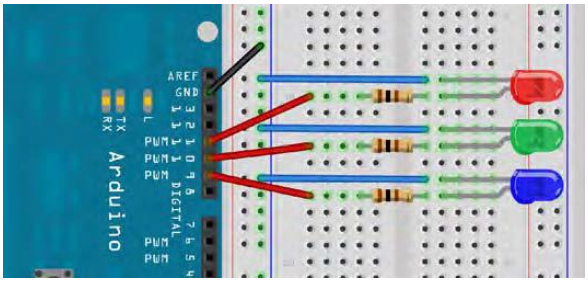
Projeto 8 – RGB Mood Lamp (Lampada do humor)
Prof. David M Soares

Vamos criar uma lampada (conjunto de tres leds, verde, vermelho e azul) que muda de cor aleatoriamente.
Lembre que com essas cores podemos fazer a cor que desejamos, basta variar as intensidades relativas. Os Pixels da tela do monitor de um PC colorido tem essas três cores. Entao o material necessário será: 3 Leds G, R, B e três resistores de 150 ou 100 Ohms.



Tres leds e três resistores de 100 Ohms

Codigo:

```
// Project 8 - Mood Lamp
float RGB1[3];
float RGB2[3];
float INC[3];
int red, green, blue;
int RedPin = 11;
int GreenPin = 10;
int BluePin = 9;
void setup() {
        randomSeed(analogRead(0));
        RGB1[0] = 0;
        RGB1[1] = 0;
        RGB1[2] = 0;
        RGB2[0] = random(256);
        RGB2[1] = random(256);
        RGB2[2] = random(256);
}
void loop()
        {
        randomSeed(analogRead(0));
        for (int x=0; x<3; x++) {
        INC[x] = (RGB1[x] - RGB2[x]) / 256; }
        for (int x=0; x<256; x++) {
                red = int(RGB1[0]);
                green = int(RGB1[1]);
                blue = int(RGB1[2]);
                analogWrite (RedPin, red);
                analogWrite (GreenPin, green);
                analogWrite (BluePin, blue);
                delay(100);
                RGB1[0] -= INC[0];
                RGB1[1] -= INC[1];
                RGB1[2] -= INC[2];
        }
        for (int x=0; x<3; x++) {
                RGB2[x] = random(556)-300;
                RGB2[x] = constrain(RGB2[x], 0, 255);
                delay(1000);
        }
}
```

Estude e carregue o programa.
Veja as cores mudando.
Na funçao setupescrevemos:

randomSeed(analogRead(0));
O comando randomSeed cria um numero pseudo aleatorio. O micro controlador procura um numero em uma tabela. Ajustando uma "semente" seed, dizemos ao micro aonde começar a contar. Neste caso começamos do valor lido em Analog Pin 0. Como nao ha nada em 0, lemos ruido analogico. Dessa maneira temos a semente. Usando a funçao **random()** geramos um numero aleatorio**.**

Temos então, 2 conjuntos de valores RGB armazenados em duas matrizes de três elementos.
RGB1 contem os valores iniciais.
RGB1[0] = 0;
RGB1[1] = 0;
RGB1[2] = 0;
O conjunto RGB2 contem os valores para os quais RGB transitam:
RGB2[0] = random(256);
RGB2[1] = random(256);
RGB2[2] = random(256);
Neste caso um numero aleatorio entre 0 e 255 é gerado.
A função **random (1000)** devolverá um número entre 0 e 999.
O numero 256 tem a ver com o PWM.
No programa principal você olha o valor inicial e o final e analisa qual incremento é necessário para progredir em 256 passos:

```
for (int x=0; x<3; x++) {
INC[x] = (RGB1[x] - RGB2[x]) / 256; }
```

Voce tem agora um outro loop:

```
for (int x=0; x<256; x++) {
red = int(RGB1[0]);
green = int(RGB1[1]);
blue = int(RGB1[2]);
analogWrite (RedPin, red);
analogWrite (GreenPin, green);
analogWrite (BluePin, blue);
delay(100);
RGB1[0] -= INC[0];
RGB1[1] -= INC[1];
RGB1[2] -= INC[2];
}
```

Isto ajusta o RGB1 e escreve nos pinos 9, 10, 11, diminue o increment respective, repetindo o processo 256 vezes. O delay(100) controla a velocidade do processo.
Apos isso, RGB1 é muito aproximadamente igual a RGB2.
A seguir montamos outro conjunto de RGB2:
Veja o Loop:

```
for (int x=0; x<3; x++) {
RGB2[x] = random(556)-300;
RGB2[x] = constrain(RGB2[x], 0, 255);
delay(1000);
}
```

Pega-se o numero aleatório entre 556 e zero e subtrai-se de 300. Com isso aumenta-se a possibilidade de geração de cores primarias (Porquê?).
Para nao ter números negativos enviados aos pinos PWM, Usamos a função **constrain()**.