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

Laboratório de Sistemas de Controle: Módulo Carrinho de impressão

Monitora: Leilane de Jesus dos Anjos
Orientador: Raphael Teixeira Barros

Tucuruí-PA
2024

Sumário

1 Introdução:

A implementação do conhecimento teórico com a prática apresenta um papel fundamental na construção da aprendizagem de qualquer aluno, uma vez que, além de solidificar o entendimento sobre um determinado assunto a partir da experiência direta com problemas do mundo real ele possibilita também o desenvolvimento de habilidades como a medição, observação, análise de dados e resolução de problemas a partir do pensamento crítico.

Neste contexto, a bancada carrinho de impressão surge como um intermediador na matéria de laboratório de sistemas de controle para unir teoria á prática. A estrutura base deste sistema dinâmico didático é facilmente encontrado em qualquer impressora da qual possui a função de rastrear e posicionar-se com precisão em um ponto pré-definido, para isto utiliza-se de conhecimentos da área de controle e de programação como C++ e Python.

1.1 Objetivos:

1.1.1 Objetivo Geral:

- O objetivo geral deste trabalho é projetar e implementar um sistema dinâmico que possa ser utilizado em laboratório por alunos de graduação em Engenharia Elétrica para a realização de experimentos e estudos em matérias como Análise de Sistemas de Lineares e Sistemas de Controle.

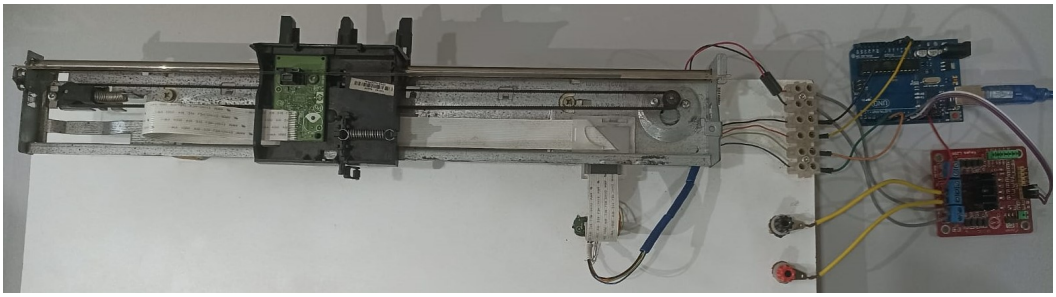
1.1.2 Objetivo Específico:

- Estudar detalhadamente cada componente do sistema, incluindo atuadores, sensores e interfaces de comunicação;
- Projetar e implementar as conexões físicas e lógicas entre os componentes do sistema, garantindo a integridade e a funcionalidade do sistema completo;

2 Bancada Módulo de Impressão:

A bancada carrinho de impressão, mostrada na Figura (), é um sistema dinâmico didático projetado para servir como caso de estudo na matéria de Laboratório de sistemas de controle, a sua estrutura base consiste da junção de um motor DC conectado por uma correa de polímero ao carrinho.A implementação do Driver L298N, um Arduíno UNO, uma fita codificada e um detector óptico (*encoder linear*) na planta permite deixar o sistema operante de modo que ao receber um sinal de tensão, $x(t)$, nos terminais de alimentação, o motor DC gira entorno do seu próprio eixo fazendo com que a correa de polímero e consequentemente o carrinho se movimente, com o *encoder* energizado e o carrinho em movimento é coletado o sinal de posição, $y(t)$, com o auxílio de uma faixa codificada.

Figura 1: Bancada carrinho de impressão.

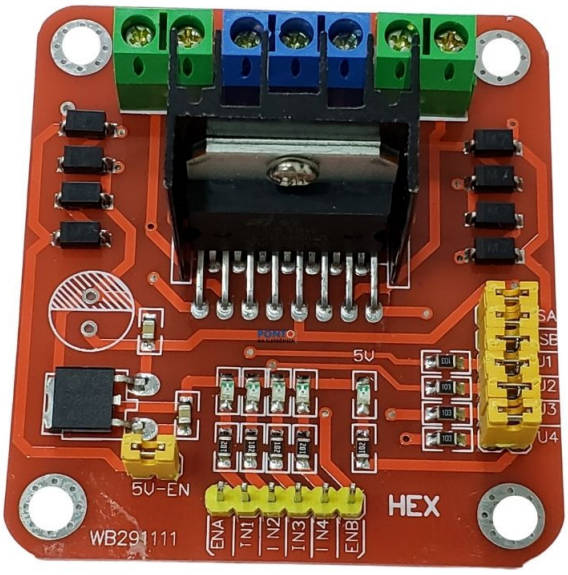


2.1 Descrição dos elementos do sistema:

2.1.1 Driver L298N:

O driver L298N, ponte H ou simplemente driver de potência, como mostra a Figura 2, é o componente eletrônico responsável por fazer a ligação entre a fonte de tensão e o motor DC, possui a função de realizar o controle do sentido de rotação e da velocidade do motor a partir de dois sinais sendo um o sinal que determina o sentido de rotação do eixo do motor e o outro o sinal PWM.

Figura 2: Drive Ponte H dupla L298N



Fonte: Ponto da Eletrônica[1].

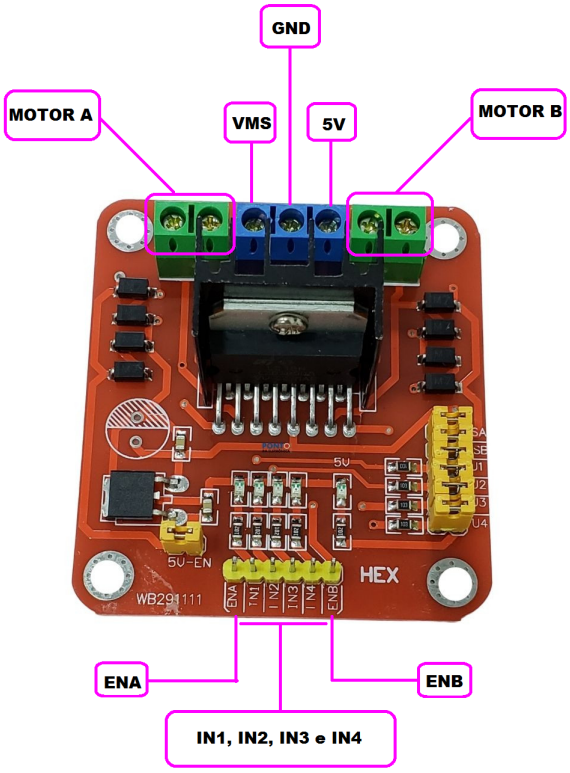
O PWM do inglês *Pulse Width Modulation*, ou Modulação por Largura de Pulso é uma técnica que tem por finalidade o controle de potência em uma cargar ao variar a largura de pulso de uma forma de onda, ou seja, ao variarmos o seu ***Duty Cycle***, ou Ciclo de Trabalho consequentemente variamos a quantidade de potência entregue ao sistema

Em razão do baixo custo e consumo de energia é comumente realizado o controle da variação de potência de forma digital, nos dias atuais o controlador PWM pode ser encontrado facilmente em microcontroladores como o Arduíno que converte um sinal analógico em um sinal digital (sinal PWM), ao controlar o Ciclo de Trabalho através de um valor entre 0 à 255, logo quanto mais perto de 255 estiver operando o sinal PWM mais próximo de 100% estará o Ciclo de Trabalho, fornecendo assim mais energia ao meu sistema

Para que a ponte H possa realizar tais funções é necessário ter conhecimento de como fazer as devidas conexões, a Figura 3, denota o que cada borne deste componente eletrônico representa, onde:

[1]Disponível em: https://www.pontodaeletronica.com.br/modulo-ponte-h-l298n-controlador-duplo-para-motores.html, Acesso em: 09 de Agosto de 2023.

Figura 3: Bornes da Ponte H



Fonte: Adaptado pelo autor.

- **Motor A** e **Motor B** são os bornes onde os motores DC são conectados, como a bancada carrinho de impressão utiliza somente um motor optou-se pelo **Motor A**;
- **ENA** e **ENB** são responsáveis pelo controle de velocidade do motor;
- **IN1** e **IN2** são responsáveis pelo controle do sentido de rotação do **Motor A**, enquanto que **IN3** e **IN4** são responsáveis pelo sentido de rotação do **Motor B**. Este controle se dar através de um código computacional no Arduíno a partir das condições mostradas na Tabela 1:

Tabela 1: Sentido de rotação do motor DC.

Sentido	IN1	IN2
Horário	HIGH	LOW
Anti-Horário	LOW	HIGH
Freio	HIGH	HIGH
Freio	LOW	LOW

Fonte: Autor.

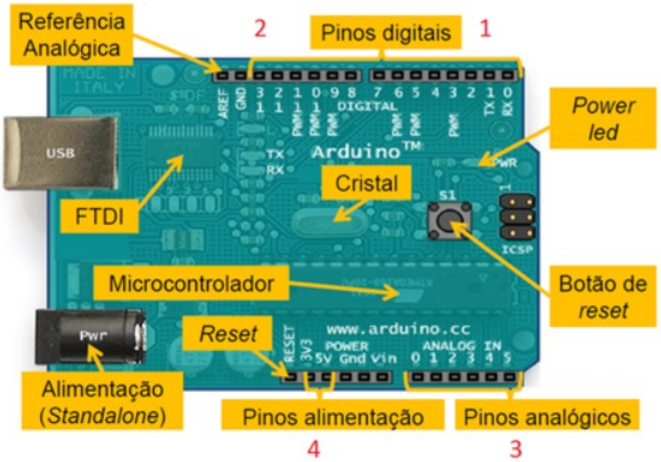
- **VMS** é responsável por permitir uma alimentação externa quanto se necessita energizar um equipamento com mais de 5V;
- O borne **5V** é usado quando se deseja energizar um equipamento com até 5V;
- **GND** é o borne de aterramento da Ponte H;

2.1.2 Arduíno:

É um dispositivo de prototipagem de código aberto constituído de hardware e software. Por apresentar baixo custo e fácil uso é amplamente utilizado na criação de projetos como, por exemplo, óculos anti-sono, termômetro, robô autônomo, automatização de ambientes, entre outros.

O hardware deste dispositivo é composto por diversos componentes tendo como principais um microcontrolador, fonte de alimentação, pinos de conexão como o GND, pinos de voltagem, pinos analógicos e digitais além de um conector USB, botão de resert e um indicador LED de energia, como mostra a Figura 4. Sua programação é realizada via IDE(Integrated Development Environment,ou Ambiente de Desenvolvimento Integrado) da qual utiliza como base a linguagem C/C++ para o desenvolvimento de softwares que serão inseridos na placa Arduíno sendo possível realizar o comando programado.

Figura 4: Bornes do Arduíno.



Fonte: Ilton Barbacena, [2]

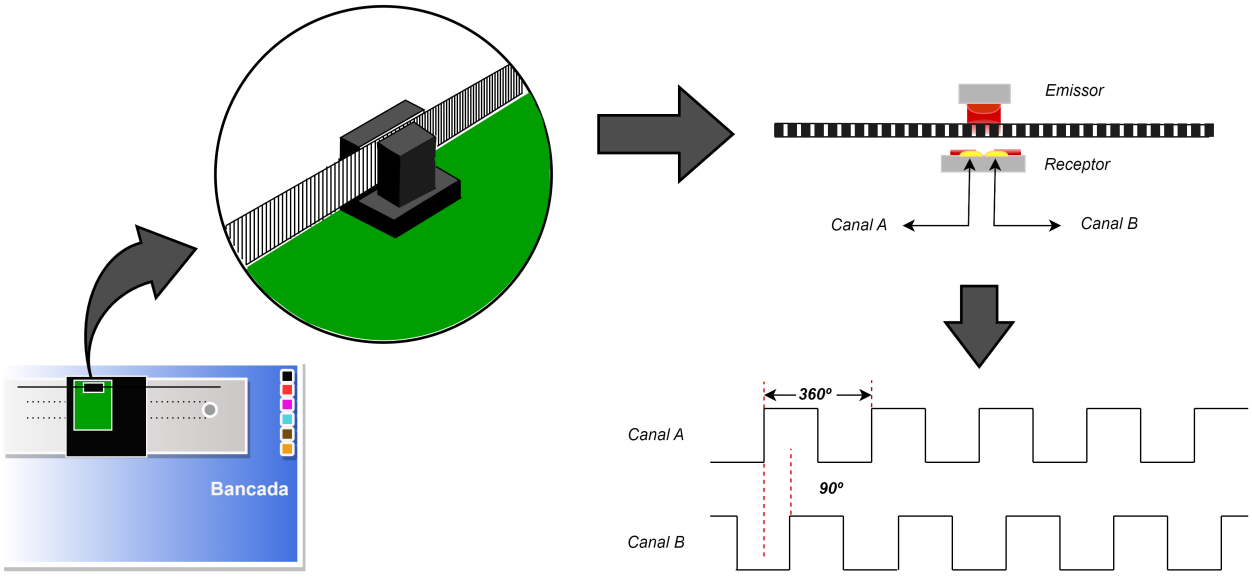
[2]Disponível em: http://iltonluiz.vhost.ifpb.edu.br/expotec/arduino.html, Acesso em: 09 de Agosto de 2023.

2.1.3 Detector óptico:

É um equipamento utilizado para realizar a medição de posição de um objeto, do qual consiste de uma fita codificada fixa e uma parte móvel que possui nele um codificador acoplado composto por dois elementos um emissor e um receptor de infravermelhos, o receptor pode apresentar apenas um canal se realizar a medição de velocidade e dois canais para a medição de posição, como o sistema da qual estamos trabalhando se trata de um carrinho de impressão o receptor do codificador possui dois canais.

Ao ser energizado o emissor do encoder linear projeta uma luz infravermelho nos dois canais do receptor que estão lado a lado o resultado são dois sinais deslocados em um ângulo de 90°, estes dois sinais são enviados para o microcontrolador que a partir de código os converte em valores que indicam posição. Na Figura 5, tem-se uma ilustração destes procedimento descrito.

Figura 5: Ilustração do Encoder linear

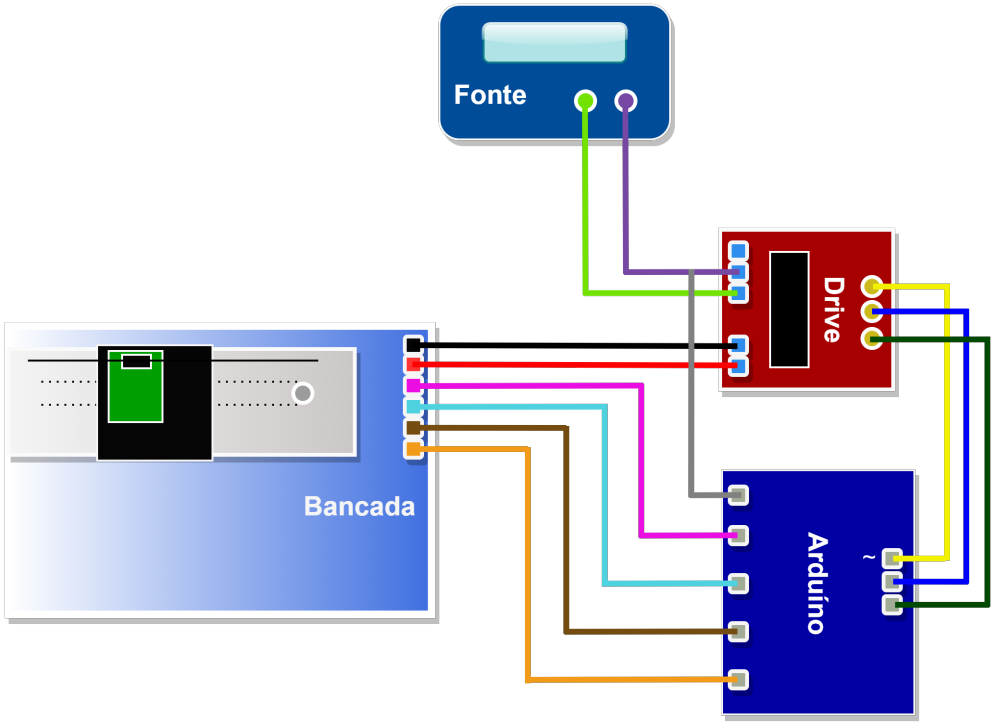


Fonte: Autor.

2.2 Interconexões entre os elementos do sistema:

Tendo conhecimento da funcionalidade de cada elemento que estar implementado na bancada é necessário compreender como eles estão conectados, a Figura 6 denota estas conexões de modo que:

Figura 6: Ilustração das conexões entre os elementos do sistema.



Fonte: Autor.

- ● e ● representa a conexão entre o motor DC e os bornes do **Motor A** do Drive;

- ● e ● representa a conexão entre os pinos **GND** e **5V** do Arduíno e os pinos **GND** e **5V** do encoder linear;

- ● e ● representa a conexão entre o **CANAL A** e **CANAL B** do encoder e os pinos digitais do Arduíno. Onde se é necessário que um dos canais esteja conectada em uma porta digital especifica, da qual dependerá do modelo Arduíno utilizado, para a utilização da função *attachInterrupt()*[3]. A Tabela 2, mostra os modelos de Arduíno e seus respectivos pinos para a utilização da função.

Tabela 2: Pinagem Digital para a função *attachInterrupt()*

Modelos	Pinos Digitais
Uno, Nano, Mini	2, 3
Uno WiFi Rev.2	todos os pinos digitais
Mega, Mega2560, MegaADK	2, 3, 18, 19, 20, 21

Fonte:Autor.

[3]Disponível em: https://reference.arduino.cc/reference/tr/language/functions/external-interrupts/attachinterrupt/

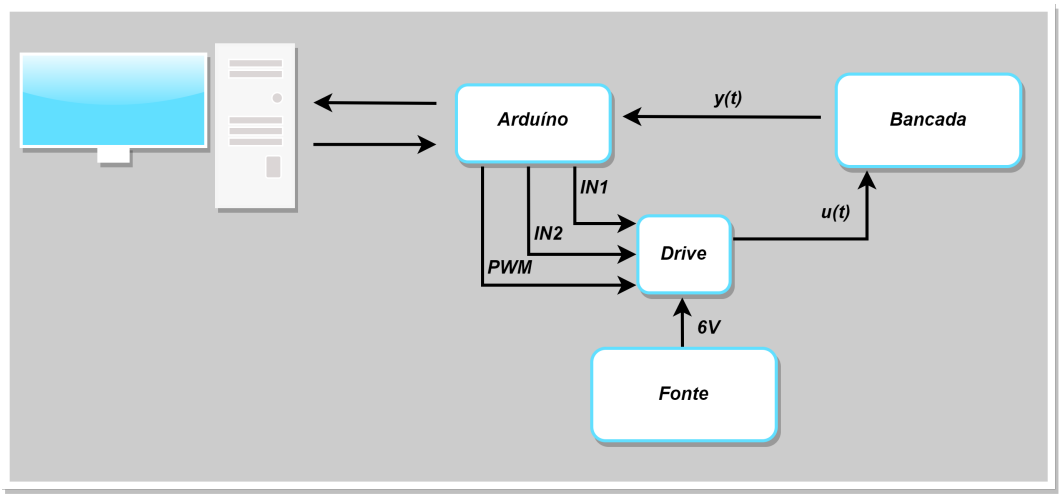
- ● , ● e ● representa a conexão entre o borne PWM, IN1, IN2 do drive e os pinos digitais do Arduíno.

- ● representa a conexão entre o borne VMS do drive e o terminal positivo da fonte de tensão, ● ,● representa a conexão entre o terra do drive, do Arduíno e da fonte de tensão.

2.3 Princípio de funcionamento:

A bancada carrinho de impressão é acionada a partir do programa computacional *python*, que estabelece uma comunicação serial entre o computador e a placa Arduíno UNO. O *loop* de comunicação consiste em: O arduíno recebe do computador o valor de tensão a ser aplicado nos terminais do motor DC, que a partir de um código, Firmware, gravado em sua própria placa , converte o valor de tensão em um valor PWM, que varia entre 0 á 255, este valor é entregue aos pinos **ATIVA MA** do Driver, que juntamente com os pinos **IN1** e **IN2**, definem o sentido de rotação do motor. O Driver, L298N, por sua vez, estabelece a ligação entre a fonte de alimentação e o Motor DC, responsável por movimentar o carrinho de impressão, da qual, possui em sua estrutura um encoder linear energizado com 5V disponibilizados pelo Arduíno. O sinal de posição lido nos terminais do encoder são entregues ao Arduíno e posteriormente enviados ao computador. A Figura 7 mostra o diagrama funcional da bancada.

Figura 7: Diagrama Funcional do sistema dinâmico.



Fonte: Autor

3 Interface Computacional

3.1 Algoritmo computacional IDE Arduíno:

O algoritmo computacional do Arduíno pode ser analisado em três partes, sendo estas dadas por:

- Definição de variáveis e funções

Nesta primeira etapa são definidos os pinos e os tipos de variáveis que serão utilizadas além das funções quando necessários para o código. O *Código auxiliar - Parte 1*, do nosso sistema apresenta inicialmente a definição dos pinos digitais 7, 8 e ∼9 do Arduíno como sendo $S1$, $S2$ e o $pinoPWM$, estes pinos são responsáveis por fornecer o sinal de sentido de rotação do motor ($S1$ e $S2$) e o sinal PWM ($pinoPWM$).

Logo após tem-se definido as variáveis ***ENTRADA_1*** e ***ENTRADA_2*** como os pinos 2 e 3 do Arduíno, estas duas variáveis são responsáveis por receber os sinais coletado dos **CANAIS A** e **B** do encoder linear. Posteriormente são definidos algumas variáveis como inteiras e uma função denominada **inter_entrada_1()**, esta função é responsável por ler os sinais vindo dos **CANAIS A** e **B** ,através da função *digitalRead()* do Arduíno, compara-las por meio de uma estrutura condicional, da qual determinará se ocorrerá o incremento ou decremento da variável *pos_sensor*.

1 Código auxiliar - Parte 1

```
#define S1 7
#define S2 8
#define pinoPWM 9

#define ENTRADA_1 2
#define ENTRADA_2 3

int valorPWM = 0;
volatile int pos_sensor = 0;
int tensao_motor = 0;

void inter_entrada_1(){
    if(digitalRead(ENTRADA_1) == digitalRead(ENTRADA_2)){
      pos_sensor++;
    }
    else {
      pos_sensor--;
    }
}
```

- Configurações:

Nesta etapa são realizados as devidas configurações das pinagens descristas anteriormente ao determinar a função que cada uma irá desempenhar ao longo do código além de realizar a configuração da comunicação entre o computador e a placa Arduíno. Em *Código auxiliar-Parte 2* é realizado este procedimento ao definir o modo de operação de um pino através da função ***pinMode(pino, modo)*** que configura o pino desejado no modo determinado, como por exemplo, ***pinMode(S1, OUTPUT)*** determina o pino S1 como pino de saída. Outras configurações que são feitas neste trecho de código é definir a velocidade de transmissão de dados pela comunicação serial a partir da função ***Serial.begin()*** e um tempo de espera limite para a recepção de dados na porta serial dada pela função ***Serial.setTimeout()***.

2 Código auxiliar - Parte 2

```
20 void setup() {
21   pinMode(pinoPWM, OUTPUT);
22   pinMode(S1, OUTPUT);
23   pinMode(S2, OUTPUT);
24
25   pinMode(ENTRADA_1, INPUT);
26   pinMode(ENTRADA_2, INPUT);
27
28   attachInterrupt(0, inter_entrada_1, CHANGE);
29
30   Serial.begin(9600);
31   Serial.setTimeout(5);
32 }
```

Além das funções descritas anteriormente uma função a mais é acrescida no código fonte para o funcionamento da leitura de posição do sistema BCI, que no caso é a função ***attachInterrupt(pin, ISR, mode)*** que nos permite forçar uma interrupção em uma porta especifica do Arduíno para realizar a leitura do sinal dos canais do encoder linear. Sendo:

- *pin* o número do pino que se deseja realizar a interrupção;

- ISR a função chamada que se deseja realizar ao ocorrer a interrupção;

- *mode* define quando ocorrerá a interrupção, por exemplo, $mode = CHANGE$, fará com que a interrupção seja chamada quando ocorre mudança de nível lógico no pino determinado em *pin*.

Ao definir *pin* normalmente se utilizar a função ***digitalPinToInterrupt(pin)*** mas pode-se utilizar o número de interrupção que estar associado ao número da pinagem que varia de acordo com a placa Arduíno utilizada, como mostra a Tabela 3 .

Tabela 3: Tabela com os valores de interrupção

Quadro	pin = 0	pin=1	pin=2	pin=3	pin=4	pin=5
Uno, Ethernet	2	3	-	-	-	-
Mega2560	2	3	21	20	19	18

- Laço de repetição:

É neste trecho de código que é apresentado os comandos requeridos para o funcionamento do sistema a partir de uma estrutura de condições. Quando o Arduíno é energizado é este trecho de código que fica se repetindo até que ocorra uma interrupção realizada pelo usuário.

O *Código auxiliar - Parte 3*, apresenta esta estrutura de condições, onde o Arduíno aguarda a chegada de informações vindas da comunicação serial e utiliza a função ***Serial.available()*** para verificar a existência de dados na porta serial, caso haja é utilizado a função ***Serial.parseInt()*** para realizar a leitura destes dados que contém o valor PWM, a partir deste valor novas condições são impostas que seja possível gerar o sinal do sentido de rotação do motor, dado por ***digitalWrite(pino, nível Lógico)***, e o sinal PWM dada por ***analogWrite(pinoPWM, valorPWM)***. A medida que uma condição é satisfeita é recolhido um valor de posição através da função ***Serial.println()*** e após dez milissegundos o ciclo de repetição irá se repetir, devido a função ***delay()***.

3 Código auxiliar - Parte 3

```
33 void loop() {
34     if (Serial.available()>0){
35         int tensao_motor = Serial.parseInt();
36         if(tensao_motor>256){ //para a esquerda
37             digitalWrite(S1,HIGH);
38             digitalWrite(S2,LOW);
39             int valorPWM = round(abs(tensao_motor - 257));
40             analogWrite(pinoPWM, valorPWM);
41             }
42         else if(tensao_motor<256){
43             digitalWrite(S1,LOW);
44             digitalWrite(S2,HIGH);
45             int valorPWM = round(abs(255 - tensao_motor));
46             analogWrite(pinoPWM, valorPWM);
47             }
48         else{
49             digitalWrite(S1,LOW);
50             digitalWrite(S2,LOW);
51             //analogWrite(pinoPWM, valorPWM);
52 
53             }
54             Serial.println(pos_sensor);
55             delay(10);
56           }
57     }
```

3.2 Algoritmo computacional Python

O algorítimo computacional *Python* é a linguagem fonte trabalhada no sistema BCI, uma vez que é nela que está inserido toda a lógica de programação principal, como por exemplo, a definição de funções auxiliares, sinais de entrada, leis de controle, produção de gráficos, dentre outros. O seu código pode ser dividido em 7 partes, onde temos:

- Importação de bibliotecas

Nesta primeira etapa são definidas as bibliotecas que serão utilizadas ao longo do código sendo estas vista em ***Código fonte - Parte 1***.

4 Código fonte - Parte 1

```
1  ####################--- Importação das Bibliotecas ---####################
2  import serial
3  import numpy as np
4  import time as t
5  import matplotlib.pyplot as plt
6  from scipy.signal import square,sawtooth
7  ##########################################################################
```

Onde:

- *serial*: é responsável pela interação entre os dois ambientes computacionais, apesar de ser importada com *serial* ao baixa-la no programa ***Visual Studio Code*** utilize o termo pyserial;
- *numpy*: é responsável pela programação cientifica e numérica;
- *times*: dispõe de funções relacionadas ao tempo;
- *matplotlib.pyplot*: permite a plotagem de gráficos;
- *scipy.signal*: auxilia na criação de sinais;

- Porta de comunicação serial

Importado as biblioteca é necessário definir a porta Arduíno que realizará a comunicação serial permitindo o envio e a coleta de dados, tenha em mente que esta porta varia de acordo com o Arduíno utilizado e é recomendado verifica-la no IDE do Arduíno. A sua representação no código é mostrado em **Código fonte - Parte 2**.

5 Código fonte - Parte 2

```
8  ##############--- Defindo a Porta para a Comunicação Serial ---#############
9  Porta = 'COM8'
10 ##########################################################################
```

- Função para definição do ponto de origem

Para qualquer ensaios realizado o sistema precisa iniciar sempre em um mesmo ponto de partida, dado isto é definida uma função que centraliza o carrinho de impressão antes da execução de um ensaio.

6 Código fonte - Parte 3

```
11 ######################--- Função definindo origem ----####################
12 > def inicializa_carro():...
13
14 inicializa_carro()
15 ##########################################################################
```

- Definindo variáveis.

O ***Código fonte - Parte 4***, nos fornece as variáveis que serão utilizadas para a criação do sinal de entrada e saída.Onde temos:

7 Código fonte - Parte 4

```
16 ########################--- Definindo variáveis ---#######################
17 Ts = 20e-3
18 Amplitude_max = 6
19 fre = 1
20 Amplitude = 5
21 setpoint = 0
22
23 NumAmostras = 300
24 tempo = np.zeros(NumAmostras)
25 y = np.zeros(NumAmostras)
26 x = np.zeros(NumAmostras)
27 ##########################################################################
```

- Ts: o período de amostragem;
- $Amplitude_max$: é o valor de tensão de alimentação do sistema BCI;
- fre: guarda o valor de frequência do sinal de entrada;
- $Amplutde$: guarda o valor de amplitude do sinal de entrada;
- $setpoint$: guarda o valor médio do sinal de entrada;

- *NumAmostras*: guarda o valor total de amostras que serão coletadas em um ensaio.
- *tempo*: guarda o tempo levado para realizar a coleta das amostras;
- y , e x: guardam os dados dos sinais de saída e de entrada do sistema.

- Criação de sinais de entrada.

Em ***Código fonte - Parte 5*** é mostrado como são criados os sinais onda quadrada, dente de serra e o sinal seno que serão utilizados ao longo dos experimentos solicitados.

8 Código fonte - Parte 5

```
28 ###############--- Criação dos Sinais de Entrada ---#########################
29 for n in range(NumAmostras):
30     if(n<30) or (n>NumAmostras-30):
31         x[n]= 0
32     else:
33         x[n] = Amplitude*square(2*np.pi*fre*n*Ts)+setpoint
34         x[n] = Amplitude*sawtooth(2*np.pi*fre*n*Ts)+setpoint
35         x[n] = Amplitude*np.sin(2*np.pi*fre*n*Ts)+setpoint
36 ##########################################################################
```

- Interação entre o Python, Arduíno e o Sistema BCI

É nesta etapa onde ocorre o envio e coleta de dados, para isto utiliza-se a função ***serial.Serial(port= , baudrate= , timeout=*** , mostrada em ***Código fonte - Parte 6***, que é responsável por realizar a conexão entre o programa Python, o Arduíno e o sistema BCI, enquanto a mesma estiver aberta e tiver dados disponíveis para a leitura, que é verificado a partir da função e da condição ***conexao.inWaiting()>0***, estará ocorrendo a coleta do sinal de saída a partir da função ***conexao.readline().decode()*** e o envio do sinal de entrada a partir da função ***conexao.write((str(round(sinal_enviado))).encode())***, quando o laço de repetição se encerrar a conexão serial irá encerrar fazendo com que o ensaio termine.

9 Código fonte - Parte 6

```
37 ##############--- Conexão Python-Arduino- Sistema BCI ---#################
38 conexao = serial.Serial(port=Porta, baudrate=9600, timeout=0.005)
39 t.sleep(1)
40
41 print('Inicializando Leitura')
42
43 for n in range(NumAmostras):
44     if(conexao.inWaiting()>0):
45         y[n] = conexao.readline().decode()
46         y[n] = y[n]/120.2
47
48     t.sleep(Ts)
49     sinal_enviado = 256*(x[n] + Amplitude_max)/Amplitude_max
50
51     conexao.write((str(round(sinal_enviado))).encode())
52     if (n > 0):
53         tempo[n] = tempo[n-1] + Ts
54
55 t.sleep(2*Ts)
56 conexao.write('256'.encode())
57 print('\nFim da coleta.')
58 conexao.close()
59 ##########################################################################
```

- Geração de Gráficos

Terminado a conexão serial e tendo em mãos os dados de entrada e saída pode ser realizado a plotagem gráfica a partir da seguinte estrutura mostrada em ***Código fonte - Parte 7***.

10 Código fonte - Parte 7

```
60 ######################--- Geração de Gráficos ---##########################
61 plt.figure(figsize=(10, 10))
62 plt.plot(tempo, x, '-bo', linewidth=1.2)
63 plt.plot(tempo, y, '-ro', linewidth=1.2)
64 plt.xlabel('Tempo(s)')
65 plt.ylabel('Tensão(V)')
66 plt.grid()
67 plt.show()
```