PROFESSOR DANILO *PROJETOS DE CIÊNCIAS – 23/03/2022*

AULA 08

Hoje vamos trabalhar com a porta PWM. Primeir, vamos entender o que é a porta PWM.

O QUE É PWM

Voltemos à aula 05, aquela que usamos um transistor e que voltaremos a usar, provavelmente, na próxima aula: lá, nós vimos que o Arduino é incapaz de produzir sinais analógicos, mas ele é capaz de emular isso.

As portas que podem ser usadas com tais tecnologias são marcadas com um ~ no Arduino.

Me baseei fortemente no artigo abaixo para montar este material:

https://www.embarcados.com.br/pwm-do-arduino/

Na figura a seguir destacamos as portas PWM do Arduino UNO. São eslas as portas digitais 3, 5, 6, 9, 10 e 11.

Não precisa decorer, pois basta localizar onde estão as portas com o ~ ao lado. Note também que no próprio Arduino ele te lembra ao escrever “DIGITAL PWM ~”.

Abaixo da figura, transcrevo parte de um texto do site sugerido acima:

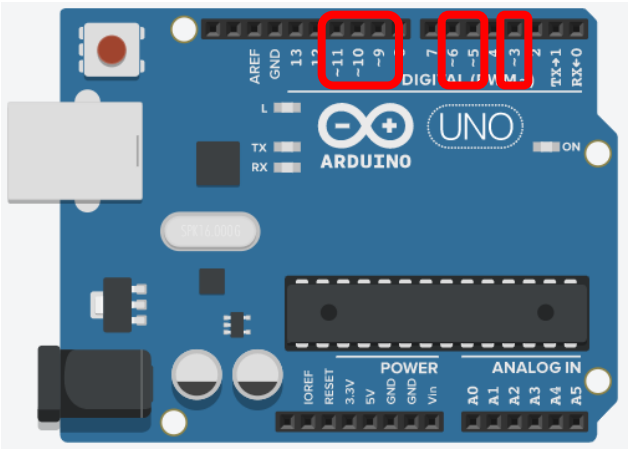


Figura 1: detalhe mostrando quais as portas do Arduino UNO funcionam como portas PWM.

“PWM, do inglês *Pulse Width Modulation*, é uma técnica utilizada por sistemas digitais para variação do valor médio de uma forma de onda periódica. A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto. Esse tempo é chamado de *duty cycle*, ou seja, o ciclo ativo da forma de onda. No gráfico abaixo são exibidas algumas modulações PWM:”

Mas é claro que o ciclo de trabalho (*duty cycle*) não pode ser de alguns segundos, portanto, qual seria esta frequência? Isso depende de cada pino ou mesmo do microcontrolador, pois o que estamos usando (Arduino UNO) é o ATMega 328p (o verdadeiro cérebro do Arduino), mas no nosso caso a resposta é de 490 Hz, podendo chegar a 980 Hz no pino 5 e 6.

Olha que interessante: se você ficar ligando e desligando um LED muito rapidamente com, digamos, 5 V, o LED poderá funcionar como se estivesse sendo alimentado com uma tensão menor. Por exemplo, se usarmos uma tensão de 5 V e um *Duty Cycle* de 50% então é como se alimentássemos o LED com 2,5 V. Esta tensão equivalente chamaremos de V_{out}, a tensão máxima de alimentação (5 V no Arduino) de V_{CC} então podemos ter a seguinte relação:

$$V_{out} = \frac{Duty\ Cycle}{100} \cdot V_{CC}.$$

As portas PWM podem ser usadas para controle de velocidade de motores, variação a luminosidade de LEDs, gerar sinais de áudio ou gerar sinais analógicos por razões diversas etc.

No Arduino, quando formos escrever o código, ao *Duty Cycle* não será indicado diretamente, isto é, não indicamos que o tempo de cclo ativo é um número que varia de 0% (totalmente desligado) até 100% (totalmente ligado): usamos valores que variam de 0 a 255. Assim, a fórmula acima pode ser reescrita como:

$$V_{out} = \frac{\text{Valor informado ao Arduino}}{255} \cdot V_{CC}.$$

Como exemplo, se informarmos 0, então a saída ficará completamente desligada; informando 255 ficará totalmente ligada; informando 127, ficará desligado o mesmo tempo que ficaria desligado e assim por diante.

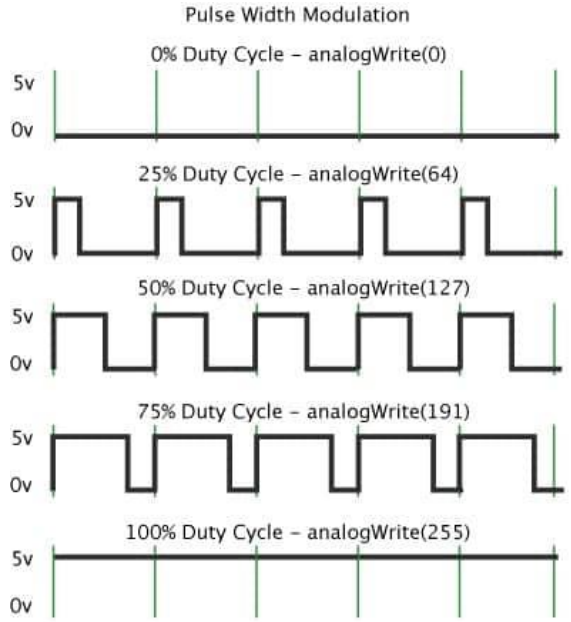


Figura 2: note que as portas PWM simulam uma saída digital ligando e desligando a porta muito rapidamente.

Na práticam como usamos isso no Arduino?

Primeiramente, quando falamos de portas analógicas de entrada ou as de saída emuladas (chamaremos, eventualmente, as portas PWM de saídas analógicas emuladas, pois o Arduino usa uma função que faz referência à palavra “*analógic*”), não precisam ser configuradas no setup(), assim basta irmos para o comando:

```
analogWrite(PINO, VALOR);
```

O PINO deve ser um número dentre os presentes na seguinte lista: (3, 5, 7, 9, 10, 11). No caso do VALOR, deve ser um número entre 0 e 255.

Note que temos um total de 256 números possíveis e que $2^8 = 256$, por isso dizemos que esta porta é uma saída de 8 bits.

LED RGB

Existem dois tipos de LEDs RGB, sendo todos eles com um pino maior e, na sequência da esquerda para a direita, deixando o pino maior à esquerda (como na figura abaixo), temos a seguinte ordem para os pinos:

VERMELHOR
COMUM
VERDE
AZUL

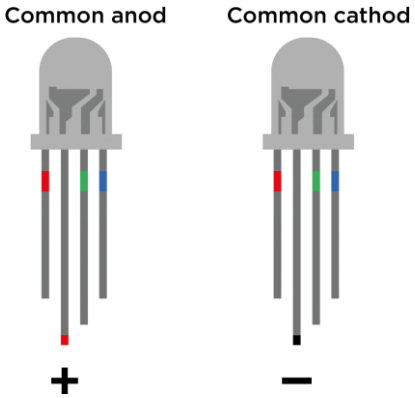


Figura 3: LEDs com anodo comum (à esquerda) e catodo comum (à direita)

Basicamente, o COMUM é o fio que é comum à todos os três LEDs, pois como podemos imaginar, um LED RGB é nada mais que três LEDs encapsulados em um único corpo. Assim, o LED com ânodo comum, deve ter o pino mais longo conectado ao positivo e para ligar cada um dos LEDs devemos conectar cada uma ao GND. Por exemplo: para que o LED da esquerda fique verdemlho, devemos conectar o pino mais à esquerda ao GND e o segundo à fonte de energia.

O segundo LED, que nós usaremos em sala, tem o GND como pino comum, sendo chamado de cátodo comum. Portanto, se quisermos ligar o segundo LED acima com a cor vermelha, devemos ligar a fonte de energia no primeiro pino e o GND no segundo.

O CIRCUITO

Como dito acima, vamos utilizar um LED com cátodo comum, isto é, vamos utilizar o LED representado à direita na figura acima. Da esquerda para a direita, faremos a seguinte conexão:

Pino PWM 3 (~3)
GND
Pino PWM 5 (~5)
Pino PWM 6 (~6)

A sequência de fotos abaixo pode te ajudar bem como a imagem a seguir.

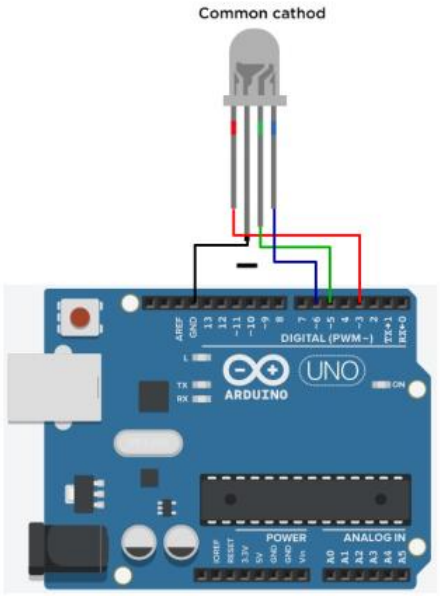
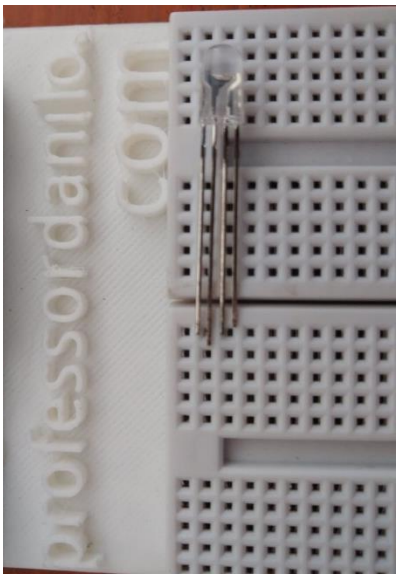
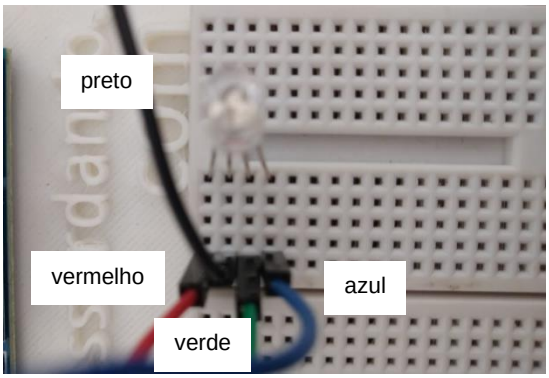


Figura 4: Esquema da nossa montagem

Para ver como fica no circuito que você possui, isto é, na placa de ensaio, veja a sequência de imagens a seguir. Se preferir ver colorida, baixe este arquivo via QR-code, após as imagens.





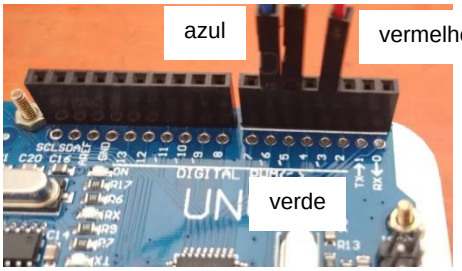


Figura 5: sequência de fotos mostrando a montagem do circuito

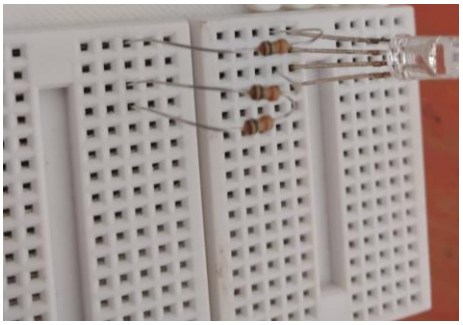
Para baixar este arquivo e poder ver as imagens acima coloridas, acesse:

Figura 6: QR-code para baixar este material em formato pdf

ATENÇÃO: NÓS NÃO UTILIZAREMOS RESISTORES PARA PROTEGER OS LEDS. SE QUISER PROTEGER OS LEDS UTILIZE UM RESISTOR PARA CADA LED/COR.

Mas qual seria a ventagem de usar os resistores? Várias: proteção dos LEDs, proteção para o Arduíno, maior possibilidade de escolhas de cores quando usarmos o comando analogWrite(PINO, VALOR);. Sem os resistores, VALOR DEVE TER VALOR MÁXIMO IGUAL A 10. É possível que os LEDs funcionem com valores maiores, mas tentar isso é colocar em risco a integridade tanto do Arduino quanto dos LEDs.

Então, por que não usaremos resistores? Apenas por praticidade, pois não temos o LED adequado para atingirmos um bom brilho nos LEDs e, como temos somente uma aula semanal, quanto menor o circuito, melhor (gerenciamento de tempo).



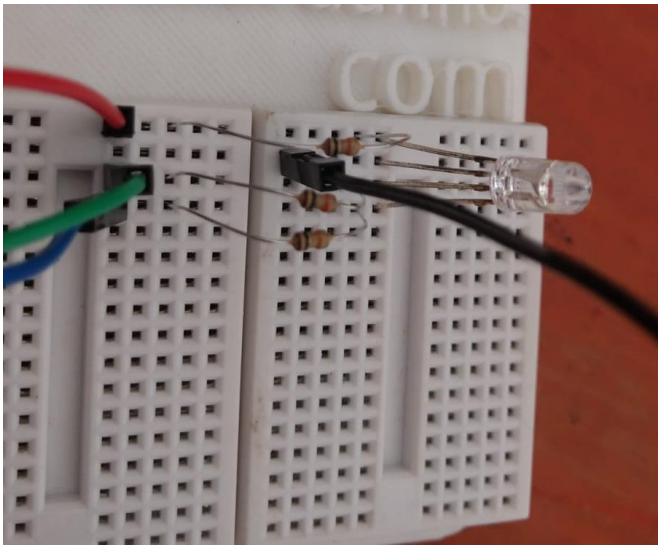


Figura 7: Detalhes das conexões dos resistores e dos cabos caso tente usar o resistor para aumentar os valores nas saídas PWM

SKETCH

O programa está na página seguinte para facilitar a cópia. Começaremos aqui então a discussão do código.

Na primeira linha, declaramos variáveis para que fique mais fácil mudar os pinos, caso seja necessário.

Note que o setup está vazio: somente temos um comentário no seu interior, mas ele poderia ser omitido. Isso ocorre porque não precisamos declarar as portas analógicas como entrada ou saída, uma vez que as portas de A0 até A5 são todas de entradas e as portas PWM de números 3, 5, 6, 9, 10 e 12 são sempre saídas.

Olhe que dentro do loop temos apenas as chamadas para as funções:

```
Vermelho();
Verde();
Azul();
Ciano();
Magenta();
Amarelo();
Branco();
```

Estes itens são chamados de funções e podemos defini-las separadamente para serem chamadas dentro de outras funções.

```
int redPin = 3, greenPin = 5, bluePin = 6;

void setup() {
//portas analógicas não precisam ser configuradas
}

void loop() {
  Vermelho();
  delay(1000);
  Verde();
  delay(1000);
  Azul();
  delay(1000);
  Ciano();
  delay(1000);
  Magenta();
  delay(1000);
  Amarelo();
  delay(1000);
  Branco();
  delay(1000);
}

void Vermelho(){
  analogWrite(redPin, 10);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 0);
}
void Verde(){
  analogWrite(redPin, 0);
  analogWrite(greenPin, 10);
  analogWrite(bluePin, 0);
}
void Azul(){
  analogWrite(redPin, 0);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 10);
}
void Ciano(){
  analogWrite(redPin, 0);
  analogWrite(greenPin, 10);
  analogWrite(bluePin, 10);
}
void Magenta(){
  analogWrite(redPin, 10);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 10);
}
void Amarelo(){
  analogWrite(redPin, 10);
  analogWrite(greenPin, 10);
  analogWrite(bluePin, 0);
}
void Branco(){
  analogWrite(redPin, 10);
  analogWrite(greenPin, 10);
  analogWrite(bluePin, 10);
}
```

Vamos analisar mais detalhadamente apenas uma das funções:

```
void Vermelho(){
  analogWrite(redPin, 10);
  analogWrite(greenPin, 0);
  analogWrite(bluePin, 0);
}
```

Esta função, chamada de Vermelho(), é sempre ativada (dizemos que é chamada) sempre que escrevemos Vermelho(); em alguma parte do código. Isto é, quando ela é chamada, o LED RGB terá apenas o LED correspondente ao vermelho aceso e as demais apagadas.

Note que antes de começarmos a descrever o que a função faz, escrevemos `void`, que significa "vazio", isto significa que esta função executa uma rotina, mas não devolve nenhum valor.

No próximo bimestre, o seu professor irá passar algumas atividades de programação, permitindo a você fazer um melhor uso deste conhecimento.

As demais funções seguem o mesmo princípio. Vamos descrever apenas quais pinos devem ser acionados quando chamamos cada uma das funções.

Tabela 1: Cor do LED desejada e pinos a serem ligados para se obter a cor desejada.

COR	PINOS LIGADOS
Vermelha	redPin
Vermelha	greenPin
Azul	bluePin
Ciano	greenPin bluePin
Magenta	redPin bluePin
Amarelo	redPin greenPin
Branco	redPin greenPin bluePin

ATENÇÃO

CUIDADO: NOTE QUE OS VALORES NA FUNÇÃO analogWrite NÃO PASSAM DE 10. NÃO ALTERE ESSE VALOR, COLOCANDO VALORES MAIORES, SEM O USO DE RESISTORES.

Você pode colocar valores menores, se desejar.

Tente outras cores, se desejar.

Tabela 2: Novas cores para você se divertir.

COR	PINOS LIGADOS

Laranja	`analogWrite(redPin, 10);`
	`analogWrite(greenPin, 5);`
	`analogWrite(bluePin, 0);`

Anil	`analogWrite(redPin, 3);`
	`analogWrite(greenPin, 0);`
	`analogWrite(bluePin, 5);`

Violeta	`analogWrite(redPin, 6);`
	`analogWrite(greenPin, 0);`
	`analogWrite(bluePin, 10);`

Note que com as cores acima você pode reproduzir todas as cores que dizermos ter o arco-íris:

Vermelho, Laranja, Amarelo, Verde, Azul, Anil e Violeta.