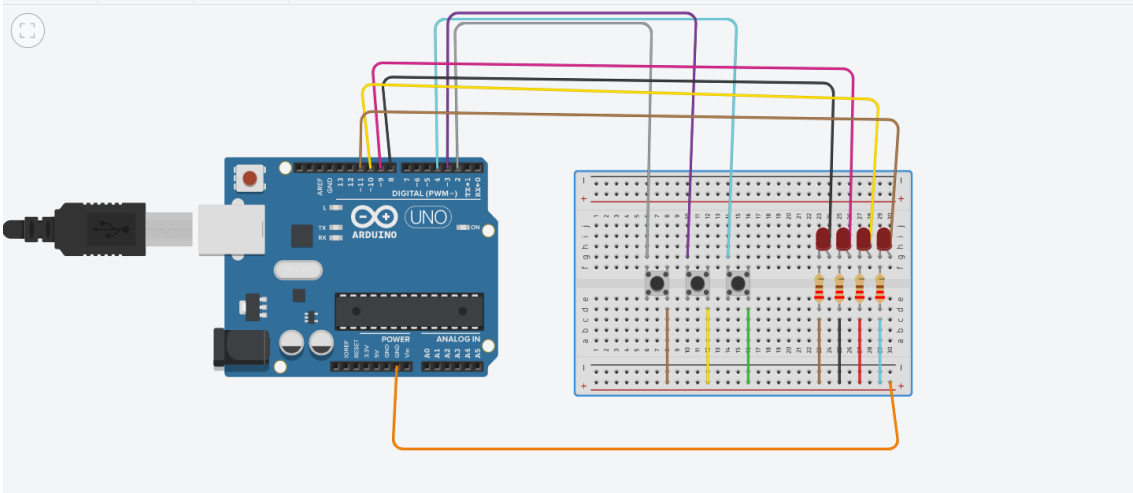
Relatório:

Montagem do circuito:



Na montagem do circuito procedeu-se à montagem de 4 *leds*, ligados desde o pino digital 1 ao 4, alimentados energeticamente pelo GROUND, utilizando resistências de 220 Ω.

Os três botões de pressão encontram-se ligados no modo *input pullup*, sendo que o correspondente à operação XOR se encontra ligado ao pino digital 2, o OR ao pino 3, e o AND ao pino 4. Todos são alimentados pelo GROUND comum aos *leds*.

Código:

O randomSeed() presente no setup() utiliza as oscilações de voltagem presentes no pino analógico A5 para tornar mais eficientes as funções random utilizadas no loop().

```
22
23 int leitor_porta_serie() {
24   if (Serial.available() > 0) {
25     String a = Serial.readStringUntil("\n");
26     valor = a.toInt();
27     feito = true;
28     Serial.println("Valor lido:" + String(valor, BIN));
29   }
30   return valor;
31 }
32
```

Dentro das funções globais que serão mais tarde chamadas dentro do loop(), a função leitor_porta_serie() vai ler todos os carateres até ao final da linha da string introduzida pelo utilizador de seguida convertendo-a em inteiro. Também atualiza a variável booleana definida globalmente e a variável global inteira valor.

```
33
34 ////Efetuar operações
35 int operacoes(int operacao, byte primeiro_numero, int porta_serie) {
36   if (operacao == 2) {
37     primeiro_numero = primeiro_numero ^ porta_serie;
38     return (primeiro_numero);
39
40   }
41   if (operacao == 4) {
42     primeiro_numero = primeiro_numero & porta_serie;
43     return (primeiro_numero);
44   }
45   if (operacao == 3) {
46     primeiro_numero = primeiro_numero | porta_serie;
47     return (primeiro_numero);
48   }
49
```

A função operacoes() vai efetuar uma operação diferente consoante o número definido na variável local operacao.

```
52 //////Mostrador de tempo
53 void leds(unsigned long timer) {
54   if ((millis() - timer) > (tempo_de_jogo) * (1.0 / 4)) {
55     digitalWrite(8, HIGH);
56   }
57   if ((millis() - timer) > ((tempo_de_jogo) * (1.0 / 2))) {
58     digitalWrite(9, HIGH);
59   }
60   if ((millis() - timer) > ((tempo_de_jogo) * (3.0 / 4))) {
61     digitalWrite(10, HIGH);
62   }
63   if ((millis() - timer) >= ((tempo_de_jogo))) {
64     digitalWrite(11, HIGH);
65   }
66 }
```

A função leds vai ativar sucessivamente conforme se verifique cada uma das condições, ou seja, caso tempo supere ¼ do tempo acende o led associado ao pino 8, caso supere ½ acende o pino 9 e assim sucessivamente.

```
90  while ((millis() - timer) < tempo_de_jogo )/////Inicio do round
91  {
92    leds(timer);
93
94    int valor_serie = leitor_porta_serie();
95
96    if (feito == true) {
97      ////Operações a executar(verificação de botões ativos)
98      if (digitalRead(3) == LOW) {
99        numero_inicial = operacoes(3, numero_inicial, valor_serie);
100       Serial.println("Valor calculado:" + String(numero_inicial, BIN));
101       feito = false;
102       Serial.println("Introduz um valor:");
103
104     }
105     else if (digitalRead(2) == LOW && (botoes_disponiveis == 1 || botoes_disponiveis == 3)) {
106       numero_inicial = operacoes(2, numero_inicial, valor_serie);
107       Serial.println("Valor calculado:" + String(numero_inicial, BIN));
108       feito = false;
109       Serial.println("Introduz um valor:");
110     }
111     else if (digitalRead(4) == LOW && (botoes_disponiveis == 2 || botoes_disponiveis == 3)) {
112       numero_inicial = operacoes(4, numero_inicial, valor_serie);
113       Serial.println("Valor calculado:" + String(numero_inicial, BIN));
114       feito = false;
115       Serial.println("Introduz um valor:");
```

No loop() é inicializada uma condição *while* que regula o tempo disponível em cada *round*(tempo_de_jogo) e os controlos disponíveis dentro do mesmo. Dentro dessa condição é então inicializada a função que liga os *leds* ordenadamente, que por utilizar as mesmas variáveis que a condição *while* vai estar sincronizada com a mesma.

Para controlar a utilização de operações, inicialmente armazena-se numa variável inteira o valor convertido dado pela função de leitura de porta série. Para além disso, sempre que esta função é executada (sempre que se escreve algo na porta série) a variável booleana é atualizada permitindo avançar para a próxima condição, ou seja, as operações só ficam disponíveis caso algo tenha sido escrito na porta série.

Assim que as operações estejam disponíveis, a operação executada depende do botão clicado e de quais estejam disponíveis, exceto com a operação OR, que está sempre disponível. Assim que um dos botões disponíveis tenha sido premido a função operacoes() vai ser chamada, com o argumento operacao a controlar a condição a ser efetuada dentro da função. Na primeira iteração o cálculo vai ser executado entre o número inicial e o valor lido na porta série, no entanto a variável correspondente ao número inicial vai sendo atualizada para que nas seguintes iterações o cálculo seja feito com o valor previamente calculado. Após cada operação a variável booleana volta ao estado inicial para se reentrar na condição *if* inicial e se poder continuar a efetuar operações.

```
      if (numero_inicial == target) {////Vitória!!
        Serial.println("GANHOU");
        vitoria = true;
        break;
      }
    }
  }
  if (vitoria == false) {
    Serial.println("Perdeu");
  }
  for (int i = 8; i <= 11; i++) {
    digitalWrite(i, LOW);
  }
}
```

Por fim caso não se tenha verificado a condição de vitória, a variável booleana “vitória” (inicializada no loop() a *false)* é atualizada impedindo assim que quando saia da condição *while* imprima a mensagem de derrota, e por sua vez caso a variável booleana não atualize (perder o jogo) só imprime a mensagem de derrota.