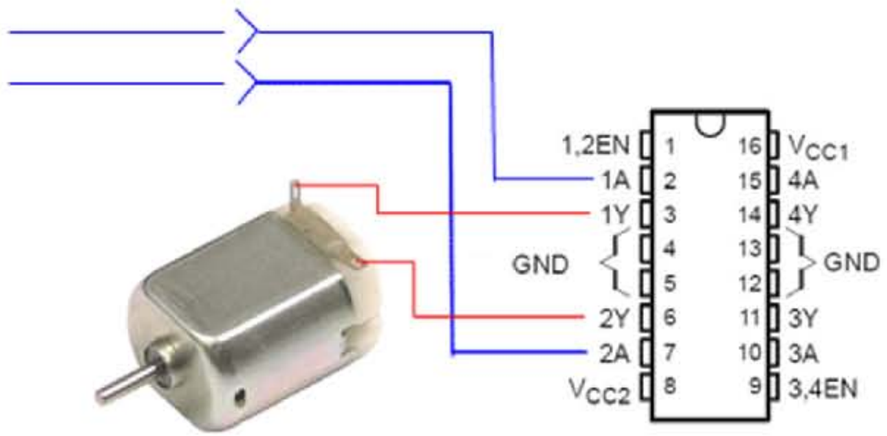
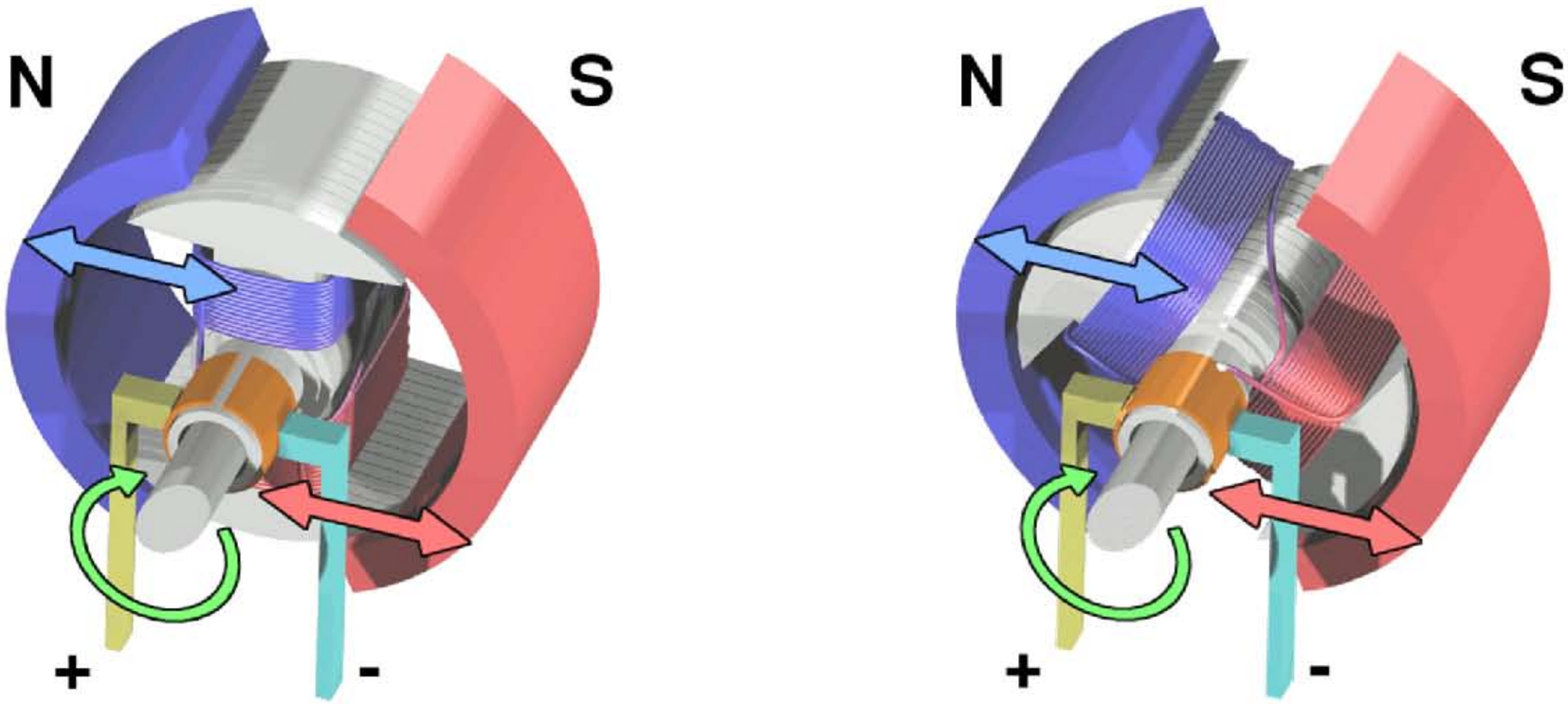
PWM AVR

Pulse Width Modulation

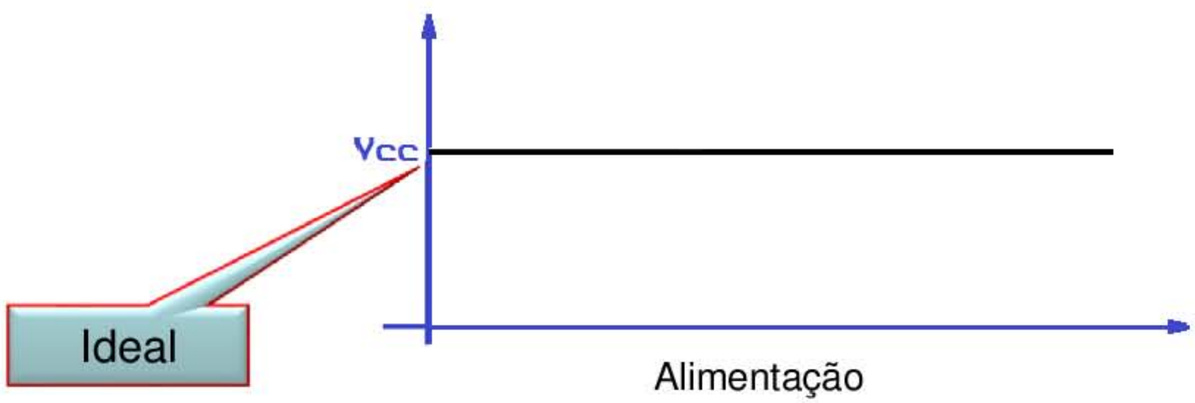


© Prof. Engº esp Luiz Antonio Vargas Pinto
www.vargasp.com
Revisado Novembro 2020

Características

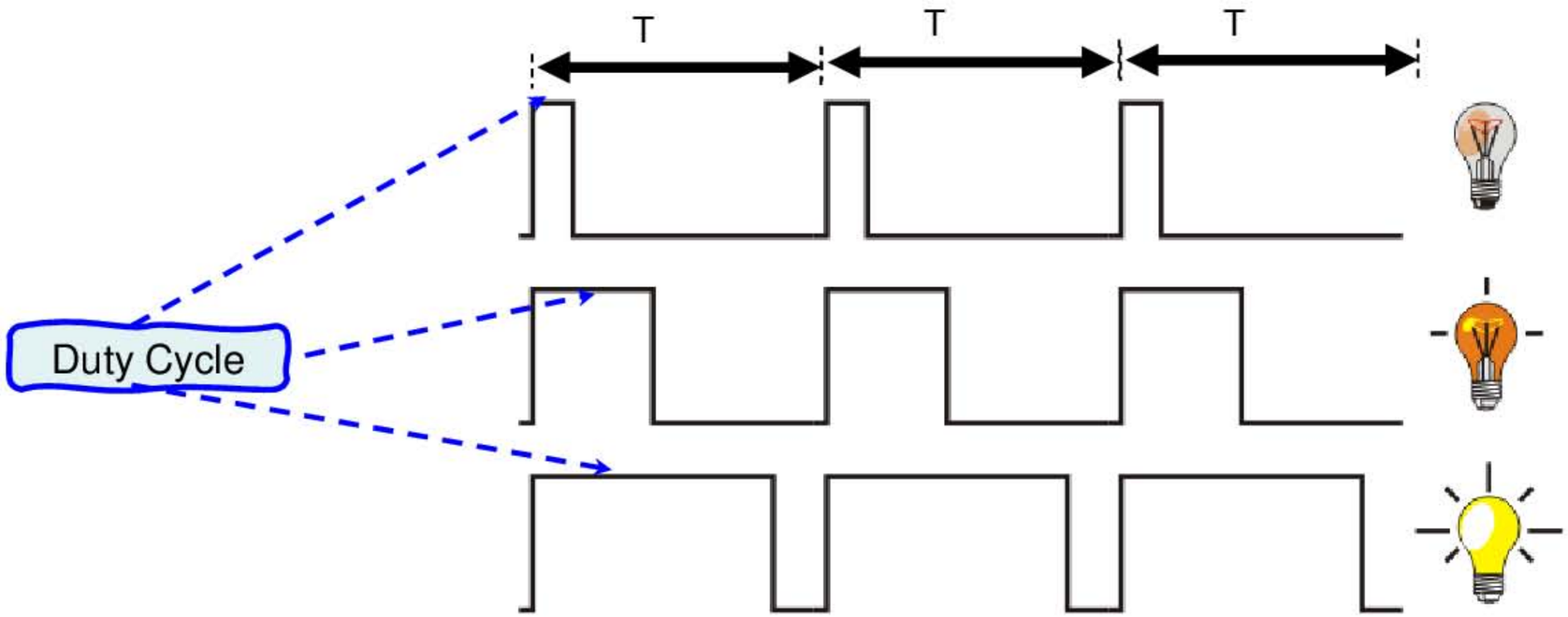


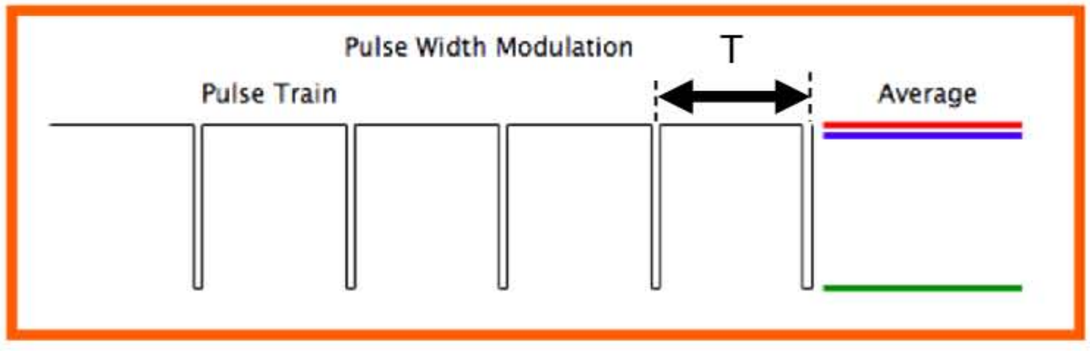
Rotação em RPM



© Prof. Vargas

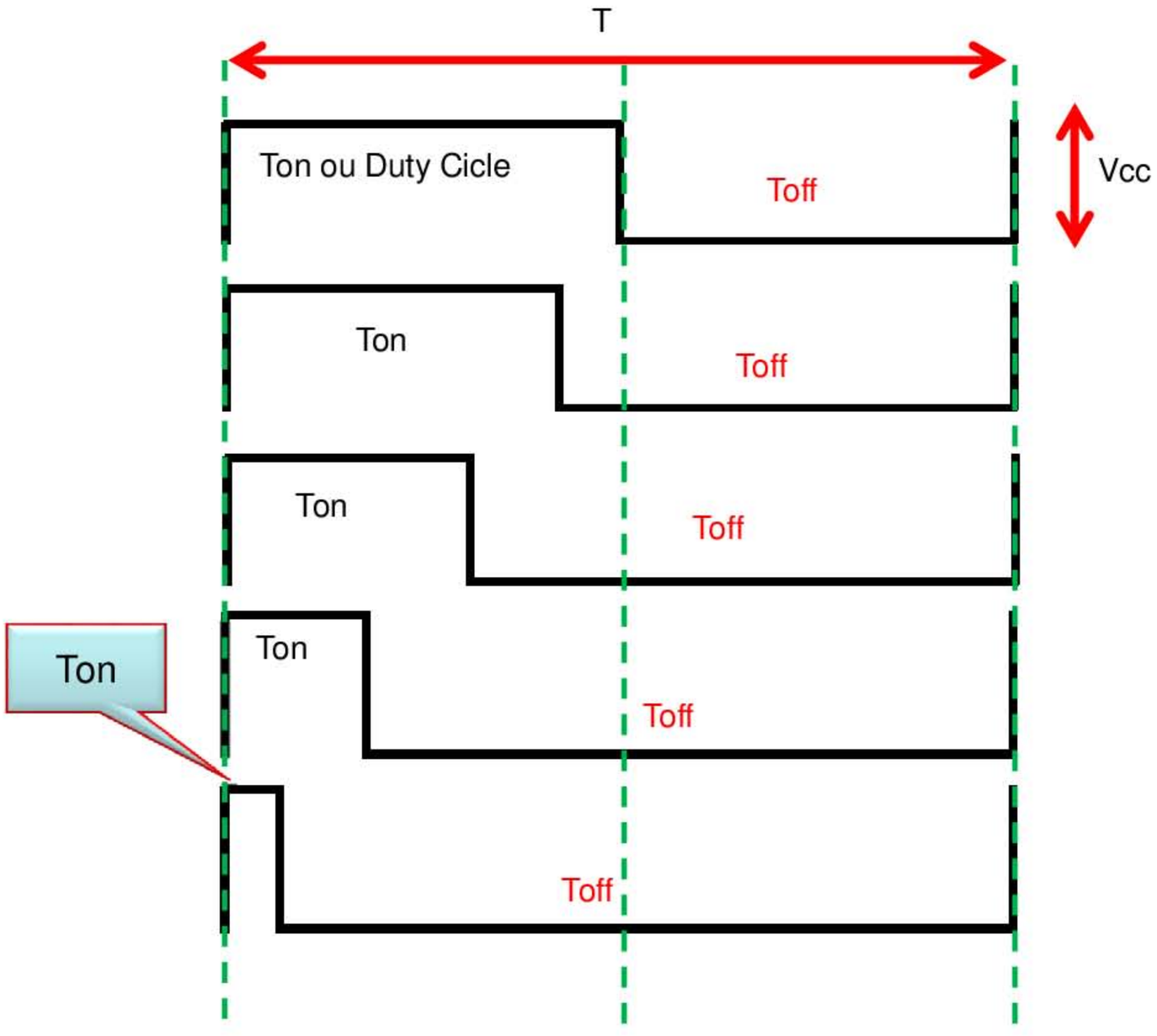
Processo





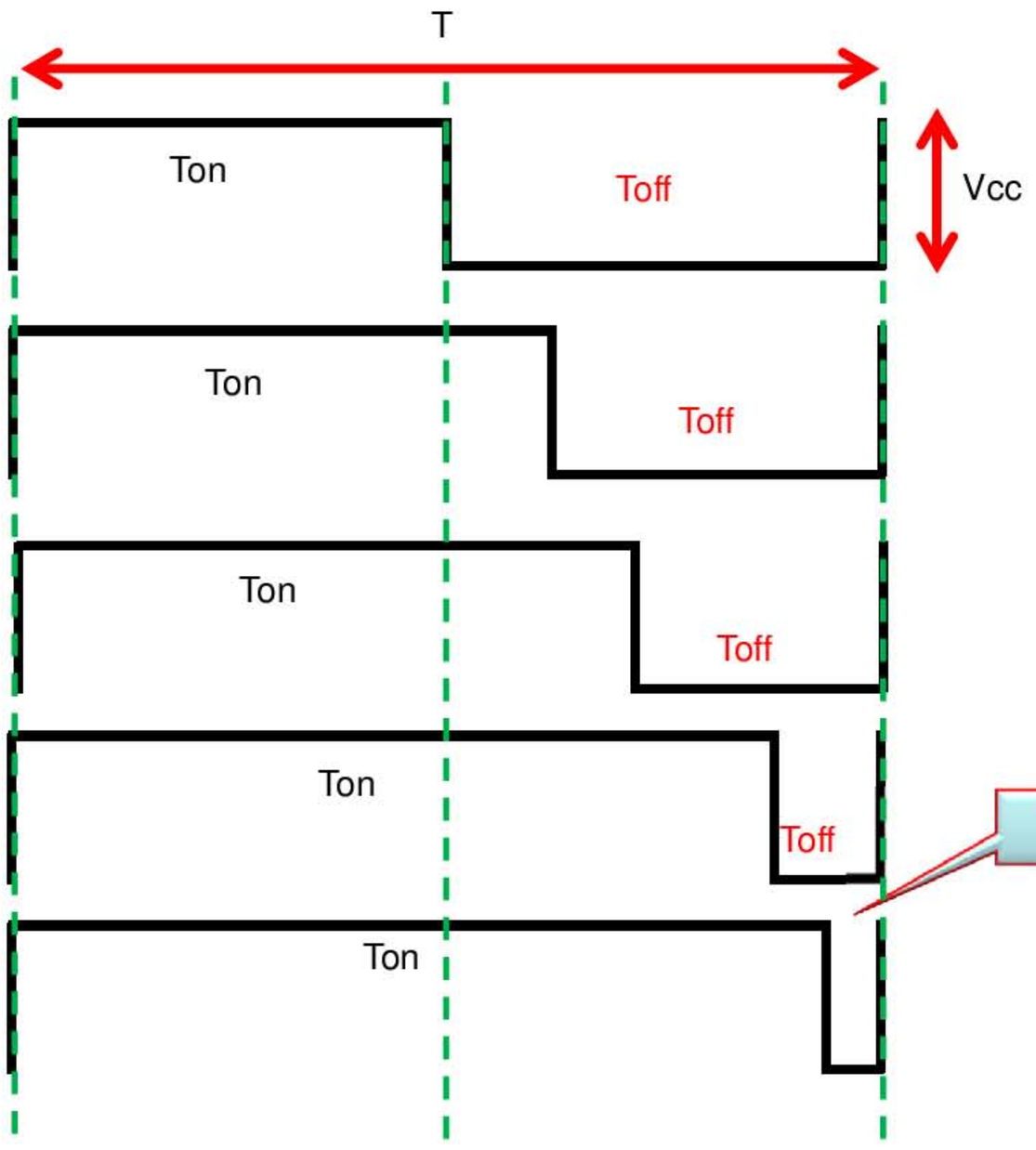
© Prof. Vargas

Intervalando...



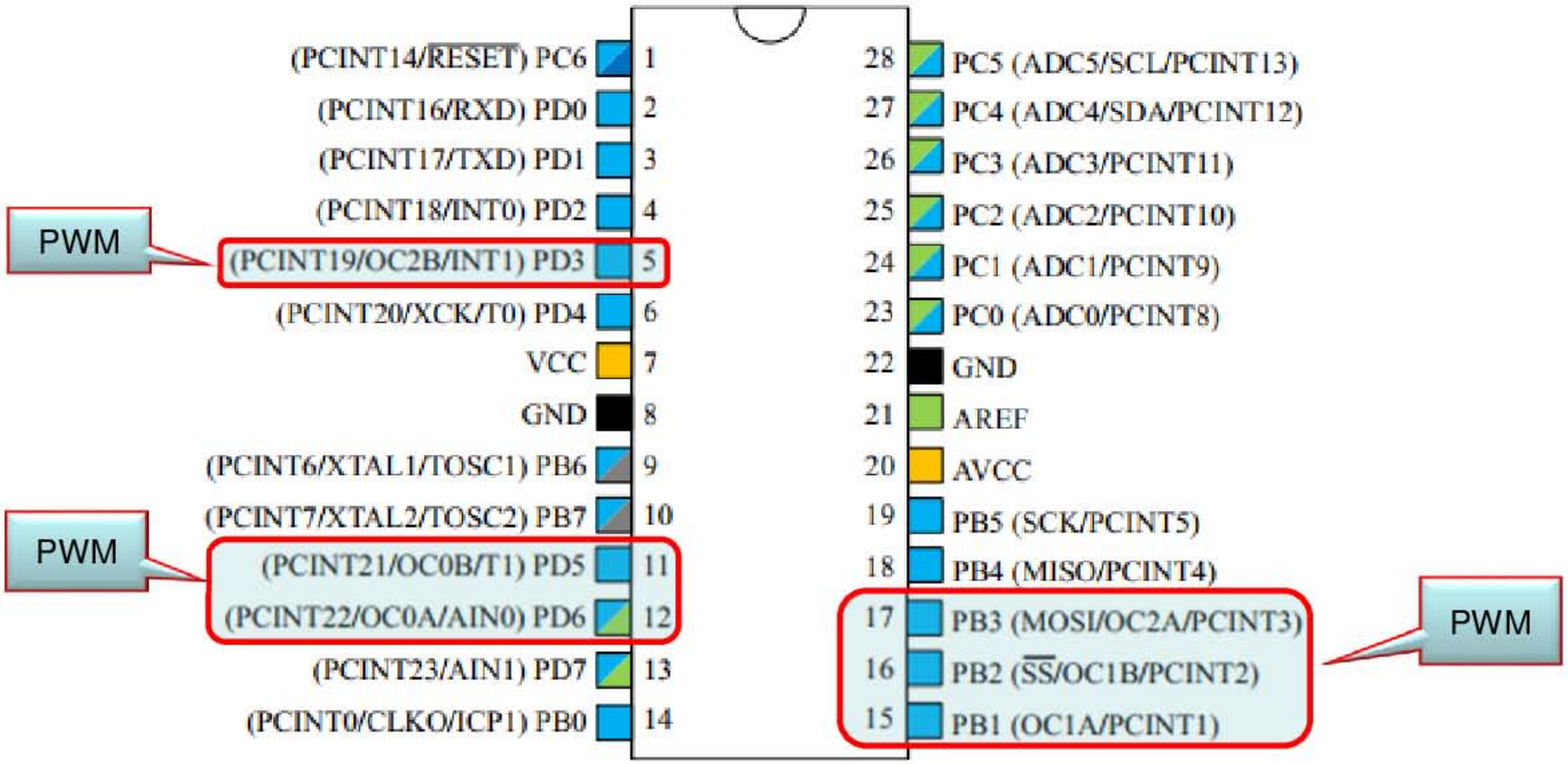
© Prof. Vargas

ou...



© Prof. Vargas

Características



© Prof. Vargas

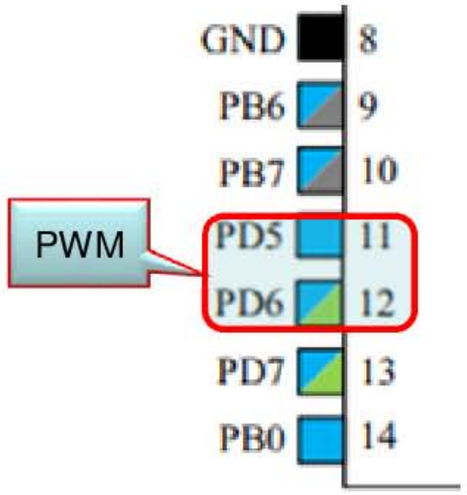
Considerações

- O maior problema não consiste na geração do sinal de PWM mas a f_{PWM} que depende de cada processo.
- Máquinas elétricas DC, por exemplo, funcionam bem com 450 Hz ou 900 Hz

© Prof. Vargas

Usando o ATMEGA

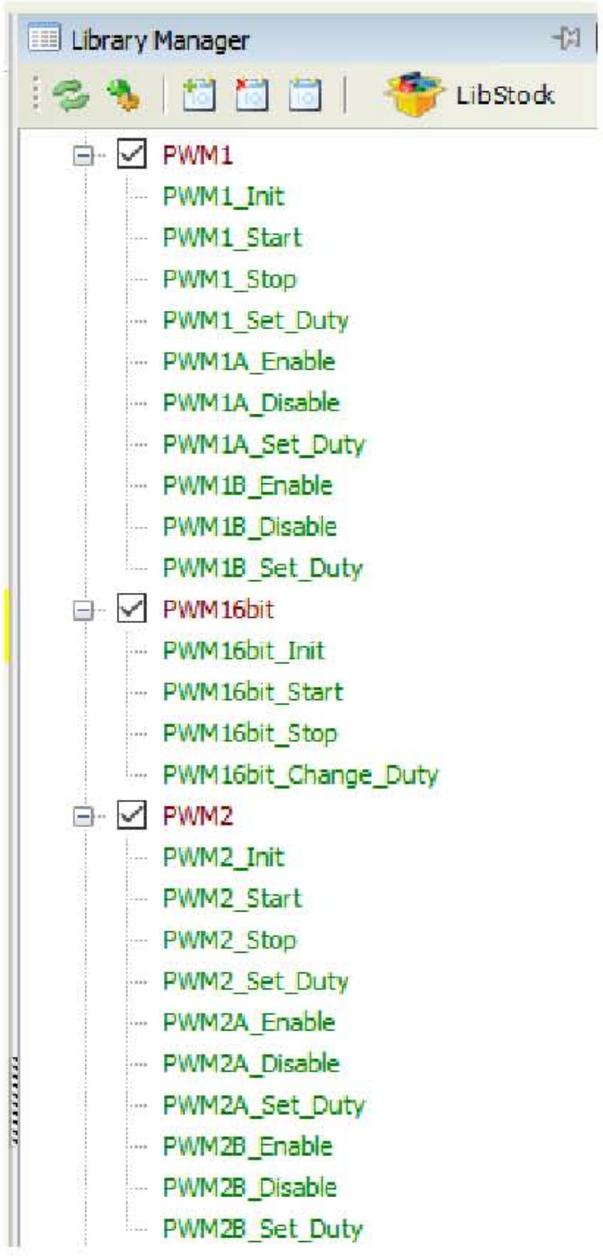
- Como se pode observar, o Atmega usa os pinos 5, 11, 12, 15, 16 e 17 como saída PWM.
- No caso de nossa simulação usaremos os pinos 11 e 12



© Prof. Vargas

Library Manager

- Estas são as bibliotecas para usar os geradores de PWM usando o MikroC for AVR



© Prof. Vargas

PWMx_Init(wave_mode, prescaler, inverted, duty)

wave_mode

Valor	Descrição
_PWM1_FAST_MODE	Fast mode para PWM1
_PWM1_FAST_MODE_OCRA	Fast mode para PWM1
_PWM1_PHASE_CORRECT_MODE	Phase Correct mode para PWM1
_PWM1_PHASE_CORRECT_MODE_OCRA	Phase Correct mode para PWM1
_PWM2_FAST_MODE	Fast mode para PWM2
_PWM2_PHASE_CORRECT_MODE	Phase Correct mode para PWM2

© Prof. Vargas

PWMx_Init(wave_mode, prescaler, inverted, duty)

- prescaler: prescale value N = 1,8,64,256 or 1024

Valor	Descrição
_PWM1_PRESCALER_1	1 = Sem prescaling
_PWM1_PRESCALER_8	8
_PWM1_PRESCALER_32	32 Consultar datasheet.
_PWM1_PRESCALER_64	64
_PWM1_PRESCALER_128	128 Consultar datasheet.
_PWM1_PRESCALER_256	256
_PWM1_PRESCALER_1024	1024
_PWM2_PRESCALER_1	1 = Sem prescaling
_PWM2_PRESCALER_8	8
_PWM2_PRESCALER_32	32 Consultar datasheet.
_PWM2_PRESCALER_64	64
_PWM2_PRESCALER_128	128 Consult datasheet.
_PWM2_PRESCALER_256	256
_PWM2_PRESCALER_1024	1024

2º PWM

© Prof. Vargas

PWMx_Init(wave_mode, prescaler, inverted, duty)

- inverted: inverte ou não o sinal PWM

Valor	Descrição
_PWM1_INVERTED	PWM1 modo invertido
_PWM1_NON_INVERTED	PWM1 modo não invertido
_PWM2_INVERTED	PWM2 modo invertido
_PWM2_NON_INVERTED	PWM2 modo não invertido

2º PWM

© Prof. Vargas

PWMx_Init(wave_mode, prescaler, inverted, duty)

- duty: Ajusta de 0 to 255 o duty cycle
- Exemplo:

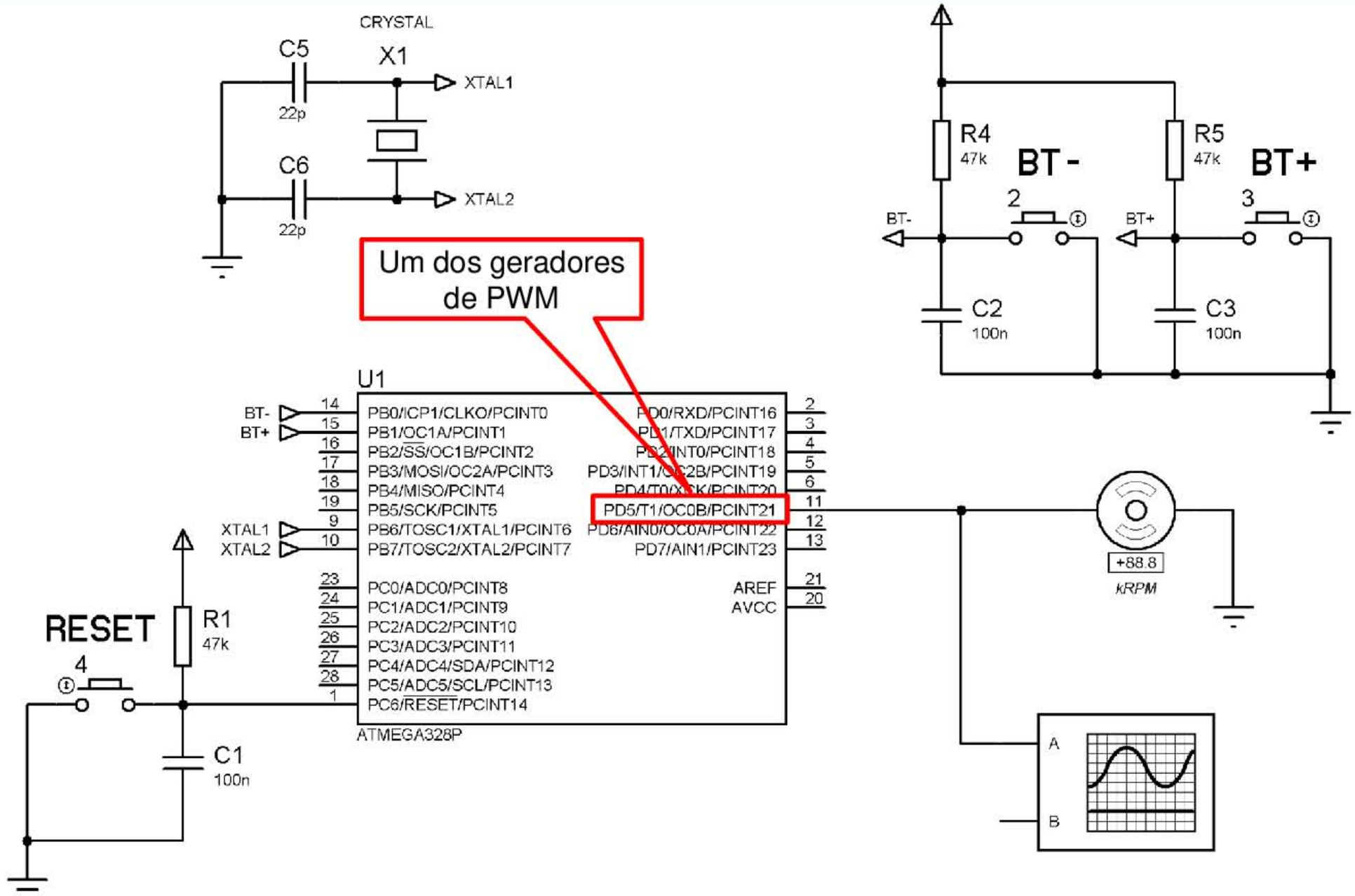
```
current_duty = 85 ;
PWM1_Init(_PWM1_FAST_MODE, _PWM1_PRESCALER_8, _PWM1_NON_INVERTED, current_duty);
```

- PWM1_Set_Duty ( );
- Exemplo:

```
PWM1_Set_Duty(230);     // aproximadamente 90% do máximo
```

© Prof. Vargas

Hardware de teste



© Prof. Vargas

Software de teste

```
/*-----------------------------------------------------------
  Projeto  : AVRPWM1.C
  Data     : 02/11/2019
  Autor    : Prof. Vargas
  Descrição:
   Mostra o funcionamento simples de um sinal PWM sobre um motor
   de cooler de 5Vcc.  O gerador do AVR gera um PWM com Freq_PWM
   = 7142 Hz e permite um controle sobre a velocidade do motor
   usando os Botões B+ (conectado no PORTB.1) e B- (conectado no
   PORTB.0) respectivamente para controlar aumento ou diminuição
   da velocidade de rotação do motor.
   A interface pode permitir uso de outras máquinas controladas
   por PWM, mesmo em AC.
-----------------------------------------------------------*/
```

© Prof. Vargas

Software de teste

```
char duty;

void main()
{
    DDB0_bit = 0;           //BT+
    DDB1_bit = 0;           //BT-
    DDC0_bit = 0;           // PORTC0 será Input
    DDC1_bit = 0;           // PORTC1 será Input
    DDD5_bit = 1;           //PORTD5 será saída PWM
    duty  = 127;            // Prepara duty com 50%
    //Define características do PWM nativo
    PWM1_Init(_PWM1_FAST_MODE, _PWM1_PRESCALER_8, _PWM1_NON_INVERTED, duty);
```

© Prof. Vargas

Software de teste

```
do
{
    if (PINB0_bit)        // Detecta se o Botão + foi apertado
    {
       Delay_ms(40);      // atraso para evitar efeito de boucing
       duty = duty + 10;   // Incremento do duty
       if (duty >= 237)
       {
          duty = 237;
       }
       PWM1_Set_Duty(duty);   //efetiva
    }
    if (PINB1_bit)        // Detecta se o Botão (-) foi apertado
    {
       Delay_ms(40);      // atraso para evitar efeito de boucing
       duty = duty-10;    // Decremento do duty
       if (duty <= 17)
       {
          duty = 17;
       }
       PWM1_Set_Duty(duty);  // efetiva
    }
  } while(1);    // Endless loop
}
```

© Prof. Vargas

Obrigado. Nos vemos na próxima !

© Prof. Vargas