

03

## Processando a resposta

### Transcrição

Até o momento vimos a estrutura de controle funcionando, mas ainda não processamos a resposta do usuário! Por enquanto estivemos incrementando a variável!

O processamento da resposta ocorre quando o usuário pressiona um botão, portanto, é preciso ler o que foi clicado como certo ou errado. Na verdade, já existe uma função que faz isso o `checaRespostaJogador()` e o gabarito do jogo encontra-se na `sequenciaLuzes()`. Em `processaRespostaUsuario` nós podemos adicionar o `int resposta = checaRespostaJogador` e perguntar se o valor contestado é correto, `if(resposta == sequenciaLuzes())`. Ainda, podemos acrescentar `leds_respondidos` que servirá como índice dessa pergunta. Dessa maneira, ocorrerá uma comparação estabelecida pelo operador lógico `==`, o valor do item e o próprio elemento. Ainda, podemos adicionar um `else` para o caso do jogador ter contestado errado. Por fim, adicionamos o monitor serial! Teremos:

```
void processaRespostaUsuario() {
    int resposta = checaRespostaJogador();

    if(resposta == sequenciaLuzes[leds_respondidos]) {
        leds_respondidos++;
    }else{
        Serial.println("resposta errada");
    }
}
```

Compilando isso e observando a `protoboard` e também o `console` não obtemos respostas satisfatórias sobre o que está acontecendo. Então, podemos adicionar ao `processaRespostaUsuario` o `Serial.println(resposta)` para melhor visualizar o que está acontecendo:

```
void processaRespostaJogador() {
    int resposta = checaRespostaUsuario();

    Serial.println(resposta);

    if(resposta == sequenciaLuzes[leds_respondidos]) {
    }else{
        Serial.println("resposta errada");
    }
}
```

Novamente compilamos e enviamos para observar o Console e a `protoboard`. Com isso teremos:

The screenshot shows the Arduino IDE's Serial Monitor window. The title bar says "COM3". The main area of the window displays the following text repeatedly:

```

usuário respondendo
-1
resposta errada
usuário respondendo
-1

```

Repare que a todo momento o resultado assinalado é `-1`. O motivo disso é porque o jogador ainda não pressionou nada! Ou seja, é preciso tratar esse momento em que o jogador ainda não escolheu nenhum botão. Assim, podemos interromper a execução e dessa maneira nós acrescentamos `if(resposta == INDEFINIDO)` e como essa função não retorna nada, apenas adicionamos `return`. Teremos o seguinte no código:

```

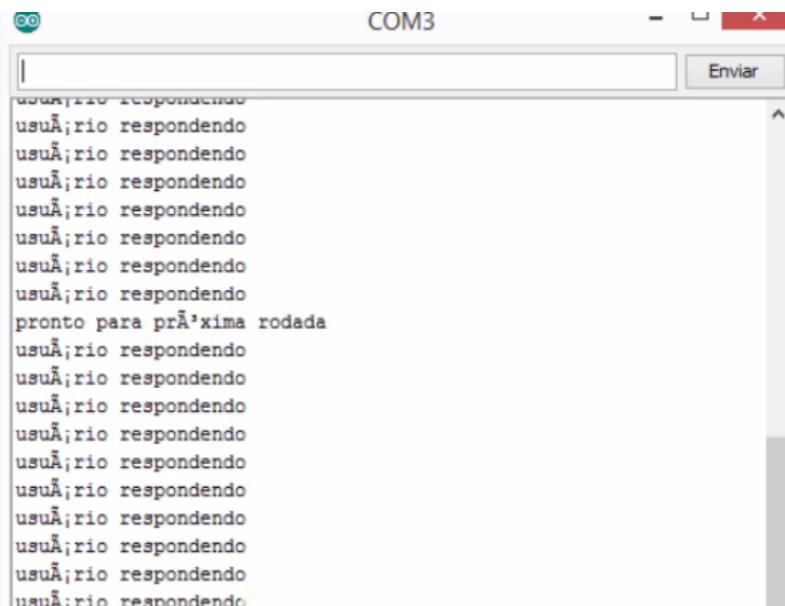
void processaRespostaUsuario() {
    int resposta = checaRespostaJogador();

    if(resposta == INDEFINIDO){
        return;
    }

    if(resposta == sequenciaLuzes[leds_respondidos]) {
        leds_respondidos++;
    }else{
        Serial.println("resposta errada");
    }
}

```

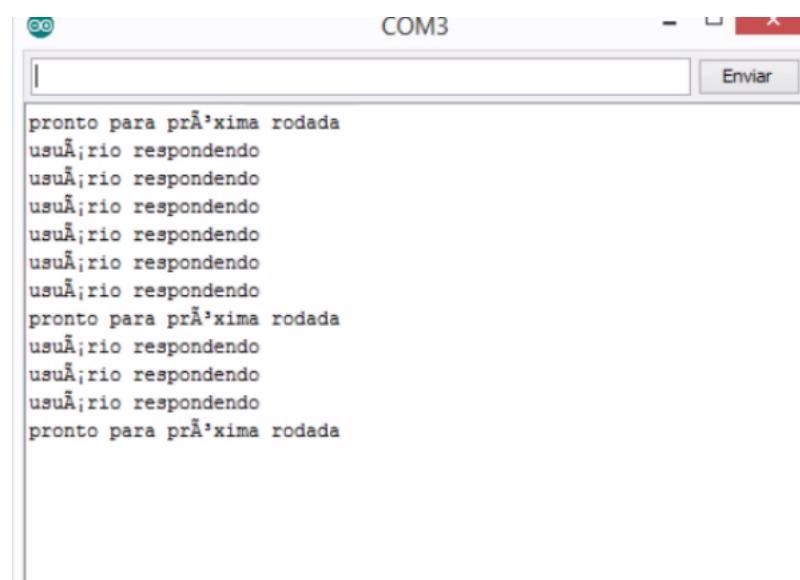
Compilando isso e mexendo no **Arduino** podemos verificar, pelo console, que enquanto aguardamos a resposta do jogador é mostrado `usuário respondendo`. Podemos fazer um teste e contestar errado a um LED, nesse caso, veremos que ele avisa que a resposta é errada! Assim como, ao pressionar certo ele avisa que passamos para a próxima rodada. Observe:



A partir da observação no console percebemos que ocorre um equívoco, pois em determinado ponto não é mostrado mais nada! Portanto, será preciso investigar o porquê dessa situação ocorrer! Por qual motivo não conseguimos que ele toque uma única vez! Se observarmos descobrimos que ele nunca toca quando a rodada é exatamente igual a sequência. E isso é fácil de corrigir, o `preparaNovaRodada` deve ser menor ou igual que `TAMANHO_SEQUENCIA`. Assim, teremos o seguinte, alterando o caracter para `<=` :

```
void preparaNovaRodada(){
    rodada++;
    leds_respondidos = 0;
    if(rodada <= TAMANHO_SEQUENCIA){
        tocaLedsRodada();
    }
}
```

Vamos compilar e testar para verificar se funcionou! A medida que formos respondendo aos LEDS podemos observar o que acontece no Console:



Entretanto, o feedback ainda não está satisfatório. Quando o jogo for finalizado com sucesso, seria interessante se ele dissesse de alguma maneira que terminou! E se utilizarmos uma sequência que pisca os LEDs de uma única vez para sinalizar que o jogo finalizou com sucesso?

Vamos em `void loop()` e acrescentamos `jogoFinalizadoSucesso()` e teremos o seguinte:

```
void loop() {
    switch(estadoAtual()) {
        case PRONTO_PARA_PROX_RODADA;
            Serial.println("pronto para próxima rodada");
            preparaNovaRodada();
            break;
        case USUARIO_RESPONDENDO;
            Serial.println("usuário respondendo");
            processaRespostaUsuario();
            break;
        case JOGO_FINALIZADO_SUCESSO;
            Serial.println("jogo finalizado sucesso");
            jogoFinalizadoSucesso();
            break;
        case JOGO_FINALIZADO_FALHA;
            Serial.println("jogo finalizado falha");
            break;
    }
    delay(500);
}
```

E nós renomeamos a função `void piscaSequencia1()`. Nós a chamaremos de `jogoFinalizadoSucesso()`. Teremos o seguinte:

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

Agora temos um feedback introduzido, mas vamos aproveitar para adicionar uma sequência que só será realizada quando o usuário errar. Assim, apagamos o `placaSequencia2` e trocamos por `jogoFinalizadoFalha`. Teremos:

```
void jogoFinalizadoFalha() {
    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);
}
```

E adicionamos o `jogoFinalizadoFalha` junto ao `void loop()`:

```

void loop() {
    switch(estadoAtual()) {
        case PRONTO_PARA_PROX_RODADA;
            Serial.println("pronto para próxima rodada");
            preparaNovaRodada();
            break;
        case USUARIO_RESPONDENDO;
            Serial.println("usuário respondendo");
            processaRespostaUsuario();
            break;
        case JOGO_FINALIZADO_SUCESSO;
            Serial.println("jogo finalizado sucesso");
            jogoFinalizadoSucesso();
            break;
        case JOGO_FINALIZADO_FALHA;
            Serial.println("jogo finalizado falha");
            jogoFinalizadoFalha();
            break;
    }
    delay(500);
}

```

Agora os *feedbacks* estão prontos. Mas, ainda tem uma questão a ser resolvida! Vamos observar o `estadoAtual`! Como sabemos que estamos no estado do jogo com falha ou sucesso? Nós adicionamos ao `else` o `if` e `rodada == TAMANHO_SEQUENCIA + 1`. Ainda, acrescentamos outro `else` e colocamos `return JOGO_FINALIZADO_FALHA`. Teremos:

```

int estadoAtual() {
    if(rodada <= TAMANHO_SEQUENCIA)
    }
    if(leds_respondidos == rodada){
        return PRONTO_PARA_PROX_RODADA;
    }else{
        return USUARIO_RESPONDENDO;
    }
}else if(rodada == TAMANHO_SEQUENCIA + 1){
    return JOGO_FINALIZADO_SUCESSO;
}else{
    return JOGO_FINALIZADO_FALHA;
}
}

```

Mas, em nenhum momento conseguimos cair nesse caso! Estamos interagindo com um jogo de memória onde a regra do jogo é que se o usuário respondeu errado, temos que parar o jogo e avisar que ele errou. No `processaRespostaUsuario` falamos que a contestação é errada! Assim, acrescentamos `rodada = TAMANHO_SEQUENCIA + 2`. Observe o que acontece:

```

void processaRespostaUsuario() {
    int resposta = checaRespostaJogador();

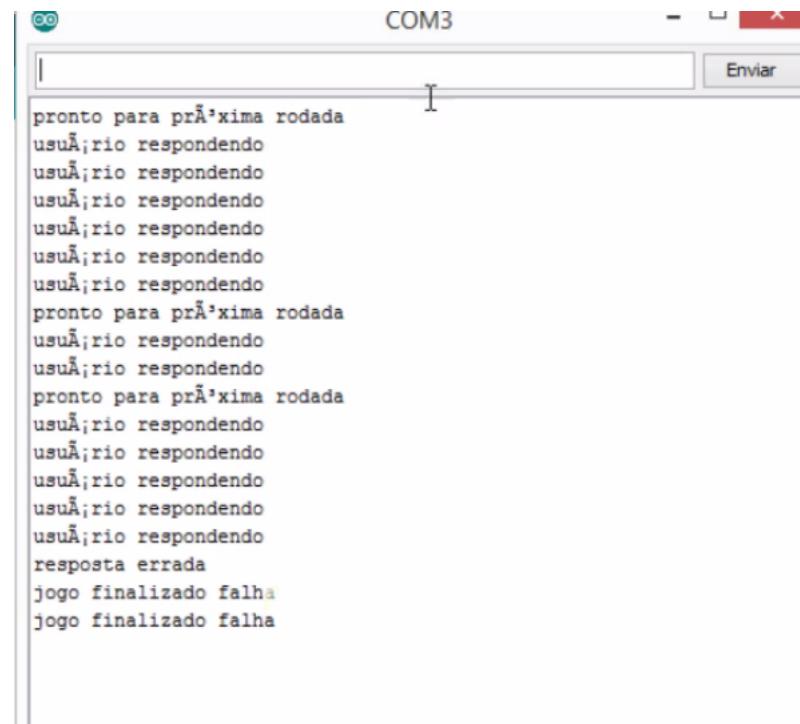
    if(resposta == INDEFINIDO){
        return;
    }

    if(resposta == sequenciaLuzes[leds_respondidos]) {

```

```
    leds_respondidos++;
}else{
    Serial.println("resposta errada");
    rodada = TAMANHO_SEQUENCIA + 2;
}
}
```

Vamos observar o console! Vamos simular um jogo, nele acertamos algumas perguntas e depois erramos uma. Perceba que no caso do erro recebemos a resposta `jogo finalizado falha`:



```
pronto para proxima rodada
usuario respondendo
usuario respondendo
usuario respondendo
usuario respondendo
usuario respondendo
usuario respondendo
pronto para proxima rodada
usuario respondendo
usuario respondendo
pronto para proxima rodada
usuario respondendo
usuario respondendo
usuario respondendo
usuario respondendo
resposta errada
jogo finalizado falha
jogo finalizado falha
```

Agora já temos todas as informações necessárias para criar o jogo **Genius!!!**