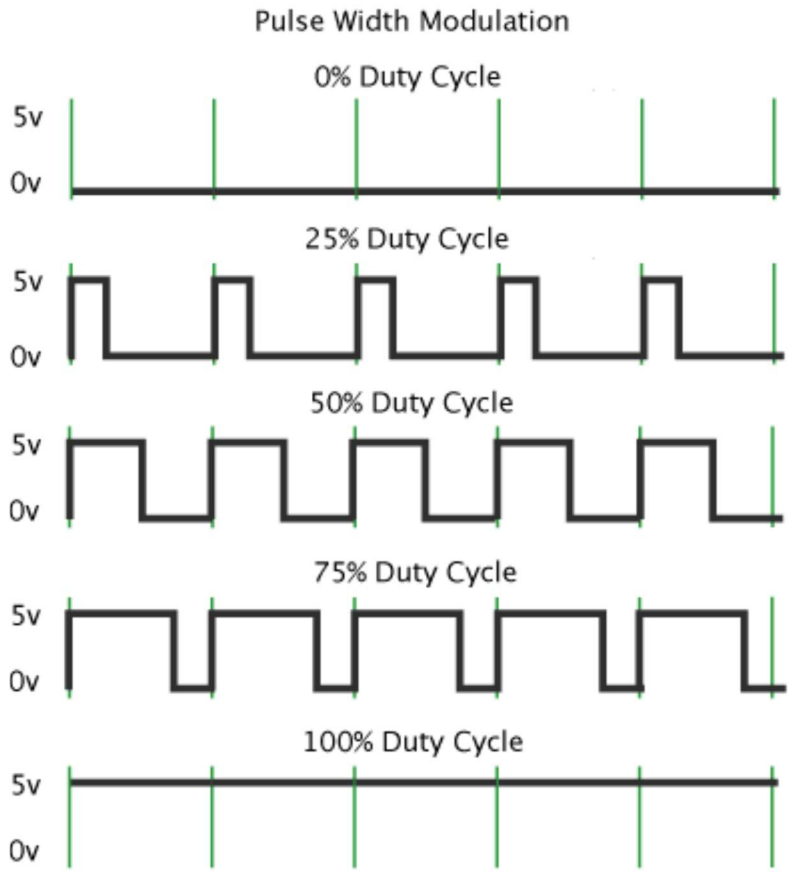
Experiência 8 – Led controlado por potenciômetro

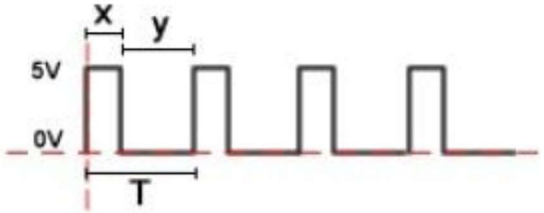
8.1. PWM

PWM (Pulse Width Modulation – Modulação por Largura de Pulso) é uma técnica para obter resultados analógicos por meios digitais. Essa técnica consiste na geração de uma onda quadrada em uma frequência muita alta em que pode ser controlada a porcentagem do tempo em que a onda permanece em nível lógico alto. Esse tempo é chamado de Duty Cycle e sua alteração provoca mudança no valor médio da onda, indo desde 0V (0% de Duty Cycle) a 5V (100% de Duty Cycle) no caso do Arduino.



Sinal PWM

O duty cycle é a razão do tempo em que o sinal permanece na tensão máxima (5V no Arduino) sobre o tempo total de oscilação, como está ilustrado na figura abaixo:



Duty cycle

Duty Cycle (%) = (x/x+y)*100% = (x/T)*100%

Vmédio = Vmax*Duty Cycle(%)

O valor do Duty Cycle usado pelo Arduino é um número inteiro armazenado em 8 bits, de forma que seu valor vai de 0 (0%) a 255 (100%).

8.1.1. Usando a saída PWM

Para escrever um sinal na saída PWM utiliza-se a função analogWrite, que recebe como parâmetros o pino PWM e o valor do duty cycle, respectivamente. Esse último parâmetro é guardado em 8 bits, de modo que esse valor deve estar entre 0 (0% de duty cycle) e 255 (100% de duty cycle).

```
//Escreve no pino 9 um sinal PWM com 50% de duty cycle (50% de 255=127)
analogWrite(9,127);

//Escreve no pino 10 um sinal PWM com 25% de duty cycle (25% de 255=64)
analogWrite(10,64);
```

Variando o duty cycle altera-se também o valor médio da onda, de modo que o efeito prático obtido com o PWM em algumas aplicações é um sinal com amplitude constante e de valor igual ao valor médio da onda. Isso permite que se possa, por exemplo, controlar a intensidade do brilho de um LED, ou a velocidade de um motor de corrente contínua.

8.2. Experiência – Led com controle de intensidade

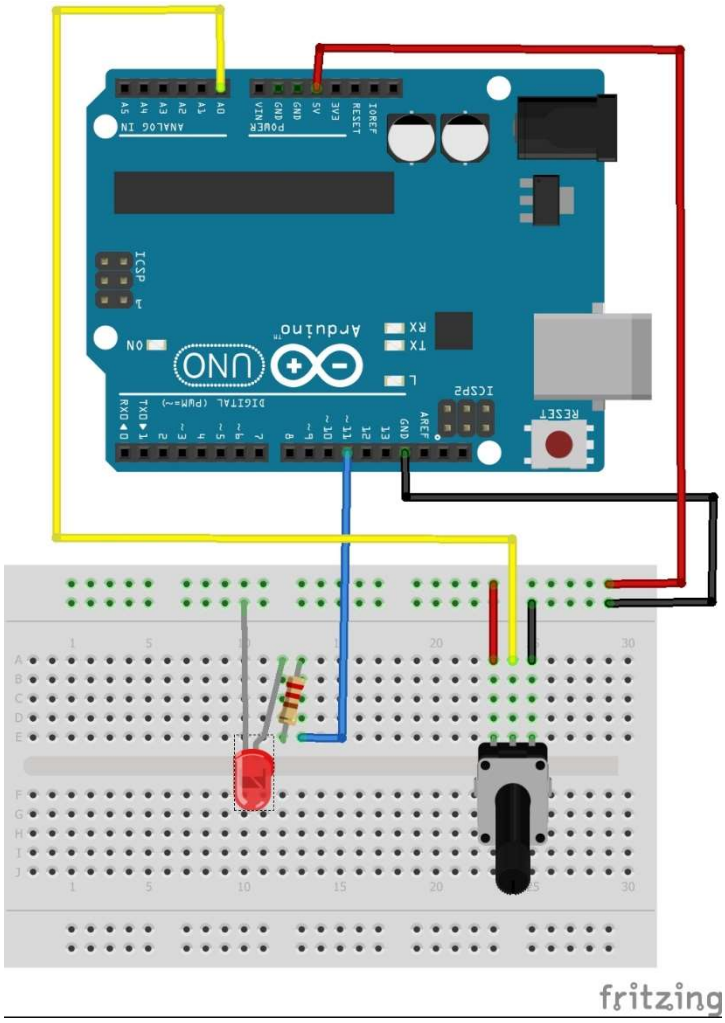
8.3. Ingredientes

- Potenciômetro
- LED 5mm
- Resistor 330Ω
- Fios Jumper’s
- Protoboard

8.4. Misturando os ingredientes

Agora vamos conectar os componentes do projeto. Para isso, monte seu circuito conforme a figura a seguir.

Garanta que seu Arduino esteja desligado durante a montagem e que o seu LED esteja conectado corretamente, com a perna mais longa (Anodo) conectado ao resistor e a perna menor (catodo) ao GND.



Circuito da Experiência 8

8.4. Levando ao forno

Conecte seu Arduino ao computador e abra a IDE Arduino. No menu Tools, certifique que a porta serial (serial port) está selecionada e se a placa configurada é a que você está usando (board).

8.5. Preparando a cobertura

Crie um programa (sketch) e salve com o nome de “programa_pwm”. Com o seu programa salvo, escreva nele o código conforme escrito abaixo.

```
// Daremos um nome ao pino que está conectado o LED
int ledPin = 11;
int pwmPin = A0;

unsigned int valorLido;
unsigned int pwm;

//Função "setup" roda uma vez quando a placa e ligada ou resetada
void setup() {
      pinMode(ledPin, OUTPUT);
      Serial.begin(9600);
}

// Função que se repete infinitamente quando a placa é ligada
void loop() {
      valorLido = analogRead(pwmPin); // valor entre 0 e 1023

      //mostra o valor lido na porta analógica
      Serial.println(valorLido);

      pwm = map(valorLido, 0, 1023, 0, 255); // Mudança de escala


      //Escreve no led um sinal PWM proporcional ao valor Lido
      analogWrite(ledPin,pwm);
}
```

Depois de escrever o código, Clique em Carregar ou -> para que o programa seja transferido para seu Arduino.

8.6. Experimentando o prato

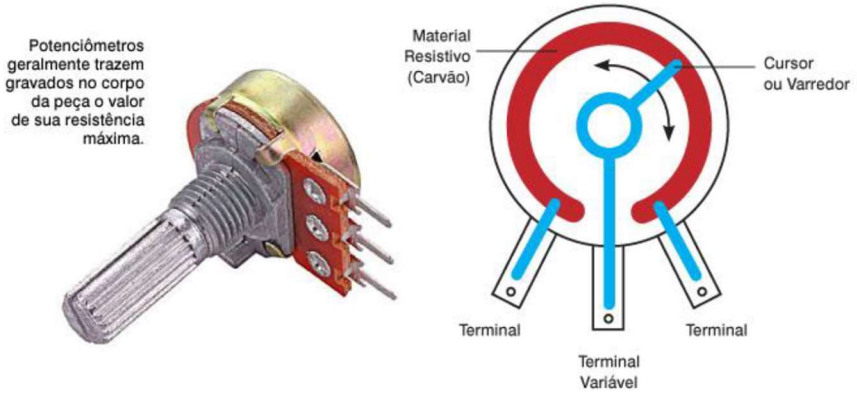
Se tudo der certo, conforme girarmos o potenciômetro, a intensidade de luz emitida pelo LED diminuirá ou aumentará.

8.7. Entendendo o Hardware

- *Potenciômetro e trimpot*

Potenciômetros e trimpot são resistores variáveis e ajustáveis, e por isso são usados para controle analógico de funcionalidades de alguns aparelhos eletrônicos, tal como o volume de um aparelho de som.

Eles costumam possuir três pernas. Dois são ligados às extremidades da resistência (terminal e terminal) e a terceira a um cursor que anda de ponta a ponta da resistência (terminal variável). Dessa forma, podemos usar o resistor entre os terminais ou terminal variável e terminal.



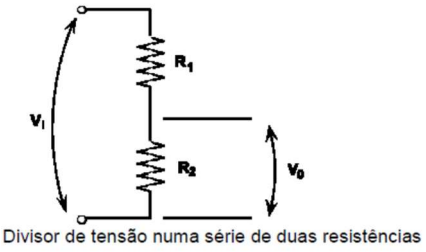
Potenciômetro

- *Divisor de tensão*

Quando temos n resistência associadas em série temos o nome de divisor de tensão. Em um circuito divisor de tensão, temos uma queda de tensão em cada resistência igual ao produto da resistência com a corrente do circuito.

Como a corrente do circuito é calculada pela divisão da tensão em cima de todos os resistores dividido pela soma dos resistores, teremos a tensão em cima de um resistor igual a resistência desse resistor vezes a tensão total dividida pela soma dos resistores.

O exemplo a seguir mostra como funciona o cálculo o para dois resistores.



$$V_0 = \frac{R_2}{R_1 + R_2} \times V_i$$

Divisor de tensão

Quando usamos um potenciômetro, podemos usar a propriedade do divisor de tensão. Sabemos que a tensão total e a resistência total são fixas. Dessa forma, o divisor de tensão vai variar com a resistência A0 e GND.

8.8. Entendendo o programa

Como já explicado no início do capítulo, o PWM pode ser usado para simular uma saída analógica. Variando um sinal de saída de PWM de 0 a 255, estamos variando o Duty Cycle, que por sua vez, resulta numa saída de 0V a 5V.

Como a intensidade de luz no LED está diretamente ligada à quantidade de corrente que passa por ele e essa corrente é proporcional a tensão do resistor em série com o LED, conforme variamos a tensão do pino 11 através do PWM, alteramos a intensidade de luz emitida pelo LED.

```
valorLido = analogRead(A0); // valor entre 0 e 1023
```

Para variarmos o PWM, usamos o valor analógico lido no potenciômetro e armazenamos na variável valorLido, que armazena valores entre 0 e 1023. Porém, para que seja possível usar essa variável para controlar o PWM, devemos mudar sua escala para 0 a 255. Para isso usaremos a função `map()`. Tal função recebe uma variável e muda sua escala, neste caso, de 0 a 1023 lida na entrada analógica para 0 a 255 nível digital.

```
pwm = map(valorLido, 0, 1023, 0, 255); // Mudança de escala
```

Depois de mudarmos de escala, basta escrevermos o valor de PWM na saída do LED.

```
analogWrite(led,pwm);//Escreve no led um sinal PWM proporcional ao valorLido
```

8.9. Comunicação Serial

A comunicação serial é amplamente utilizada para comunicar o Arduino com outros dispositivos como módulos ZigBee, Bluetooth, entre outros. A comunicação com o computador é possível através do conversor serial USB presente nas placas. A biblioteca padrão do Arduino possui algumas funcionalidades para a comunicação serial de modo a facilitar a utilização desta função.

- *Serial Monitor*

A possibilidade de visualizar na tela do computador os dados coletados e processados pelo Arduino é fundamental, visto que é uma forma rápida e prática de visualizar esses dados. Para isso, usamos o Serial Monitor.

O Serial Monitor disponibiliza uma interface gráfica que facilita a comunicação entre o Arduino e um computador. Após selecionarmos a porta serial adequada, o serial monitor pode ser aberto pelo ícone no canto superior direito da janela, onde é representado por uma lupa. No programa, iniciamos a comunicação serial programando a velocidade de comunicação usando a função:

```
Serial.begin(9600); //Inicia porta serial e define a velocidade de transmissão
```

- *Enviando Dados*

Na maioria das vezes, o envio de dados pela porta serial pode ser feito através das funções *print* e *println*. A função print imprime os dados na serial utilizando o código ASCII. Essa função recebe dois parâmetros e retorna a quantidade de bytes que foram escritos.

```
Serial.print(var); // Imprime o conteúdo da variável var
Serial.print("olá");// Imprime o texto olá
```

O parâmetro “var” contém o valor a ser impresso. A função é sobrecarregada de modo a ser capaz de receber qualquer tipo padrão (char, int, long, float etc).

Quando o texto estiver entre aspas, isso simboliza que você deseja imprimir um texto. Caso contrário, você deseja imprimir uma variável.

A função *println* imprime o valor desejado semelhantemente a função print, porém imprime na sequência os caracteres ‘\r’ e ‘\n’ de modo que o cursor vai para a linha seguinte (nova linha).

- *Recebendo dados*

Para o Arduíno receber dados do monitor serial utilizamos as funções de biblioteca *available* que verifica se tem caractere recebido e quantos são e *read* que lê um caractere por vez.

```
If(Serial.available()>0)  // verifica se tem caractere recebido
entrada=Serial.read();//le o caractere recebido e guarda na variavel “entrada”
```

Exercícios

1- Monte mais 1 led e modifique o programa para que os 2 leds sejam controlados pelo potenciômetro.

2 – Modifique o programa para que os leds funcionem invertidos. Com a variação do potenciômetro 1 led vai acendendo e outro led vai apagando.

3 – Monte mais 1 led. Conforme o potenciômetro vai variando, os leds vão acendendo progressivamente.

4 – Monte o buzzer e quando o último led acender apite o buzzer junto.

5 – Transforme a leitura do potenciômetro em uma função.

6 – Transforme a mudança de escala (cálculo) em uma função.

7 – Transforme o controle de cada dos led em uma função.

8 – Transforme o controle do buzzer em uma função.

9 – Transforme a escrita no led em uma função.