Configuração do PWM da placa Arduino UNO.

Por Eng. Roberto Bairros dos Santos

www.bairrospd.com

Data: 07/07/2017

Sumário

Prefácio.

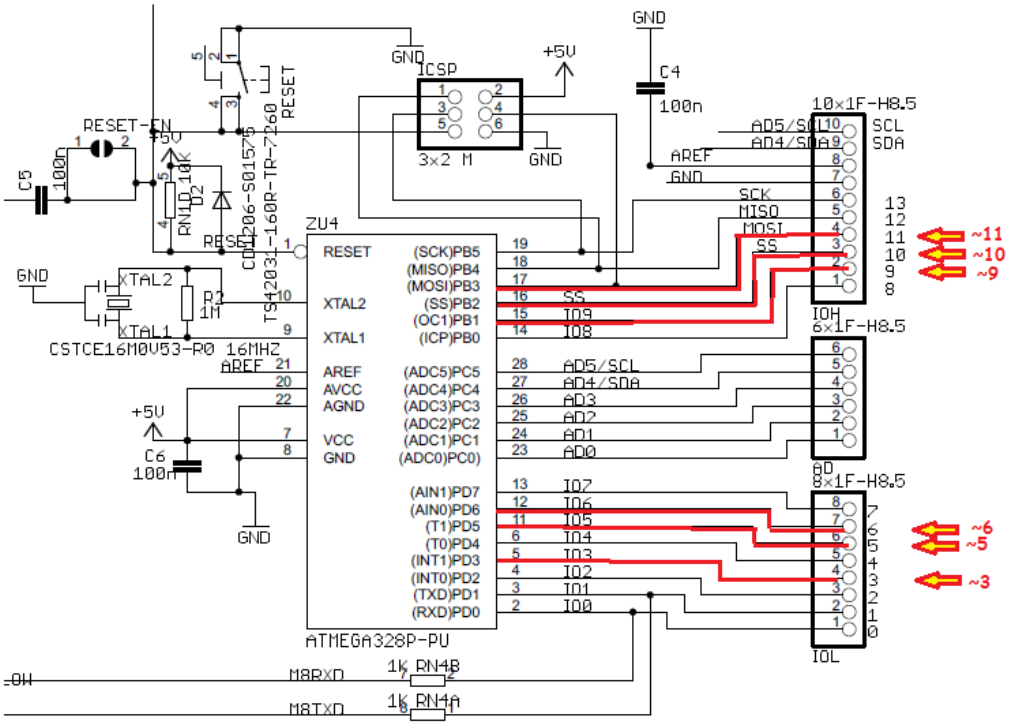
A função analogWrite() usa os temporizadores do microcontrolador ATmega328P, neste tutorial você vai ver como estes temporizadores são configurados para uso normal, sem alterar a frequência original da Arduino.

O detalhe da configuração pode ser visto no tutorial 01 Detalhando o PWM do Arduino.

Relação entre os temporizadores e os pinos do Arduino UNO.

Para aplicar estes conhecimentos você deve saber a relação entre os pinos da placa Arduino e o Temporizador, para isto você deve olhar o diagrama e identificar as conexões entre os pinos do microcontrolador e os pinos de saída da placa Arduino UNO. Os pinos na placa Arduino que suportam PWM são marcados com o sinal "~"!

A figura abaixo mostra esta ligação.



A tabela abaixo mostra a relação entre os pinos do microcontrolador o temporizador que controla aquele pino e o pino da placa Arduino UNO. A frequência que aparece na descrição é referente a frequência normal de operação!

Pino Arduino	Pino MEGA386P	Descrição
3	PD3	(PCINT19/OC2B/INT1) 490Hz
5	PD5	(PCINT21/OC0B/T1) 980Hz
6	PD6	(PCINT22/OC0A/AIN0) 980Hz
9	PB1	(OC1A/PCINT1) 490Hz
10	PB2	(SS/OC1B/PCINT2) 490Hz
11	PB3	(MOSI/OC2A/PCINT3) 490Hz

A figura abaixo mostra a relação entre os temporizadores e seus registradores e o pino da saída Arduino.

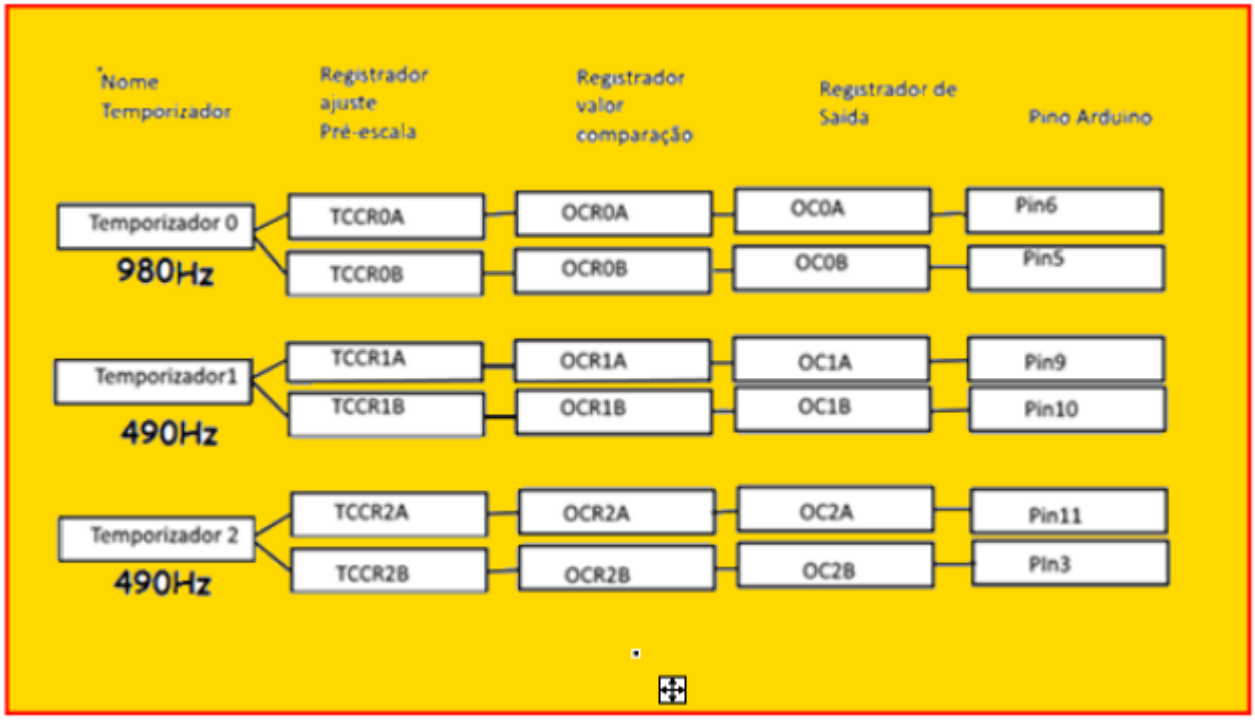
Numa aplicação primeiro você escolhe o pino ou pinos onde será usado o PWM, depois você usa a figura para escolher o temporizador e saber quais os registradores estão associados ao pino escolhido!

A frequência mostrada a direita é a frequência normalmente usada na instrução analogWrite()!

Os pinos 5 e 6 trabalham com uma frequência de PWM mais alta dos que os outros pinos.

Note que cada temporizador controla dois pinos do Arduino!

Outro detalhe importante que o valor de contagem máxima (TOP) é de 255!



Configuração normal do Timer 0 para a placa Arduino UNO.

A figura mostra o valor dos registradores do TIMER 0 lembrando que o registrador que define o ciclo de trabalho (duty cycle) é o registrador TCCR0A que comando o pino de saída 06 e TCCR0B que comanda o pino de saída 05!

Este temporizador opera no modo “Fast PWM” a uma frequência de 976Hz com período de 1,024ms!

Este são os dois pinos que operam com a frequência mais alta de PWM quando a placa UNO é usada sem alteração nos temporizadores!

TCCR0A(Inicial) =163=0b10100011

7	6	5	4	3	2	1	0
COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00
1	0	1	0	0	0	1	1

Clear OC0B on Compare Match, set OC0B at BOTTOM, (non-inverting mode)	1	0	1	0

TCCR0B(Inicial) =3=0b00000011

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

Mode	WGM02	WGM01	WGM00	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on(1)(2)
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX

Table 15-9. Clock Select Bit Description

CS02	CS01	CS00	Description
0	1	1	$clk_{I/O}$/64 (From prescaler)

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

N=64

N	fOCnxPCPWM	t(ms)
64	976.5625	1.024

Configuração normal do Timer 1 para a placa Arduino UNO.

A figura mostra o valor dos registradores lembrando que o registrador que define o ciclo de trabalho (duty cycle) é o registrador ODRC0A que comando o pino de saída 09 e OCR0B que comanda o pino de saída 10!

Este temporizador opera no modo “PWM, Phase Correct, 8-bit” a uma frequência de 490Hz com período de 2,040ms!

TCCR1A(Inicial)=161=0b101000001

COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10
1	0	1	0	0	0	0	1

Clear OC0B on Compare Match, set OC0B at BOTTOM, (non-inverting mode)	1	0	1	0

TCCR1B(Inicial)=3 =0b00000011

7	6	5	4	3	2	1	0
ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
0	0	0	0	0	0	1	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
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM

Table 16-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	1	1	$clk_{I/O}$/64 (From prescaler)

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

N=64

	N	fOCnxPCPWM	t(ms)
3	64,00	490,20	2,040

Configuração normal do Timer 2 para a placa Arduino UNO.

A figura mostra o valor dos registradores lembrando que o registrador que define o ciclo de trabalho (duty cycle) é o registrador ODRC0A que comando o pino de saída 11 e OCR0B que comanda o pino de saída 03!

Este temporizador opera no modo “PWM, Phase Correct” a uma frequência de 490Hz com período de 2,040ms!

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM

Table 18-9. Clock Select Bit Description

CS22	CS21	CS20	Description
1	0	0	clk_{T2S}/64 (From prescaler)

$$f_{OCnxPCPWM} = \frac{f_{clk\_I/O}}{N \cdot 510}$$ N=64

TIMER 2 PWM, phase correct

CS	N	fOCnxPWM	t(ms)
4	64	490,1960784	2,04

Como medir a frequência dos temporizadores na placa Arduino UNO.

O diagrama abaixo mostra como medir a frequência dos temporizadores.

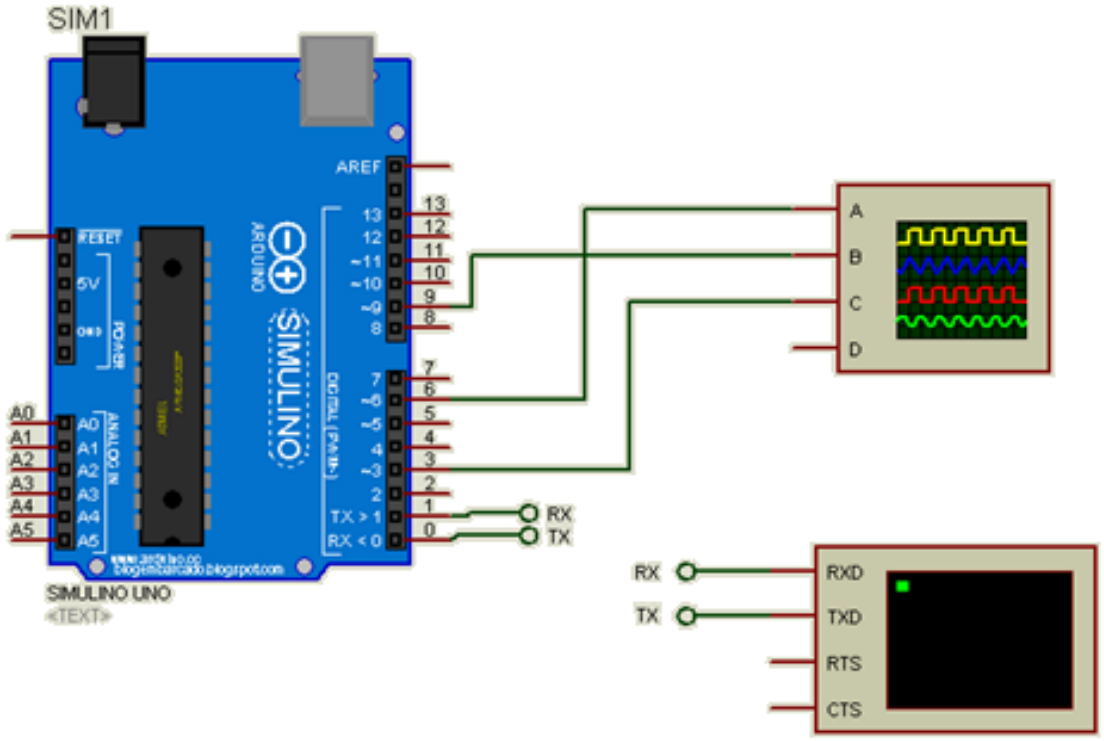
O osciloscópio foi ligado para mostrar uma sáida de cada temporizador.

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 diagrama.

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



O programa é mostrado abaixo e foi gerado a partir do exemplo “ Echo” da comunicação serial.

Para mostrar o valor somente uma vez foi inserido a linha abaixo.
numero=Serial.readBytesUntil(13,sbuffer,255);

Para ativar o PWM você deve configurar o pino como saída e o PWM só é ativado a partir das instrução analogWrite()!

Note que ao ligar o circuito a serial não mostra nenhum valor pois a primeira instrução fica esperando um dado via serial, para que os valores possam ser visualizados digite qualquer coisa e o ENTER ou simplesmente ENTER no terminal da serial!

echo

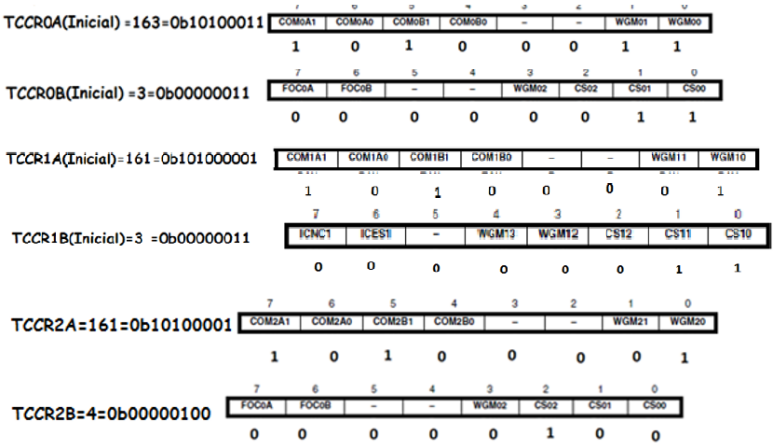
```
int  numero;int k;int n;char sbuffer[255];
void setup() {
  // put your setup code here, to run once:
  // initialize serial communication at 9600 bits per second:
  Serial.begin(9600);
  pinMode(3,OUTPUT);pinMode(11,OUTPUT);pinMode(9,OUTPUT);pinMode(10,OUTPUT);pinMode(6,OUTPUT);pinMode(5,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:
  analogWrite(3,127);analogWrite(11,127);analogWrite(9,127);analogWrite(10,127);analogWrite(6,127);analogWrite(5,127);
  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.println("TCCR0A=");Serial.println(TCCR0A);  Serial.println("TCCR0B=");Serial.println(TCCR0B);
    Serial.println("TCCR1A=");Serial.println(TCCR1A); Serial.println("TCCR1B=");Serial.println(TCCR1B);
    Serial.println("TCCR2A=");Serial.println(TCCR2A);Serial.println("TCCR2B=");Serial.println(TCCR2B);
    Serial.println();Serial.setTimeout(30000);delay(1);
  }
}

```

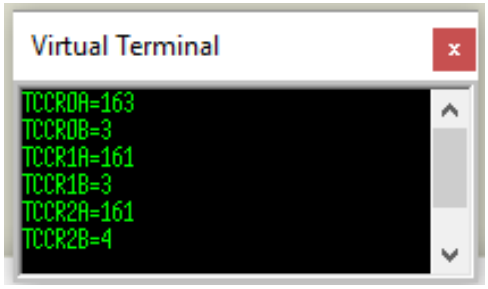
Para copiar e colar.

```
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);pinMode(11,OUTPUT);pinMode(9,OUTPUT);pinMode(10,OUTPUT);pinMode(6,OUTPUT
);pinMode(5,OUTPUT);
}
void loop() {
  // put your main code here, to run repeatedly:

analogWrite(3,127);analogWrite(11,127);analogWrite(9,127);analogWrite(10,127);analogWrite(6,127);analo
gWrite(5,127);
  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.println("TCCR0A=");Serial.println(TCCR0A);  Serial.println("TCCR0B=");Serial.println(TCCR0B);
    Serial.println("TCCR1A=");Serial.println(TCCR1A); Serial.println("TCCR1B=");Serial.println(TCCR1B);
    Serial.println("TCCR2A=");Serial.println(TCCR2A);Serial.println("TCCR2B=");Serial.println(TCCR2B);
    Serial.println();Serial.setTimeout(30000);delay(1);
  }
}
```

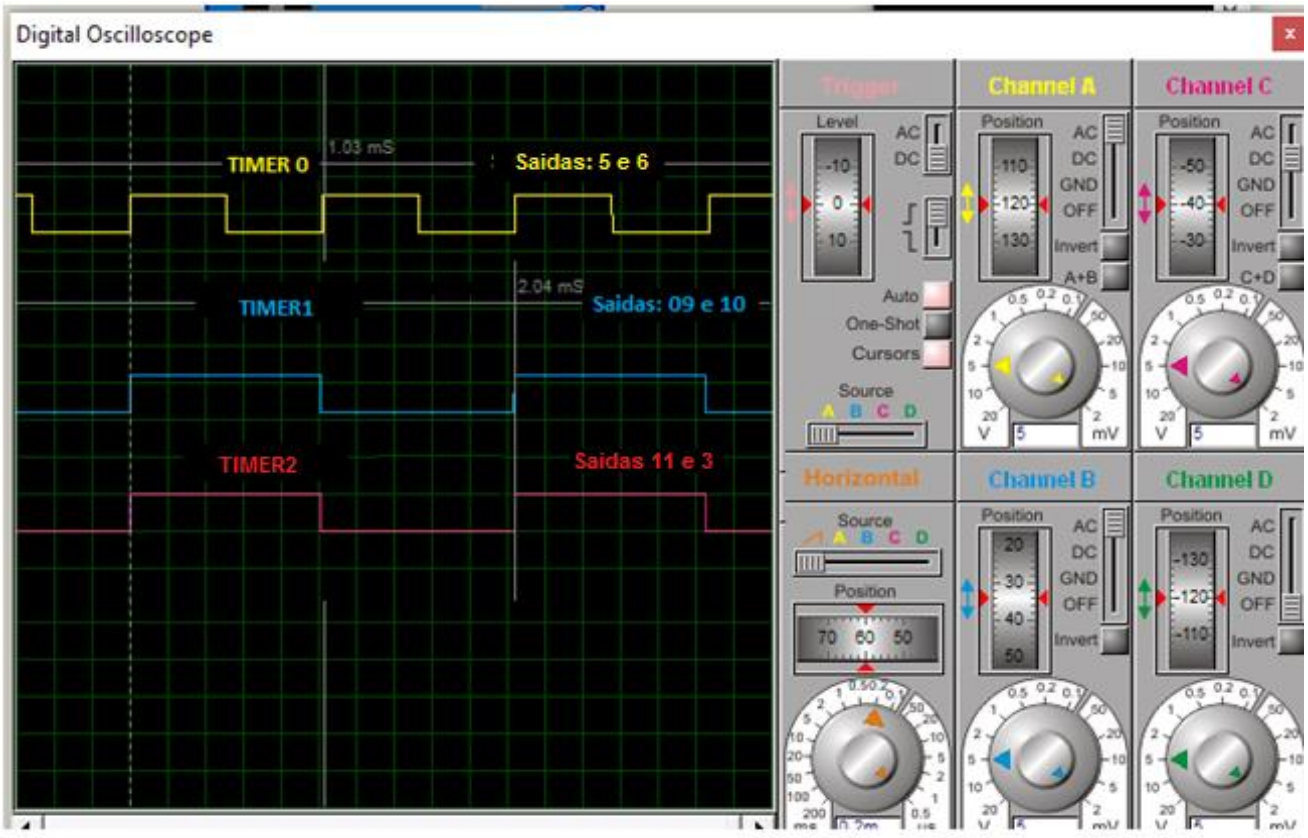
O resultado é mostrado na figura abaixo com os tempos comentados e a caixa do terminal mostrando os valores dos temporizadores.

Os valores dos registradores dos temporizadores são mostrados no terminal, estes são os valores padrões da placa Arduino UNO e você pode conferir nas tabelas mostradas antes!





O osciloscópio mostra os tempos de um ciclo de cada um dos temporizadores



Resumo dos temporizadores da placa Arduino UNO.

A tabela abaixo mostra um resumo quanto a frequência do PWM dos temporizadores usados na placa Arduino UNO, esta tabela será útil para alterar a frequência!

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

Conclusão.

Para trabalhar com a função analogWrite() do Arduino na placa UNO as saídas 5 e 6 trabalham na frequência mais alta de 980Hz e as saídas 3, 11, 9 e 10 operam a frequência mais baixa de 490Hz. O modo de operação das saídas 5 e 6 é “Fast PWM” e as outras saídas operam no modo “Phase Correct” por isto a frequência mais baixa.

Créditos.

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