Como alterar a frequência do PWM da placa Arduino UNO.

Por Eng. Roberto Bairros dos Santos

www.bairrospd.com

Data: 07/07/2017

Sumário

Prefácio.

Este tutorial mostra como alterar a frequência de operação da função analogWrite() da placa Arduino UNO!

Existem duas formas:

- Mantendo a configuração original dos temporizadores e só alterando os bits de ajuste da frequência.
- Alterando a configuração original dos temporizadores e os bits de frequência.

Você deve estar ciente que ao mudar a configuração dos temporizadores algumas funções de tempo do Arduino podem não funcionar corretamente!

Para você entender melhor este tutorial é recomendável ver os tutoriais no site www.bairrospd.com que explicam como funciona o temporizador do microcontrolador ATMega e como medir as frequências dos temporizadores da placa Arduino UNO na sua configuração padrão.

O circuito e o programa para testar as alterações.

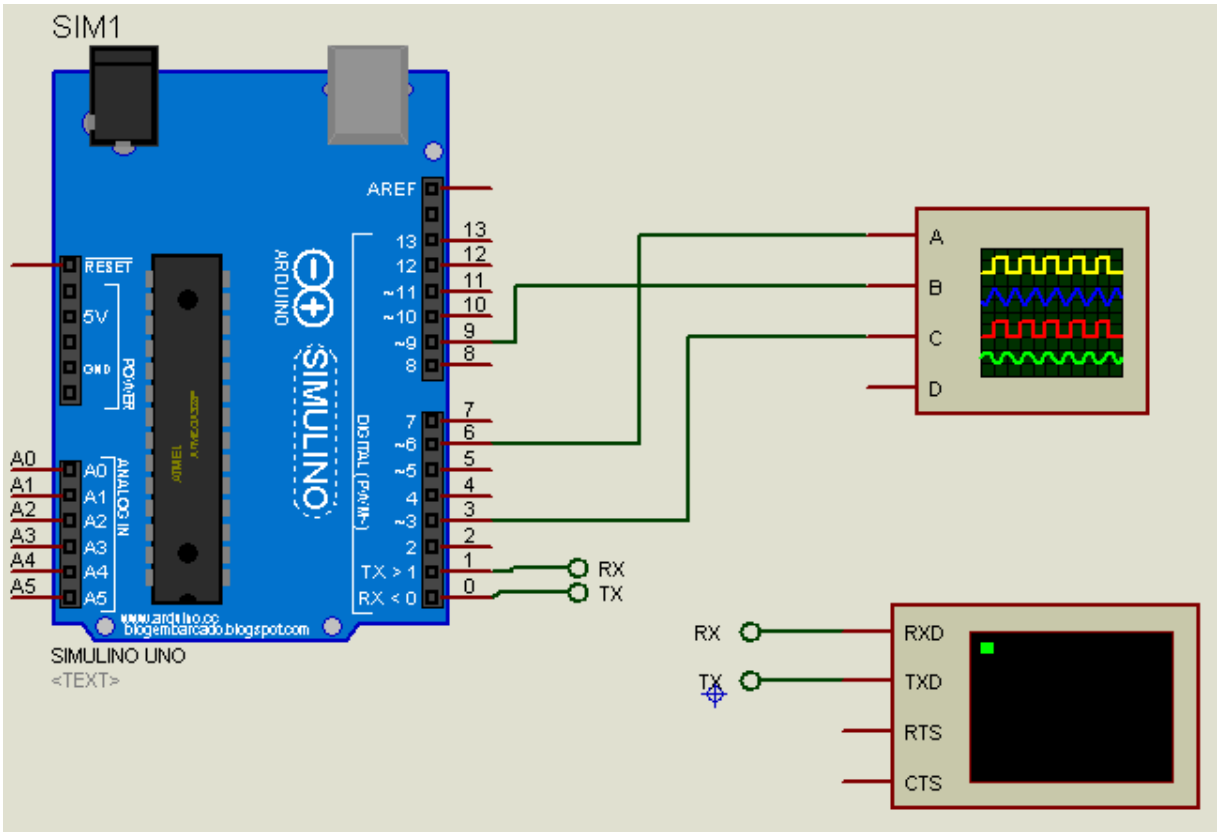
O diagrama abaixo mostra como medir a frequência dos temporizadores.

O osciloscópio foi ligado para mostrar uma saída de cada temporizador.

Neste trabalho foi usado o programa ISIS do PROTEUS com o simulador de Arduino SIMULINO.

Para ver o resultado no terminal da serial foi usado o terminal do Proteus.

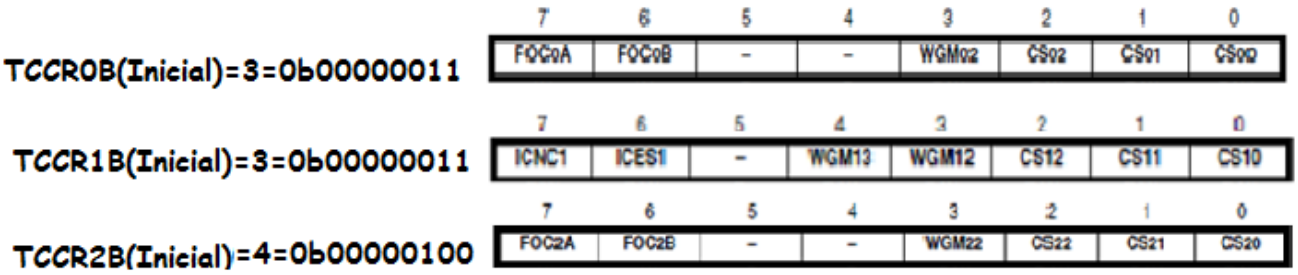
O osciloscópio está ligado nos pinos 6 (TIMER0) cor amarelo, 9 (TIMER1) cor azul, 3(TIMER2), cor vermelho!



Como alterar a frequência alterando somente os bits de configuração da frequência.

Este é o método mais simples, você só irá mudar os bits CS0x, CS1x e CS2x de qualquer um dos temporizadores TCCRnB!

O valor padrão para cada um dos temporizadores da placa Arduino UNO é mostrado abaixo.



A tabela abaixo mostra um resumo com todas as possibilidades de alterações das frequências dos temporizadores usados para gerar o PWM na placa Arduino UNO.

Esta tabela será útil para alterar a frequência, a linha em amarelo mostra o valor inicial!

TCCR0B=3=0b00000011

TIMER 0 Fast PWM

CS	N	fOCnxPWM	t(ms)
1	1	62500	0,016
2	8	7812,5	0,128
3	64	976,5625	1,024
4	256	244,140625	4,096
5	1024	61,03515625	16,384

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

Pinos 05 e 06

TCCR1B(Inicial)=3 =0b00000011

TIMER 1 PWM, phase correct

CS	N	fOCnxPCPWM	t(ms)
1	1	31372,55	0,031875
2	8	3921,57	0,255
3	64	490,20	2,040
4	256	122,55	8,160
5	1024	30,64	32,640

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

Pinos 09 e 10

TCCR2B (Inicial) =4=0b00000100

TIMER 2 PWM, phase correct

CS	N	fOCnxPWM	t(ms)
1	1	31372,54902	0,031875
2	8	3921,568627	0,255
3	32	980,3921569	1,02
4	64	490,1960784	2,04
5	128	245,0980392	4,08
6	256	122,5490196	8,16
7	1024	30,6372549	32,64

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

Pinos 11 e 03

Note que o temporizador 2 é aquele que possui mais opções de frequência, os outros temporizadores trabalham com frequências mais altas.

Existem várias maneiras de escrever em linguagem “C” as alterações dos bits “CSn” do registrador TSSRnB.

Uma forma bem simples é mostrada abaixo onde foi usada a técnica de criar uma máscara com a função “AND” e depois combinar com a função “OR” para chegar ao valor final .

Na máscara com a função “AND” onde os bits do operador iguais a um são aqueles que não são alterados e os bits iguais zero são zerados.

Na operação final com a função “OR” ocorre o contrário, os bits do operador igual a zero não são alterados, somente os bits iguais a um são alterados!

Para esta alteração você pode escrever o número no formato binário ou decimal.

O exemplo abaixo mostra como alterar o temporizador 2 para a frequência de 3921 Hz.

TCCR2B (Inicial) =4=0b00000100

TCCR2B(3921Hz)=2=0b00000010

TIMER 2 PWM, phase correct

CS	N	$f_{OCnxPWM}$	t(ms)
1	1	31372,54902	0,031875
2	8	3921,568627	0,255
3	32	980,3921569	1,02
4	64	490,1960784	2,04
5	128	245,0980392	4,08
6	256	122,5490196	8,16
7	1024	30,6372549	32,64

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

Pinos 11 e 03

Veja como escrever a alteração usando número inteiro.

```
//ajuste do temporizador 2 da saida 11 e 3 para 3kHz
TCCR2B=(TCCR2B & 0b11111000) | 2;
```

0b00000101 TCCR2B inicial 490 Hz
0b11111000 AND

(TCCR2B & 0b11111000) ➪ 0b00000000
0b00000010 OR

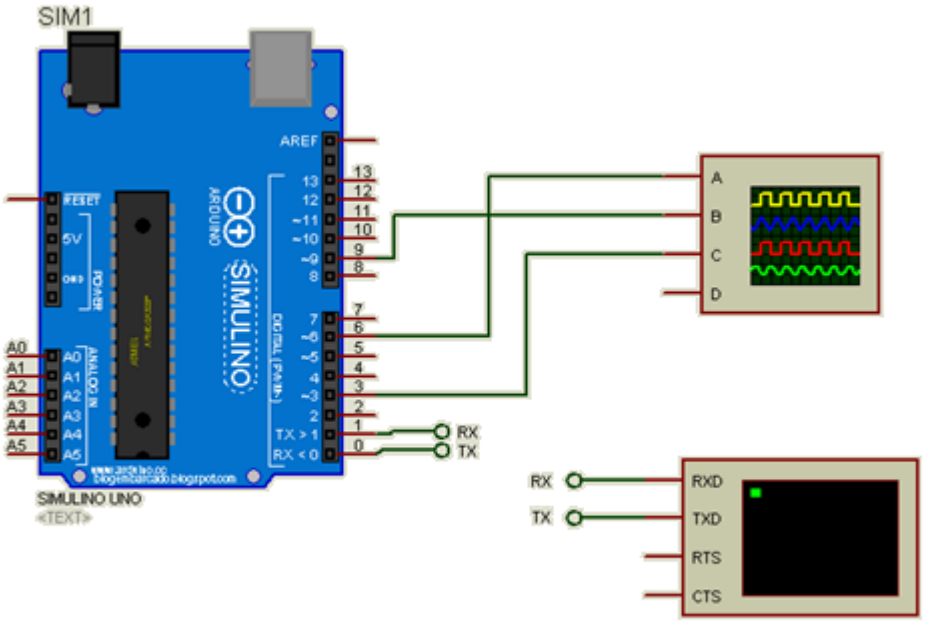
| 2; ➪ 0b00000010 TCCR2B final 3921 Hz

O você pode escrever o número 2 em binário ou hexadecimal.

```
TCCR2B=(TCCR2B & 0b11111000) | 0x02;
TCCR2B=(TCCR2B & 0b11111000) | 0b00000010;
```

Programa exemplo.

O circuito para teste é mostrado abaixo.



O programa exemplo abaixo altera a frequência do pino 03 para 3921 Hz Temporizador 2 onde o CS foi alterado usando o valor inteiro, que me parece o mais prático!

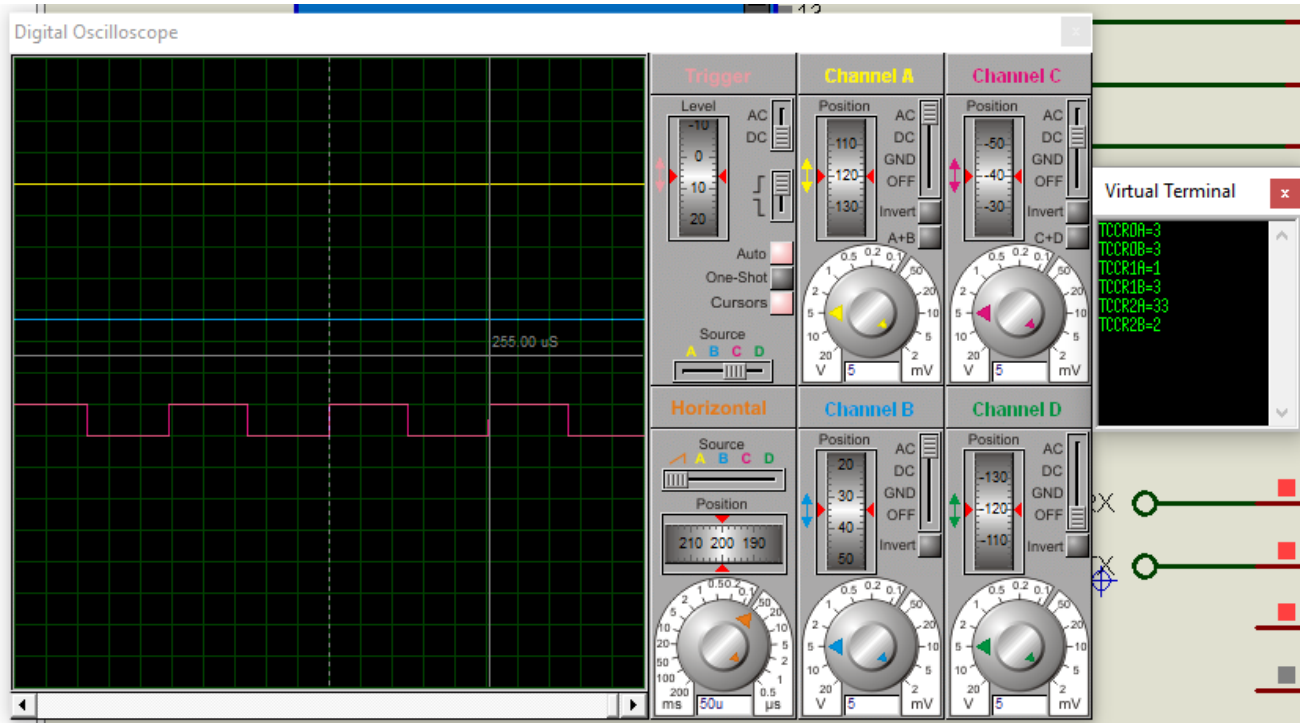
echo

```
int  numero;int k;int n;char sbuffer[255];
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(3,OUTPUT);
  TCCR2B=(TCCR2B & 0b11111000)| 2;//altera a frequência do temporizador 2 canais 11 e 03 para 3921Hz
}
void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(3,127);

  if(Serial.available())//se recebeu alguma coisa via serial
  {
    for (k=0;k<255;k++){sbuffer[k]=0;}//limpa o buffer
    numero=Serial.readBytesUntil(13,sbuffer,255);//le o dado da entrada serial até receber o ENTER
    Serial.print("TCCR0A=");Serial.println(TCCR0A); Serial.print("TCCR0B=");Serial.println(TCCR0B);
    Serial.print("TCCR1A=");Serial.println(TCCR1A);Serial.print("TCCR1B=");Serial.println(TCCR1B);
    Serial.print("TCCR2A=");Serial.println(TCCR2A);Serial.print("TCCR2B=");Serial.println(TCCR2B);
    Serial.println();Serial.setTimeout(30000);delay(1);
  }
}
```

Para copiar e colar.

```
int  numero;int k;int n;char sbuffer[255];
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(3,OUTPUT);
  TCCR2B=(TCCR2B & 0b11111000)| 2;//altera a frequência do temporizador 2 canais 11 e 03 para 3921Hz
}
void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(3,127);

  if(Serial.available())//se recebeu alguma coisa via serial
  {
    for (k=0;k<255;k++){sbuffer[k]=0;}//limpa o buffer
    numero=Serial.readBytesUntil(13,sbuffer,255);//le o dado da entrada serial até receber o ENTER
    Serial.print("TCCR0A=");Serial.println(TCCR0A); Serial.print("TCCR0B=");Serial.println(TCCR0B);
    Serial.print("TCCR1A=");Serial.println(TCCR1A);Serial.print("TCCR1B=");Serial.println(TCCR1B);
    Serial.print("TCCR2A=");Serial.println(TCCR2A);Serial.print("TCCR2B=");Serial.println(TCCR2B);
    Serial.println();Serial.setTimeout(30000);delay(1);
  }
}
```

O resultado da medição o pino 3 é mostrado na figura, note que eu continuo mostrando todos os registradores, mas a frequência do pino 3 é muito mais lata do que a dos outros pinos!



Como alterar a frequência alterando a configuração original dos temporizadores e os bits de frequência.

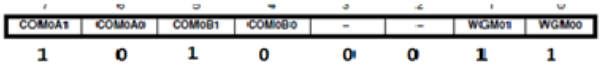
Este método é mais amplo você obtém mais possibilidades de valores de frequência, mas tem que alterar completamente a configuração normal dos temporizadores.

A forma normal dos temporizadores é mostrada abaixo, somente o Temporizador 0 usa o modo “Fast PWM”!

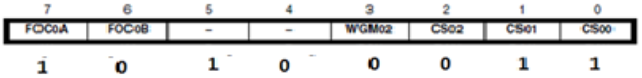
Os modos possíveis de operar são muitos e fogem ao escopo deste trabalho mostrar todos, assim vamos mostrar somente aqueles que fazem os temporizadores funcionar no modo ”Fast PWM’ e “PWM, phase correct”, sem alterar os bits COMnx!

A figura mostra os registradores dos temporizadores com seu valor normal para atuar na instrução analogWrite()!

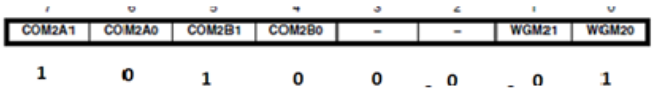
TCCOA(Inicial) =163=0b10100011



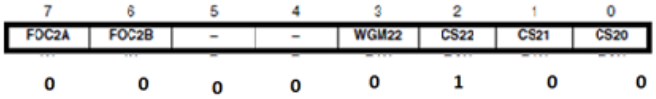
TCCR0B(Inicial) =3=0b00000011



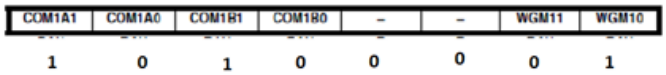
TCCR2A (Inicial) =161=0b10100001



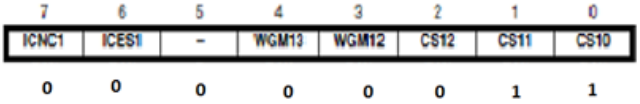
TCCR2B (Inicial) =4=0b00000100



TCCR1A(Inicial)=161=0b101000001



TCCR1B(Inicial)=3 =0b00000011



As tabelas mostram como programar os temporizadores nos dois modos.

Começando pelo TIMER0, neste temporizador o modo Fast PWM deve ser o preferido.

Pinos 05 e 06 **TIMER 0**

Table 15-8. Waveform Generation Mode Bit Description

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

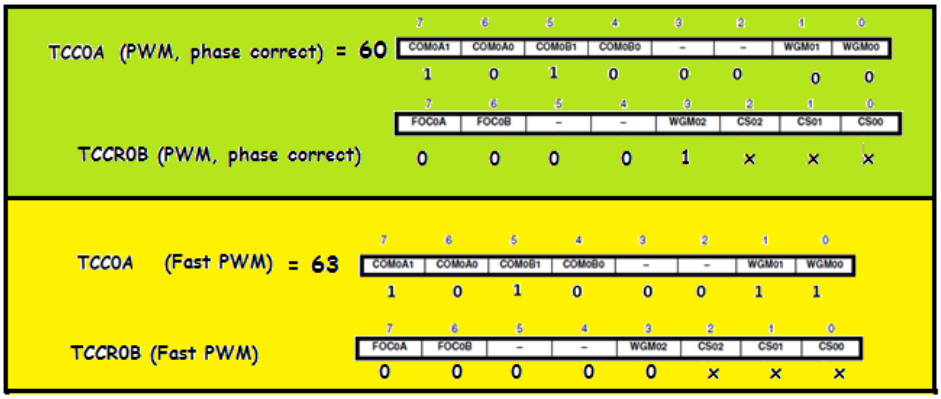


Table 15-9. Clock Select Bit Description

CS	CS02	CS01	CS00	Description
	0	0	0	No clock source (Timer/Counter stopped)
1	0	0	1	$clk_{I/O}$/(No prescaling)
2	0	1	0	$clk_{I/O}$/8 (From prescaler)
3	0	1	1	$clk_{I/O}$/64 (From prescaler)
4	1	0	0	$clk_{I/O}$/256 (From prescaler)
5	1	0	1	$clk_{I/O}$/1024 (From prescaler)
	1	1	0	External clock source on T0 pin. Clock on falling edge.
	1	1	1	External clock source on T0 pin. Clock on rising edge.

TIMER 0 Fast PWM

CS	N	fOCnxPWM	t(ms)
1	1	62500	0,016
2	8	7812,5	0,128
3	64	976,5625	1,024
4	256	244,140625	4,096
5	1024	61,03515625	16,384

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

TIMER 0 PWM, phase correct

CS	N	fOCnxPCPWM	t(ms)
1	1	31372,55	0,031875
2	8	3921,57	0,255
3	64	490,20	2,040
4	256	122,55	8,160
5	1024	30,64	32,640

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

No Timer 1 o PWM de 8 bits é o mais prático!

Os modos escolhidos são aqueles onde o valor do TOP é o máximo 0xFF (255).

A tabela abaixo mostra como escolher os registradores onde os valores indicados por ” x” devem ser preenchidos conforme o de CSnx da pré-escala!

Pinos 09 e 10 TIMER 1

Table 16-4. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

TCCR1A (PWM, phase correct) =161

7	6	5	4	3	2	1	0
COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10
1	0	1	0	0	0	0	1

TCCR1B (PWM, phase correct)

7	6	5	4	3	2	1	0
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
0	0	0	0	0	x	x	x

TCCR1A (Fast PWM) =161

7	6	5	4	3	2	1	0
COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10
1	0	1	0	0	0	0	1

TCCR1B (Fast PWM)

7	6	5	4	3	2	1	0
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
0	0	0	0	1	x	x	x

Table 16-5. Clock Select Bit Description

CS	CS12	CS11	CS10	Description
	0	0	0	No clock source (Timer/Counter stopped).
1	0	0	1	$clk_{I/O}$/1 (No prescaling)
2	0	1	0	$clk_{I/O}$/8 (From prescaler)
3	0	1	1	$clk_{I/O}$/64 (From prescaler)
4	1	0	0	$clk_{I/O}$/256 (From prescaler)
5	1	0	1	$clk_{I/O}$/1024 (From prescaler)
	1	1	0	External clock source on T1 pin. Clock on falling edge.
	1	1	1	External clock source on T1 pin. Clock on rising edge.

TIMER 1 Fast PWM

CS	N	$f_{OCnxPWM}$	t(ms)
1	1	62500	0,016
2	8	7812,5	0,128
3	64	976,5625	1,024
4	256	244,140625	4,096
5	1024	61,03515625	16,384

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

TIMER 1 PWM, phase correct

CS	N	$f_{OCnxPCPWM}$	t(ms)
1	1	31372,55	0,031875
2	8	3921,57	0,255
3	64	490,20	2,040
4	256	122,55	8,160
5	1024	30,64	32,640

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

Por fim o TIMER2, neste temporizador o FAst PWM deve ser o modo preferido.

Pinos 11 e 03 TIMER 2

Table 18-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

TCCR2A (PWM, phase correct) =61

7	6	5	4	3	2	1	0
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20
1	0	1	0	0	0	0	1

TCCR2B (PWM, phase correct)

7	6	5	4	3	2	1	0
FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
0	0	0	0	0	×	×	×

TCCR2A (Fast PWM) =163

7	6	5	4	3	2	1	0
COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20
1	0	1	0	0	0	1	1

TCCR2B (Fast PWM)

7	6	5	4	3	2	1	0
FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00
0	0	0	0	0	×	×	×

Table 18-9. Clock Select Bit Description

CS	CS22	CS21	CS20	Description
	0	0	0	No clock source (Timer/Counter stopped).
1	0	0	1	clk_{T2S}/(No prescaling)
2	0	1	0	clk_{T2S}/8 (From prescaler)
3	0	1	1	clk_{T2S}/32 (From prescaler)
4	1	0	0	clk_{T2S}/64 (From prescaler)
5	1	0	1	clk_{T2S}/128 (From prescaler)
6	1	1	0	clk_{T2S}/256 (From prescaler)
7	1	1	1	clk_{T2S}/1024 (From prescaler)

TIMER 2 Fast PWM

CS	N	fOCnxPWM	t(ms)
1	1	62500	0,016
2	8	7812,5	0,128
3	32	1953,125	0,512
4	64	976,5625	1,024
5	128	488,28125	2,048
6	256	244,140625	4,096
7	1024	61,03515625	16,384

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

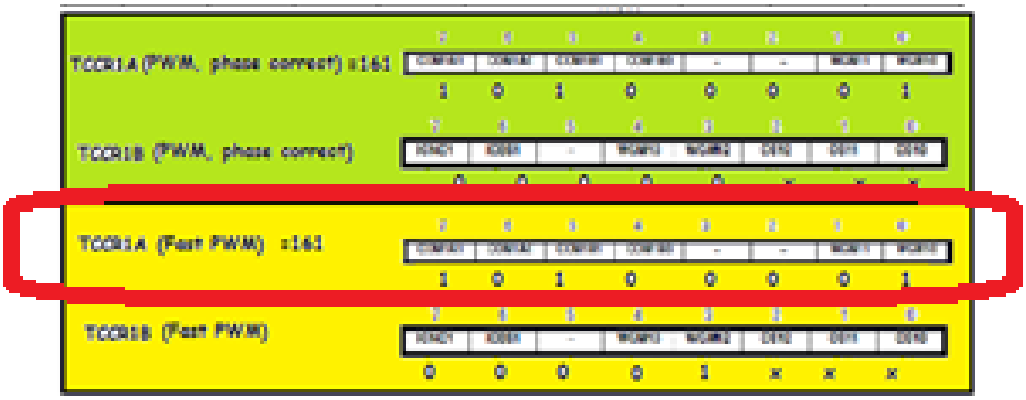
TIMER 2 PWM, phase correct

CS	N	fOCnxPWM	t(ms)
1	1	31372,54902	0,031875
2	8	3921,568627	0,255
3	32	980,3921569	1,02
4	64	490,1960784	2,04
5	128	245,0980392	4,08
6	256	122,5490196	8,16
7	1024	30,6372549	32,64

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

Exemplo 1:

Alterando a frequência do pino 9 para 7812Hz(0,128 ms =128 us) usando o modo “Fast PWM”, para isto você deve alterar o Temporizador 1 como descrito abaixo.



TIMFR 1 Fast PWM

CS	N	fOCnxPWM	t(ms)
1	1	62500	0,016
2	8	7812,5	0,128
3	64	976,5625	1,024
4	256	244,140625	4,096
5	1024	61,03515625	16,384

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot (1 + TOP)}$$

TIMER 1 PWM, phase correct

CS	N	fOCnxPCPWM	t(ms)
1	1	31372,55	0,031875
2	8	3921,57	0,255
3	64	490,20	2,040
4	256	122,55	8,160
5	1024	30,64	32,640

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{2 \cdot N \cdot TOP}$$

Aqui parece melhor programar em binário.

TCCR1A=0b10100001;//Timer 1 operando no modo “Fast PWM”

TCCR1B=0b00001010;//Frequência de 7812 Hz 0,128ms =128us

O programa completo:

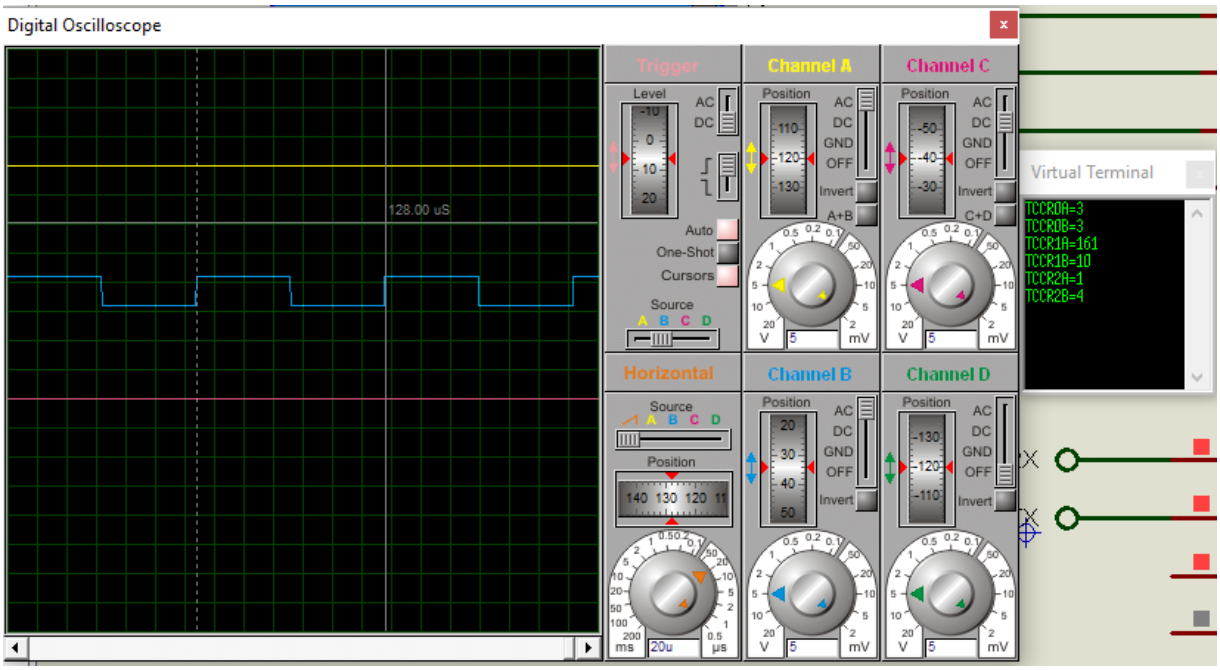
echo $

```
int  numero;int k;int n;char sbuffer[255];
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(9,OUTPUT);
  TCCR1A=0b10100001;//Timer 1 operando no modo "Fast PWM
  TCCR1B=0b00001010;//Frequência de 7812Hz 0,128 ms =128 us
}
void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(9,127);
  if(Serial.available())//se recebeu alguma coisa via serial
  {
      for (k=0;k<255;k++){sbuffer[k]=0;}//limpa o buffer
    numero=Serial.readBytesUntil(13,sbuffer,255);//le o dado da entrada serial até receber o ENTER
    Serial.print("TCCR0A=");Serial.println(TCCR0A);Serial.print("TCCR0B=");Serial.println(TCCR0B);
    Serial.print("TCCR1A=");Serial.println(TCCR1A);Serial.print("TCCR1B=");Serial.println(TCCR1B);
    Serial.print("TCCR2A=");Serial.println(TCCR2A);Serial.print("TCCR2B=");Serial.println(TCCR2B);
    Serial.println();Serial.setTimeout(30000);delay(1);
  }
}
```

Para copiar e colar.

```
int  numero;int k;int n;char sbuffer[255];
void setup() {
 // put your setup code here, to run once:
 // initialize serial communication at 9600 bits per second:
 Serial.begin(9600);
 pinMode(9,OUTPUT);
 TCCR1A=0b10100001;//Timer 1 operando no modo “Fast PWM”
 TCCR1B=0b00001010;//Frequência de 7812 Hz 0,128ms =128us
}
void loop() {
 // put your main code here, to run repeatedly:
 analogWrite(9,127);
 if(Serial.available())//se recebeu alguma coisa via serial
 {
   for (k=0;k<255;k++){sbuffer[k]=0;}//limpa o buffer
   numero=Serial.readBytesUntil(13,sbuffer,255);//le o dado da entrada serial até receber o ENTER
   Serial.print("TCCR0A=");Serial.println(TCCR0A); Serial.print("TCCR0B=");Serial.println(TCCR0B);
   Serial.print("TCCR1A=");Serial.println(TCCR1A);Serial.print("TCCR1B=");Serial.println(TCCR1B);
   Serial.print("TCCR2A=");Serial.println(TCCR2A);Serial.print("TCCR2B=");Serial.println(TCCR2B);
   Serial.println();Serial.setTimeout(30000);delay(1);
 }
}
```

O Resultado:


Digital Oscilloscope
128.00 uS
Trigger
Channel A
Channel C
Horizontal
Channel B
Channel D
Virtual Terminal
TCCR0A=3
TCCR0B=3
TCCR1A=161
TCCR1B=10
TCCR2A=1
TCCR2B=4

Exemplo 2:

Alterando a frequência do pino 3 para 7815Hz(0,128 ms =128 us), para isto você deve alterar o Temporizador 2 como descrito abaixo.

TIMER 2 Fast PWM

CS	N	fOCnxPWM	t(ms)
1	1	62500	0,016
2	8	7812,5	0,128
3	32	1953,125	0,512
4	64	976,5625	1,024
5	128	488,28125	2,048
6	256	244,140625	4,096
7	1024	61,03515625	16,384

$$f_{OCnxPWM} = \frac{f_{clk\_I/O}}{N \cdot 256}$$

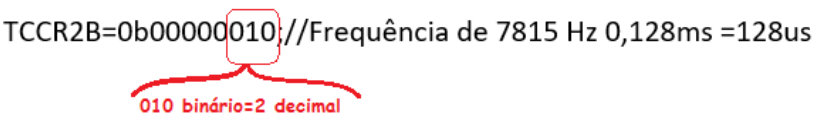
TIMER 2 PWM, phase correct

CS	N	fOCnxPWM	t(ms)
1	1	31372,54902	0,031875
2	8	3921,568627	0,255
3	32	980,3921569	1,02
4	64	490,1960784	2,04
5	128	245,0980392	4,08
6	256	122,5490196	8,16
7	1024	30,6372549	32,64

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$

TCCR2A=0b10100011;//Timer 2 operando no modo “Fast PWM”

TCCR2B=0b00000010;//Frequência de 7815 Hz 0,128ms =128us



O programa:

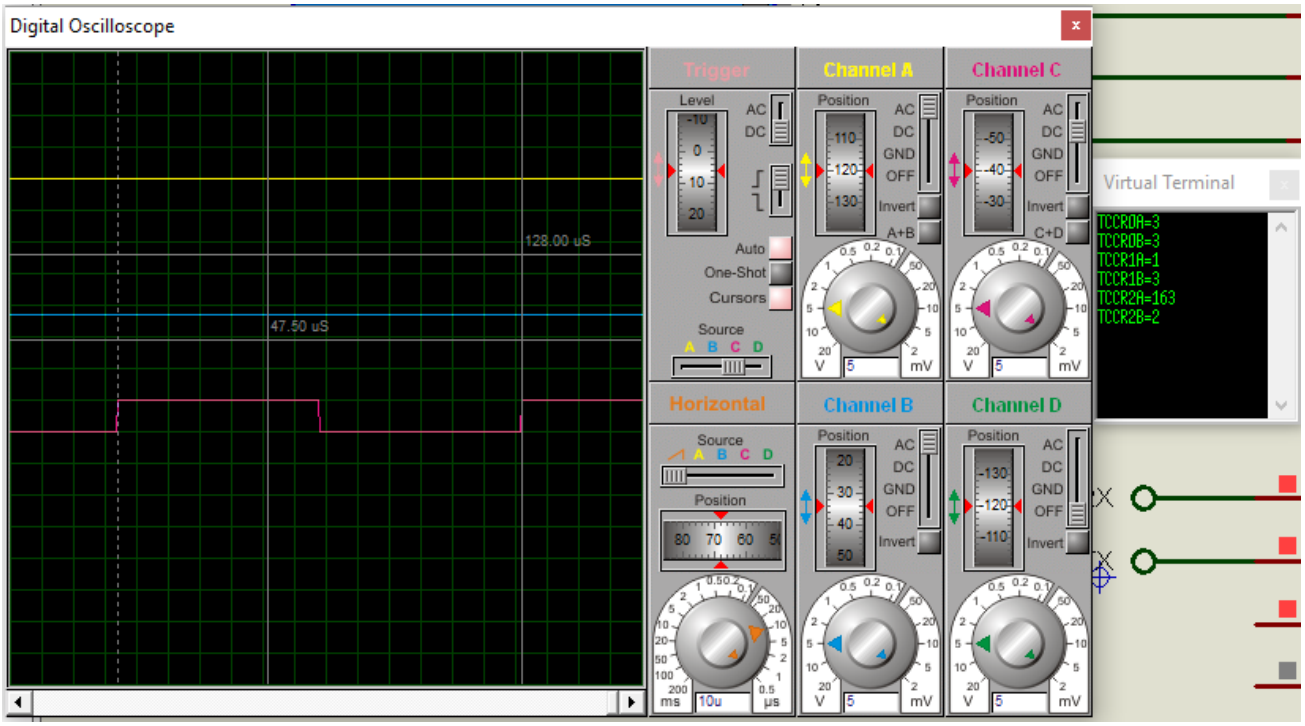
echo $

```
int  numero;int k;int n;char sbuffer[255];
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(3,OUTPUT);
  TCCR2A=0b10100011;//Timer 2 operando no modo “Fast PWM”
  TCCR2B=0b00000010;//Frequência de 7815 Hz 0,128ms =128us

}
void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(3,127);
  if(Serial.available())//se recebeu alguma coisa via serial
  {
    for (k=0;k<255;k++){sbuffer[k]=0;}//limpa o buffer
    numero=Serial.readBytesUntil(13,sbuffer,255);//le o dado da entrada serial até receber o ENTER
    Serial.print("TCCR0A=");Serial.println(TCCR0A); Serial.print("TCCR0B=");Serial.println(TCCR0B);
    Serial.print("TCCR1A=");Serial.println(TCCR1A);Serial.print("TCCR1B=");Serial.println(TCCR1B);
    Serial.print("TCCR2A=");Serial.println(TCCR2A);Serial.print("TCCR2B=");Serial.println(TCCR2B);
    Serial.println();Serial.setTimeout(30000);delay(1);
  }
```

Para colar e copiar.

```
int  numero;int k;int n;char sbuffer[255];
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(3,OUTPUT);
  TCCR2A=0b10100011;//Timer 2 operando no modo “Fast PWM”
  TCCR2B=0b00000010;//Frequência de 7815 Hz 0,128ms =128us
}
void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(3,127);
  if(Serial.available())//se recebeu alguma coisa via serial
  {
    for (k=0;k<255;k++){sbuffer[k]=0;}//limpa o buffer
    numero=Serial.readBytesUntil(13,sbuffer,255);//le o dado da entrada serial até receber o ENTER
    Serial.print("TCCR0A=");Serial.println(TCCR0A); Serial.print("TCCR0B=");Serial.println(TCCR0B);
    Serial.print("TCCR1A=");Serial.println(TCCR1A);Serial.print("TCCR1B=");Serial.println(TCCR1B);
    Serial.print("TCCR2A=");Serial.println(TCCR2A);Serial.print("TCCR2B=");Serial.println(TCCR2B);
    Serial.println();Serial.setTimeout(30000);delay(1);
  }
}
```

O resultado:


Digital Oscilloscope
128.00 uS
47.50 uS
Trigger
Channel A
Channel C
Horizontal
Channel B
Channel D
Virtual Terminal
TCCR0A=3
TCCR0B=3
TCCR1A=1
TCCR1B=3
TCCR2A=163
TCCR2B=2

Conclusão:

Você viu como alterar a frequência do PWM do Arduino usando dois métodos, no primeiro você simplesmente altera a frequência; no segundo você altera o modo de gerar o PWM e a frequência.

Referências.

Bibliografia.

Manual da Atmel para o microcontrolador ATmega328P.

PDF:

Sites: www.bairrospd.com

SEO: www.bairrospd.com, Arduino, PWM, alterando a frequência do PWM, senóide, LED, eletrônica, tutorial