Relatório de TI

Nota: Coisas não referidas neste relatório, estão eventualmente explicadas em comentários ao longo do código.

O meu código está dividido em 4 funções bastante importantes: a que mostra as sequências, a que desliga todos os leds, a que faz o varrimento, e a que liga os leds.

1. Irei começar pela função que **mostra as sequências** (Figura 1), esta função tem como seu parâmetro a variável ***j*** que é uma variável que aumenta sempre que se clica no botão sim/não.

```
void mostrarSequencia(byte j)
{
  byte count = 0;
  Serial.println(">> Sequencia:");
  for(byte i = 0; i <= 127; i++)
  {
    if(i >> j & 0b1)
    {
      Serial.print(i); Serial.print(" ");
      count++;
      if((count %= 25) == 0) Serial.println("");
    }
  }
  Serial.println("\n> O numero esta nesta sequencia?");
}
```

Figura 1 – Mostra as Sequências

2. A função que **liga os leds** (Figura 2), tem como parâmetro o número até agora calculado, e com as operações bitwise liga os leds nas posições corretas.

```
void ligarLedsBinario(byte num)
{
  for(byte i = 0; i < 7; i++){
    bool res = (num >> i) & 0b1;
    if(res) digitalWrite(6+i, HIGH);
  }
}
```

Figura 2 – Liga os leds

3. A função que **desliga os leds** (Figura 3), é aquela que no fim dos 5 segundos de mostrar o número, *desliga os leds*, e preparar bem para a função do varrimento funcionar perfeitamente.

```
void desligarLeds()
{
  for(int i = 0; i <= 7; i++) digitalWrite(6+i, LOW);
}
```

Figura 3 – Desliga os leds

4. Por último e menos importante, a função do **varrimento** (Figura 4), faz com que no fim de apagar os leds (função anterior), os leds acendam e apaguem sucessivamente e da direita para a esquerda (vista de cima).

```
void varrimentoLeds()
{
  for(int i = 0; i < 7; i++)
  {
    if((i-1) < 0) digitalWrite(6+i, HIGH);
    else if(((i-1) >= 0) && (i < 7))
    {
      digitalWrite(6+i, HIGH);
      digitalWrite(6+i-1, LOW);
    }
    delay(300);
  }
  digitalWrite(12, LOW);
}
```

Figura 4 - Varrimento

Quanto ao resto do código eu usei o Debounce com *millis()* para garantir que o botão só é premido uma única vez, e para tornar mais fácil a implementação do long press.

Os *if's statements* no caso de eu clicar num dos botões é feito da seguinte forma: Se eu clicar num dos botões e for o botão sim (respetivamente, não), então faz a função do botão sim (respetivamente, não).

Também usei uma estratégia para se estiver a segurar o botão não acontecer nada até o largar. Por exemplo, se eu quiser mostrar a primeira sequência tenho que clicar no botão e larga-lo antes de passarem dois segundos, no caso de eu não largar nos dois segundos, acontece o **reset** que resumindo é apenas um reset nas variáveis (iguala todas as variáveis a 0).

Segue ainda em anexo foto de **montagem do tinkercad** (Figura 5).

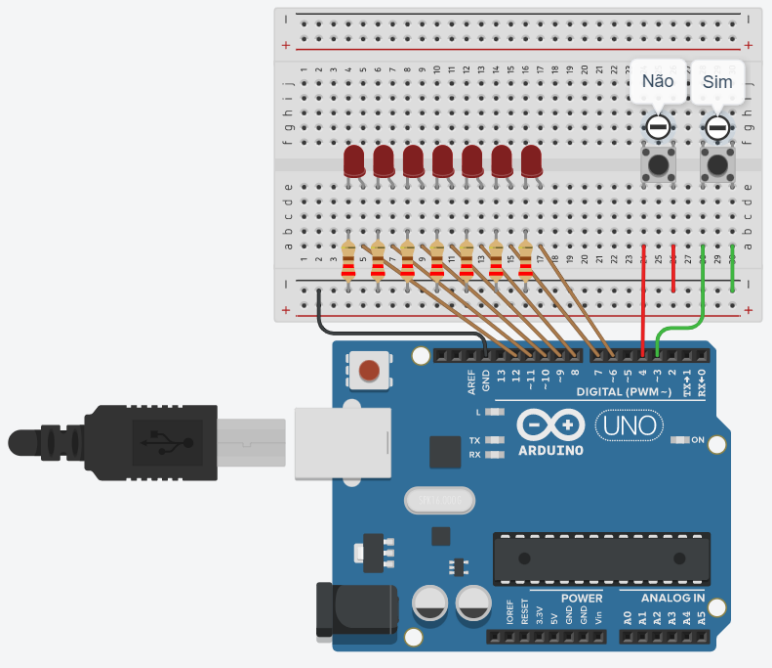

Figura 5 – Circuito tinkercad

Filipe Alexandre Rodrigues

2021218054