01

Sorteando a cor

Transcrição

Vamos olhar com atenção a função `iniciaJogo()`. Ela está da seguinte maneira:

```
void iniciaJogo() {
    sequenciaLuzes[0] = LED_AZUL;
    sequenciaLuzes[1] = LED_VERDE;
    sequenciaLuzes[2] = LED_VERMELHO;
    sequenciaLuzes[3] = LED_AMARELO;
}
```

Com esse formato nós sempre teremos a mesma sequência de LEDs acendendo e esse não é o objetivo do jogo! Para resolver esse problema vamos utilizar uma função que gere um número randômico. Podemos acrescentar `int randomico = random(1, 10)`. Atenção, quando escrevemos dessa maneira temos um intervalo de números de 1 até 9, o número 1 é incluído, mas o 10 é o limite. Caso quiséssemos um intervalo de dez números, deveríamos escrever `(1, 11)`. Com o auxílio do monitor serial podemos verificar se isso funciona. Teremos o seguinte código:

```
void iniciaJogo() {
    int randomico = random(1,11);//1-10
    Serial.println(randomico);
    sequenciaLuzes[0] = LED_AZUL;
    sequenciaLuzes[1] = LED_VERDE;
    sequenciaLuzes[2] = LED_VERMELHO;
    sequenciaLuzes[3] = LED_AMARELO;
}
```

Na função `loop` nós podemos retirar o `Serial.println(checaRespostaJogador);` escrevendo duas barras na frente, `//`, ou seja, apenas inserindo um comentário. Agora, podemos compilar isso e enviar. Abrindo o Monitor verificamos que ele gera o número 8. Vamos apagar o que tínhamos escrito:

```
int randomico = random(1,11);//1-10
    Serial.println(randomico);
```

Acrescentamos que a primeira posição deve ser um número randômico que vai variar das posições do LED verde até o azul. Portanto, escrevemos que `sequenciaLuzes[0] = random(LED_VERDE, LED_AZUL + 1)`. Observe que adicionamos o `+1`, pois dessa maneira, temos certeza de que o LED azul estará incluído no intervalo. E repetimos isso para os demais LEDS:

```
void iniciaJogo() {

    sequenciaLuzes[0] = random(LED_VERDE, LED_AZUL + 1);//2-5 (6)
    sequenciaLuzes[1] = random(LED_VERDE, LED_AZUL + 1)
    sequenciaLuzes[2] = random(LED_VERDE, LED_AZUL + 1)
    sequenciaLuzes[3] = random(LED_VERDE, LED_AZUL + 1)
}
```

Mas, atenção! A todo momento estivemos em busca de um código mais compacto e organizado possível, portanto, vamos criar uma função que sorteie a cor! Ela será a `int sorteiaCor()`. Vamos adicionar um `return random (LED_VERDE, LED_AZUL +1)` e utilizamos essa função junto ao `iniciaJogo()`. Podemos realizar uma construção usando o `for`! Teremos `for(int indice = 0; indice < TAMANHO_SEQUENCIA; indice ++)`. O comentário `//2-5 (6)` nós deslocaremos para junto do `sorteiaCor`. Utilizando isso o código fica ainda mais compacto! Observe:

```
sequenciaLuzes[indice] = sorteiaCor();
```

Teremos o seguinte

```
void iniciaJogo() {
    for(int indice = 0; indice < TAMANHO_SEQUENCIA; indice ++) {
    sequenciaLuzes[indice] = sorteiaCor();
    }
}

int sorteiaCor() {
    return random(LED_VERDE, LED_AZUL + 1); // 2-5 (6)
}
```

Agora, podemos sortear o jogo! Para saber qual jogo está saindo, vamos tirar o comentário `/*` do `void loop`. Assim, poderemos verificar a sequência sendo gerada randomicamente!

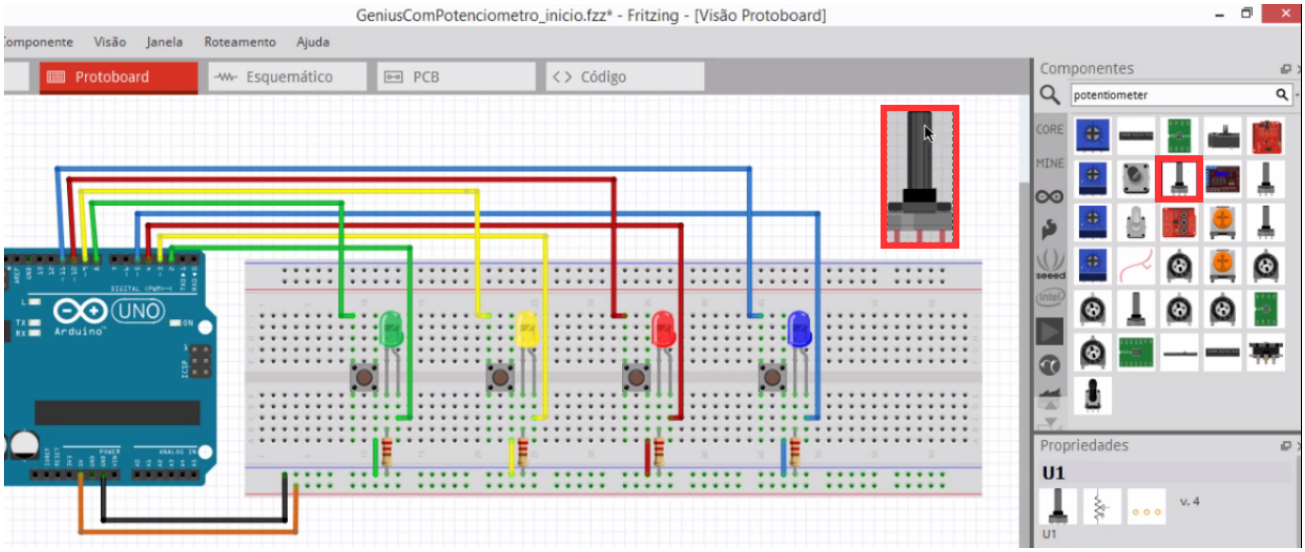
Nós compilamos isso e enviamos para o **Arduino** que irá iniciar uma sequência gerada randomicamente! Mas, ele mostra uma sequência que se repetirá, mesmo que nós apertemos o "Reset" a sequência será a mesma. Ou seja, a função `random()` não está funcionando.

Vamos entender o que aconteceu! Um computador só sabe fazer funções matemáticas através de circuito e ele usa isso para fazer praticamente tudo, por exemplo, gerar vídeos, áudio, receber e-mails, etc. Dessa maneira, quando falamos para o computador que ele deve sortear algo e que isso deve ser aleatório é preciso que nós alimentemos essa função com algo que seja igualmente aleatório.

Normalmente, o sistema operatório possui um relógio que é utilizado para alimentar a função que gera um número aleatório. O relógio é representado por números que são convertidos em uma determinada hora e quando o jogo começa nós não sabemos que número é esse.

O grande problema, entretanto, é que nós não temos um relógio no **Arduino**. E isso nos causa problemas, portanto, teremos que ser criativos para lidar com a situação. E se pudéssemos controlar o número que estamos passando para função? E toda vez que mudássemos esse número, nós trocássemos o jogo? Uma maneira de fazer isso é utilizar um número que seja lógico. Como podemos fazer isso?

Certamente, não podemos fazer isso utilizando uma porta lógica, pois ela aceita apenas 2 valores, ou 0 ou 1. Dessa forma, vamos utilizar uma porta analógica que pega o sinal contínuo do ambiente. Por exemplo, poderíamos colocar um sensor de temperatura, pressão e esse número contínuo seria pego e convertido em um número inteiro. Assim, podemos criar projetos utilizando esses artefatos! O *potenciômetro* é muito interessante, pois é uma peça simples e fácil de ser utilizada. Observe o potenciômetro no **Fritzing**:



O potenciômetro possui uma resistência variável o que significa que somos nós que dizemos qual é a sua resistência. Assim, ao rodá-lo nós podemos passar instruções para que fique com mais ou menos resistência. Isso é legal, pois, o sinal resultante da resistência pode gerar nosso número.

Para fazer isso é preciso, primeiro, aprender como o potenciômetro funciona. Existe a força, na linha de baixo e o terra na linha de cima. O que faremos é ligar os dois sinais nas portas externas do potenciômetro. O primeiro fio que ligaremos será o terra, em preto, e o outro será o força, que ficará na cor laranja. A porta que sobra é a que fornece o sinal resultante, significa que o potenciômetro vai receber nessa entrada 5v e através dessa regulagem conseguiremos um sinal menor se aumentarmos a resistência ou maior se a diminuirmos. Então, vamos alimentar esse sinal no **Arduino**. Vamos ligá-lo, por exemplo, a porta 0.

Resumindo, vamos ligar uma porta exterior na força (laranja) e a outra porta exterior no terra (preto) e o do meio (cinza) será o sinal resultante que nós conectamos a porta analógica, ou seja, estamos ligando na porta 0. Observe: