TRABALHO PRÁTICO INDIVIDUAL DE TECNOLOGIA INFORMÁTICA

O objetivo deste jogo é acertar no resultado da operação entre dois números codificados em formato binário.

É executada uma de três operações binárias **AND, OR ou XOR**, que é indicada a partir de uma cor do led RGB.

O resultado da operação é indicado pelo número de vezes que o botão é pressionado, havendo um tempo limite para se indicar o número que é de 7 segundos.

No código que realizei para este trabalho comecei por indicar todas as variáveis que iria usar ao longo do mesmo.

```
int botao = 2;
int vermelho = 5;
int verde = 3;
int azul = 4;
int contador = 0;
int ledsverdes=random(0,16);
int ledsvermelhos=random(0,16);
int red=5,green=3,blue=4;
int varrimento;
unsigned long resettime= 1000;
int lastbuttonState= HIGH;
int buttonState;
unsigned long debounceDelay = 5;
unsigned long lastDebounceTime = 0;
bool flag=true;
bool flag2 = true;
```

De seguida configurei a parte do setup onde coloquei:

-O Serial.begin e todos os leds incluindo o rgb e o botão.

```
void setup()
{
  Serial.begin(9600);
  pinMode(10, OUTPUT);
  pinMode(11, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(13, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(7, OUTPUT);
  pinMode(8, OUTPUT);
  pinMode(9, OUTPUT);
  pinMode(botao, INPUT_PULLUP);
  pinMode(vermelho,OUTPUT);
  pinMode(verde,OUTPUT);
  pinMode(azul,OUTPUT);
```

De seguida avançamos para a parte mais complexa que é a parte do loop.

Nessa parte comecei por fazer com que nos fossem mostradas as frases iniciais do jogo **(" Welcome to the big math game! ") e ("Press Button to start!")** usando a ajuda das variaveis flag que chamei no inicio do código, com isso consegui fazer com que as mensagens não fossem spammadas por todo o código.

De seguida configurei o contador que é o número de vezes que o botão é clicado para o ususário tentar acertar o número certo dado pelo jogo ,isto, usando o **debounce**, **LastButtonState**, **millis** e o **LastDebounceTime** e ainda outra **Flag** para o primeiro click não contar para o contador mas sim para iniciar o jogo**.**

```
void loop(){
  flag=true;
  flag2 = true;
  while(flag2){
    Serial.println("#### Welcome to the big math game! ###");
    Serial.println("Press Button to start!");
  flag2 = false;
  }

  contador=0;
  while(flag){
  int reading = digitalRead(botao);

    if (reading != lastbuttonState){
     lastDebounceTime= millis();
       }
    if ((millis()-lastDebounceTime)< debounceDelay){
      if (reading!= buttonState){
        buttonState=reading;
          if (buttonState != HIGH){
              flag=false;
```

Após configurado o contador avançamos para a parte do jogo em si, onde comecei por gerar os dois números random de 0 a 15 pois os números a partir do 16 em binário já não seriam possíveis de representar apenas com 4 leds teria de ser sempre com um número de leds superior.

Depois de ter os números gerados tínhamos de os associar aos leds, para isso usei os deslocamentos com a ajuda da divisão inteira onde me iria dar sempre um resultado de 0 ou 1 que era o ideal para colocar na função digitalWrite para me acender e desligar os leds e assim ficavam associados os leds aos números gerados.

Após isso configurei as operações e associei cada operação a uma cor do led RGB, gerando aleatoriamente uma função para os dois números random gerados acima.

```
int resultado=0;
   int ledsverdes=random(0,16);
   int ledsvermelhos=random(0,16);
  for(int i=10;i<=13;i++){
  digitalWrite(i, (ledsvermelhos>> (i-10)) % 2);
  }
  for(int i=6;i<=9;i++){
    digitalWrite(i, (ledsverdes>> (i-6)) % 2);
  }


  int operacao=random(3,6);
    if(operacao==5){
  digitalWrite(red,1);
      resultado=ledsvermelhos & ledsverdes;
}
  else if(operacao==4){
      digitalWrite(blue,1);
      resultado=ledsvermelhos | ledsverdes;
}
  else{
    digitalWrite(green,1);
    resultado=ledsvermelhos ^ ledsverdes;
```

De seguida defini o tempo de resposta do usuário para 7 segundos usando uma variável de **tempo atual, uma de tempo inicial e usando ainda o millis** e defini que durante esses 7 segundos cada click do botao ira somar um ao contador e assim essa soma total no fim dos 7 segundos vai ser a resposta final do usuário.

De seguida configurei o reset do botao mas não consegui totalmente fazer esse reset pois após o reset no tinkercard ocorriam alguns erros quando continuado a ser jogado o jogo, onde usando o millis o lastDebounceTime defini que se o botao for premido mais de que 1 segundo voltaria tudo ao inicio do loop e para isso chamei a função loop.

```
int tempoinicial=millis();
int tempoatual=millis();
while((tempoatual-tempoinicial)<7000){
tempoatual=millis();
  int reading = digitalRead(botao);
  if (reading != lastbuttonState){
   lastDebounceTime= millis();
     }
  if ((millis()-lastDebounceTime)< debounceDelay){
    if (reading!= buttonState){
      buttonState=reading;
        if (buttonState != HIGH){
            contador++;
            Serial.println(contador);
      }
    }
  }
  lastbuttonState=reading;
  if ((millis()-lastDebounceTime)>resettime){
      if (buttonState == LOW){
      Serial.println("O programa vai ser resetado");
      delay(1000);
      loop();
      }
   }
```

```
if(contador==resultado){
  Serial.println("Ganhou o jogo!!!!");
    int joao = millis();
  while((millis()-joao)<5000){
      for (int i = 0; i<4;i++){
        digitalWrite(operacao, HIGH);
        digitalWrite(6 + i, (ledsverdes>>i&1));
        digitalWrite(10 + i, (ledsvermelhos>>i&1));
      }
      delay(500);
      for (int i = 0; i<4;i++){
        digitalWrite(operacao, LOW);
        digitalWrite(6 + i, (ledsverdes>>i)&0);
        digitalWrite(10 + i, (ledsvermelhos>>i)&0);
      }
      delay(500);
  }
  varrimento = 1;
  while(varrimento<=3)
  {
    for (int i=0; i<=4; i++){
      digitalWrite(6+i, 1);
      digitalWrite(13-i, 1);
      delay(500);
    }
    for (int i=0; i<=4;i++){
      digitalWrite(9-i, 0);
      digitalWrite(10+i, 0);
      delay(500);
    }
    varrimento++;
  }

  digitalWrite(operacao,0);

  loop();
```

Por fim foi só realizar o que acontecia caso o usuário ganhasse ou caso o usuário perdesse ou seja, caso o resultado final do contador fosse igual ao resultado dos dois números random com a respetiva função fornecida pela cor do led RGB. Quando o usuário vence os leds respetivos aos números calhados devem piscar e de seguida deve ocorrer um varrimento de todos os leds. Já quando o usúario perde todos os leds devem apagar exceto o led RGB que deve piscar durante 5 segundos.

Aqui encontramos o modo vitória á esquerda e o modo derrota á direita.

```
}else{
  Serial.println("Perdeu o jogo");
  for(int i=0;i<=3;i++){
    digitalWrite(6,LOW);
    digitalWrite(7,LOW);
    digitalWrite(8,LOW);
    digitalWrite(9,LOW);
    digitalWrite(10,LOW);
    digitalWrite(11,LOW);
    digitalWrite(12,LOW);
    digitalWrite(13,LOW);
    digitalWrite(operacao,0);
    delay(1250);
    digitalWrite(operacao,1);
    delay(1250);

  }
  digitalWrite(operacao,0);
  loop();
}
```