILOSCÓPIO

O funcionamento da placa osciloscópio é apoiada por um *software* especializado em cálculo numérico criado pela empresa MathWorks (MATLAB, 1983). Ele foi inicialmente escrito em “C”, e sua ênfase é na análise numérica. Também é caracterizado pelas funções prontas para cálculo matricial, processamento de sinais e construção automática de gráficos, dentre outras.

O programa principal deste TCC foi escrito em Matlab, o ambiente de desenvolvimento interativo onde a matriz numérica é o foco. Apesar de não requerer dimensionamento de matrizes a princípio, nem sempre esta estratégia é a melhor a ser abordada. O Matlab é uma linguagem de programação que permite a resolução de problemas numéricos muito mais rapidamente que em outras linguagens, como o “C”.

Alternativamente a este programa em Matlab foi desenvolvido outro emulador de osciloscópio mais simples, na linguagem “C”. Este faz uso do ambiente de desenvolvimento criado pela empresa desenvolvedora de *software* Bloodshed, conhecido como “DEV C++” (BLOODSHED,2005).

O funcionamento básico da parte física do projeto pode ser acompanhado na figura 3. O sinal analógico a ser medido, no tempo ou frequência, entra inicialmente no bloco de condicionamento de sinal. Sua função é formatar o mesmo de tal maneira que possa ser medido adequadamente pelo conversor A/D TLC5540. Este condicionamento é realizado através de resistores variáveis e amplificadores operacionais. Com a amplitude ajustada em nível DC conforme as exigências do TLC 5540, a digitalização é continuamente desenvolvida na taxa de 40 milhões de amostras por segundo, no máximo. Estes dados de 8 *bits* são armazenados na

memória de 2 Kbytes, que captura o resultado da conversão A/D. Quando a memória estiver cheia, o módulo arduino UNO (ou outro compatível da mesma família, como o 2009) interrompe a conversão e prepara a memória para leitura. Este bloco de 2Kbytes é então lido e enviado serialmente por um cabo USB (*Universal Serial Bus*) que conecta a placa do Arduino a um computador externo. O padrão de barramento serial USB foi projetado para simplificar a conexão de periféricos existentes para computador (NASCIMENTO, 2004). Estes periféricos são geralmente teclados, mouses, impressoras e webcams, mas neste projeto o periférico conectado é o módulo arduino. O protocolo de comunicação serial USB é uma padronização de *software* e *hardware* criada em 1995 por um grupo de empresas de tecnologia. O padrão USB tornou o uso destes periféricos muito mais simples para o usuário das mais diversas plataformas de microcomputadores.

Antes do USB era usado o padrão RS232 nas portas seriais dos computadores, uma normalização elétrica e lógica que definia como deveria ser a troca de dados seriais em baixa velocidade (TAKASE, 2006). Um exemplo de comunicação serial RS232 é a conexão entre o dispositivo apontador do tipo *mouse* com a porta serial RS232 que era padrão nos computadores dos anos noventa.

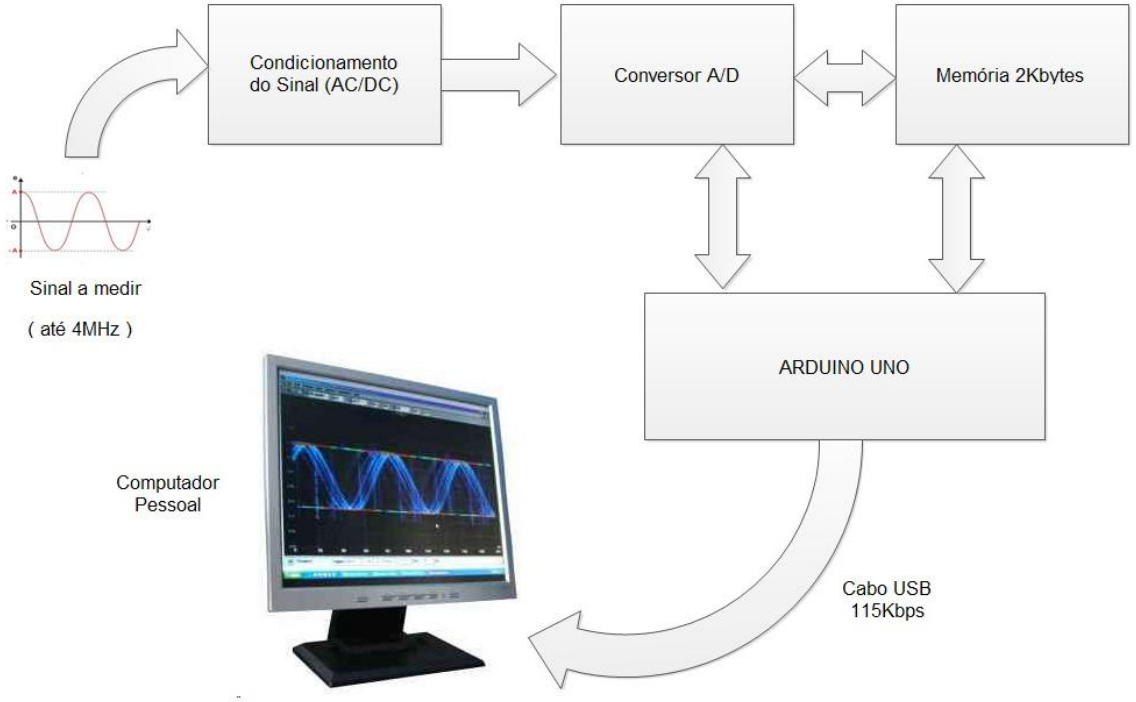


Figura 3 - Diagrama em blocos do osciloscópio proposto
Fonte: Autoria própria

A velocidade no cabo USB é 115 Kbps e não 40 Mbps como seria necessário se não houvesse esta memória *buffer*. Este computador, com seu monitor associado, possibilitará a visualização dos dados em tela gráfica através de um programa construído na linguagem Matlab. Desta forma, a sequência binária de 2048 bytes poderá ser representada, no domínio do tempo ou da frequência, ao término das rotinas de processamento digital de sinal. Este circuito atua em um modo chamado de “rajada”, isto é, faz amostras de 2Kbytes cíclicas do sinal de entrada. É uma estratégia tipicamente usada nos osciloscópios digitais básicos e bem adequada ao objetivo de emular um osciloscópio.

Este projeto buscará a ampla utilização dos tópicos vistos anteriormente no curso de Processamento Digital de Sinais da UTFPR, utilizando principalmente os conceitos de laços e vetores em ambiente de programação Matlab. Uma pesquisa adicional permitirá a leitura de uma porta serial externa dentro do Matlab. Esta porta será conectada a um conversor A/D externo para a posterior transmissão serial dos dados adquiridos. Outra implementação será necessária para gerar os gráficos na tela, bem como na construção de uma interface gráfica onde sejam oferecidos botões para o usuário escolher como será a visualização do oscilograma ou o espectro gerado pelas frequências do sinal digitalizado. Os conhecimentos de programação aplicáveis ao *hardware* do Arduino para acesso a porta USB e ao Matlab para desenhar as formas de onda na tela, mostrarão que é possível programar facilmente a leitura indireta de um conversor A/D através de um microcontrolador associado a uma memória de baixo custo.

Com relação ao objetivo específico de criar uma versão em linguagem C, além daquela construída em Matlab, cabe uma explicação. O ambiente de desenvolvimento em Matlab é um software comercial que exige o dispêndio mínimo de 99 dólares americanos para ser adquirido. Na planilha de custos deste projeto não foi considerado este valor porque a UTFPR disponibiliza aos seus alunos um laboratório de microcomputadores com o Matlab instalado. Já a IDE DEV C++ é gratuita e disponível para não estudantes desta instituição federal. Sob este ponto de vista pode ser interessante empregar apenas ferramentas que não gerem despesas extras aos que forem construir o projeto. Esta versão em linguagem C será bem básica, possuindo apenas a funcionalidade de visualizar a forma de onda na tela, no domínio do tempo. Apenas a versão em Matlab oferecerá a funcionalidade de visualização no domínio da frequência, assim como a estimativa

da amplitude e várias outras facilidades. Os programas em C são executados conforme representado na figura 4. A figura 5 tem uma projeção de como seria a saída gráfica gerada em linguagem C. Para montar esta janela, o DEV C++ empregará a biblioteca desenvolvida por Main (2006), também de domínio público.

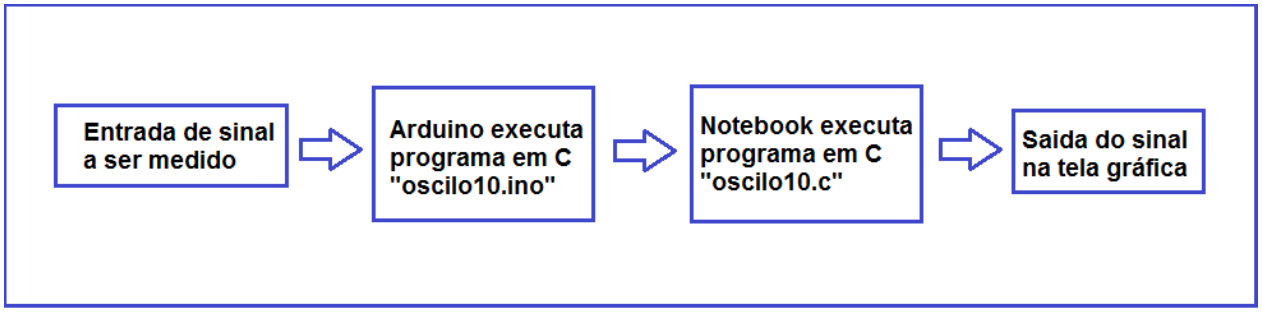


Figura 4 – Programas em C sendo executados
Fonte: Autoria própria

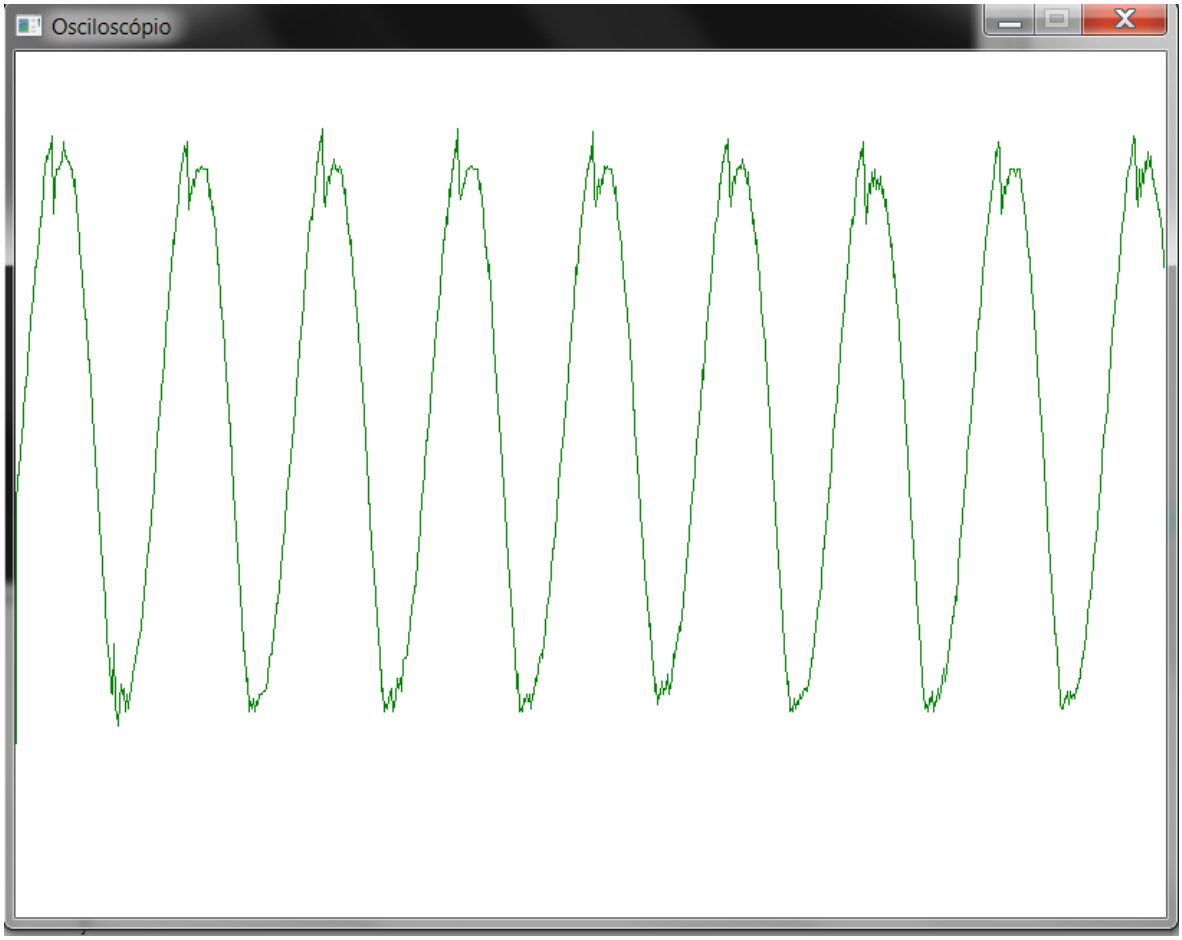


Figura 5 - Uma senóide de 400 KHz capturada pelo programa em C
Fonte: Autoria própria

4 FUNCIONAMENTO

4.1 PROGRAMAS EM LINGUAGEM C

O osciloscópio digital objeto deste projeto é basicamente um circuito integrado TLC5540 associado a uma memória externa em uma configuração semelhante a adotada por Radig (2008). O sinal de corrente contínua (DC) ou alternada (AC) fornecido por uma fonte externa - um gerador de funções, por exemplo - entra pelo terminal “X3” ou “X4”, conforme o esquema listado no apêndice A. Após condicionamento de amplitude e nível DC pelo circuito integrado IC7, o sinal analógico é injetado no pino 19 do TLC5540. Este circuito integrado, o IC9, converte o sinal em uma palavra binária de 8 bits que é disponibilizada no barramento como “D1 ... D8”. A velocidade de conversão A/D é de quarenta milhões de amostras por segundo, ou 40 MSPS (*mega samples per second*). Um programa escrito em linguagem C que gera código executável compatível com o microcontrolador ATMEGA8, criado por Atmel (2011), enche um *buffer* de 2048 octetos, também denominado de 2 Kbytes, com amostras geradas pelo IC9. Estes dados vão permitir que um microcomputador PC, que também executa código gerado por um compilador C para microprocessadores da plataforma Intel x86, possa recebê-los serialmente, gerando um gráfico cartesiano na resolução de 800x600 *pixels*, emulando assim um osciloscópio digital. Cabe aqui notar que a família de microcontroladores ATMEGA8 foi adotada neste projeto por ser a base da plataforma de *hardware* aberta conhecida como arduino, descrita por Banzi (2011). Segundo o mesmo autor, o projeto arduino é considerado para a construção de protótipos o equivalente ao linux para os sistemas operacionais.

A descrição do funcionamento sempre vai se referir ao diagrama elétrico do apêndice A e aos *softwares* que estão listados nos apêndices B e C. Como mencionado, o conversor A/D adotado foi o TLC5540. Seu circuito de entrada faz uso do potenciômetro R2 como divisor de tensão. Além de permitir o ajuste da amplitude do sinal observado na tela, R2 evita que um sinal de amplitude elevada danifique este circuito integrado facilmente. O potenciômetro R3 controla a altura vertical através de um nível DC sobreposto. Ambos atuam com o apoio de IC7.

O *firmware* descrito no apêndice C é executado por IC8 que tem a função de preencher a memória IC4 com os dados gerados por IC9. Notar que apesar da memória ter a capacidade de armazenar 65536 octetos, apenas 2048 são usados.

Assim que IC4 recebe 2 Kbytes de dados, IC8 deixa a memória em modo leitura e chaveia o multiplexador IC5 para que um sinal de relógio (*clock)* mais lento que os 40MHz gerados pelo oscilador a cristal Q2, possa ser emitido *bit* a *bit* por IC8 de tal forma que a memória IC4 seja lida *byte* a *byte*.

Cada *byte* armazenado em IC4 é lido pelo ATMEGA8 e transmitido serialmente pelo seu pino 28. Esta transmissão segue o protocolo RS232 assíncrono e não usa sua UART interna (*Universal Asynchronous Receiver/Transmitter*). Uma biblioteca de transmissão serial por *software* é empregada ao invés da UART do IC8 devido aos pinos 2 e 3 serem usados para a leitura dos *bytes* da memória IC4. A transformação de paralelo para serial é emulada por esta biblioteca disponível gratuitamente para a plataforma arduino. A conversão do protocolo RS232 para o protocolo *Universal Serial Bus* (USB) é feita pelo circuito integrado IC10. Na versão DUEMILANOVE (2009) da placa arduino, este integrado é um FT232RL, já na versão UNO é um ATMEGA8U2-MU. Ambos executam funções equivalentes.

Um microcomputador PC padrão x86 recebe os dados seriais enviados a 115200 bits por segundo através de um cabo USB conectado entre o terminal X1 e sua porta serial “COMX”, sendo X o número da porta padrão para comunicação assíncrona do PC. Geralmente X é o número 1. No circuito montado, o sistema operacional Windows 7 reconheceu como COM11, uma porta USB serial virtual criada pelo circuito IC10.

Os dados recebidos por esta porta COM11 do PC são tratados por um código compilado em C gerado pela IDE “Dev C++”, concebida pela desenvolvedora de *software* Bloodshed (2005). IDE é a interface gráfica e significa *Integrated Development Environment*. Este programa usa a função “Readbyte”, listada na linha 36 do apêndice B, criada graças as API do Windows (WIN API, 1995), sigla para *application programming interfaces*. Esta função faz a leitura dos *bytes* oriundos da placa do osciloscópio digital. Inicialmente o *loop* vai procurar o início do *frame*, ou bloco, identificado por três *bytes* 7E consecutivos, em notação hexadecimal. Estes *bytes* são produzidos pelo código que é executado em IC8 e servem como cabeçalho para delimitar o início de um bloco de dados a ser tratado.

Em seguida a descoberta do início do *frame*, um *loop* “for” armazena 2048 *bytes* no vetor “i[x]”, conforme listado na linha 84. Então começa a procura pelo sincronismo do sinal, que também é gerado por *software*, não por *hardware*. A variável “s” contém o valor inicial a ser procurado na forma de onda digitalizada

recebida. Dentre os valores de tensão 0 a 255, é buscado inicialmente o sincronismo de valor 120, o que corresponde a aproximadamente metade da tensão máxima do sinal. Quem faz esta procura é o “if” da linha 89. Se o sincronismo for encontrado nos primeiros 1246 bytes do vetor “i[x]”, o programa poderá iniciar a impressão dos dados na tela gráfica. Se o sincronismo não for encontrado entre os valores 118 e 122, o contador de sincronismo é incrementado em dez unidades e um novo patamar de sincronismo é procurado nos dados. As linhas 106 a 110 executam esta rotina de autossincronismo, varrendo outros valores possíveis até a forma de onda se estabilizar na janela gráfica.

A tela é limpa na linha 111 e os dados recebidos são plotados através do *loop* que inicia na linha 113 e termina na linha 116. Caso alguma tecla seja pressionada, a função “kbhit”, do “while” da linha 68, habilitará o fechamento da janela gráfica, da porta serial e finalizará o programa, tudo possibilitado pelas funções listadas nas linhas 120 e 122.

4.2 PROGRAMA EM LINGUAGEM MATLAB

No apêndice D está listado o fonte do programa “Osc4MHzV4.m“, que emula no ambiente Matlab um osciloscópio digital semelhante ao criado pelo algoritmo em C, com o diferencial de implementar algumas facilidades não presentes anteriormente, como a análise do espectro do sinal medido e a aplicação de filtros digitais em tempo real.

Tudo começa na linha 12, onde um comando específico do Matlab avisa ao compilador para usar a biblioteca gráfica OpenGL (1992), um padrão de livre uso, criado e especificado por um conjunto empresas de *hardware* gráfico 2D e 3D. Esta linha é opcional, pode ser comentada caso a placa gráfica apresente alguma incompatibilidade com esta biblioteca. Na próxima linha, a instrução “delete(instrfindall)” auxilia a evitar o erro denotado pelo aviso na forma “*Port: COM11 is not available*”, tão comum quando muitas manipulações nas portas de comunicação são realizadas neste ambiente.

Após a linha 14 ter limpado variáveis e fechado as janelas sem uso, a função “scanports()” concebida por Slazas (2011), varre as portas seriais ativas do microcomputador PC hospedeiro. Isto é necessário para determinar qual a porta USB e/ou Serial RS232 está disponível para comunicação com o Arduino. No

programa em C anterior era necessário compilar o fonte com a informação prévia de que a COM11, por exemplo, está presente. Com “scanports()” esta tarefa é automatizada em tempo de execução. Seu código fonte foi obtido gratuitamente no sítio da Mathworks indicado nas referências.

Da linha 16 a 21 é criada e configurada a porta de comunicação USB com a velocidade de 115 Kbps. No trecho delimitado pelas linhas 22 a 26 as variáveis e vetores são inicializados com valores adequados.

A porta de comunicação é aberta no index 27. Um reset por software na placa do arduino é executado entre as linhas 28 e 30. Isto é necessário para evitar que o programa principal encontre o módulo arduino travado. Os botões e os *slider´s* são definidos nas linhas 34 a 82. Estas entidades gráficas criam na tela uma interface homem-máquina parecida com os botões liga/desliga e potenciômetros deslizantes lineares, os *slider´s* encontrados na eletrônica.

O programa todo é basicamente um laço infinito delimitado pelo “while” presente na linha 86 que por sua vez é fechado com o “end” da distante linha 297. A primeira tarefa deste laço é procurar o sincronismo do *frame,* enviado pelo arduino através da USB. Os “if´s” da linha 90 vão procurar a assinatura “7E-7E-7E” no fluxo de dados recebidos, de forma idêntica ao programa em C. Esta estrutura de protocolo orientado a octeto (*byte*), não dígito binário (*bit*), tem vantagens e desvantagens. A simplicidade é vantajosa, pois permite que o programa em Matlab trate este protocolo sem perda no desempenho, o que é positivo para a visualização da forma de onda na tela com uma taxa de renovação, ou *frame rate*, satisfatória. Se o processamento do protocolo exigir um custo computacional alto, a animação gráfica será lenta e insatisfatória ao usuário. Mas esta estratégia tem alguns inconvenientes. O primeiro é a possível confusão de um cabeçalho “7E-7E-7E” com um ruído aleatório no sinal amostrado ou até o próprio bloco de dados. Para evitar parcialmente esta falha, o programa Matlab faz a leitura de apenas 2047 octetos após o cabeçalho. Mesmo assim alguns erros podem ocorrer e ter como conseqüência uma perda de sincronismo momentânea na tela. Felizmente o resultado final é um pequeno salto horizontal na forma de onda em exibição, o que não compromete a visualização confortável. A alternativa seria empregar um protocolo orientado a *bit* como aquele usado nos canais de dados de alta velocidade hdlc (*high-level data link control*), que emprega a técnica da inserção de *bits* conhecida como *bit stuffing*. Isto garante que as sequências de bits do cabeçalho

não sejam confundidas com os delimitadores ou mensagens de controle, conforme ensinado por Fall (2011). Este processamento preferencialmente não deve ser executado pelo *software*, mas por um *hardware* dedicado, afim de não resultar na desagradável lentidão gráfica.

Encontrado o sincronismo, o “fread” da linha 95 carrega os 2047 *bytes* amostrados pelo arduino para o vetor “n” do programa. O teste feito na linha 97 é uma forma encontrada para evitar erros na recepção de dados. Se a função “numel” retornar um número diferente de 2047, significa que o pacote de dados recebido é inválido. Na prática foi verificado que erros geram vetores com valores geralmente inferiores a 2047 elementos, sendo comum retornar o valor zero. Desta forma o quadro pode ser descartado e uma nova leitura é realizada. Mais uma vez as portas seriais e o arduino são reinicializados. O ciclo termina na linha 109. Este método de verificação da integridade dos dados recebidos é muito inferior ao tradicional *cyclic redundancy check* ou CRC, explicado por Ritter (1986), mas também oferece pouco custo computacional.

Descoberto o sincronismo de quadro, o algoritmo parte para a busca do sincronismo da forma de onda, isto é, aquele ponto escolhido para iniciar o desenho da função na tela que deve ser igual em todas as amostras a fim de gerar no usuário a ilusão de que o sinal está parado. Nos osciloscópios tradicionais este sincronismo, também denominado de gatilho ou *trigger*, é feito por *hardware.* Tipicamente é desempenhado por um transistor ou amplificador operacional ajustado para disparar com determinado nível de tensão. Aqui a solução de *software* foi adotada para reduzir o custo com *hardware* adicional não essencial.

Quando o “if” da linha 112 encontra um sincronismo dentro da janela entre os números 118 ao 122 e, ao mesmo tempo, é uma rampa de tensão discreta crescente, o ponto desejado é salvo na variável “a”. Se o sincronismo não for encontrado até o *byte* número 1246, é sinal que não haverá espaço para mostrar o sinal na tela de 800 pixels. Neste caso a procura é abortada e a forma de onda desenhada sem sincronismo mesmo, o que será interpretado pelo usuário como uma falha de sincronismo do osciloscópio. O motivo de gerar o gráfico sem sincronismo é não deixar a tela “travada” por falta de renovação dos ciclos de sinal. Esta opção, apesar de não ser perfeita, é melhor que a alternativa de só mostrar a forma de onda quando o sincronismo é encontrado.

Entre as linhas 125 e 130 existe um laço que implementa uma rotina de procura automática de sincronismo para os casos onde o valor inicial com “s=120” não é possível de ser encontrado na amostra recebida. Este valor é incrementado em 10 para que uma nova pesquisa seja desempenhada. Caso negativo, os valores vão subindo até 250. Se mesmo assim o valor for inadequado, o laço retornará a 10, fazendo uma varredura ascendente de valores na esperança de encontrar um patamar funcional.

Um vetor “xx”, com valores iniciais 1 até 800, é estabelecido na linha 132 para indexar os *pixels* horizontais da janela gráfica. O vetor “B” da linha 133 tem estes valores de tensão instantâneos que vão ser desenhados na vertical para os gráficos no domínio do tempo, sempre com a resolução de *8 bits* determinada pelo TLC5540.

Neste ponto começam as diferenças de tratamento do sinal que só existem na versão do osciloscópio em Matlab. Entre 134 e 136 são definidos os parâmetros básicos para os cálculos da Transformada Rápida de Fourier relativa ao sinal no domínio da frequência. No intervalo entre 138 e 141 são extraídos os valores dos *slider´s* da frequência de corte e da posição vertical do traço (y). Também são calculadas as tensões pico-a-pico “Vpp” e o nível de tensão contínua “Vdc” do sinal. Em 143, a função “switch” permitirá que os valores gerados pelos botões pressionados sejam devidamente processados. Os dois botões mais básicos são os que selecionam o tempo e a frequência. O botão tempo indica ao programa que o usuário quer um gráfico que reflita uma forma de onda no domínio do tempo, aquela tradicionalmente mostrada pelos osciloscópios analógicos. A figura 6 mostra uma tela gerada quando o botão tempo é clicado.

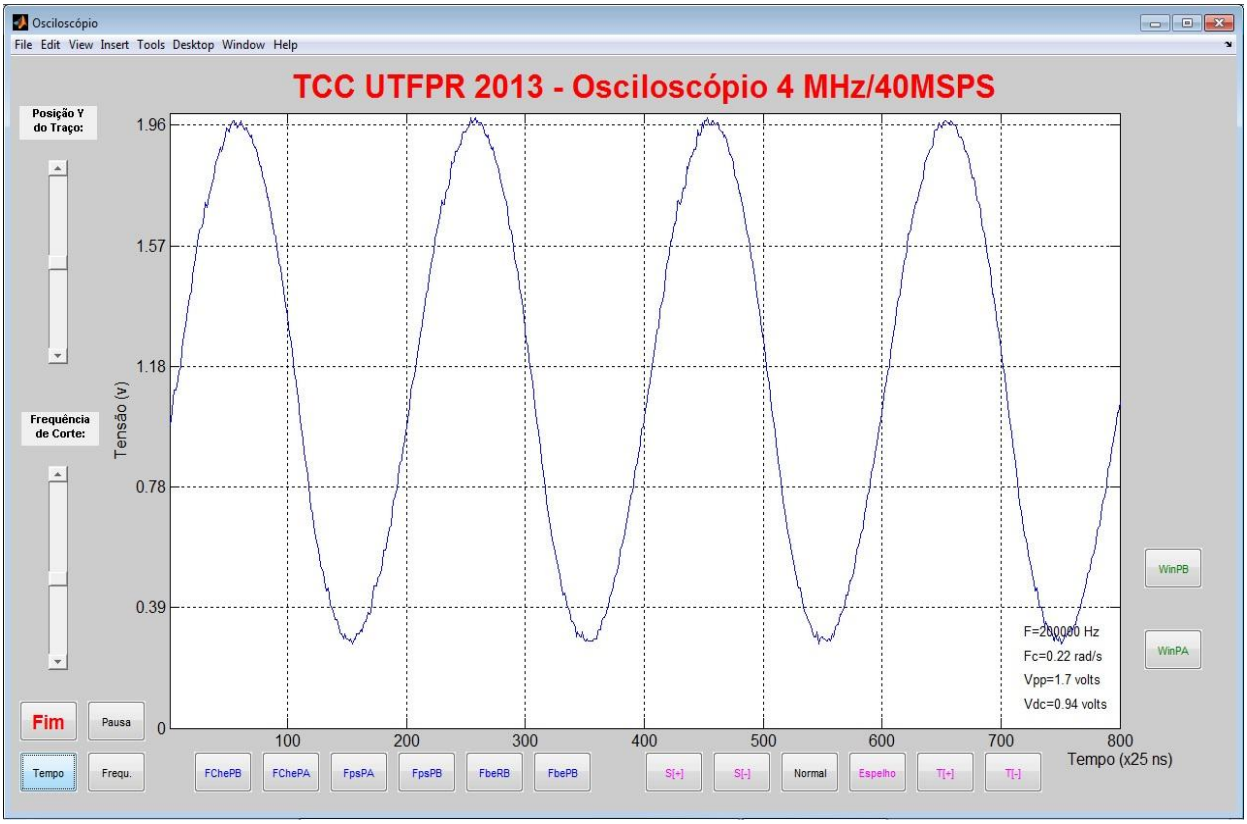


Figura 6 - Tela gerada quando o botão tempo é pressionado (senóide 200 KHz)
Fonte: Autoria própria

O botão frequência força a criação de um gráfico que é a representação do espectro do sinal obtido numericamente pelo cálculo da função temporal através da transformada de Fourier, cálculo este realizado em tempo real, na medida em que os dados das tensões discretas de *8 bits* chegam. A figura 7 ilustra uma tela gerada quando o botão frequência é acionado.

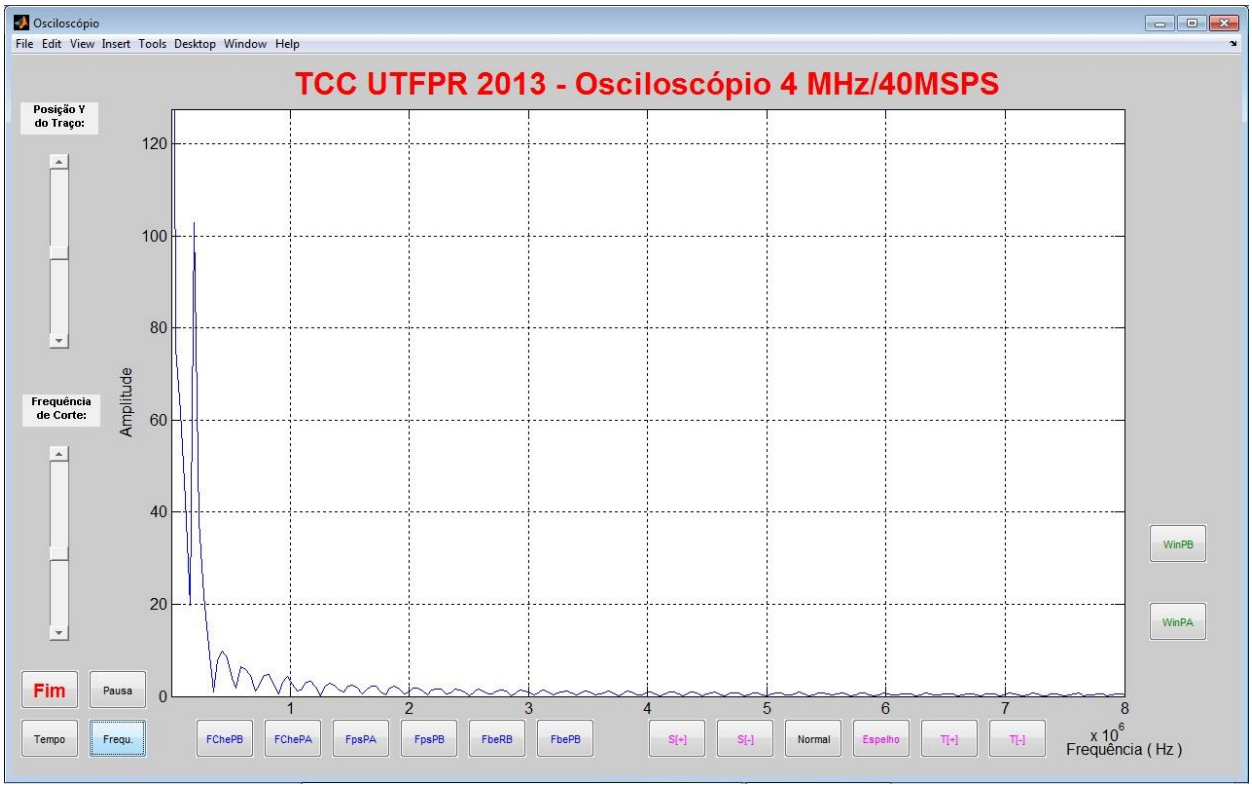


Figura 7 - Espectro gerado pela senóide quando o botão frequência é acionado
Fonte: Autoria própria

A aplicação dos filtros digitais ao sinal começa na linha 148. Para o caso de ter sido pressionado o botão com a sigla “FpsPB”, que gera a variável “tela=2”, será calculado um filtro de pólo simples passabaixa. Este, e a maioria dos outros filtros digitais que o osciloscópio emula, foram ensinados nas aulas da disciplina Processamento Digital de Sinais ministrada pelo professor Janeczko (2011), sempre com sinais simulados numericamente. Notar que a função “fpspb(SLI,B)” é a responsável pelo cálculo deste filtro passabaixa. Ela tem como parâmetros de entrada a variável “SLI”, que é um número entre 0 e 0.5 dependente da posição relativa do *slider* da frequência de corte do filtro selecionado e o vetor “B” que carrega a forma de onda discretizada no domínio do tempo. O código fonte da função que aplica este filtro ao sinal desenhado na tela está registrada no apêndice F. A figura 8 mostra o efeito de “alisamento” provocado pelo filtro na forma onda senoidal da figura 6.

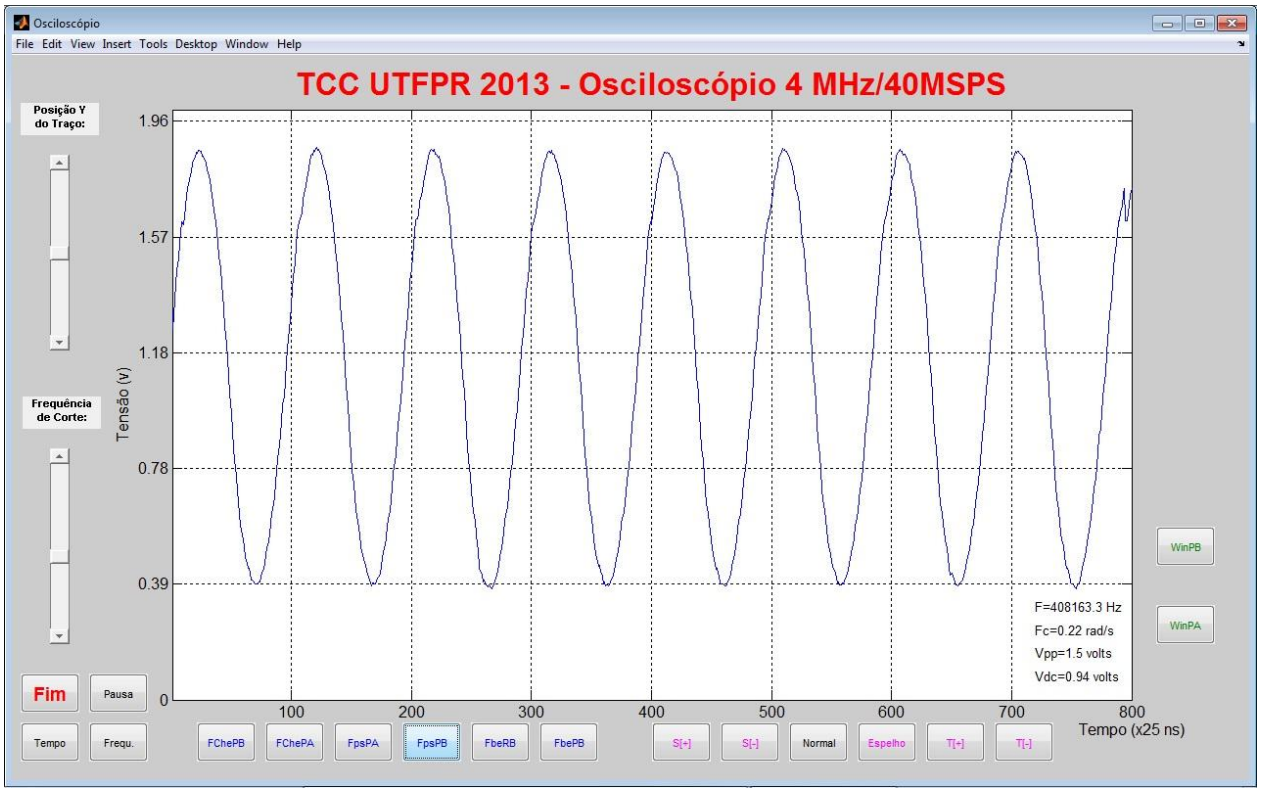


Figura 8 - Senóide de 400KHz após o botão FpsPB ser pressionado
Fonte: Autoria própria

De forma análoga, as linhas de programa 150 a 161, realizam os cálculos para filtrar o sinal de entrada com os filtros Chebyschev passabaixa e passa-alta, pólo simples passa-alta (apêndice G), banda estreita passabanda (apêndice H) e rejeitabanda (apêndice I). Os filtros Chebyschev não estão descritos nos apêndices por serem filtros implementados na biblioteca do próprio Matlab. Todos são filtros do tipo IIR, cuja característica é ter a resposta ao impulso com um número infinito de coeficientes não nulos, ao contrário dos filtros FIR, que tem resposta ao impulso com um número finito de coeficientes não nulos, segundo Haykin (2001).

Nos indexadores 162 e 168 um valor particular de sincronismo pode ser determinado pelo usuário através dos botões “s[+]” e “s[-]”, permitindo um ajuste fino na disposição do sinal no eixo cartesiano y. Não confundir este parâmetro com o gerado pelo *slider* de ajuste da posição y do traço, que promove apenas um acréscimo ou decréscimo na altura relativa da forma de onda com relação ao eixo vertical. Sua utilidade é apenas a de criar uma adequação visual, sendo um valor falso, diferente da tensão “Vdc” indicada anteriormente. Pode ser desfeito

pressionando a tecla “normal”, tratada pela linha 174. Uma variação de cálculo da transformada de Fourier é desenvolvida pelo botão “espelho” em 177.

Nas linhas 179 e 182 são selecionados os filtros tipo FIR chamados de “Window-Sinc”. As funções definidas por “winsinc(B,.0401,SLI,80,'hamming','high')” e “winsinc(B,.0401,SLI,80,'hamming','low')”, foram criadas por Sanchez (2005) e disponibilizadas para livre uso pelo sitio do Matlab. Com elas, o vetor B de entrada dará origem um vetor B de saída que foi filtrado pelo algoritmo “Window-Sinc” passa-alta e passabaixa, respectivamente. Os filtros são aplicados após os botões “WinPA” e “WinPB” serem pressionados.

Um novo conjunto de seleções é trabalhado a partir do referencial 188. O laço “for” iniciado em 192 desenha o sinal na tela gráfica fazendo uso da função “line” em 195 ou da função “rectangle” em 198. A variável “cong” define se o gráfico será pausado ou dinâmico, a gosto do usuário. O motivo de não usar o onipresente “plot” para traçar este gráfico no domínio do tempo é o melhor controle do pixel. Esta foi a forma encontrada para traçar um gráfico com interpolação linear pelo “line” ou o “pixel expandido” criado pela função “rectangle”, cujo resultado é representado na figura 9. Este caminho também permite o tratamento do sinal pixel a pixel quando este é desenhado na tela, como selecionado pelo “if” opcional da linha 193. Ele define uma faixa de valores onde ruídos transitórios gerados pela etapa de chaveamento do conversor analógico/digital podem ser eliminados. Comentando as linhas 193 e 201, este “if” opcional pode ser eliminado antes da compilação. A decisão pela sua eliminação deve ser baseada na avaliação da ocorrência ou não destes ruídos transitórios, denominados *spikes,* gerados pelo funcionamento do TLC5540, como retratado na figura 10*.* Uma curiosidade é a necessidade de um comando aparentemente inócuo como o “plot(1,1)”, antes de iniciar o desenho da janela. Sem ele o correto posicionamento e apagamento dos *pixels* são comprometidos.

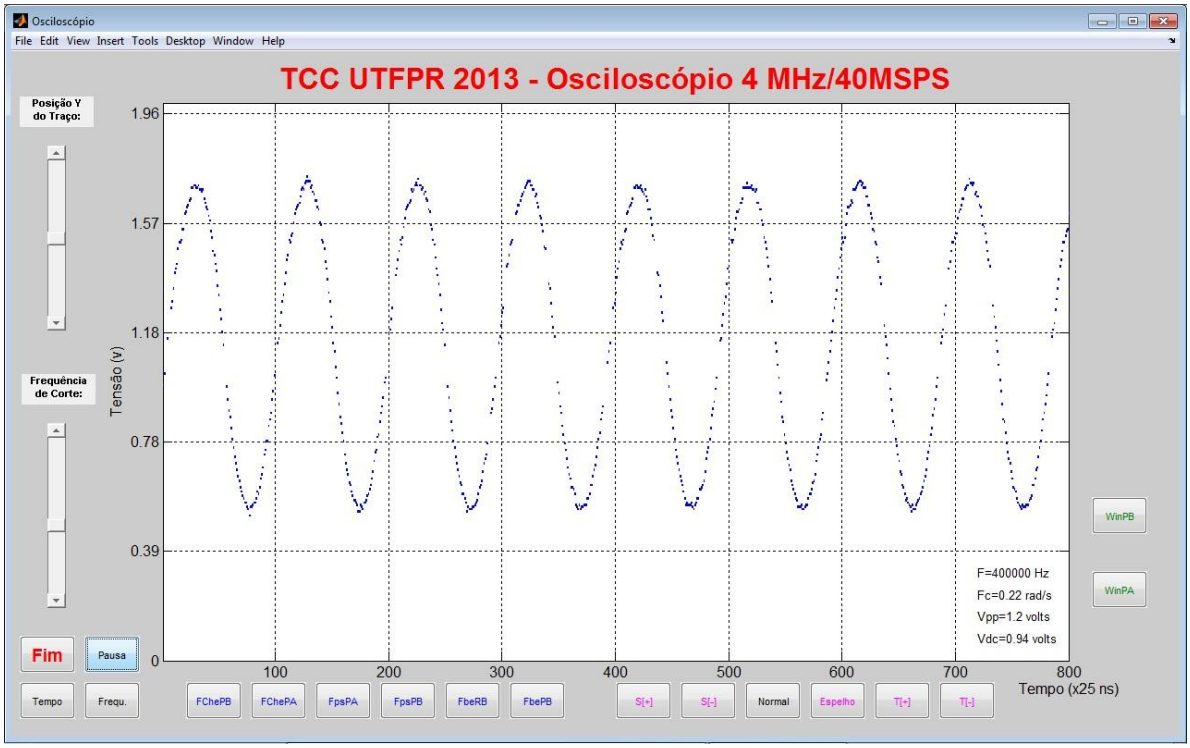


Figura 9 - Osciloscópio em modo ponto, dado pela função “rectangle”
Fonte: Autoria própria

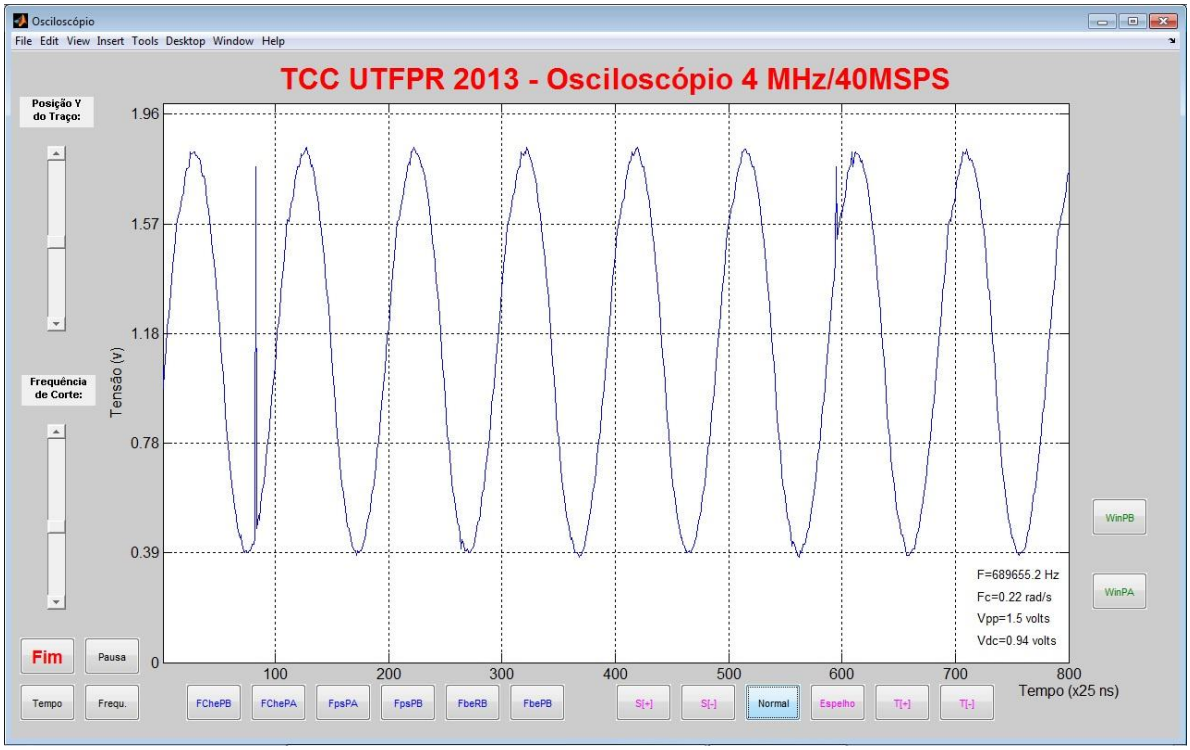


Figura 10 - *Spikes* sobrepostos a uma senóide de 400KHz
Fonte: Autoria própria

As escalas, os rótulos ou *labels* e os valores das tensões “Vdc” e “Vpp”, são impressos no intervalo entre 204 a 211. A frequência de corte “Fc” é mostrada em radianos por segundo.

A rotina que calcula a frequência do sinal periódico em tempo real inicia em 213 com a função “peakfinder(B)”, concebida por Yoder (2012). Também de domínio público, esta função devolve no vetor D as coordenadas de todos os picos, ou pontos de máximo, encontrados no vetor B. Primeiro a coordenada “x” do segundo pico do sinal é subtraída daquela do primeiro pico e o resultado é multiplicado pelo período da amostragem dado pelo oscilador Q2. Invertendo o resultado obtido, a frequência instantânea em hertz é descoberta. Esta informação só é válida para a posição 1 da chave de frequência de amostragem pois nas posições 2 a 5 o período não é mais 25 nanossegundos. A figura 11 mostra o local onde o Vpp, Vdc, Fc e F são impressos.

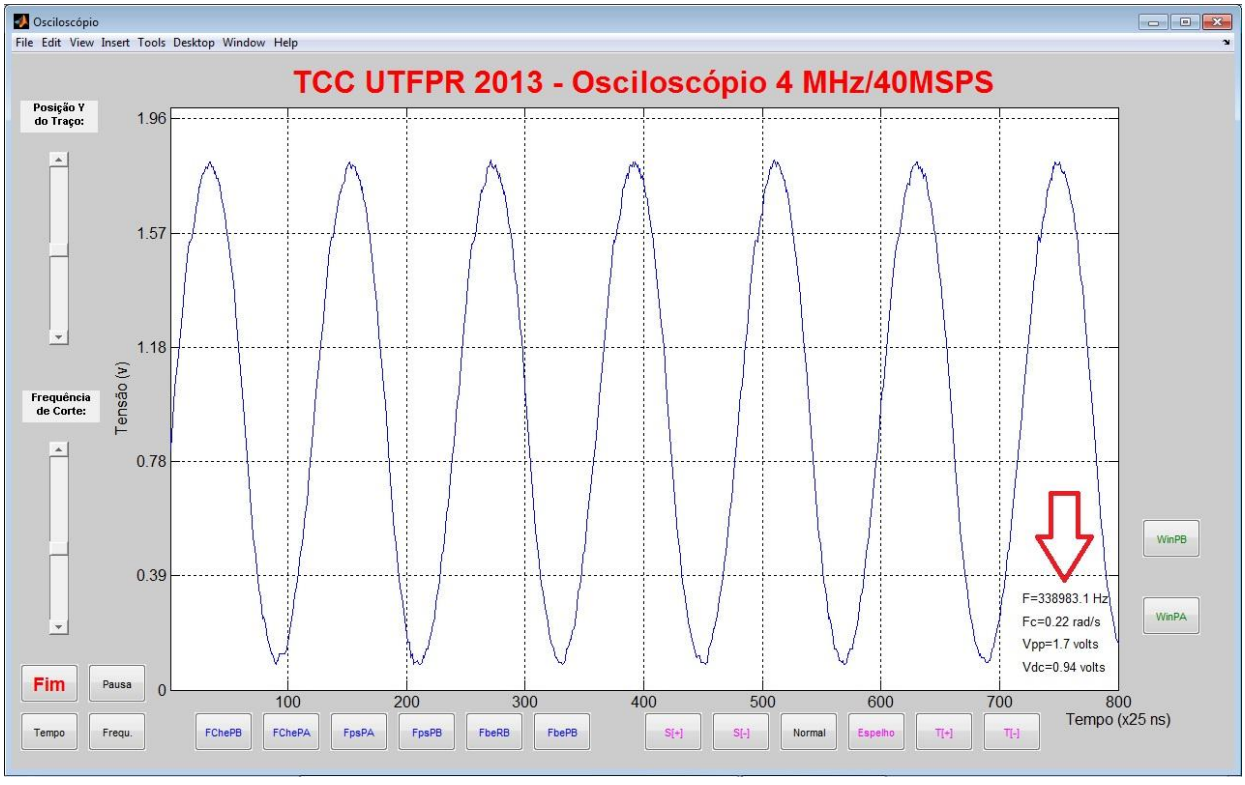


Figura 11 - Onde são impressos os valores F, Fc, Vpp e Vdc
Fonte: Autoria própria

A função “fft(B,NFFT)/L”, indicada em 220, calcula a transformada de Fourier do sinal. Ela é que vai permitir o traçado do conteúdo de frequências presentes no vetor “B”, originado pela amostragem dos níveis de tensão em relação ao tempo. Dois tipos de apresentação para este gráfico de espectro do sinal são possíveis. O primeiro é o via “plot”, que desenha uma linha contínua que segue aproximadamente o envelope das magnitudes das frequências presentes neste vetor, como já visto na figura 7. O segundo é dado pelo “stem”, que traça um gráfico mais discreto e preciso, conforme a figura 12 demonstra. Em seguida as escalas e os “*labels*” são definidos.

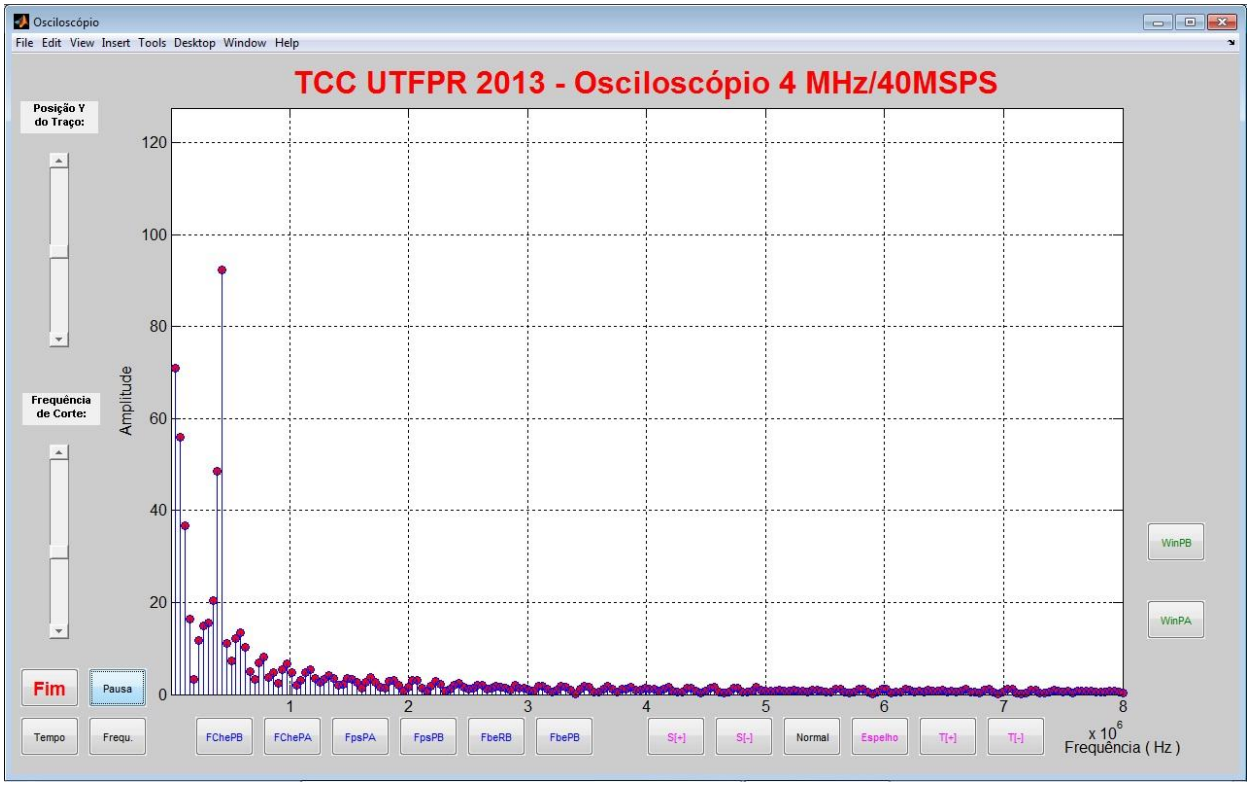


Figura 12 - Espectro gerado pela função “stem”
Fonte: Autoria própria

A linha 234 é rotulada como “normal” para indicar que as condições iniciais podem ser restauradas pressionando o respectivo botão e o gráfico possa ser criado via “plot”, sem qualquer tratamento mais apurado do pixel, como visto em 193.

Em 253 uma variação de transformada de Fourier nomeada como “Espelho” é calculada pela função “fftshift(abs(fft(B)))”. Ao clicar o botão respectivo, um gráfico espelhado do espectro é construído a partir do vetor “B”. O resultado é visto na figura 13. Se o botão “Fim” for selecionado, a variável “sai” receberá o valor 1 e a

porta serial que estava aberta será fechada, encerrando o programa. De forma semelhante, o botão “Pausa” ajusta a variável “congela” para 1. A exibição da animação gráfica no monitor permanecerá congelada até um novo clique neste elemento ser comandado.

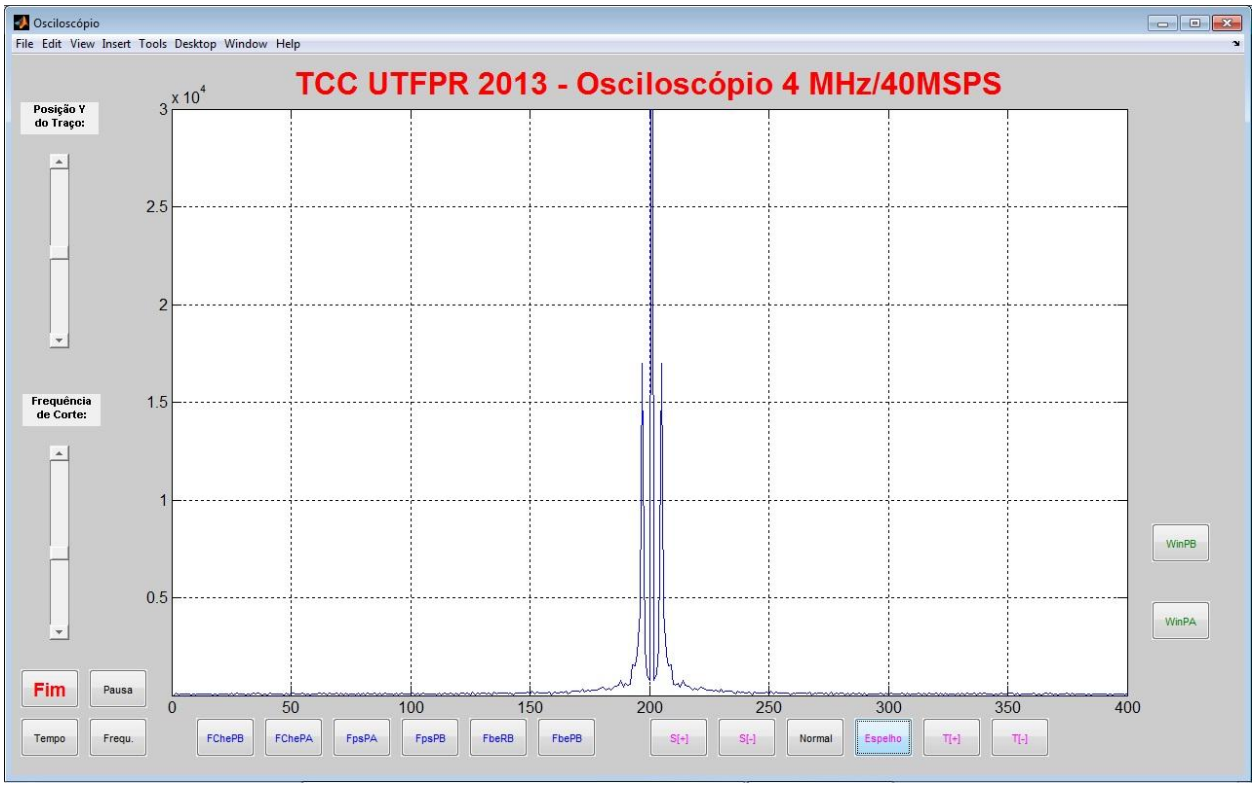


Figura 13 - Espectro do mesmo sinal calculado por “fftshift(abs(fft(B)))”
Fonte: Autoria própria

O incremento e decremento nas escalas de tempo são feitos em 276 e 284. Este ajuste permite que na escala básica de tempo varie de 250 nanossegundos até 20 microssegundos, que combinada com a chave rotativa S1, garante a visualização adequada para sinais de 100 Hz a 4 MHz. Na numeração 293 um “if” testa se ainda existe algum *byte* no *buffer* da porta serial. Caso exista, uma leitura “fantasma” é realizada com a intenção de esvaziar esta memória temporária que armazena os dados seriais recebidos. Este procedimento é importante, pois evita que um *buffer* não lido a tempo seja sobrescrito e ocasione perda de dados pelo programa.

A estrutura listada entre 299 e 313 trata erros que não foram previstos anteriormente. Basicamente promove o fechamento da janela ativa sem apagar as variáveis que serão possivelmente úteis em uma pesquisa de pane. O módulo

arduino é reinicializado e a porta serial fechada. Após uma pausa de um segundo, o programa é executado novamente com a expectativa de que o erro, seja ele de comunicação ou outro diverso, tenha desaparecido. As linhas 308 e 309 devem ter seus comentários retirados caso as reinicializações sejam constantes devido a um erro não resolvido por esta estrutura. Com este procedimento o programa terminará com uma mensagem na linha de comando que reportará a linha onde ocorreu o erro.

5 MONTAGEM

Para realizar a montagem do osciloscópio é necessário possuir a lista de materiais que consta do apêndice J. O circuito integrado IC6 é opcional se for usada uma fonte externa +/- 8.5 Volts para alimentar o amplificador operacional IC7. Para IC6 alimentar diretamente IC7, o esquema da figura 14 deve ser consultado. O pino 2 do IC6 deve ser conectado ao pino 8 do IC7 (V+). O pino 6 do IC6 deve ser conectado ao pino 4 do IC7 (V-).

O circuito integrado IC6 é o MAX232 (DRIVER, 2004), um duplo reforçador/receptor de linhas de dados que inclui um gerador de tensão chaveada baseado em capacitores externos para fornecer os níveis de tensão padrão RS232 a partir de uma única fonte de 5 volts. Cada receptor converte entradas RS232 em níveis 5 volts TTL/CMOS. Estas siglas significam “*transistor-transistor logic*” e “*complementary metal–oxide–semiconductor logic*”.

O circuito integrado IC7 é o OPA2604 (AMP, 2003), um amplificador operacional duplo com entrada FET (*field effect transistor*), projetado para ótimo desempenho em AC (*alternate current*). Apresenta muito baixa distorção, baixo ruído e ampla largura de banda para aplicações que exigem excelente desempenho dinâmico.

Esta aplicação do MAX232 não é trivial. Este conversor de níveis RS232/TTL está sendo usado apenas para aproveitar sua fonte chaveada interna para converter os 5v em +/-8.5v para alimentar o amplificador operacional OPA2604, não para converter sinais RS232 em TTL e viceversa.

Toda a montagem eletrônica é realizada em uma placa de circuito impresso padrão com dez centímetros de comprimento por cinco de largura. Na figura 17 é mostrada sua face superior com os componentes já soldados. A face inferior sem os componentes soldados é vista na figura 24. Na lista de materiais sua denominação é Y1.

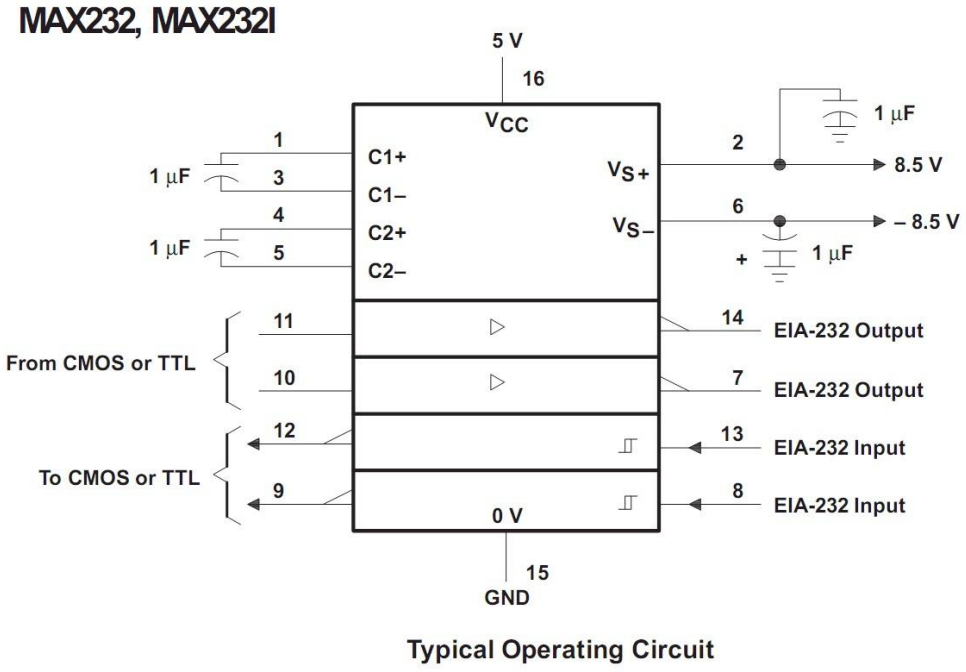


Figura 14 - Circuito Integrado MAX232 como fonte de simétrica de 8.5 V
Fonte: www.ti.com (2012).

Além da placa Y1, é necessário adquirir a placa adaptadora de circuito impresso SOP20/DIP, mostrada na figura 15, a fim de ligar o CI9 na placa de 10cm x 5cm (Y1).

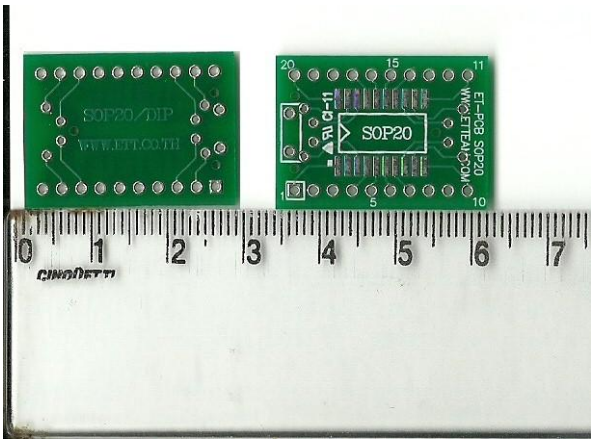


Figura 15 - Placa adaptadora SOP20/DIP (Y2)
Fonte: Autoria própria

Na lista de material do apêndice J, vários itens são do “kit arduino”. Isto significa que não devem ser adquiridos sozinhos, pois já vêm soldados na placa deste kit, que tem a aparência mostrada na figura 16.

Figura 16 - Placa do kit arduino UNO com o cabo USB
Fonte: www.arduino.cc (2012).

Toda a interligação dos componentes eletrônicos que são vistos na figura 17, deve seguir o ESQUEMA ELÉTRICO do APÊNDICE A. A montagem final está registrada nas figuras 18 a 20. Esta técnica de interligação com fios de cobre encapados é dita *wire-up* porque dispensa uma base fixa de circuito impresso para a união dos terminais elétricos.

A Figura 21 indica as funções dos botões da interface.

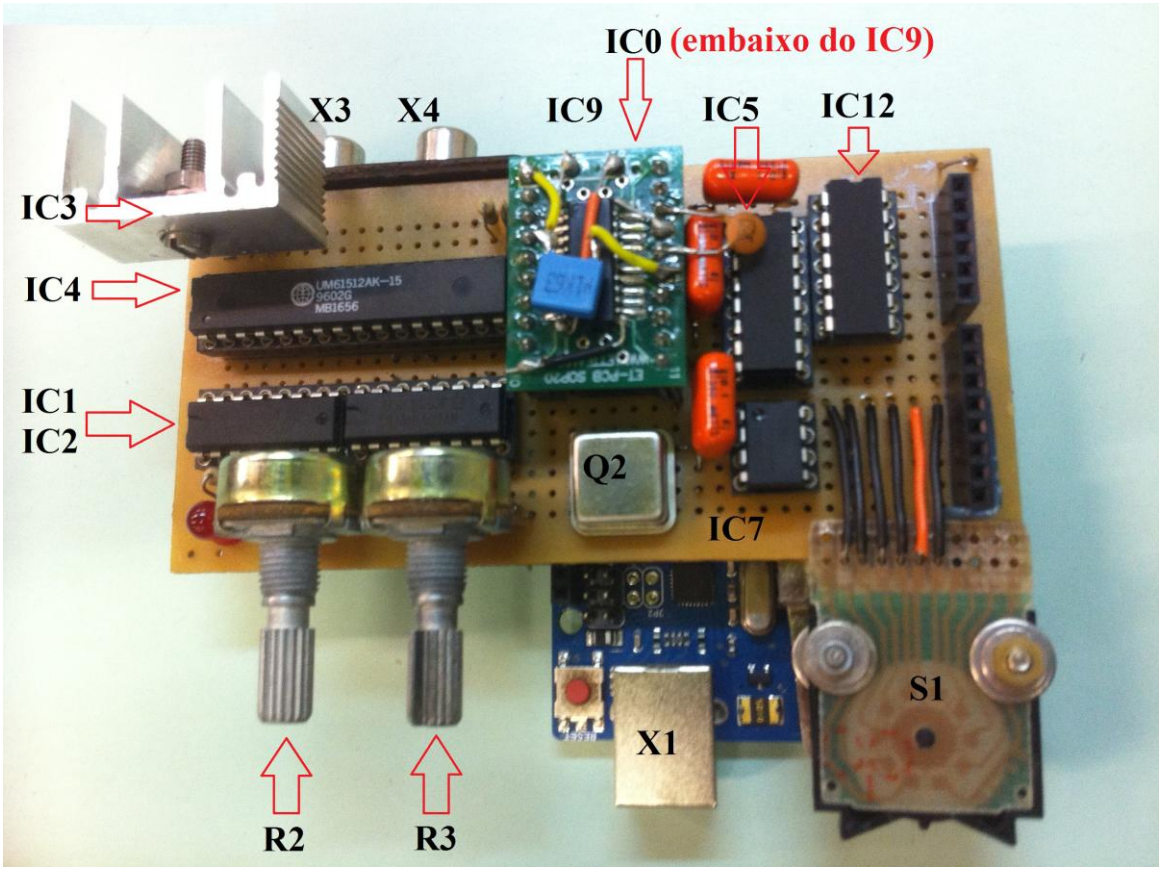


Figura 17 - Placa osciloscópio conectada na placa arduino UNO
Fonte: Autoria própria

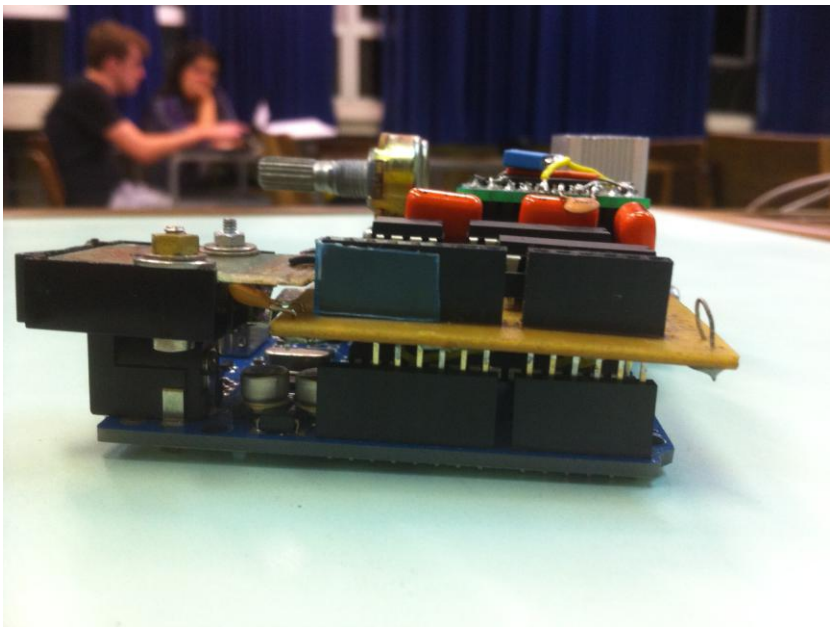


Figura 18 - Vista lateral da placa osciloscópio sobre o arduino
Fonte: Autoria própria

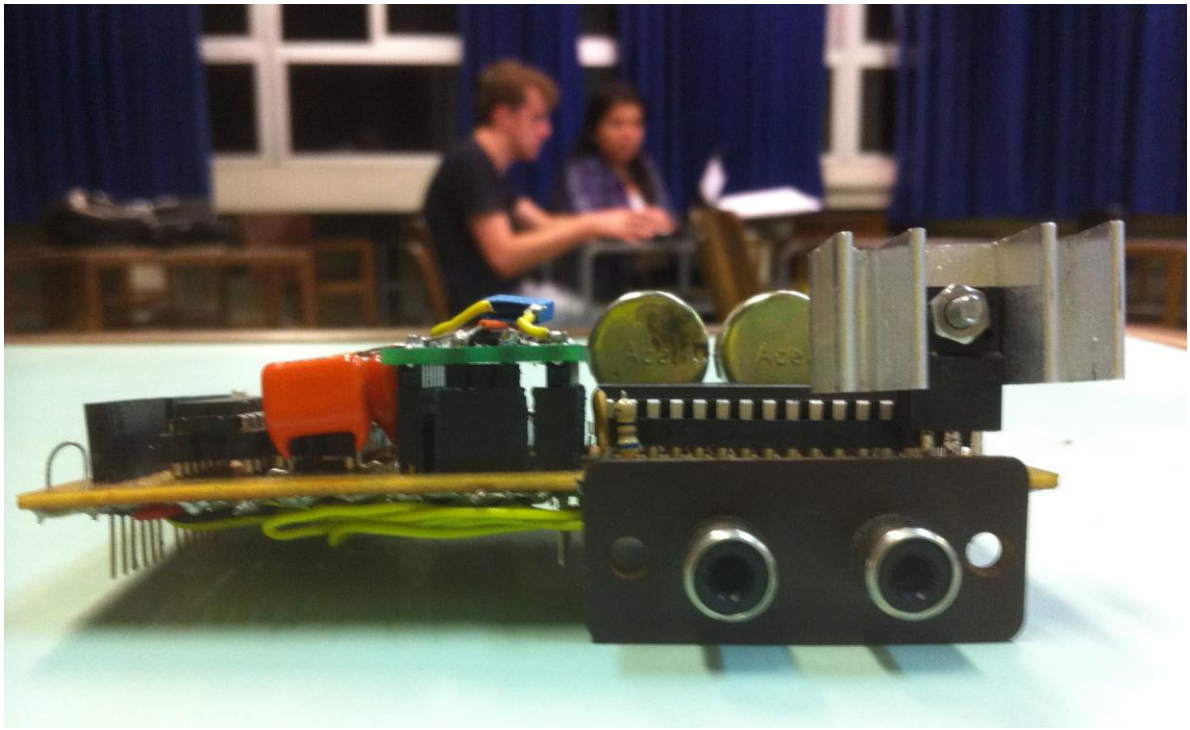


Figura 19 - Vista frontal da placa osciloscópio sem a placa arduino embaixo
Fonte: Autoria própria

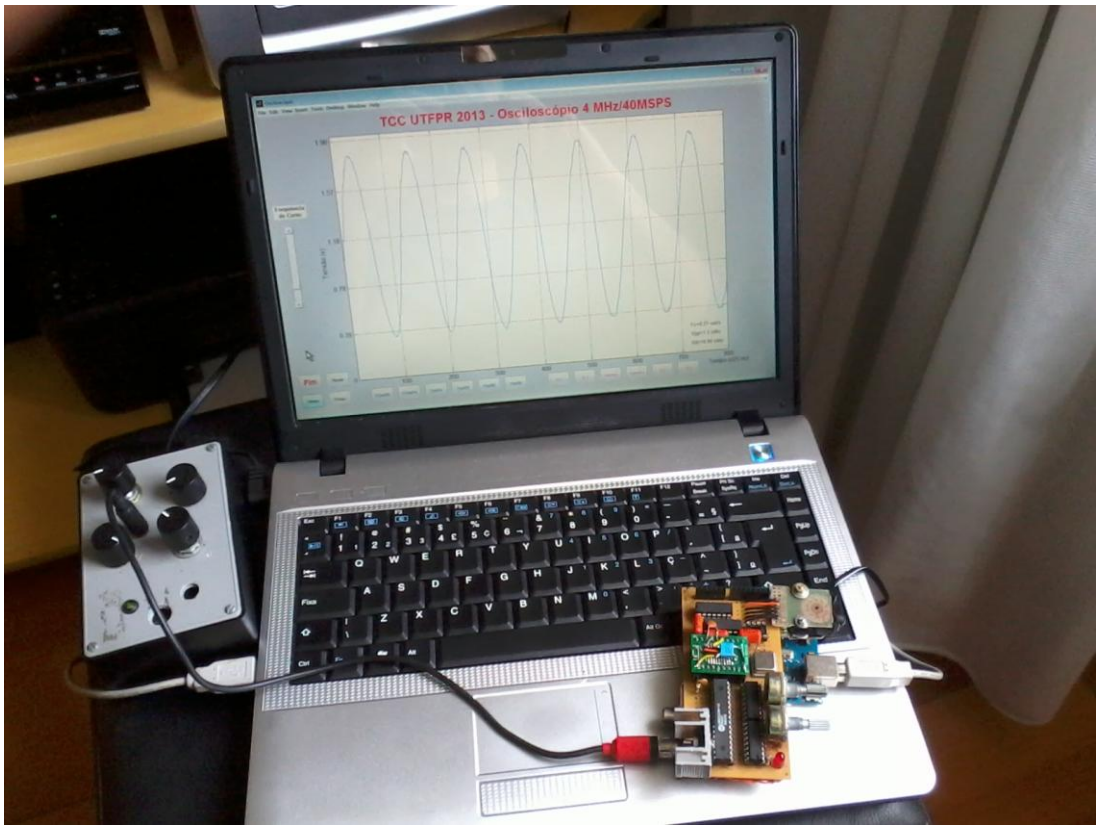


Figura 20 - Osciloscópio conectado a um gerador de funções e ao *notebook*
Fonte: Autoria própria

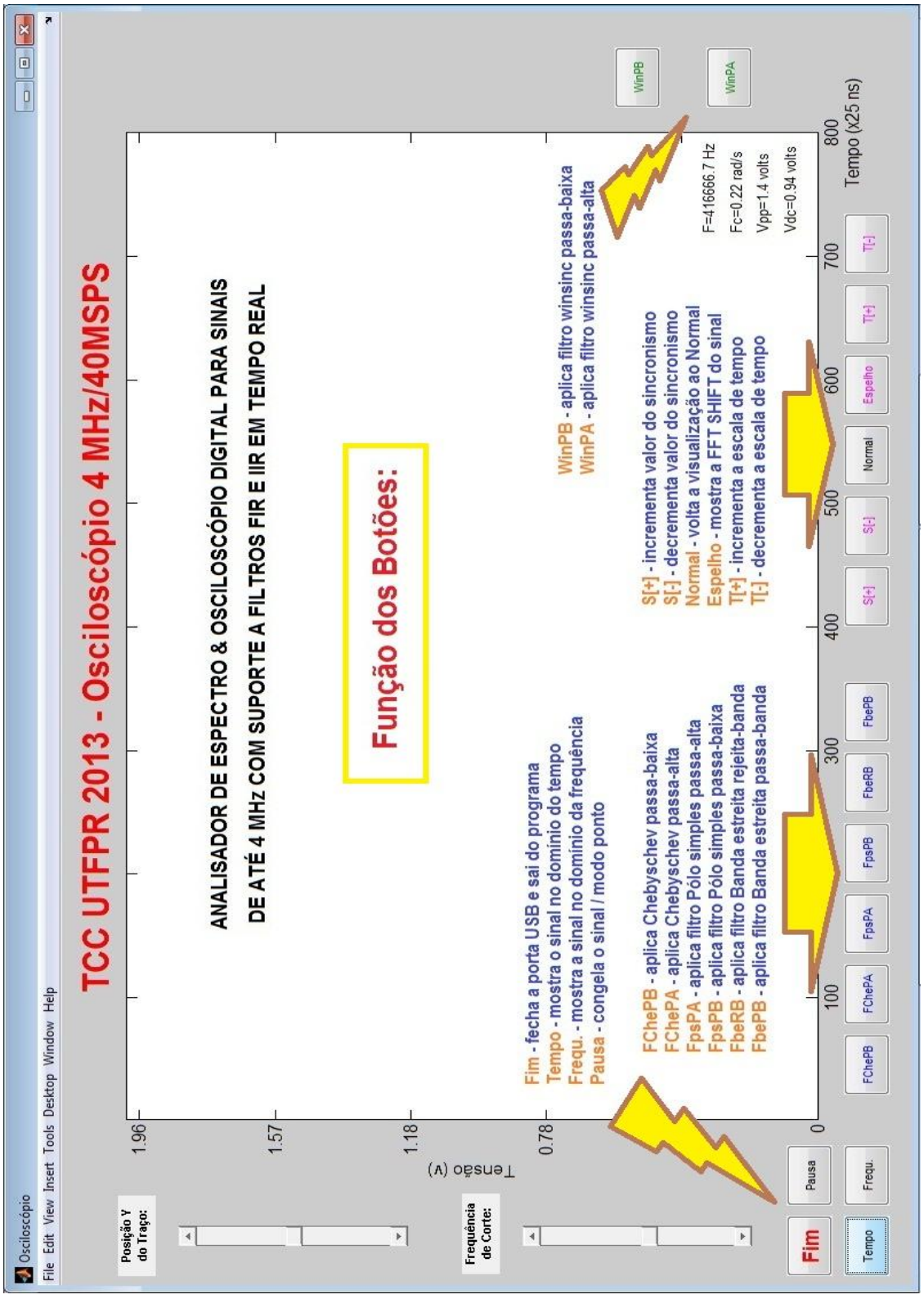


Figura 21 - Função dos botões da interface
Fonte: Autoria própria

5.1 INSTALAÇÃO

O primeiro passo para instalar os programas necessários ao funcionamento do circuito é transferir o código fonte listado no apêndice C ao ATMEGA8 do arduino. Este programa, assim como os demais, deve ser transferido sem os números de linha que constam nas colunas (1 a 62). Com apenas a placa arduino, modelo UNO ou 2009, conectada via cabo USB com o computador pessoal, inicialmente deve-se configurar o modelo do arduino utilizado e a porta de comunicação. Por exemplo, arduino UNO e porta COM11. Se houver dúvida quanto ao número da porta que o sistema operacional reconheceu, efetuar uma consulta ao Painel de Controle, Sistema e Segurança, Gerenciador de Dispositivos. Com o código listado no apêndice C copiado para a IDE arduino 1.0, clicar no ícone de *upload*. O programa será compilado e transferido pelo cabo USB ao arduino, gerando uma mensagem "OK" caso não ocorram erros. Terminada a transferência, desoldar o resistor RN4 da figura 16 a fim de liberar a porta USB para transmissão de dados pelo pino 28 do IC8.

O segundo passo é compilar o programa em Matlab do apêndice D. Os fontes das listagens dos apêndices " F, G, H e I " também devem ser transferidos para a mesma pasta escolhida para ele. Aberto o Matlab com este fonte, o programa deve ser testado antes da compilação definitiva através da execução pela tecla F5. Para gerar o arquivo ".exe" correspondente a este ".m" já testado, o comando "mcc -m Osc4MHzV4.m -o Osc4MHzV4" deve ser digitado na linha de comando do Matlab. Ao término da lenta compilação que demora vários minutos, o programa "Osc4MHzV4.exe" pode ser executado diretamente da linha de comando do Windows 7, com o Matlab fechado previamente. É aconselhável criar um atalho para este executável.

Antes, porém, a placa do circuito osciloscópio deve estar encaixada no arduino e o cabo USB conectado no computador pessoal. A fonte de 9 volts também deve estar conectada ao arduino pois o circuito não deve ser alimentado apenas pelo cabo USB. Se nenhum sinal for aplicado na entrada X3, uma linha aproximadamente reta deve aparecer na tela conforme visto na figura 22. Ao injetar uma senóide de 200KHz / ~2Volts nesta mesma entrada, o sinal deverá ser próximo ao da figura 23. Caso nada apareça e o circuito esteja sem panes, um ajuste em R3

e/ou R2 deve ser procedido até que a forma de onda correspondente ao sinal seja visualizada.

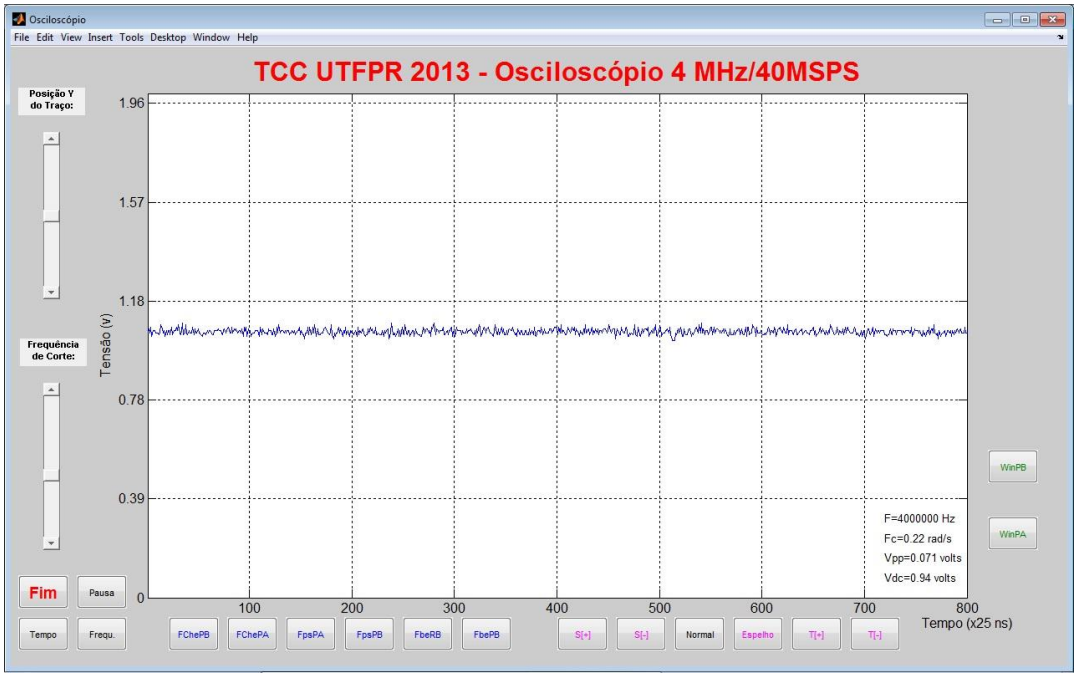


Figura 22 - Osciloscópio sem sinal na entrada
Fonte: Autoria própria

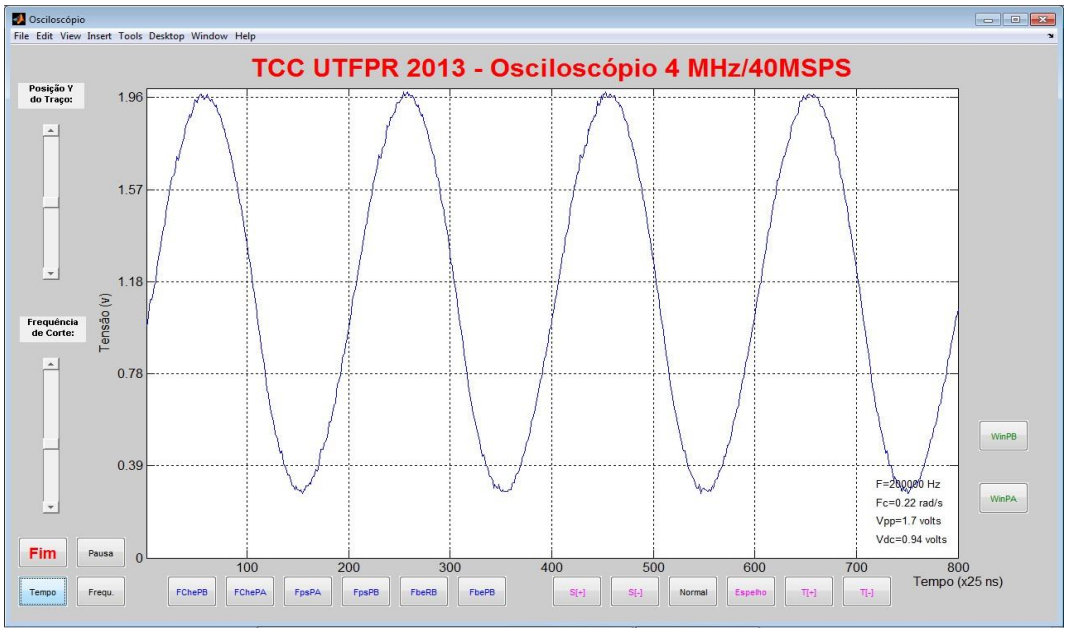


Figura 23 - Osciloscópio com sinal senoidal na entrada
Fonte: Autoria própria

Para compilar o programa em C do apêndice B, a IDE Bloodshed DEV C++ versão 4.9.9.2 deve ter previamente instalada a biblioteca de Michael Main, conforme descrito em “http://www.uniqueness-template.com/devcpp/”. O arquivo “graphics.h” deve ser copiado para a pasta “C:\Dev-Cpp\include” e o “libbgi.a” para “C:\Dev-Cpp\lib”. Ambos arquivos e o compilador DEV C++ podem ser descarregados gratuitamente nos *sites* indicados nas referências.

5.2 ENSAIOS

Consultando a página na internet de um conceituado fabricante de osciloscópios digitais, foi possível recolher alguns dados para determinar a faixa de frequência máxima do osciloscópio. A empresa Tektronix (2012) considera que cinco amostras por período do sinal são o mínimo de pontos para a visualização satisfatória da forma de onda mostrada na tela, mas o ideal para uma reconstrução mais precisa e confortável são dez amostras por ciclo de sinal. O critério de Nyquist (1928) afirma que a taxa de amostragem deve ser superior ao dobro da máxima frequência do sinal a ser digitalizado, porém quanto mais amostras forem realizadas, menor será o erro da interpolação gerado pela função “plot”.

Se o módulo arduino tivesse sido programado para fazer amostragens com tempo variável, ao invés de tempo fixo, a máxima frequência que o osciloscópio poderia medir chegaria a 40MHz, teoricamente. É uma técnica de amostragem chamada de “tempo equivalente”, explicada nas notas de aplicação da Tektronix (2012). Difere-se da técnica de “tempo real”, empregada neste protótipo, ao usar um circuito de disparo para ler a tensão da forma de onda em momentos aleatórios do ciclo de sinal. Só é válida para sinais periódicos. As amostragens tomam vários ciclos completos do sinal para serem finalizadas. Esta técnica é usada em vários modelos de osciloscópios comerciais a fim de aumentar sua faixa de frequência útil.

Fazendo uso de um microcomputador PC baseado no microprocessador Intel Core2Duo 2.2 GHz, foi possível desenhar na tela dez senóides na frequência de 4MHz, todas com aproximadamente 70% da amplitude original. Então, o ensaio feito sob este critério, determina que a frequência máxima prática do circuito montado seja cerca de 4MHz, isto é, atende apenas a faixa inferior das radiofrequências. Esta faixa máxima é geralmente adequada para a maioria dos experimentos de laboratório dos estudantes de tecnologia ou engenharia.

Para expandir o limite de 4MHz para a máxima frequência a ser exibida, mantendo as amostragens em tempo real, um circuito mais elaborado deveria ser montado. Com a adição de uma memória e conversor A/D mais rápidos e a posterior transmissão dos mesmos em velocidade menor para o PC, a frequência máxima pode chegar a bilhões de amostras por segundo, como encontrado nos melhores osciloscópios digitais disponíveis. A dificuldade para alcançar este objetivo é o custo elevado dos componentes, a manufatura da placa de circuito impresso que se torna mais crítica e a indisponibilidade dos conversores A/D nos fornecedores de componentes eletrônicos que atuam no mercado nacional. Outras melhorias no programa podem ser facilmente adicionadas, como a adição de mais um canal de entrada para exibição simultânea de duas formas de onda. Como apresentado, o protótipo é um osciloscópio monocanal. Outro ponto a ser aperfeiçoado é o controle automatizado da amplificação/atenuação do sinal mostrado na tela, assim como a comutação da faixa de amostragem (faixas de 1 a 5), pois o programa apresentado só considera a escala 1, sendo as demais dependentes de comutação manual na chave S1 por parte do usuário.

As dificuldades para o funcionamento do circuito foram superadas aos poucos. Quando este projeto foi montado, não funcionou satisfatoriamente na primeira tentativa. As formas de onda na tela mostravam interrupções aleatórias que sugeriam erro na transmissão de dados entre a placa do osciloscópio e o PC. Só após uma redução física da placa de circuito impresso, isto é, uma compactação da montagem, é que os sinais apareceram com poucas interrupções, de forma análoga aqueles visualizados em um osciloscópio tradicional. Esta compactação na montagem diminui a influência das capacitâncias e indutâncias parasitas no funcionamento do circuito.

A principal causa dos *spikes* foi não ter sido confeccionada uma placa de circuito impresso tradicional, mas sim uma placa de circuito impresso padrão como a da figura 24. O *datasheet* para o circuito integrado conversor analógico/digital empregado avisa a necessidade de uma placa de circuito impresso onde os planos de terra e das tensões de alimentação (+5Vcc e GND) sejam complementares, de tal forma que uma blindagem contra interferências seja efetiva. Mas a falta desta blindagem pode ser compensada por uma construção cuidadosa da placa de circuito impresso padrão, o mais compacta possível. O motivo de não ter sido construída uma placa dedicada foi testar a viabilidade de usar esta configuração mais

econômica, o que foi comprovado conforme visto nas diversas cópias de tela deste relatório e na demonstração prática de funcionamento. Este fato é fundamental para facilitar a montagem aos alunos que não tem experiência em confeccionar uma placa pelo método fotográfico. Também garante um custo baixo de montagem, uma vez que não será obrigatório adquirir os materiais típicos para a fabricação caseira por este processo, tais como lâmpada ultravioleta, percloreto de ferro, papel para transferência térmica, cortador e furador para circuito impresso, caneta ácidorresistente, impressora laser para impressão, etc.

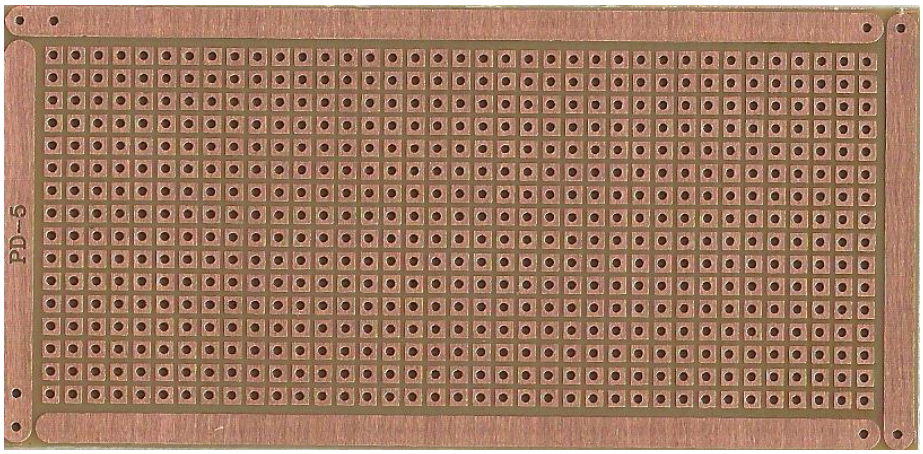


Figura 24 - Placa de circuito impresso padrão usada
Fonte: Autoria própria

6 AFERIÇÃO

Este capítulo tem como finalidade registrar os testes finais da placa osciloscópio feitos no laboratório da sala Q103 da UTFPR campus Curitiba. Os professores César Janeczko e Luís Alberto Lucas acompanharam todos os testes que aferiram se os parâmetros projetados para o osciloscópio construído corresponderam ou não aos valores quantitativos e qualitativos esperados. As principais perguntas a serem respondidas foram: a placa osciloscópio realmente consegue mostrar sinais de até 4MHz? Os valores de tensão e frequência medidos são aceitáveis?

O osciloscópio digital comercial usado como padrão de comparação é o Tektronix modelo TDS1001B de 40MHz (500 MSPS), conforme a figura 25.

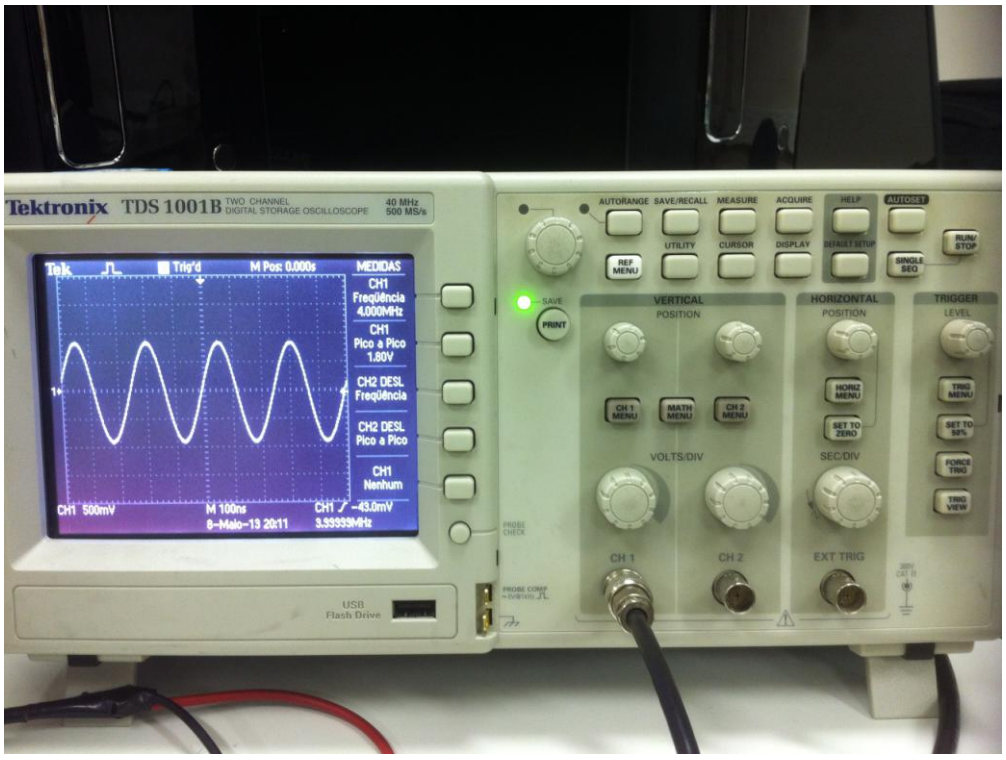


Figura 25 - Tektronix modelo TDS1001B de 40MHz
Fonte: Autoria própria

O gerador de funções adotado é apresentado na figura 26, um Agilent modelo 33521A que tem capacidade para gerar sinais até 30MHz.

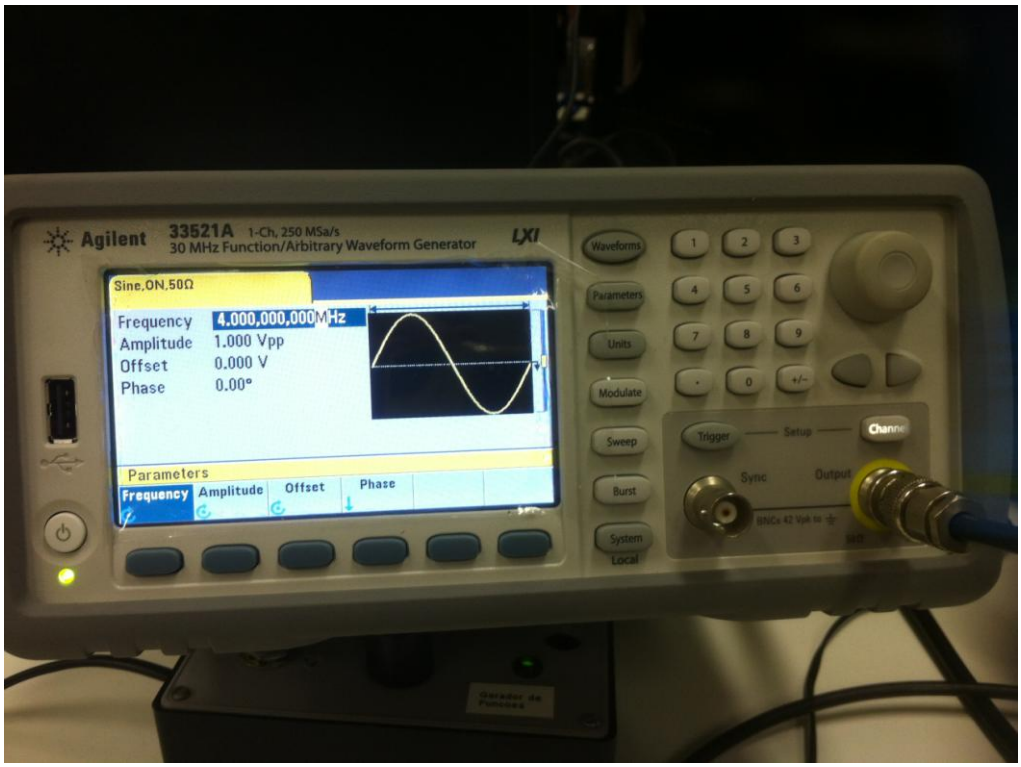


Figura 26 - Agilent modelo 33521A
Fonte: Autoria própria

A configuração de teste é vista na figura 27. O gerador de funções Agilent 33521A foi ajustado para que sua saída alimente ao mesmo tempo a entrada da placa osciloscópio deste projeto e a entrada CH1 do osciloscópio Tektronix TDS1001B.

A placa osciloscópio digitaliza o sinal gerado pelo Agilent 33521A e envia serialmente os dados digitais para uma porta USB do *notebook* Core2Duo que executa o programa emulador de osciloscópio “Osc4MHzV4.m”.

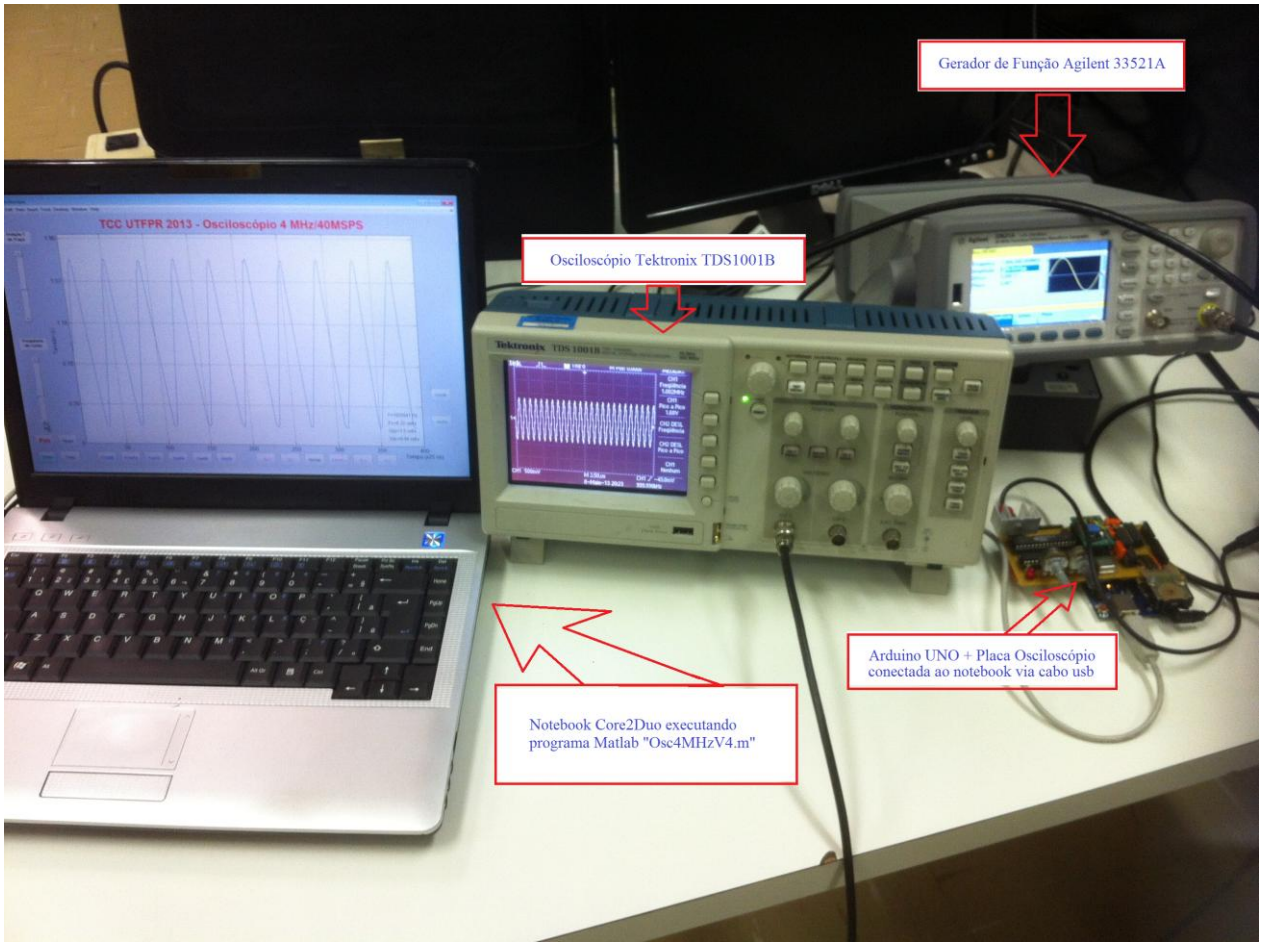


Figura 27 – Configuração montada para teste
Fonte: Autoria própria

Na figura 28 o primeiro teste está registrado. O gerador de funções gera um sinal senoidal de 4 MHz/1.8v que é medido pelo osciloscópio Tektronix como sendo 4.019 MHz. O programa “Osc4MHzV4.m” calcula esta frequência como 4000000 Hz. Portanto, o resultado é plenamente concordante com o ajustado no gerador de funções. Na tela do notebook nota-se que o sinal senoidal está com uma aparência um pouco mais distorcida que a do osciloscópio comercial. O Tektronix registrou a tensão pico-a- pico como sendo 1.8 volts e o notebook 1.5 volts. O erro do notebook, em relação ao gerador, foi zero na frequência e 16.7% na tensão. Os erros registrados nesta aferição e nas posteriores serão discutidos no próximo capítulo.

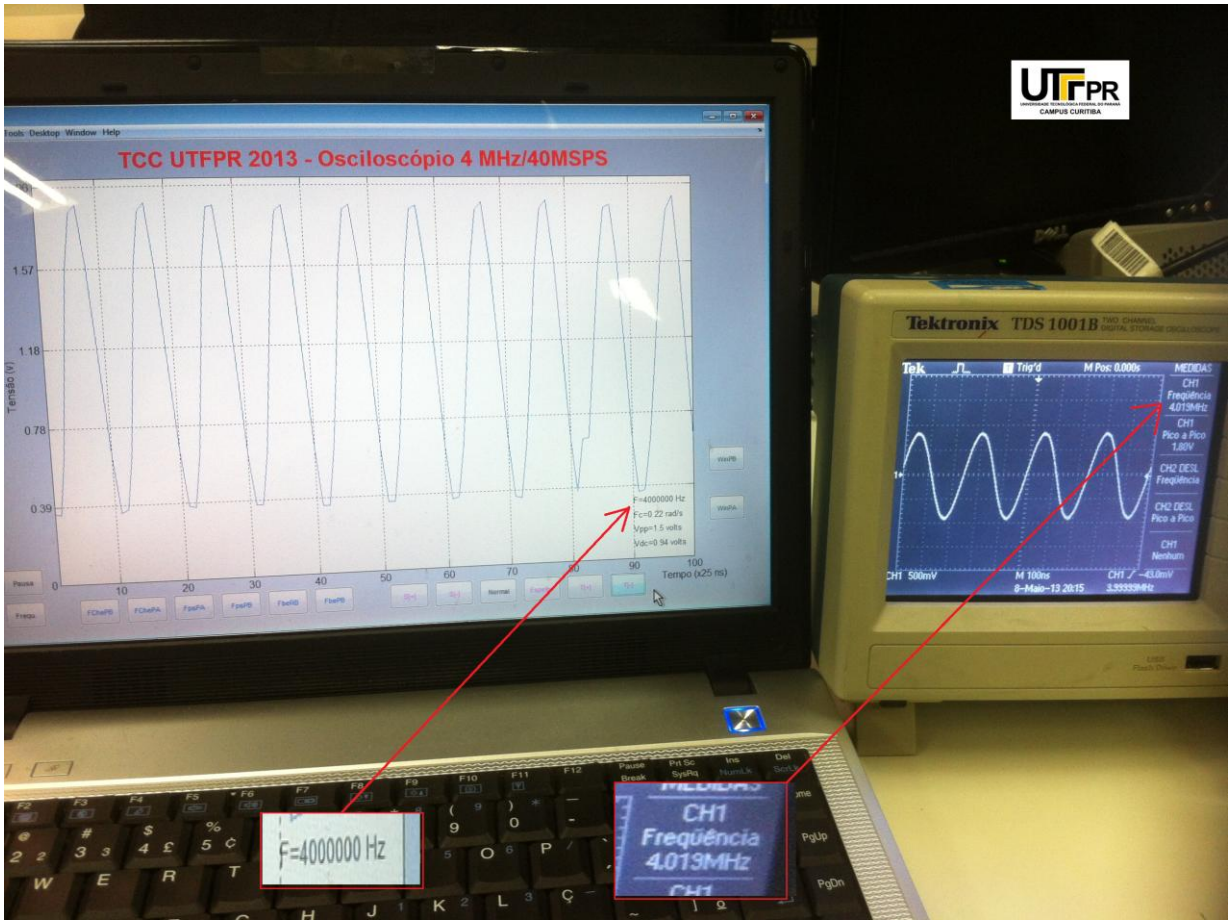


Figura 28 – Teste com senóide de 4 MHz
Fonte: Autoria própria

Na figura 29 o teste é repetido com 4 MHz/2.0v mudando a forma de onda no gerador de funções para quadrada. O notebook registra novamente 4000000Hz e desta vez o osciloscópio Tektronix mostra 4.000 MHz, ambos concordando com o valor ajustado no gerador de funções. O Tektronix registrou a tensão pico-a-pico como sendo 2.0 volts e o notebook 1.8 volts. O erro do notebook foi zero na frequência e 10.0 % na tensão.

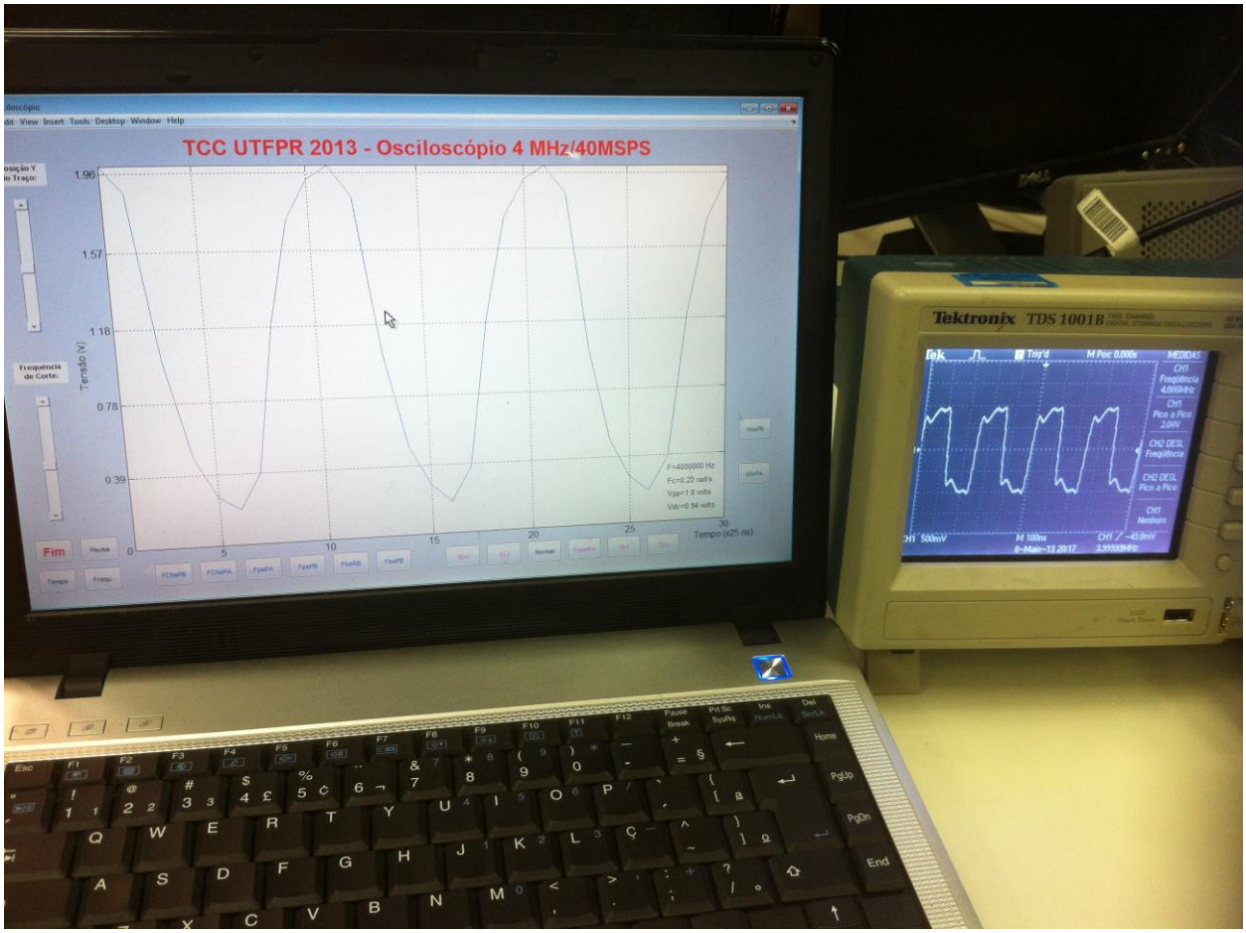


Figura 29 – Teste com onda quadrada de 4 MHz
Fonte: Autoria própria

A forma de onda aparece com algumas distorções tanto na tela do osciloscópio Tektronix como na tela do notebook. Estas distorções já eram esperadas, pois a taxa de amostragem finita limita a capacidade de reconstrução perfeita da onda quadrada, que é a de digitalização mais difícil.

Na figura 30 o teste é com uma onda quadrada ajustada no gerador para 1MHz e 1.70 volts pico-a-pico. O notebook registrou 1212121Hz / 1.6v e o osciloscópio Tektronix 1.000MHz / 1.74 v. As telas de ambos mostraram formas de onda quadrada com desenhos muito semelhantes. O erro do notebook foi de ~21.21% na frequência medida e ~5.88 % na tensão.

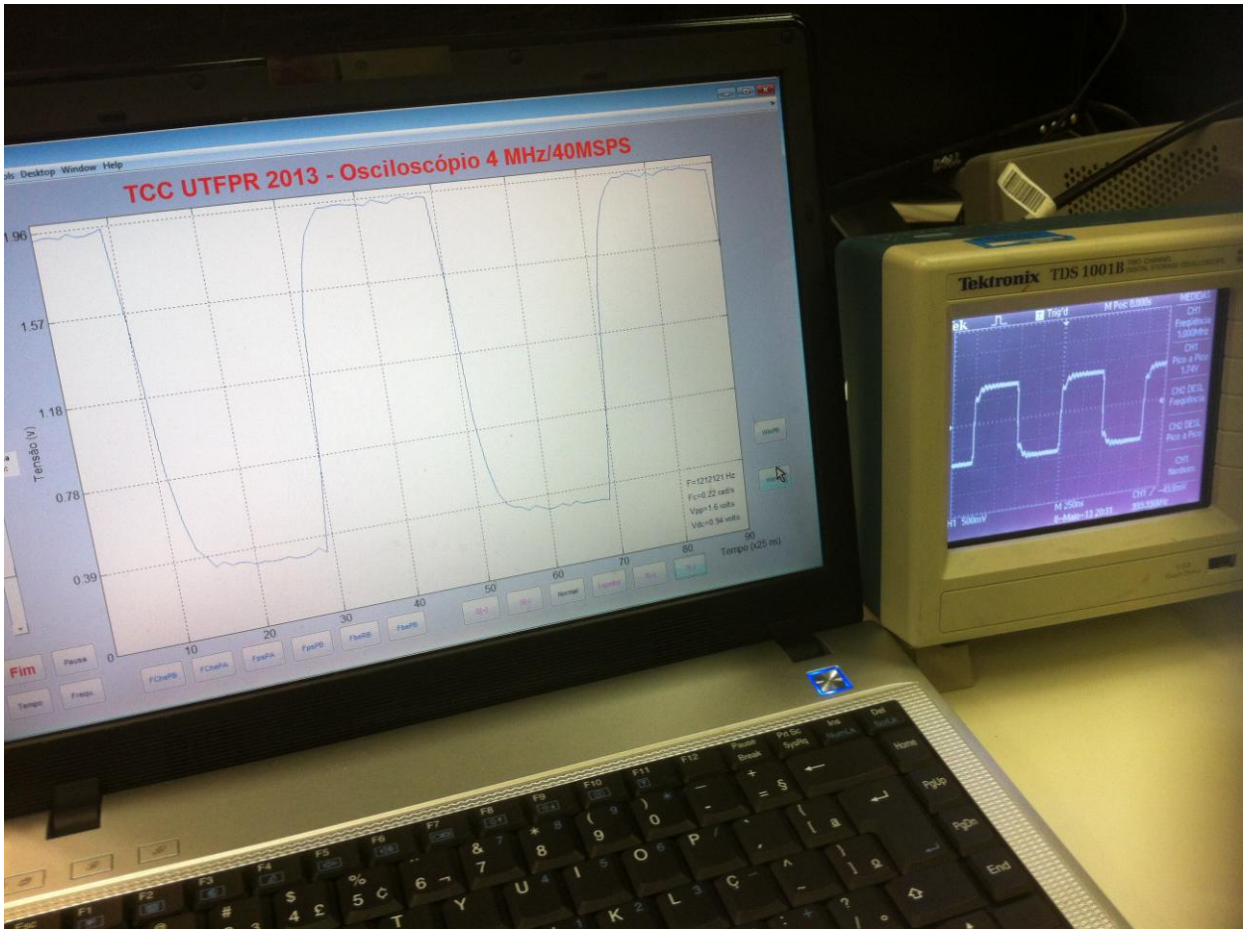


Figura 30 – Teste com onda quadrada de 1 MHz
Fonte: Autoria própria

Na figura 31 o teste é com uma onda triangular ajustada no gerador para 200KHz e 2.0 volts pico-a-pico. O notebook registrou 199005Hz / 1.8v e o osciloscópio Tektronix 198KHz / 2.0 v. As telas de ambos mostraram as formas de onda triangular com desenhos muito semelhantes. O erro do notebook foi de ~0.49% na frequência medida e 10.00 % na tensão.

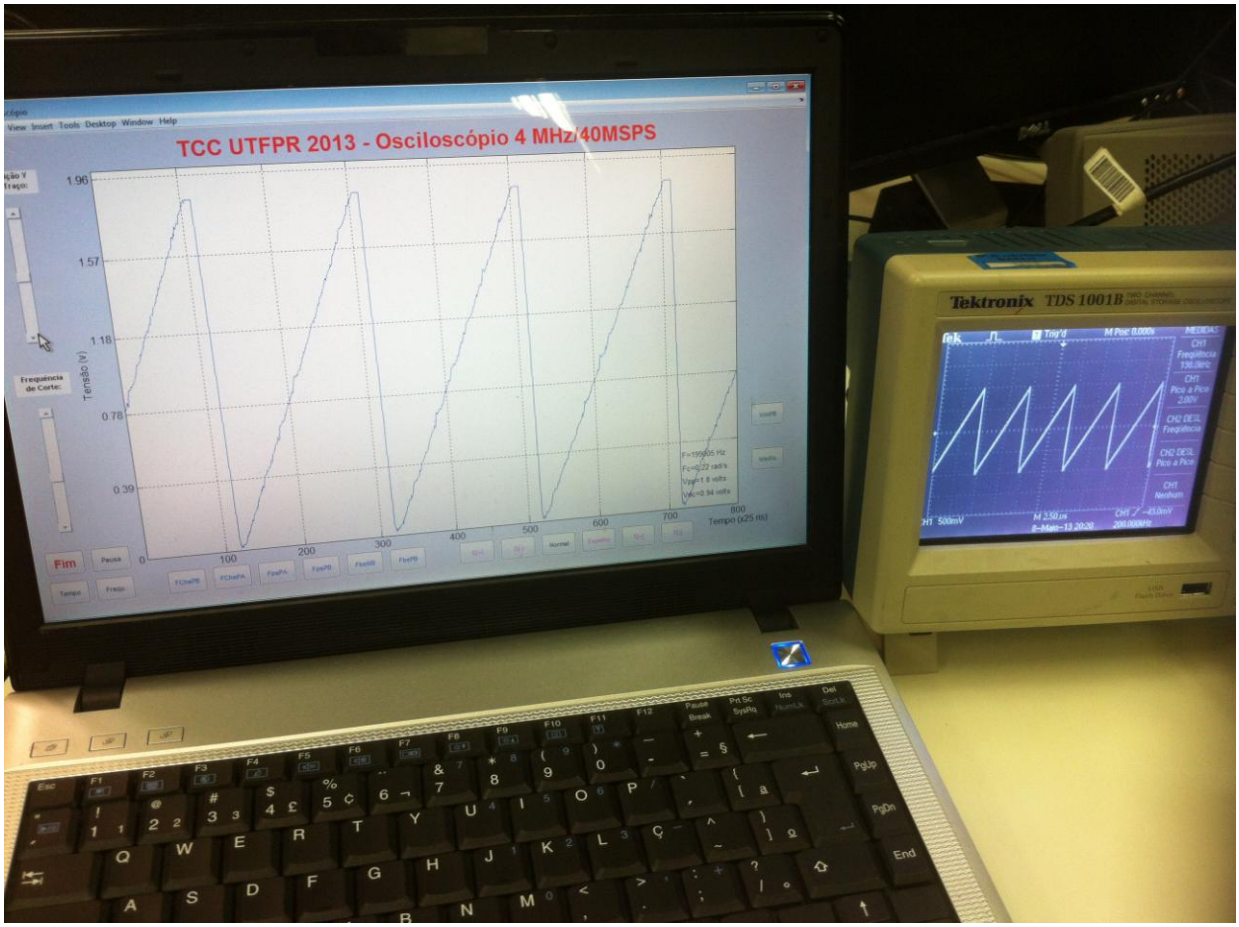


Figura 31 – Teste com onda triangular de 200KHz
Fonte: Autoria própria

O erro do *notebook* em relação ao gerador, calculado pela média aritmética das medidas realizadas, foi 5.4% na frequência e 10.6% na tensão.

7 CONCLUSÃO

Os resultados gerados pelo programa em Matlab foram comparados com os gerados por um osciloscópio comercial Tektronix modelo TDS1001B, disponível no laboratório da UTFPR. Eles demonstraram que é possível e economicamente viável a construção de um osciloscópio digital e analisador de espectro para sinais de até 4 MHz com finalidades estudantis, laboratoriais ou técnicas pois os erros nas medidas estão em patamares aceitáveis. Os piores erros foram os de tensão. O conversor TLC 5540 apresenta apenas 7 *bits* (não 8 *bits*) como o número efetivo de *bits* para a taxa de 40 MSPS, conforme informa o *datasheet* da Texas, 1995. Este limitador, por si só, impõe um erro de quase um ponto porcentual. O fato das tensões medidas sempre estarem abaixo da gerada revela um erro a ser investigado em um trabalho futuro. Uma hipótese a ser testada é o fato de não ter sido implementado um circuito de compensação resistor/capacitor na entrada do amplificador operacional OPA2604.

O objetivo específico de usar um microcomputador como um osciloscópio digital foi, portanto, alcançado. O protótipo apresentou recursos semelhantes a um osciloscópio digital comercial, como as medidas de frequência, período e tensão demonstraram nos ensaios.

O principal motivador para adotar a linguagem Matlab para a versão final do osciloscópio 4MHz foi o tempo de desenvolvimento. Seria perfeitamente possível continuar desenvolvendo o programa em C do apêndice B ou até mesmo em Basic, como exemplificado no apêndice E, mas as dificuldades decorrentes da elaboração das diversas funções não existentes nestas linguagens inevitavelmente atrasariam o cronograma de entrega deste TCC. Com o Matlab não foi necessário gastar tempo criando rotinas para cálculos de transformadas de Fourier, alguns tipos de filtros, formatação dos gráficos, interfaces gráficas, comunicação serial, etc.

O custo de construção foi bem menor do que o de um equipamento comercial, permanecendo abaixo de um terço do salário mínimo brasileiro (R$187,60). A memória UM61512 não precisou ser adquirida, foi retirada sem custo de uma placa mãe de processador Intel 486DX fabricada em 1995. Como esta memória cache de 40MHz é instalada em soquete de pinagem padrão DIL, seu uso a partir de lixo eletrônico descartado é possível e recomendado. Este foi o motivo de

não se adquirir uma memória de 2 Kbytes, mas sim aproveitar esta com 64 Kbytes. A chave rotativa S1 e o oscilador a cristal também foram reaproveitados de um antigo modem Elebra de 1986. Esta chave foi especificada para ter dois pólos por cinco posições devido a previsão futura de utilizar o ramo que está no momento sem uso como um elemento de realimentação ao circuito. Com isto a chave pode, após processamento do arduino e transmissão serial ao programa em Matlab, sinalizar a troca de escalas de tempo na tela de forma automática e não manual como implementado até aqui. Para efetivar esta facilidade o *firmware* do arduino necessitará de atualização.

A didática das aulas de Processamento Digital de Sinais tem muito a ganhar com a construção deste projeto, pois o estudante pode comprovar que aqueles algoritmos de filtros e de transformada rápida de Fourier vistos em aula realmente funcionam para os sinais analógicos do mundo físico. A teoria tem sua necessária comprovação experimental (prática).

O programa do apêndice C demonstrou um método simples para fazer um microcontrolador ATMEGA emular um osciloscópio em linguagem “C”, dentro do ambiente “Arduino IDE”. O mesmo pode ser afirmado do programa do apêndice D, desenvolvido em linguagem “C” sob a IDE DEV C++. Implementar um osciloscópio e analisador de espectro na linguagem “MatLab”, com opção de filtragem em tempo real do sinal de entrada, foi um objetivo factível, como verificado experimentalmente. A construção deste osciloscópio digital, com o apoio da arquitetura aberta “Arduino”, não se mostrou inviável, como poderia ser avaliado em um primeiro momento devido ao relativo pequeno poder de processamento deste microcontrolador de 8 *bits* comparado com os mais atuais, e custosos, de 32 *bits*. A principal dificuldade logística foi adquirir os componentes eletrônicos fundamentais, como o circuito integrado TLC 5540 e seu adaptador SOP20/DIP.

Felizmente este conversor de baixo custo já está disponível nos principais fornecedores de componentes que operam nacionalmente, sem precisar recorrer a uma importação por compra direta através de cartão de crédito internacional, com o inconveniente adicional da lentidão alfandegária.

Com relação a outras tecnologias que poderiam ter sido abordadas neste projeto, uma em especial deve ser comentada. É a implementação do osciloscópio através de circuitos *field-programmable gate array (*FPGA). Este é, sem dúvida, o melhor caminho para conseguir um desempenho profissional a custo razoável. Neste

projeto o FPGA não foi utilizado porque faria o custo final ser pelo menos três vezes maior. As placas de desenvolvimento que dão suporte a tecnologia FPGA ainda são muito mais dispendiosas que as de microcontroladores genéricos como o do arduino UNO. Para viabilizar sua construção dentro do orçamento proposto seria necessário utilizar as placas FPGA dos laboratórios da UTFPR. Como elas teriam que ser devolvidas no final do projeto, o objetivo de ter um osciloscópio em casa - durante todo o curso - não seria alcançado. O quadro 2 tem as especificações finais levantadas para o projeto osciloscópio 4MHz.

Especificação	**Valor**	**Comentário**
Máxima frequência visualizável	4MHz	Com 10 amostras por ciclo
Máxima taxa de amostragem	40 MSPS	Determinada pelo TLC5540
Resolução vertical	8 *bits*	Determinada pelo TLC5540
Tipo de interface	USB 1.1	Determinada pelo FT232RL
Dimensões do circuito	10x5x4cm	Placa padrão + arduino UNO
Mínima frequência visualizável	100Hz	Determinada pelo *firmware*
Máxima tensão sem atenuador 1:10	±2.0v	Sem ponteira de atenuação 1:10
Máxima tensão com atenuador 1:10	±20v	Com ponteira atenuadora 1:10
Impedância das 2 entradas (DC/AC)	100kΩ	Determinada por R2/ OPA2604
Consumo do circuito	0.6A	Medido para carga máxima
Precisão da tensão medida	±10.6%	Média da aferição
Precisão da frequência medida	±5.4%	Média da aferição
Largura de banda analógica	20MHz	Determinada pelo OPA2604
Tamanho da memória *buffer*	2Kbytes	Expansível para 64Kbytes
Quantidade de canais	1	Expansível para 2 ou mais
Sistema operacional adotado	Windows 7	Testado na versão SP2 32 bits

Quadro 2 – Especificações do projeto
Fonte: Autoria própria

Concluindo, o projeto objeto deste TCC demonstrou na prática que estudantes possuidores de um computador pessoal, semelhante ao *notebook* de entrada empregado, podem ensaiar os principais circuitos vistos no laboratório acadêmico fora da universidade, sem ter que adquirir um osciloscópio comercial cujo custo é, no mínimo, cinco vezes maior que o do circuito montado.

REFERÊNCIAS

AMP. Disponível em: <http://www.ti.com/lit/ds/sgls209/sgls209>. Acesso em: 14 dez. de 2012.

ATMEL, Microcontrolador ATMEGA8. Disponível em: <http://www.atmel.com/images/doc2486.pdf>. Acesso em: 14 dez. de 2012.

BANZI, Massimo. **Getting Started with Arduino**. California: O'Reilly Media/Make, 2011.

BLOODSHED. Disponível em: <http://www.bloodshed.net/>. Acesso em: 14 dez. de 2012.

DRIVER. Disponível em: <http://www.ti.com/lit/ds/symlink/max232.pdf>. Acesso em: 14 dez. de 2012.

FALL, Kevin. **TCP/IP Illustrated Volume 1 Second Edition**. Boston: Addison-Wesley Professional, 2011.

HAYKIN, Simon; VEEN, Barry V. **Sinais e Sistemas**. 5. ed. Tradução José Carlos Barbosa dos Santos. Porto Alegre: Bookman, 2001.

IDE ARDUINO. Disponível em: <http://www.arduino.cc/>. Acesso em: 14 dez. de 2012.

IDE DEV C++. Disponível em: <http://www.bloodshed.net/>. Acesso em: 14 dez. de 2012.

JANECZKO, César. Disponível em: < http://pessoal.utfpr.edu.br/janeczko/>. Acesso em: 14 dez. de 2012.

LONGAIR, Malcolm. **Theoretical Concepts in Physics**. Cambridge: Cambridge University Press, 2003.

MAIN, Michael. **Borland BGI Graphics emulation for the MingW (GCC port) Compiler**. Disponível em: <http://winbgim.codecutter.org>. Acesso em: 14 dez. de 2012.

MATLAB. Disponível em: <http://www.mathworks.com/>. Acesso em: 14 dez. de 2012.

MICROSOFT. Disponível em: <http://www.microsoft.com/pt-br/default.aspx/>. Acesso em: 14 dez. de 2012.

NASCIMENTO, José Aguiar**. Home Page**. Disponível em: <http://www.dimap.ufrn.br/~aguiar/Livros/Conectiva9Usuario/usb.html>. Acesso em: 14 dez. de 2012.

NYQUIST, H. **Certain topics in telegraph transmission theory**. Trans. AIEE, vol. 47, pp. 617–644, 1928.

Open GL. Disponível em: < http://www.opengl.org/>. Acesso em: 14 dez. de 2012.

RADIG, Ulrich. Disponível em: <http://www.ulrichradig.de/home/index.php/avr/avr-dso>. Acesso em: 14 dez. de 2012.

RITTER, Terry. Disponível em: < http://www.ciphersbyritter.com/arts/crcmyst.htm>. Acesso em: 14 dez. de 2012.

SANTOS, Marcelo M. Osciloscópio Digital para IBM-PC. **Revista Saber Eletrônica**, São Paulo, n. 228, p. 46-52, jan. 1992.

SANCHEZ, Alejandro. **Applies a windowed sinc filter**. Disponível em: < http://www.mathworks.com/matlabcentral/fileexchange/8574-winsinc>. Acesso em: 14 dez. de 2012.

SLAZAS, Robert. **Scanports function scans for available serial ports**. Mathworks, 2011. Disponível em: <http://www.mathworks.com/matlabcentral/fileexchange/33273-scanports-function-scans-for-available-serial-ports/content/scanports.m>. Acesso em: 14 dez. de 2012.

TAKASE, Fabio Kawaoka. **Home Page**. Disponível em: < http://monoceros.mcca.ep.usp.br/ESL/disciplinas/pmr2300-computacao-para-automacao/pmr2300_comunicacaoseriali.pdf>. Acesso em: 14 dez. de 2012.

TEKTRONIX, Inc. Disponível em: <http://www.tek.com/>. Acesso em: 14 dez. de 2012.

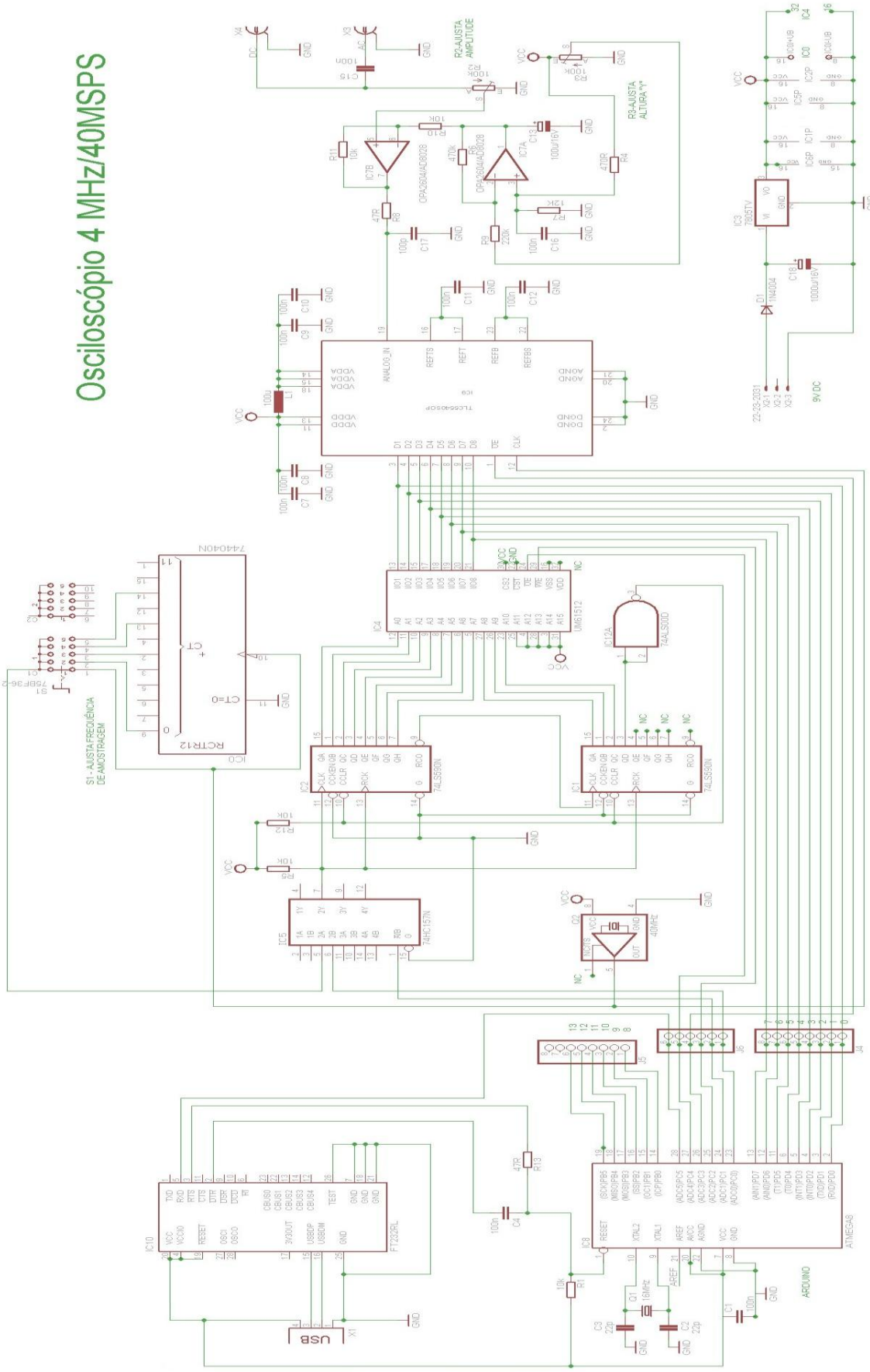
TEXAS, Instruments Inc. Disponível em: <http://www.ti.com/lit/ds/symlink/tlc5540.pdf>. Acesso em: 14 dez. de 2012.

TORRES, Gabriel**. Clube do hardware**. Disponível em: <http://www.clubedohardware.com.br/printpage/ Como-Conversores-Analogico-Digital-Funcionam/1307>. Acesso em: 14 dez. de 2012.

WIN API. Microsoft, 1995. Disponível em: < http://msdn.microsoft.com/en-us/library/ff802693.aspx>. Acesso em: 14 dez. de 2012.

YODER, Nathanael. **PeakFinder finds local maxima or minima in a noisy signal**. Disponível em:< http://www.mathworks.com/matlabcentral/fileexchange/25500-peakfinder> Acesso em: 14 dez. de 2012.

APÊNDICE A – Esquema elétrico do projeto OSCILOSCÓPIO DIGITAL DE 4 MHz



APÊNDICE B – Osciloscópio Digital implementado em linguagem C (oscilo10.c)

```
// CURSO DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES da UTFPR
// OSCILOSCÓPIO DIGITAL DE 4 MHz PARA ESTUDANTES DE TECNOLOGIA
// Programação: Marcelo Machado dos Santos código 70343
// Documentação: Gustavo Henrique Duarte código 1052608
// TCC orientado pelo Professor: César Janeczko, M.Sc.

1 // Programa compilado na IDE Bloodshed DEV C++ versão 4.9.9.2
2 // Necessita instalar a biblioteca de Michael Main conforme
3 // descrito em http://www.uniqueness-template.com/devcpp/
4
5   #include <stdio.h>
6   #include <math.h>
7   #include <stdlib.h>
8   #include <iostream>
9   #include <graphics.h>
10  #include <conio.h>
11  #include <windows.h>
12  using namespace std;
13  HANDLE hPort = CreateFile("COM11",
14  GENERIC_WRITE|GENERIC_READ,0,NULL,OPEN_EXISTING,FILE_ATTRIBUTE_NORMAL,NULL);
15  DCB dcb;
16
17  bool writebyte(char* data)
18     {
19     DWORD byteswritten;
20     if (!GetCommState(hPort,&dcb))
21        {
22        cout<<"\nPorta Serial COM: sem acesso!\n";
23        return false;
24        }
25     FillMemory(&dcb, sizeof(dcb),0);
26     dcb.BaudRate = CBR_115200;
27     dcb.ByteSize = 8;
28     dcb.Parity = NOPARITY;
29     dcb.StopBits = ONESTOPBIT;
30     if (!SetCommState(hPort,&dcb))
31     return false;
32     bool retVal = WriteFile(hPort,data,1,&byteswritten,NULL);
33     return retVal;
34     }
35
36  int ReadByte()
37     {
38     int Val;
39     BYTE Byte;
40     DWORD dwBytesTransferred;
41     DWORD dwCommModemStatus;
42     if (!GetCommState(hPort,&dcb))
43     return 0;
44     SetCommMask (hPort, EV_RXCHAR | EV_ERR);
```

```
45     WaitCommEvent (hPort, &dwCommModemStatus, 0);
46     if (dwCommModemStatus & EV_RXCHAR)
47     ReadFile (hPort, &Byte, 1 , &dwBytesTransferred , 0);
48     Val = Byte;
49     return Val;
50     }
51
52  void limpa()
53        {
54        FlushFileBuffers( hPort );
55        PurgeComm (hPort,PURGE_TXABORT|PURGE_RXABORT|PURGE_TXCLEAR|PURGE_RXCLEAR);
56        }
57
58    int main()
59    {
60     int left, top, bottom, right;
61     short x=0,y=0,a=0,c=0;
62     byte s=120,fora=0,n1=0,n2=0,n3=0,k=0;
63     byte i[2059];
64     char data=65;
65     writebyte(&data);
66     initwindow(800,600,"Osciloscópio",left=200,top=50);
67     setcolor (GREEN);
68       while(!kbhit())
69       {
70        while (k==0)
71           {
72           limpa();
73           if (n1==0x7E && n2==0x7E && n3==0x7E && k==0)
74              {
75              k=1;
76              }
77           n1=ReadByte();
78           n2=ReadByte();
79           n3=ReadByte();
80           }
81           limpa();
82        for (x = 0; x < 2047; x++)
83           {
84           i[x]=ReadByte();
85           }
86           x=0;
87     while (fora==0)
88     {
89     if (i[x]>s-2 && i[x]<s+2 && i[x]<i[x+2] && i[x+2]<i[x+3] && fora==0 && x<800)
90        {
91        a=x;
92        fora=1;
93        break;
94        }
95     x=x+1;
96     if (x>1246)
```

```
97        {
98        x=0;
99        fora=1;
100        break;
101        }
102    }
103    if (fora==0)
104        {
105        s=s+10;
106        if (s>250)
107            {
108            s=10;
109            }
110        }
111      cleardevice();
112      moveto(0,479);
113      for (x = 0; x < 799; x++)
114          {
115          lineto(x,600-int(i[a+x]*2.35));
116          }
117          k=0,c=0,a=0,fora=0,x=0,n1=0,n2=0,n3=0;
118          limpa();
119     }
120      closegraph();
121      CloseHandle(hPort);
122      return 0;
123  }
```

APÊNDICE C – Programa para fazer um microcontrolador ATMEGA emular um osciloscópio em linguagem C, versão arduino (oscilo10.ino)

```
1  // CURSO DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES da UTFPR
2  // OSCILOSCÓPIO DIGITAL DE 4 MHz PARA ESTUDANTES DE TECNOLOGIA
3  // Programação: Marcelo Machado dos Santos código 70343
4  // Documentação: Gustavo Henrique Duarte código 1052608
5  // TCC orientado pelo Professor: César Janeczko, M.Sc.
6  // Osciloscópio Sincronizado 4MHz/40MSPS + arduino UNO + A/D TLC5540
7  // Funcionou em 21/04/2013 - programa compilado na IDE Arduino 1.0
8
9  #include <SoftwareSerial.h>
10 #define rxPin 9
11 #define txPin 19
12 SoftwareSerial mySerial =  SoftwareSerial(rxPin, txPin);
13 unsigned int x;                    //0-65535
14 byte b;                           //0-255
15
16 void setup()
17 {
18 noInterrupts();
19 pinMode (A4,OUTPUT);          //A4=output
20 pinMode (A3,OUTPUT);          //A3=output
21 pinMode (A2,OUTPUT);          //A2=output
22 pinMode (A1,OUTPUT);          //A1=output
23 pinMode (A0,OUTPUT);          //A0=output
24 digitalWrite(A2,HIGH);            //memoria em read
25 digitalWrite(A4,HIGH);            //memoria desabilitada
26 DDRD=B00000000;                   //porta d input
27 pinMode(rxPin, INPUT);
28 pinMode(txPin, OUTPUT);
29 mySerial.begin(115200);          //tx soft serial
30 }
31
32 void loop()
33 {
34 noInterrupts();
35 digitalWrite(A1,LOW);           //74157 chaveia clk int
36 digitalWrite(A3,LOW);           //habilita a/d
37 digitalWrite(A2,LOW);           //memoria em write
38 digitalWrite(A4,LOW);           //memoria habilitada
39
40 digitalWrite(13,HIGH);          //led on
41 delayMicroseconds(5000);     //Enche buffer 5ms
42 digitalWrite(13,LOW);           //led off
43 delayMicroseconds(5000);     //Enche buffer 5ms
44
45 digitalWrite(A3,HIGH);        //desabilita a/d
46 digitalWrite(A2,HIGH);        //memoria em read
47 digitalWrite(A4,HIGH);        //memoria desabilitada
48 digitalWrite(A1,HIGH);        //74157 chaveia clk ext
49 digitalWrite(A4,LOW);         //memoria habilitada
```

```
50
51 mySerial.write(126);            //7E
52 mySerial.write(126);            //7E
53 mySerial.write(126);            //7E
54 for (x=0;x<2047;x++)
55   {
56   digitalWrite(A0,LOW);
57   digitalWrite(A0,HIGH);      //clk externo
58   b=PIND;                     //le memoria
59   mySerial.write(b);
60   }
61 digitalWrite(A4,HIGH);         //memoria desabilitada
62 }
```

APÊNDICE D – Programa que emula no ambiente Matlab um osciloscópio digital semelhante ao criado pelo algoritmo em C (Osc4MHzV4.m)

```
% CURSO DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES da UTFPR
% OSCILOSCÓPIO DIGITAL DE 4 MHz PARA ESTUDANTES DE TECNOLOGIA
% Programador: Marcelo Machado dos Santos código 70343
% Documentação: Gustavo Henrique Duarte código 1052608
% Orientador: Prof. Ms. César Janeczko

  % Osciloscópio Sincronizado 4MHz/40MSPS + arduino UNO + A/D TLC5540
  % Para compilar no Matlab R2009b: >mcc -m Osc4MHzV4.m -o Osc4MHzV4
  % Funcionou em 21/04/2013 usando: Matlab conectado na porta serial,
% Core2 Duo T6600@2.2GHz,4GBytes,Windows 7 (32 bits),Vídeo 1280x800

 opengl hardware
 delete(instrfindall);
 clc,clear,close all;
 last=scanports();
 eval('Serial1 = serial(cell2mat(last));');
 set(Serial1,'BaudRate',115200);
 set(Serial1,'DataBits',8);
 set(Serial1,'Parity','none');
 set(Serial1,'StopBits',1);
 set(Serial1,'FlowControl','none');
 s=120;k=0;sai=0;congela=0;tela=10;grade=1;t=0;fora=0;fim=0;desl=60;
 spectro=2;sync=0;EscX=800;inc=3;ponto=0;d=0;cong=0;x=1;a=0;off=270;
 n=uint8(zeros(2047));
 B=uint8(zeros(2047));
 Serial1.InputBufferSize = 2050;
 fopen(Serial1);
 set(Serial1, 'DataTerminalReady', 'off'); pause(.3);
 set(Serial1, 'DataTerminalReady', 'on'); pause(.3);
 set(Serial1, 'DataTerminalReady', 'off'); pause(.3);
 F=figure('Name','Osciloscópio','NumberTitle','off','units',...
    'normalized','outerposition',[0 0 1 1]);

 i1 = uicontrol('Style', 'pushbutton','ForegroundColor',...
    'red','FontWeight','bold','fontsize',14,'String','Fim','Position',...
    [30-desl/3 70 60 40],'Callback','sai=1','TooltipString','Fim do programa!');
 i2 = uicontrol('Style', 'pushbutton','ForegroundColor','black',...
    'String','Tempo','Position',[30-desl/3 20 60 40],'Callback','tela=1');
 i3 = uicontrol('Style', 'toggle','ForegroundColor','black',...
    'String', 'Pausa','Position',[100-desl/3 70 60 40],'Callback','congela=1');
 i4 = uicontrol('Style', 'pushbutton','ForegroundColor','black',...
    'String','Frequ.','Position',[100-desl/3 20 60 40],'Callback','tela=0');
 i5 = uicontrol('Style', 'pushbutton','ForegroundColor','blue',...
     'String', 'FbePB','Position',[600-desl 20 60 40],'Callback','tela=6');
 i6 = uicontrol('Style', 'pushbutton','ForegroundColor','blue',...
    'String', 'FChePB','Position',[250-desl 20 60 40],'Callback','tela=3');
 i7 = uicontrol('Style', 'pushbutton','ForegroundColor','blue',...
    'String', 'FChePA','Position',[320-desl 20 60 40],'Callback','tela=4');
 i8 = uicontrol('Style', 'pushbutton','ForegroundColor','blue',...
```

```
    'String', 'FpsPA','Position',[390-desl 20 60 40],'Callback','tela=5');
i9 = uicontrol('Style', 'pushbutton','ForegroundColor','blue',...
     'String', 'FpsPB','Position',[460-desl 20 60 40],'Callback','tela=2');
i10= uicontrol('Style', 'pushbutton','ForegroundColor','blue',...
    'String', 'FbeRB','Position',[530-desl 20 60 40],'Callback','tela=7');
i11= uicontrol('Style', 'pushbutton','ForegroundColor','magenta',...
    'String', 'S[+]','Position',[925-off 20 60 40],'Callback','tela=8');
i12= uicontrol('Style', 'pushbutton','ForegroundColor','magenta',...
    'String', 'S[-]','Position',[995-off  20 60 40],'Callback','tela=9');
i13= uicontrol('Style', 'pushbutton','ForegroundColor','black',...
    'String', 'Normal','Position',[1065-off  20 60 40],'Callback','tela=10');
i14= uicontrol('Style', 'pushbutton','ForegroundColor','magenta',...
    'String', 'Espelho','Position',[1135-off  20 60 40],'Callback','tela=11');
i15= uicontrol('Style', 'pushbutton','ForegroundColor','magenta',...
    'String', 'T[+]','Position',[1205-off  20 60 40],'Callback','inc=1;');
i16= uicontrol('Style', 'pushbutton','ForegroundColor','magenta',...
    'String', 'T[-]','Position',[1275-off  20 60 40],'Callback','inc=0;');
i17= uicontrol('style','slide','unit','pix','position',[40 140 20 200],...
    'min',.01,'max',.5,'val',.22);
i18= uicontrol('style','slide','unit','pix','position',[40 440 20 200],...
    'min',-1000,'max',1000,'val',0);
i19= uicontrol('Style', 'pushbutton','ForegroundColor',[0,0.5,0],...
    'String', 'WinPA','Position',[1440-off  140 60 40],'Callback','tela=12');
i20= uicontrol('Style', 'pushbutton','ForegroundColor',[0,0.5,0],...
    'String', 'WinPB','Position',[1440-off  220 60 40],'Callback','tela=13');
str1(1) = {'Frequência'}; str1(2) = {'de Corte:'};
i21= uicontrol('Style','text','Position',[12 360 80 33],'String',str1,...
    'FontSize',8,'fontweight','b');
str2(1) = {'Posição Y'}; str2(2) = {'do Traço:'};
i22= uicontrol('Style','text','Position',[10 660 80 33],'String',str2,...
    'FontSize',8,'fontweight','b'); pause(1);
set(0,'defaultaxesfontsize',12,'defaultaxeslinewidth',0.9,...
    'defaultlinelinewidth',1,'defaultpatchlinewidth',0.9);

try

while 1

    while (k==0)
          n1 = fread(Serial1,1); n2 = fread(Serial1,1); n3 = fread(Serial1,1);
          if (n1==126 && n2==126 && n3==126 && k==0)  % procura 7E-7E-7E
             k=1;
          end
    end

    n = fread(Serial1,2047);

    if numel(n)~=2047
       fclose(Serial1);delete(Serial1);clear Serial1
       delete(instrfindall);
        last=scanports();
        eval('Serial1 = serial(cell2mat(last));');
```

```
102         fopen(Serial1);
103         set(Serial1, 'DataTerminalReady', 'off'); pause(.3);
104         set(Serial1, 'DataTerminalReady', 'on'); pause(.3);
105         set(Serial1, 'DataTerminalReady', 'off'); pause(.3);
106         disp('Reinicializando...')
107         pause(1);
108         n = fread(Serial1,2047);
109     end
110
111     while fora==0 && numel(n)== 2047
112         if n(x)>(s-2) && n(x)<(s+2) && n(x)<n(x+2) && n(x+2)<n(x+3) && fora==0
113             a=x;
114             fora=1;
115             break
116         end
117     x=x+1;
118         if x>1246                       % 1246 = 2046 - 800 pixels
119             x=1;
120             fora=1;
121             break
122         end
123     end
124
125 if fora==0
126     s=s+10;
127     if s>250
128         s=10;
129     end
130 end
131
132       xx=1:EscX;
133       B=n(a+xx);
134       Fs = 40000000;                % Frequência de amostragem 40MHz
135       T = 1/Fs;                     % Periodo da amostragem
136       L = EscX;                     % Comprimento do sinal
137       t = (0:L-1)*T;
138       SLI = get(i17,'value');
139       POS = get(i18,'value');
140       Vpp = (max(B)-min(B))*(2/255);
141       Vdc = s*(2/255);
142
143       switch (tela)
144       case {0}                      % Frequência
145           spectro=1;
146       case {1}                      % Tempo
147           spectro=0;
148       case {2}                      % Pólo simples passabaixa
149           B=fpspb(SLI, B);
150       case {3}                      % Chebyschev passabaixa
151           [Bf,Af] = cheby1(2,2,SLI*2,'low');
152           B=(filter(Bf,Af,B));
153       case {4}                      % Chebyschev passa-alta
```

```
154         [Bf,Af] = cheby1(2,2,SLI*2,'high');
155         B=(filter(Bf,Af,B));
156     case {5}                    % Pólo simples passa-alta
157         B=fpspa(SLI, B);
158     case {6}                    % Banda estreita passabanda
159         B=fbepb(SLI, 0.01, B);
160     case {7}                    % Banda estreita rejeitabanda
161         B=fberb(SLI, 0.45, B);
162     case {8}                    % s[+]
163         s=s+10;
164         if s>max(B)
165            s=(max(B)-min(B))/2;
166         end
167         tela=14;
168     case {9}                    % s[-]
169         s=s-10;
170         if s<min(B)
171            s=(max(B)-min(B))/2;
172         end
173         tela=14;
174     case {10}                    % Normal
175         POS=0;
176         spectro=2;
177     case {11}                    % Espelho
178         spectro=3;
179     case {12}                    % winsinc passa-alta
180         [xf,Rh,Fh] = winsinc(B,.0401,SLI,80,'hamming','high');
181         B=xf;
182     case {13}                    % winsinc passabaixa
183         [xf,Rh,Fh] = winsinc(B,.0401,SLI,80,'hamming','low');
184         B=xf;
185     end
186
187     switch (spectro)
188     case {0}                    % Tempo
189
190         if cong==0
191         cl=1;yl=n(a+1);plot(1,1);       % Cls
192             for xx = 1:EscX
193                 if abs(n(a+xx)-n(a+xx+1))<8 && abs(n(a+xx+1)-n(a+xx+2))<8
194                   if ponto==0
195                     line([xx cl],[n(a+xx) yl]+POS);
196                     %line([xx cl],[n(a+xx) yl]+POS,'LineWidth',4);
197                   else
198                     rectangle('Position',[xx,n(a+xx)+POS,.7,.7],'EdgeColor','b')
199                   end
200                   cl=xx;yl=n(a+xx);
201                 end
202             end
203         end
204         h = xlabel('Tempo (x25 ns)');
205         pos = get(h,'pos');
```

```
206         set(h,'pos',[EscX -10 0])
207         axis([1  EscX  0  255])
208         ylabel('Tensão (v)')
209         set(gca,'YTickLabel','0|0.39|0.78|1.18|1.57|1.96')
210         text(EscX-EscX/10,10,['Vdc=',num2str(Vdc,2),' volts'])
211         text(EscX-EscX/10,20,['Vpp=',num2str(Vpp,2),' volts'])
212         text(EscX-EscX/10,30,['Fc=',num2str(SLI,2),' rad/s'])
213         [D]=peakfinder(B);
214             if numel(D)>1
215                 fre=1/((D(2)-D(1))*25e-9);
216                 text(EscX-EscX/10,40,['F=',num2str(fre,7),' Hz'])
217             end
218     case {1}                        % Frequência
219         NFFT = 2^nextpow2(L);
220         Y = fft(B,NFFT)/L;
221         f = Fs/2*linspace(0,1,NFFT/2+1);
222         if cong==0
223            if ponto==0
224             plot(f,2*abs(Y(1:NFFT/2+1)))
225            else
226             stem(f,2*abs(Y(1:NFFT/2+1)),'fill','MarkerFaceColor','red')
227            end
228         end
229         h = xlabel('Frequência ( Hz )');
230         pos = get(h,'pos');
231         set(h,'pos',[ EscX*10000 -10 0])
232         axis([1  EscX*10000  0  255/2])
233         ylabel('Amplitude')
234     case {2}                        % Normal
235        if cong==0
236         plot (B+POS)
237         ponto=0;
238         h = xlabel('Tempo (x25 ns)');
239         pos = get(h,'pos');
240         set(h,'pos',[EscX -10 0])
241         axis([1  EscX  0  255])
242         ylabel('Tensão (v)')
243         set(gca,'YTickLabel','0|0.39|0.78|1.18|1.57|1.96')
244         text(EscX-EscX/10,10,['Vdc=',num2str(Vdc,2),' volts'])
245         text(EscX-EscX/10,20,['Vpp=',num2str(Vpp,2),' volts'])
246         text(EscX-EscX/10,30,['Fc=',num2str(SLI,2),' rad/s'])
247         [D]=peakfinder(B);
248             if numel(D)>1
249                 fre=1/((D(2)-D(1))*25e-9);
250                 text(EscX-EscX/10,40,['F=',num2str(fre,7),' Hz'])
251             end
252        end
253     case {3}                        % Espelho
254        if cong==0
255         plot(fftshift(abs(fft(B))))
256        end
257         EscX=400;
```

```
258           axis([0  400  1  30000])
259       end
260
261       grid on
262       title('TCC UTFPR 2013 - Osciloscópio 4 MHz/40MSPS','fontsize',24,...
263           'FontWeight','bold', 'Color', 'red')
264       drawnow;
265
266        if sai==1                       % Fim !
267             fclose(Serial1);delete(Serial1);clear Serial1;
268             delete(instrfindall);clc,clear,close all;
269             disp('Fim do programa!')
270             return
271        end
272        if congela==1                    % Pausa
273             cong=~cong; EscX=800;
274             ponto=1; congela=0;
275        end
276        if inc==1                       % T[+]
277               if (EscX>1 && EscX<800)
278                  EscX=EscX+10;
279               else
280                  EscX=400;
281               end
282               inc=3;
283        end
284        if inc==0                       % T[-]
285               if (EscX>10 && EscX<800)
286                  EscX=EscX-10;
287               else
288                  EscX=400;
289               end
290               inc=3;
291        end
292        a=0; fora=0; k=0; x=1;
293        if Serial1.BytesAvailable >0
294           fread(Serial1, Serial1.BytesAvailable);
295        end
296
297 end
298
299 catch
300
301          close all;                    % Erro!
302          set(Serial1, 'DataTerminalReady', 'on'); pause(.1);
303          set(Serial1, 'DataTerminalReady', 'off'); pause(.1);
304          set(Serial1, 'DataTerminalReady', 'on'); pause(.1);
305          fclose(Serial1);delete(Serial1);clear Serial1;delete(instrfindall);
306          disp('Ocorreu um erro...')
307          %inicio do debug
308          %rethrow(lasterror)
309          %return
```

```
310         %fim do debug
311         pause(1);
312         run Osc4MHzV4
313 end
```

APÊNDICE E – Programa em Basic (SANTOS, 1992)

```
' Fonte do programa OSC3.BAS em Turbo Basic 1.1
'
' Autor: Marcelo Machado Santos  Data: 02/07/91
'
' Engenharia Industrial Eletrica - CEFET - PR
'
defint a-z:dim a%(2048)
gosub apresentacao
cong=0:sinc=1:k=2:sc=32
inicio:
cls:screen 2
gosub moldura
out &H303,&B10001011                    'PA=OUT;PB=INP;PC=INP
out &H300,&B100                         'ATIVUS:INT CLK;W:EN
delay .5                                'Atraso p/ encher buffer
out &H303,&B10000010                    'PA=OUT;PB=INP;PC=OUT
out &H300,&B011                         'Inverte niveis em PA
for n=0 to 2047
a(n)=inp(&H301)                         'Le dado de PB
out &H302,0:out &H302,1                 'Gera EXT CLK
next n
if sinc=0 then semsinc
for n=0 to 2047
if a(n)>sc-2 and a(n)<sc+2 then teste
denovo:
next n
sc=sc+1:if sc=60 then sc=0
goto inicio
sai:
pset (0,100)
for x=n to n+639
if k=2 then pset (x-n,199-(a(x)*3))
if k=1 and a(x)>0 and a(x)<63 then line- (x-n,199-(a(x)*3))
next x
leteclado:
k$=inkey$
if k$="l" or k$="L" then k=1
if k$="p" or k$="P" then k=2
if k$="s" or k$="S" then end
if k$="c" or k$="C" then cong=1
if k$="d" or k$="D" then sinc=0
if k$="h" or k$="H" then sinc=1
if k$="r" or k$="R" then cong=0
if k$="m" or k$="M" then gosub apresentacao:goto inicio
if k$="g" or k$="G" then gosub salvatela
if k$="v" or k$="V" then gosub recuperatela
if cong=1 then leteclado
cls
goto inicio
' Portas usadas no CI 8255:
' PA=&H300; PB=&H301; PC=&H302; CONTROLE=&H303
apresentacao:
cls:screen 0
?"===============  Osciloscopio Digital - MMS HARD/SOFT - 6 BIT'S  ==============
?:?:?:?"Teclas de Controle :":?:?
?"[L]inha (tela c/ pontos unidos)"
?"[P]onto (tela c/ pontos nao unidos)"
?"[S]ai para o sistema operacional"
?"[C]ongela a varredura da tela"
?"[R]etorna a varredura da tela"
?"[D]esabilita sicronismo"
?"[H]abilita sincronismo"
?"[G]rava tela atual no disco"
?"[V]isualiza arquivo de tela"
?:?"[M]enu":?
input "Quer definir o nome do arquivo de tela (S/N)";r$
if r$="s" or r$="S" then cls:files:?:input "Qual o nome";tela$ else tela$="TELA"
e=&B1111100001111111:+=&B1111011110111101
return
moldura:
line(0,0)-(639,199),,b:line(319,0)-(319,200),,,f:line(639,100)-(0,100),,,e
return
teste:
if a(n)<a(n+1) and a(n+1)<a(n+2) then sai else denovo
semsinc:
pset (0,100)
for x=0 to 639
if k=2 then pset (x,199-(a(x)*3))
if k=1 then line- (x,199-(a(x)*3))
next x
goto leteclado
salvatela:
def seg=&HB800
bsave TELA$,0,&H3FFF
return
recuperatela:
def seg=&HB800
cls:screen 2
bload TELA$
ciclo:
if inkey$="" then ciclo
return
```

APÊNDICE F – Código do filtro fpspb

```
function[h] = fpspb(fc, x)

  tx = length(x);
  pi = 3.14159265;
  r = exp ( (-2 * pi * fc) );
  a0 = 1 - r;
  b1 = r;

  h(1) = a0 * x(1);

  for i = 2:tx
    h(i) = (a0 * x(i)) + (b1 * h(i - 1));
  end
  for i=7:tx
      h(i-6) =  h(i);
  end
  end
```

APÊNDICE G - Código do filtro fpspa

```
function[j] = fpspa(fc, x)

  tx = length(x);
  pi = 3.14159265;
  r = exp( (-2 * pi * fc) );
  a0 = (1 + r) / 2;
  a1 = -(1 + r) / 2;
  b1 = r;
  j(1) = a0 * x(1);

  for i = 2:tx
    j(i) = (a0 * x(i)) + (a1 * x(i -1)) + (b1 * j(i - 1));
  end

  end
```

APÊNDICE H - Código do filtro fbepb

```
function[h] = fbepb(fc, bw, x)

  tx = length(x);
  pi = 3.14159265;
  r = 1 - 3 * bw;
  k = 1 - (2 * r * cos(2 * pi * fc) + r^2) / (2 - 2 * cos(2 * pi * fc));
  a0 = 1 - k;
  a1 = 2 * (k - r) * cos(2 * pi * fc);
  a2 = r^2 - k;
  b1 = 2 * r * cos(2 * pi * fc);
  b2 = -r^2;
  h(1) = a0 * x(1);
  h(2) = a0 * x(2) + a1 * x(1) + b1 * h(1);
  h(3) = a0 * x(3) + a1 * x(2) + a2 * x(1) + b1 * h(2) + b2 * h(1);

  for i = 3:tx
    h(i) = a0 * x(i) + a1 * x(i - 1) + a2 * x(i - 2) + b1 * h(i - 1) + b2 * h(i - 2);
  end
end
```

APÊNDICE I - Código do filtro fberb

```
function[h] = fberb(fc, bw, x)

  tx = length(x);
  pi = 3.14159265;
  r = 1 - 3 * bw;
  k = (1 - (2 * r * cos(2 * pi * fc)) + r^2) / (2 - 2 * cos(2 * pi * fc));
  a0 = k;
  a1 = -2 * k * cos(2 * pi * fc);
  a2 = k;
  b1 = 2 * r * cos(2 * pi * fc);
  b2 = -r^2;

  h(1) = a0 * x(1);
  h(2) = a0 * x(2) + a1 * x(1) + b1 * h(1);
  h(3) = a0 * x(3) + a1 * x(2) + a2 * x(1) + b1 * h(2) + b2 * h(1);

  for i = 3:tx
    h(i) = a0 * x(i) + a1 * x(i - 1) + a2 * x(i - 2) + b1 * h(i - 1) + b2 * h(i - 2);
  end

end
```

APÊNDICE J – Lista de Materiais

Parte	Valor	Nomenclatura
C1	100n	Capacitor de Cerâmica 50v
C2	22p	Capacitor de Cerâmica 50v
C3	22p	Capacitor de Cerâmica 50v
C4	100n	Capacitor de Cerâmica 50v
C7	100n	Capacitor de Cerâmica 50v
C8	100n	Capacitor de Cerâmica 50v
C9	100n	Capacitor de Cerâmica 50v
C10	100n	Capacitor de Cerâmica 50v
C11	100n	Capacitor de Cerâmica 50v
C12	100n	Capacitor de Cerâmica 50v
C13	100u	Capacitor eletrolítico 16v
C15	100n	Capacitor de Cerâmica 50v
C16	100n	Capacitor de Cerâmica 50v
C17	100p	Capacitor de Cerâmica 50v
C18	1000uF	Capacitor eletrolítico 16v
D1	1N4004	Diodo retificador 50v/1A
IC0	744040N	Circuito integrado DIL16
IC1	74LS590N	Circuito integrado DIL16
IC2	74LS590N	Circuito integrado DIL16
IC3	7805TV	Circuito integrado 5v/1A
IC4	UM61512	Memória SRAM 64Kx8bit DIL32
IC5	74HC157N	Circuito integrado DIL16
IC6	MAX232	Circuito integrado DIL16 (opcional)
IC7	OPA2604	Circuito integrado DIL08
IC8	ATMEGA8	Circuito integrado do kit arduino
IC9	TLC5540SOP	Circuito integrado SOP24
IC10	FT232RL	Circuito integrado do kit arduino
IC12	74ALS00D	Circuito integrado DIL14
J4	---	Conector PINHD-1X8
J5	---	Conector PINHD-1X8
J6	---	Conector PINHD-1X8
L1	100uH	Indutor de 100 micro Henrys
Q1	16MHz	Cristal 16MHz do kit arduino UNO
Q2	40MHz	Oscilador a Cristal 40MHz/5v
R1	10k	Resistor de carbono do kit arduino UNO
R2	100k	Potenciômetro linear de carbono
R3	100k	Potenciômetro linear de carbono
R4	470R	Resistor de carbono 1/8W 5%
R5	10k	Resistor de carbono 1/8W 5%
R6	470k	Resistor de carbono 1/8W 5%
R7	12K	Resistor de carbono 1/8W 5%
R8	47R	Resistor de carbono 1/8W 5%
R9	220k	Resistor de carbono 1/8W 5%
R10	10k	Resistor de carbono 1/8W 5%
R11	10k	Resistor de carbono 1/8W 5%
R12	10k	Resistor de carbono 1/8W 5%
R13	100R	Resistor de carbono do kit arduino UNO
S1	---	Chave rotativa 2 pólos por 5 posições

X1	---	Conector fêmea USB do kit arduino UNO
X2	---	Conector fêmea AMP do kit arduino UNO
X3	---	Conector fêmea RCA coaxial
X4	---	Conector fêmea RCA coaxial
Y1	---	Placa de circuito impresso padrão 5x10cm
Y2	---	Placa de circuito impresso adaptadora SOP20/DIP