02

Lidando com sequência de luzes

Transcrição

E se quiséssemos uma sequência diferente? Teríamos que mudar a ordem de chamada da função `piscaLed()`, não é mesmo?! Antes, vamos tentar uma abordagem um pouco diferente! Vamos começar criando uma nova função: `piscaSequencia1()`. Dentro dela nós só copiaremos o conteúdo que tínhamos no `loop()`.

```
void loop(){
    piscaSequencia1();
}


void piscaSequencia1(){
    piscaLed(LED_VERDE);
    piscaLed(LED_AMARELO);
    piscaLed(LED_VERMELHO);
    piscaLed(LED_AZUL);
    delay(MEIO_SEGUNDO);
}
```

Já conseguimos desacoplar, ou seja, modularizar nosso código, uma vez que é preciso apenas chamar a nova função para que todas as luzes pisquem numa sequência. Mas e se quiséssemos acender todas as luzes ao mesmo tempo? Para isso, vamos dar uma arrumada na casa e criar uma nova função, que realize essa tarefa para nós:

```
void loop(){
    piscaSequencia1();
    piscaSequencia2();
}


void piscaSequencia2(){
    digitalWrite(LED_VERDE, HIGH);
    digitalWrite(LED_AMARELO, HIGH);
    digitalWrite(LED_VERMELHO, HIGH);
    digitalWrite(LED_AZUL, HIGH);

    delay(UM_SEGUNDO);


    digitalWrite(LED_VERDE, LOW);
    digitalWrite(LED_AMARELO, LOW);
    digitalWrite(LED_VERMELHO, LOW);
    digitalWrite(LED_AZUL, LOW);

    delay(MEIO_SEGUNDO);
}
```

Nessa nova função todas as luzes se acendem e, então, aguardamos por 1 segundo e apagamos todas elas. Teste no seu Arduino!

Se aproximando mais do jogo

Agora temos um conhecimento maior sobre como manipular os LEDs, tanto na parte de *eletrônica* quanto na parte da *programação*. Uma lição importante é que sempre podemos melhorar um pouco o nosso código! Por exemplo, não faz parte do jogo as luzes piscarem assim que o **Arduino** é ligado. Além disso, na `setup()` existe muita configuração de porta. Por que não tentamos arrumar isso?

```
void setup(){
    Serial.begin(9600);
    iniciaPortas();
}


void iniciaPortas(){
    pinMode(LED_VERDE, OUTPUT);
    pinMode(LED_AMARELO, OUTPUT);
    pinMode(LED_VERMELHO, OUTPUT);
    pinMode(LED_AZUL, OUTPUT);
}


void loop(){
    //retirado o código de testes anteriores
}
```

Nosso array de LEDs

A função `iniciaPortas()` faz a configuração inicial para nós. Para que fiquemos mais próximos do jogo, precisamos perceber a nossa necessidade, ou seja, **gerar** uma sequência de luzes a qual o jogador terá que imitar. Para fazer tal feito em *C++* é preciso criar um `array` ou `vetor`, ou seja, uma variável que guarde o valor de **diversos** LEDs e não apenas de 1.

```
//nosso array de LEDs
int sequenciaLuzes[] = [LED_AZUL, LED_VERDE, LED_VERMELHO, LED_AMARELO];


void loop(){
    piscaLed(sequenciaLuzes[0]);
}
```

Perceba que após a variável foi adicionado o `sequenciaLuzes`, dois colchetes e um número dentro. Esta é a forma que indicamos exatamente **qual** LED queremos que pisque. A contagem sempre começa pelo **0**, então, no caso, mandamos piscar o `LED_AZUL`. Caso quiséssemos o `LED_VERMELHO`, indicaríamos entre os colchetes o número **2**.

Mas, e se quiséssemos mudar a sequência? E se quiséssemos que a sequência fosse iniciada somente quando o jogador clicasse em algum botão? Para fazer isso é preciso isolar essa parte da programação. E como já aprendemos, *isolar = criar uma função*. Observe:

```
//definimos o tamanho da nossa sequência
#define TAMANHO_SEQUENCIA 4
```

```
int sequenciaLuzes[TAMANHO_SEQUENCIA];


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

Veja que definimos também o `tamanho` da nossa sequência. Fizemos isso, pois no *C++*, não podemos declarar um `array` sem definir o seu tamanho. A única exceção, inicializarmos a variável logo no seu momento de criação, que nem estávamos fazendo antes. Porém, como queremos mudar seu conteúdo, e inicializar com valores diferentes mais pra frente no jogo, precisaremos indicar o seu tamanho.

Para fazermos essa configuração inicial de sequência, basta colocarmos no `setup()`.

```
void setup(){
    Serial.begin(9600);
    iniciaPortas();
    iniciaJogo();
}
```

Percorrendo o array

Agora que temos uma sequência de LEDs, podemos muito bem mandar piscar cada um deles nessa sequência. Mas não iremos fazer isso do jeito que já vimos. Dessa vez, usaremos um laço de repetição:

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

O que fizemos? Dentro do nosso laço `for`, criamos a variável `indice`, que começará com o valor 0. Após cada interação, ela aumenta o seu valor em 1*(indice++)*. Dessa maneira, nós percorremos toda a nossa sequência de luzes, sem ter que ficar escrevendo diversas linhas de código. O melhor, é que se aumentarmos o tamanho da nossa sequência de LEDs, podemos continuar com o mesmo código.