



PADO
Labs



Microcontroladores



Prof.º: Pablo Jean Rozário



pablo.jean@padotec.com.br



/in/pablojeanrozario



<https://github.com/Pablo-Jean>

PWM

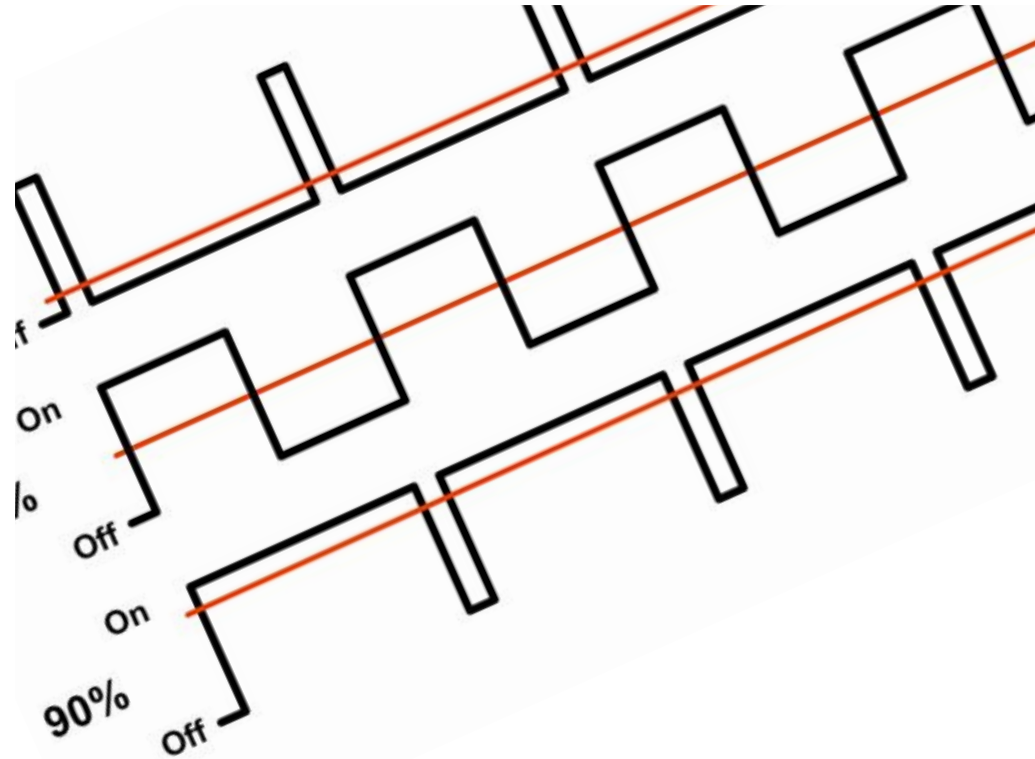
Índice da Aula #9



- Introdução
- Tensão Média
- Potência na Carga e mais
- PWM no STM32G0
- Funções utilizadas
- Lista de Exercícios #9

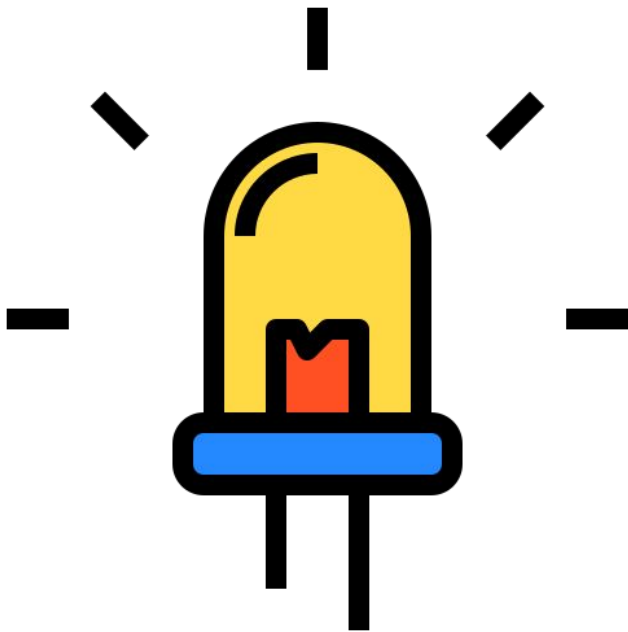
PWM

Power Width Modulation



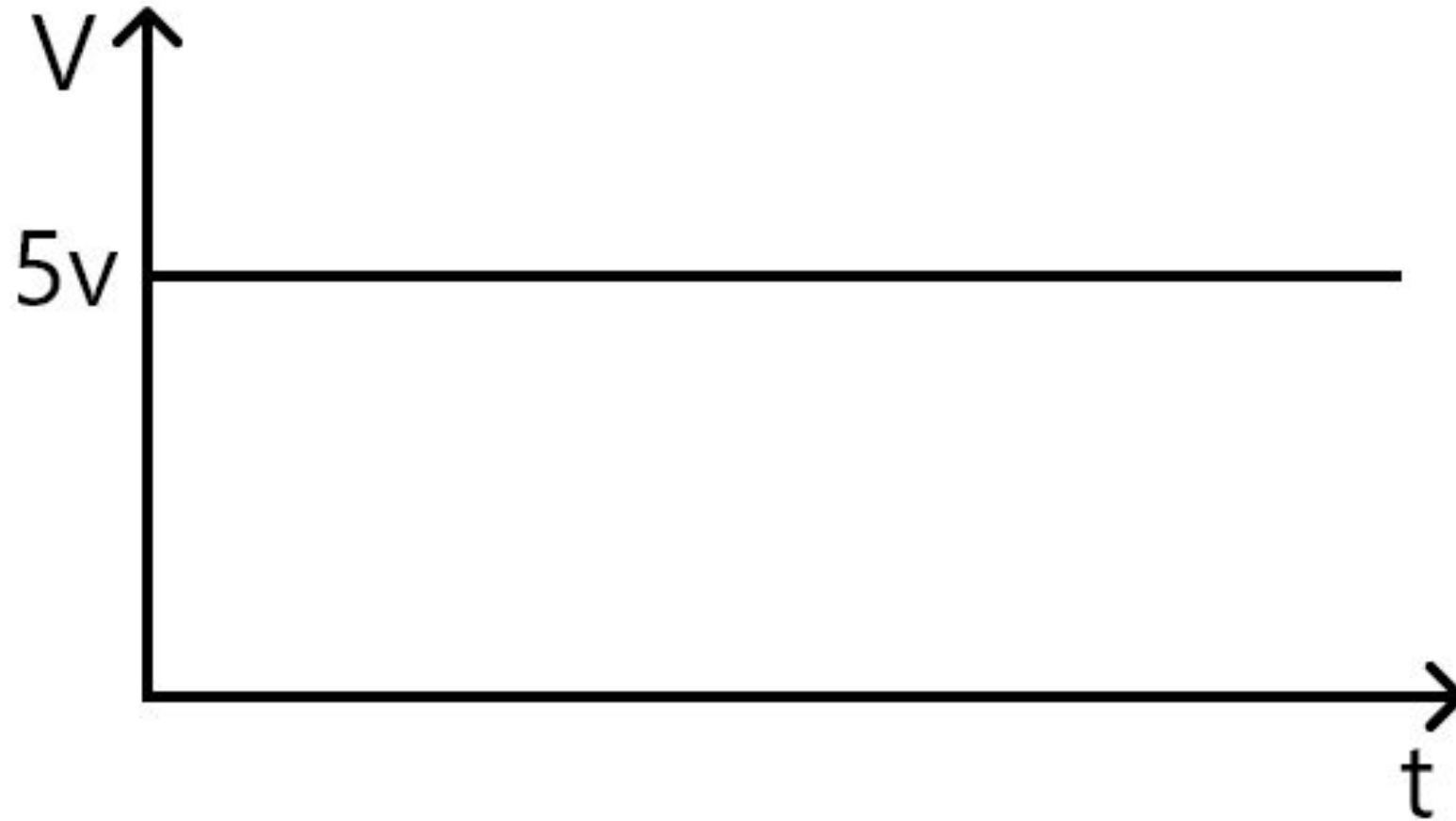
Introdução

PWM ou *Power Width Modulation*, consistem em uma técnica de controlar a potência entregue a uma carga utilizando um sinal digital, eliminando a necessidade de um circuito analógico.

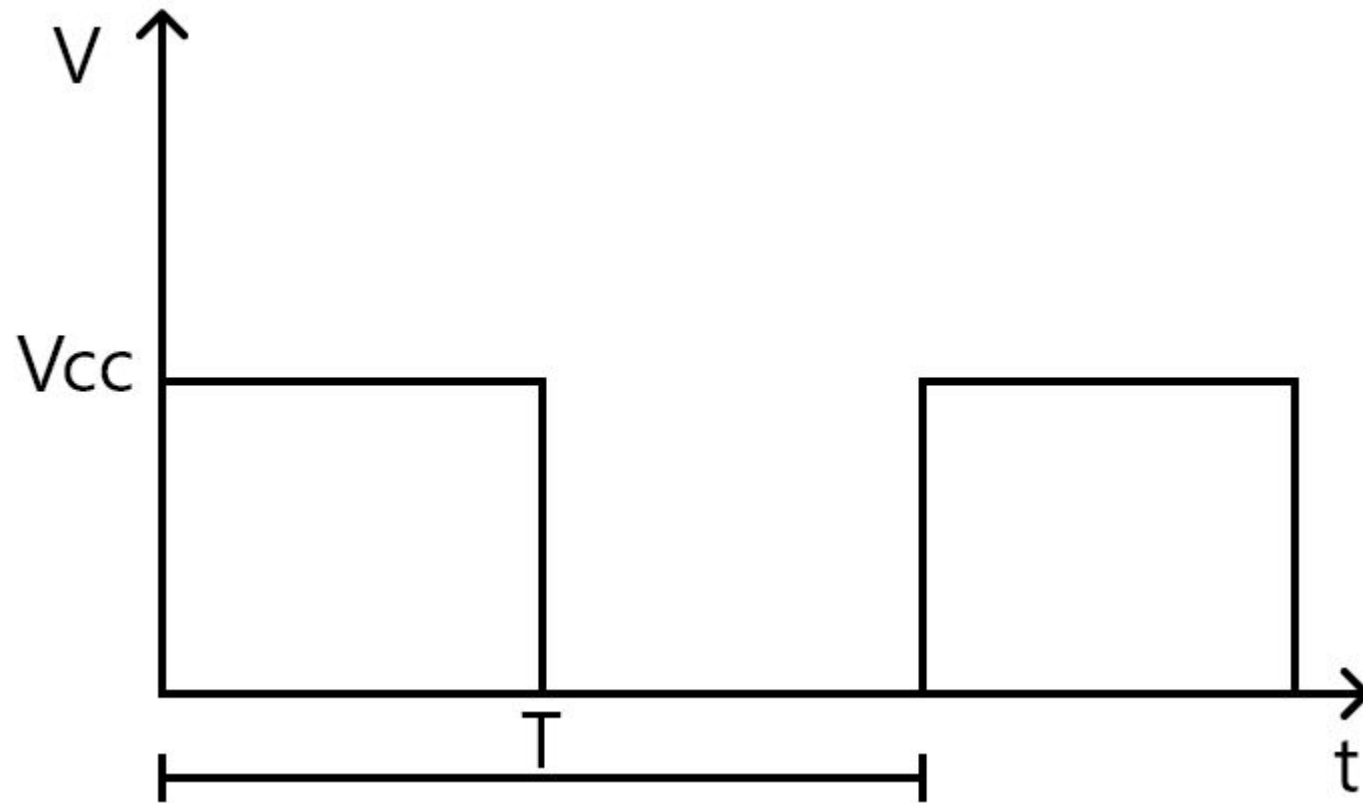


Podemos aplicar a LEDs, motores de corrente contínua, entre outros dispositivos.

Introdução



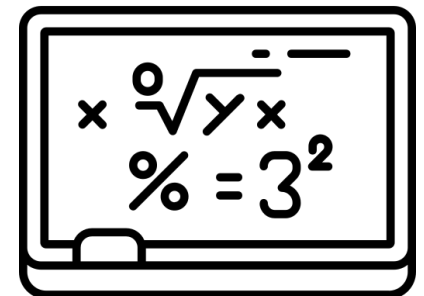
Introdução



Tensão Média

Considerando um tempo **t1** onde a carga está ligada, e **t2** com a carga desligada, podemos calcular a **tensão média** que é entregue a carga.

$$V_{med} = \frac{V_{cc} * t_1 + 0 * t_2}{T}$$



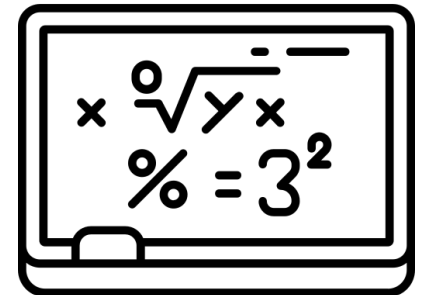
Tensão Média

Podemos simplificar

$$V_{med} = \frac{V_{cc} * t_1}{T}$$

Onde:

- **Vmed** é a tensão média na carga;
- **Vcc** é a tensão de alimentação da carga;
- **t1** é o tempo em que a carga está ligada;
- **T** é o período, sendo **T = t1 + t2**.

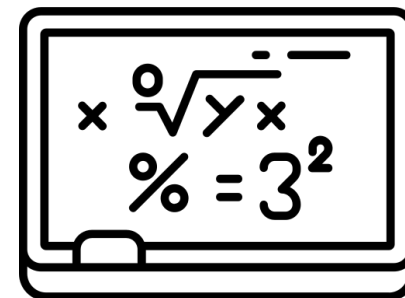


Duty Cycle



Inserindo o conceito de *Duty Cycle*, que nada mais é que a razão entre o tempo ligado com o período total.

$$DC = \frac{t_1}{T}$$



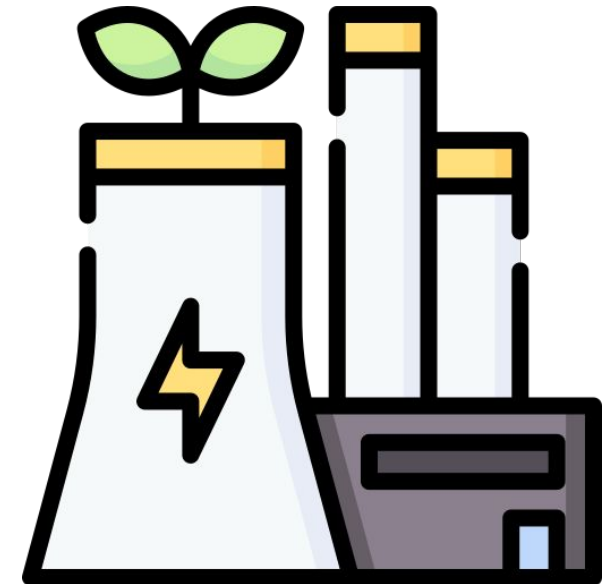
E simplificar a equação de tensão média:

$$V_{med} = V_{cc} * DC$$

Potência na Carga

Com a tensão média aplicada na carga, e conhecemos essa tensão média, podemos ainda calcular a potência que está sendo entregue na carga.

$$P = V_{med} * i_{med} = \frac{(V_{cc} * DC)^2}{R}$$

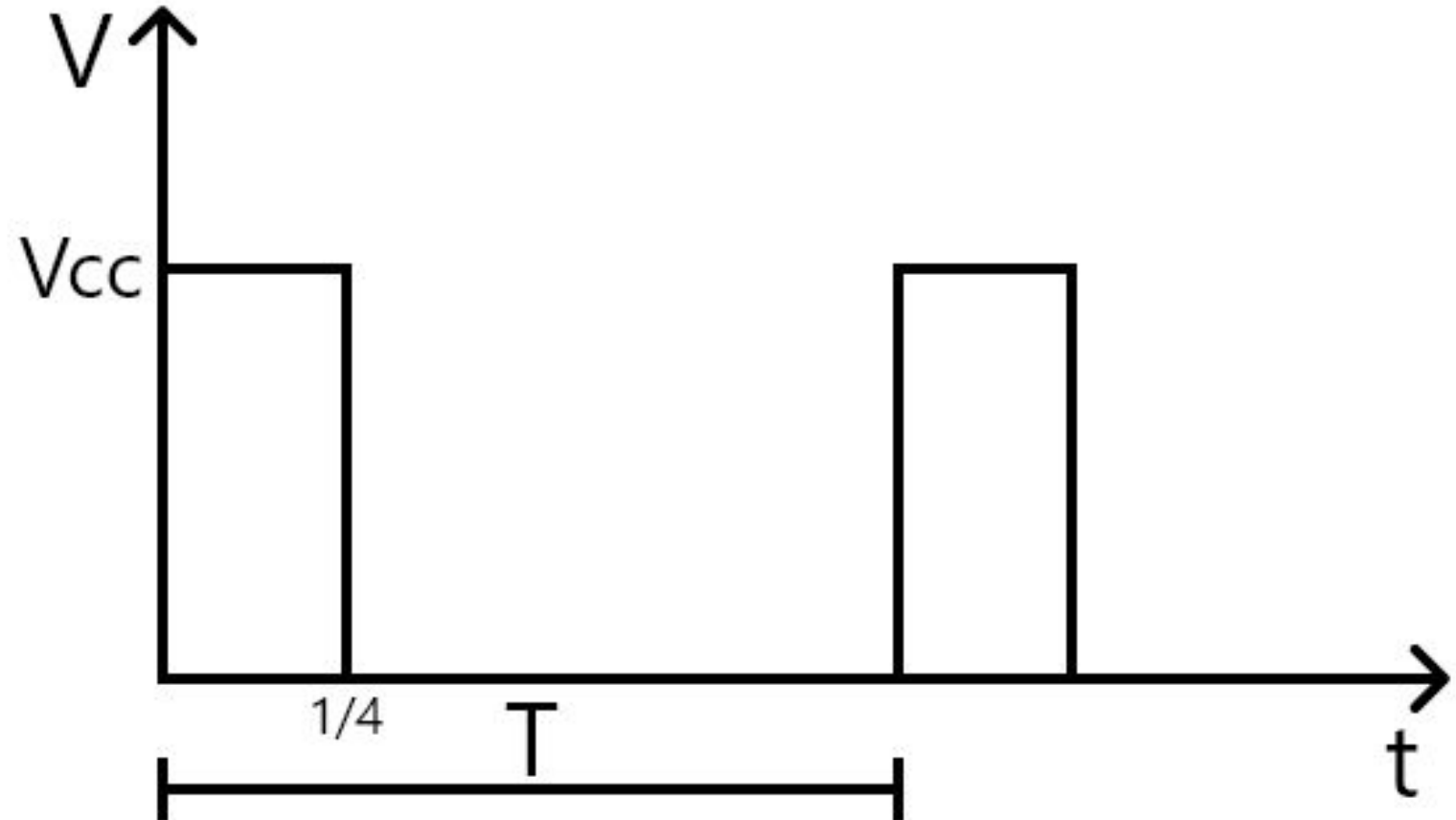


Exemplo Rápido

$V_{cc} = 12V$

DC = 25%

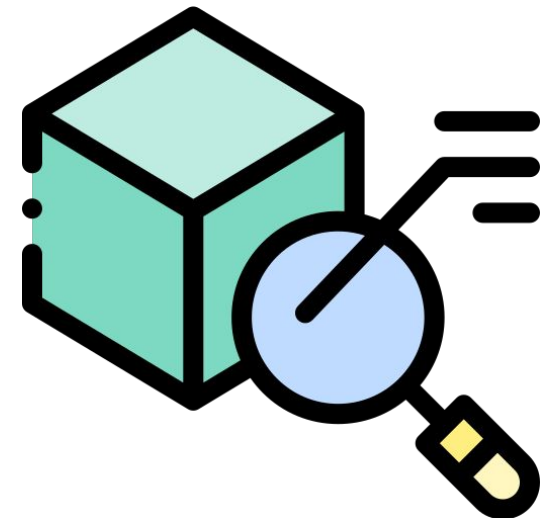
$R = 500 \text{ Ohm}$



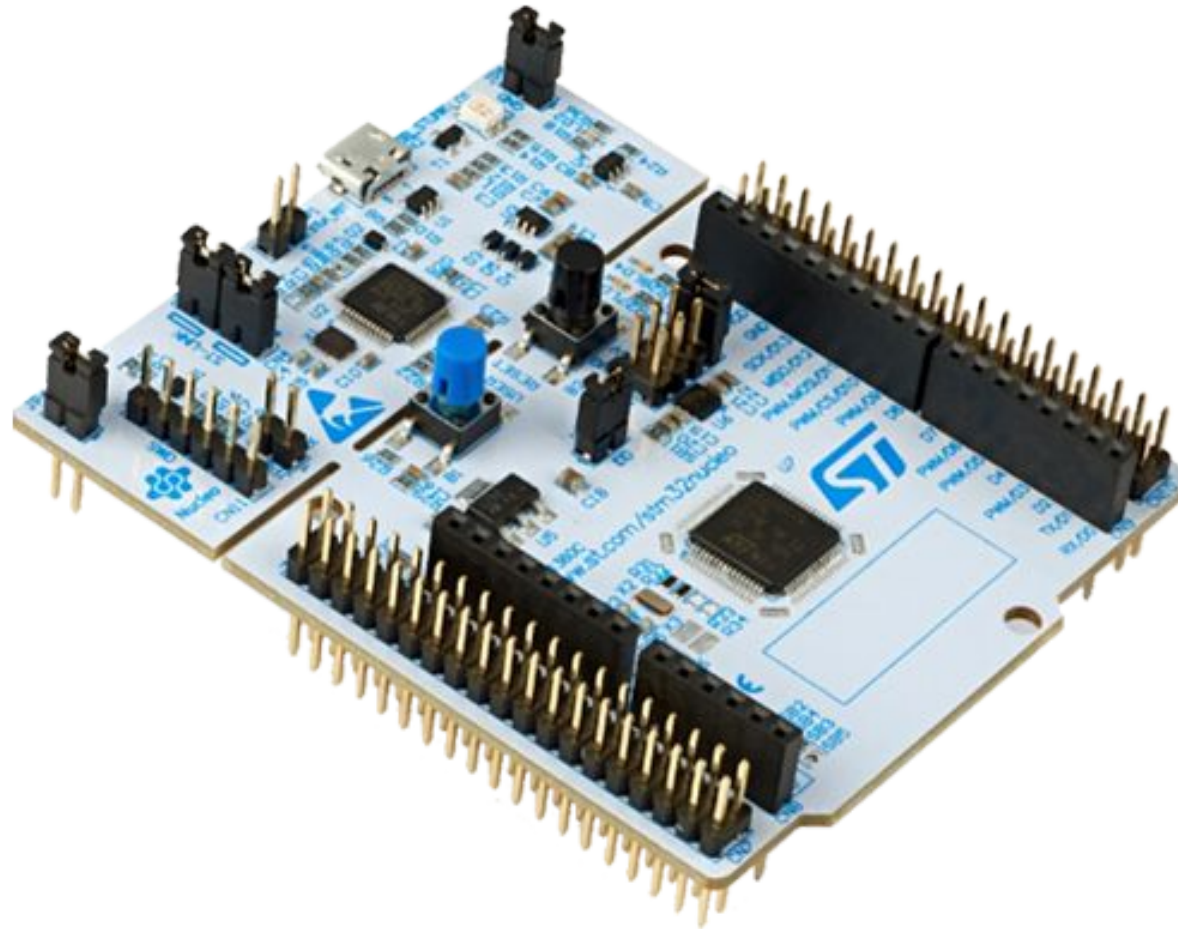
Frequência

Em geral, configuramos o PWM para trabalhar com frequências da ordem de **kHz** (kilo-hertz), principalmente quando queremos *dimmerizar* LEDs, e assim não enxergarmos ele piscando.

Motores de corrente contínua, costumamos utilizar frequência da ordem de **50Hz** à **100Hz**.



PWM no STM32G0



Frequência do Timer


Relembrando a equação para calcular a frequência de interrupção de um *timer*.

$$f_{tim} = \frac{f_{sys}}{(ARR + 1) * (PSC + 1)}$$

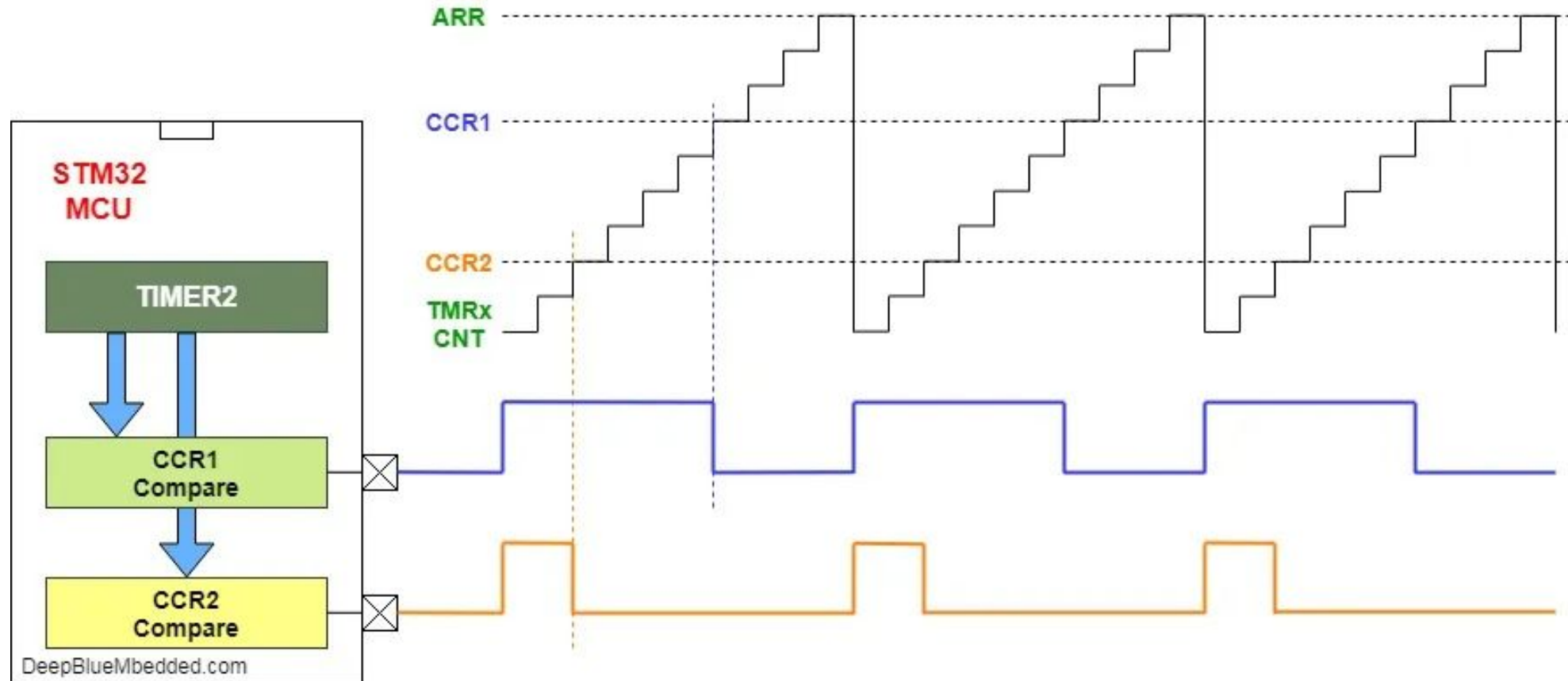


Duty Cycle no Timer

O *Duty Cycle* em um timer configurado como **PWM** depende do valor no registrador do *Capture/Compare* (**CCRx**). Abaixo temos um exemplo com o **CCR1** e o **CCR2** habilitados.



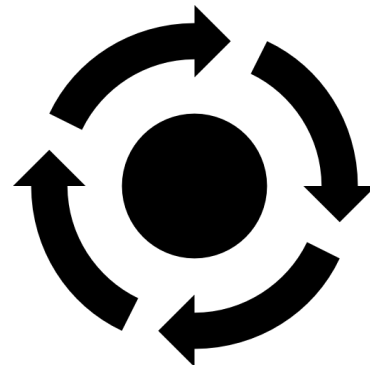
O diagrama ilustra a configuração de um timer STM32 MCU para gerar uma onda de PWM. O timer é configurado com o **CCR1** e o **CCR2** habilitados. O gráfico de onda de PWM mostra a relação entre o valor de comparação no **CCR1** e o **CCR2** e o valor no registrador de *Capture/Compare* (**CCRx**). A onda de PWM é gerada com base nos valores de comparação no **CCR1** e **CCR2**.



Duty Cycle no Timer

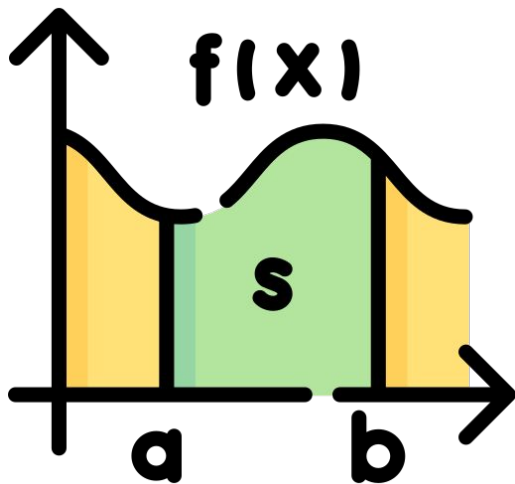
O *Duty Cycle* é independente entre os canais de PWM, **MAS**, a frequência é a mesma, pois é um mesmo contador que gera a frequência.

Por isto, se for necessário uma frequência diferente, devemos utilizar um segundo timer.



Duty Cycle no Timer

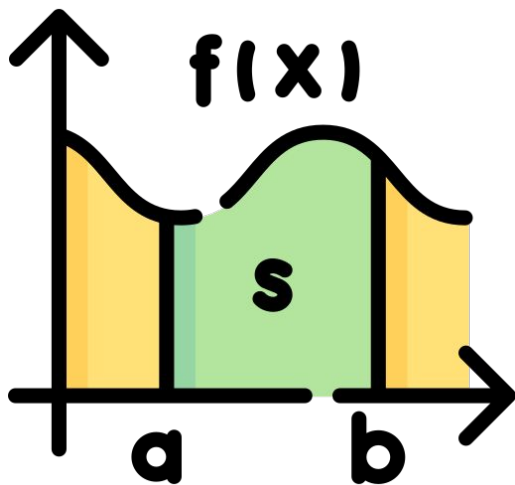
Como vimos, o valor do *Duty Cycle* é uma relação do **CCR_x** com o **ARR**, e disto, temos a equação abaixo para calcular o valor DC.



$$DC_{pwm} = \frac{CCR_x}{ARR}$$

Duty Cycle no Timer

Podemos também, utilizar regra de 3, para encontrar uma equação que nos de o valor do **CCR_x** de acordo com um valor de **DC** (de 0 à 100)



$$CCR_x = \frac{DC}{100} * ARR$$

Observação

Os *basic timers* NÃO possuem a funcionalidade de PWM (muito menos o RTC).



Localização dos Terminais



Acesse e **STM32G0B1xB/xC/xE**, e consulte a **tabela 13 à 20**.

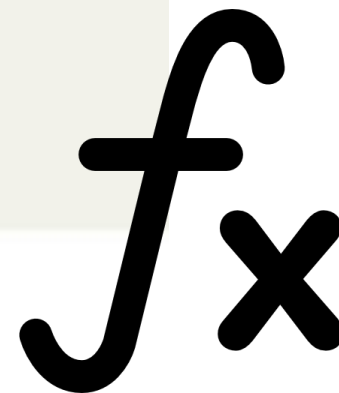
Port	AF0	AF1	AF2
PB0	SPI1_NSS/ I2S1_WS	TIM3_CH3	TIM1_CH2N
PB1	TIM14_CH1	TIM3_CH4	TIM1_CH3N
PB2	-	SPI2_MISO/ I2S2_MCK	-
PB3	SPI1_SCK/ I2S1_CK	TIM1_CH2	TIM2_CH2
PB4	SPI1_MISO/ I2S1_MCK	TIM3_CH1	-
PB5	SPI1_MOSI/ I2S1_SD	TIM3_CH2	TIM16_BKIN
PB6	USART1_TX	TIM1_CH3	TIM16_CH1N

STM32G0 - Funções Utilizadas



Para iniciar o PWM, temos que chamar a função descrita abaixo

```
1  /* Inicia o modo de PWM do Timer, onde htim refere-se ao handle do
   timer ,
2  * e o parametro Channel e o canal que desejamos ligar , podendo ser
3  * TIM_CHANNEL_1: TIM Channel 1
4  * TIM_CHANNEL_2: TIM Channel 2
5  * TIM_CHANNEL_3: TIM Channel 3
6  * TIM_CHANNEL_4: TIM Channel 4
7  */
8  HAL_TIM_PWM_Start(TIM_HandleTypeDef *htim, uint32_t Channel);
9
```



STM32G0 - Funções Utilizadas



E para parar o PWM utilizamos:

```
10  /* Parao timer em modo PWM, onde htim refere-se ao handle do timer ,
11     * e o parametro Channel e o canal que desejamos suspender , podendo ser
12     * TIM_CHANNEL_1: TIM Channel 1
13     * TIM_CHANNEL_2: TIM Channel 2
14     * TIM_CHANNEL_3: TIM Channel 3
15     * TIM_CHANNEL_4: TIM Channel 4
16     */
17  HAL_TIM_PWM_Stop(TIM_HandleTypeDef *htim , uint32_t Channel)
```

f_x

STM32G0 - Funções Utilizadas



Para alterar o valor do **CCR_x**, utilizamos a função abaixo:

```
__HANDLE__
12  * refere-se ao handle do timer, e o parametro __CHANNEL__ e o canal
    que
13  * que desejamos escrever o valor:
14  * TIM_CHANNEL_1: TIM Channel 1
15  * TIM_CHANNEL_2: TIM Channel 2
16  * TIM_CHANNEL_3: TIM Channel 3
17  * TIM_CHANNEL_4: TIM Channel 4
18  * __COMPARE__ e o valor que desejamos definir
19  */
20  __HAL_TIM_SET_COMPARE(__HANDLE__, __CHANNEL__, __COMPARE__);
```

f_x

STM32G0 - Funções Utilizadas



Podemos ainda ler o registrador **CCR_x** com a função descrita abaixo, retornando o valor do registrador.

```
1  /* Le o valor do registrador do Capture/Compare do timer , onde
   __HANDLE__
2  * refere-se ao handle do timer , e o parametro __CHANNEL__ e o canal
   que
3  * que desejamos ler o valor
4  * TIM_CHANNEL_1: TIM Channel 1
5  * TIM_CHANNEL_2: TIM Channel 2
6  * TIM_CHANNEL_3: TIM Channel 3
7  * TIM_CHANNEL_4: TIM Channel 4
8  */
9  __HAL_TIM_GET_COMPARE(__HANDLE__, __CHANNEL__);
```

f_x

STM32G0 - Mais Funções

Para consultar mais funções do módulo de PWM, utilize a documentação **UM2319:Description of STM32G0 HAL and low-layer drivers**, nos capítulos:

- **48 HAL TIM Generic Driver**
- **49 HAL TIM Extension Driver**

Sim, é os mesmos capítulos do *timer*.



Dúvidas ??



Referências

ADAFRUIT. **Choosing Decay Mode and PWM Frequency**. 2021.

<https://learn.adafruit.com/improve-brushed-dc-motor-performance/choosing-decay-mode-and-pwm-frequency> . Acesso em 16 de Março de 2022.

GUIMARÃES, Fábio. **PWM (Pulse Width Modulation)**. 2017.

<https://mundoprojetado.com.br/pwm-pulse-width-modulation/> . Acesso em 16 de Março de 2022.

MAGDY, Khaled. **STM32 PWM Example – Timer PWM Mode Tutorial**. 2021.

<https://deepbluembedded.com/stm32-pwm-example-timer-pwm-mode-tutorial/> . Acesso em 16 de Março de 2022.

ST MICROELECTRONICS. **RM0444 - Reference Manual**. 5. ed. [S.l.], 2020.

STM32G0x1 advanced Arm ® -based 32-bit MCUs.

__. **STM32G0B1xB/xC/xE**. 2. ed. [S.l.], 2021. Arm ® Cortex ® -M0+ 32-bit MCU, up to 512KB Flash, 144KB RAM, 6x USART, timers, ADC, DAC, comm. I/Fs, 1.7-3.6V.

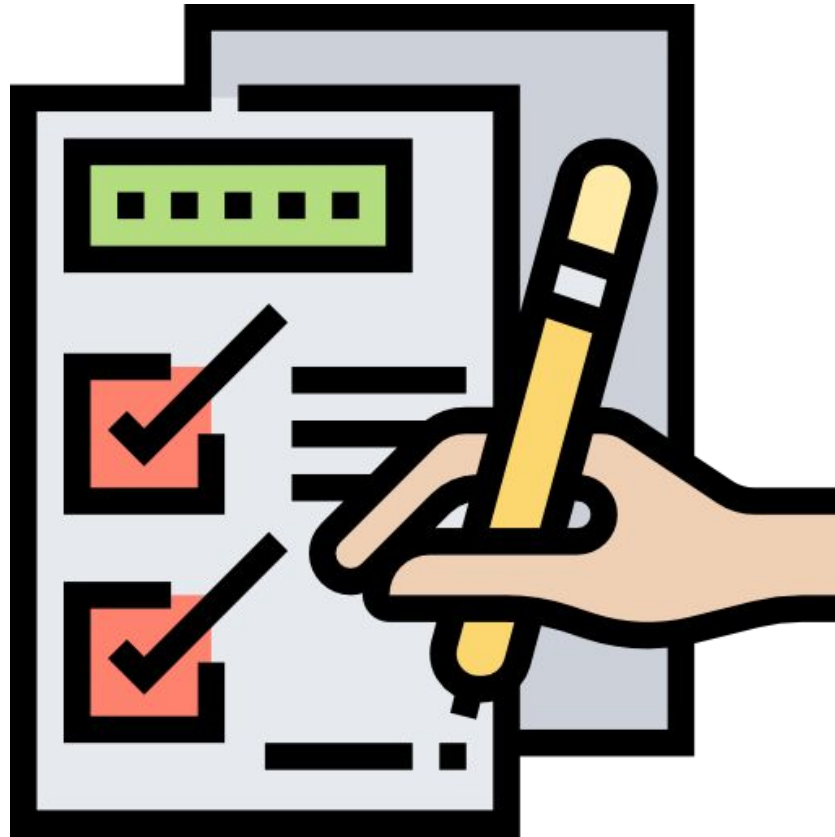
__. **UM2319: Description of STM32G0 HAL and low-layer drivers**. 2. ed. [S.l.], 2020.



Mão na Massa



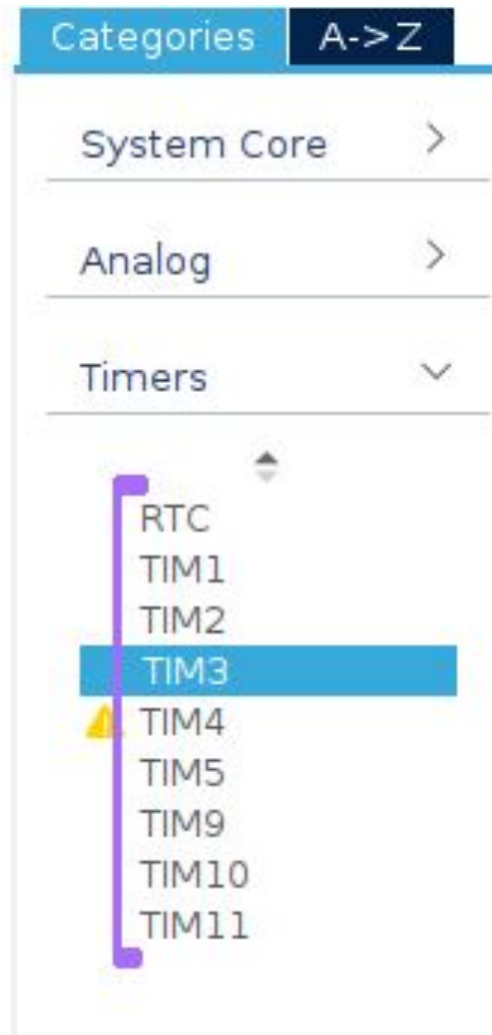
Lista de Exercícios #9



PWM



PWM - Configuração Inicial



Selecione o timer e/ou os timers desejados, que suportam o PWM

PWM - Configuração Inicial

TIM3 Mode and Configuration

Mode

Slave Mode

Trigger Source

☒ Internal Clock

Channel1

Channel2

Channel3

Channel4

Combined

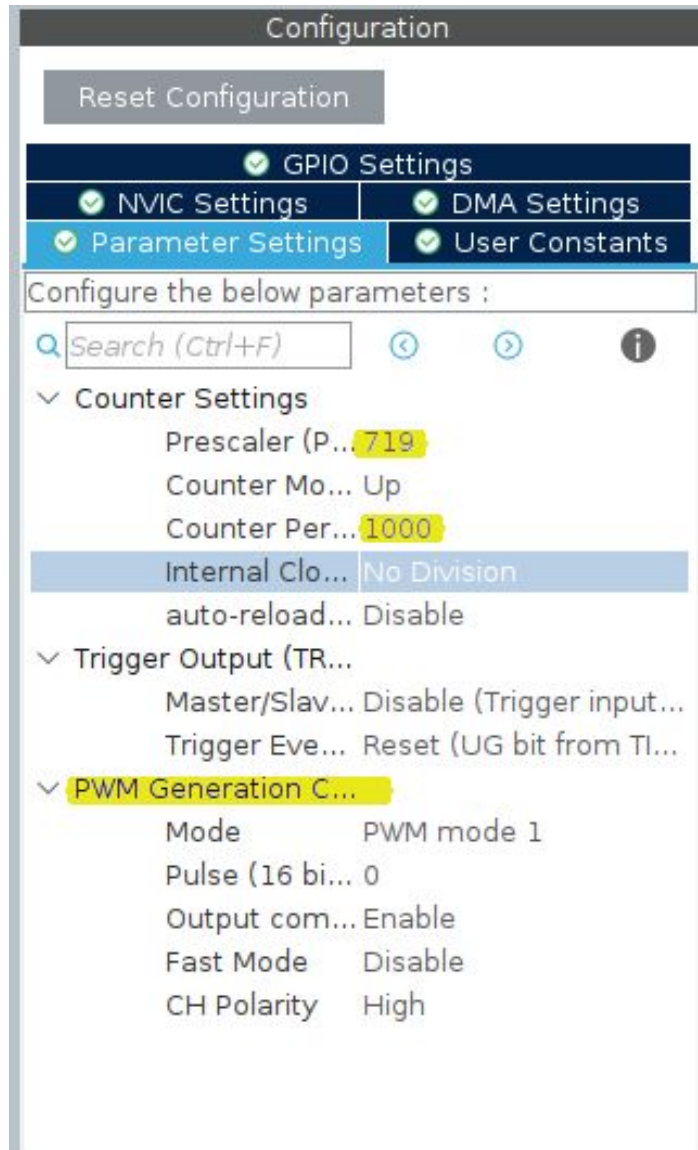
☐ XOR and

☐ One Pulse

Selecione o **Internal Clock** para habilitar o *timer*.

Selecione os canais que serão necessários e selecione a opção no *dropdown* **PWM Generation CHx**.

Timer - Configuração Inicial



Em geral, alteramos a apenas o **PSC** e o **ARR**.

Em casos muito específicos as opções do PWM são alteradas.



PADO
Labs