



Universidade do Estado de Minas Gerais
Fundação Educacional de Ituiutaba
Engenharia de Computação



CURSO BÁSICO ARDUINO Aula 3

João Ludovico Maximiano Barbosa

Rafael Caetano da Silva

2016

UTILIZANDO O PWM

Neste exemplo iremos utilizar a modulação de largura de pulsos (PWM) através da função “`analogWrite()`” para mudar a intensidade de brilho de um LED de acordo com o sinal PWM. O `analogWrite()`, liga e desliga um pino digital rapidamente, de forma a mudar o valor da tensão nesta porta de acordo com a velocidade de ligar e desligar em um determinado tempo.

Para montar o circuito conecte o ânodo (a perna mais longa, positivo) do seu LED ao pino de saída digital 9 no arduino através de um resistor de 220 ohm, conecte o cátodo (a perna mais curta, negativo) diretamente para ao terra (ground) do arduino conforme mostra a Figura 1.

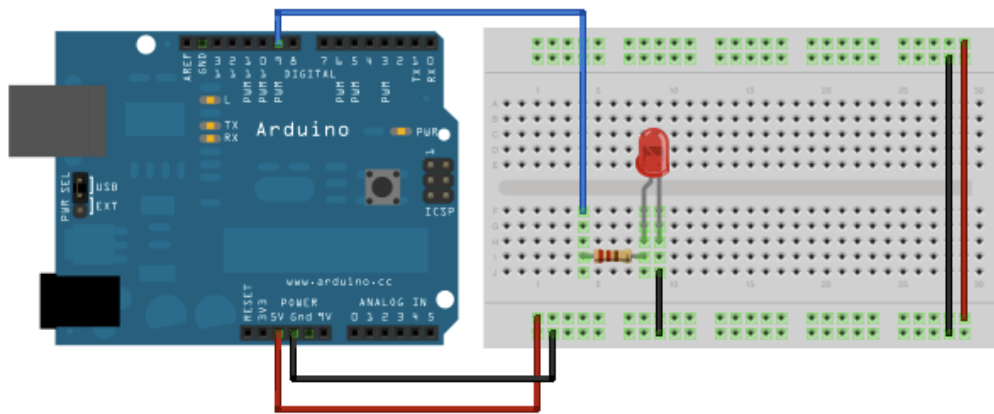


Figura 1 – Ligando um LED ao pino 9 PWM

No desenvolvimento do código iremos declarar o pino 9 como saída, no bloco SETUP através da variável `ledPin`.

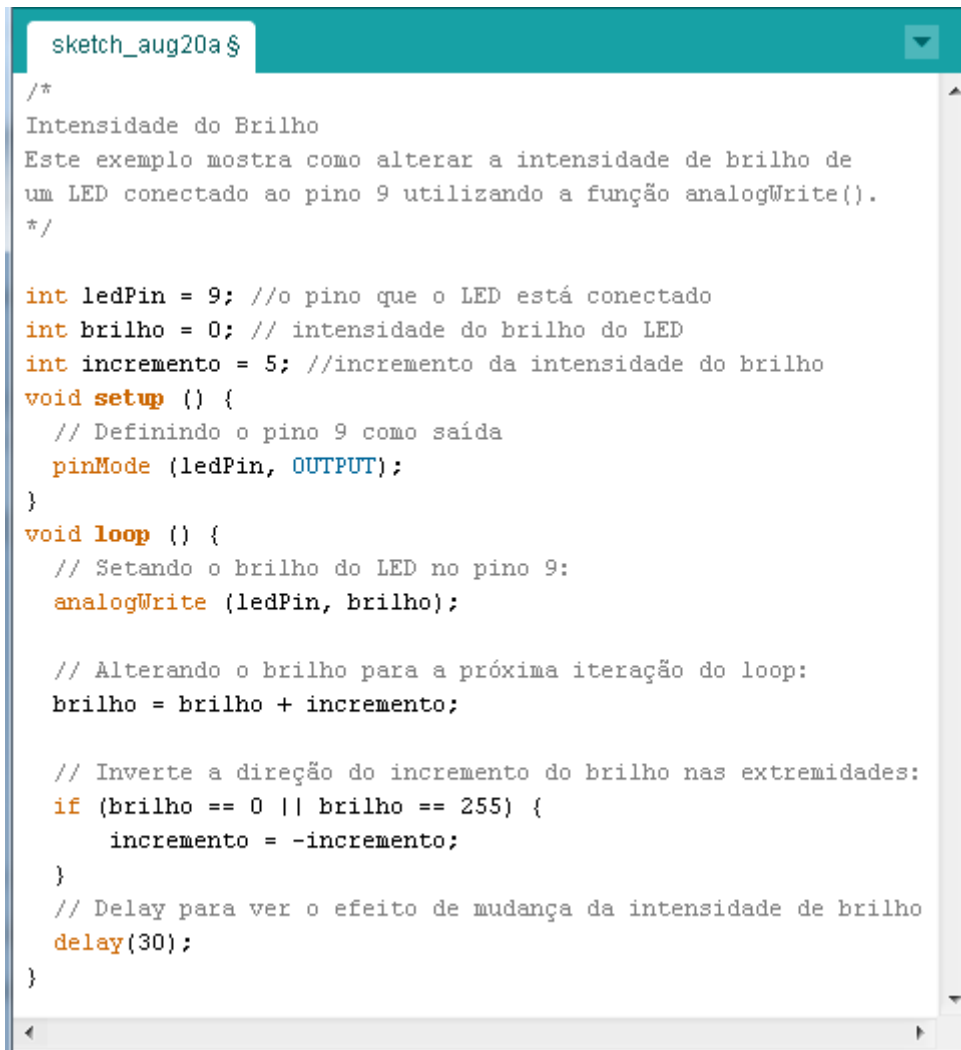
A função “`analogWrite()`” que faz o controle PWM exige que informemos dois parâmetros: o primeiro é o pino que será utilizado, e o segundo é o valor PWM que queremos colocar nesta porta.

Para mudarmos a intensidade do brilho do LED de ligado a desligado, iremos aumentar gradualmente o valor PWM de 0 (pulso todo desligado) a 255 (pulso todo ligado), e depois voltaremos a 0, mais uma vez para completar o ciclo.

No código mostrado na Figura 2, o valor de PWM é definido usando uma variável chamada `brilho`. Cada vez que passamos por um loop, o valor do PWM na variável `brilho` aumenta ou diminui de acordo com o valor de incremento.

Se a variável brilho chegar a um dos valores extremos (0 ou 255), invertemos o valor do incremento. Em outras palavras, se o incremento é 5, então ele passará a ser -5, e se o incremento for -5, então ele se tornará 5. Na próxima iteração do loop, esta mudança faz com que o brilho mude de direção também.

Na Figura 2 podemos ver o código utilizado para este exemplo.



```
sketch_aug20a$
/*
  Intensidade do Brilho
  Este exemplo mostra como alterar a intensidade de brilho de
  um LED conectado ao pino 9 utilizando a função analogWrite().
  */

int ledPin = 9; //o pino que o LED está conectado
int brilho = 0; // intensidade do brilho do LED
int incremento = 5; //incremento da intensidade do brilho
void setup () {
  // Definindo o pino 9 como saída
  pinMode (ledPin, OUTPUT);
}
void loop () {
  // Setando o brilho do LED no pino 9:
  analogWrite (ledPin, brilho);

  // Alterando o brilho para a próxima iteração do loop:
  brilho = brilho + incremento;

  // Inverte a direção do incremento do brilho nas extremidades:
  if (brilho == 0 || brilho == 255) {
    incremento = -incremento;
  }
  // Delay para ver o efeito de mudança da intensidade de brilho
  delay(30);
}
```

Figura 2 – Código exemplo de utilização do PWM

Atividade:

- Altere o valor de incremento e observe o que acontece;
- Altere também o valor do delay;

Utilizando a entrada analógica

Para este exemplo iremos utilizar um potenciômetro que é um botão que fornece uma resistência variável, que podemos ler no arduino como um valor analógico.

Na montagem do circuito iremos ligar o potenciômetro a uma das entradas analógicas do arduino (Figura 3) e imprimir o valor lido no serial monitor.

Os pinos do potenciômetro serão conectados da seguinte forma: um dos extremos será conectado ao terra (ground) e o outro pino do extremo será conectado aos 5V, e o pino do meio será conectado a entrada analógica 0 conforme mostra a Figura 3.

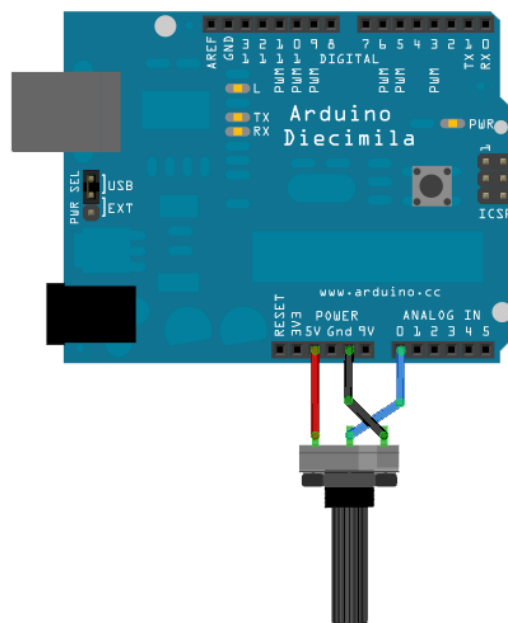


Figura 3 – Conectando o potenciômetro ao arduino

No início do código, iremos criar uma variável chamada “sensorPin” que receberá o valor “A0” que representa o numero da porta analógica zero, na qual o potenciômetro será ligado. Iremos criar também a variável “sensorValue” que armazenará o valor lido na porta analógica.

Neste exemplo iremos utilizar a função “analogRead()” que converte a tensão de entrada de 0 a 5 volts, para um valor digital entre 0 e 1023. Isto é feito por um circuito conversor analógico-digital que já vem embutido no arduino.

Ao girarmos o botão do potenciômetro, alteramos o valor da resistência em ambos os lados do pino central do potenciômetro. Com isso, as resistências relativas entre o pino central e os dois pinos externos alteram, fazendo com que uma tensão diferente chegue à

entrada analógica. Quando o botão é girado totalmente em uma direção, não há nenhuma resistência entre o eixo central e o pino conectado ao terra (ground), desta forma a tensão aplicada ao pino central é 0 volts e `analogRead()` retorna 0. Quando o botão é girado totalmente na outra direção, não há nenhuma resistência entre o eixo central e o pino conectado aos 5 volts, com isso a tensão aplicada ao pino central é de 5 volts e `analogRead()` retorna 1023. Se o botão for girado entre estas extremidades, a função `analogRead()` retorna um número entre 0 e 1023, que é proporcional à quantidade de tensão aplicada ao pino.

Na Figura 4 podemos ver o código utilizado para este exemplo.

The image shows a screenshot of an Arduino IDE window titled "sketch_aug20a\$". The code is written in C++ and is designed to read the value of an analog potentiometer connected to pin A0. It includes comments in Portuguese explaining the purpose of each line. The code defines two variables: `sensorPin` (A0) and `sensorValue` (0). It sets up the serial communication at 9600 baud and configures pin A0 as an input. In the `loop` function, it reads the analog value from A0 and prints it to the serial monitor.

```
int sensorPin = A0; // pino em que o potenciometro esta conectado
int sensorValue = 0; // variável para armazenar o valor lido na entrada
                        //analógica

void setup () {
  // Inicializando comunicação serial:
  Serial.begin(9600);
  // Definindo o sensorPin como entrada:
  pinMode (sensorPin, INPUT);
}

void loop () {
  // Lendo a entrada analógica:
  sensorValue = analogRead(sensorPin);
  //Imprimindo o valor lido no serial monitor
  Serial.print("Valor lido: ");
  Serial.println(sensorValue);
}
```

Figura 4 – Código de utilização da entrada analógica

Utilizando o Sensor Ultrassônico HC-SR04

O sensor ultrassônico HC-SR04 funciona como um detector de objetos, permitindo medir distancias mínimas de 2 centímetros até a distancias máximas de 5 metros com uma precisão de 3 milímetros. Estes sensores emitem um sinal ultrassônico que reflete em um objeto e retorna ao sensor (eco). O sinal de retorno é captado, permitindo-se deduzir a distância do objeto ao sensor tomando o tempo de trânsito do sinal (PINHEIRO, 2011).

A velocidade do sinal ultrassônico é de aproximadamente 340 m/s, desta forma se o sensor estiver a uma distância “**d**” do objeto, o sinal percorrerá uma distância equivalente a “**2d**” para sair e retornar ao sensor. Sabendo esses conceitos é possível calcularmos a distancia de um objeto pela formula (PINHEIRO, 2011):

$$velocidade = \frac{distancia}{tempo} \Rightarrow v = \frac{2d}{t} \Rightarrow d = \frac{v \cdot t}{2} \Rightarrow d = 170 \cdot t$$

O sensor HC-SR04 é composto por 4 pinos, sendo eles:

VCC: alimentação de 5V;

TRIG: pino de gatilho;

ECHO: pino de eco;

GND: terra.

Para a montagem do circuito, iremos ligar o pino GND do HC-SR04 ao GND (terra) do arduino e o VCC do sensor iremos conecta-lo ao pino de 5V do arduino. O pino ECHO (eco) do sensor será ligado ao pino 13 do arduino, e o pino TRIG (emissor) do sensor será conectado ao pino 12 do arduino (Figura 5).

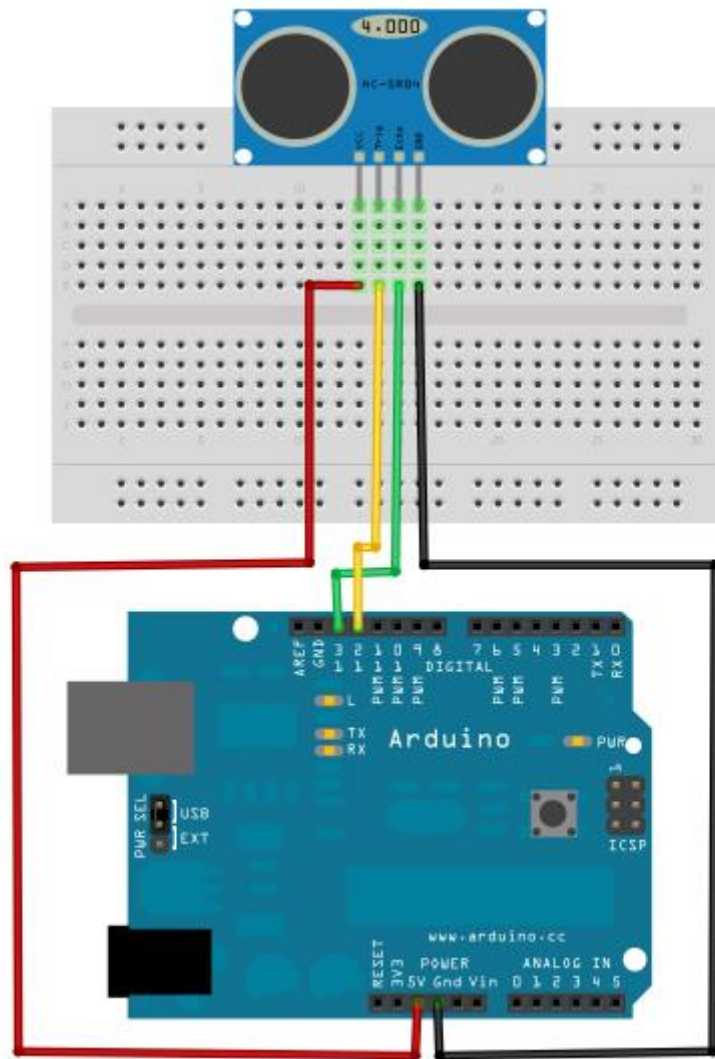


Figura 5 – Diagrama de ligação do sensor HC-SR04 com o arduino

Após a montagem do circuito iremos carregar o código da () na IDE do arduino. Para a escrita deste código foi utilizado a biblioteca “ultrasonic”, sendo assim, para que o seu código compile você deverá colocar esta biblioteca na pasta “libraries” da sua IDE.

```
sketch_aug27a $
#include "Ultrasonic.h"
#define echoPin 13 //Pino 13 recebe o pulso do echo
#define trigPin 12 //Pino 12 envia o pulso para gerar o echo
//iniciando a função e passando os pinos
Ultrasonic ultrasonic(12,13);

void setup()
{
  Serial.begin(9600); //inicia a porta serial
  pinMode(echoPin, INPUT); // define o pino 13 como entrada (recebe)
  pinMode(trigPin, OUTPUT); // define o pino 12 como saída (envia)
}

void loop()
{
  //seta o pino 12 com um pulso baixo "LOW" ou desligado ou ainda 0
  digitalWrite(trigPin, LOW);
  // delay de 2 microssegundos
  delayMicroseconds(2);
  //seta o pino 12 com pulso alto "HIGH" ou ligado ou ainda 1
  digitalWrite(trigPin, HIGH);
  //delay de 10 microssegundos
  delayMicroseconds(10);
  //seta o pino 12 com pulso baixo novamente
  digitalWrite(trigPin, LOW);
  // função Ranging, faz a conversão do tempo de
  //resposta do echo em centímetros, e armazena
  //na variável distancia
  int distancia = (ultrasonic.Ranging(CM));

  Serial.print("Distancia em CM: ");
  Serial.println(distancia);
  delay(1000); //espera 1 segundo para fazer a leitura novamente
}
```

Figura 6 – Código de teste para o sensor ultrassônico HC-SR04
Fonte: PINHEIRO, 2011

Após ter enviado o código para o arduino, abra o serial monitor da sua IDE para verificar se o seu sonar está mostrando a distância do objeto à frente dele.