PONTIFÍCIA UNIVERSIDADE CATÓLICA DE GOIÁS
ESCOLA DE CIÊNCIAS EXATAS E DA COMPUTAÇÃO

VITOR DE ALMEIDA SILVA
Matricula: 2016.1.0033.0549-7

LISTA DE EXCÍCIOS N1

TALLES MARCELO G DE A BARBOSA

GOIÂNIA,
2019

VITOR DE ALMEIDA SILVA
Matricula: 2016.1.0033.0549-7

LISTA DE EXCÍCIOS N1

Trabalho apresentado como requisito parcial para obtenção de nota na disciplina Sistemas Embarcados no Curso de Engenharia da computação, na Pontifícia Universidade Católica de Goiás.
Talles Marcelo G de a Barbosa

GOIÂNIA,
2019

Exercícios 1: pág 48-49: números 1.1 à 1.12

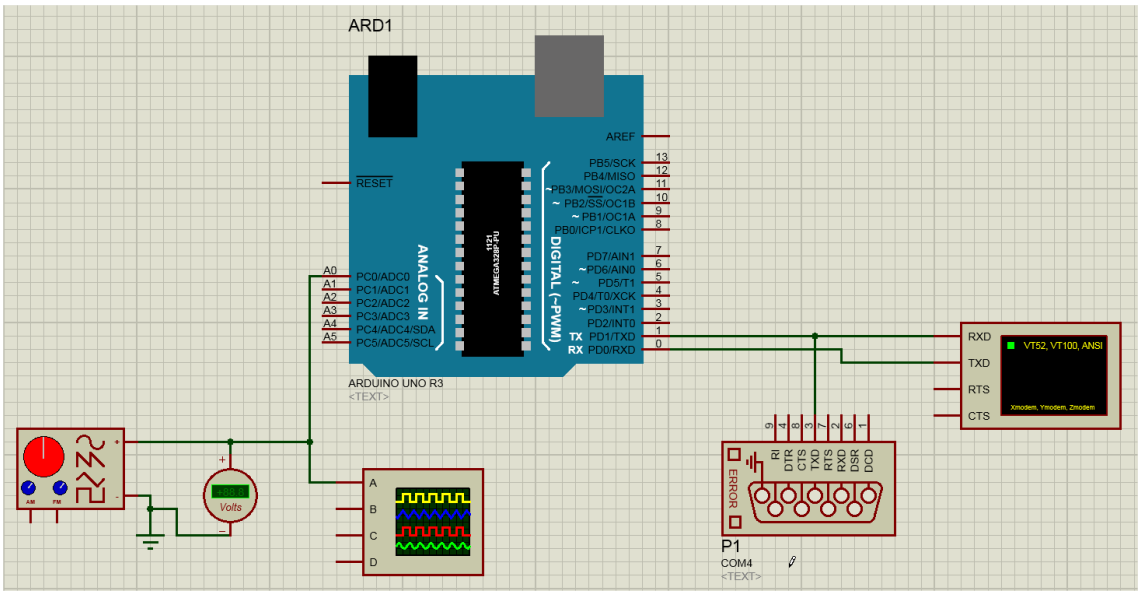
//Estes exercícios encontram-se feitos a mão na pasta e escaneados pelo camScanner (ANEXO 1 LISTA)

Exercícios 2: Amostragem de sinal (experimento com o Proteus)

Informações relevantes:

O microcontrolador escolhido foi um Arduino uno R3 presente na Livraria do Proteus. Para realizar a amostragem foi usado a geradora de sinal, que pode ser configurada para sinais bipolar ou unipolar, o sinal é recebido na porta A0 do Arduino (porta analógica), e é convertido para um valor correspondente e mandado pela porta serial. A Figura 1 mostra como ficou o arranjo.

Figura 1: Circuito para amostragem de Sinais



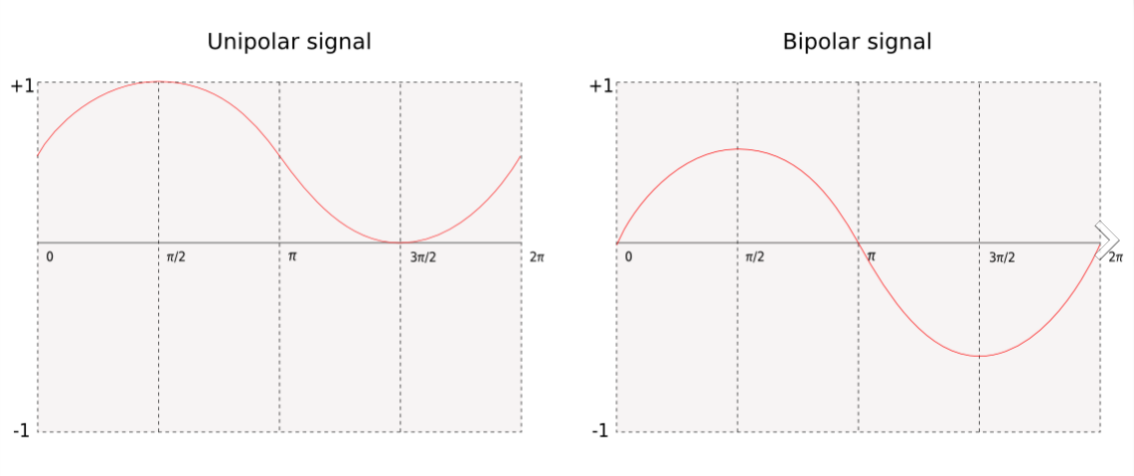
Fonte: Autor

Antes de prosseguir é preciso definir o que é uma onda Unipolar e bipolar:

- **Onda Unipolar:** é aquela cujo os valores variam em um único intervalo por vez, sendo este intervalo de 0 algum valor positivo ou de 0 a algum valor negativo;

- **Onda Bipolar:** Em contrapartida, uma onda bipolar é aquela cujos valores variam tanto do lado positivo do eixo y quando do lado negativo do eixo y, as Figura 2, mostram exemplos de ondas Unipolar e Bipolar;

Figura 2: Onda unipolar e onda bipolar



Fonte: (WIKIBOOKS, 2011)

Também é importante destacar que a precisão do converso A/D do Arduino é de 10 bits. O Arduino também é alimentado com uma tensão de 5volts, e por consequência a tensão de referência para o conversor A/D também vai varia entre 0 a 5 Volts na configuração padrão.

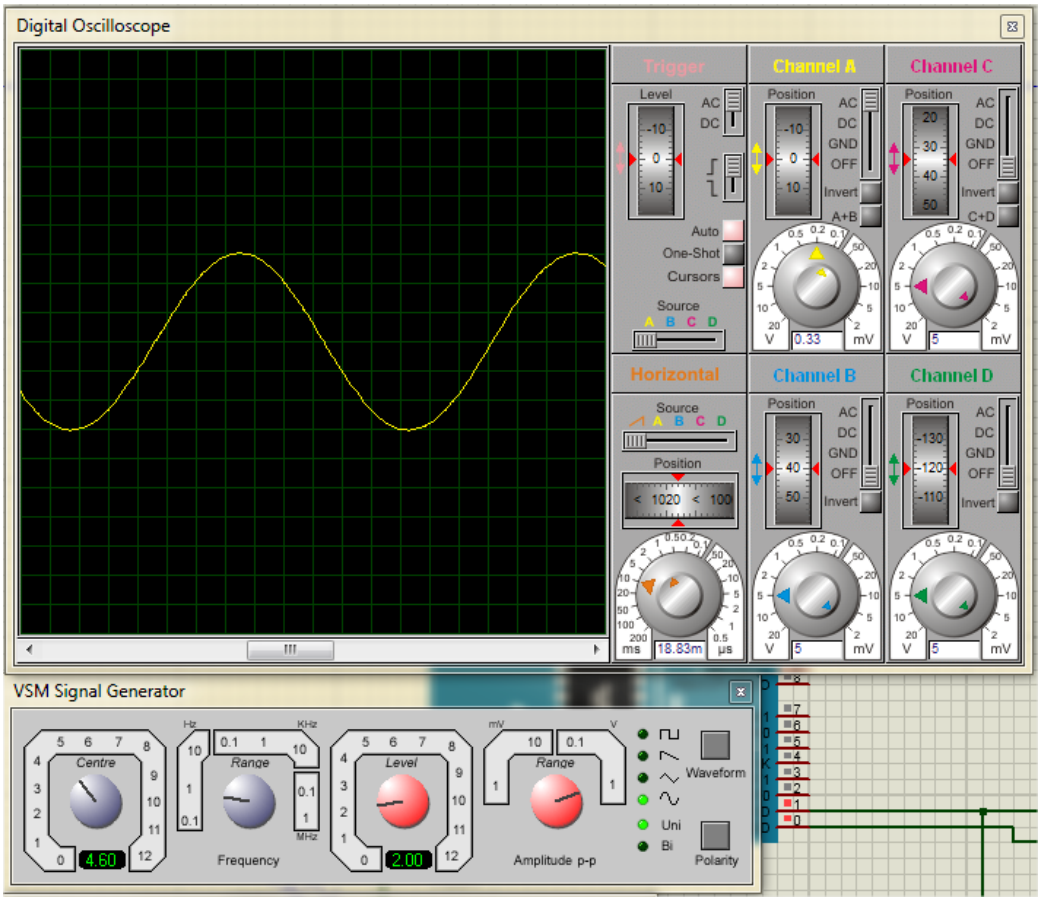
1) Utilizando o software Proteus e algum microcontrolador, realizar a amostragem de um sinal senoidal UNIPOLAR com frequências de 1Hz, 10Hz e 100Hz, verifique o aliasing. Amplitude máxima 5mv

A amostragem foi feita pela porta analógia 0 do Arduino, para representar o período de amostragem foi feito uso da função Delay(), a função mede o tempo em milissegundos, ou seja, é necessário um parâmetro de valor 1000 ms para se obter 1s (importante destacar que essa função não é muito precisa, logo esse valor pode não ser o real).

SINAL DE 1HZ:

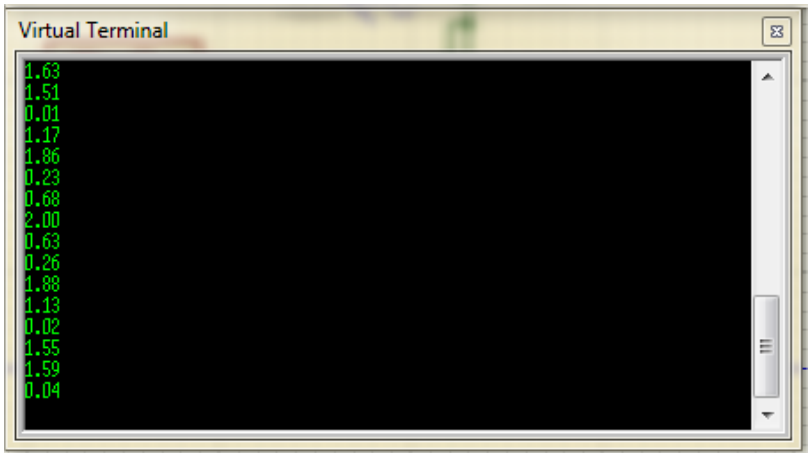
A Figura 3 e 4, mostra como é a forma do sinal de 1Hz no osciloscópio, e sua amostragem no terminal virtual. Para melhor visualização do sinal multipliquei por 1000 a amplitude logo o sinal gerado terá nó máximo 5 volts de amplitude, coloquei em 2 volts.

Figura 3: Sinal de 1 Hz unipolar, 2 volts de amplitude



Fonte: Autor

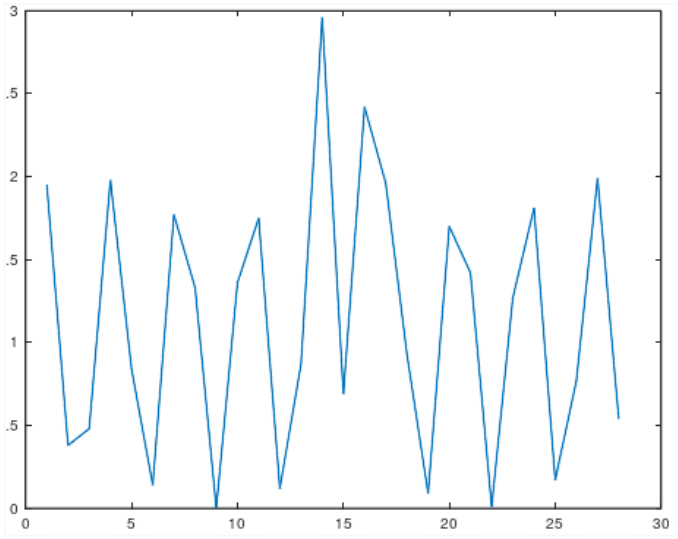
Figura 4: Sinal amostrado



Fonte: Autor

O sinal unipolar concentrou todo os valores da senoide no quadrante positivo do eixo y, dessa forma é possível se amostrar o que aparece no espaço de 0 a 5 volts. Para essa amostragem foi utilizado um delay de 500ms. A figura 5, mostra o resultado da amostragem em um gráfico plotado no software Octave.

Figura 5: Gráfico amostrado



Fonte: Autor

É possível notar que a amostragem não representou fielmente o sinal original, isso ocorre por conta do efeito chamado ***eleasing***, que é quando a quantidade de amostras não é suficiente para representar o sinal original, nesse caso podemos amentar a frequência de amostragem.

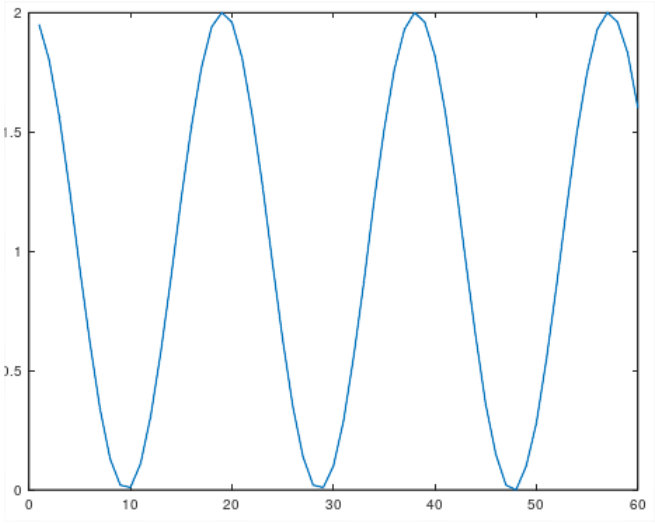
Por exemplo:

para Fn=1*10=10Hz

Pm=1/100*1000=10ms

A Figura 6 mostra o sinal amostrado com a nova frequência de amostragem.

Figura 6: sinal com 10ms de intervalos de amostragem



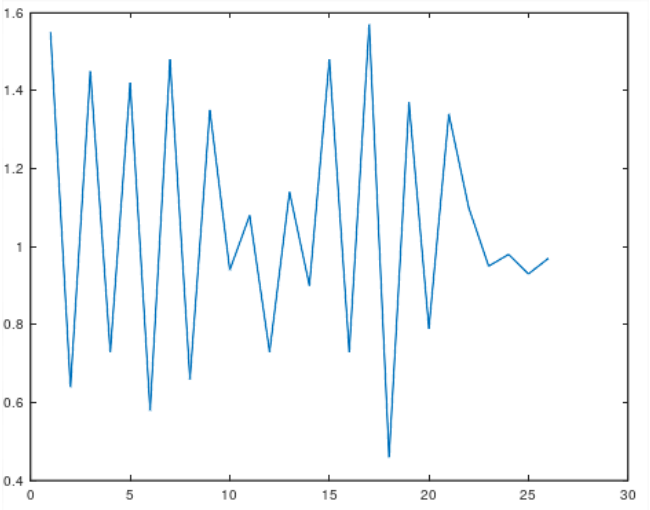
Fonte: Autor

Desta vez é possível notar que o sinal ficou bem fiel ao original. Alguém poderia imaginar que caso fosse mantido o período de amostragem de 500ms e aumentado a frequência do sinal poderia melhorar o resultado, contudo existem alguns contratempos:

- No mundo real nem sempre é possível mudar a frequência do sinal;
- Isso infligiria o teorema de amostragem de Nyquist-Shannon (Fn=2*Fmax).

Com propósito de ilustração a Figura 7 mostra o resultado obtido caso fosse aumentada a frequência do sinal e não a de amostragem.

Figura 7: Sinal amostrado com 1hz de frequência e Pm=500ms



Fonte: Autor

SINAL DE 10HZ:

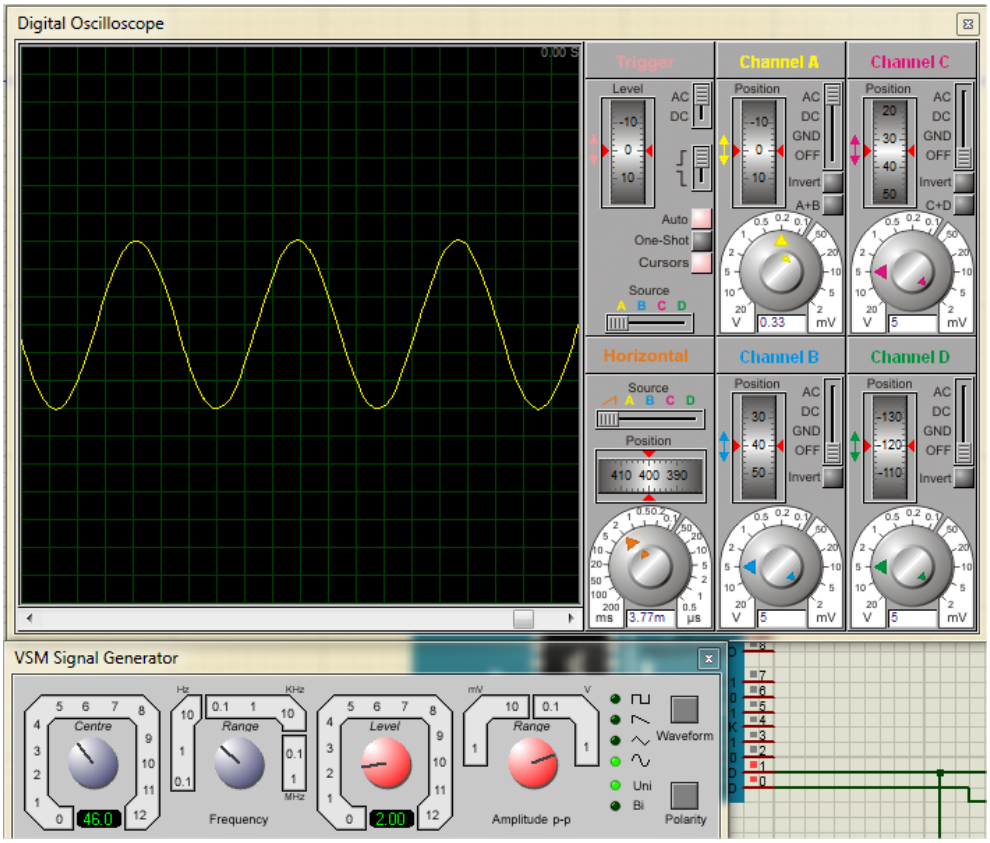
Para este caso o período de amostragem é:

Fn=10*2=20Hz;

Pm=1/20*1000= 50ms;

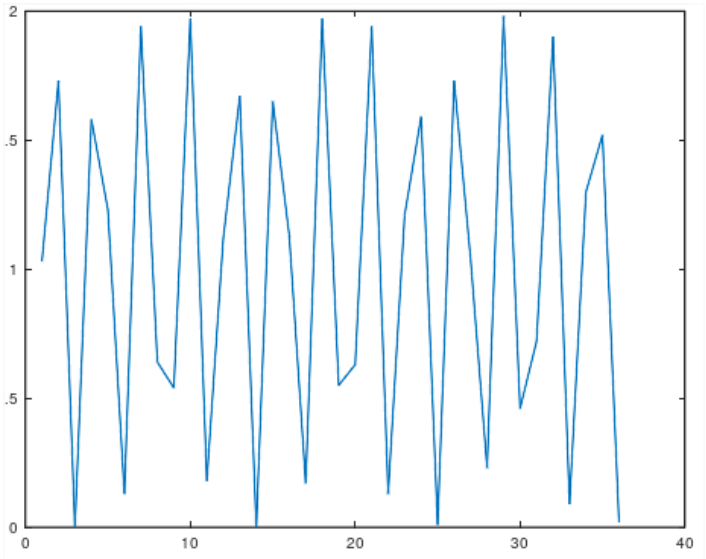
A Figura 8 e 9 mostram o sinal gerado de 10Hz no osciloscópio e a amostragem com a Pm especificada.

Figura 8: Sinal de 10Hz unipolar



Fonte: Autor

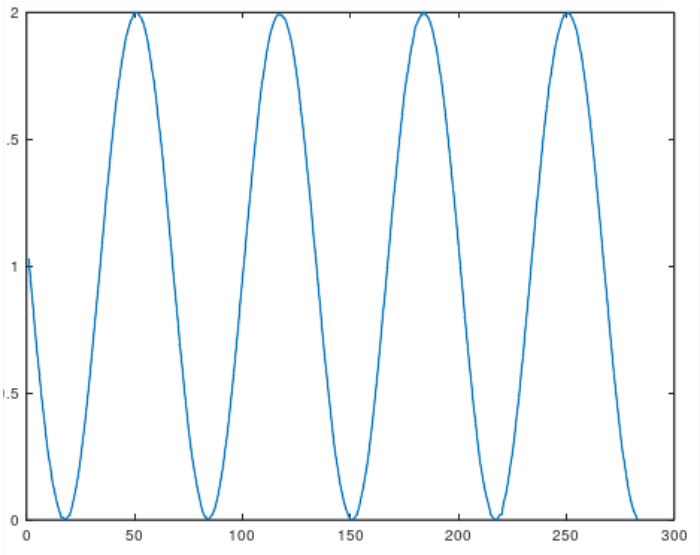
Figura 9: Sinal plotado no octave 50ms de período de amostragem



Fonte: Autor

Novamente pode se notar a ocorrência do ***alising*****,** desta forma, reduzindo o período de amostragem é possível obter um resultado melhor. A Figura 10 mostra o resultado com período de amostragem de 20ms.

Figura 10: Sinal amostrado com Pm=20ms



Fonte: Autor

SINAL DE 100HZ:

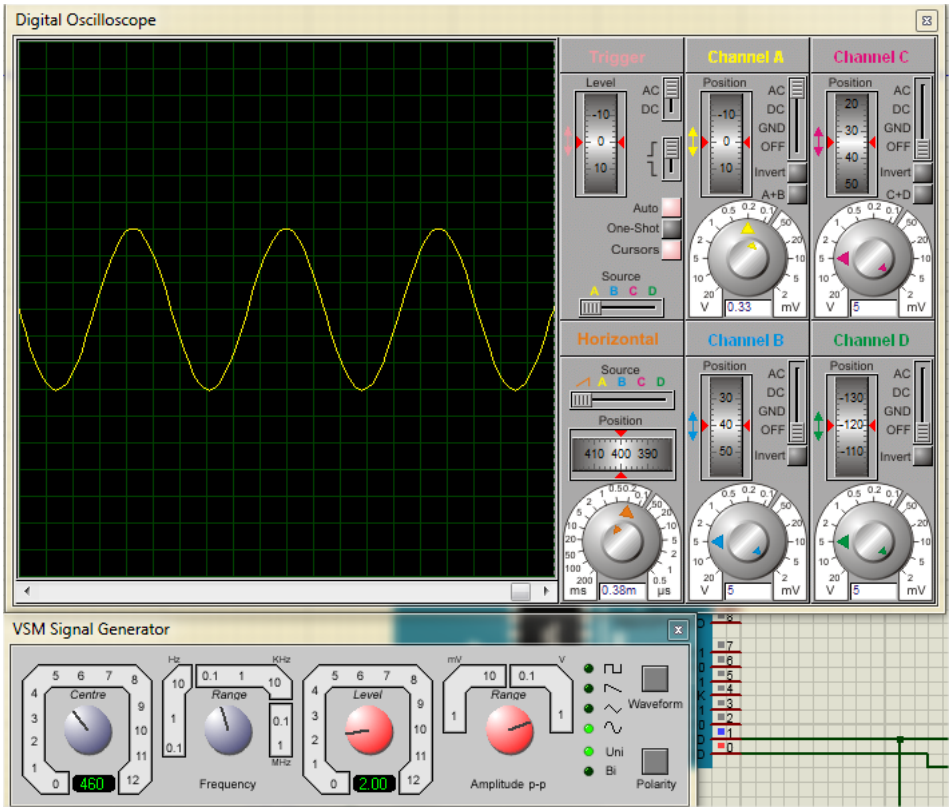
Para este caso o período de amostragem é:

Fn=100*2=200Hz;

Pm=1/200*1000= 5ms;

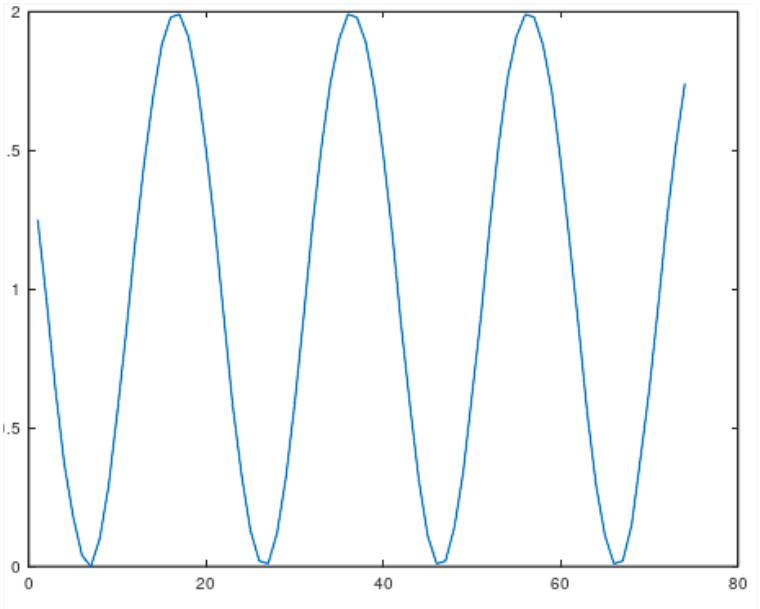
Figura 11 e 12 mostram o sinal no oscilador e amostrado com o período calculado.

Figura 11: Sinal no oscilador 100Hz



Fonte: Autor

Figura 12: Sinal de 100hz amostrado com Pm=5ms



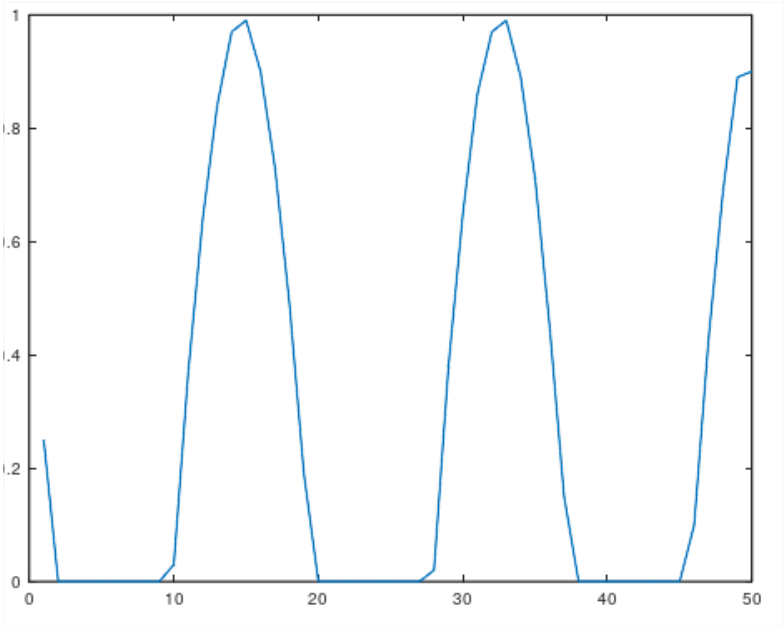
Fonte:Autor

Nesse caso o efeito de *alising* não foi tão significante, então não existe a necessidade de reduzir o período de amostragem. O teorema de frequência de Nyquist foi o suficiente para amostrar o sinal de 100Hz.

SINAL BIPOLAR:

A diferença entre amostrar um sinal unipolar e bipolar usando o Arduino está na quantidade de valores que serão coletados pelo conversor A/D. A tensão de referência usada pelo Arduino varia de 0 a +5 volts, dessa forma ao se amostrar o sinal não será possível coletar os valores negativos, logo, o sinal visualizado será cortado no ponto 0 do zero do eixo y. A Figura 13 mostra um exemplo para o sinal de 100Hz do tipo bipolar com 2 volts de amplitude.

Figura 13: sinal bipolar amostrado com o Arduino



Fonte: Autor

A forma de evitar isso poderia é realizar um deslocamento de offiset levando todo o sinal para cima do eixo x, entre o intervalo de 0 a 5 volts, com isso ele poderia ser completamente visualizado. Como a geradora de sinais já fornece uma opção de sinal unipolar ele foi utilizado nos exercícios anteriores, porém, o circuito para realizar o deslocamento será utilizado mais a frente.

2) Aplicar um filtro ante *aliasing* no sinal e verificar o resultado

O *alising* ocorrer quando a frequência do sinal é bem mais alta que a frequência de amostragem, dessa forma, os valores amostrados podem não ser suficiente para representar o sinal de entrada. Uma medida para solucionar este problema pode ser a aplicação de um filto *anti-alising*, um filtro passa baixa por exemplo.

Nesse tipo de filtro pode ser controlado o valor da frequência de corte de forma a manter a frequência dentro de um intervalor, a formula (1) mostra a regra:

(1) Fc=1/(2*pi*R*C)

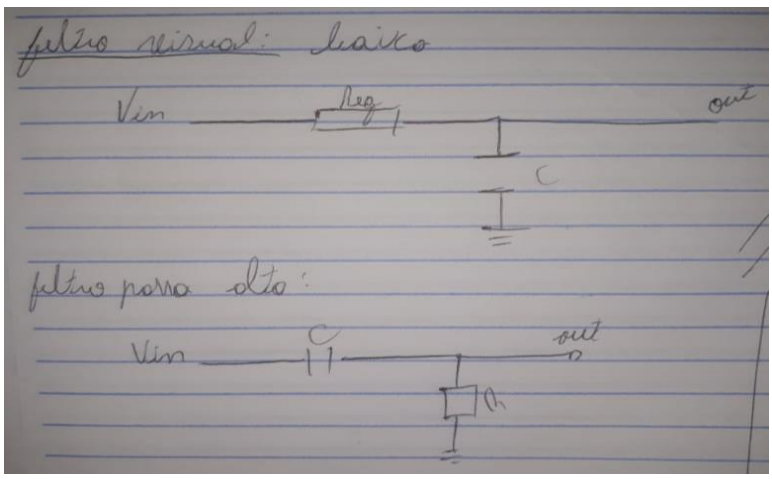
- R: resistência;
- C: valor da capacitância.

Informações:

- É ideal que o capacitor não seja polarizado;
- A faixa de resistência tem que ser um pouco mais alta que o calculado.

Filtros passa baixa e passa alta seguem o modelo mostrado na Figura 14.

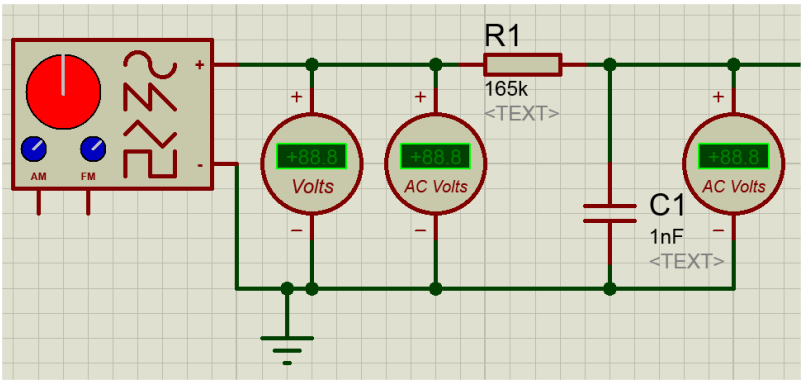
Figura 14: filtro passa baixa e filtro passa alta



Fonte: Autor

Esquema do circuito já pronto no Proteus é mostrado na Figura 15.

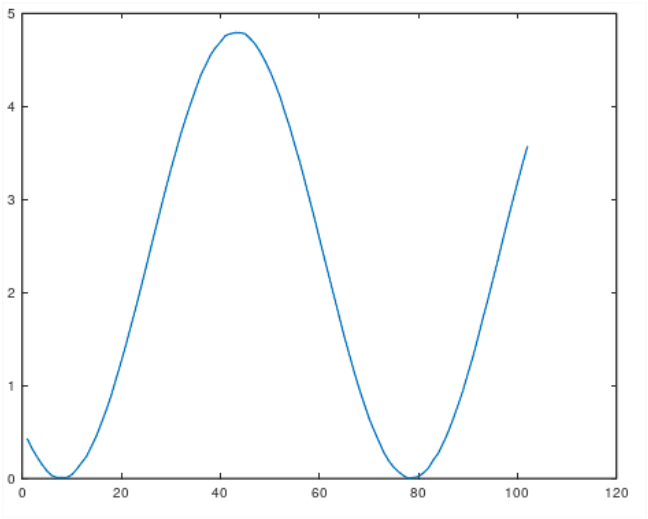
Figura 15: Filtro passa baixa no Proteus



Fonte: Autor

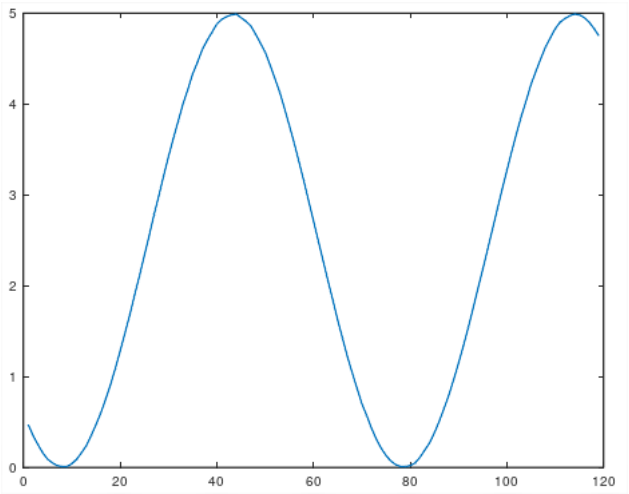
A frequência de corte escolhida foi de 1hz. O resultado do sinal amostrado sem o filtro é apresentado na Figura 16, a Figura 17 mostra o sinal amostrado com o filtro passa baixa (sinal de 5 volts).

Figura 16: Sinal amostrado sem o Filtro Fc=1hz



Fonte: Autor

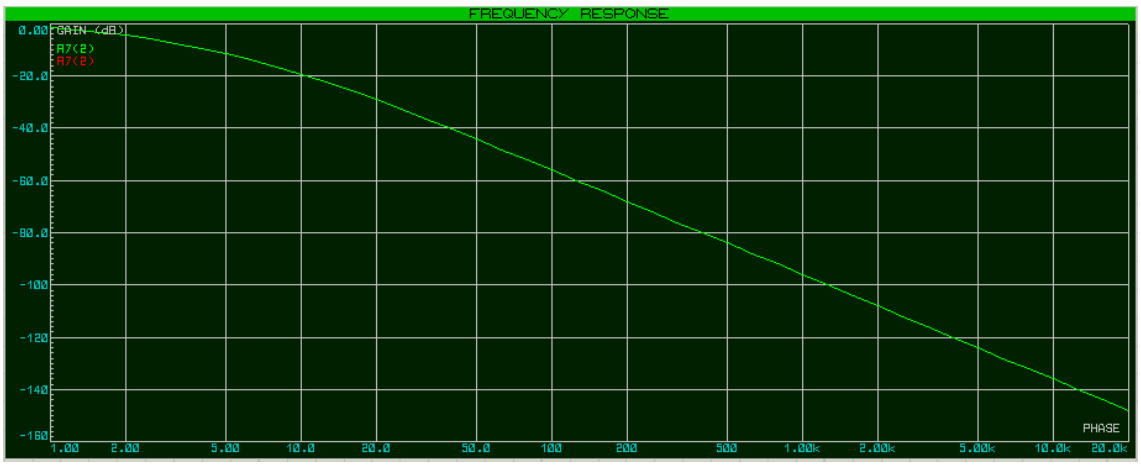
Figura 17: Sinal amostrado com o Filtro Fc=1hz



Fonte: Autor

Agora, reconfigurando o filtro para R=10k e C=4uF para uma frequência de corte equivalente a 4Hz e aumentado a ordem para 2, tem-se a resposta em frequência da Figura 18. A figura 19 mostra o filtro de segunda ordem.

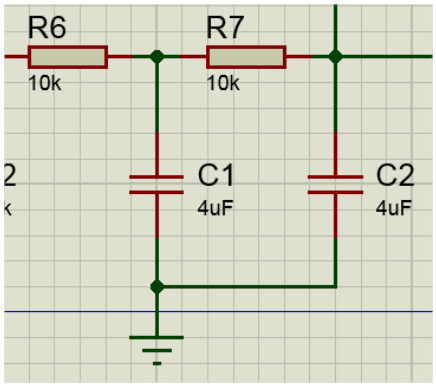
Figura 18: resposta em frequência



Fonte: Autor

Note que o filtro não é ideal e está longe de realizar o corte em 4Hz, oque ocorre é uma defasagem do sinal, conforme a frequência vai aumentado o sinal é defasado, até que em um momento é cortado.

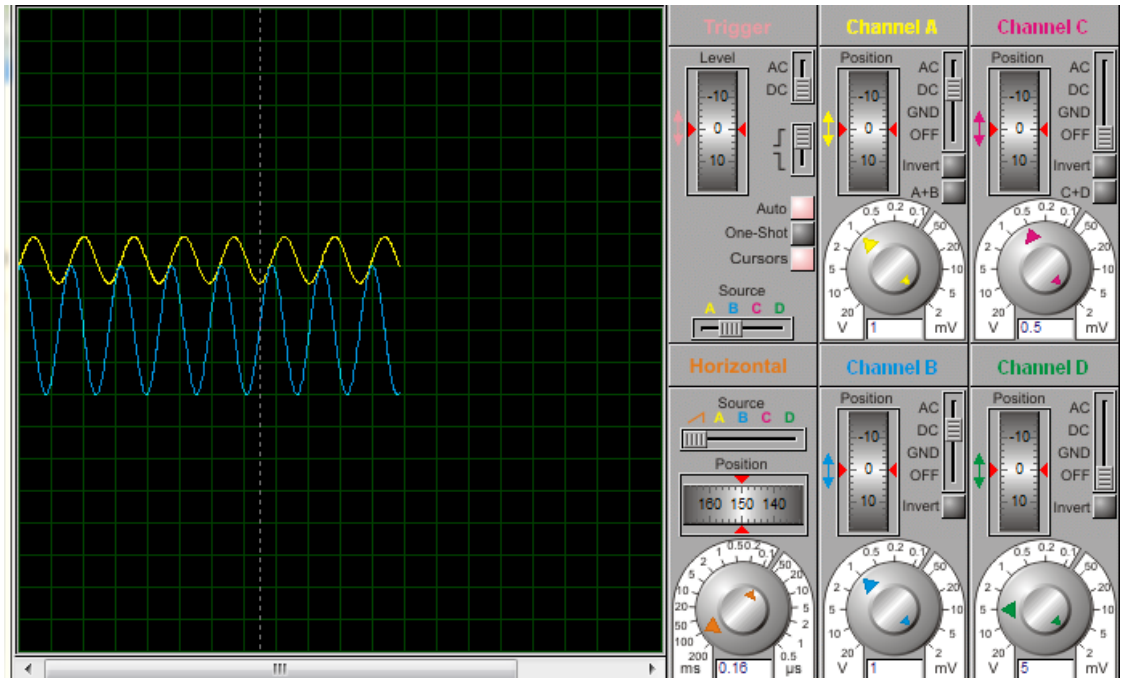
Figura 19: Filtro antieliasing de segunda ordem



Fonte: Autor

Como teste foi aumentada a frequência do sinal de entrada para 4Hz, a Figura 20 mostra o resultado.

Figura 20: Defasagem do sinal com 4Hz



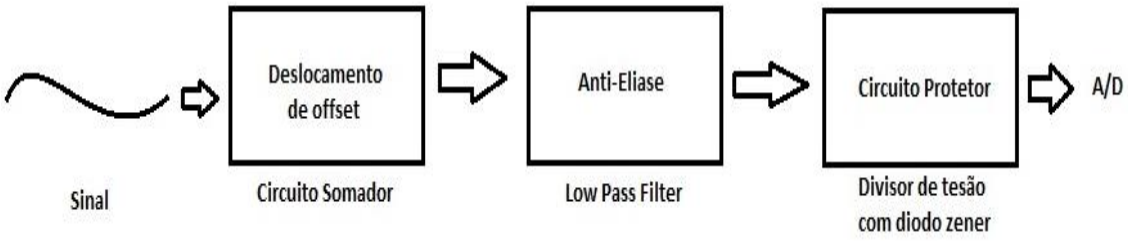
Fonte: Autor

É possível notar a defasagem do sinal, isso ocorre por conta do filtro que tem sua frequência de corte configurada em 4Hz. Porém, o ideal é que o sinal fosse cortado para 0volts, mas isso não ocorre por conta de que o filtro não é ideal como já mensionado. Seria necessário fazer mais ordens e cálculos de corte para se aproximar do corte em 1 Hz.

3) **Tratando o sinal para realização do deslocamento de offiset e filtro: usando circuito somador + filtro passa baixa (ant alise) + circuito protetor (diodo zener).**

Nesta etapa o sinal irá ser tratado antes de ser amostrado pelo Arduino, o sinal irá passar por um circuito que segue o modelo da Figura 21.

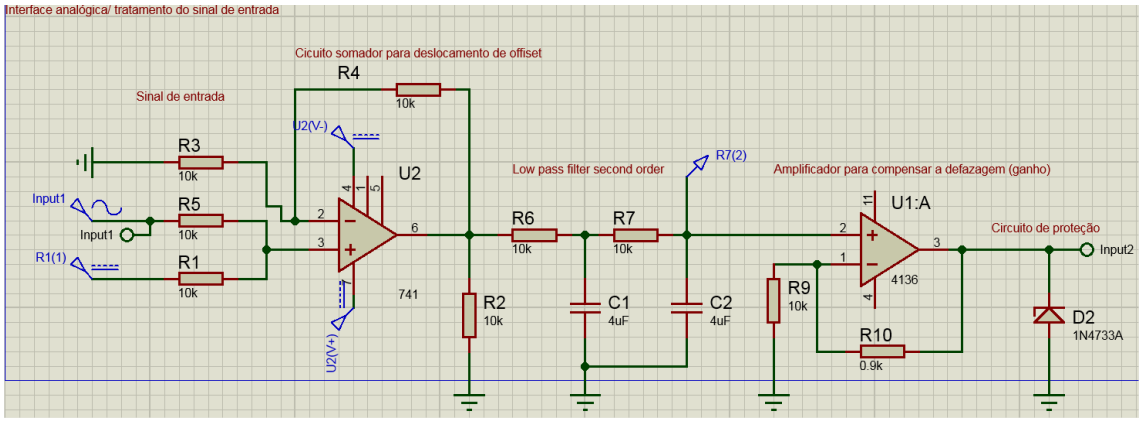
Figura 21: circuito para tratar sinal



Fonte: Autor

Dessa forma o circuito resultante ficou como ilustrado na Figura 22.

Figura 22: Circuito par tratar sinal no Proteus



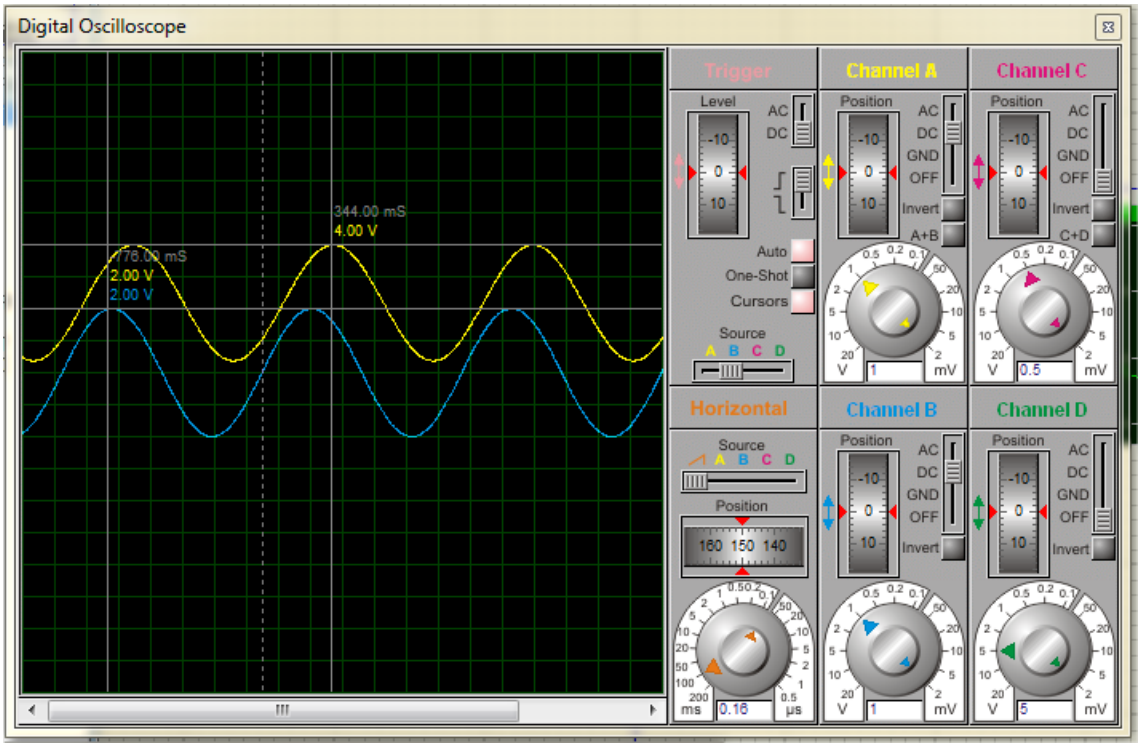
Fonte: Autor

- O circuito somador serve para somar a tensão da fonte de referência Dc com a tensão do sinal senoidal nesse caso a Dc tem 2 volts e a senoidal também, dessa forma o sinal é deslocado. Para chegar ao final uma tensão de 4 volts foi preciso somar 2V Dc + 2V Ac e montar um circuito de ganho 1.09;

- O Circuito de filtro passa baixar serve para manter a frequência do sinal abaixo da frequência de corte calculada para evitar o *alising;*
- O diodo atua como circuito protetor, de forma que sua tensão de saturação é de 5 volts para evitar que o Arduino seja danificado.
- Foi adicionado antes do diodo zener um amplificador com um ganho de 1.09 para compensar a defasagem do sinal ao passar pelo filtro.

Para realizar a amostragem foi selecionado um sinal de entrada de 2 volts bipolar e com frequência de 1Hz. O resultado do sinal ao passar pelo circuito de tratamento pode ser visto na Figura 24.

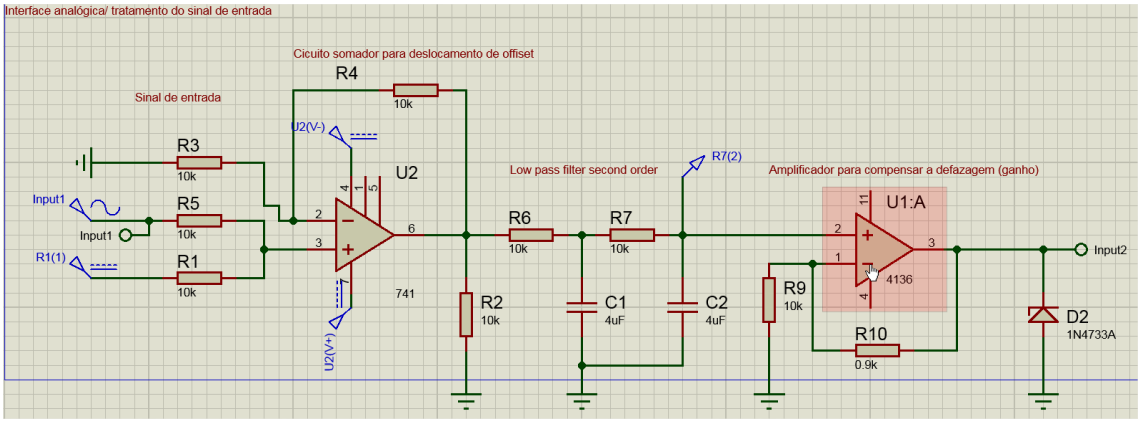
Figura 24: Sinal tratado pelo circuito



Fonte: Autor

É importante destacar que ao final do circuito foi adicionado um amplificado de ganho 1.09 para dar estabilidade ao sinal e compensar a defasagem antes de conectar no pino A0 do Arduino. Figura 25, o local onde foi adicionado o amplificador.

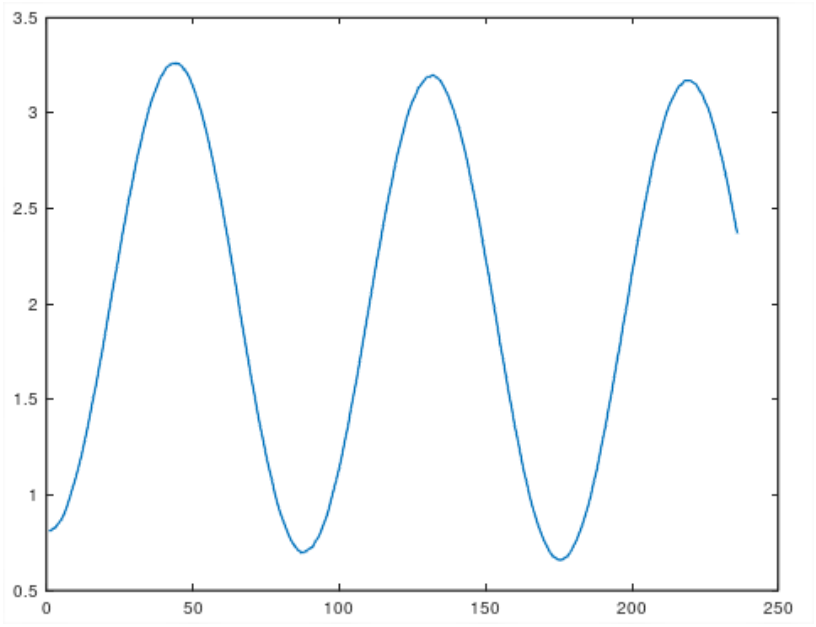
Figura 25: Amplificador



Fonte: Autor

Seguindo a frequência de Nyquist, a frequência de amostragem para um sinal de 1 Hz seria 2 Hz, porém, Foi indicado anteriormente que nem sempre isso irá ser o suficiente, desta forma é recomendado usar na prática uma frequência de amostragem 10x o valor da maior frequência do sinal. Dessa forma, para um sinal de 1Hz, a frequência de amostragem será de 10Hz, sendo o período de 50ms. A Figura 26 mostra o sinal tradado e amostrado.

Figura 26: Sinal tratado e amostrado



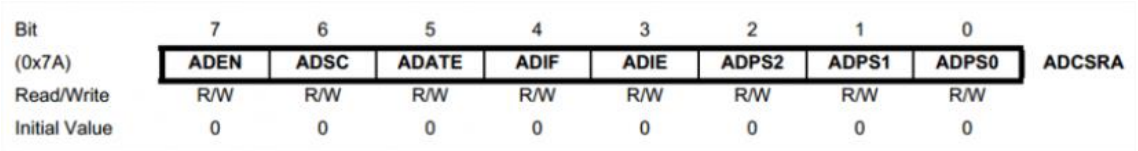
Fonte: Autor

4) Parte digital, Código para conversão A/D usando os registradores e interrupção.

Nessa etapa foi realizado uma aprimoração no código de forma a se ter uma melhor taxa de amostragem. Para isso foram modificados as configurações no Registrador do conversor A/D. O clock do conversor A/D por ser alterado configurando o fator de divisão do clock do sistema e do clock do ADC.

Utilizando o Registrador ADCSRA é possível fazer essa configuração. O bloco prescaler controla o clock do conversor A/D, assim o clock do conversor A/D será uma fração do clock do oscilador principal, conforme o fato do prescaler. A Figura 27 mostra o Registrador ADCSRA e a Figura 28 mostra os possíveis valores para o fator de divisão (SOUZA, 2014).

Figura 27: Registrador ADCSRA



Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Fonte: (SOUZA, 2014)

Figura 28: Valore de Prescaler

ADC Prescaler Selections

ADPS2	ADPS1	ADPS0	Division Factor
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Fonte: (SOUZA, 2014)

Dado isso é possível tentar usar a interrupção lançada pelo bit ADIF, antes abilitando o ADIE e o bit 1 do SREG. Porém não foi possível realizar esse tipo de tratamento, dessa forma optou-se por usar o contador timer1, que conta com

uma biblioteca já implementada e permite o uso de interrupção ao final da contagem utilizando a função ***Timer1.atachinterupt(função).*** A seguir é exposto o código, as configurações do ADC e o uso do Timer1.

```
#include "TimerOne.h"
// Variáveia para armazenar os resultados
unsigned long tempo_inicio;
unsigned long tempo_fim;
unsigned long valor;

//sbi é uma macro para setar o bit (do segundo argumento) do endereço (do primeiro argumento) para
//1

const unsigned char PS_16 = (1 << ADPS2);
//linha 5 tabela  clock 1Mhz   50khz amostras
const unsigned char PS_32 = (1 << ADPS2) | (1 << ADPS0);
 //linha 6 tabela  clock=500 khz   31khz //amostras
const unsigned char PS_64 = (1 << ADPS2) | (1 << ADPS1);
//linha 7 tabela  clock=250khz    16Khz amostra
const unsigned char PS_128 = (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0);
// linha 8 tabela  clock=padrão = 8600Hz amostra
//estas atribuições são concatenações de valores

void setup() {
 Serial.begin(9600);

 //configura o preescaler do ADC
 ADCSRA &= ~PS_128;  //limpa configuração da biblioteca do arduino

//valores possiveis de prescaler só deixar a linha com prescaler desejado
//PS_16, PS_32, PS_64 or PS_128
//ADCSRA |= PS_128; // 128 prescaler
//ADCSRA |= PS_64; // 64 prescaler
//ADCSRA |= PS_32; // 32 prescaler
//ADCSRA |= PS_16; // 16 prescaler
//pega a interrupção lançada pelo bit aden do ADCSA
```

```
//como eu não consegui gerar uma interrupção usando os bits do proprio ADCSRA com o SREG
//optei por combinar a interrupção do timer 1 juntamente com a mudança de clock do A/D
//para começar incio o ADC com clock padrão

Timer1.initialize(100000);
// Inicializa o Timer1 e configura para um período de 0,1 segundos (parametro é dado em micro
//segundos  1*10^(-6)s)

/*exemplo:
 * para um sinal de 1Hz a frequência de Niquist seria 1/2 segundos
 * porem na pratica esse valor não é suficiente por isso aumentados de 2, para 10 vezes a maior freq
do sinal para as amostras logo
 * a freqN para um sinal de 1Hz é 1/10 seg, ou 0,1 segundos   ( timer1.initialize(100000))
 */

Timer1.attachInterrupt(callback);
 // Configura a função callback() como a função para ser chamada a cada interrupção do Timer1

pinMode(A0, INPUT);
}

void callback()
{
  //leitura
  tempo_inicio = micros(); //marca tempo de inicio de leitura

  float  valorAD = analogRead(A0);     //lé o valor de entrada no pino A0
  float Vd= valorAD * (5.0 / 1023.0);  //faz o caluclo para que o valor seja mostrado entre 0 e 5 volts

  tempo_fim = micros();              //le tempo no fim da conversão

  //exibe valor lido e tempo de conversão

  //Serial.print("Valor= ");

  Serial.println(Vd);

 //Serial.print(" -- Tempo leitura = ");
```

```
  //Serial.print(tempo_fim - tempo_inicio);
  //Serial.println(" us");

  // este delay pode vir a ser pulado caso o timer lance a interrupção
   delay(500);
}
void loop()
{

  //estado de espera

}
```

1) Realizando amostragem do sinal de 1Hz, 10Hz e 100Hz

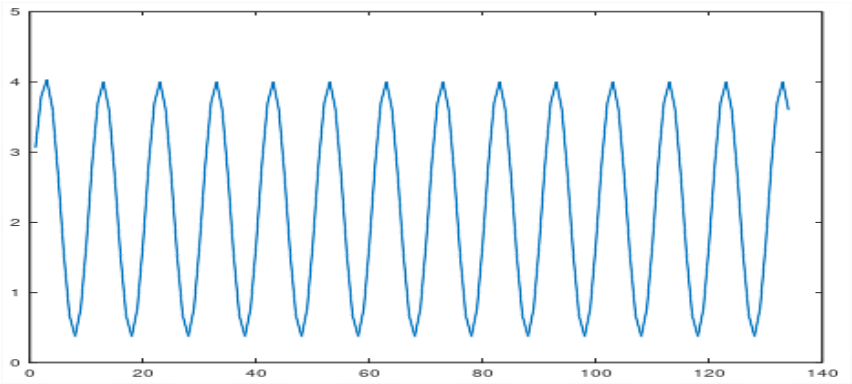
Comando : Timer.initialize(100000); //0.1 s

O parâmetro da função de inicializar o timer1 é dado em microssegundos, dessa forma, para saber o período de amostragem basta usar a função (2).

(2) FREQa= 1/(10 *FREQmax) * 10^(6);

A formula usa multiplica a frequência por 10 ao contrário de 2 para compensar o alising na prática. O conversor A/D foi deixado nas configurações padrão, isso é, aproximadamente uma taxa de 9600Hz (nos cálculo 8600 Hz) de amostragem. A Figura 29 mostra o resultado da amostragem.

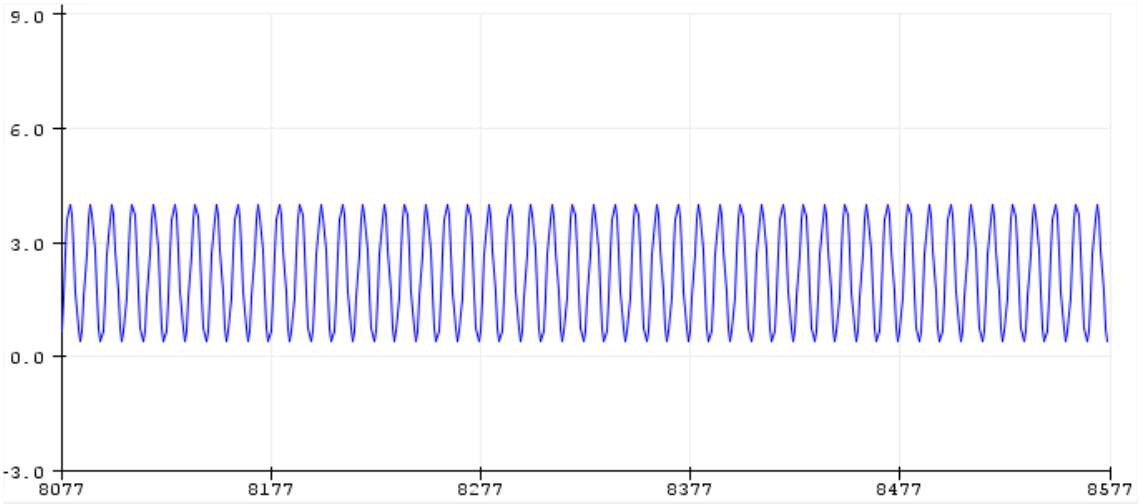
Figura 29: Amostragem de sinal de 1Hz usando interrupção período 100000us



Fonte: Autor

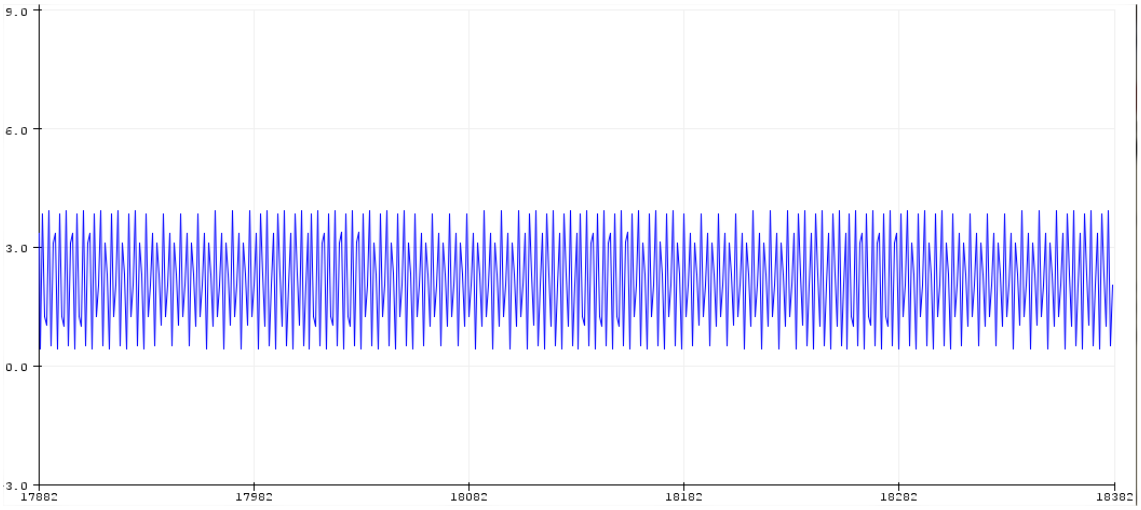
As Figuras 30 e 31 mostram as amostragens dos sinais de 10Hz e 100Hz respectivamente.

Figura 30: Sinal de 10Hz período 10000 us



Fonte: Autor

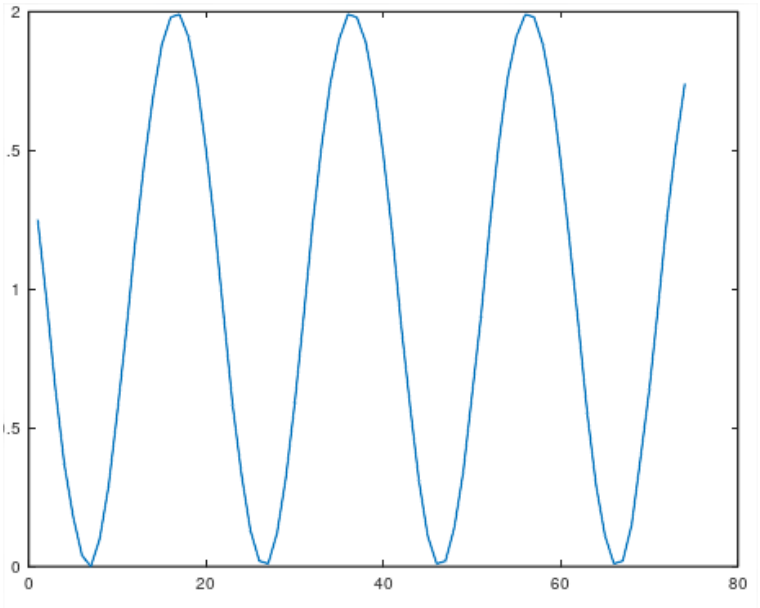
Figura 31: Sinal de 100Hz período 100 us



Fonte: Autor

O código com interrupção com timer não conseguiu amostrar muito bem o sinal de 100Hz, mesmo configurando o A/D para o prescaler de 16 que é o clock máximo. A figura 31 mostra o teste do sinal de 100hz usando o código inicial com delay de 10, ele conseguiu amostrar melhor.

Figura 31: Sinal de 100hz amostrado com Pm=5ms



Fonte:Autor

REFERÊNCIA

SOUZA, Fábio. **Arduino UNO - Taxa de amostragem do conversor A/D.** [s. L.]: Embarcados, 2014. Disponível em: <https://www.embarcados.com.br/arduino-taxa-de-amostragem-conversor-ad/>. Acesso em: 17 set. 2019.

WIKIBOOKS. **Sound Synthesis Theory/Modulation Synthesis.** [s. L.]: Wikibooks, 2011. Disponível em: <https://en.wikibooks.org/wiki/Sound_Synthesis_Theory/Modulation_Synthesis>. Acesso em: 10 set. 2019.