Internet das Coisas

Aula 10 – Porta digital PWM como saída

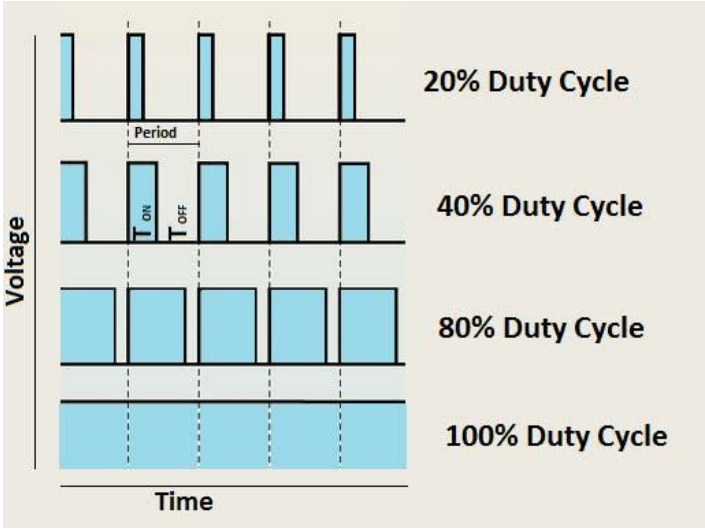
Prof. Rodrigo Barreto rodrigo.barreto@qi.edu.br / Fone/Whats: (51) 98412-5651

Portas digitais PWM

As portas PWM, assim como as portas digitais convencionais, também só conseguem trabalhar com dois estados e, consequentemente, com tensões absolutas de 0V ou de 5V. Porém, enquanto nas portas digitais convencionais o estado de saída permanece constante após sua escrita, nas portas PWM a tensão é gerada em forma de onda que possui uma determinada frequência, alternando entre os dois estados um determinado número de vezes por segundo. Alterar a frequência da alternância entre os dois estados, acaba criando uma tensão média na saída, mais próxima de 0V se o espaçamento for muito alto ou mais próxima de 5V se o espaçamento for muito curto, perto de uma onda linear.

Portas digitais PWM

Chamamos isso de Duty Cycle. Olhando a figura ao lado veja que quando o Duty Cycle é de 100%, o tempo em que a tensão permanece em 5V é de 100% e, portanto, a tensão de saída é muito próxima a 5V. Já quando o Duty Cycle está com 20%, somente em 20% do tempo a tensão está em 5V (estando nos outros 80% em 0V) e isso faz com que a tensão média da saída seja de 20% dos 5V (no caso, 1V).



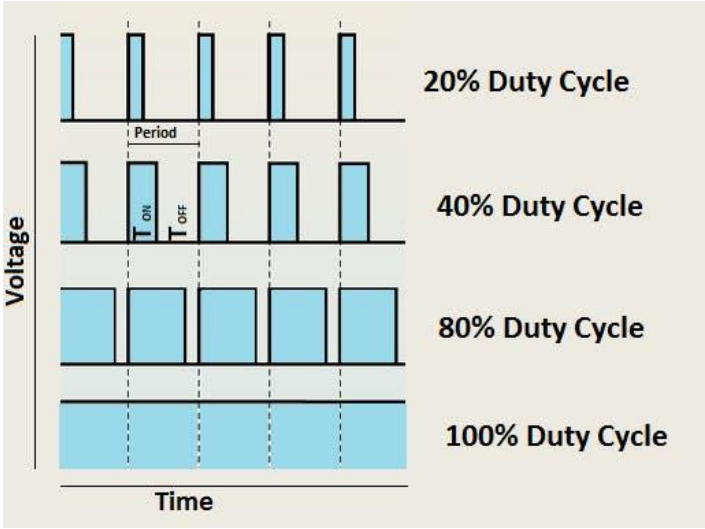
Portas digitais PWM

A mesma lógica vale para qualquer valor do Duty Cycle:

Duty Cycle em 40% = 5V . 40% = 2V

Duty Cycle em 75% = 5V . 75% = 3,75V

Duty Cycle em 90% = 5V . 90% = 4,5V



Portas digitais PWM

Esta é a forma como as portas digitais PWM, mesmo trabalhando apenas com tensão de saída de 0V ou de 5V, conseguem entregar nas suas saídas tensões médias distintas entre esses dois valores. Isso tem bastante utilizada quando necessitamos alimentar componentes com tensões menores do que 5V ou quando queremos variar a corrente através da variação da tensão para controlar sua intensidade, velocidade, etc. Um exemplo típico é o led RGB. Se as entradas dos 3 pinos de cor forem conectadas à portas PWM, conseguimos ajustar a mistura e a cor de saída.

Portas digitais PWM

O Arduino nos traz um método que permite esse ajuste do Duty Cycle, porém, diferente do convencional onde definimos uma porcentagem entre 0 e 100, pela característica binária das portas, o Arduino nos permite informar um valor entre 0 e 255 (sendo 0 o equivalente à 0% e o 255 sendo o equivalente à 100%). Inicialmente isso provocará uma certa estranheza e nos obrigará a fazer cálculos simples (regra de 3).

Portas digitais PWM

Para facilitar, podemos trabalhar com estes dois fatores de mapeamento abaixo:

- Cada 2,5 equivale a aproximadamente 1%
- Cada 50 equivale a aproximadamente 1V

Portas digitais PWM

Exemplo 1: necessitamos regular o Duty Cycle de uma porta digital PWM para garantir uma saída de 40% dos 5V. Qual o valor entre 0 e 255 devemos utilizar?

Resposta: 40 . 2,5 = 100

Exemplo 2: necessitamos regular o Duty Cycle de uma porta digital PWM de modo a nos garantir uma saída de 2,7V. Qual o valor entre 0 e 255 devemos utilizar?

Resposta: 2,7 . 50 = 135

Ajustando o Duty Cycle das portas PWM

Sabendo disso tudo, o ajuste do Duty Cycle (e consequentemente da tensão de saída) das portas PWM se torna simples através do uso do método abaixo:

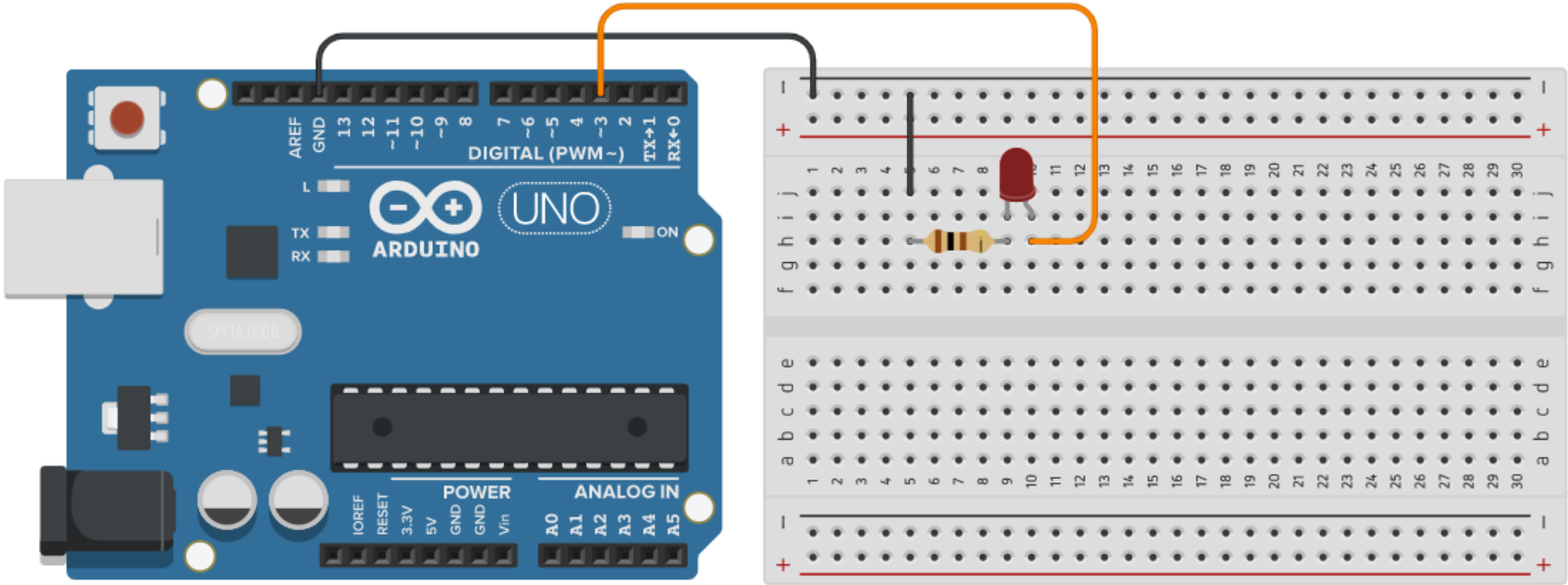
```
analogWrite( <porta>, <valor_pwm> );

<porta>
   número da porta que queremos definir o estado de saída

<valor_pwm>
   valor da modulação PWM entre 0 e 255 que irá gerar uma tensão
   média de saída entre 0V e 5V
```

Exemplo 1: ajustando luminosidade do led

Com estes conhecimentos, podemos realizar o controle de leds também, mas diferentemente do exemplo da aula passada, ao invés de somente ligar ou desligar o led, podemos regular a sua intensidade através do ajuste do valor do Duty Cycle com o uso do método analogWrite. Para este exemplo, vamos ligar o pino positivo de entrada do led à porta 3 (que é PWM) e o negativo a um resistor de 220Ω que leva ao negativo do Arduino (GND).

Exemplo 1: ajustando luminosidade do led

Assim, teremos um esquema eletrônico como o abaixo:



Exemplo 1: ajustando luminosidade do led

Agora na programação, primeiramente vamos dentro do setup definir a porta 3 como de saída através do pinMode e depois garantir que ela já inicia com 0V. Poderíamos fazer isso com um digitalWrite(3, LOW), mas como vamos trabalhar com o PWM, podemos também usar um analogWrite, passando como valor do Duty Cycle, 0.

```
void setup(){

 pinMode(3, OUTPUT);
 analogWrite(3, 0);

}
```

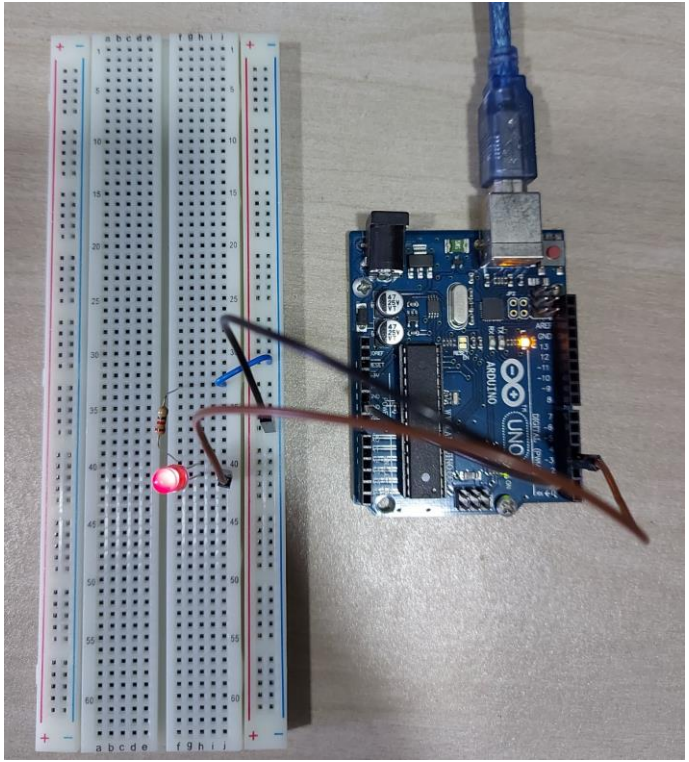
Exemplo 1: ajustando luminosidade do led

Após, dentro do loop, vamos iniciar o Duty Cycle com valor 0 e a cada meio segundo vamos aumentar 25 (ou seja, vamos incrementar cerca de 0,5V por vez). Assim, teremos:

```
void loop(){

  delay(500);
  analogWrite(3, 0);
  delay(500);
  analogWrite(3, 25);
  delay(500);
  analogWrite(3, 50);
  delay(500);
  analogWrite(3, 75);
  delay(500);
  analogWrite(3, 100);
  delay(500);
  analogWrite(3, 125);
  delay(500);
  analogWrite(3, 150);
  delay(500);
  analogWrite(3, 175);
  delay(500);
  analogWrite(3, 200);
  delay(500);
  analogWrite(3, 225);
  delay(500);
  analogWrite(3, 250);

}
```

Exemplo 1: ajustando luminosidade do led

Foto do circuito montado e funcional, com uma tensão de saída próxima a 0,8 V e com uma intensidade de luz baixa no led difuso vermelho.

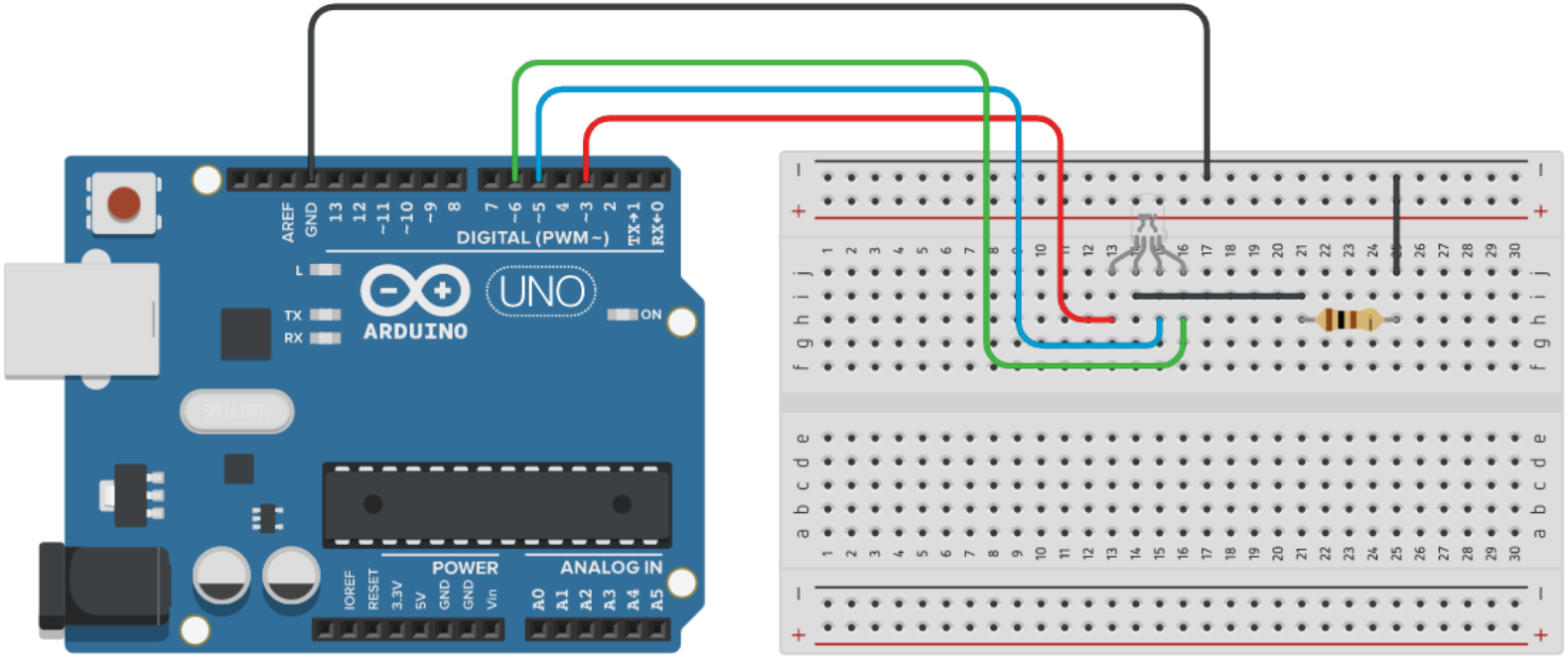


Exemplo 2: ajustando cor de led RGB

Dentro da mesma ideia, podemos usar as portas PWM para controlar as correntes dos 3 elementos de cor de um led RGB e, assim, as intensidades das 3 cores primárias, regulando a mistura e a cor de saída. Vamos então ligar o pino na cor vermelha na porta 3, o pino da cor azul na porta 5 e o pino da cor verde na porta 6. Perceba que a porta 4 foi “pulada”, pois ela não é uma porta PWM. Já o pino do negativo comum do led, deve ser ligado a um resistor (de 220Ω em nosso exemplo) conectado ao negativo, para baixar a corrente e preservar o led, como já vimos algumas vezes.

Exemplo 2: ajustando cor de led RGB

Assim, teremos um esquema eletrônico como o abaixo:



Exemplo 2: ajustando cor de led RGB

Agora na programação, primeiramente vamos dentro do setup definir as portas 3, 5 e 6 como de saída através do pinMode e depois garantir que elas já iniciem com 0V. Poderíamos fazer isso com um digitalWrite com estado LOW, mas como vamos trabalhar com o PWM, podemos também usar um analogWrite, passando como valor do Duty Cycle, 0.

```
void setup(){

 pinMode(3, OUTPUT);
 pinMode(5, OUTPUT);
 pinMode(6, OUTPUT);

 analogWrite(3, 0);
 analogWrite(5, 0);
 analogWrite(6, 0);

}
```

Exemplo 2: ajustando cor de led RGB

Agora, dentro do loop, a cada 1 segundo vamos ajustar os Duty Cycle nas 3 portas, com valores diferentes e, assim, obtemos cores RGB diferentes. Vamos lembrar que a porta 3 está ligada ao vermelho (R), a porta 6 está ligada ao verde (G) e a porta 5 está ligada ao azul (B). No exemplo ao lado, iremos construir as seguintes cores RGB:

```
void loop(){

  delay(1000);
  analogWrite(3, 140);
  analogWrite(6, 70);
  analogWrite(5, 160);

  delay(1000);
  analogWrite(3, 25);
  analogWrite(6, 90);
  analogWrite(5, 210);

  delay(1000);
  analogWrite(3, 100);
  analogWrite(6, 100);
  analogWrite(5, 30);

}
```

R: 140 G: 70 B: 160

R: 25 G: 90 B: 210

R: 100 G: 100 B: 30

Exemplo 2: ajustando cor de led RGB

Foto do circuito montado e funcional, emitindo uma mistura RGB com cor de saída em tonalidade azul.