UNIVERSIDADE FEDERAL DO PARANÁ
ANDERSON MOREIRA SILVA
DANIEL DE ANDRADE USSUNA

USBScope 2.10

Curitiba-PR
2011

UNIVERSIDADE FEDERAL DO PARANÁ
DANIEL DE ANDRADE USSUNA
ANDERSON MOREIRA SILVA

USBScope 2.10

Projeto de Conclusão de Curso apresentado ao curso de Engenharia Elétrica da Universidade Federal do Paraná como requisito parcial para a obtenção de título de Engenheiro Eletricista.

Orientador: Márlio José do Couto Bonfim.

Curitiba-PR
2011

AGRADECIMENTOS

Agradecemos ao Professor Márlio José do Couto Bonfim pela oportunidade e ajuda prestada no decorrer deste trabalho técnico que tanto acrescentou em nossos conhecimentos acadêmicos e profissionais. Agradecemos ainda pelo fornecimento e confiança da utilização do espaço e equipamentos do Laboratório de Microeletrônica Medidas e Instrumentação (LAMMI) do departamento de Engenharia Elétrica da Universidade Federal do Paraná.

Gostaríamos de agradecer aos colegas de pesquisa, tanto os de iniciação científica, quanto os de mestrado e aos colegas de turma que não pouparam esforços em fornecer ajuda quando solicitados.

Agradecemos a família por todo o apoio fornecido nos momentos de esforço e dificuldades.

RESUMO

Este trabalho apresenta o desenvolvimento de um osciloscópio digital portátil e de baixo custo. Composto de uma placa de aquisição de dados que se comunica através da porta USB com um *software* multiplataforma instalado no microcomputador. Este *software* é responsável por apresentar graficamente no monitor o sinal analisado, além de executar funções matemáticas diversas, capazes de fornecer informações importantes como magnitude de tensão, valores médios e rms, freqüência e FFT. A maior parte dos equipamentos existentes suporta a aplicação no sistema operacional Windows. O USBScope 2.10 apresenta os principais resultados aplicados no sistema operacional Linux UBUNTU 11.10.

ABSTRACT

This work presents the development of a low cost portable digital oscilloscope. Composed of a data acquisition board that communicates through a USB port with a software installed on the PC platform. This software is responsible for presenting graphically on the monitor the analized signal, and perform several mathematical functions that can provide important information such as voltage magnitude, average and rms values, frequency and FFT. Most of the existing equipments supports the application only in the Windows operating system. The USBScope 2.10 presents the main results applied to the Linux operating system Ubuntu 11.10.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE FLUXOGRAMAS

LISTA DE TABELAS

1. INTRODUÇÃO

1.1. CONTEXTUALIZAÇÃO

O osciloscópio é um equipamento eletrônico capaz de representar graficamente a variação de tensão em função do tempo. O que torna a utilização deste equipamento fundamental é a capacidade que possui de apresentar não só a medida quantitativa, mas também qualitativa, do sinal analisado.

Osciloscópios digitais, além da representação de sinais, permitem o armazenamento dos dados de forma digitalizada, para que funções matemáticas complexas sejam aplicadas a estes fornecendo informações importantes e de grande utilidade no estudo, pesquisa e manutenção de circuitos elétricos e eletrônicos.

O projeto USBScope 2.10 apresenta-se como a continuidade do projeto de graduação realizado por alunos do curso de Engenharia Elétrica da Universidade Federal do Paraná no ano de 2007 [1].

Nesta primeira versão (USBScope 1.10) o principal objetivo foi desenvolver um osciloscópio digital de dimensões reduzidas, ou seja, uma placa que aquisição de dados, que se comunica com um microcomputador através da porta de comunicação USB (*Universal Serial Bus*). O microcomputador por sua vez possui um *software*, Figura 1, de alto nível que é responsável por receber os dados digitais, tratá-los matematicamente e representá-los no monitor.

Este *software* possui botões que representam as características físicas de controle de um osciloscópio digital convencional. Este osciloscópio, desenvolvido para sistema operacional Windows, possui dois canais, taxa de conversão de dados de 60 KSPS, resolução de 10 bits, oito ajustes de escala de amplitude e 50 V para amplitude máxima do sinal analisado.

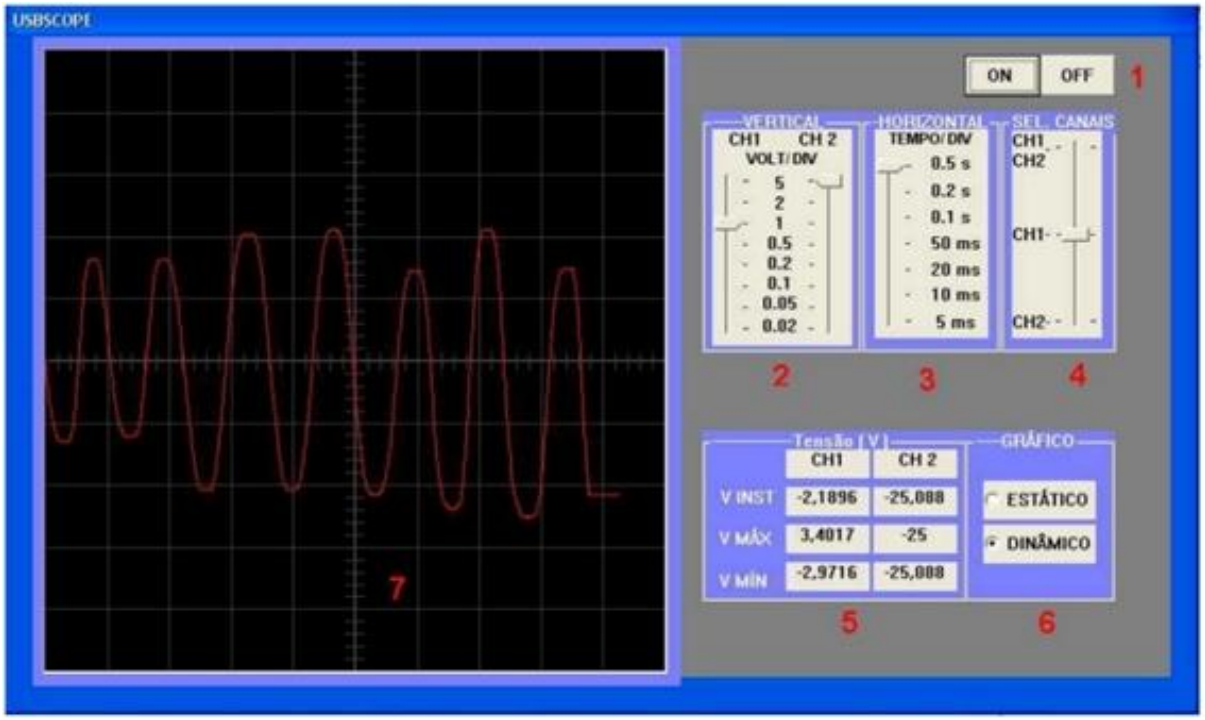


Figura 1: Interface gráfica USBSCope 1.1

Na Figura 1 é mostrada a tela do USBScope 1.1, os números em vermelho representam as seguintes funções:

1) Botão *ON* inicia a visualização gráfica e o botão *OFF* encerra o aplicativo gráfico.

2) Seletor de escala vertical para os canais 1 e 2, com uma faixa de 0,02 V a 0,5 V por divisão.

3) Seleção de escala horizontal em segundos por divisão.

4) Seletor de canais com três opções, CH1 – CH2 para visualização simultânea dos canais 1 e 2, CH1 para visualizar somente o canal 1 e CH2 para visualizar somente canal 2.

5) Indicador dos valores de tensão instantânea, tensão máxima e tensão mínima do quadro atual de plotagem, para os canais 1 e 2.

6) Opção de escolha da visualização gráfica, no modo dinâmico a tela será constantemente atualizada e no modo estático o gráfico fica congelado na plotagem atual da tela.

7) Tela de visualização gráfica dos canais 1 e 2, com dez divisões no eixo “X” e no eixo “Y”.

1.2. MOTIVAÇÃO

Devido o elevado custo dos osciloscópios de bancada, acima de R$ 1.500,00 em média, a utilização desta ferramenta por técnicos e estudantes de engenharia elétrica fica por muitas vezes limitada em laboratórios. Outra característica que dificulta a utilização destes equipamentos em campo por profissionais é a elevada dimensão que possuem, o que interfere em seu transporte.

Pensando em atender cursos técnicos e estudantes de engenharia que disponibilizam de pouco recurso financeiro e profissionais da área que necessitam desta ferramenta em dimensões reduzidas, portátil, foi elaborado o projeto USBScope.

1.3. OBJETIVOS

Este trabalho apresenta o desenvolvimento de um osciloscópio digital portátil, de baixo custo e fácil utilização, possuindo interface de usuário amigável e multiplataforma.

As principais características que protótipo desenvolvido deverá possuir são:

- Dimensões aproximadas de um *Pendrive;*
- Dois canais de entrada analógica;
- Baixo Custo;
- Comunicação USB;
- Taxa de Amostragem máxima de 1 MSPS;
- 16 escalas de tempo;
- Oito escalas de amplitude;
- Máxima amplitude do sinal de entrada de 100 V;
- *Software* robusto com interface amigável;

- Disponibilizar funções que ajudem o usuário a interpretar a onda de forma completa;
- Botões de ajuste das escalas de tempo e tensão;
- Funções de *trigger*;
- Operações matemáticas sobre os canais (soma, subtração, mutiplicação, divisão);
- FFT (Transformada Rápida de *Fourier*);
- Visualização de valores do sinal (valor médio, RMS, de pico, pico-a-pico);
- Função para salvar os pontos e imagem da forma de onda;

2. REVISÃO BIBLIOGRÁFICA

2.1. TAXA DE NYQUIST

Segundo o Teorema de *Nyquist* [1], para que seja possível a recuperação de um sinal analógico através de suas amostras, deve-se utilizar de uma taxa de amostragem de pelo menos duas vezes superior a máxima frequência do sinal analisado.

$$f_s = 2f_m \quad (1)$$

2.2. INTERPOLAÇÃO LINEAR

O *software* do USBScope 2.10 utiliza interpolação linear [2] para a ligação entre as amostras, o que torna insatisfatório o uso da taxa de *Nyquist* para a realização da amostragem, pois pode causar um efeito de triangulação no sinal visto pelo usuário. Para resolver este problema e melhorar a qualidade do sinal apresentado, cada período do sinal é representado com 4 amostras.

Osciloscópios digitais fazem uso desta teoria aplicada aos conversores analógicos digitais (A/D), pois estes componentes são os responsáveis pela amostragem do sinal.

2.3. ESTADO DA ARTE

Atualmente o mercado dispõe de vários tipos e modelos de osciloscópios, entre os portáteis e os de bancada. Osciloscópios de bancada possuem largura de banda que chegam à ordem de giga-hertz, a faixa de tensão pode atingir a ordem de centenas até milhares de volts. Os osciloscópios portáteis apresentam vantagem pela dimensão, peso e preço reduzidos. Porém são destinados à medições de tensão de menor amplitude que os osciloscópios de bancada, com escala de centenas de volts e largura de banda consideravelmente menor, da ordem das centenas de kilo-hertz, em alguns casos até a ordem de mega-hertz.

A seguir estão apresentados alguns osciloscópios portáteis encontrados no mercado.

O PC XY USB Oscilloscope Disco [3], apresentado na Figura 2, é um osciloscópio digital portátil desenvolvido pela empresa *HobbyLab*. Sua aplicação se da através do sistema operacional Windows, possui dois canais, largura de banda de 100KHz por canal, resolução de 10bits, sete ajustes de escala de amplitude, sete ajuste de escala de tempo, 40Volts de máxima amplitude de sinal analisado e função de *trigger*. O custo previsto para este equipamento é de R$220,00.

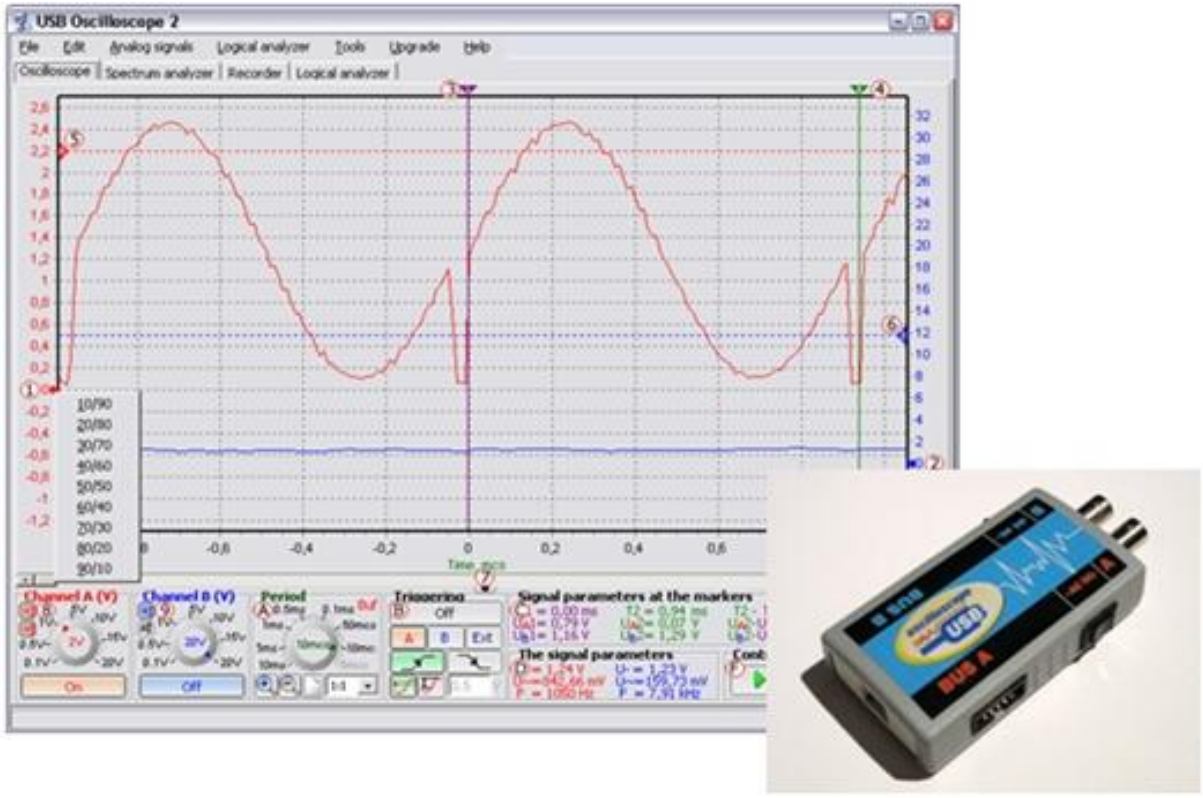


Figura 2: Osciloscópio PC XY USB Disco

Parallax USB Oscilloscope [4], apresentado na Figura 3, é outro equipamento semelhante desenvolvido pela empresa Parallax. Opera em plataforma Windows, possui dois canais, largura de banda de 200 KHz, resolução de 8bits, sete ajustes de escala de amplitude, quatorze ajuste de escala de tempo, 40Volts de máxima amplitude de sinal analisado e função de *trigger*. O custo previsto para este equipamento é de R$255,00.

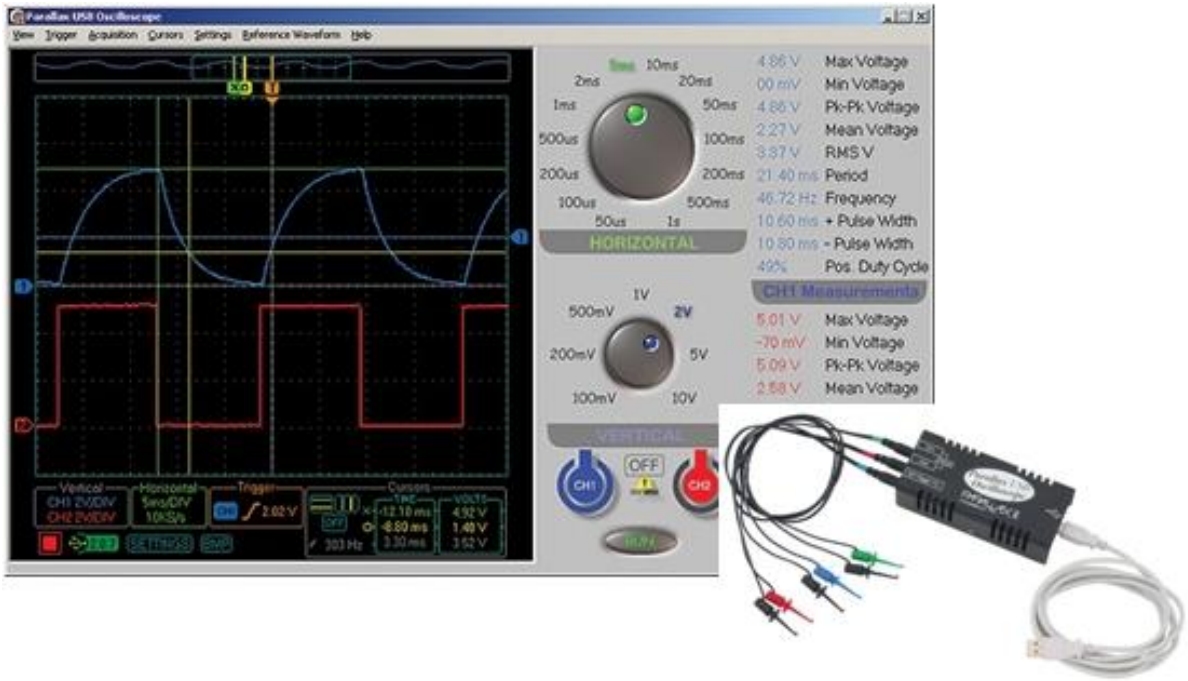


Figura 3: Osciloscópio Parallax USB

O osciloscópio PoScope [5], apresentado na Figura 4 possui a mesma interface gráfica do XY USB Oscilloscope Disco. Desenvolvido pela PoLabs, opera em plataforma Windows. O sistema possui dois canais, largura de banda de 200 KHz, resolução de 10 bits, sete ajustes de escala de amplitude, sete ajuste de escala de tempo, 40 volts de máxima amplitude de sinal analisado e função de *trigger*. O custo previsto para este equipamento é de R$320,00.

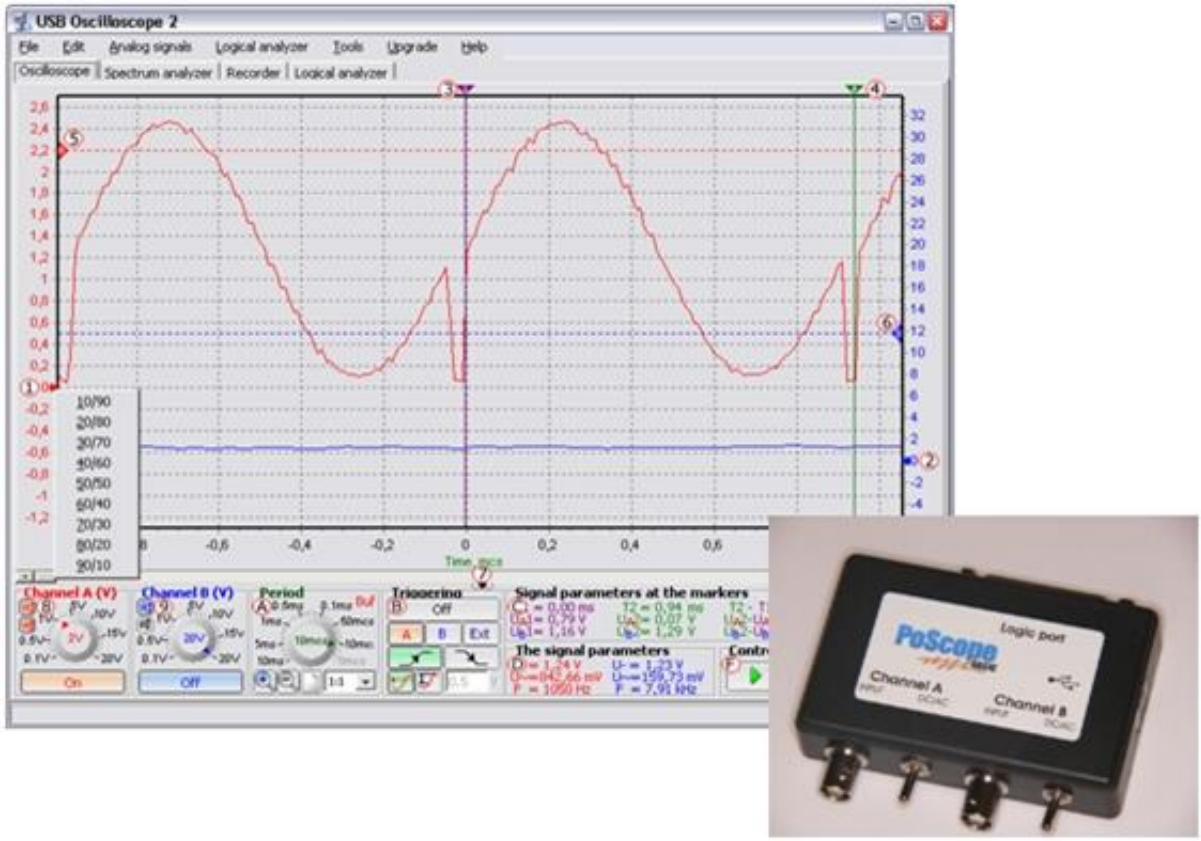


Figura 4: Osciloscópio PoScope

PS40M10 [6], Figura 5, não é apenas um osciloscópio digital, possui também as funções de voltímetro, medidor de freqüência e analisador de espectro. Produzido pela EasySYNC e desenvolvido para operar em plataforma Windows, possui apenas um canal, largura de banda de 5 MHz, resolução de 10 bits e 50 volts de máxima tensão de entrada. O custo previsto para este equipamento é de R$340,00.

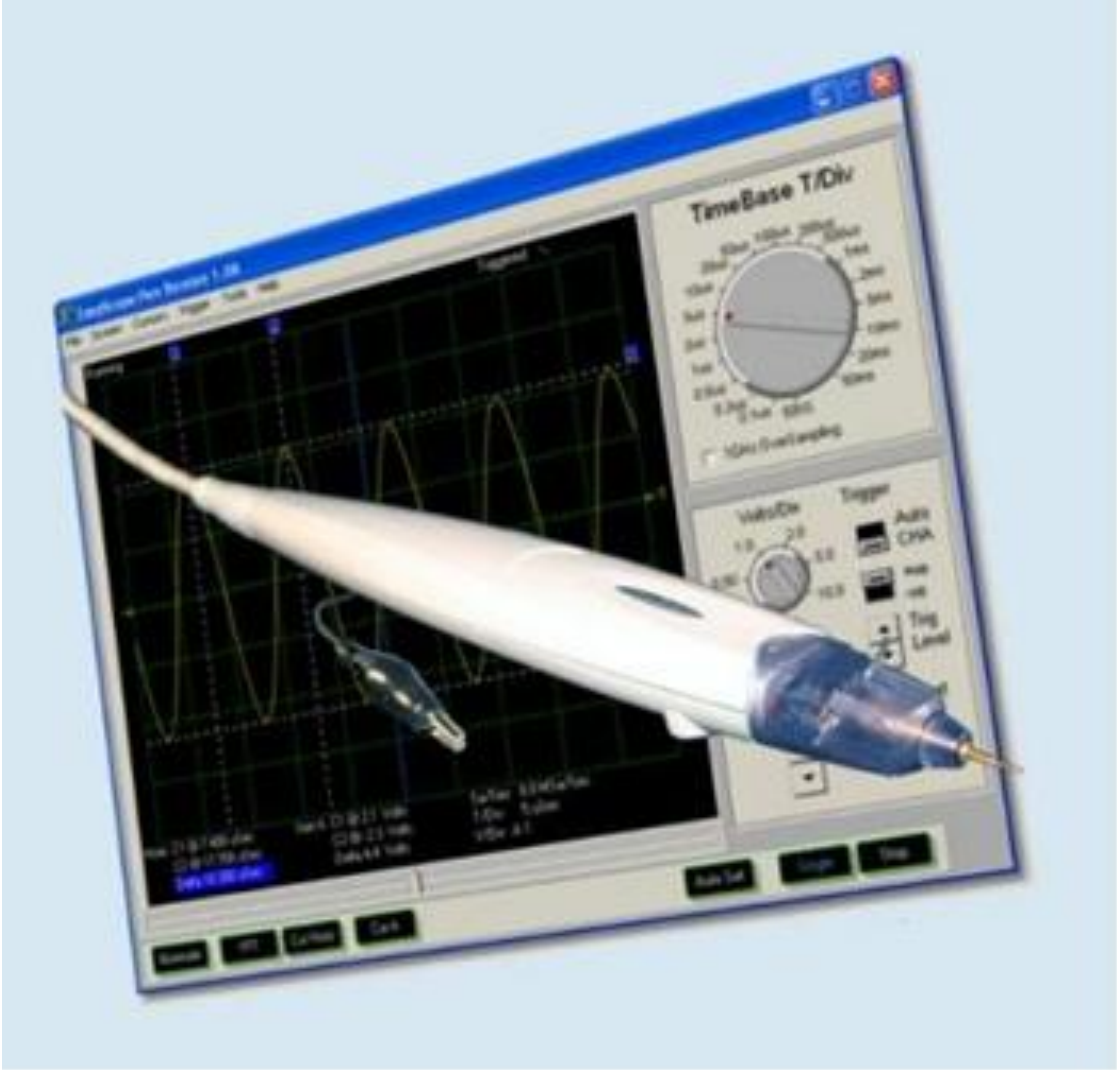


Figura 5: Osciloscópio PS40M10

A Tabela 1 apresenta a comparação entre as principais características dos osciloscópios digitais portáteis comercializados e os objetivos pretendidos para o USBScope 2.10.

Osciloscópio	B (Hz)	Resolução (bits)	Vin (volts)	S.O.	Custo (R$)
PC XY	100 K	10	40	Windows	220,00
Parallax	200 K	8	40	Windows	255,00
PoScope	200 K	10	40	Windows	320,00
PSM4010	5 M	10	50	Windows	340,00
USBScope 2.10	400 K	10	100	Multi Plataforma	< 300,00

Tabela 1: Comparação entre os osciloscópios digitais portáteis

3. DESENVOLVIMENTO

Através de pesquisa efetuada no decorrer do trabalho USBScope 1.1, uma nova concepção do projeto foi estabelecida, ao que chamamos de USBScope 2.10. Esta concepção conta com a utilização de uma memória externa operando como um *buffer*. Enquanto na versão 1.10 cada dado convertido era transmitido para o microcomputador, agora há o armazenamento de uma quantidade mais elevada de dados convertidos, que na sequência são transportados em pacotes para o microcomputador. Esta mudança afeta diretamente a taxa de amostragem do sinal, pois não há o atraso causado pelo transporte do dado para a USB entre uma amostra e a seguinte.

Novos componentes eletrônicos foram utilizados para atingir melhorias na taxa de amostragem, sendo que o primeiro ponto analisado para a utilização dos componentes foi justamente de preservar a característica portátil do projeto. Esta característica foi mantida utilizando componentes de montagem em superfície (SMD) e de comunicação serial que possuem poucos pinos.

Componentes que possuem comunicação paralela apresentam taxas de comunicação mais elevadas, porém possuem quantidade de pinos mais elevada e necessitam de maior área para sua aplicação em placas de circuito impresso (PCI).

O microcontrolador utilizado, PIC18F2550, dispõe do protocolo SPI (*Serial Peripheral Interface*) implementado em seu sistema. Este protocolo opera através de três pinos e para que aconteça a correta comunicação do dispositivo mestre, o microcontrolador, com o escravo, deve-se interligar os pinos de cada dispositivo como mostrado na Figura 6. Neste protocolo todas as atividades de entrada e saída de dados estão sincronizadas pelo *clock* serial.

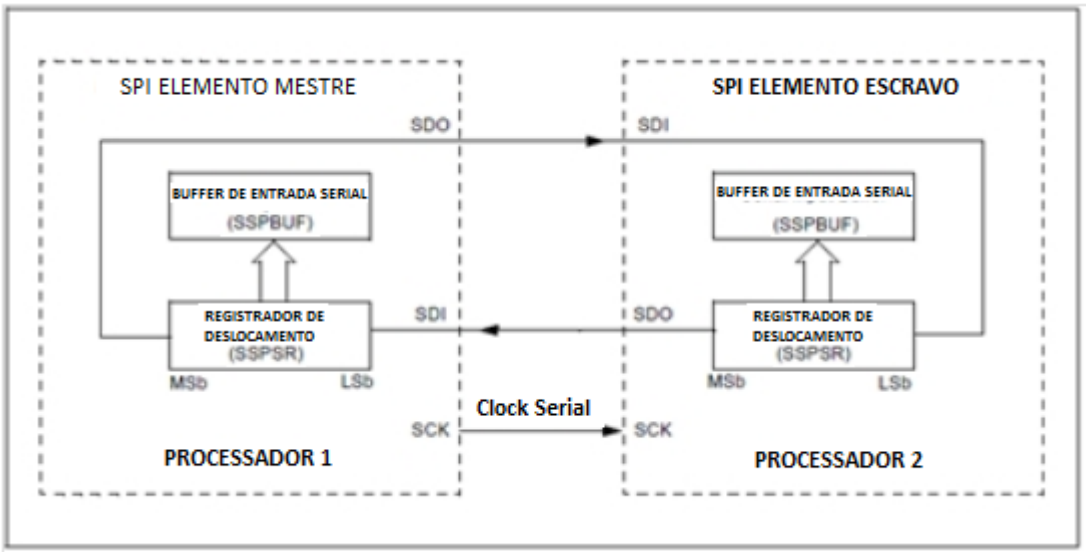


Figura 6: Diagrama de conexão SPI

O software passa a ser escrito de uma forma mais organizada, utilizando o paradigma de orientação à objeto. Mais funções foram acrescentadas, como o uso do *trigger*, funções matemáticas sobre os pontos dos canais, porém a mudança mais significativa é a possibilidade de visualização da transformada discreta de *Fourier* (DFT).

3.1. HISTÓRICO

No decorrer do desenvolvimento do USBScope 2.0, problemas de sincronismo levaram a mudanças no *hardware* que influenciam diretamente na comunicação USB e criam possibilidade de mudança da linguagem de programação do *software* para aplicação nos diferentes sistemas operacionais. Com estas mudanças o projeto passou a ser intitulado USBScope 2.10.

3.1.1. HARDWARE

O diagrama da versão 2.0 é mostrado na Figura 7. Nesta versão o microcontrolador é responsável por controlar o recebimento e envio de dados pela USB, as aquisições e transmissões de dados da memória externa, selecionar o canal convertido e a quantidade de aquisições do A/D e ajustar o ganho dos amplificadores operacionais.

A linha que liga diretamente os amplificadores operacionais e o microcontrolador representa a conexão física destes componentes, para que com a utilização de comparadores internos do microcontrolador seja possível implementar a função de t*rigger*. O sinal dos dois canais é representado pela função X(t), e sua conexão se dará através de um cabo com ponta de prova.

O microcomputador, através do *software,* analisará os dados digitais recebidos e representará a forma de onda no monitor.

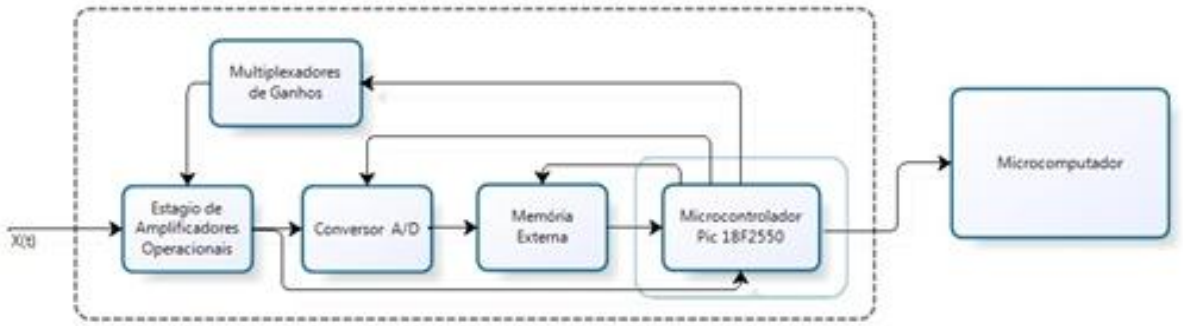


Figura 7: Diagrama em bloco do Projeto USBScope 2.0

3.1.1.1. MICROCONTROLADOR PIC18F2550

Também utilizado na versão 1.1, decidiu-se por este microcontrolador, pois além da comunicação SPI ele possui implementado em seu *hardware* o protocolo USB compatível com as versões 1.1 (Low Speed - *1,5Mbps*) e 2.0 (*High Speed* - 12Mbps). Outro aspecto que contribuiu pela sua escolha foi o de existir muitos exemplos com este dispositivo disponíveis na internet.

O PIC18F2550 possui 32 Kbytes de memória de programa, 2 Kbytes de memória de dados, comunicações seriais USB, UART, SPI e I2C, quatro *timers, clock* máximo de 48MHz e 24 pinos de I/O (entrada e saída).

Este microcontrolador possui 10 entradas para o módulo conversor analógico digital com resolução de 10 bits e máxima taxa de amostragem de 80 KSPS. Como um dos objetivos a ser atingido é de utilizar taxa de amostragem de 1 MSPS, ao invés da utilização do conversor AD interno do microcontrolador, será utilizado um conversor AD externo.

3.1.1.2. CONVERSOR ANALÓGICO DIGITAL

A escolha do conversor analógico digital, AD7912, utilizado na versão 1.1 do projeto foi mantida, pois este conversor atende as necessidades para o desenvolvimento de um equipamento com as características dos osciloscópios portáteis apresentados na revisão bibliográfica.

O AD7912 utiliza o processo de aproximações sucessivas. Sua taxa de amostragem é de até 1 MSPS, resolução de 10 bits e baixo consumo de corrente elétrica, 4 mA. Possui dois canais de entrada analógica e comunicação SPI de até 16 Mbps.

A tensão de referência deste conversor é fornecida pela fonte de alimentação aplicada ao pino Vdd e os pinos de controle e comunicação são apresentados na Tabela 1.

Pino	**Descrição**
/CS	Chip Select – Habilita a conversão e a transferência de dados seriais
DOUT	Data Out – os dados transmitidos são lidos através deste pino
DIN	Data In – A seleção do canal e modo de operação é feito por este
SCLK	Serial Clock – Esta é a fonte de Clock para o processo de conversão
Vin0,Vin1	Entradas Analógicas
VSS	GND
VDD	VCC e Vreferência

Tabela 2: Função dos pinos do Conversor AD

Para este conversor, existem três modos de operação. O normal, o *power-down* e o *daisy-chain*. O modo normal é o que apresenta a taxa de transferência de dados mais elevada, por este motivo o componente será programado para operar desta maneira.

O processo de conversão inicia com a rampa de descida do pino *Chip Select* (/CS). Em seguida deve-se informar o canal que será convertido e o modo de operação. Um byte com esta informação é enviado da seguinte maneira.

Apenas o terceiro e quarto bits são utilizados, o resto é ignorado. O terceiro bit mais significativo é o bit identificador de canal. O quarto bit mais significativo, STY, esta relacionado com o modo de funcionamento do dispositivo. A conversão com STY = CHN força o dispositivo a operar em modo normal. A Figura 8 apresenta o byte de seleção de canal e modo de operação enviado ao conversor AD.

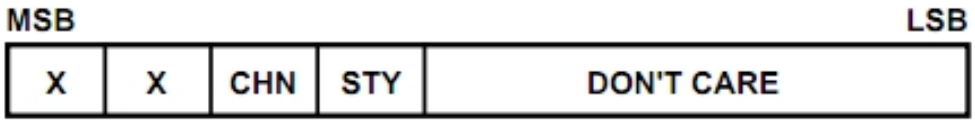


Figura 8: Processo de envio de dado de canal convertido e modo de operação

O resultado da conversão é fornecido ao pino DOUT como um fluxo de dados seriais composto por dois bytes de oito bits. Este fluxo de dados possui dois zeros à esquerda, seguidos do bit que identifica o canal convertido, o bit que indica o modo atual de operação, e 10-bits de dados de conversão, sendo o primeiro bit o mais significativo. Por fim dois zeros à direita. Após o envio destes 16 bits, e com a rampa de subida do pino /CS a conversão é finalizada. A Figura 9 apresenta a sequência dos dois bytes emitidos pelo conversor AD.

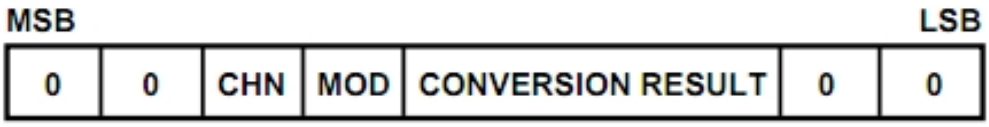


Figura 9: Formato do dado convertido pelo AD

3.1.1.3. MEMÓRIA F-RAM

Os dados convertidos e armazenados na memória externa não precisam ficar registrados no equipamento após o desligamento do sistema. Por este motivo a pesquisa se baseou na consulta de memórias voláteis de tecnologia RAM (*Random Access Memory*).

Grande parte das memórias RAM consultadas apresentaram comunicação paralela, e como a escolha pelo conversor AD de comunicação SPI já havia sido estabelecida houve a necessidade da utilização de uma memória que possuísse este mesmo padrão de comunicação.

Pesquisas posteriores relacionadas á memórias que possuem comunicação SPI levaram a memórias de tecnologia não volátil como EEPROM (*Electrically Erasable Programmable Read Only Memory*), Flash e F-RAM (*Ferroelectric Nonvolatile RAM*).

Memórias EEPROM e Flash podem ser apagadas e re-programadas várias vezes, porém além de possuir limitados ciclos de escrita e leitura, exigem tensões elevadas e que a escrita ocorra mais lentamente.

Memórias F-RAM possuem ilimitados ciclos de leitura e gravação, consumo de energia aproximadamente 3000 vezes menor do que dispositivos EEPROM típicos e velocidade de gravação quase 500 vezes superior. Algumas memórias F-RAM possuem compatibilidade de pinos para substituição direta de EEPROM

A memória FM25CL64 é uma memória com tecnologia F-RAM e por se tratar de uma tecnologia mais nova, seu custo é mais elevado do que o das tecnologias convencionais RAM.

Apesar de não haver a necessidade de utilizar a característica não-volátil da memória F-RAM, ela possui comunicação SPI de até 16 Mbps, pequena quantidade de pinos e velocidade adequada para aplicação no projeto.

Esta memória tem capacidade de armazenamento de 64 Kbits distribuídos em 8.192 endereços de 8 bits. Possui também proteção de escrita via *Hardware* e via *software*, consumo de corrente em *Standby* de aproximadamente 1 uA e operando com *clock* de 16 MHz de 7 mA . A função dos pinos de controle e alimentação desta memória esta descrita na Tabela 2.

Pino	Descrição
/CS	*Chip Select* – Habilita a memória para recebimento ou envio de dados
/WP	*Write Protect* – Previne escritas no Registrador de Status da Memória
/HOLD	*Hold* – Interrompe uma operação da memória
SI	*Serial Input* – Todos os dados são escritos por este pino
SO	*Serial Out* – todos os dados são lidos através deste pino
SCK	*Serial Clock* – Todas as atividades de escrita e leitura são sincronizadas com o clock serial
VSS	GND
VDD	VCC entre 3 e 3,6V

Tabela 3: Função dos Pinos da Memória FM25CL64

Os dados de entrada são capturados pela memória durante a borda de subida do *clock* e os dados lidos são emitidos na borda de descida. Existem dois modos de operação da SPI que podem ser configurados para o correto funcionamento da memória. São eles o Modo 0 e o Modo 3. O que diferencia estes dois modos é o estado inativo do *clock* serial. No Modo 0 o estado inativo do *clock* é configurado como nível baixo, enquanto no Modo 3 o estado inativo do *clock* é configurado como nível alto.

Esta memória possui seis códigos específicos, Tabela 4, que são interpretados pelo protocolo SPI e que controlam as funções realizadas pela memória. Podem ser divididas em três categorias.

A primeira categoria é a de comandos que não possuem operações subseqüentes. Eles realizam uma função simples tal como habilitar uma operação de escrita. A segunda categoria é a de comandos seguidos por um Byte. Eles operam no Registrador de Status. A terceira categoria inclui comandos para as operações de memória, seguidos pelo endereço e um ou mais bytes de dados.

Nome	Descrição	Valor Binário	Valor Decimal
WREN	Habilita *Latch* de Escrita	0000 0110b	6
WRDI	Desabilita Escrita	0000 0100b	4
RDSR	Leitura do Registrador de Status	0000 0101b	5
WRSR	Escrita no Registrador de Status	0000 0001b	1
READ	Leitura de dados da memória	0000 0011b	3
WRITE	Escrita de dados na memória	0000 0010b	2

Tabela 4: Códigos de Operação da memória FM25CL64

A operação de escritas na memória começa com o código de operação *WREN* enviado pelo dispositivo mestre após a rampa de descida de /CS. O próximo código de operação *WRITE* é enviado, seguido por dois bytes de endereço. Os 3 bits superiores do endereço são ignorados. No total os 13 bits definem o endereço do primeiro byte de dados da operação de escrita. Este é o endereço de inicio do primeiro byte de dados da operação de escrita. Bytes subseqüentes são bytes de dados, os quais são escritos seqüencialmente. Os endereços são incrementados internamente enquanto o barramento continua a emitir *clock* serial e /CS é mantido no nível lógico baixo. Se o último endereço, 1FFFh, é alcançado o contador retorna para 0000h. O dado é escrito primeiro pelo bit mais significativo. A rampa de subida de /CS termina uma operação de escrita.

A operação de leitura ocorre inicialmente com a rampa de descida de /CS, em seguida deve-se enviar o código de operação *READ*. Na seqüência envia-se um valor de endereço de dois bytes. Os 3 bits superiores do endereço são ignorados. No total, os 13 bits especificam o endereço do primeiro byte a ser lido. Depois do código de operação e o endereço serem emitidos, o dispositivo emite o dado lido nos próximos 8 clocks através do pino SO. A entrada SI é ignorada durante a leitura dos bytes de dados. Bytes subseqüentes são bytes de dados, que são lidos seqüencialmente. Os endereços são incrementados internamente enquanto o barramento continua a enviar o clock e o /CS é mantido baixo. Se o ultimo endereço, 1FFFh, é alcançado o contador retorna para 0000h. O dado lido envia primeiro o bit mais significativo. A rampa de subida de /CS termina uma operação de leitura.

3.1.1.4. MULTIPLEXADORES DE COMUNICAÇÃO

Os pinos de comunicação (SDI e SCLK) da memória devem ser multiplexados para que a comunicação deste dispositivo com o microcontrolador ou com o conversor AD seja estabelecida. Isto se faz necessário, pois o microcontrolador envia os códigos de operação e o AD envia os dados convertidos para a memória pelo mesmo pino de entrada de dados (SDI).

Dois multiplexadores SN74LVC2G53DCUR são aplicados para esta função. O pino de controle, pino A dos dois multiplexadores, é acionado pelo mesmo pino do microcontrolador e a Tabela 4 apresenta a sua lógica de operação.

Nível lógico pino A	Comunicação com
0	Microcontrolador
1	Conversor AD

Tabela 5: Operação dos multiplexadores de comunicação

3.1.1.5. CIRCUITO ANALÓGICO

Para ser encaminhado ao conversor AD, o sinal analógico precisa que sua amplitude máxima seja de 4 V, tensão de referência do AD. Para que valores maiores ou menores estejam dentro desta faixa um estagio de amplificadores operacionais é inserido logo na entrada do circuito.

Dois amplificadores operacionais, um para cada entrada, fazem uso da configuração de amplificador inversor, Equação 2. Os oito ganhos dos amplificadores são selecionados pelo microcontrolador através dos pinos ADDA, ADDB e ADDC dos multiplexadores MAX4051 que selecionam a resistência Rf, como mostrado na Figura 10. O ganho é selecionado de acordo com a configuração dos pinos de controle do multiplexador apresentado na Tabela 5.

Osciloscópios utilizam por padrão oito divisões verticais relacionadas a amplitude do sinal, este mesmo padrão é aplicado ao *software* USBScope 2.0. Na Tabela 5 é possível ainda verificar a escala selecionada para cada ganho dos amplificadores.

$$A = \frac{V_o}{V_i} = -\frac{R_f}{1\,M\,\Omega} \quad (2)$$

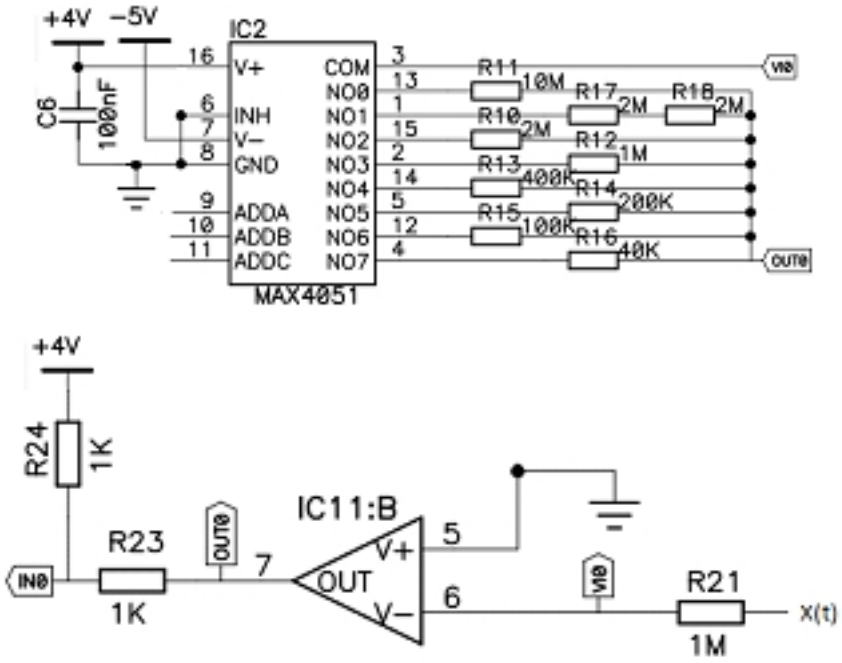


Figura 10: Configuração inversora do estagio de amplificadores operacionais

ADDA	ADDB	ADDC	Ganho (A)	Escala (V/div)
0	0	0	10	0.1
1	0	0	4	0.25
0	1	0	2	0.5
1	1	0	1	1
0	0	1	0.4	2.5
1	0	1	0.2	5
0	1	1	0.1	10
1	1	1	0.04	25

Tabela 6: Seleção do ganho dos amplificadores operacionais

O divisor de tensão, inserido na saída dos amplificadores operacionais, acrescenta um nível DC para deslocamento do sinal de saída, de modo que este não possua valores negativos, pois o AD utilizado opera apenas com tensões positivas. Além disso, esse divisor resulta em um ganho de 0,5 V/V.

3.1.1.6. CIRCUITOS AUXILIARES

O circuito integrado LM2660 é um conversor de tensão que utiliza a técnica de capacitor chaveado. Através de uma tensão positiva e da utilização de capacitores, este componente gera uma tensão negativa de mesma amplitude. Esta tensão é necessária para que os amplificadores operacionais trabalhem com tensão simétrica, possibilitando a amplificação de sinais negativos.

O regulador de tensão ZXCL330E5TA fornece a tensão de 4.3 V estabilizada para alimentar o conversor AD, e a memória externa.

O esquemático completo do circuito do USBScope 2.0 pode ser acompanhado na Anexo 1 e Anexo 2.

3.1.2. FIRMWARE

3.1.2.1. LINGUAGENS DE PROGRAMAÇÃO

As linguagens de programação mais aplicadas no desenvolvimento de *firmware* de microcontroladores da família PIC são assembly e C. Ambas possuem suas vantagens e desvantagens de aplicação.

A linguagem assembly possui como vantagem principal um maior desempenho de processamento, pois permite uma maior interação do projetista com o *hardware*, ou seja, acesso a registradores e seqüências de inicialização. Por outro lado, um código nesta linguagem gera uma grande desproporção entre o conjunto de instruções e as tarefas executadas pelo processador, obrigando o programador a decompor cada tarefa em operações elementares que além de demorado é um processo que contribui para a ocorrência de erros e não ajuda a manter a estruturação do código.

As razões que explicam a popularidade da linguagem de programação C para aplicações em sistemas embarcados são além da grande quantidade de compiladores com elevado desempenho que facilitam o desenvolvimento, o fácil acesso ao *hardware* e os baixos requisitos de memória exigidos.

Os compiladores atuais, em sua maioria, permitem uma programação mista, onde a linguagem de alto nível é a principal linguagem aplicada, mas em caso de rotinas específicas que se pretende a máxima eficiência do código o assembly possa ser aplicado.

3.1.2.2. COMPILADOR MIKROC

A comunicação USB acontece com o transporte de dados por dois pinos, Data+ e Data-. Dispositivos personalizados conectados a USB podem exigir um *driver* de dispositivo personalizado e específico, ou podem pertencer a uma classe de dispositivos. Caso possuam funcionalidade semelhante podem utilizar o mesmo *driver*.

A classe mais comum de dispositivos USB é a de Dispositivos de Interface Humana (HID). *Drivers* para esta classe de dispositivos se encontram em todos os sistemas operacionais o que torna interessante sua utilização.

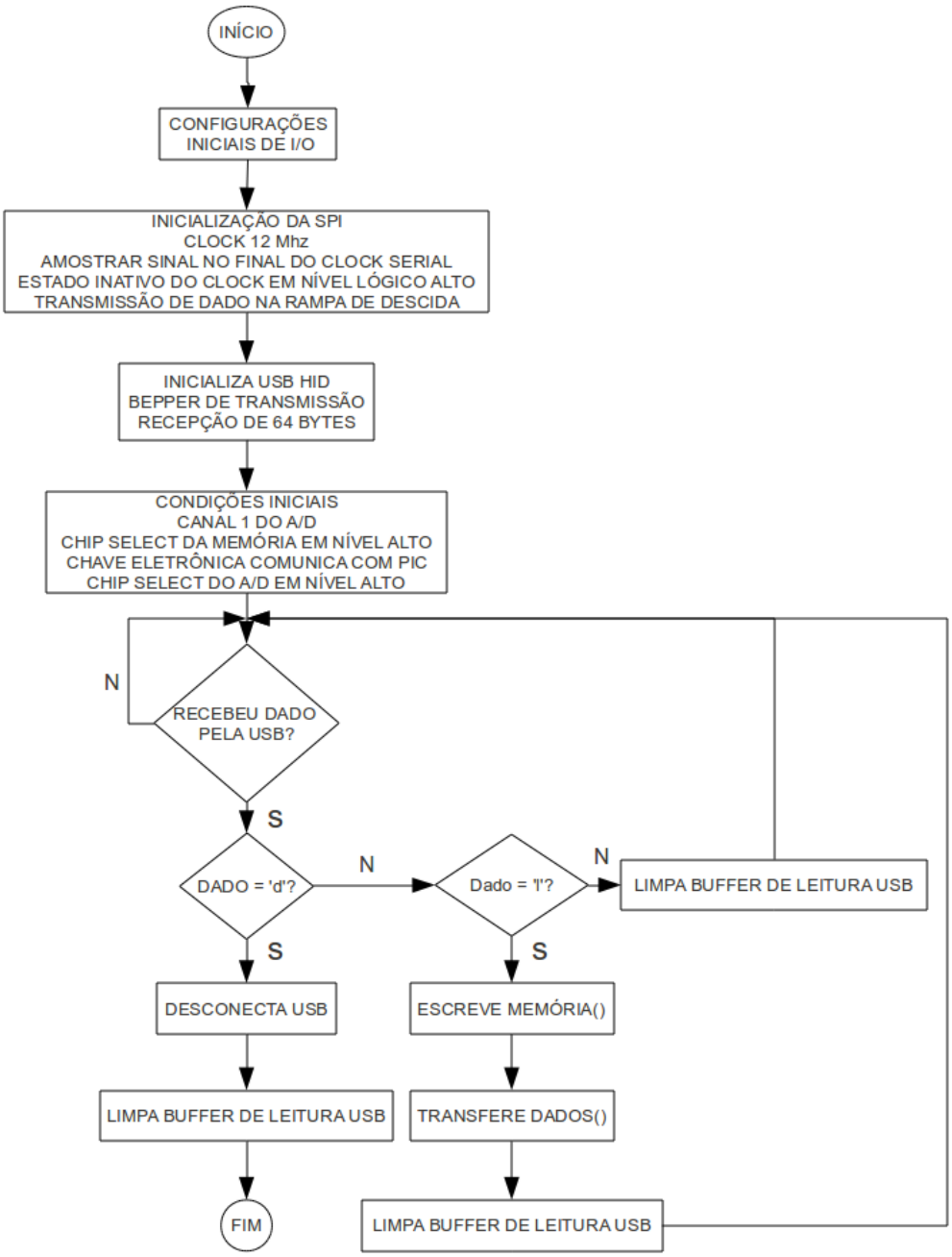
Todos os dispositivos USB possuem uma hierarquia de descritores que contém várias das suas características, como a identificação do fabricante, a versão do dispositivo, a versão de suporte USB, os requisitos de energia, o número e tipo de parâmetros, dentre outros.

O compilador MikroC-Pro possui uma ferramenta auxiliar que gera os descritores HID para que então possam ser incluídos no *firmware*. Além desta ferramenta este compilador possui um terminal de comunicação HID e uma extensa biblioteca de funções, dentre elas USB e SPI.

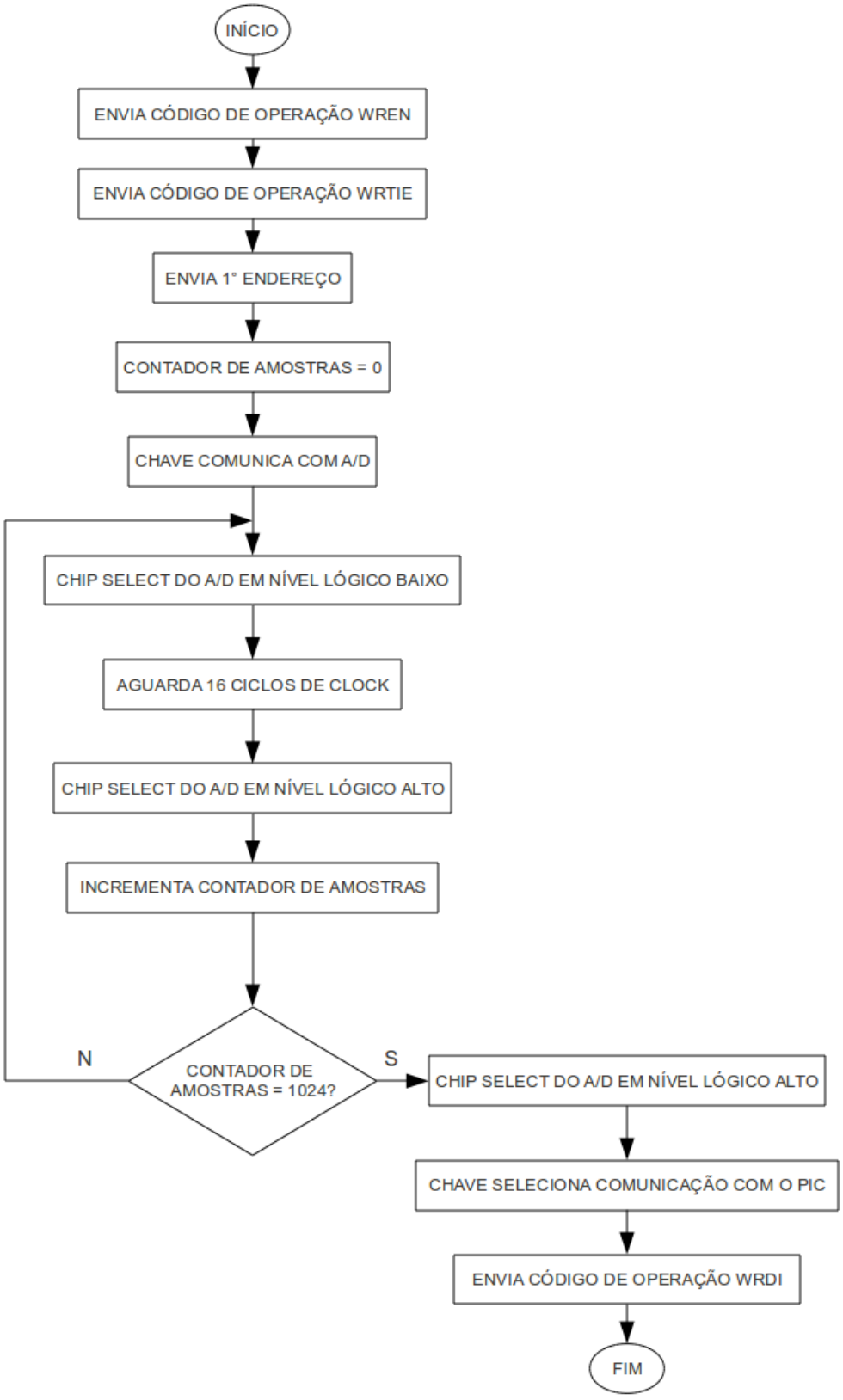
O firmware que permite o teste de conversão, gravação, leitura e transporte de dados via USB foi elaborado utilizando programação mista, pois a conversão do sinal pelo AD e a gravação destes dados na memória exige que o pino /CS do AD seja mantido em nível lógico baixo exatamente por 16 ciclos de *clock*. Esta função foi ajustada utilizando instruções em assembly do tipo *NOP*.

Os Fluxogramas 1, 2 e 3 apresentam o funcionamento do *firmware* que controla as conversões, grava os dados na memória e transmite-os para a USB. Para o desenvolvimento deste *firmware* foi estabelecido que uma conversão do AD

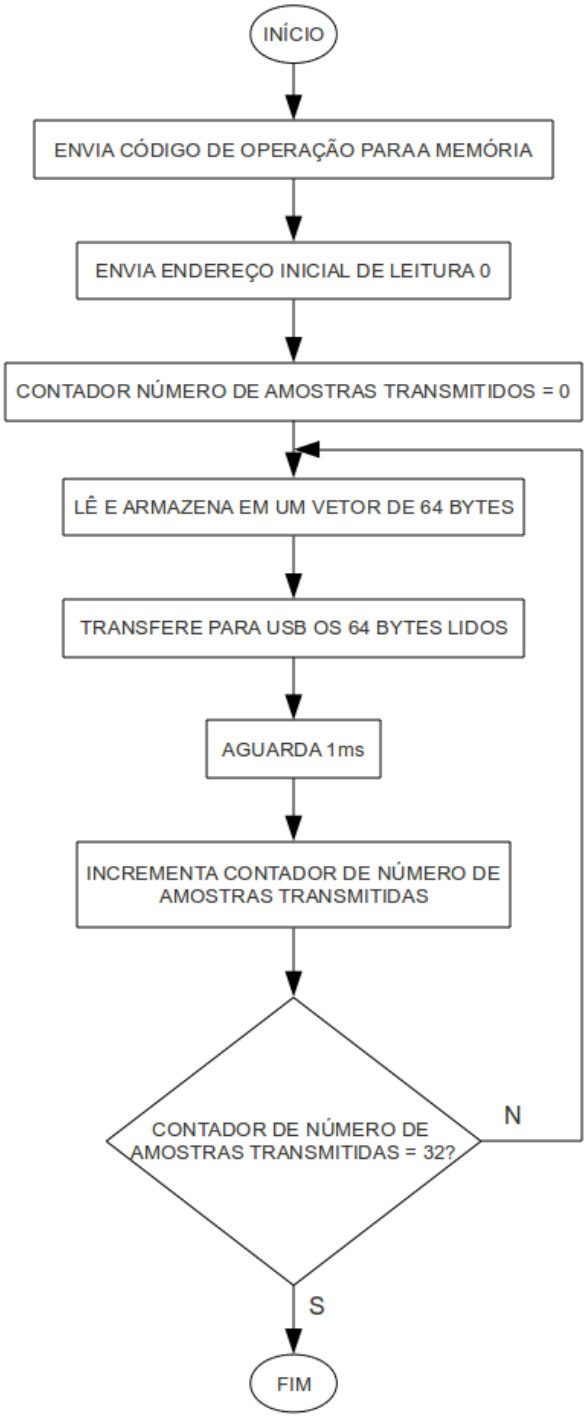
ocupar dois bytes da memória, pois uma conversão do AD corresponde a dezesseis bits de dados.



Fluxograma 1: Função *Main* do *Firmware*



Fluxograma 2: Função de conversão de dados e escrita na memória



Fluxograma 3: Função leitura de dados da memória e transferência para o microcomputador

3.1.3. SOFTWARE

Tomando como base o projeto do USBScope 1.0, foi decidido utilizar a IDE *Visual Basic* 6 para a implementação de uma primeira versão do *software* USBScope 2.0, por ser uma linguagem de fácil aprendizado, orientada a eventos e uma das primeiras a oferecer suporte a *drag and drop* (arrastar e soltar), permitindo a seus usuários criar aplicações simples ou complexas com bastante facilidade.

Após a criação desta primeira versão em *Visual Basic*, e uma vez aprendidos os possíveis erros de comunicação e particularidades do projeto, se criaria uma versão utilizando a IDE Visual C++ 6, pertencente ao mesmo pacote do Visual Studio 6. Escolheu-se montar o *software* desta forma devido principalmente ao uso da ferramenta EasyHID, que implementa a comunicação USB entre o aplicativo e o *hardware* correspondente.

O EasyHID é uma ferramenta disponibilizada em um pacote de aplicativos para a programação de microcontroladores conhecido como microcode e ofertado gratuitamente pela empresa mecanique. A função deste programa é gerar códigos com dll's próprias para a comunicação com dispositivos HID. Além disto, existe a possibilidade de criação de código para a manipulação dos sinais oriundos da porta USB em três diferentes linguagens, *Visual Basic* 5, Visual C++ 6 e Delphi 5.

Ao se montar o código em C++, se teriam todas as facilidades e a robustez que esta linguagem proporciona, como a facilidade da orientação a objeto e suporte a execução de código concorrente (*multithreading*).

3.1.3.1. DESENVOLVIMENTO DO CÓDIGO

Visual Basic é uma IDE *(Integrated Development Enviroment)* criada pela Microsoft, parte integrante do pacote Visual Studio, e derivada de outra linguagem mais antiga da época da plataforma DOS chamada BASIC. Basic significa Beginners All-purpose Symbolic Instruction Code (código de instruções simbólicas para todos os fins para iniciantes), sendo uma linguagem de fácil aprendizado, apresentando uma interface amigável com bons recursos de debug.

A última versão da plataforma derivada da linguagem BASIC foi o *Visual Basic* 6, sendo sucedido, a partir da versão 2008 da plataforma Visual Studio, pelo *Visual Basic* .NET ou simplesmente *Visual Basic*, mudando sua estrutura e ficando compatível com a plataforma .NET.

Possui suporte mínimo à orientação a objeto, apenas para a criação de classes, porém sem as facilidades criadas por herança, polimorfismo, e outras características que facilitam a criação e manutenção de código. É uma IDE proprietária, compatível apenas com a plataforma Windows, custando atualmente em sua versão profissional $599,00, e em sua versão acadêmica $100,00, separadamente ao Visual Studio.

Para o *software* do USBScope 2.0, a única grande desvantagem apresentada, seria não apresentar um suporte a *threads*, podendo apresentar um comportamento semelhante com o uso da função *DoEvents*(), que permite uma execução mais controlada de diversos processos em fila, porém não concorrentes.

3.1.3.2. IMPLEMENTAÇÃO

A interface de usuário do *Visual Basic 6* depende diretamente dos códigos gerados pelo *EasyHID*. A seguir são mostrados os passos de criação do código e a descrição dos eventos gerados por ele.

Na Figura 11 é mostrada a tela inicial do *MicroCode Studio* e o menu onde se encontra a ferramenta *EasyHID.* Na Figura 12, é apresentada a tela inicial do *EasyHID*, com os campos para a edição do nome de companhia, produto, que são obrigatórios, e número serial do aplicativo, que é opcional, além de explicações sobre a ferramenta.

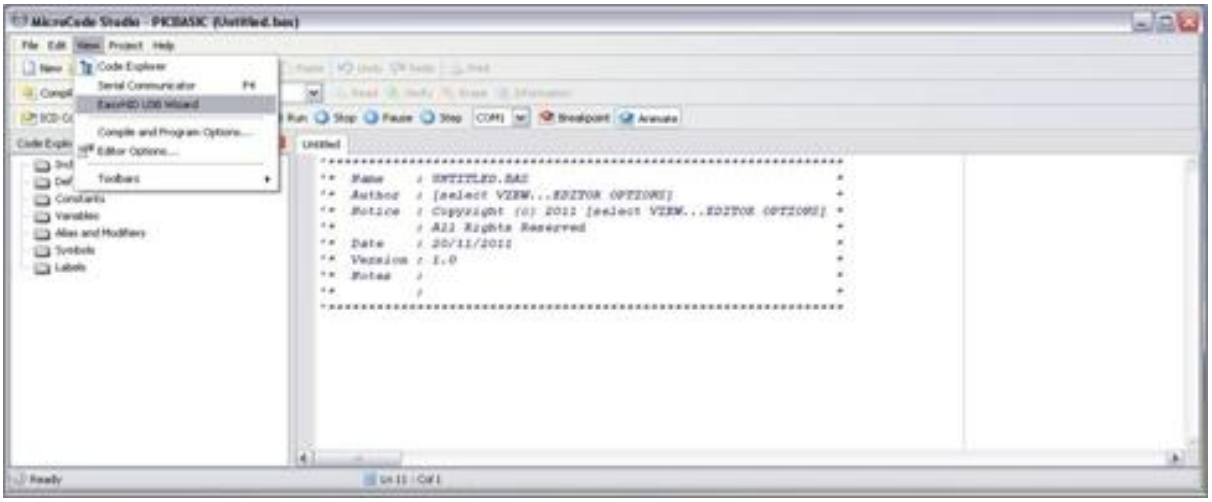


Figura 11: Tela inicial do Microcode Studio

Figura 12: Tela Inicial EasyHID

Uma vez definidos os nomes de companhia e produto, deve-se colocar as identificações de vendedor (VID) e de produto (PID), Figura 13. O VID é um número único no mundo, e assinado para desenvolvedores USB através do site [6]. Como este projeto é apenas para fins acadêmicos, utilizamos identificações genéricas, 1 para PID e o VID está escrito em hexadecimal, de modo que o número 4660 na base

16 é igual ao número 1234 na base 10, valor este que deve ser definido do lado do *hardware*.

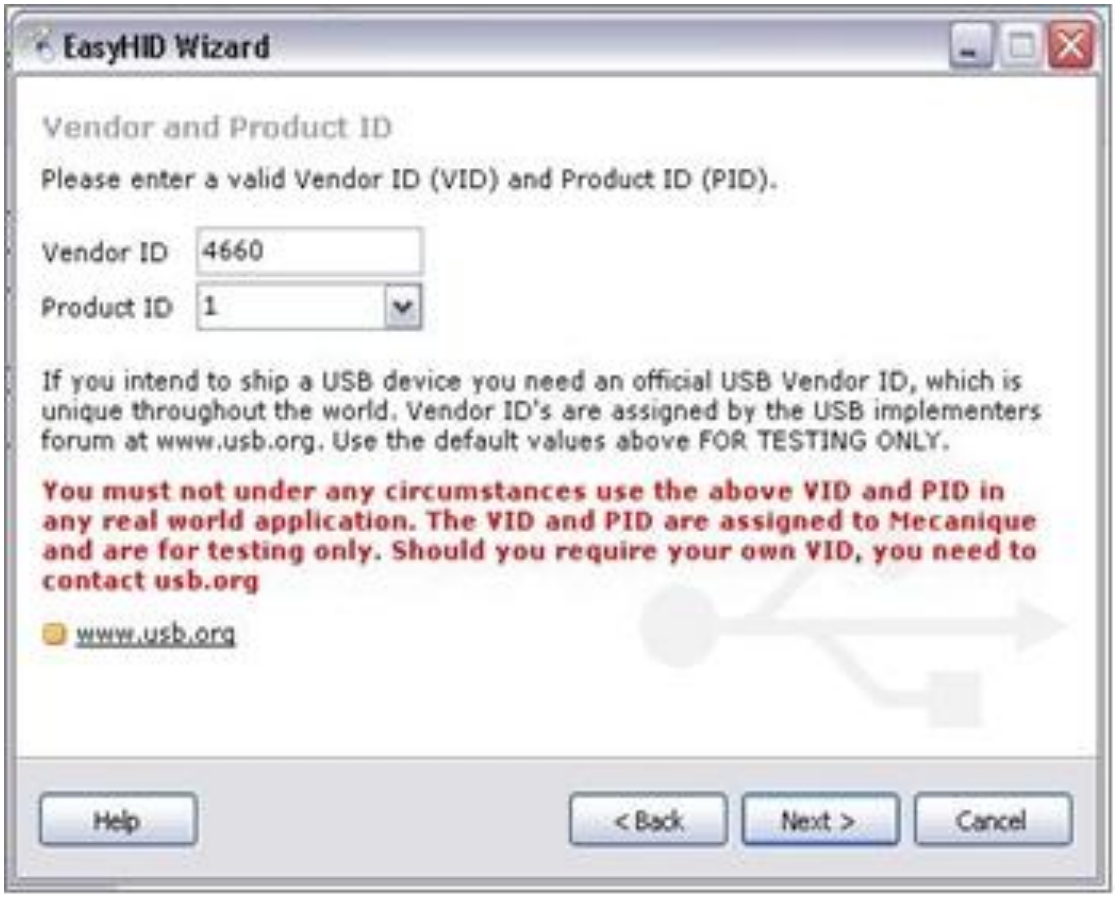


Figura 13: Identificação VID e PID

Em seguida são definidas as características de comunicação, consumo de energia, e tamanho dos *buffer's* de entrada e saída de dados que o aplicativo terá, como mostrado na Figura 14. Para o *software* foram definidos os valores como apresentado na Tabela 6.

Polling (Input)	**1 ms**
Polling (Output)	**1ms**
Bus Power	**(100 x 2) mA**
Buffer (Input)	**64 bytes**
Buffer (Output)	**64 bytes**

Tabela 7: Parâmetros para criação do *software*

Os valores de polling, que definem o tempo de atualização na leitura dos *buffer's* de entrada e do de saída, foram colocados em seus valores mínimos. O

tamanho dos *buffer's* de entrada e saída foram colocados em seus valores máximos e a corrente consumida pelo aparelho é de aproximadamente 200 mA.

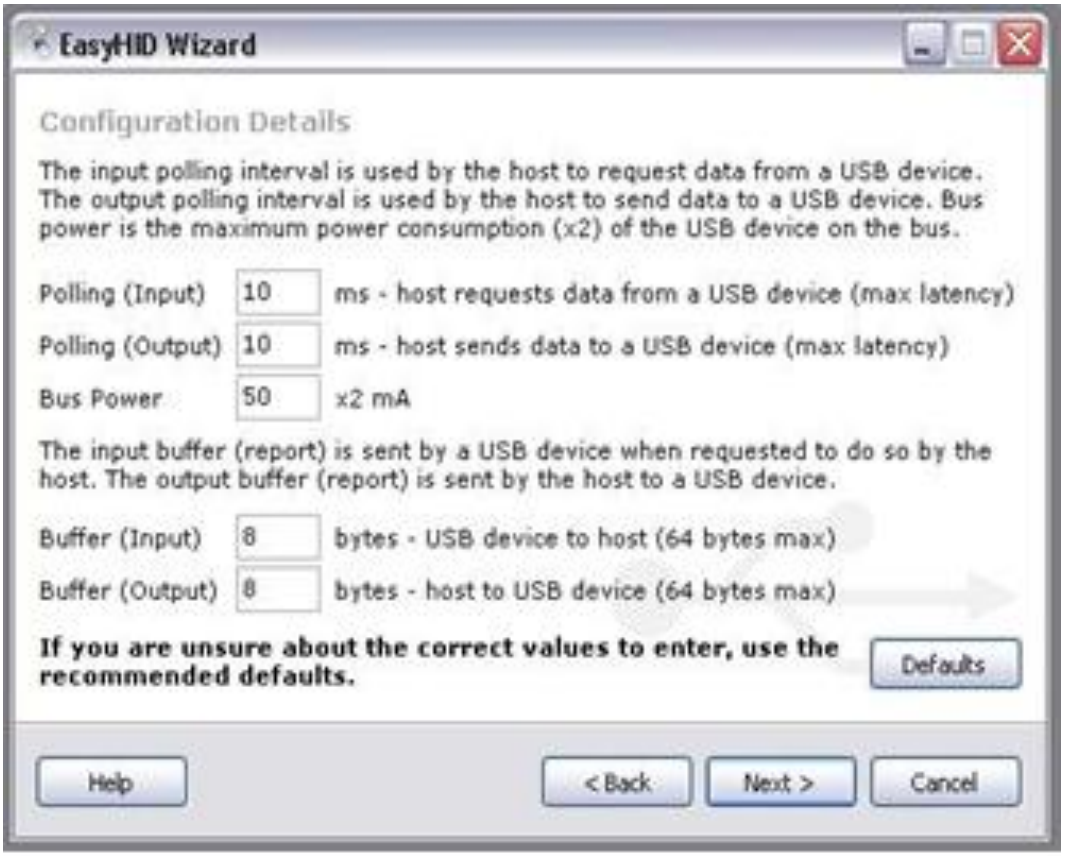


Figura 14: Configuração da comunicação

Com as configurações do dispositivo definidas, basta agora definir quais linguagem e microcontrolador PIC o usuário deseja usar, e em qual diretório se salvar o projeto, como mostrado na Figura 15, e então o projeto é gerado como na Figura 16.

Figura 15: Configurações de compilação

Figura 16: Geração do código

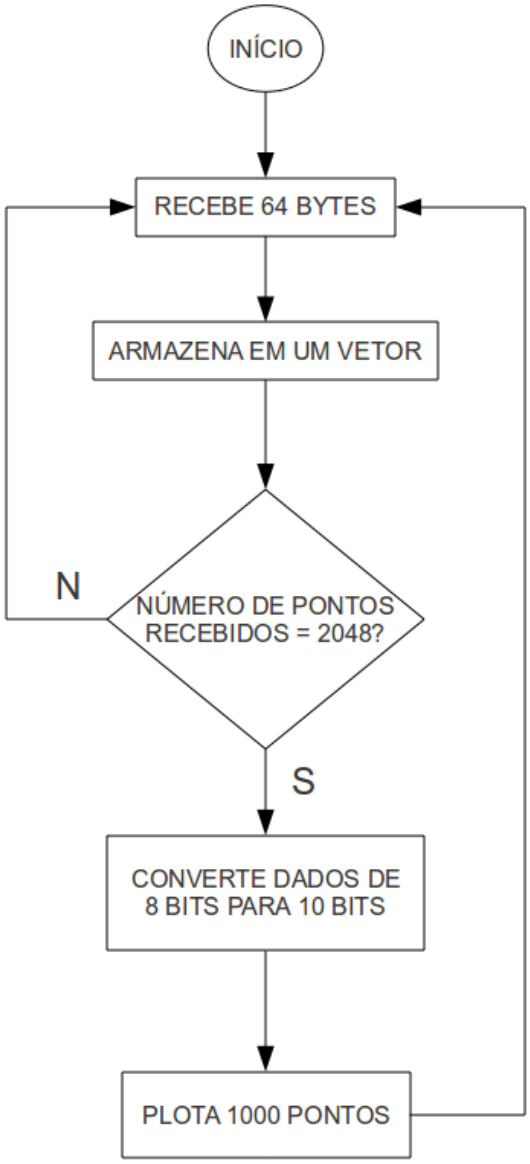
O *EasyHID* cria, além dos arquivos normais de projeto, dois outros arquivos um .bas e um .frm. O arquivo .bas, define o código dos eventos necessários para acesso à porta USB, como *Plugged(), UnPlugged(), OnRead(), OnWrite(),* etc. O arquivo .frm, é um form, onde se consomem estes eventos.

Do ponto de vista do que é recomendado para a criação de *softwares* visuais, não é correto se inserir códigos de tratamento de informação em componentes visuais, pois estes são constantemente alterados na maioria dos projetos, o que acarreta em mudanças constantes em toda a manutenção dada ao código.

3.1.3.3. INTERFACE

Uma tela básica do USBScope 2.0, é apresentada na Figura 29. A mesma apresenta a tela, para visualização da onda, e alguns controles definidos por canal, como habilitação e desabilitação do canal e indicativo dos valores médio e RMS da onda capturada.

A plotagem de pontos do programa acontece no evento *OnRead(),* gerado na criação do código pelo *EasyHID,* como já explicado anteriormente. O Fluxograma 4 exemplifica o algoritmo implementado para a plotagem.



Fluxograma 4: Programa VB6

O programa recolhe 1024 pontos com valores que variam de 0 até 1023 (10 bits). Porém como a transmissão é feita com uma sequencia de 8 bits, um ponto de informação é representado por dois bytes de entrada. Isto faz com que, para receber e armazenar os 1024 pontos, tendo um *buffer* de entrada de 64 bytes, seja necessário receber 2048 bytes, que são convertidos para valores de 10 bits, dois a dois, e armazenados em um vetor que será posteriormente plotado.

O resultado alcançado até então com a utilização do *Visual Basic* foi o demonstrado anteriormente, além da montagem separada de alguns algoritmos para o cálculo das funções que estariam presentes no *software*, como as de valor médio, RMS, pico e pico a pico. Porém com a mudança da comunicação entre o *hardware* e o *software*, agora sendo implementada pelo chip FT232RL, o algoritmo de recebimento de dados foi alterado, o que exigiu uma mudança também na montagem do *software*.

A mudança que o chip FT232R causou no programa é sentida na possibilidade de criação de *softwares* em diferentes sistemas operacionais, além de utilizar a USB como uma porta serial emulada. Isto causa um decréscimo da velocidade de transmissão, quando comparado com a que se tinha anteriormente com o uso do HID. Para uma transmissão de 2048 *bytes*, o USBScope 2.0 levava em torno de 20 ms para a transmissão, enquanto o USBScope 2.10 leva em torno de 300 ms. Mesmo sendo uma perda considerável no tempo de transmissão, não atrapalha o projeto consideravelmente, poisa tualização da tela se dará em torno de 0,5 s, o que não é desagradável a percepção humana.

Surgiram oportunidades diversas na escolha da linguagem de programação e IDE a serem utilizadas, de forma que ambas deveriam seguir determinados requisitos para serem escolhidas ou não, requisitos estes que eram:

- Suporte para comunicação serial
- Suporte a *multi-threading*
- Biblioteca gráfica de fácil utilização
- Orientação a Objeto

Além de preferencialmente, ter outras características como:

- Ambiente fácil de configurar e programar

- Gratuito
- Multi-Plataforma

Ficou decidido, pelas características apresentadas das linguagens estudadas, utilizar Java, pois além de atender a todos os requisitos desejados pelo *software*, possui ainda uma portabilidade única, que é importante para uma aplicação em qualquer *hardware*, como por exemplo, *notebooks*, telefones celulares, *tablets*, etc. Do que havia sido feito anteriormente, muda-se a estrutura interna do programa, porém os algoritmos de plotagem de dados e cálculo de funções, serão reutilizados.

Java é uma linguagem de programação desenvolvida pela empresa *Sun Microsystems*, agora parte da *Oracle Corporation*, tendo um sintaxe muito parecida com o C e o C++, porém um conceito de orientação a objeto mais forte, não aceitando, por exemplo, herança múltipla em seu código.

Java é compilada para um *bytecode* e executada em uma máquina virtual JVM (*Java Virtual Machine*) independente da arquitetura do computador, o que a torna altamente portável para qualquer arquitetura de *hardware* e sistema operacional. É atualmente a linguagem de programação mais popular do mundo desde 2006 [6], seguida de C e C++.

Como IDE de programação foi escolhido o *Netbeans*, que atende a todas as exigências listadas acima, sendo a IDE de mais fácil uso encontrada. Está em sua versão 7.0.1, juntamente com a versão JDK *(Java SE Development Kit*).

3.1.3.4. MODO DE IMPLEMENTAÇÃO

Para a criação do *software*, será usado o modelo de três camadas de forma adaptada. Neste modelo define-se o *software* em camada de apresentação, a interface com o usuário propriamente dita, camada de negócio, que no caso do USBScope 2.10 deu-se o nome de camada de processamento, onde ocorrerá todo o tratamento de código e eventos do *software*, e camada de persistência, que nos meios empresariais, é onde fica a informação que é armazenada nos bancos de dados, no caso do USBScope 2.10 será a camada de comunicação, que receberá e transmitirá os dados oriundos da porta USB.

Utilizar três camadas deixa o código mais organizado e modular, facilitando a manutenção do mesmo. Neste caso a classe de processamento funciona como uma classe gerente, fazendo chamadas as funções visuais da camada de apresentação e recebendo e transmitindo dados para a camada de comunicação.

3.1.4. DIFICULDADES ENCONTRADAS

3.1.4.1. USB

Com o intuito de diminuir o número de protocolos de comunicação, criando um padrão de transmissão de dados entre periféricos e computadores, grandes fabricantes de computadores como Apple, HP, Intel, NEC, Microsoft entre outras, formularam o padrão USB. Estes fabricantes disponibilizam vasta documentação para as várias formas de acesso USB possíveis.

Para que pudéssemos entender qual o melhor método a ser aplicado ao projeto, muito conteúdo e exemplos sobre o assunto USB foi analisado até o entendimento que a classe HID era a melhor opção, tanto pelo fácil acesso ao sistema operacional quanto por ferramentas de apoio ao desenvolvimento disponível.

3.1.4.2. MICROCONTROLADOR PIC18F2550 E COMPILADOR MIKROC

Para efetuar o controle de aquisições de dados do AD nas taxas mais elevadas inicialmente foi considerada a utilizar as funções de *delay* existentes no compilador MikroC, porém estas rotinas, como na maioria dos compiladores em C, apresentam aproximações de tempo o que impede o controle de 16 ciclos de *clock* necessários para uma amostra.

A utilização de um dos *timers* do microcontrolador também não funcionou, pois este método faz uso da função de inturrupção do compilador. Este desvio de programa ultrapassa o valor de *clock* pretendido, não sendo aceitável sua utilização.

Como este compilador aceita programação mista, o problema de conversão para as taxas mais elevadas foi solucionado inserindo a instrução NOP (linguagem assembly) para ajuste dos ciclos de *clock* necessários para uma conversão.

3.1.4.3. PROBLEMAS DE SINCRONISMO

Para que a comunicação do microcomputador com o microcontrolador PIC18F2550 via USB seja estabelecida é necessário que um *clock* de 48 MHz com variação de no máximo 1% seja aplicado ao microcontrolador. Inicialmente utilizando um cristal oscilador de 16 MHz foi possível gerar os 48 MHz para comunicação USB utilizando o circuito multiplicador de freqüência PLL (*Phase Lock Loop*) do PIC18F2550. Este mesmo *clock,* de 16 MHz, estava sendo utilizado como *clock* de referência para as conversões do AD. Testes, utilizando um osciloscópio do laboratório (Lecroy LT584), mostraram que o PLL do microcontrolador apresentava defasagens aleatórias (perda de sincronismo de fase) que ocasionavam problemas na comunicação SPI entre o AD e a memória.

Na tentativa de solucionar este problema, foi utilizado um gerador de *clock* externo (Agilent P4432B) de 48 MHz, desativando a utilização do PLL interno do PIC. Com isso o problema de sincronismo foi resolvido, e foi desenvolvido um oscilador a cristal, de pequenas dimensões, inserido na placa de circuito impresso original. No processo de desenvolvimento deste circuito oscilador foram efetuadas simulações em computador e medições em um analisador de redes (HP 3577A) para verificar a freqüência de oscilação e a estabilidade do circuito. O circuito foi montado sobre um plano condutor de cobre de pequenas dimensões e inserido de forma adaptada na placa de aquisição de dados. Foram testadas várias topologias de osciladores com transistores bipolares diversos. Foram necessários aproximadamente três meses para obter resultados satisfatórios com o circuito mostrado na Figura 17.

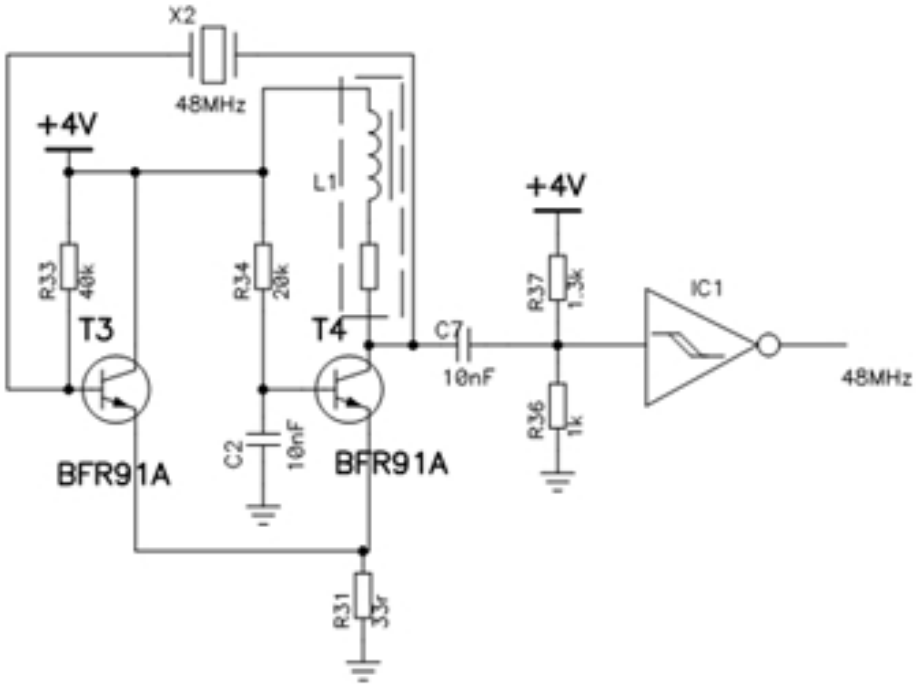


Figura 17: Oscilador a cristal

O *clock* de referência, para o conversor AD e a memória, passou a ser fornecido pelo pino CLKO do microcontrolador, que fornece a frequência do *clock* de entrada (48 MHz) dividido por 4. Dessa forma o clock do AD passou a ser 12 MHz, o que causou uma diminuição na taxa de amostragem de 1 MSPS para 750 KSPS.

As análises e testes de transmissão subsequentes apresentaram resultados coerentes confirmando a correção do problema de sincronismo. Estes resultados serão apresentados no capítulo resultados.

Após aproximadamente um mês de testes, o circuito oscilador externo apresentou problemas de funcionamento por razões desconhecidas, e como o cronograma já estava em atraso decidiu-se partir para uma solução alternativa.

Dentre as opções analisadas, a que se apresentou mais adequada foi o circuito integrado FT232RL. Este componente é um conversor USB serial programável, possui pinos de saída que podem ser configurados com os *clocks* de 48, 24 e 12 MHz além de o fabricante disponibilizar *drivers* para aplicação em diversos sistemas operacionais, como Windows 32 e 64 bits, Linux, MacOS e Android.

3.2. USBSCOPE 2.10

A Figura 18 apresenta no diagrama em blocos as mudanças aplicadas ao circuito do USBScope. Neste novo formato o FT232RL passa a ser responsável por receber e transmitir os dados da USB e fornecer o clock de 48 MHz para o microcontrolador. De acordo com o fabricante a estabilidade do oscilador que é melhor que 0,2% (apesar de ser um oscilador integrado no próprio *chip*), o que o torna adequado para a aplicação neste projeto.

Testes comprovaram que a utilização de um *clock* de referência externo ao microcontrolador favorecia a ocorrência de perda de sincronismo de fase, por isso decidiu-se pela utilização do *clock* de 12 MHz fornecido pela saída CLKO do microcontrolador.

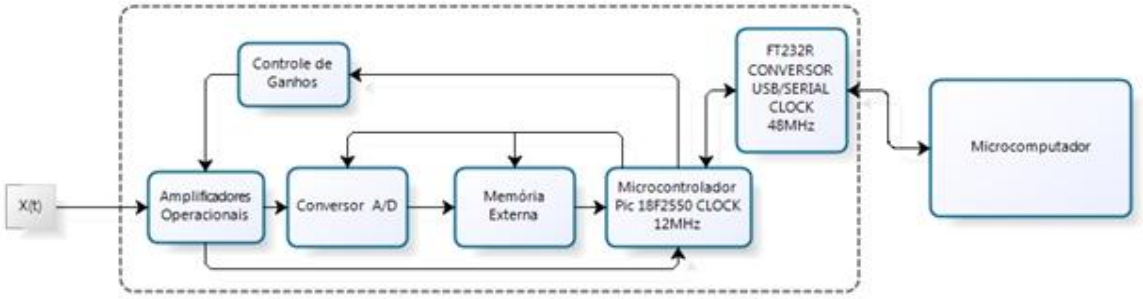


Figura 18: Diagrama em Blocos USBScope 2.10

3.2.1. HARDWARE

O controle de ganhos do circuito analógico sofreu mudança com a substituição dos mutiplexadores por potenciômetros digitais. A Figura 19 apresenta esta nova configuração.

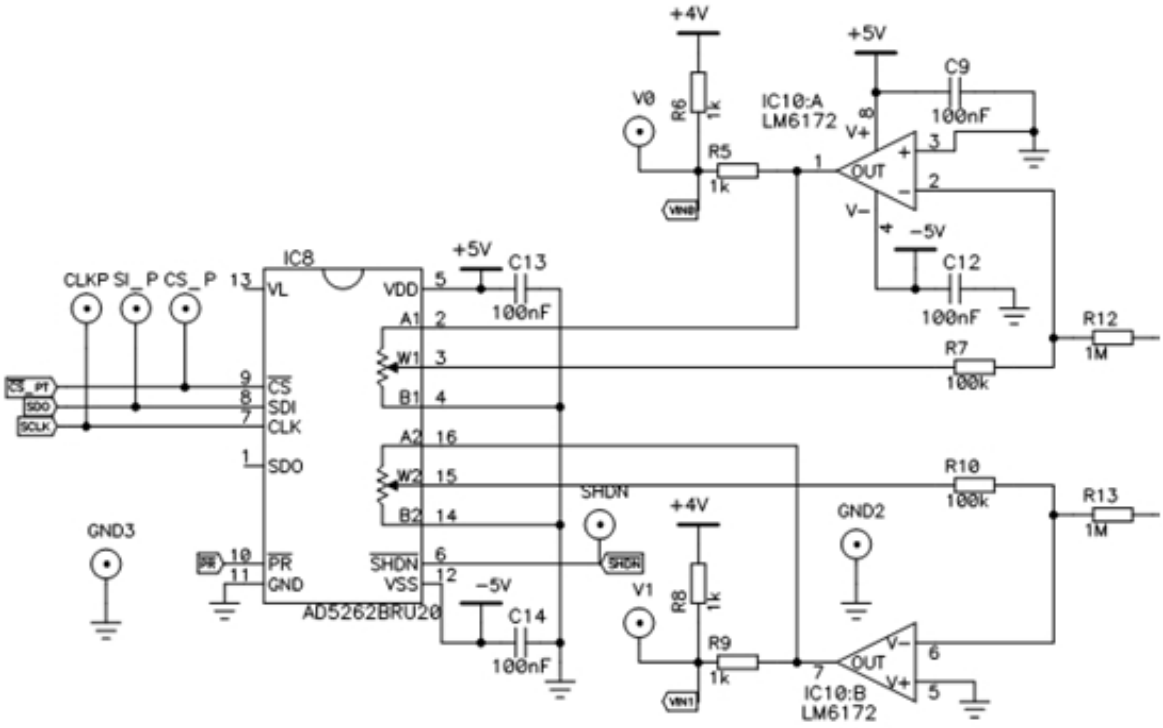


Figura 19: Controle de ganhos do circuito analógico

A equação 3 apresenta o ganho do amplificador operacional controlado pelo potenciômetro digital.

$$A = -\frac{100\text{k}\,\Omega\, R_A}{1\text{M}\,\Omega}\left(\frac{1}{100\text{k}\,\Omega}+\frac{1}{R_A}+\frac{1}{R_B}\right) \quad (3)$$

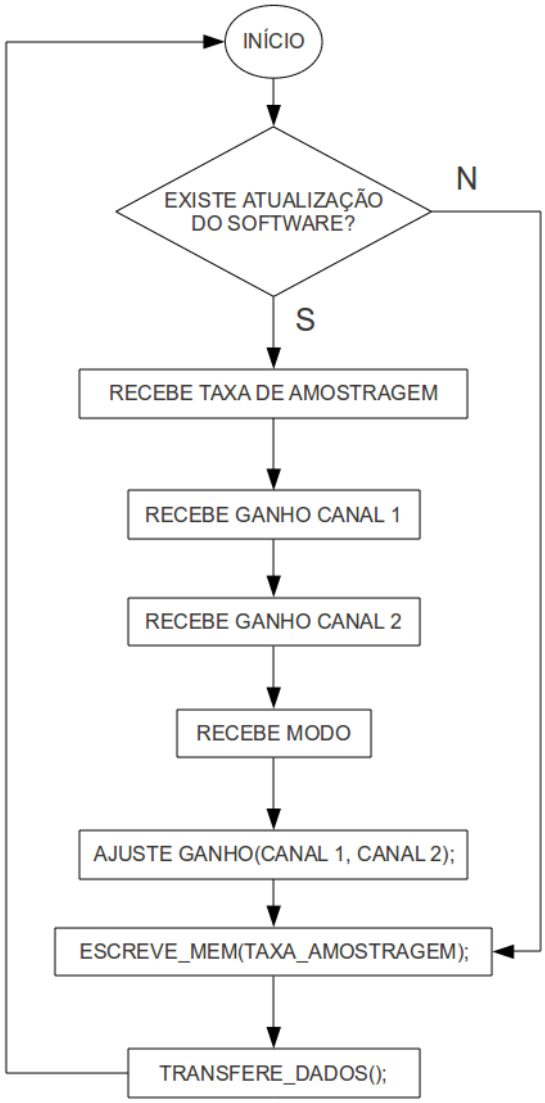
Para o controle de conversões do AD e armazenamento das amostras na memória foi utilizado no USBScope 2.10 o mesmo *firmware* da versão 2.0, modificando apenas a comunicação do microcontrolador de USB para UART (serial).

O esquemático completo do circuito do USBScope 2.10 pode ser acompanhado no Anexo 3 e Anexo 4.

3.2.2. FIRMWARE

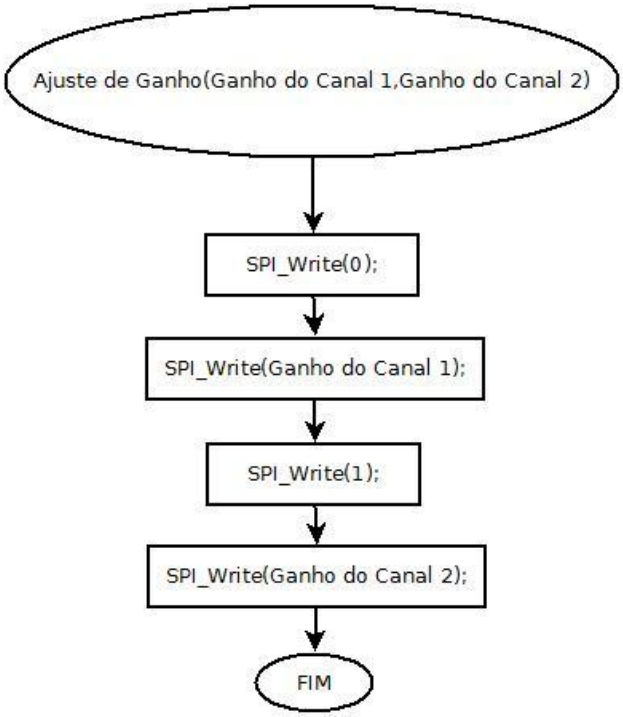
A função principal (*MAIN*) do *firmware* foi modificada para que opere em concordância com o *software* alto nível, ou seja, torna-se dependente do *software* para receber dados de atualização de taxa de amostragem, ganho dos canais e modo de operação. Este modo de operação informa se o USBScope 2.10 deve fazer

a Leitura do canal 1 ou do canal 2 ou de ambos no modo ALT. Após atualização destes parâmetros os dados convertidos são armazenados na memória externa e a seqüência transmitido para o microcomputador. O Fluxograma 5 apresenta esta operação.



Fluxograma 5: Função Main USBScope 2.10

A função de ajuste de ganhos dos amplificadores se apresenta de maneira simplificada, Fluxograma 6. Utilizando o protocolo SPI deve-se enviar dois bytes. O primeiro corresponde ao potenciômetro digital que será ajustado, valor 0 para o potenciômetro 1 e valor 1 para o potenciômetro 2. O segundo byte corresponde ao valor digital ajustado, valores entre 0 e 255, que correspondem a proporção do valor nominal do potenciômetro, 20 KΩ.

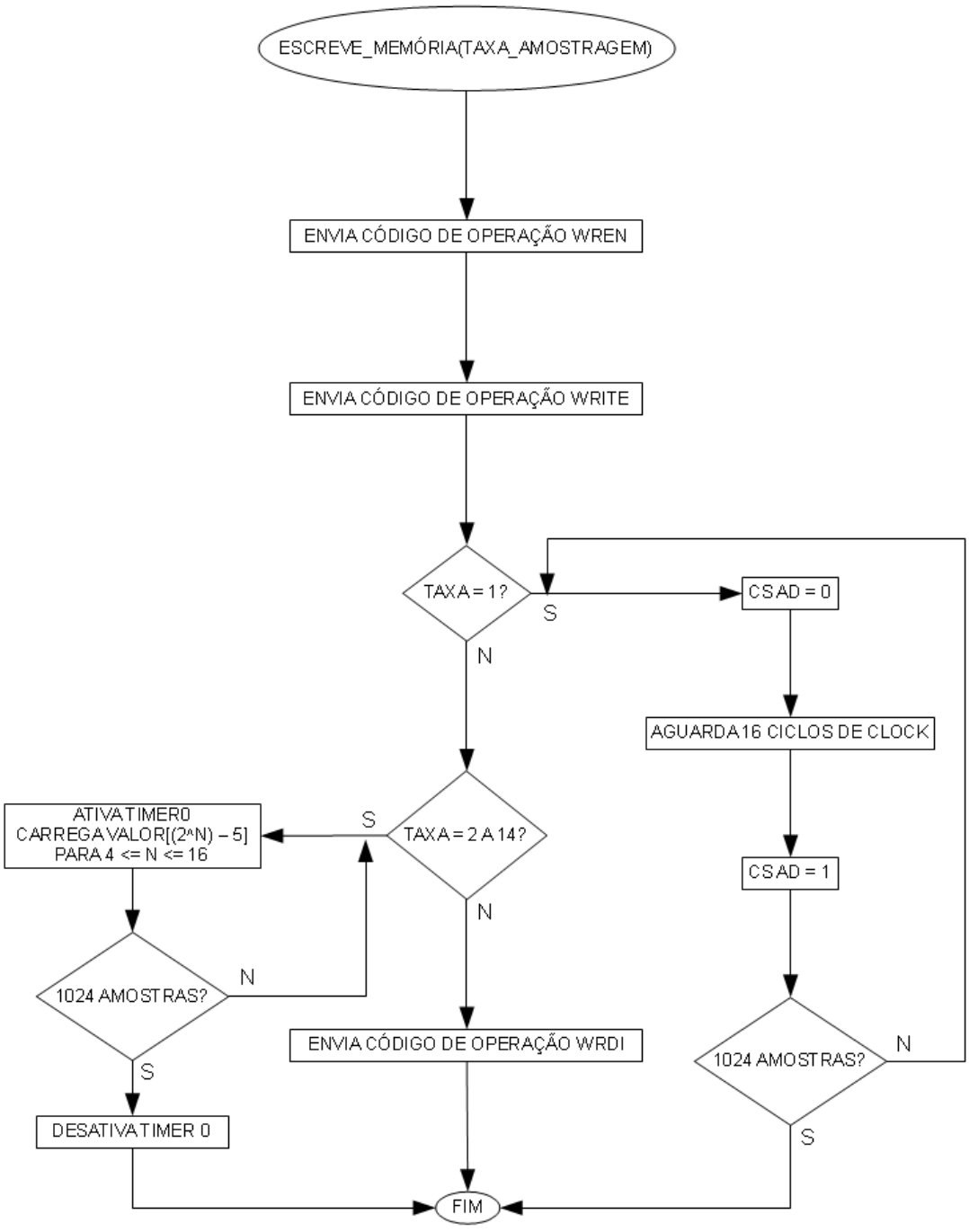


Fluxograma 6: Função Ajuste de Ganho dos Amplificadores

A variação da taxa de amostragem do sinal ocorre mudando o intervalo entre amostras armazenadas na memória na proporção do tempo de aquisição de uma amostra, como mostrado na equação 4.

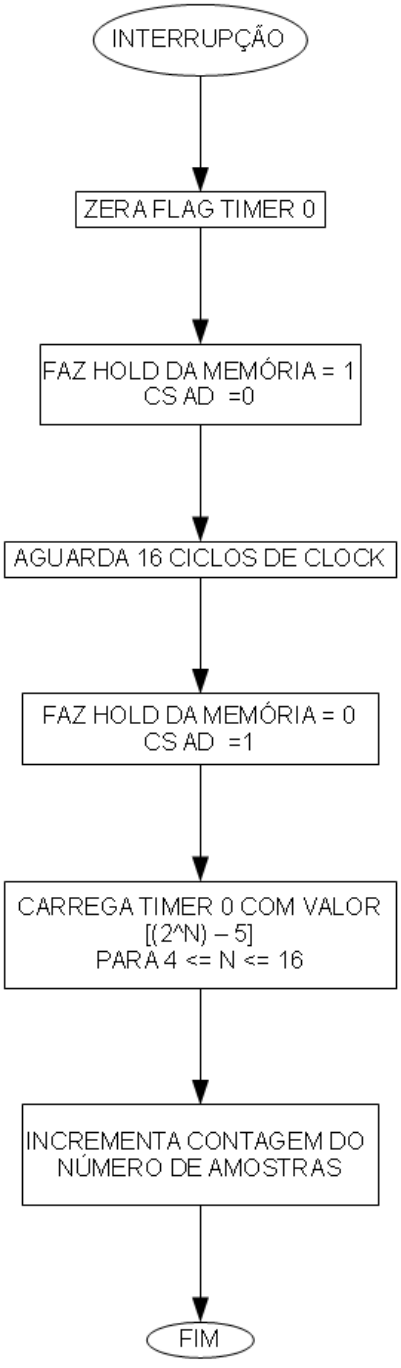
$$Tintervalo = 2^{y}\ x\ Tamostra \quad (4)$$

Onde y varia entre 0 e 16. Fazendo o uso do o *Timer* 0 do microcontrolador é possível carregar estes valores de 16 bits resultando em 14 variações de escalas de tempo. A máxima taxa de aquisição de dados permanece igual a função aplicada ao USBScope 2.0 e para as taxas que superiores, de 2 a 14, Fluxograma 7, faz-se necessário ajuste de um valor 5 correspondente aos *clocks* utilizados em comandos no inicio da função.



Fluxograma 7: Função de conversão de dados e escrita na memória

O estouro do *Timer* resulta em acionamento da interrupão, e a conversão de uma amostra e a contagem do número de amostras efetuadas é feita neste momento. O Fluxograma 8 apresenta o funcionamento desta operação.



Fluxograma 8: Interrupção *Timer* 0 para controle de conversão e armazenamento de dados na memória

A função responsável pela leitura e transferência de dados permanece a mesma apresentada no USBScope 2.0.

3.2.3. SOFTWARE JAVA

Como explicado anteriormente, o *software* do USBScope 2.10 possuirá duas janelas, uma para a visualização da onda, com algumas poucas funções para a manipulação do sinal, e outra para o controle dos parâmetros e visualização dos valores das funções matemáticas aplicadas no sinal. Foi pensado desta maneira para que o usuário pudesse ter mais liberdade com a janela que representa os pontos da onda, tendo mais espaço para redimensioná-la da maneira como desejar. A Figura 20 apresenta a tela de visualização do sinal.

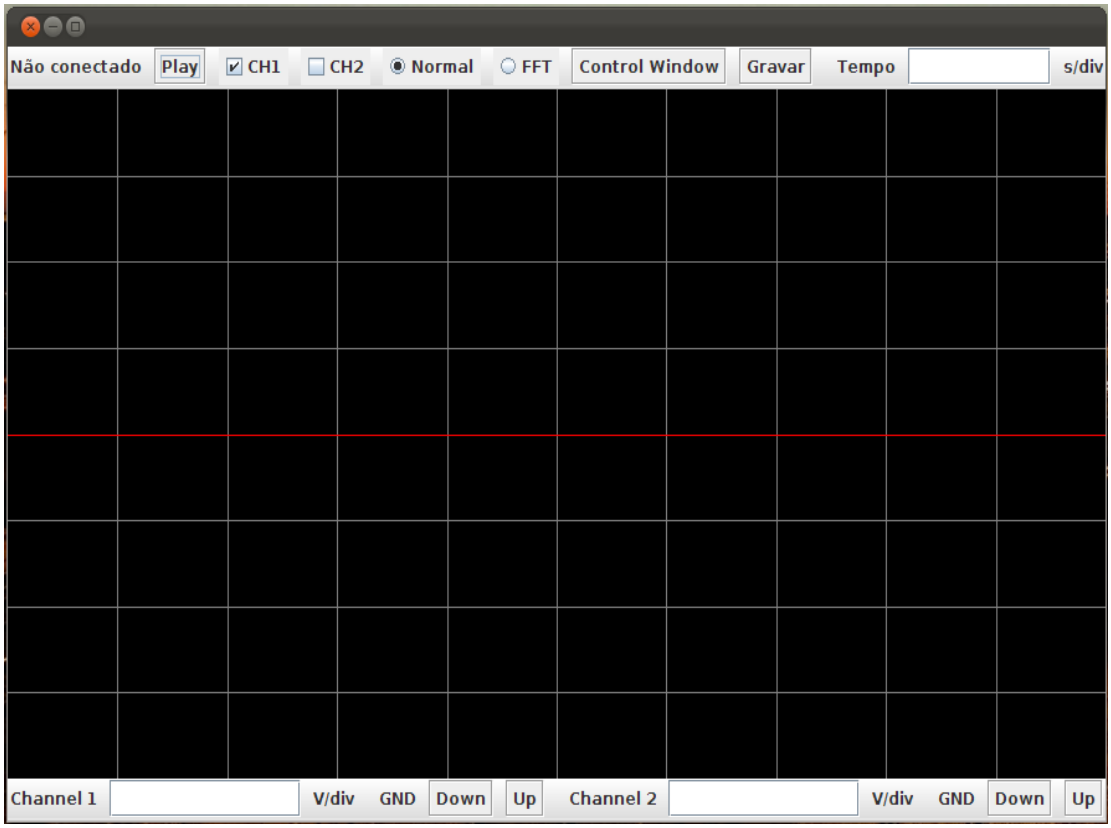


Figura 20: Interface gráfica USBScope 2.10

Esta janela possui uma tela para a visualização do sinal, dividida em 8 divisões verticais e 10 divisões horizontais, seguindo o padrão da maioria dos osciloscópios de bancada convencionais. Além disso, possui duas barras, para acesso a funções de visualização de valores e controle básico da onda, listados a seguir, separados pela barras e listados da esquerda para a direita.

- Barra superior
  - Rótulo indicativo de conexão;
  - Botão para interrompimento e reprodução da onda;

- Caixa de seleção para a visualização do sinal vindo do canal 1;
- Caixa de seleção para a visualização do sinal vindo do canal 2;
- Botão de seleção para visualização do sinal no domínio do tempo;
- Botão de seleção para visualização do sinal no domínio da frequência;
- Botão para abertura ou fechamento da janela auxiliar;
- Botão para a gravação dos pontos recebidos;
- Rótulo indicativo da escala de tempo atual do sinal;
- Caixa de edição indicativa com o valor da escala de tempo atual do sinal;
- Rótulo indicativo com a unidade de medida da escala de tempo.

- Barra inferior
  - Rótulo indicativo com o nome do canal 1, cujo valor de escala de tensão atual é apresentado;
  - Caixa de edição indicativa com o valor da escala de tensão atual para o canal 1;
  - Rótulo indicativo com a unidade de medida para a escala de tensão no canal 1;
  - Rótulo indicativo com o nome do GND (*ground*) respectivo ao canal 1;
  - Botões para o ajuste da posição vertical do sinal apresentado pelo canal 1, para baixo e para cima respectivamente;
  - Rótulo indicativo com o nome do canal 2, cujo valor de escala de tensão atual é apresentado;
  - Caixa de edição indicativa com o valor da escala de tensão atual para o canal 2;
  - Rótulo indicativo com a unidade de medida para a escala de tensão no canal 2;
  - Rótulo indicativo com o nome do GND (*ground*) respectivo ao canal 2;
  - Botões para o ajuste da posição vertical do sinal apresentado pelo canal 2, para baixo e para cima respectivamente.

Como é possível perceber a tela de visualização do sinal possui indicativos básicos de informação dos canais, para um melhor foco na parte visual do sinal, além de possuir acesso à janela secundária, para o controle dos canais e do trigger, como visto na Figura 21.

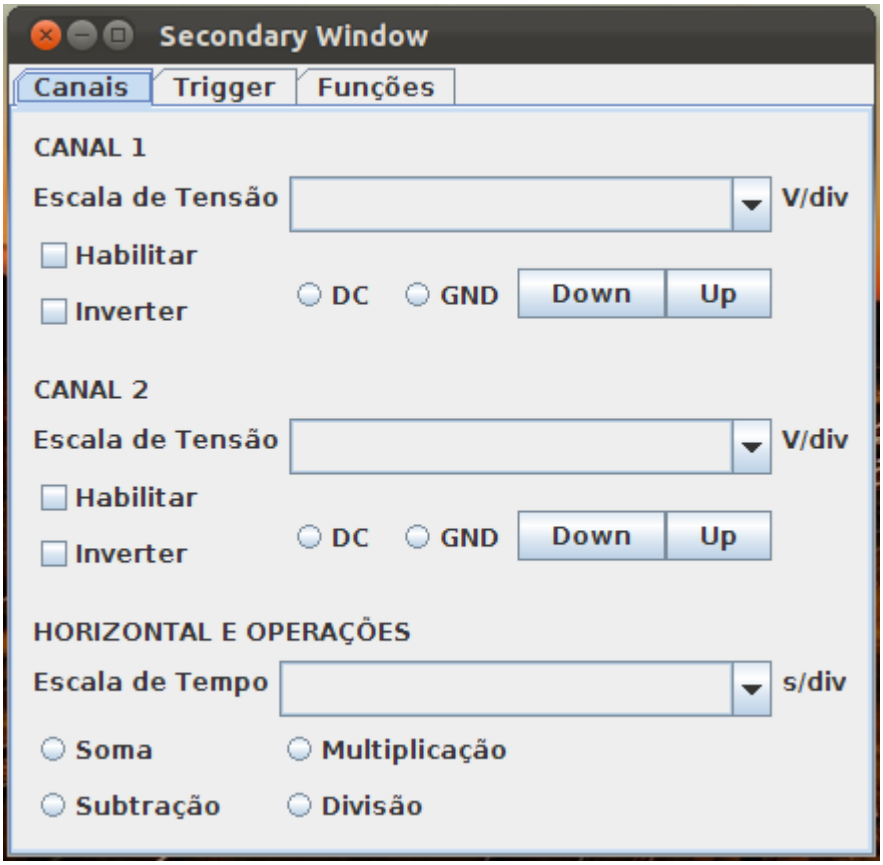


Figura 21: Janela de controle de Canais

Esta janela é dividida em abas, para um melhor aproveitamento do espaço e organização dos controles em si. Na primeira aba mostrada, é possível fazer o controle do ganho nos canais 1 e 2, além de poder inverter a visualização da onda e habilitar e desabilitar a leitura dos canais. Esta aba também possui acesso à definição da escala de tempo e operações com os sinais amostrados, soma, subtração, multiplicação e divisão.

Figura 22: Janela de controle de *Trigger*

Na Figura 22, é apresentada a segunda aba responsável pelo controle do *trigger* do sinal, no qual se procurou seguir o padrão dos osciloscópios convencionais, com a definição do nível, escolha da fonte entre os canais 1 e 2, modo de operação, automático ou normal, escolha da borda de subida ou descida, e habilitação e desabilitação do *trigger*.

Na terceira aba, Figura 23, é possível ver o resultado dos cálculos matemáticos aplicados ao sinal de entrada com os valores médio, RMS, pico e pico a pico dos canais 1 e 2.

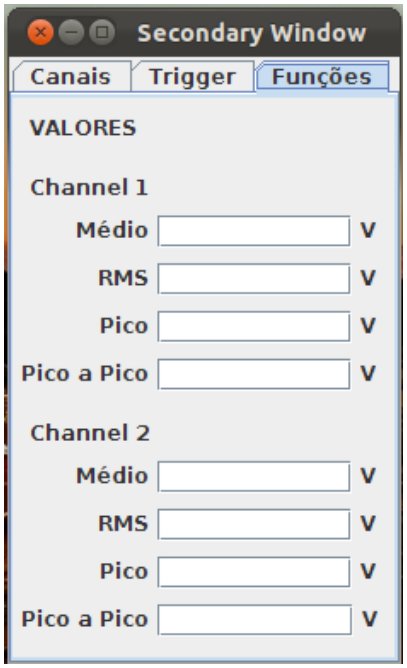


Figura 23: Janela de controle Funções

3.2.4. ESTRUTURA DO CÓDIGO

O código do *software* USBScope 2.10 foi montado de modo a seguir da maneira mais próxima possível a arquitetura de três camadas MVC (*Model*, *View*, *Controler*), porém, substituindo a parte de banco de dados pela parte de comunicação USB/Serial.

A estrutura do código ficou dividida basicamente em 4 classes principais, que participam efetivamente da comunicação com o *hardware* e interação com o usuário e 2 classes secundárias, que fornecem dados para as classes principais e executam cálculos diversos como valores médio e eficaz:

- Classes Principais:
  - *MainControl*: Responsável por todo o tratamento de código do programa, realizando um gerenciamento completo de todos os dados que entram e saem do *software*, bem como de todas as *threads* e sincronismo entre elas.
  - MainWindow: Janela de visualização do programa, sendo responsável apenas pela plotagem e passagem de eventos visuais para a classe *MainControl*;
  - SecondaryWindow: Janela responsável pela visualização das funções apicadas aos sinais de entrada, e passagem de eventos visuais para a classe *MainControl*;
  - Serial: Responsável pela comunicação com o *hardware*, entrada e saída de dados, e preenchimento do vetor de dados, que é utilizado na plotagem e na aplicação das funções matemáticas.
- Classes Secundárias
  - Channel: Responsável pelo recebimento, tratamento, e fornecimento dasalterações de dados feitas pelo usuário, como por exemplo, uma alteração nas escalas de tensão ou tempo;
  - *Trigger*: Responsável pelo recebimento, tratamento, e fornecimento das alterações de dados feitas pelo usuário, com respeito a visualização da onda.

Nenhuma das classes apresentadas possui relação interdependente de herança ou associação, de modo que é preferível se mostrar um diagrama de

objetos, no momento de uso de todos os recursos do sistema, em vez do diagrama de classes tradicional, que mostra a relação de dependência entre as classes. A Figura 24 mostra apenas a relação entre os objetos, não mostrando os métodos que cada um possui.

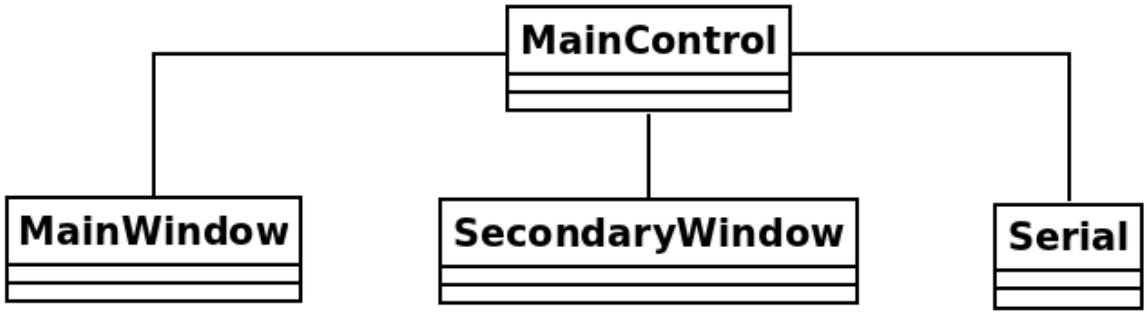
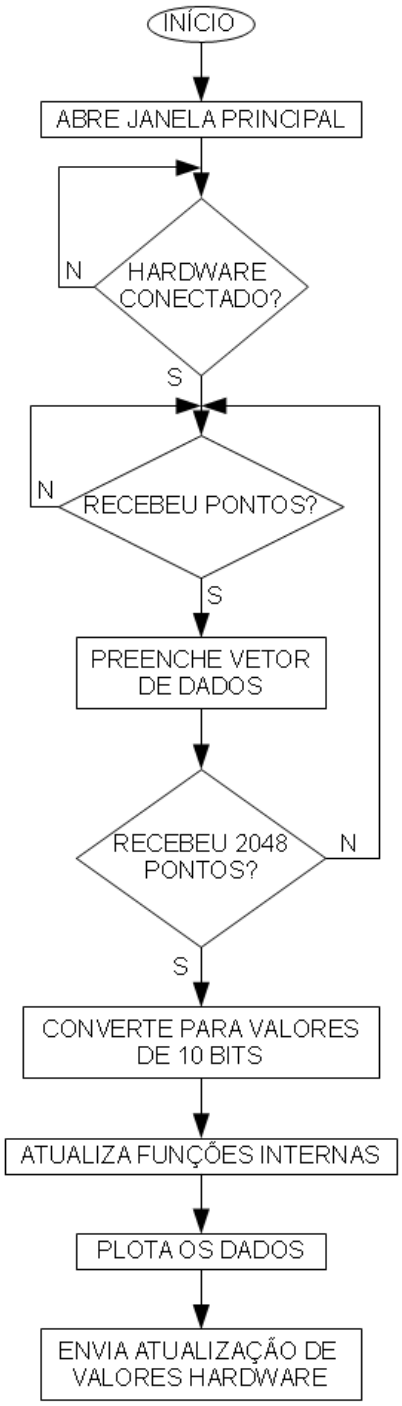


Figura 24: Diagrama de Objetos Básico

A classe *MainControl* possui instâncias de todas as outras classes do sistema, para um melhor controle e gerenciamento dos processos concorrentes. De modo que tudo o que acontece, necessitando de processamento de informação, passa necessariamente por ela, para que ela possa decidir o que mostrar ao usuário e o que deve ser enviado para o *hardware*.

3.2.5. FUNCIONAMENTO DO CÓDIGO

Além do paradigma da orientação a objeto, o estilo de programação concorrente também se faz presente no código do USBScope 2.10. O programa funciona com um processo principal que é executado desde a abertura do programa até o seu fechamento, representado pelo Fluxograma 9.



Fluxograma 9: Processo principal do software

As funções de '*Hardware* Conectado' e 'Preenche Vetor de Dados' funcionam em outra *thread*, para que não ocorra nenhum efeito de congelamento na tela enquanto ela está aberta.

3.2.6. ALGORÍTIMOS IMPORTANTES

Os algoritmos para os cálculos das diversas funções que o *software* disponibiliza foram implementados da maneira mais simples possível, para evitar o uso exagerado de recursos do sistema.

O algoritmo de sincronização teve duas implementações no decorrer do projeto. Na primeira são recolhidos 10% dos pontos anteriormente plotados, e os compara com 10% de pontos de trechos seqüenciais da onda recém recolhida, para ver se há coincidências de valores, se houver, o código mescla a parte inicial dos pontos antigos com a seguinte dos pontos novos, de modo que o usuário não tenha a impressão que os pontos plotados “caminham” na tela. Se não houver coincidência de informações, a onda nova inteira é plotada. Porém esta primeira implementação se mostrou ineficaz, com um código muito complexo e relativamente lento, por isso, utilizou-se um algoritmo de correlação cruzada.

A correlação cruzada mede o grau de semelhança entre duas funções [http://pt.edaboard.com/topic-337606.0.html], consistindo basicamente no deslocamento de uma função sobre a outra e na multiplicação dos pontos dessas duas funções no tempo, registrando o maior valor encontrado. Este valor é o maior nível de semelhança entre os dois sinais, o que é particularmente importante neste caso de sincronização, pois se a onda não for alterada pelo usuário, ou não sofrer com um ruído muito grande, as duas serão apenas defasadas e o grau de semelhança se a defasagem for zero.

O *trigger* tem seus valores considerados apenas nos algoritmos de plotagem, pois nada de seu cálculo é realizado no *hardware*. Os parâmetros do *trigger* são calculados e ficam à espera de uso ou alteração pelas funções de plotagem, que vão alinhar e recalcular seus pontos de acordo com os valores do *trigger* atuais.

A FFT foi retirada de [12], consistindo em duas classes, uma que representa os números complexos e outra com o cálculo da FFT propriamente dito, a partir de um vetor de números reais. Estes pontos são então colocados em escala e plotados na tela. Ambas as classes foram utilizadas como classes de cálculos internas a classe *MainControl*.

4. RESULTADOS

4.1. USBSCOPE 2.0

Com o desenvolvimento da versão 2.0 foi possível obter dados que demonstram o progresso do projeto.

Inicialmente constatado o reconhecimento da conexão USB do dispositivo com o microcomputador pode-se aplicar o *firmware* de controle de conversão, armazenamento e transferência de dados. 1024 bytes foram apresentados pelo terminal HID presente no compilador MikroC Pro, Figura 26. O dado transmitido para o terminal corresponde a conversão da tensão analógica 0 volts.

A Figura 25 apresenta uma amostra da tensão convertida capturada pelo osciloscópio Lecroy LT584, como se trata de uma tensão constante os dados subseqüentes possuem o mesmo valor desta primeira amostra. O canal de linha azul corresponde ao pino Chip Select do AD, o canal de linha verde corresponde ao *clock* serial, o canal de linha amarela corresponde ao dado digital do sinal convertido e o canal de linha rosa o Chip Select da memória.

Convertendo os valores binários para decimal do dado digital obtém-se 48 para o primeiro byte e 0 para o segundo, comprovando a sequência de dados recebidos pelo terminal.

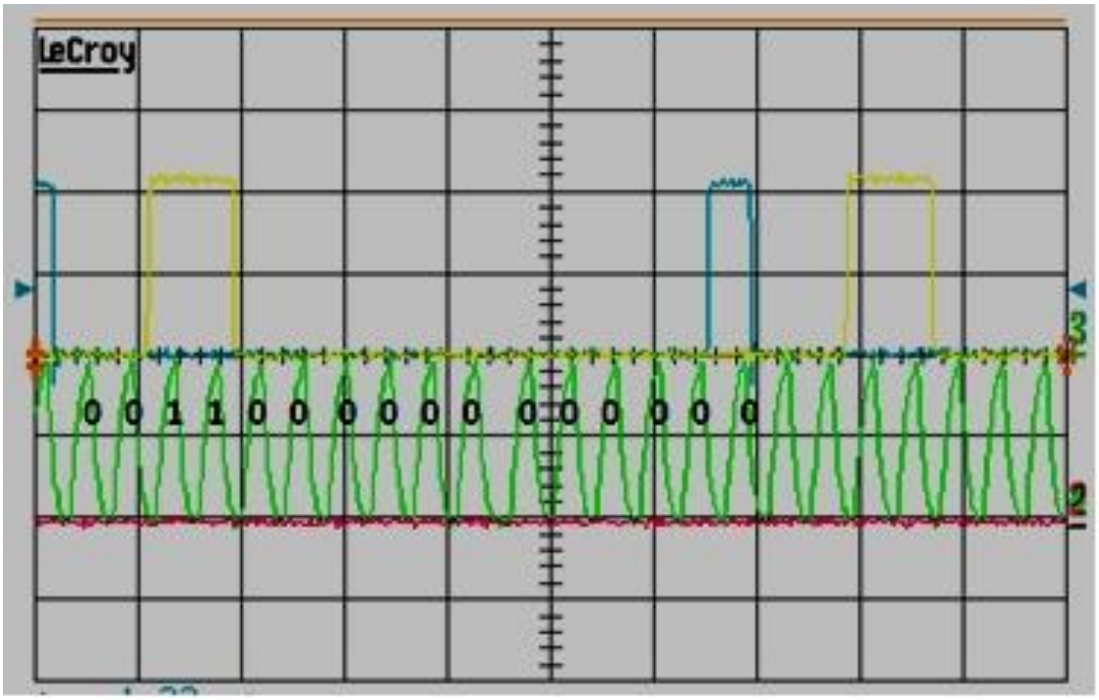


Figura 25: Amostra do sinal fornecida pelo AD

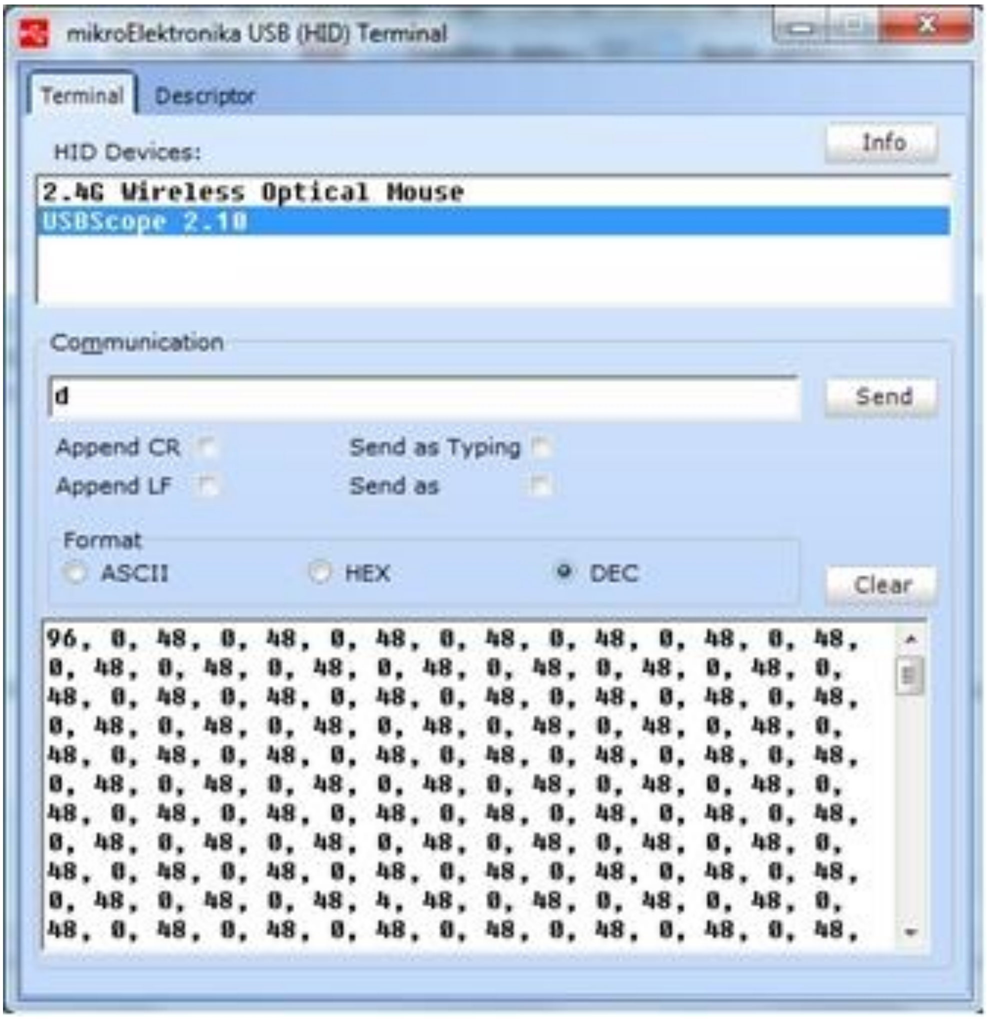


Figura 26: Dados transmitidos para o terminal HID

O estágio de amplificadores apresentou mau funcionamento e para dar sequencia ao desenvolvimento e testes foi aplicada uma tensão senoidal de 10 KHz e amplitude máxima de 3,75 V no canal 1 do conversor AD. Utilizando o procedimento de captura de dados pelo terminal HID foi possível plotá-los no programa gerador de gráficos QTIplot, Figura 27.

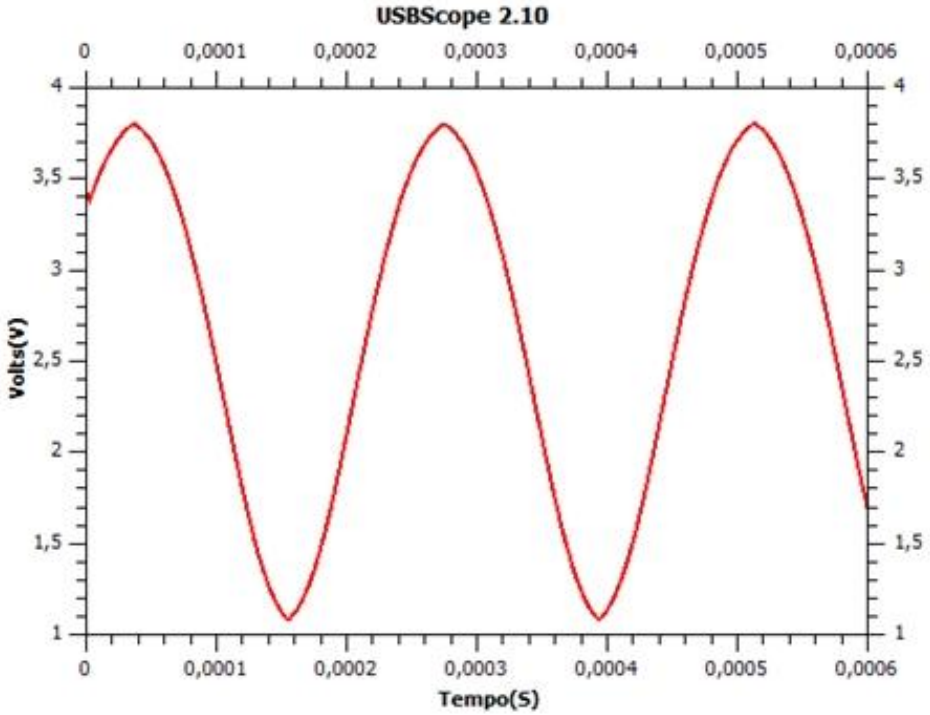


Figura 27: Sinal senoidal de um gerador de funções, adquirido pelo USBScope 2.0

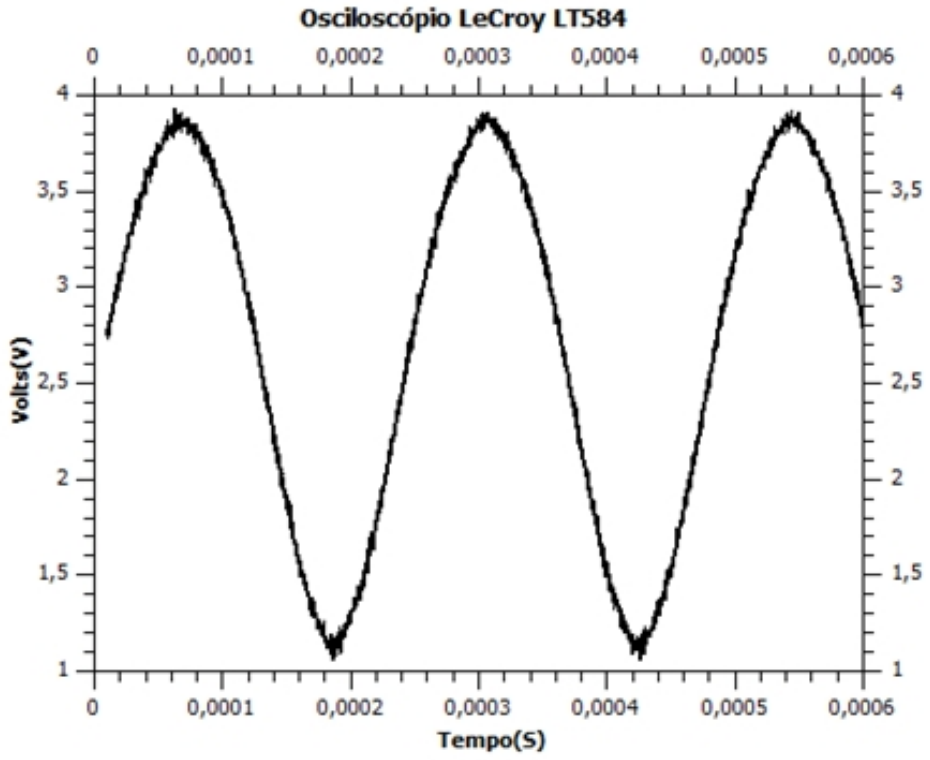


Figura 28: Sinal senoidal de um gerador de funções, adquirido pelo osciloscópio Lecroy LT584

Podemos destacar a melhora na relação sinal ruído apresentado pelo USBScope 2.0 em relação ao apresentado pelo Osciloscópio Lecroy LT584, Figura

28. Isso se deve por dois motivos. Primeiro porque osciloscópios de bancada utilizam conversores AD com resolução de 8 bits, enquanto o USBScope utiliza resolução de 10 bits e em segundo lugar a banda passante do USBScope (500 KHz) é muito inferior a do osciloscópio Lecroy (20 MHz), fazendo com que o conversor AD seja um filtro para altas freqüências.

O mesmo sinal aplicado no canal 1 do AD, Figura 23, é representado pelo *software* escrito em *Visual Basic*, Figura 29.

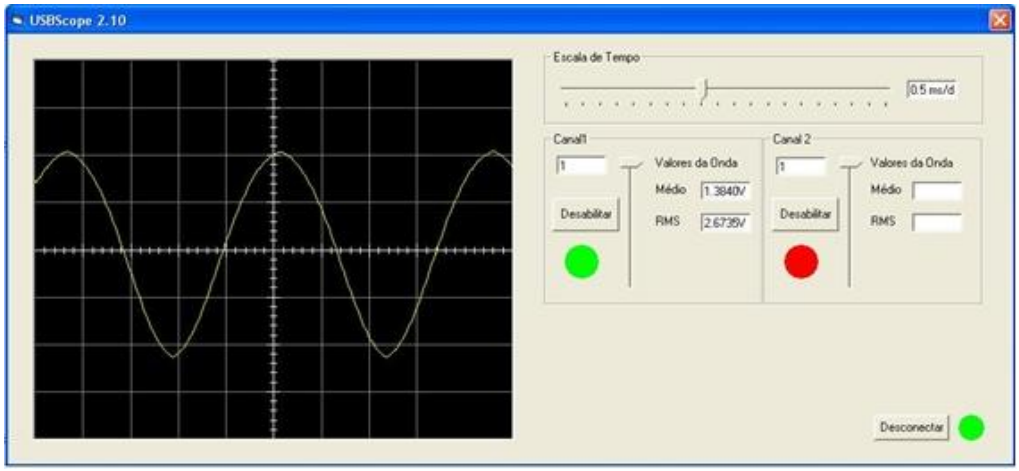


Figura 29: Interface gráfica USBScope 2.0

4.2. USBSCOPE 2.10

Após a definição do circuito da versão 2.10 e através do seu esquema foi possível desenvolver o layout da PCI com componentes SMD, Figura 30. Esta placa foi fabricada por uma empresa contratada devido as reduzidas dimensões de trilhas, vias e *pads* dos componentes.

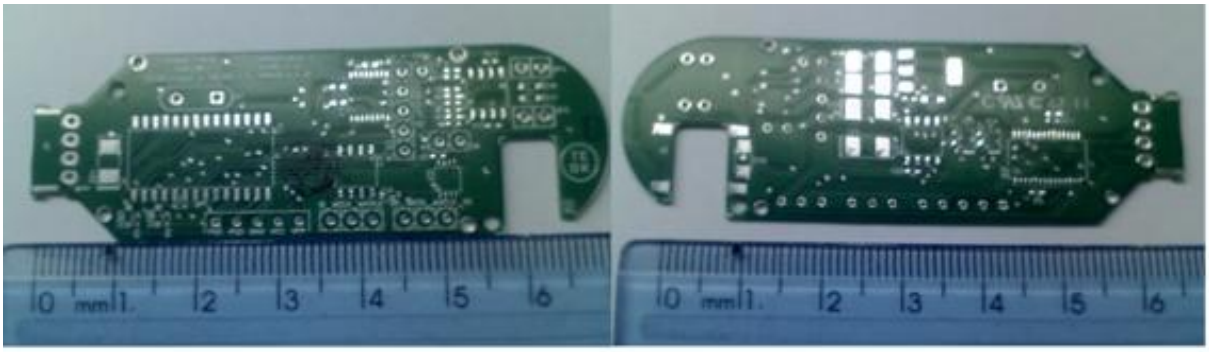


Figura 30: Placa de circuito impresso

Inserindo externamente o FT232 no circuito 2.0 foi possível dar inicio ao processo de desenvolvimento do *firmware* e *software* da versão 2.10. Um programa sem interface visual foi desenvolvido em Java para testar a comunicação serial emulada em USB aplicado ao Linux Ubuntu 11.10.

O *firmware* apresentado para a versão 2.0 foi modificado para envio de dados via comunicação UART. A transmissão foi comprovada através do *software* imprimindo no monitor os valores recebidos em formato decimal.

Ainda modificando o mesmo *firmware* foi possível desenvolver a rotina de variação da taxa de amostragem do sinal. Utilizando o contador *timer* 0 foi possível criar 11 variações da taxa de amostragem.

Testes efetuados com o novo *firmware* e circuito USBScope 2.10 apresentaram problemas de comunicação que necessitaram adaptações no *hardware*. A primeira adaptação feita se refere ao *clock* de referência de 12 MHz que deve ser fornecido pelo microcontrolador. Outra adaptação feita foi sobre os pinos de comunicação USART, pois o pino RX de comunicação serial é o mesmo pino SDO de comunicação SPI, o que impossibilita a utilização dos dois protocolos de comunicação pelo mesmo pino. Esta modificação foi possível devido a biblioteca *Software*_UART presente no compilador mikroC que permite escolher outros pinos para estabelecer comunicação serial emulada.

A Figura 31 apresenta a placa de aquisição de dados do USBScope 2.10 com seus componentes soldados e o formato final inserido no invólucro de um mp3 *player.*

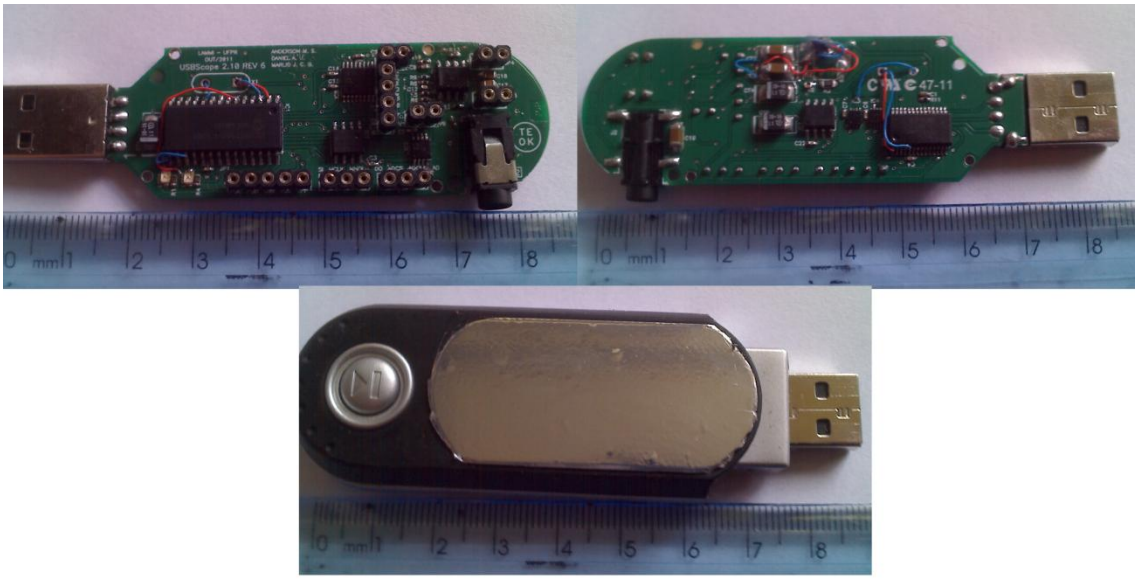


Figura 31: Placa de aquisição de dados USBScope 2.10

Outra ferramenta auxiliar presente no compilador mikroc é o terminal UART. Através deste terminal foi possível conectar o circuito ao computador e coletar dados de um sinal senoidal aplicada a entrada analógica do circuito comprovando seu funcionamento. Os dados capturados foram inseridos em um programa desenvolvido no Matlab para que seja possível visualizar o sinal capturado, Figura 32.

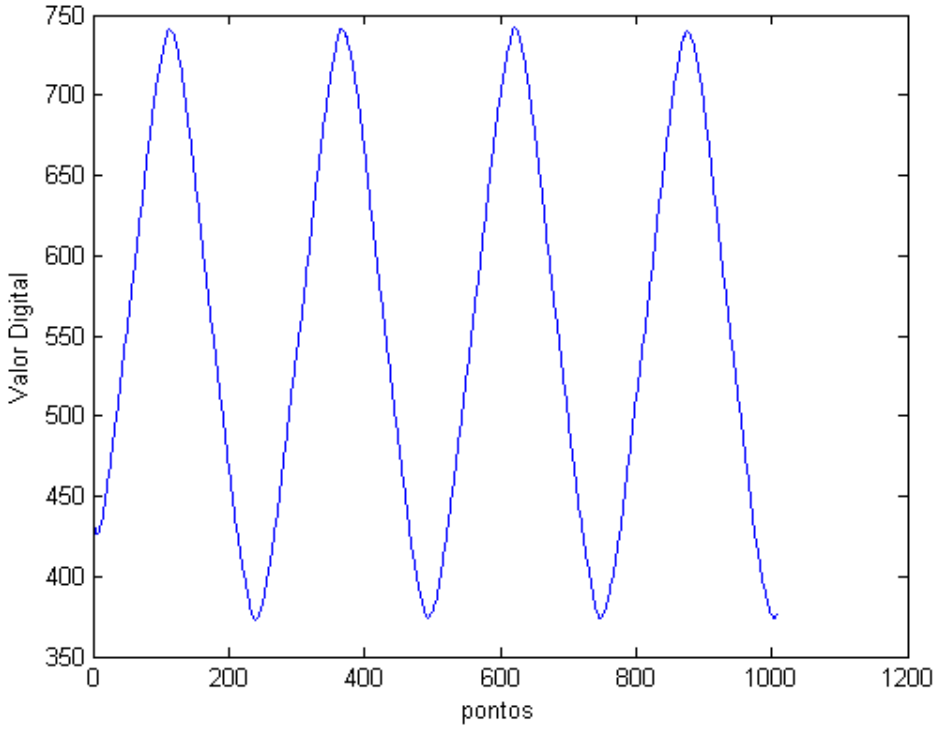


Figura 32: Sinal senoidal de um gerador de funções capturados pelo USBScope 2.10

Para gerar este gráfico foram utilizados os valores digitais de 1024 pontos capturados do canal 1.

Testes da escala de amplitude permitiram coletar os dados apresentados na Tabela 8, onde o valor digital é o valor encaminhado aos potenciômetros digitais. Este teste foi efetuado enviando dados para o circuito através do terminal UART e relacionando o sinal de saída de cada amplificador com o sinal de entrada, ambos os sinais capturados pelo osciloscópio Lecroy LT584.

Valor Digital	Escala (V/div)	G(V/V)
255	8,3	0,06
200	5	0,1
29	1	0,5
12	0,5	1
5	0,25	2
3	0,2	2,6
2	0,15	3,4
1	0,1	4,7

Tabela 8: Ganho dos amplificadores

O amplificador operacional inicialmente utilizado, LM6172, apresentou instabilidade devido a sua alta sensibilidade. Na tentativa se resolver o problema algumas mudanças no *hardware* foram efetuadas sem sucesso e optou-se pela substituição deste amplificador pelo modelo TL062 para dar continuidade ao desenvolvimento.

O sinal de 1 KHz e 1,9 Vpp aplicado a entrada do circuito comprova o funcionamento do *software,* seleção dos canais, ganho e taxa de amostragem. A Figura 33 apresenta este sinal aplicado ao canal 1, com ganho de 1 V/V e escala de amostragem de 0,4 mSeg/div e a Figura 34 apresenta este mesmo sinal, aplicado ao canal 2, com ganho 1,4 V/V e escala de amostragem de 2,3 mSeg/div.

O software permite a seleção das escalas de tensão de cada canal individualmente, conforme Tabela 8, a seleção de 11 escalas de tempo e a seleção dos modos leitura do canal1, leitura do canal 2 ou ALT, onde os dois canais são selecionados. As funções de cálculo de valores médio, RMS, pico e pico-a-pico foram alcançadas, demonstrando em tempo real para o usuário das mudanças sofridas pela onda.

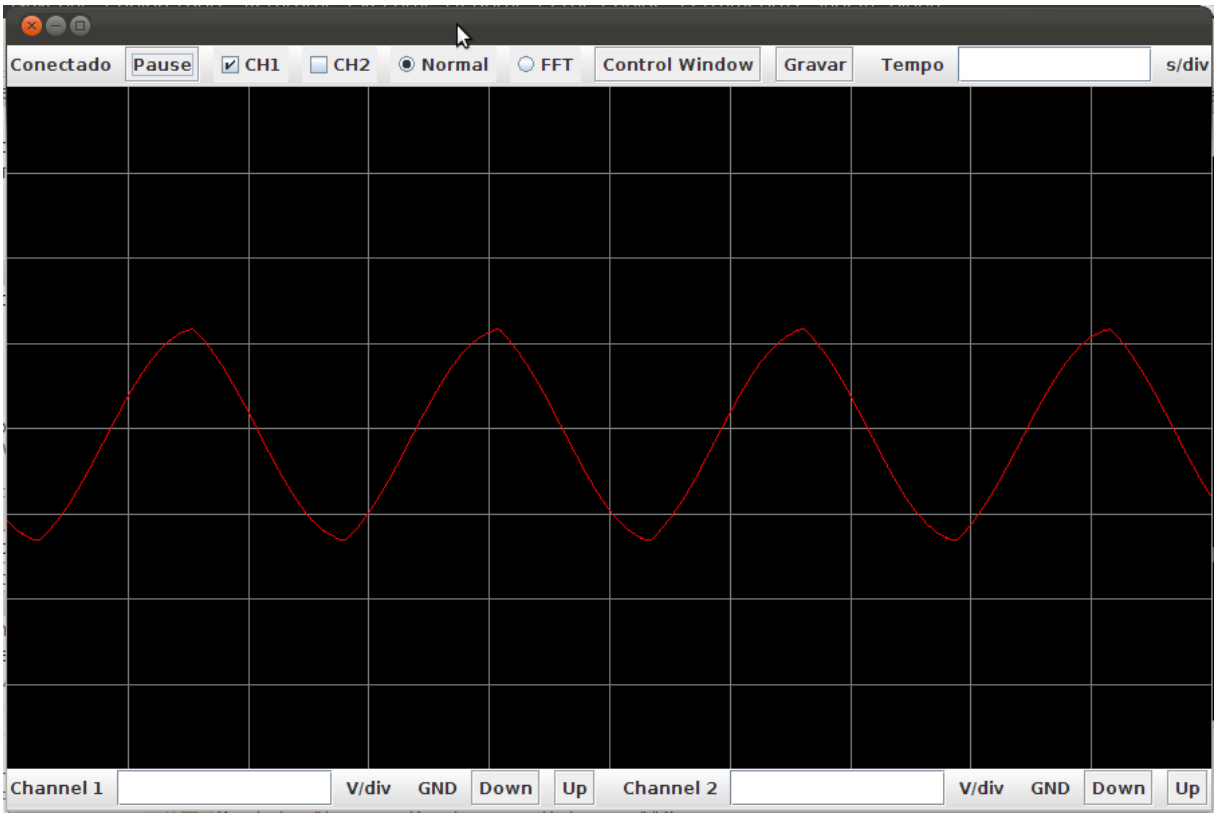


Figura 33: Sinal do canal 1 representado pelo *software*

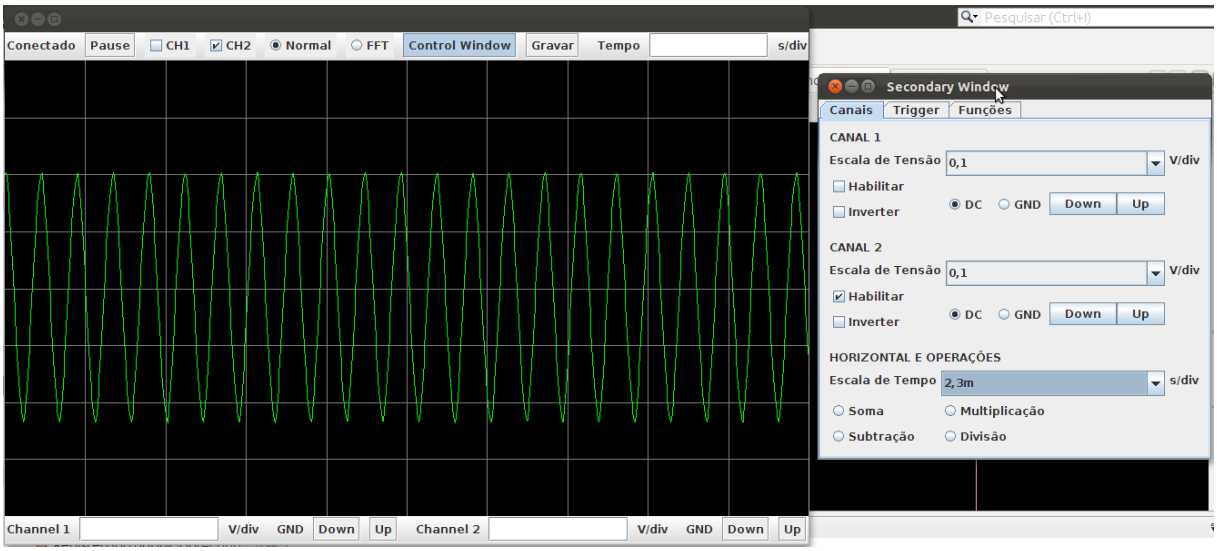


Figura 34: Sinal do canal 2 representado pelo *software*

5. CONCLUSÕES

Apesar de muitos problemas terem sido detectados no decorrer do desenvolvimento, pode-se perceber com este trabalho que padrões de comunicação serial necessitam elevado grau de sincronismo. A aplicação do projeto em sistemas multiplataforma apresenta resultado satisfatório com a utilização do chip FT232 que além de *drivers* apresenta *clock* estável fornecido ao circuito.

Variando a freqüência do sinal de entrada foi possível determinar a freqüência de corte para a máxima escala de tensão (8,3 V/div), menor ganho, obtendo aproximadamente 400 KHz e a freqüência de corte para a menor escala de tensão (0,1 V/div), maior ganho, em aproximadamente 6 KHz. Com estes dados pode-se concluir que circuito analógico deve ser simulado para se obter informações pertinentes a configuração escolhida e estabelecer mudanças que possibilitem melhoria na freqüência de corte.

Os ganhos dos amplificadores podem ser calibrados para a obtenção de valores arredondados, facilitando a conversão do sinal apresentado na tela. Com a melhoria no estágio de amplificadores operacionais e levando em conta a taxa de Nyquist e a interpolação linear efetuada pelo *software* o USBScope 2.10 poderá ser capaz de efetuar a leitura de sinais de freqüência máxima igual a 200 KHz. Com o ganho mínimo de 0,06 V/V é possível obter leituras de sinais com amplitude máxima de 67 Vpp. Esta tensão analisada pode ser superior caso seja utilizada uma ponta de prova com atenuação de 10x.

Mais escalas de tempo podem ser incluídas com a utilização do *prescaler* do *Timer* do microcontrolador.

O *software* apresentou um desempenho regular e suficientemente rápido para não causar efeitos de congelamento na tela, com atualizações de tela de 310 milissegundos, o que não prejudicou a visualização por parte do usuário.

A folha de dados, apresentado na Tabela 9 demonstra outras características importantes atingidas, no formato de folha de dados de um osciloscópio digital comercial.

DISPLAY	Colorido
Largura de Banda	400 KHz
Canais	2
Entrada de Trigger Externa	Não
Taxa de Amostragem por canal	750 KSPS
Resolução Vertical	10 bits
Máxima Tensão de Entrada	67 Vpp
Escalas de tensão	100 mV/div a 8,3 V/div
BW Limite	6 KHz em 0,1 V/div
Acoplamento de Entrada	DC
Impedância de Entrada	100 KΩ em paralelo com 5 pF
Escala da Base de Tempo	1,33 us a 0,28 seg/d
Pontos na tela	1000 pontos
Modos de operação	Canal 1, Canal 2, ALT
Comunicação	USB
Sistema Operacional	Ubuntu 11.10
Dimensões	95x35x20 cm

Tabela 9: Características USBScope 2.10

O custo previsto para a reprodução deste equipamento, levando em conta preço de componentes eletrônicos e fabricação da PCI, é de aproximadamente R$200,00.

A Tabela 10 apresenta uma breve comparação, das principais melhorias, entre a versão 1.1 e a 2.10 do projeto USBScope.

	USBScope 1.1	**USBScope 2.10**
Taxa de Amostragem	60 KSPS	750 KSPS
Escalas de Tensão	8	8
Escalas de Tempo	7	11
Máxima Tensão de Entrada	50 V pico	67 V pico-a-pico
Comunicação USB	Sim	Sim
Sistema Operacional	Windows	Linux

Tabela 10: Comparação USBScope 1.1 e USBScope 2.10

6. POSSIBILIDADES DE MELHORIA

Durante o desenvolvimento deste projeto foi possível estudar as ferramentas aplicadas e recomendações podem ser feitas para uma futura melhoria.

A utilização de um microcontrolador com freqüência de trabalho superior ao microcontrolador escolhido pode facilitar a aplicação, pois como foi notado, é interessante que o dispositivo de controle possua velocidade muito superior a dos periféricos (memória e conversor AD), permitindo a fácil operação do controle de conversões e armazenamento de dados, não necessitando da utilização de programação mista para ajuste de sincronismo.

Melhorias no circuito analógico, como a substituição do amplificador operacional, proporcionarão aumento na banda passante nas escalas de tensão menores.

A padronização das escalas de tensão proporcionará ao usuário maior facilidade na conversão de valores do sinal apresentado na tela.

O *firmware* pode ser aprimorado aplicando o conceito de interrupção para o recebimento de dados do microcomputador, melhorando desta maneira o sincronismo com o *software*.

O *software* pode ter seu desempenho melhorado, realizando um estudo mais aprofundado da parte de *threads* para a otimização do código, realizando mais operações em paralelo. Além de poder ter um melhor uso da orientação a objeto e do modelo MVC, utilizando, talvez, classes-mãe abstratas para as classes visuais que representam as janelas, mantendo propriedades e métodos comuns destas classes em uma classe genérica, e fazendo com que cada classe visual tenha uma classe correspondente para tratamento de dados, e não apenas uma classe para todo o tratamento de dados do *software*, como foi feito nesta versão.

Terminar a implementação das funções de FFT e *trigger*, seria necessário para um funcionamento completo do *software*, visto que devido ao grande número de mudanças sofridas no software no andamento do projeto e ao curto espaço de

tempo na conclusão do mesmo, estas funções não puderam ser devidamente terminadas e testadas.

7. REFERÊNCIAS BIBLIOGRÁFICAS

[1] ESPÍNDOLA, Walter Luciano; PIECARZ, Leandro Silva. USBSCOPE. Trabalho de conclusão de curso (Graduação em Engenharia Elétrica). UFPR, Curitiba. 2007.

[2] Princípio de Digitalização de Sinal Analógico. Disponível em 20/01/2012 no site https://docs.google.com/viewer?a=v&q=cache:ZL7T8LrSk5IJ:www.lncc.br/~jauvane/SMM/2002/SMM-A02.pdf+taxa+de+nyquist&hl=pt-BR&gl=br&pid=bl&srcid=ADGEESj66pu1qDX-Qzu1pBY_YAy1QYN8V1EnxytsQ81HXBS7Asws3WA0NrLTYn0zC7fxRidMEZMjg8VK89xzP_veGpS-qIxdsz_iAvRO46ZIywLMpzxC1Ce89Us2V3shuWTh-RqxzIRJ&sig=AHIEtbRLjNSbkj54bjeEfPOuFCQ50d1ZJg

[3]Aumentando a taxa de Amostragem por um Fator Inteiro . Disponível em 20/01/2012 no site http://pt.scribd.com/doc/44355911/15/II-8-2-Aumentando-a-taxa-de-Amostragem-por-um-Fator-Inteiro

PC XY USB Oscilloscope Disco e informações. Disponível em 30/11/2011 no site http://www.hobbylab.us

[4] Parallax USB Oscilloscope e informações. Disponível em 30/11/2011 no site http://www.parallax.com/StoreSearchResults/tabid/768/txtSearch/oscilloscope/List/0/SortField/4/ProductID/46/Default.aspx

[5] PoScope basic2 bundle e informações. Disponível em 30/11/2011 no site http://www.poscope.com/product.php?pid=13

[6] PS40M10 e informações. Disponível em 30/11/2011 no site http://www.easysync-ltd.com/product/598/ps40m10.html

[7] TIOBE Programming Community e informações. Disponível em 30/11/2011 no site http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[8] IBRAHIN, Dogan. Advanced Pic Microcontroller Projects in C. 544 pp. Elsevier, 2008

[9] Microcontroladores e informações. Disponível em 30/11/2011 no site http://ww1.microchip.com/downloads/en/DeviceDoc/39632e.pdf

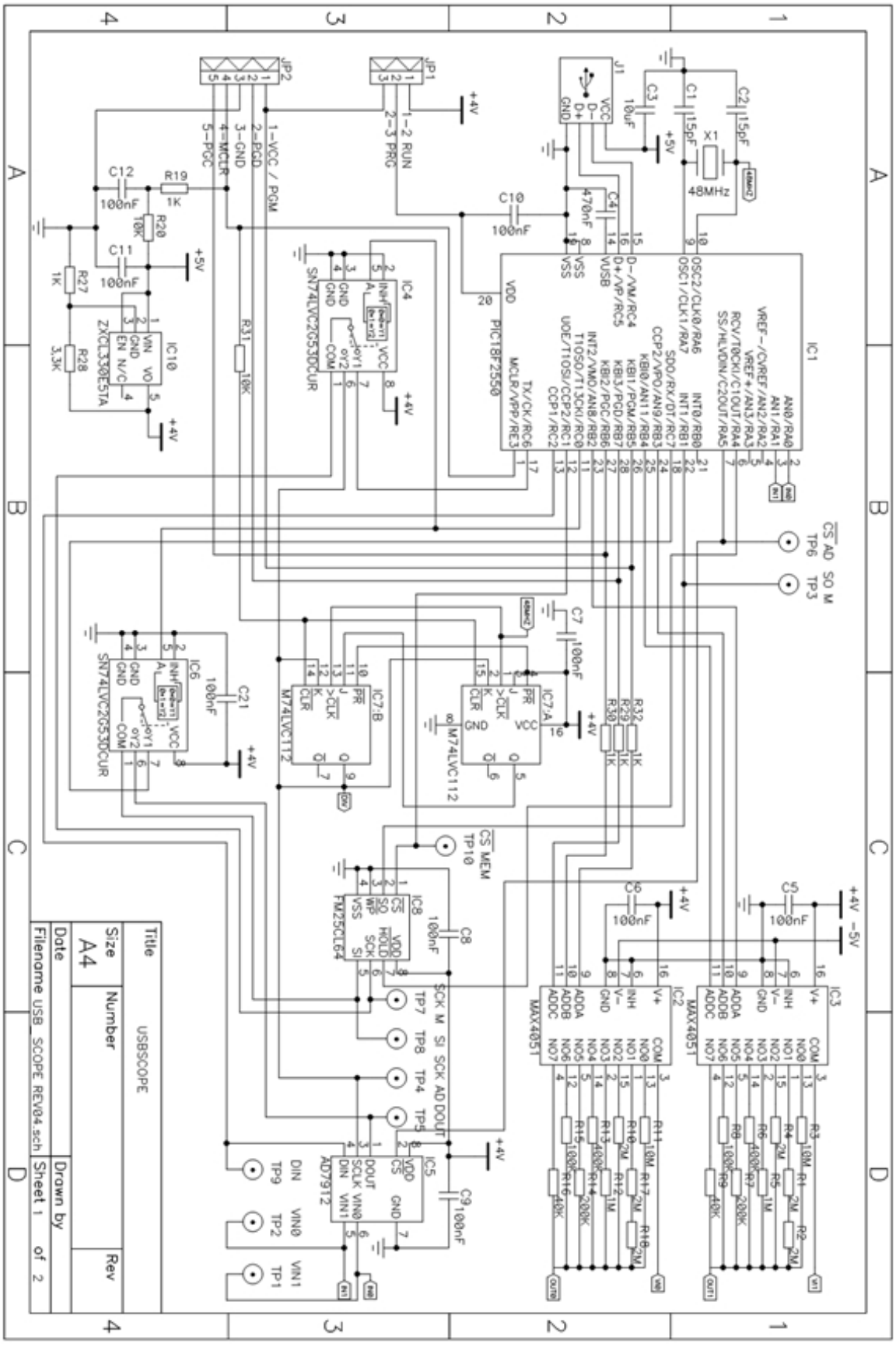
[10] Memória e informações. Disponível em 30/11/2011 no site http://www.ramtron.com/files/datasheets/FM25CL64_ds.pdf

[11] Componentes semicondutores e informações. Disponível em 30/11/2011 no site http://www.analog.com/static/imported-files/data_sheets/AD7912_7922.pdf

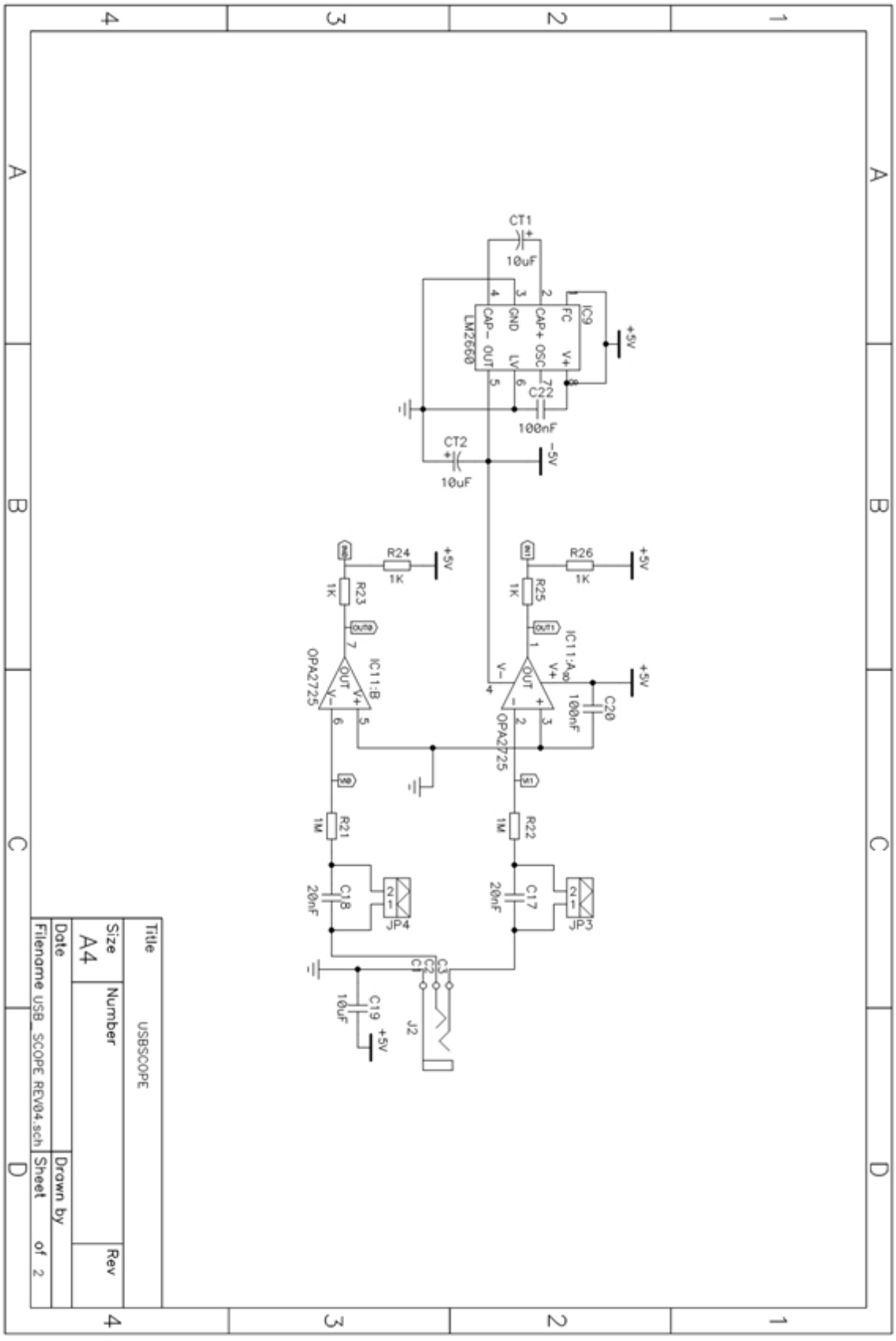
[12] Componentes e semicondutores. Disponível em 30/11/2011 no site http://focus.ti.com/lit/ds/symlink/sn74lvc2g53.pdf

[13] FFT. Java. Disponivel em 20/01/2012 no site introcs.cs.princeton.edu/97data/FFT.java.html

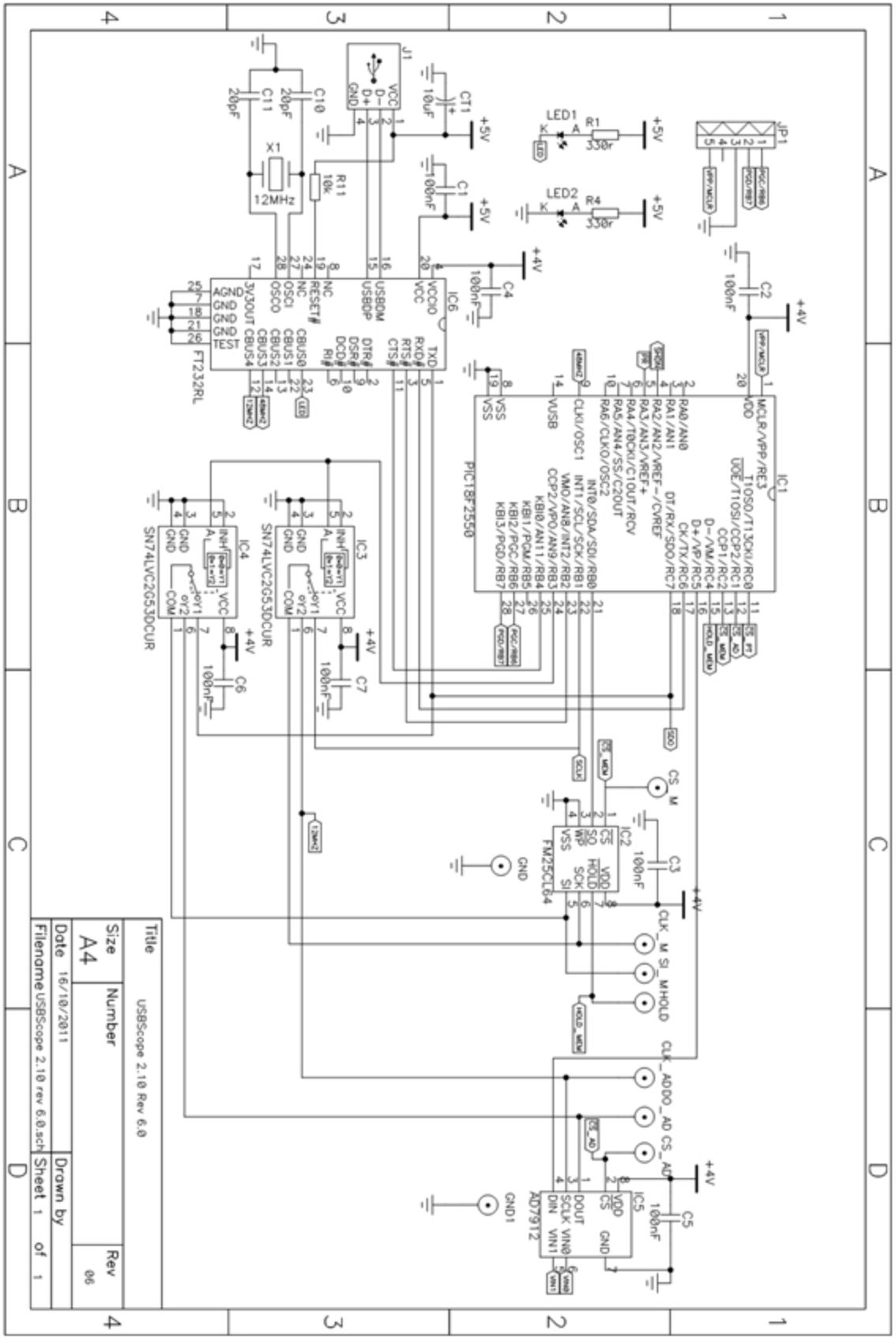
ANEXOS



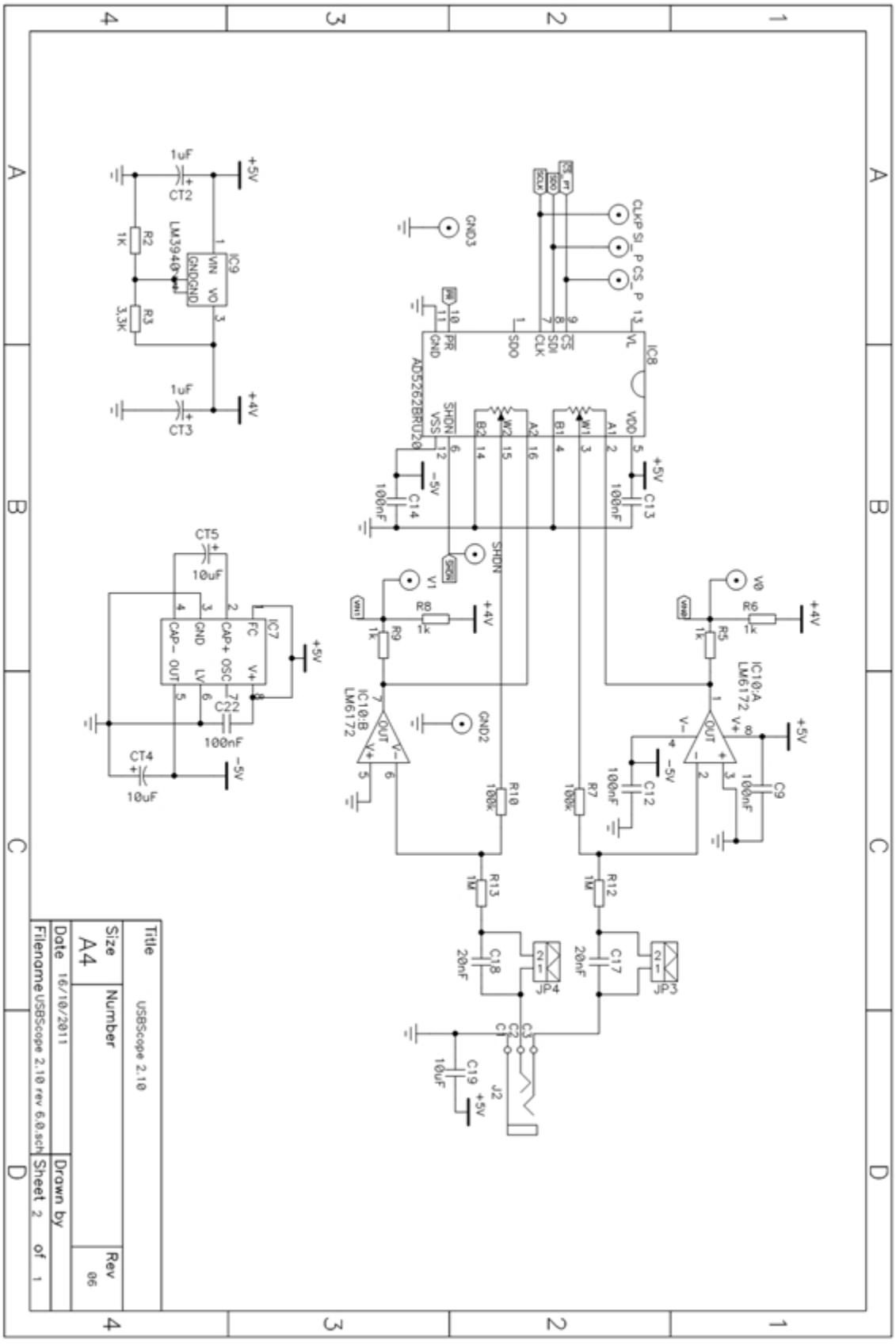
Anexo 1: Esquema elétrico USBScope 2.0 sheet 1



Anexo 2: Esquema elétrico USBScope 2.0 sheet 2



Anexo 3: Esquema elétrico USBScope 2.10 sheet 1



Anexo 4: Esquema elétrico USBScope 2.10 sheet 2