Universidade Federal do Pará
Campus Universitário de Tucuruí
Curso de Graduação em Engenharia Elétrica

Laboratório de Sistemas de Controle

Monitores: Denilson Costa da Silva
Matheus Gama da Silva

Tucuruí-PA
2022

1 Roteiro 1

1.1 Objetivo

Ao término do roteiro o discente deve compreender:

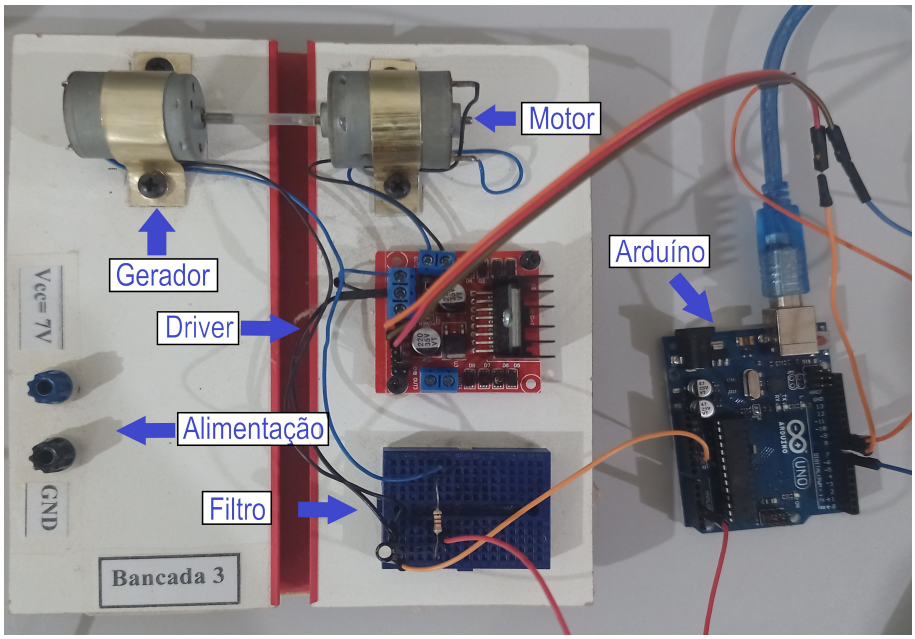
1. A funcionalidade de cada dispositivo da bancada de teste;
2. As etapas de comunicação entre cada dispositivo;
3. A instalação dos softwares necessários para a realização das atividades da disciplina;
4. Demostre conhecimento teórico a partir da resolução dos exercícios propostos.

1.2 Conhecendo a Bancada Motor- Gerador

A bancada Motor-Gerador é formada pela interconexão entre os seguintes elementos: dois motores DC, um Driver L298N, uma matriz de contatos e uma placa Arduino. A bancada permite a realização de experimentos voltados para a disciplina de Laboratório de Sistemas de Controle.

- Roteiro 01 - Conhecendo a bancada
- Roteiro 02 - Modelagem / Identificação
- Roteiro 03 - Controle: P, PI, PID
- Roteiro 04 - Controle: Realimentação de Estados
- Roteiro 05 - Controle: Realimentação por Estimação dos Estados

A bancada tem como objetivo fornecer os meios necessários para que haja uma fidelidade, no envio e no recebimento de sinais provenientes do sistema motor-gerador que é o objeto de estudo. Na figura 1 tem-se a imagem da Bancada 03. A interconexão entre os elementos são explicados de forma detalhada nas próximas seções.



Fonte: Autor

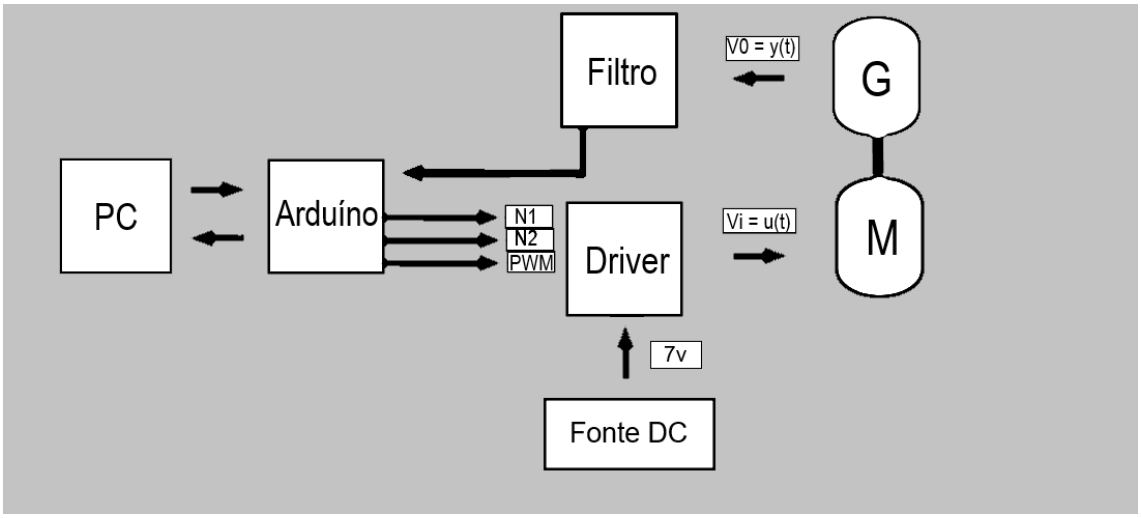
Figura 1: Bancada Motor-Gerador DC, Alimentação 7V, Driver L298N, Protoboard com Filtro RC, Arduino Uno.

Na figura 2 é observado o diagrama funcional da bancada, o mesmo tem como objetivo exemplificar a rede de comunicação entre os dispositivos na sua forma mais simples e direta.

Funcionamento básico da bancada:

A bancada é alimentada por uma fonte de tensão DC assimétrica de 7V. Essa é a tensão máxima pode ser aplicada nos terminais de alimentação do Motor DC. Há outras bancadas Motor-Gerador no laboratório que funcionam com tensão de alimentação entre 8 e 15V. Os motores DC estão acoplados eixo-a-eixo, um funciona como Motor e o outro como Gerador. A tensão gerada nos terminais do Gerador pode ser entregue para um filtro RC série, com a finalidade de atenuar o ruído de medição. A versão filtrada da tensão gerada é então entregue à placa Arduíno através de uma conexão entre o pino AD (Analógico digital). A placa Arduíno estabelece um elo de ligação entre a bancada e um computador pessoal (PC). No computador, encontra-se um código, que contém um algoritmo de comando e define o valor de tensão aplicada nos terminais do motor DC para que nos terminais do Gerador seja observado a tensão desejada.

O valor de tensão do motor DC é enviado para a placa Arduíno via comunicação serial. Na placa Arduíno, encontra-se um outro código, o Firmware, que converte o valor de tensão em um valor PWM entregue nos pino **Ativa MA** do Driver. Além do valor PWM, dois outros



fonte: Autor

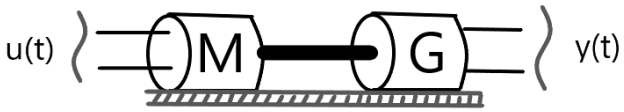
Figura 2: Diagrama Funcional da Bancada Motor-Gerador

pinos definem o sentido de rotação do eixo do motor DC,IN1 e IN2. O Driver, L298N, por sua vez, estabelece a ligação entre a fonte de alimentação e o Motor DC. Observa-se que a tensão aplicada nos terminais do Motor DC será comandada pelo sinal PWM, no entanto, a corrente é drenada pela Fonte de Alimentação. Todos os processos acima trabalham em conjunto para a obtenção dos valores mais fidedignos possíveis da planta, o sinal $u(t)$ representa valores aplicados no **Motor M**, esse sinal origina um sinal $y(t)$ na saída do **Gerador G**

Todas as etapas mostradas no diagrama de blocos e explicadas brevemente no texto acima serão explanadas com mais detalhes nas próximas secções desse documento.

1.3 Planta

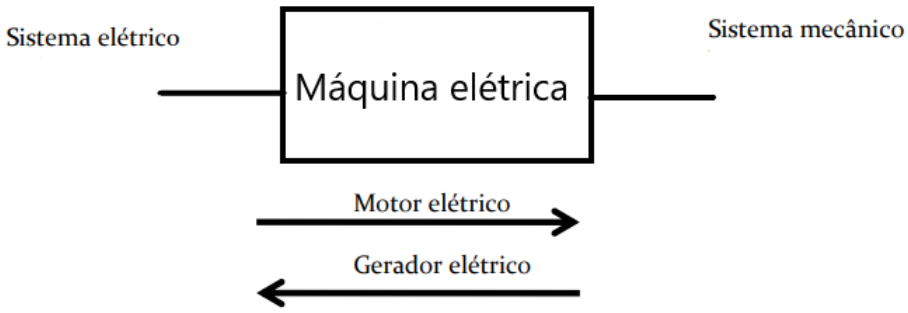
O sistema analisado na bancada consiste de um acoplamento mecânico de duas máquinas elétricas DC (de baixa potência), uma funcionando como **motor M** e outra como **gerador G**. Na figura 3 tem-se o desenho do sistema Motor-Gerador, em que $u(t)$ representa a tensão nos terminais do Motor e $y(t)$ representa a tensão nos terminais do Gerador.



fonte: Autor

Figura 3: Diagrama de conexão mecânica entre o motor **M** e do gerador **G**

Em questões construtivas, o motor e o gerador possuem os mesmos componentes, um rotor, um estator, uma armadura e entre outros componentes. Na configuração adotada, o Motor recebe um sinal de tensão nos terminais de alimentação ($u(t)$) e produz um torque no eixo do rotor. Enquanto o Gerador recebe um torque no eixo do rotor, e gera um sinal de tensão nos terminais ($y(t)$). Na figura 1.3 tem-se um diagrama, que simplifica o processo de transformação de energia.



fonte: imagem adaptada de https://microsoft-power

Figura 4: Exemplo esquemático de uma Máquina elétrica DC.

O acoplamento mecânico entre o Motor (M) e o Gerador (G) formam a planta objeto de estudo.

Inicialmente, o objetivo é estabelecer relações matemáticas entre o sinal de entrada (tensão nos terminais do Motor), $u(t)$ e o sinal de saída, tensão nos terminais do Gerador, $y(t)$.

O estudar pode ser realizado tanto no domínio do tempo, como na da frequência. Uma vez coletados esses sinais, teremos que nos deparar com alguns contra-tempos, e o mais significativo deles é o ruido gerado, que acaba "POLUINDO"o sinal de saída. A solução para esse problema é utilizar um **Low Pass Filter (LPF)** , que permitira somente a passagem de baixas frequências, sendo que os ruídos são de alta frequência

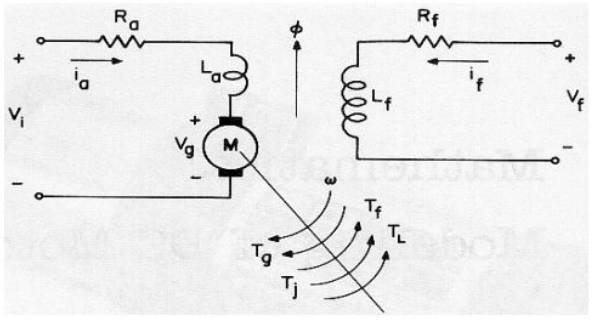
1.3.1 Função de transferência (FT)

Esse método consiste em estabelecer uma relação entre a transformada de Laplace da saída $y(t)$ pela transformada da entrada $u(t)$,assim podemos obter uma resposta no domínio da frequência s. O sistema motor - gerador vai ser determinado pela seguinte equação.

$$H(s) = \frac{Y(s)}{U(s)} \tag{1}$$

onde $U(s)$ é a entrada e $Y(s)$ a saída do nosso sistema.
Podemos estabelecer a função de transferência de um motor DC através de duas formas, A primeira pela equação diferencial que rege o sistema e outra através dos dados coletados em $u(t)$ e $y(t)$. Na figura abaixo temos uma imagem de um esquema eletromecânico de um motor DC,

Figura 5: Esquema eletromecânico de um motor DC



Fonte:http://professorcesarcosta.com.br/

Onde temos que:

- R_a: Resistência da armadura.
- L_a: Indutância do enrolamento da armadura.
- v_g: Tensão gerada na armadura.
- O enrolamento de campo é representado por uma resistencia R_f e uma L_f, com o fluxo de campo liquido de entreferro sendo designado por Φ.

- ω: Velocidade do eixo de armadura
- T_g, T_f, T_j, T_L: torque gerado, atrito, inercia e na carga.

Assim obtemos a seguinte equação, que descreve a tensão na malha:

$$Vi(t) = R_a \times I_a + L_a \times \frac{di_a(t)}{dt} + v_g(t) \tag{2}$$

a partir da transformada de $Vi(t)$ podemos estabelecer uma FT entre tensão e a corrente. alguns parâmetros indicados pela equação diferencial podem ser desprezados a titulo de calculo, simplificando mais ainda a equação

Outra forma de encontrar FT é pelos sinais $u(t)$ e $y(t)$, onde não precisaremos das equações diferencias que regem um motor DC. a FT só funciona para intervalos lineares. A função de transferência do sistema e estudo é de primeira ordem. Estamos interessados em analisar a relação entre a tensão de entrada e saída do sistema motor gerador.Esse sistema é dado pela seguinte equação.

$$H(s) = \frac{Y(s)}{U(s)} = \frac{k_m}{\tau s + 1} \tag{3}$$

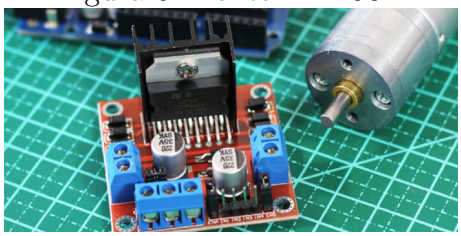
Onde temos que:

- k_m: é o ganho do motor;
- τ: é a constante de tempo do motor;
- $Y(s)$: é a tensão de saída do sistema;
- $U(s)$: é a tensão de entrada do sistema;

1.4 Driver Ponte H L298N

Na figura 6 é mostrado o Driver L298N utilizado na bancada.

Figura 6: Ponte H L298N



Fonte: www.fliflop.com

Esse módulo é projetado para controlar cargas indutivas como relés, solenoides, motores DC e motores de passo, permitindo o controle não só do sentido de rotação do motor, como também da sua velocidade, utilizando os pinos PWM do Arduíno:
O Driver L298N é o elo de ligação entre a Fonte de Alimentação e o Motor DC. O valor de tensão e a polaridade são gerados com base na informação do sinal PWM e dos sinais de sentidos. O Sinal de PWM informa a magnitude da tensão a ser entregue para o Motor DC. Enquanto os pinos de sentido indicam a polaridade da tensão gerada e como consequência o sentido de rotação do eixo do Motor.

1.4.1 Sinal PWM

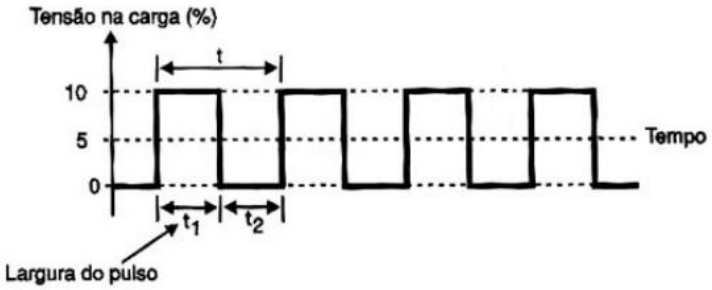
O sinal PWM do inglês *Pulse Width Modulation*, nos possibilita fazer o controle da potência aplicada em um sistema, refere-se ao conceito de pulsar rapidamente um sinal digital em um condutor. Além de várias outras aplicações. Através do sinal PWM podemos ajustar a temperatura de um sistema, luminosidade de uma lampada, simular uma tensão estática variável e é comumente aplicada no controle de motores elétricos.

Como funciona o PWM ?

PWM (Pulse Width Modulation) ou Modulação de Largura de Pulso, é utilizado em estudos relacionados a sistemas digitais, consiste em uma técnica para variar o valor médio de uma forma de onda periódica. Assim é possível controlar o nível de chaveamento de uma onda no tempo, alternando os níveis lógicos e mantendo sua frequência,
Dessa forma, seria como se um circuito, cuja fonte de tensão fornece 10 volts, tivesse uma chave que fica abrindo e fechando em uma determinada frequência. Veja a onda que será formada no circuito abaixo:

O PWM funciona exatamente dessa forma, entretanto digitalmente. E assim é possível regular o tempo em que o sinal estará em nível lógico alto em uma determinada frequência.

Figura 7: Tensão x Tempo



Fonte: https://athoselectronics.com

O tempo em que o sinal está em nível lógico alto, é chamado de **Duty Cycle**, ou, **Ciclo Ativo.**

$$P = \left(\frac{t_{high}}{t_{high} + t_{low}}\right) \times V \times I$$

- P : Potência Média (Watts);
- t_{high}: Tempo em nível logico alto (segundos ou milissegundos);
- t_{low}: Tempo em nível logico baixo (segundos ou milissegundos);
- V: Tensão fornecida pela fonte (Volts);
- I: Corrente (Amperes);

Funcionamento do PWM do Arduíno

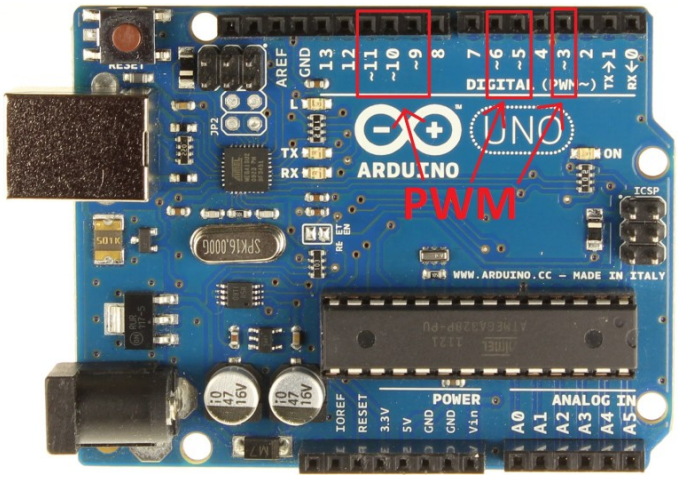
No Arduíno, podemos encontrar algumas portas PWM, a quantidade delas depende do modelo do Arduíno. Existem Arduinos com mais portas PWM, e outros com menos, sendo que diversos sensores ou módulos necessitam obrigatoriamente de portas PWM para funcionar. Veja abaixo as portas PWM de um Arduíno Uno:

A função PWM do Arduíno - analogWrite ()

Dessa forma, para controlar os ciclos ativos nas portas PWM do Arduíno, existe uma função específica em sua programação, o analogWrite ().

- Sintaxe: analogWrite(porta , valor de 0 a 255);
- Exemplo: analogWrite(9, 200);

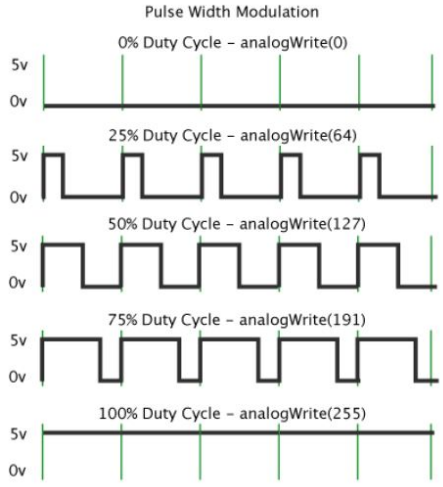
Figura 8: Pinos de conexões do PWM no Arduíno Uno



Fonte: https://athoselectronics.com

Utiliza-se a função analogWrite() para determinar o tempo de um ciclo ativo em uma saída PWM. Assim, o tempo do ciclo ativo é controlado com um valor de 0 a 255, sendo 255 o máximo e sempre ativo.
Veja na imagem abaixo, o comportamento da onda de acordo com o valor dado na programação (de 0 a 255):

Figura 9: Exemplo de modulação da saída PWM do Arduíno



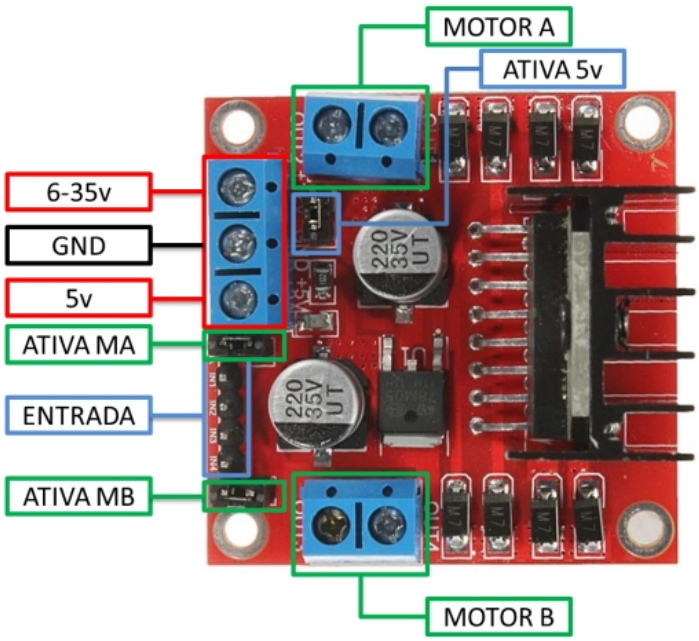
Fonte: https://athoselectronics.com

1.4.2 Funcionamento:

Especificações básicas do Driver L298N

- Tensão de Operação: de 4V à 35V
- Controle de até 2 motores DC
- Tensão lógica: 5V

Figura 10: Conexões do Driver L298N



Fonte:www.fliflop.com

- (**Motor A**) e (**Motor B**) se referem aos conectores para ligação de até 2 motores DC
- (**Ativa MA**) e (**Ativa MB**) – são os pinos responsáveis pelo controle PWM dos motores A e B. Se estiver com jumper, não haverá controle de velocidade, pois os pinos estarão ligados aos 5v. Esses pinos podem ser utilizados em conjunto com os pinos PWM do Arduíno.
- (**Ativa 5V**) e (**5V**) – Este Driver Ponte H L298N possui um regulador de tensão integrado. Quando o driver está operando entre 6-35V, este regulador disponibiliza uma

saída regulada de +5v no pino (5v) para um uso externo (com jumper), podendo alimentar por exemplo outro componente eletrônico. Portanto não alimente este pino (5v) com +5v do Arduíno se estiver controlando um motor de 6-35v e jumper conectado, isto danificará a placa. O pino (5v) somente se tornará uma entrada caso esteja controlando um motor de 4-5,5v (sem jumper), assim poderá usar a saída +5v do Arduíno.

- (**6-35v)** e (**GND**) – Aqui será conectado a fonte de alimentação externa quando o driver estiver controlando um motor que opere entre 6-35v. Por exemplo se estiver usando um motor DC 12v, basta conectar a fonte externa de 12v neste pino e (GND).
- (**Entrada**) – Este barramento é composto por IN1, IN2, IN3 e IN4. Sendo estes pinos responsáveis pela rotação do Motor A (IN1 e IN2) e Motor B (IN3 e IN4).

Na bancada em estudo não utilizaremos Ativa MB, pois estamos trabalhando apenas com a aplicação de tensão em um motor, no caso motor B.

A tabela abaixo mostra a ordem de ativação do Motor A através dos pinos IN1 e IN2

Figura 11: Sentido de rotação

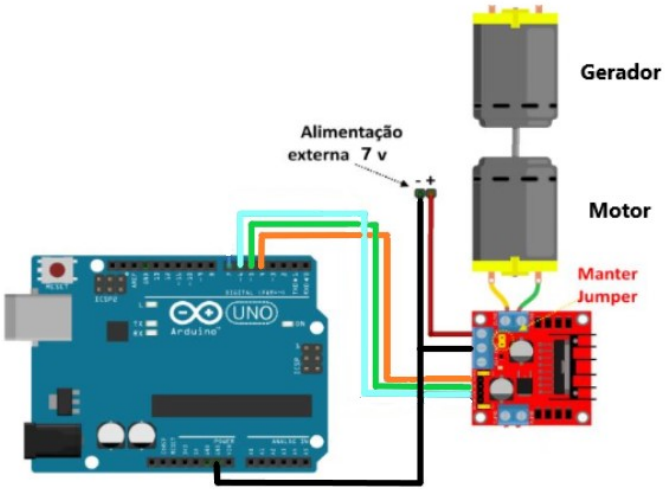
MOTOR	IN1	IN2
HORÁRIO	5v	GND
ANTI-HORÁRIO	GND	5v
PONTO MORTO	GND	GND
FREIO	5v	5v

Fonte: www.fliflop.com

O circuito utiliza alimentação externa de 7V DC. Nesse caso precisamos colocar o jumper em Ativa 5v:

- O fio laranja fraca representada a entrada **PWM** do Arduíno
- O fio verde representa o **IN1**
- O fio azul representa o **IN2**

Figura 12: Conectando ao Arduíno

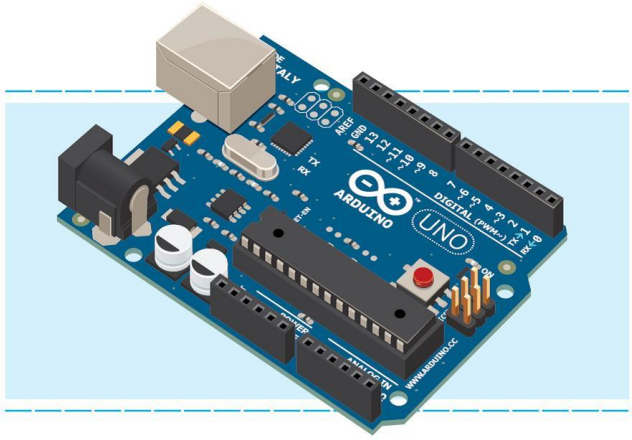


Fonte: www.fliflop.com

1.5 A placa Arduíno Uno

O arduino é uma plataforma de prototipagem e de hardware livre. O arduíno é utilizado como um microcontrolador Antel, tendo um circuito de entrada/saída que pode ser conectada a um computador via cabo USB. É constituido por: possuir um conector de fonte de energia de 7-12V, Porta USB, Botão de Resert, LED's TX e RX, LED Pino 13 é o é o único LED constituido no seu Arduíno Uno, Pinos Digitais, Microcontrolador ATMEGA, Power LED, Pinos GND, 5V e 3.3V e Analoc in.

Figura 13: Imagem ilustrativa da placa Arduíno UNO



fonte: Adaptado pelo Autor

Com a utilização do algoritmo na plataforma do Arduíno, faça as ligações dos pinos dados no código com o Arduíno, lembre - se que o Arduíno suporta no máximo 5V.

1.5.1 Programa Arduíno

O programa do Firmware do Arduíno pode ser analisado em três partes:

Parte 01:

Nessa sequência, são definidos os pinos e os tipos das variáveis manipuladas. Por exemplo, o comando: $int \quad pinoAD = A3$ cria a constante "pinoAD" do tipo inteiro para receber os valores do pino A3.

```
// recebe o sinal de saída do condicionador de sinais
#define pinoAD     A3

// saída do sinal PWM para a ponte H
#define pinoPWM    9

// representam as portas que controlaram o sentido de giro do motor
//representados na Ponte H como IN1 e IN2
#define pinoSent1  7
#define pinoSent2  8
```

Agora são definidas as variáveis do tipo inteiro, que auxiliaram na lógica de programação.

```
// Recebe o valor vindo da comunicação serial (computador - Arduíno)
int r = 0;

// Armazena o valor PWM, em uma escala entre
// 0 a 255 bits
int valorPWM = 0;

// Armazena o valor vindo do pinoAD
int valorAD = 0;
```

Parte 02: Nessa parte do programa, é realizada a configuração dos pinos (setup). A função $pinMode(pino, modo)$ configura o 'pino' especificado no 'modo' determinado. Por exempo, pinMode(pinoSentido1, OUTPUT) configura o pino definido para variável piniSentido1 na modalidade saída.

```
void setup() {
// define o pinosent1, pinosent2 e //pinoPWM como saída.
  pinMode(pinoSent1, OUTPUT);
  pinMode(pinoSent2, OUTPUT);
  pinMode(pinoPWM, OUTPUT);
// Inicializa a comunicação serial, com //uma taxa de transmisão de 9600
// com um timeout de 5 ms
  Serial.begin(9600);
  Serial.setTimeout(5);
  }
```

Parte 03: Laço principal do programa, quando energizado o Arduino fica repetindo o laço principal interruptamente.

```
void loop(){
// Serial.available() retorna a quantidade de bits na porta serial
  if (Serial.available() > 0){
// Armazena o valor recebido da comunicação do serial
    r = Serial.parseInt();
 // verifica se o valor vindo da serial é maior que 0.
 // se for verdade, somente os valores maiores que 0, entraram no primeiro if
    if (r>0){
// o sinal r>0 tem que se escrito em uma escala de 0 a 255
// valor de variação do sinal PWM
// procure a documentação da função map()
       valorPWM = map(r,0,15,0,255);
// faz com que o motor MA gire no sentido anti- horário
       digitalWrite(pinoSent1, LOW);//sentido + : pinoSent1 LOW e pinoSent2 HIGH
       digitalWrite(pinoSent2, HIGH);
// escreve o valor PWM no pinoPWM
       analogWrite(pinoPWM, valorPWM);
       }
// segue-se a mesma lógica para r<0
// mudando apenas os valores na função map e o sentido de rotação
     else if(r<0){
       valorPWM = map(r,-15,0,255,0);
```

```
        digitalWrite(pinoSent1, HIGH);
        digitalWrite(pinoSent2, LOW);
        analogWrite(pinoPWM, valorPWM);
        }
// caso r = 0 o motor para
       else{
        //valorPWM = 0;
        digitalWrite(pinoSent1, LOW);
        digitalWrite(pinoSent2, LOW);
        //analogWrite(pinoPWM, valorPWM);
        }
//
        valorAD = analogRead(pinoAD);
        Serial.println((valorAD*5)/1023);
        //Serial.println(valorAD);
  }
             }
```

A comunicação entre o PC e a placa Arduino é realizada via comunicação Serial. No Arduino, as funções *Serial.begin(boudRate)* e *Serial.setTimeout()* são utilizadas para configurar a comunicação. No exemplo, os valores de $boudRate = 9600$ e $timeOut = 5$. Uma vez configurada a comunicação Serial, o Arduino aguarda a chegada de dados na porta serial. Utiliza-se a função *Serial.available()* para checar se há dados na porta. Caso afirmativo, a função *Serial.parseFloat()* é utilizada para ler os dados. Os dados contém a informação do valor do sinal PWM a ser gerado pela placa Arduino. Um sinal PWM é gerado no pino indicado, $pinoPWM$.

O Arduino recebe no $pinoAD$ o valor de tensão, máximo de 5V com resolução de 10 bits, correspondente a versão filtrada da tensão gerada pelo Gerador. Após a leitura da porta analógica, o valor é enviado para o PC.

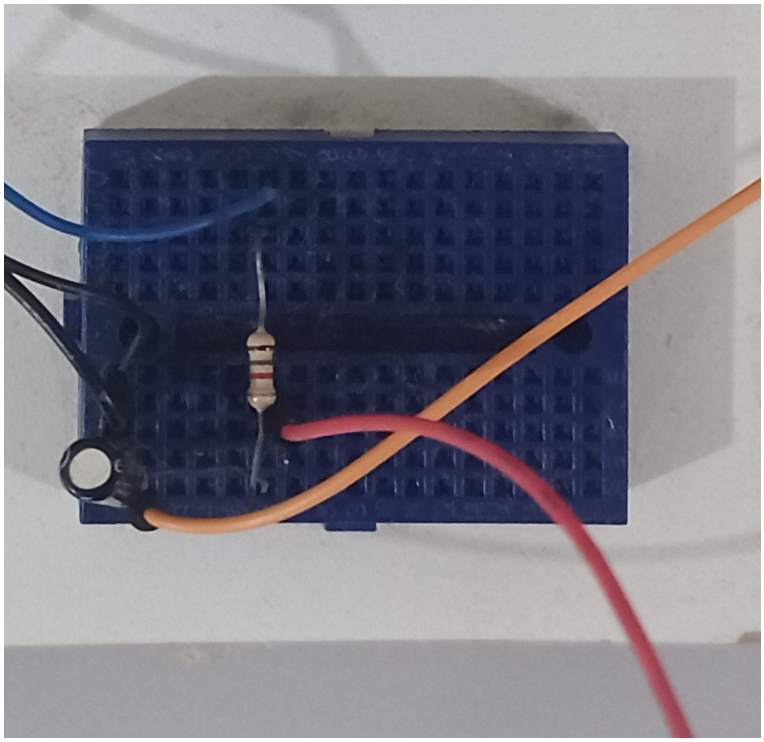
Código Arduino Simples

```
 // Bancada de Controle de velocidade: Motor-Gerador
// UFPA - Campus Tucuruí
// Monitoria de Sistemas de Controle para Engenharia
// - PGRAD - MONITORIA 03/2020
// Coodenador: Cleison Daniel Silva
// Bolsista: Felipe Silveira Piano
// Data: 26/09/2020

#define pinoAD     A1
#define pinoPWM    9
#define pinoSent1  7
#define pinoSent2  8

int valorAD = 0;
int valorPWM = 0;

void setup() {
  pinMode(pinoSent1, OUTPUT);
  pinMode(pinoSent2, OUTPUT);
  pinMode(pinoPWM, OUTPUT);
  Serial.begin(9600);
  Serial.setTimeout(5);
}
void loop()  {
  if (Serial.available() > 0) {
    valorPWM = Serial.parseInt();
    digitalWrite(pinoSent1, HIGH);
    digitalWrite(pinoSent2, LOW);
    analogWrite(pinoPWM, valorPWM);
    valorAD = analogRead(pinoAD);
    Serial.println(valorAD*5./1023.);
    //Serial.println(map(valorAD,0,5,0,255));
  }
}
```

1.6 Filtro RC Série

Filtro utilizado é um Passa-baixa, constituído por um resistor e um capacitor em série. "Lembre - se que o capacitor em frequência altas funciona como um curto circuito e para frequências baixas como circuito aberto".

Figura 14: Filtro montado da Bancada. No fio azul, chega a tensão vinda do gerador (sinal a ser filtrado), os fios laranja e preto estão em curto e são o terra do circuito, o fio vermelho contem o sinal filtrado e vai para a placa arduino.



fonte: Autor

É importante destacar a frequência de corte, f_c dada em Hz, do filtro passivo passa baixa. O comportamento esperado do filtro RC é rejeitar as componentes de frequências acima da frequência de corte. O projeto do filtro RC consiste em especificar os valores dos componentes resistivo (R) e capacitivo (C), tal que a frequência de corte atenda a expressão:

$$\boxed{f_c = \frac{1}{2\pi RC}} \tag{4}$$

Para realizar o projeto do filtro RC, para a bancada Motor-Gerador a técnica adotada FFT (Transformada Rápida de Fourier) e verificar o sinal sem o filtro e aplicar a técnica de modelagem de converter o sinal para frequência para verificar a frequência de corte do filtro

para eliminar os ruídos, um problema que você terá que perceber é verificar os níveis de tensão do arduino.

O projeto de uma filtro RC é bem simples. Consiste, basicamente, em especificar os valores do resistor e do capacitor. Esses valores são projetados para atender a frequência de corte adequada à aplicação do filtro.

Por exemplo, considere o sinal senoidal em tempo discreto $x[n] = A\sin(2\pi fnT_s - \theta) + setpoint$. Em que A representa a amplitude do sinal, f é a frequência em Hz, n é o tempo discreto, T_s é o período de amostragem, θ é a fase do sinal e *setpoint* é o nível DC do sinal.

Em Python, segue um trecho de código para gerar o sinal senoidal:

```
N = 500                          # N mero de amostras
n = np.arange(0,N-1)             # Tempo discreto com N amostras
A = 0.5                          # Amplitude
Fs = 250                         # Frequencia de amostragem, Fs = 1/Ts
f = 3                            # frequencia do sinal x[n]
setpoint = 3                     # Nivel DC
Teta = np.pi/6                   # Fase de x[n]

# Sinal x[n]
x = A*np.sin(2*np.pi*f*n/Fs - Teta) + setpoint
# Sinal x[n] + ruido
z = x + 0.05*np.random.randn(N-1)
```

A Figura contendo o sinal senoidal é apresentada a seguir:

No sinal apresentado na Figura 15 tem-se uma senoide com $f = 3Hz$ contaminada com ruido gaussiano. O projeto de um filtro RC deve preservar a componente principal do sinal, $f = 3Hz$ e atenuar as componentes acima desse valor. Nesse sentido, a frequência de corte adotada é $f_c = 10Hz$.

A Figura 16 apresenta o espectro do sinal $x[n]$, com e sem ruido. As componentes com maior energia, em $x[n]$ consistem nas frequências $f = 0Hz$, nível DC do sinal e na $f = 3Hz$. É evidenciado, também, a contribuição do ruido adicionado componentes de frequência fora da faixa do sinal.

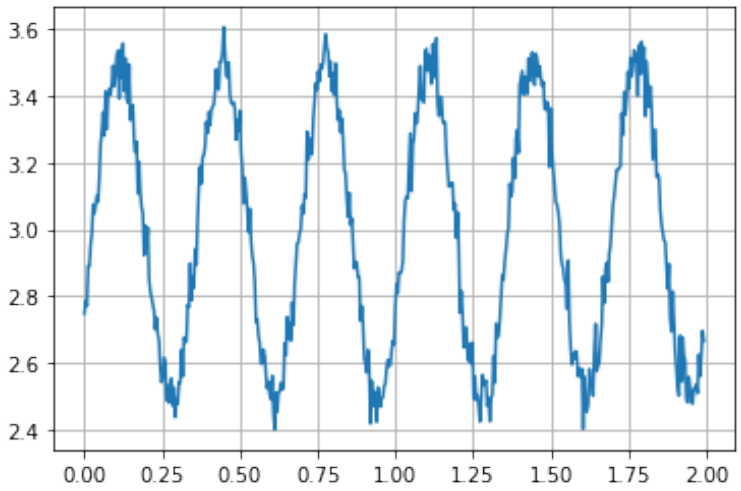
Um filtro RC apresenta função de transferência de primeira ordem,

$$H(s) = \frac{\omega_c}{s + \omega_c}$$

Em que ω_c é a frequência de corte em rad/s.

A Figura 17 apresenta a resposta em frequência (magnitude e fase) do filtro RC para

Figura 15: Sinal Senoidal $x[n]$ contaminado com ruido



fonte: Autor

$f_c = 10Hz$.

Na Figura 18 o sinal após o filtro é mostrado em contraste com o sinal antes do filtro RC. É possível observar que a versão filtrada do sinal apresenta um atraso (defasamento em fase) em relação ao sinal original. No entanto, as componentes DC e em $10Hz$ estão preservadas.

A Figura 19 apresenta do espectro dos sinais: original, contaminado por ruido e filtrado.

1. Exemplo:

Projete um filtro passa baixa RC, para uma frequência de corte de 1000Hz.

1. Primeiramente determine um Capacitor (C) que você deseja usar no projeto para depois dimensionar qual o resistor (R) irá usar.

2. Utilizando um capacitor na faixa:

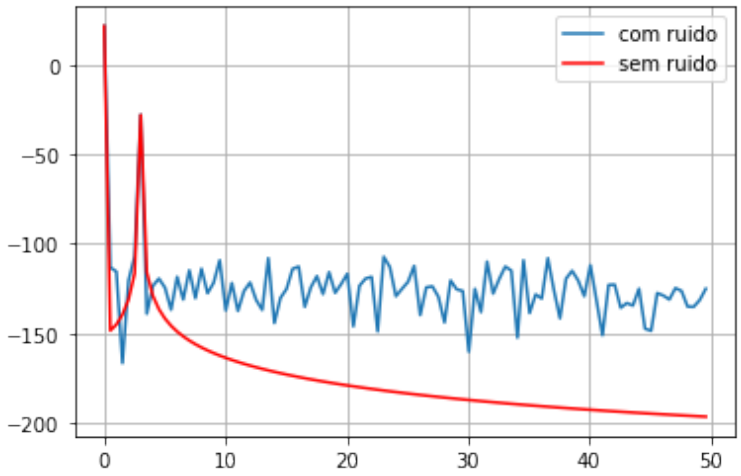
$$C = 0,1\mu F$$

3. dimensionamento do resistor .

$$\boxed{R = \frac{1}{2\pi C f_c}} \tag{5}$$

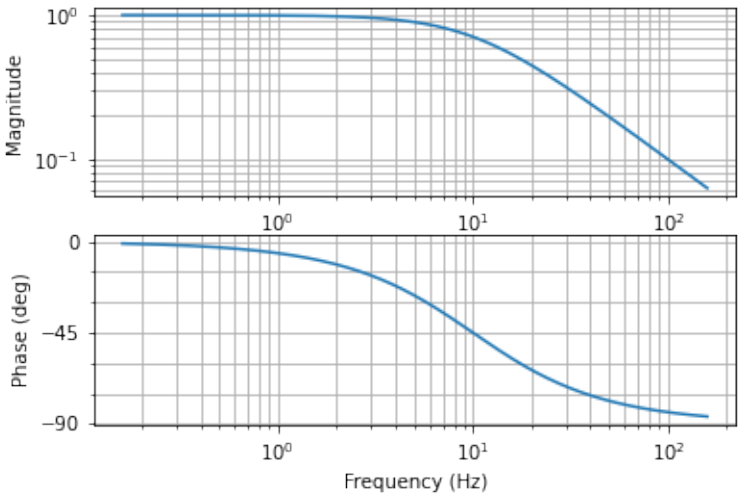
$$R = \frac{1}{2\pi * 1000 * 0,1 * 10^{-6}} = 1591,549\Omega$$

Figura 16: Espectros do sinal senoidal $x[n]$, com e sem ruido



fonte: Autor

Figura 17: Resposta em Frequência de $H(s)$



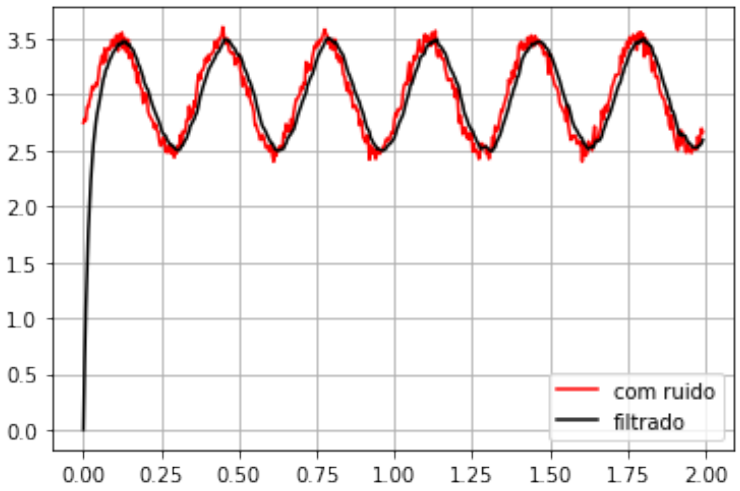
fonte: Autor

Após ter realizado a projeção, substitua por um resistor comercial.

$$R = 1600\Omega$$

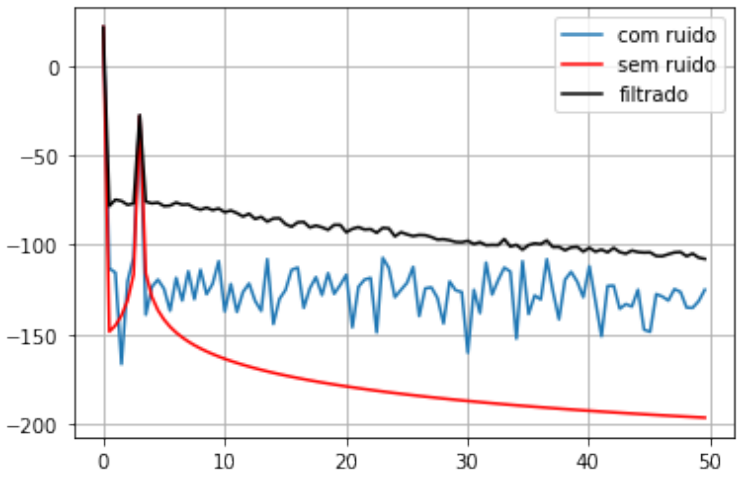
Para o projeto ser realizado, os componentes dentro do laboratório para o projeto de filtro.

Figura 18: Sinal $x[n] filtrado$.



fonte: Autor

Figura 19: Espectro do Sinal $x[n]$ filtrado.



fonte: Autor

Um sinal ruidoso, o filtro tem a função de eliminar esses ruidos.

A frequência de corte f_c ela tem que ser $-3dB$ do módulo do ganho.

$$H(j\omega) = \frac{a}{j\omega + a}$$

em que ω é dado em rad/s e $a = \frac{1}{RC}$.

$$|H(j\omega)| = \frac{|a|}{|j\omega + a|} = \frac{1}{|j\frac{\omega}{a} + 1|} = \frac{1}{\sqrt{1 + (\frac{\omega}{a})^2}}$$

Resistor	**Capacitor**
1kΩ	$0.1\mu F$
$1.2k\Omega$	$0.33\mu F$
$1.5k\Omega$	$0.47\mu F$
1.8kΩ	$1\mu F$
2kΩ	$2.2\mu F$
2.2kΩ	$4\mu F$
2.7kΩ	$10\mu F$
3.7kΩ	$22\mu F$
3.9kΩ	$27\mu F$
4.7kΩ	$33\mu F$
6.8kΩ	$0.33\text{n}F$
8.2kΩ	$2.2\text{n}F$
10kΩ	3.3n F
12kΩ	3.9n F
15kΩ	5.6n F
18kΩ	10n F
22kΩ	$27\text{n}F$

Tabela 1: Componentes Presente no Laboratório

Questionário

1. Por que utilizar um filtro passa baixa e não o passa alta? justifique.

2. Após verificar na bancada os elementos do filtro, faça uma simulação da resposta em frequência e verifique se está frequência de corte simulada está compatível com a calculada?

1.7 Software Visual Studio Code e Google colab

Para que possamos manipular analisar o nosso sistema motor-gerador, utilizaremos dois ambientes de progamação, **VScode** e **Google colab** ambos nos possibilitam trabalhar com a linguagem python.

- **VScode** será utilizado para que possamos armazenar os sinais aplicados ao sistema $u(t)$ e $y(t)$.
- **Google colab** será onde faremos o processo de identificação do sistema em estudo

1.7.1 Interface do VScode

Em seu coração, Visual Studio Code é um editor de código. Como muitos outros editores de código, o VS Code adota uma interface de usuário comum e layout de um explorador à esquerda, mostrando todos os arquivos e pastas que você tem acesso, e um editor à direita, mostrando o conteúdo dos arquivos que você abriu.

Código no VScode

1.7.2 Interface do Google colab

1.8 Comentários Cleison

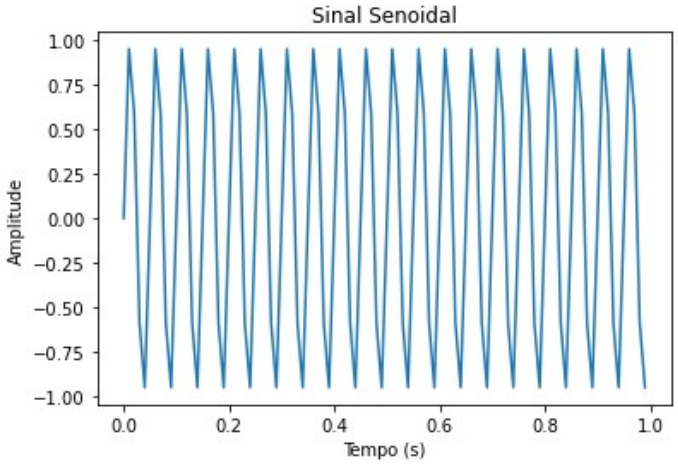
- Computador: apresentar o código em python escrito para controlar o Motor-Gerador;
- Arduino: placa utilizada para a interface entre o computador e o driver de corrente.
- Circuito atuador
- Motor-Gerador: apresentar a planta do ponto de vista de controle. A planta pode ser vista como dois sistemas de 1 ordem em cascata.
- Filtro RC: entender da função de transferência do Filtro RC, a resposta em frequência, o projeto do filtro, tabela de valores de resistores e capacitores comerciais, etc.

```
        """
Bancada Motor-Gerador
UFPA - Campus Tucuru
Monitoria de Sistemas de Controle para Engenharia
 PGRAD - MONITORIA 03/2020
Coodenador: Cleison Daniel Silva
Bolsista: Felipe Silveira Piano
Data: 27/09/2020
"""

import serial
import numpy as np
import matplotlib.pyplot as plt
import time as t
from scipy.signal import square,sawtooth


################################################

numAmostras = 800
tempo = np.zeros(numAmostras)
y = np.zeros(numAmostras)

Ts = 0.02

fre = 0.5
Amplitude = 0
setpoint = 3.5
#a = 2*np.ones(int(numAmostras/2))
#b = 4*np.ones(int(numAmostras/2))
#u = np.concatenate([a,b]) #degrau
r = np.zeros(numAmostras)
toc = np.zeros(numAmostras)
#########################
```

```
for n in range(numAmostras):
    r[n] = Amplitude*square(2*np.pi*fre*n*Ts) + setpoint
    #r[n] = Amplitude*sawtooth(2*np.pi*fre*n*Ts) + setpoint
    #r[n] = Amplitude*np.sin(2*np.pi*fre*n*Ts) + setpoint
    #r[n] = u[n]

print('\nEstabelecendo conex o.')
conexao = serial.Serial(port='COM6', baudrate=9600, timeout=0.005)

t.sleep(1)
print('\nIniciando coleta.')

for n in range(numAmostras):
    tic = t.time()

    if (conexao.inWaiting() > 0):

        y[n] = conexao.readline().decode()

    u = (r[n]*255)/7
    conexao.write(str(round(u)).encode())

    t.sleep(Ts)

    if (n > 0):
        tempo[n] = tempo[n-1] + Ts
    toc[n] = t.time() - tic
conexao.write('0'.encode())
print('\nFim da coleta.')
conexao.close()
print('media=',np.mean(r))

print('\nPer odo real:', np.mean(toc))
print('Nivel_DC:', np.mean(y[tempo>2]))

plt.figure(figsize=(10,10))
```

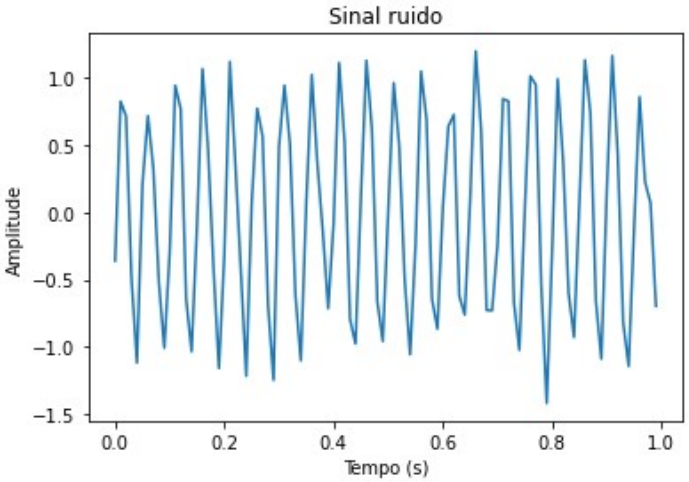
```
plt.subplot(211)
plt.plot(tempo,r,'-b',linewidth=1.2)
plt.xlabel('Tempo(s)')
plt.ylabel('Tens o (V)')
plt.grid()
plt.title('Onda Quadrada - Malha Aberta')
plt.legend(loc='lower right', labels=('Sinal de Entrada','Sinal de Sa da')

plt.subplot(212)
#plt.plot(tempo,r,'-b',tempo,y,'-r',linewidth=1.2)
plt.plot(tempo,y,'-ro',linewidth=1.2)
plt.xlabel('Tempo(s)')
plt.ylabel('Tens o (V)')
plt.grid()
# plt.title('Tens o de Sa da - Malha Aberta')
plt.show()

dados=np.stack((tempo,r,y),axis=-1)

# np.savetxt("C:/Users/DTIC/Desktop/Laborat rio de Controle/Motor_Gerador_
\end{}
```

Figura 20: Sinal Amostra



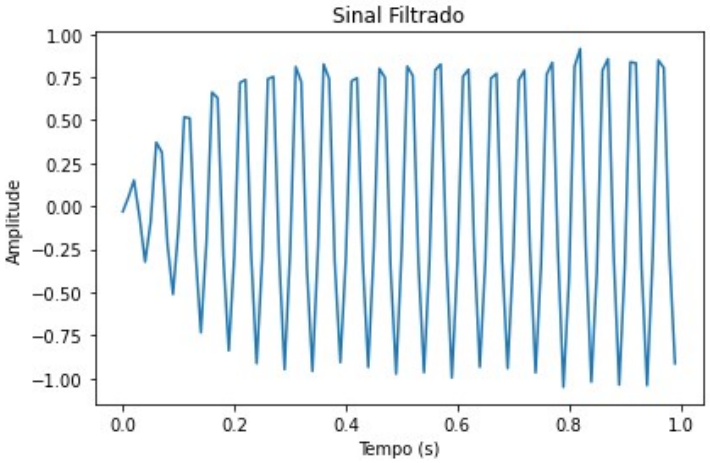
fonte: Simulação Python

Figura 21: Sinal Ruido



fonte: Simulação Python

Figura 22: Sinal Filtrado



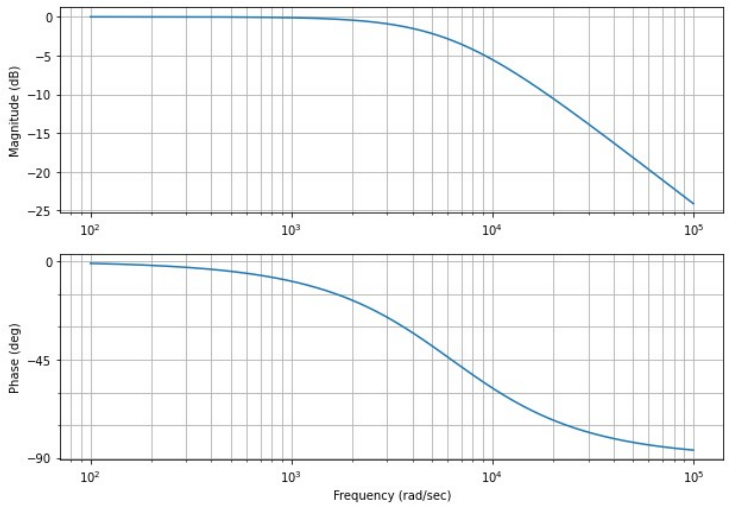
fonte: Simulação Python

Figura 23: Código em python para verificar a resposta em frequência (rad/s)

```
R = 1600
C = 0.1*10**-6
a = 1/(R*C)
T = ct.tf([a],[1,a])
plt.figure(figsize=(10, 7))
ct.bode(T,Hz=False);
```

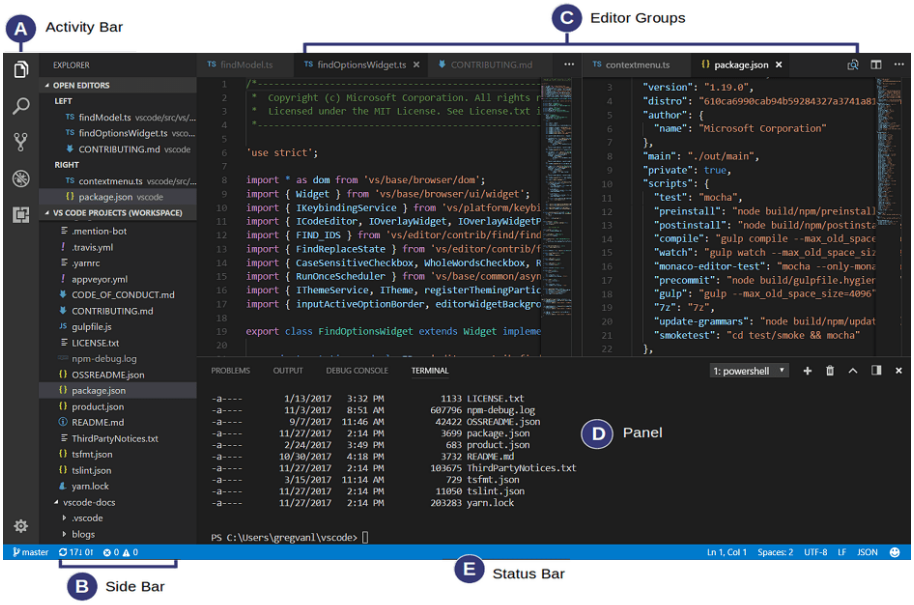
fonte: Autor.

Figura 24: Resposta em frequência de Filtro RC passa baixas



fonte: Autor.

Figura 25: interface VScode



fonte: https://code.visualstudio.com

Figura 26: interface google colab

fonte: https://colab.research.google.com