

## Preparando a resposta do usuário

### Transcrição

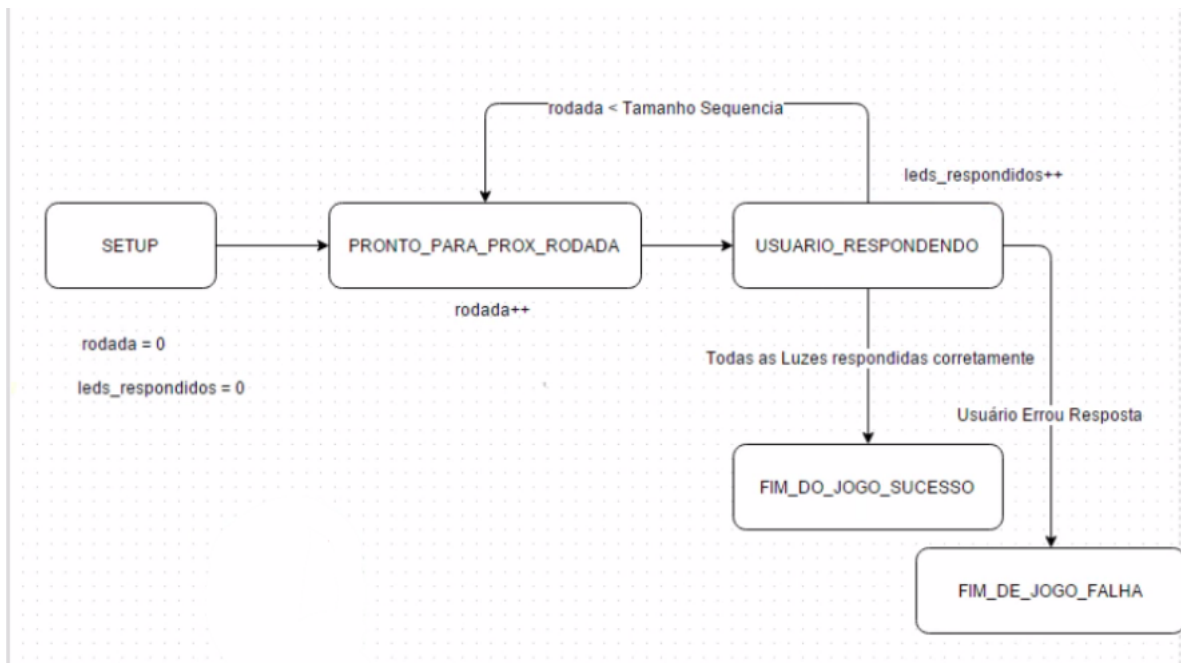
A medida que as rodadas vão passando, mais luzes vão acendendo até que depois da última rodada o jogo é finalizado e o jogador acaba sem nada para fazer. Isso ocorre pois não marcamos a transição entre `PRONTO_PARA_PROX_RODADA` para `USUARIO_RESPONDENDO`. E, assim como foi necessário criar uma variável para guardar a rodada, também será preciso criar uma que guarde o que o usuário respondeu.

Mas, antes, vamos pensar nisso utilizando o diagrama! E, embaixo do `SET UP`, adicionamos um comentário:

`leds_respondidos = 0`. Quando fazemos o `SET UP` do jogo estamos na verdade inicializando a variável, assim, toda vez que o usuário estiver respondendo vamos poder incrementar isso e fazer algo semelhante ao que temos no

`PRONTO_PARA_PROX_RODADA`. Vamos dizer que quando o estado for `USUARIO_RESPONDENDO`, então, teremos `leds_respondidos++`.

Teremos o seguinte:



Dessa maneira, no `SETUP` os `leds_respondidos` são iniciados, no `PRONTO_PARA_PROX_RODADA` a rodada é incrementada e no `USUARIO_RESPONDENDO` ela é incrementada a medida que os LEDs forem respondidos. Isso significa que na volta para o `PRONTO_PARA_PROX_RODADA` temos que pensar um pouco. Se sempre só ficarmos incrementando os LEDs isso significa que nesse trecho do `USUARIO_RESPONDENDO` para o `PRONTO_PARA_PROX_RODADA` sempre ficaremos fazendo uma conta.

Facilita nossa vida saber que quando uma nova rodada é iniciada a variável é zerada, assim, quando os LEDs forem respondidos a variável `leds_respondidos` será igual a `rodada`. Assim, se a variável `leds_respondido` for igual a zero, isso significa que nada terá sido respondido! Portanto, para encerrar essa etapa é preciso responder a um LED. Ou seja, acrescentamos a condição `leds respondidos == rodada` para transitar de `USUARIO_RESPONDENDO` para o `PRONTO_PARA_PROX_RODADA`. Apenas quando essa condição for verdadeira é que se passará para a próxima rodada.

**Resumindo:** Apenas quando as condições `rodada < Tamanho Sequencia` e `leds_respondidos == rodada` forem verdadeiras é que teremos uma próxima rodada!

Lembrando que é sempre bom pensar um pouco sobre o funcionamento do projeto antes de começar a executá-lo!

Após passar por toda essa reflexão já temos como controlar quantas repostas o usuário deu e o seu relacionamento com as rodadas. Vamos voltar ao código, nele implementaremos o `int leds_respondidos= 0` abaixo do `int rodada = 0;`:

```
int rodada = 0;
int leds_respondidos= 0
```

E no `preparaNovaRodada()` nós zeramos isso acrescentando um `leds_respondidos = 0`. Teremos:

```
void preparaNovaRodada() {
    rodada++;
    leds_respondidos = 0;
    tocaLedRodada();
}
```

No `loop()` nós colocaremos o processamento da resposta, escrevemos junto ao `USUARIO_RESPONDENDO` o `processaRespostaUsuario`. Teremos:

```
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");
            break;
        case JOGO_FINALIZADO_FALHA;
            Serial.println("jogo finalizado falha");
            break;
    }
    delay(500);
}
```

Isso requer que nós tenhamos uma função `processaRespostaUsuario()` que nós criaremos logo abaixo do `preparaNovaRodada()`. Junto a isso colocaremos o `leds_respondidos++`:

```
void processaRespostaUsuario(){
    leds_respondidos++
}
```

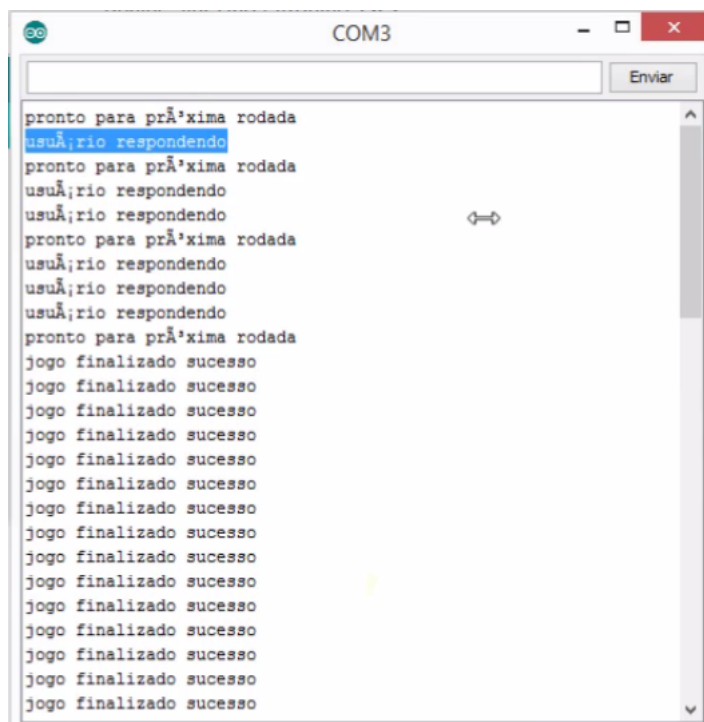
Ainda falta dizer para o programa ir ao estado `USUARIO_RESPONDENDO`. Assim, temos que dizer para `estadoAtual()` qual é a condição necessária para isso acontecer. Nós sabemos, observando o `estadoAtual`, que quando a rodada for menor que o tamanho da sequência nós estamos passando direto. Então, temos que saber quando está tudo pronto para uma próxima rodada. A criação da variável `if` é justamente para essa situação, assim, `if(leds_respondidos == rodada)`. Com isso, nós estamos falando que a partir dessa condição, nós estamos prontos para uma próxima rodada, caso contrário, o usuário ainda está respondendo, para transmitir essa mensagem nós escrevemos `else` e `return USUARIO_RESPONDENDO`:

```

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

```

Observe como foi interessante termos feito o planejamento antes! Caso contrário teria sido tortuoso buscar respostas através da prática! Podemos compilar isso e enviar! Observe o que temos:



Repare que era para ter ocorrido quatro respostas, entretanto, obtivemos apenas três! Observe que o usuário não participa da próxima rodada! Na primeira rodada ele responde uma vez, na segunda, duas vezes, na terceira, três vezes, mas na quarta rodada ele não responde e é direcionado para JOGO\_FINALIZADO\_SUCESSO .

Vamos reparar no código! Nós estamos barrando o código quando dizemos que no estadoAtual() a condição é if(rodada < TAMANHO\_SEQUENCIA) . Para resolver essa questão é preciso dar mais uma chance, ou seja, vamos colocar como quesito <= . Ficaremos com:

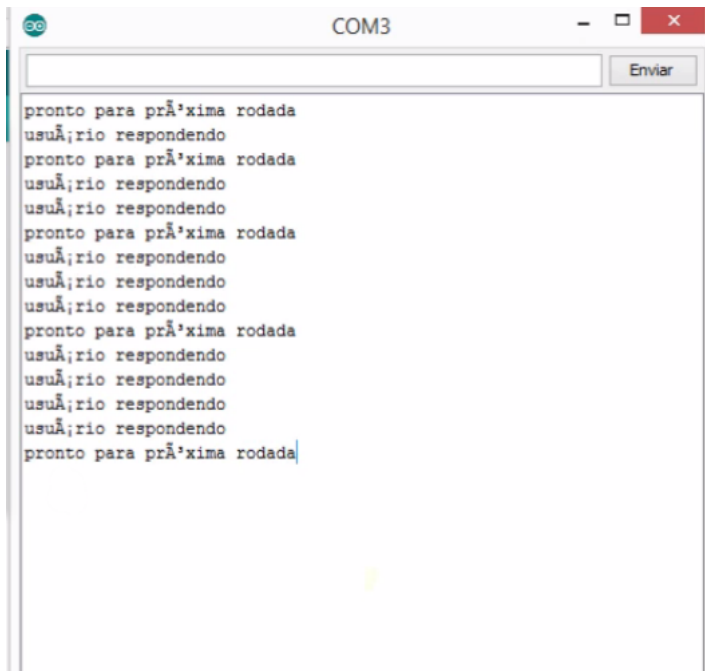
```

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

```

```
}  
}
```

Podemos mais uma vez compilar isso e enviar! Vamos observar o *console* para ver se todos os passos que desejamos são seguidos. Teremos:

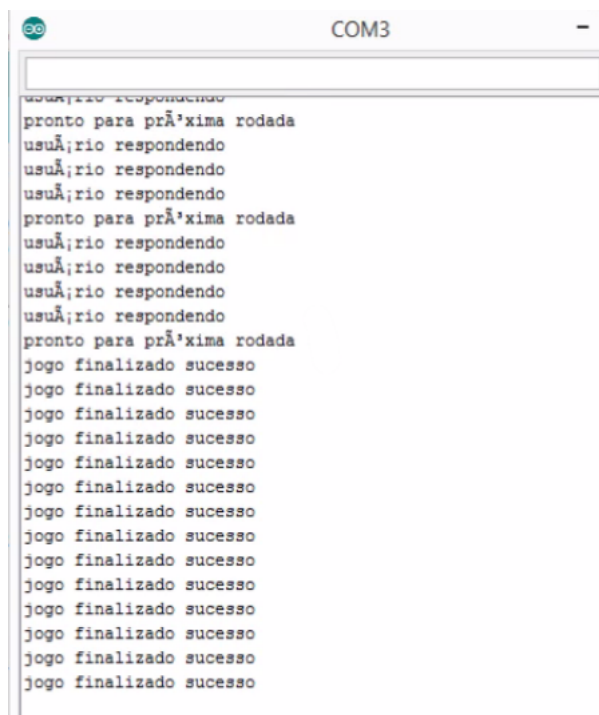


Nós ainda não conseguimos chegar ao resultado desejado, ou seja, alcançar o jogo finalizado. Isso provavelmente está relacionado a tentar tocar mais luzes do que cabem na sequência. Vamos observar o `tocaLedsRodada`. O índice que tínhamos descrito é aumentado em mais 1! Antes ele parava em 4 e agora ele para em 5 e, portanto, a sequência de luzes recebe um número maior que a quantidade de luzes que estão dentro da `array` e isso causa um erro!

Então, não podemos tocar a todo momento a sequência, o que podemos fazer para resolver esse problema é... no `preparaNovaRodada` acrescentar `if(rodada < TAMANHO_SEQUENCIA)`, assim, se o tamanho for menor que o da sequência, o LED da rodada é tocado. Teremos:

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

Dessa forma, os `Leds` serão tocados apenas se necessário. Podemos mais uma vez compilar. Observe o *console*:



```
COM3
usuário respondendo
pronto para próxima rodada
usuário respondendo
usuário respondendo
usuário respondendo
pronto para próxima rodada
usuário respondendo
usuário respondendo
usuário respondendo
usuário respondendo
pronto para próxima rodada
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
jogo finalizado sucesso
```

Finalmente teremos o que queríamos!

