Universidade do Porto

Faculdade de Engenharia

FEUP

Electronic Sand Glass

Diogo Chaves, 202004540

Joana Pereira, 202004651

Relatório do Trabalho Prático no âmbito da unidade curricular Arquitetura de Computação Embarcada, do 1ºano do Mestrado em Engenharia Eletrotécnica e de Computadores.

11/11/2023

Índice

1 Introdução

O objetivo deste trabalho foi projetar uma ampulheta eletrónica para funcionar como um temporizador, utilizando um Raspberry Pi Pico. Primeiro começamos por montar a fita de leds e os botões na configuração demonstrada na figura 1.

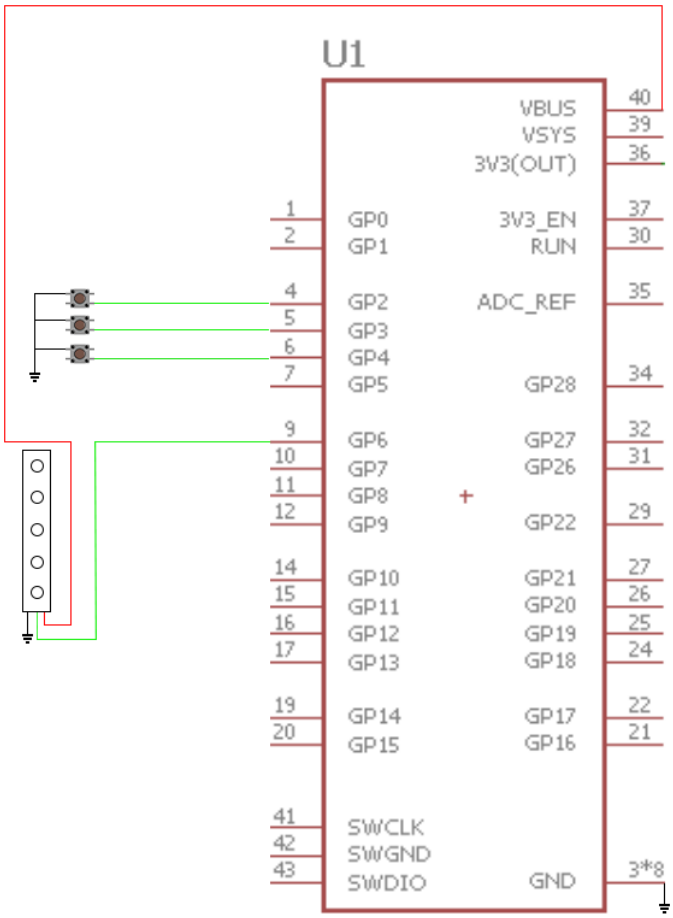

Figure 1: Montagem no raspberry

2 Máquinas de Estado

Estas foram as máquinas de estado que projetamos para este trabalho, de acordo com o guião disponibilizado.

2.1 Máquina principal

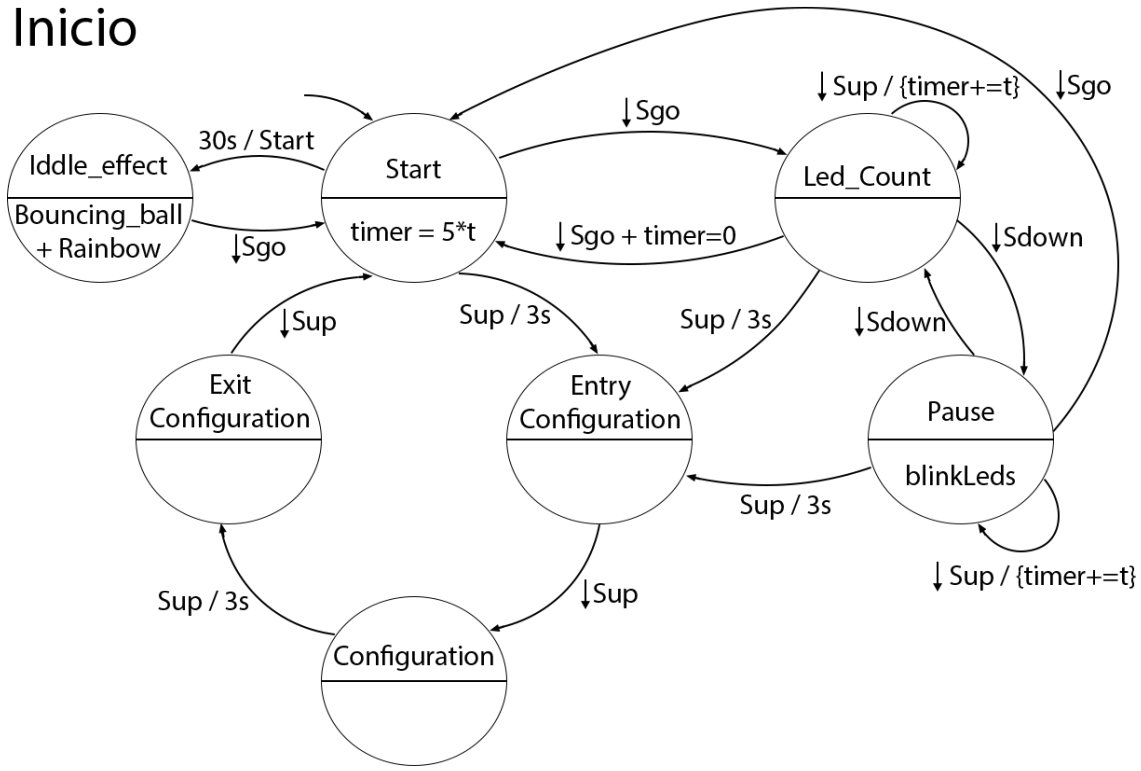

Figure 2: Máquina inicial

A contagem começa quando o botão S_{go} é pressionado. Os 5 LEDS acendem e vão apagando sucessivamente com o efeito, tempo e cor escolhidos no modo de configuração. Cada vez que o S_{down} é pressionado, a ampulheta pára/continua e sempre que S_{up} é pressionado o *timer* é incrementado com o tempo definido no modo de configuração. Já quando o temporizador estiver parado, todos os LEDS que estavam ligados durante a contagem, ficam a piscar. Quando a ampulheta acaba, todos os LEDS ficam a piscar à mesma frequência com cor vermelha. A qualquer momento, o temporizador pode ser reiniciado ao pressionar o botão S_{go}. Se o temporizador estiver parado durante 30s entramos no Modo de suspensão. Para este modo escolhemos os efeitos *rainbow* e *bouncing ball* que são reproduzidos alternadamente.

2.2 Modo de Configuração

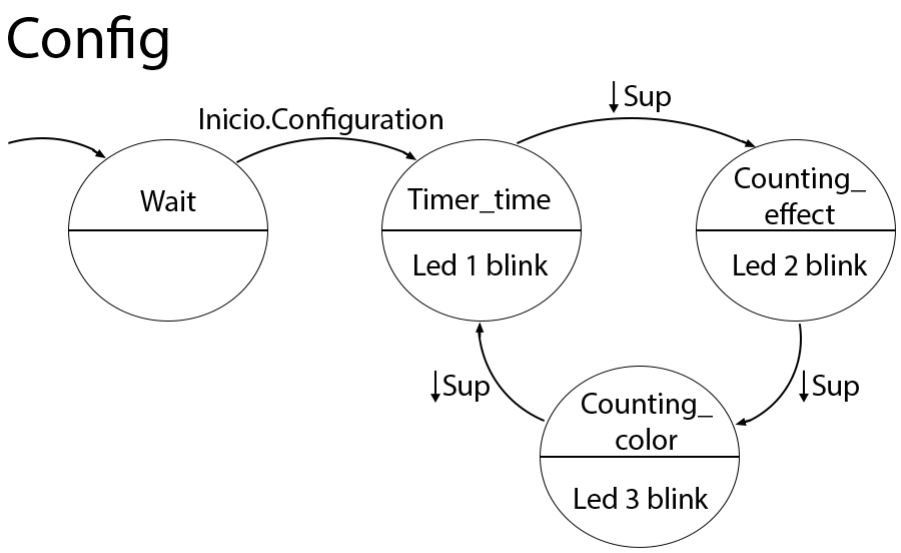

Figure 3: Máquina das configurações

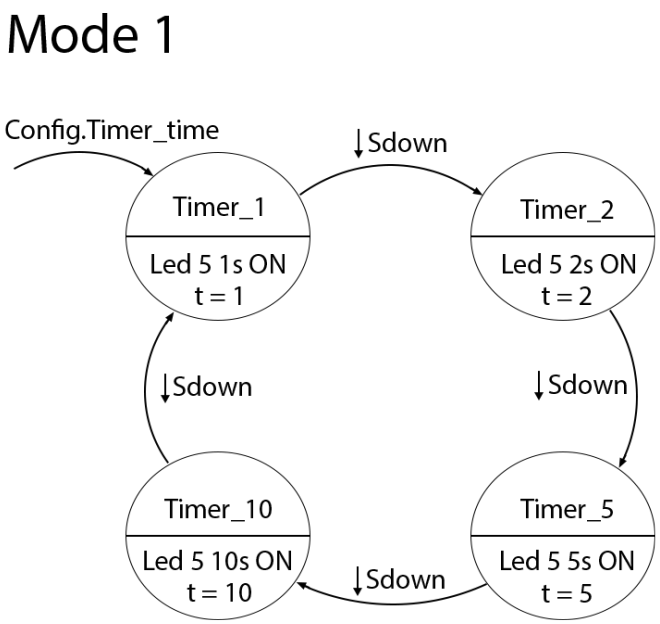

Figure 4: Modo1

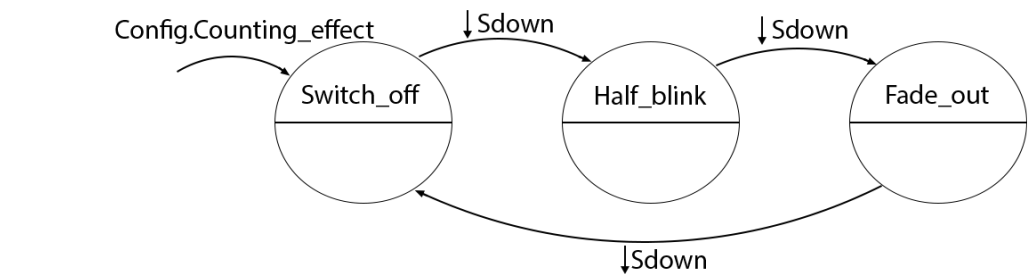

Figure 5: Modo2

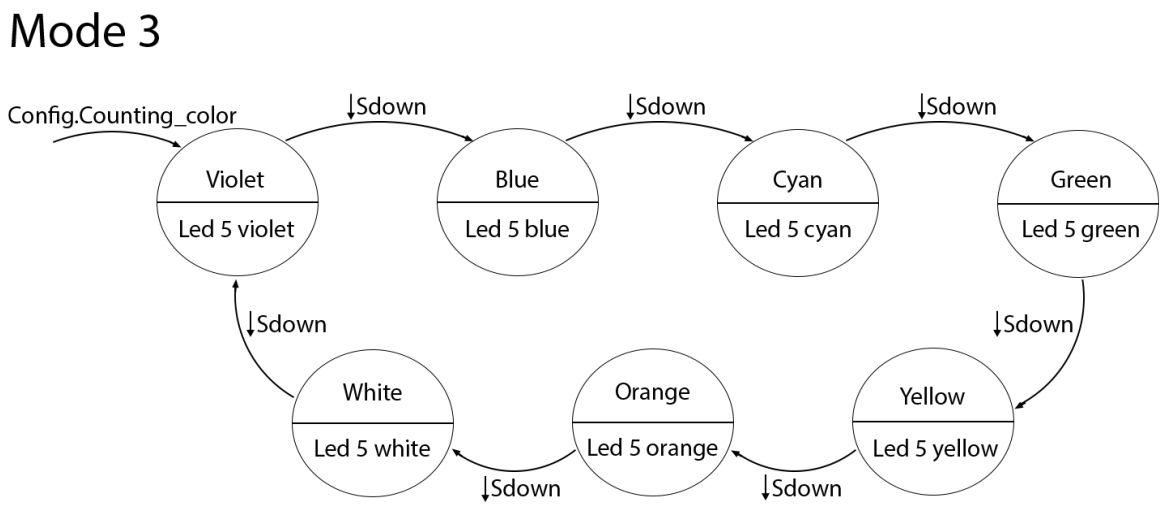

Figure 6: Modo3

Quando pressionado o botão S_{up} por mais de 3 segundos entramos no modo de configuração. Dentro deste, existem 3 modos, para alternar entre eles basta pressionar o S_{up}. O Modo 1 serve para escolher o intervalo de tempo que cada LED vai estar ligado, o Modo 2 é utilizado para selecionar qual o efeito que os LEDS vão ter quando apagarem e o Modo 3 define a cor dos LEDS. Em cada um destes modos o LED 5 é utilizado para exemplificar o tempo, o efeito e a cor que todos os LEDS vão ter quando o temporizador for ligado. Para sair deste modo basta, a qualquer momento, carregar 3 segundos no botão S_{up} e o temporizador fica pronto para começar uma nova contagem, com as novas configurações escolhidas.

3 Implementação de software

Para a implementação das máquinas de estado, utilizamos as seguintes bibliotecas: *Adafruit NeoPixel* para controlar os LEDS, *Bounce2* para controlar os botões e a *elapsedMillis* para contar o tempo.

Primeiro começamos por definir os LEDS no pin 6 e os botões S_{go}, S_{up} e S_{down} nos pins 2, 3 e 4 correspondentemente. Posto isto, inicializamos todos os *Timers* utilizados a zero e as structs das máquinas de estado, indicando o estado em que estas serão inicializadas.

3.1 Funções Led

Foram implementadas 3 funções que representam as 3 formas de atuação do temporizador, nomeadamente *Switch off*, *Half blink* e *Fade out*. Estas 3 funções, apresentam uma implementação semelhante, ou seja, todas elas fazem apagar os LEDS sucessivamente dependendo do tempo que já passou na contagem. A maior diferença é a forma como os LEDS são apagados, a primeira apenas apaga um LED ao fim do tempo definido, a segunda faz o LED piscar na segunda parte do intervalo selecionado antes de apagar e a terceira diminui a intensidade do LED durante o intervalo escolhido, até que apaga. No modo 2, estas funções são todas demonstradas utilizando o LED 5, guardado o efeito assim que S_{up} é pressionado.

```
void updateLed(int r, int g, int b);

void HalfBlink(int pin, int r, int g, int b, unsigned long t_interval,
unsigned long fast_blink);

void fade(int pin, unsigned long duration, int r, int g, int b);
```

Uma das *features* deste temporizador é demonstrar a atuação do temporizador nos modos de configuração. No modo 1, o LED 5 demonstra o tempo que um LED vai ficar ligado e esse intervalo pode ser alterado pressionando S_{down}. Na demonstração do intervalo definido usamos a função LedTime(). Após a saída deste modo de configuração, o intervalo definido é guardado.

```
void LedTime(int pin, unsigned long interval, int r, int g, int b);
```

Quanto à contagem do tempo, a função updateTimer() decrementa o tempo com o intervalo definido nas configurações. Por exemplo, se for definido um intervalo de 5 segundos por LED, o nosso timer contará 25 segundos, pois são 5 LEDS. Ou seja, o nosso decremento será de 5 segundos. Podemos também incrementar o temporizador ao longo da contagem. Isto é feito pressionando o botão Sup que faz incrementar o tempo com o intervalo definido na configuração em utilização.

```
void updateTimer( unsigned long interval);
```

Terminada a contagem, os LEDS ficam todos a piscar com um período de 200ms com a cor vermelha. A função Timeup() é responsável por essa tarefa.

```
void Time_up();
```

Quando o temporizador entra no modo de suspensão após 30 segundos sem qualquer operação, optamos por apresentar dois efeitos que são reproduzidos de forma alternada. Temos um efeito arco-íris em que os LEDS mudam de cor de forma aleatória e um efeito *bouncing ball* que, como o nome diz, imita uma bola a percorrer os LEDS com cores aleatórias.

```
void rainbow(unsigned long wait);

void Bouncing_Ball();
```

3.2 Setup

Dentro do Setup, são inicializados os 3 botões necessários para controlar todas as funcionalidades do temporizador. A inicialização foi feita no modo input pullup, pois deste modo a montagem na breadboard fica mais simples.

Nesta secção é também necessário inicializar a fita LED, cujos parâmetros já tinham sido definidos anteriormente.

```
Sgo.attach(SGO, INPUT_PULLUP);
Sup.attach(SUP, INPUT_PULLUP);
Sdown.attach(SDOWN, INPUT_PULLUP);

strip.begin();
```

3.3 Loop

Na função de loop são chamadas as máquinas de estado que ficam a correr em ciclo. Para além disso, são atualizados os valores dos botões (S_{go}, S_{up} e S_{down}) e, desta forma as máquinas transitam entre estados quando algum botão é pressionado.

```
void loop(){
    Sgo.update();
    Sup.update();
    Sdown.update();

    //Máquina principal
    INICIO();

    //Configuration mode
    CONFI_MODE();
    MODE1();
    MODE2();
    MODE3();
}
```

4 Conclusão

A implementação do temporizador revelou-se uma adição cativante ao desenrolar da cadeira. Além de cumprir a sua função prática de contar o tempo de maneira simples, o *sand glass* oferece uma excelente experiência sensorial. A simplicidade e versatilidade do sand glass tornam-o numa escolha valiosa em diversos contextos, agregando valor estético à experiência do utilizador.

Em suma, este projeto foi desenvolvido de forma consistente e cumpre todas as funcionalidades, não apresentando qualquer *bug*.