2º Relatório

Comecei o código agrupando todas as variáveis por grupos (Fig.1), em primeiro lugar estão as variáveis necessárias para o debouncing e de seguida as variáveis necessárias para o jogo em si. Se por alguma razão o utilizador quiser variar o tempo de jogo deve fazer alterando o valor da última variável.

```
//------------------------------Debounce-----------
int lastbuttonState_AND= HIGH;
int lastbuttonState_OR= HIGH;
int lastbuttonState_XOR= HIGH;
unsigned long debounceDelay = 15;
unsigned long lastDebounceTime_AND = 0;
unsigned long lastDebounceTime_OR= 0;
unsigned long lastDebounceTime_XOR = 0;
int buttonState_AND,buttonState_OR,buttonState_XOR;
//---------------------------------Jogo-------------
int numero_random;
int base,target,lido;
int primeiro_bit,segundo_bit;
int o;
String a;
unsigned long time;
unsigned long tempo_de_jogo=60000;
```

Figura 1-Variáveis

No setup usei 2 fors (Fig.2) para definir os leds como output e os botões como pullup, de seguida chamei uma função não necessária a este projeto, mas o que eu acredito ser uma boa adição e uma maneira de facilitar a compreensão do jogo ao utilizador.

```
void setup()
{
  Serial.begin(9600);
  for (int i=8; i<=11; i++){
    pinMode(i, OUTPUT);}
  for (int i=2; i<=4; i++){
    pinMode(i, INPUT_PULLUP);}
  instrucoes();
  randomSeed(analogRead(0));
}
```

Figura 2-Setup

Depois do setup temos a função loop (Fig.3) que contem o jogo. Começo por atribuir 2 valores random a 2 variáveis (base e target) e por chamar uma função que define e informa o jogador sobre as operações/botões disponíveis (Fig.4).

```
void loop(){
  //Geração de numeros aleatórios
  target=gera_num();
  base=gera_num();
  //Operações permitidas
  operacoes();
  //Instruções/Objetivos para o jogo
```

Figura 3-Geração de um número aleatório

```
void operacoes(){
    o=random(1,4);
    primeiro_bit=bitRead(o,0);
    segundo_bit=bitRead(o,1);
  Serial.print("Operacoes permitidas:");
    if (primeiro_bit==1){
      Serial.print("XOR-VERMELHO, ");
    }
    if (segundo_bit==1){
      Serial.print("AND-AMARELO, ");
    }
    Serial.println("OR-BRANCO");
}
```

Figura 4-Função responsável pelas operações

De seguida é imprimido no monitor todas as informações relativas as variáveis necessárias para o jogo (Fig.5) e é criado também uma condição de tempo (Fig.6) com a ajuda do millis() que faz com que o código fique "preso" nessa condição tornando possível varias rondas do jogo. Quando esta condição (Fig.6) deixar de ser verdadeira significa que o tempo de jogo foi excedido .

```
int segundos = tempo_de_jogo/1000;
Serial.print("Valor Target: ");
Serial.println(target,BIN);
//Serial.println(target,BIN);
Serial.println("Valor Base:"+String(base, BIN));
Serial.println("Tem " + String(segundos) + " segundos para advinhar o numero'
Serial.println("Introduza um valor:");
```

Figura 5-Variáveis necessárias ao jogo

```
time = millis();
while((millis() - time) < tempo_de_jogo){
```

Figura 6-Condição de tempo

Seguidamente é chamada uma função responsável por ligar as luzes (Fig.7) ao início do jogo e por desligar 1 luz a cada ¼ do tempo total do jogo exceto a última luz led que se desliga quando faltar 1 segundo para o jogo acabar. A função ler_serial() (Fig.8) é responsável por ler o valor inserido por o jogador no monitor, sendo depois esse valor mudado do tipo string para o tipo inteiro e atribuído a uma nova variável.

```
void luzes(){
  if ((millis()-time)==100){
    for (int i=8; i<=11; i++){
      digitalWrite(i, 1);}
  }
  if((millis() - time) == (tempo_de_jogo- 1000)){
    digitalWrite(8,0);
  }
  for(int i=1;i<=3;(i++)){
    if((millis() - time) == (tempo_de_jogo *(i*0.25))){
    digitalWrite((12-i),0);
    }
  }
}
```

Figura 7-Função responsável pelas luzes

```
void ler_serial(){
  if (Serial.available()>0){
    a= Serial.readStringUntil('\n');
    lido=a.toInt();
    Serial.println("Valor lido:"+String(lido,BIN));
    Serial.println("Prima um operador");
  }
}
```

Figura 8-Função que lê o valor inserido

Posteriormente podemos observar o chamamento das funções dos botões XOR e AND (Fig.8) que estão dependentes dos 2 primeiros bits de um número aleatório de 1 a 3, quanto ao botão OR este é independente do número gerado (esta sempre disponível).

```
if (primeiro_bit==1){
  botao_XOR();
}
if(segundo_bit==1){
  botao_AND();
}
botao_OR();
```

Figura 8-Chamamento das funções dos botões

Por fim temos 2 condições de vitoria/derrota, se o objetivo do jogo for alcançado o jogador é parabenizado e após 2,5 segundos é recomeçado o jogo (Fig.9). O único caso de derrota possível é se o jogador não tiver alcançado o objetivo dentro do tempo imposto (Fig.10). O comum a estas 2 condições é que no final é chamada

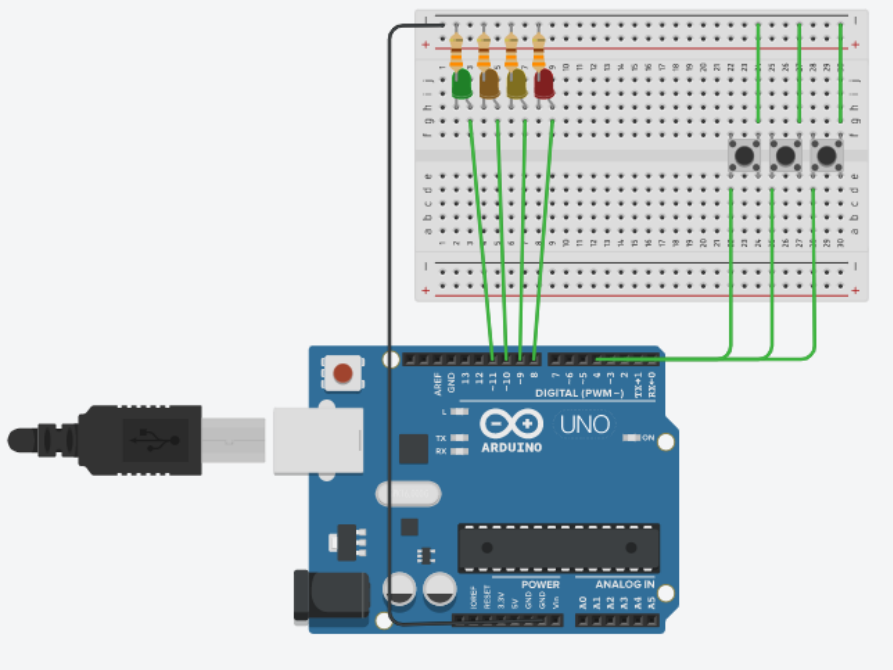
uma função que imprime várias linhas em branco de modo a separar cada jogo e imediatamente a essa função ter acabado o jogo é reiniciado.

```
//Vitoria
if(base==target){
 Serial.println("Ganhou, congratz!!!!!");
 delay(2500);
 limpar_serial();
 break;
```

Figura 9-Vitoria

```
//Derrota(Acabou o tempo)
if(base!=target){
  Serial.println("O tempo acabou, better luck next time");
}
```

Figura 10-Derrota



Projeto no tinkercad: [Link]