01

A programação de nossos botões

Transcrição

Agora, está na hora de retornar ao código e passar o trecho referente ao botão para o código original do nosso jogo. No caso, perceba que temos a função `iniciaPortas()`. Por que não usar a mesma função para iniciar os botões? Primeiramente, declaramos nossos botões:

```
#define BOTAO_VERDE 8
#define BOTAO_AMARELO 9
#define BOTAO_VERMELHO 10
#define BOTAO_AZUL 11
```

E, assim, usamos o `iniciaPortas()`:

```
void iniciaPortas(){
    pinMode(LED_VERDE, OUTPUT);
    //outros LEDs


    pinMode(BOTAO_VERDE, INPUT_PULLUP);
    pinMode(BOTAO_AMARELO, INPUT_PULLUP);
    pinMode(BOTAO_VERMELHO, INPUT_PULLUP);
    pinMode(BOTAO_AZUL, INPUT_PULLUP);
}
```

Compilando o código e enviando para o **Arduino** vemos que não temos nada de errado nele e, inclusive, nossa sequência definida há algum tempo continua sendo repetida pelos LEDs.

Lendo os botões

Já sabemos como ler um botão. Porém, como revisão, podemos adicionar dentro do nosso `loop()` o seguinte:

```
int estado = digitalRead(BOTAO_VERDE);
Serial.println(estado);
```

E ao observar no monitor a resposta verificamos que recebemos o `1`, mas quando o botão é pressionado, o sinal passa a ser `0`. Ou seja, tudo funciona perfeitamente! Vamos apenas melhorar o que já temos. Primeiramente, vamos comentar o laço `for` que temos dentro do `loop()`:

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

```
    int estado = digitalRead(BOTAO_VERDE);
    Serial.println(estado);


}
```

Mas, nós não queremos conhecer apenas o estado de **um** botão, queremos saber o de todos. Para isso, podemos criar uma função `checaRespostaJogador()`. Essa função irá nos devolver qual foi o botão apertado pelo usuário. Vamos começar a criar:

```
int checaRespostaJogador(){
    if(digitalRead(BOTAO_VERDE) == LOW){
        return LED_VERDE;
    }
    if(digitalRead(BOTAO_AMARELO) == LOW){
        return LED_AMARELO;
    }
    if(digitalRead(BOTAO_VERMELHO) == LOW){
        return LED_VERMELHO;
    }
    if(digitalRead(BOTAO_AZUL) == LOW){
        return LED_AZUL;
    }
}
```

Essa função nós leva até o `if` se o botão for apertado. Caso ele tenha sido pressionado a função retornará o LED correspondente. Mas e se e ninguém apertou o botão? Podemos retornar `INDEFINIDO` e, dessa maneira, o código fica expressivo e podemos verificar que ninguém clicou no botão, ou seja, o estado não está definido. Observe:

```
    //resto do código
    if(digitalRead(BOTAO_AZUL) == LOW){
        return LED_AZUL;
    }


    return INDEFINIDO;
}
```

Ainda, precisamos *definir* a nossa constante. Ou seja, no início do programa é preciso atribuir um valor que não pode ser igual a algo comumente utilizado para mapear portas, por exemplo. Logo, não é uma boa ideia colocar `0`, pois a atribuição de portas começa em `0`. Tão pouco é aconselhável utilizar um número como `20`, pois caso movêssemos o código para outro tipo de **Arduino** que tivesse `20` portas, nosso programa deixaria de funcionar. Bem, existe uma solução muito boa, que é `-1`. Esse é um número inteiro e nunca teremos uma porta mapeada como `-1`. Portanto:

```
#define INDEFINIDO -1
```

E dentro do `loop()` podemos apagar o código auxiliar e deixar apenas o comentário do laço. Depois disso, adicionamos uma saída ao nosso `Serial`, chamando a função. Dessa maneira, a tela ficará *"printando"* `-1` até apertarmos algum botão. Observe como fica o código:

```
loop(){
    //laço comentado


    Serial.println(checaRespostaJogador());
}
```

Uma última alteração

Para finalizar esse capítulo, podemos aprender a **piscar o LED correspondente**. E existe uma maneira muito fácil de arrumar isso! É um alívio perceber que o código está bem arrumado, pois isso torna as tarefas muito mais fáceis!

A função `piscaLed()` começará a devolver um *inteiro* que é o valor correspondente do LED. E na função `checaRespostaJogador()` deve ser retornada a função `piscaLed()` . Veja como fica o código:

```
int piscaLed(int portaLed){
    digitalWrite(portaLed, HIGH);
    delay(UM_SEGUNDO);
    digitalWrite(portaLed, LOW);
    delay(MEIO_SEGUNDO);


    return portaLed;
}


int checaRespostaJogador(){
    if(digitalRead(BOTAO_VERDE) == LOW){
        return piscaLed(LED_VERDE);
    }
    if(digitalRead(BOTAO_AMARELO) == LOW){
        return piscaLed(LED_AMARELO);
    }
    if(digitalRead(BOTAO_VERMELHO) == LOW){
        return piscaLed(LED_VERMELHO);
    }
    if(digitalRead(BOTAO_AZUL) == LOW){
        return piscaLed(LED_AZUL);
    }


    return INDEFINIDO;
}
```

Compilando e enviando poderemos verificar que isso funciona! Nossos LEDs acendem conforme clicamos nos botões!