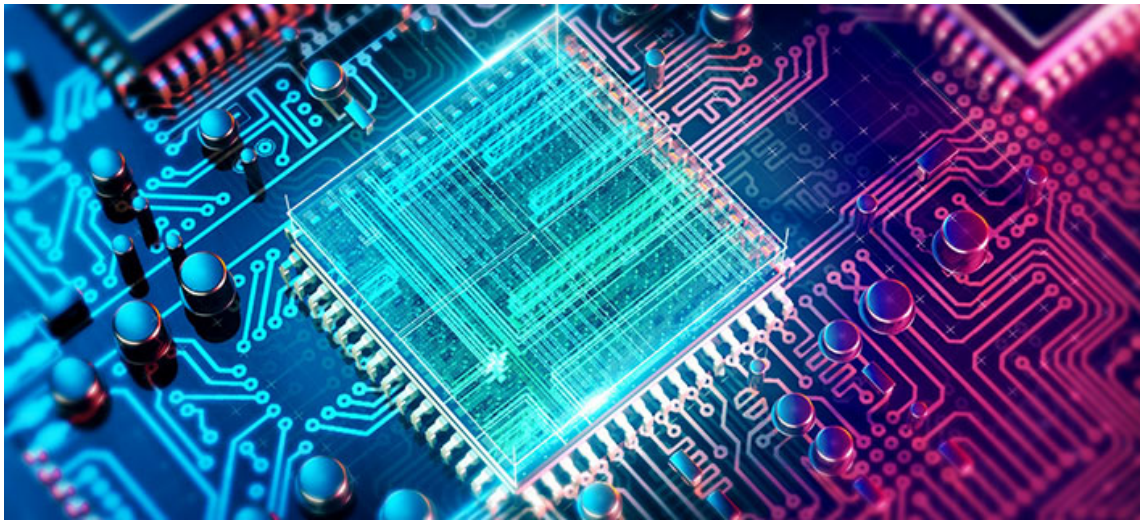
Instituto Superior de Engenharia de Lisboa

Licenciatura em Engenharia Informática e Multimédia

Computação Física - CF - 2122SV

Trabalho Prático 1

Docente Jorge Pais



Realizado por (Grupo 10):
Roman Ishchuk 43498
Pedro Silva 48965
Cláudia Sequeira 49247

27 de setembro de 2024

Conteúdo

1 Introdução **4**

2 Objetivos **4**

3 Material Utilizado **4**

4 Resolução dos Exercícios **5**

4.1 **Circuito detector de Avarias em Portas Lógicas** 5

4.1.1 Enunciado 5

4.1.2 Entradas e Saídas 5

4.1.3 Tabela de Verdade 5

4.1.4 Mapa de Karnaugh 6

4.1.5 Expressão lógica por simplificação algébrica 6

4.1.6 Circuito Lógico Simplificado 6

4.1.7 Situações de Testes e os Resultados Obtidos 7

4.2 **Somador/Subtrator Binário** 8

4.2.1 Enunciado 8

4.2.2 Somador 8

4.2.3 Subtrator 9

4.2.4 Tabela de verdade Somador - S e Carry out 9

4.2.5 Mapas de Karnaugh Somador - S e Carry out 9

4.2.6 Tabela de verdade Subtrator - S e Borrow out 10

4.2.7 Mapas de Karnaugh Subtrator - S e Borrow out 10

4.2.8 Tabela de verdade Overflow 11

4.2.9 Mapa de Karnaugh Overflow 11

4.2.10 Circuito lógico simplificado 12

4.2.11 Situação de testes e resultados obtidos 13

4.3 **Contador Binário** 14

4.3.1 Enunciado 14

4.3.2 Modelo Entradas e Saídas 14

4.3.3 Modelo Moore-Mealey 14

4.3.4 Tabelas de Verdade - F.S e F.E.S 15

4.3.5 Função de Saída 15

4.3.6 Mapas de Karnaugh 15

4.3.7 Função de Estado Seguinte 16

4.3.8 Mapas de Karnaugh 16

4.3.9 A.S.M. 17

4.3.10 Circuito Lógico Simplificado 18

4.3.11 Situações de Testes e os Resultados Obtidos 18

5 Conclusões **19**

6 Bibliografia **19**

7 Código Arduíno **20**

7.1 Circuito detetor de Avarias em Portas Lógicas 20

7.2 Somador/Subtrator 21

7.3 Contador Binário 23

Lista de Figuras

1 Modelo Entradas e Saídas — Detetor de Avarias em Portas Lógicas . 5
2 Tabela de Verdade — Detetor de Avarias em Portas Lógicas 5
3 Mapa de Karnaugh — Detetor de Avarias em Portas Lógicas 6
4 Circuito Lógico Simplificado — Detetor de Avarias em Portas Lógicas 6
5 Montagem no Arduino — Detetor de Avarias em Portas Lógicas . . . 7
6 Modelo Entradas e Saídas — Somador/Subtrator 8
7 Abordagem modular do Somador 8
8 Abordagem modular do Subtrator 9
9 Tabela de verdade de S e Carry out 9
10 Mapa de Karnaugh - S 9
11 Mapa de Karnaugh - Cyout 9
12 Tabela de verdade de S e Borrow out 10
13 Mapa de Karnaugh - S 10
14 Mapa de Karnaugh - Cyout 10
15 Tabela de verdade de Overflow 11
16 Mapa de Karnaugh de Overflow 11
17 Circuito lógico simplificado Somador/Subtrator 12
18 Montagem física do Somador/Subtrator 13
19 Exemplo de uma soma/subtração 13
20 Modelo Entradas e Saídas — Contador 14
21 Modelo Entradas e Saídas — Contador 14
22 Tabela de Verdade — Função FS 15
23 Mapa de Karnaugh-S2 15
24 Mapa de Karnaugh-S1 15
25 Tabela de Verdade — Função FES 16
26 Mapa de Karnaugh-F1 16
27 Mapa de Karnaugh-F0 16
28 Modelo Entradas e Saídas — Contador 17
29 Modelo Entradas e Saídas — Contador 18
30 Montagem no Arduino — Contador 18

1 Introdução

Este trabalho aborda o tema dos circuitos combinatórios e dos circuitos sequenciais, baseados em funções lógicas. Fomos introduzidos à utilização de tabelas de verdade para sistematização de um objetivo,e depois a utilização dos Mapas de Karnaugh para obter uma simplificação da função lógica. Aprendemos a utilizar o A.S.M. (Algorithmic State Machine) no desenho de circuitos sequenciais. No ultimo exercícios abordamos os circuitos sequenciais que carecem de flip-flops para garantir o sincronismo entre vários dispositivos.

No final, para consolidar os conhecimentos foi essencial a passagem para o mundo real (em oposição do mundo ideal, como as simulações em TinkerCad), nós utilizamos o Arduíno para isso mesmo. O Arduíno é um computador de placa única constituída por hardware e software. O mesmo permite desenvolver projetos eletrónicos, com sensores e atuadores, através da programação em C/C++.

2 Objetivos

Um dos grandes objetivos deste trabalho prático é aprender o básico das funções lógicas e aplicar os conhecimentos na implementação em Arduino.

Foi nos proposto desenvolver 3 projetos, nomeadamente, um circuito lógico que deteta avarias em portas lógicas (AND/OR), um somador/subtrator a 3 bits e um contador binário de módulo 4 (4 estados) com comportamento crescente e decrescente.

3 Material Utilizado

- Arduíno UNO R3
- Breadboard
- Cabos Jumper macho-macho
- Resistências 220Ω e 10kΩ
- Botões N.O.
- LED (vermelho)

4 Resolução dos Exercícios

4.1 Circuito detector de Avarias em Portas Lógicas

4.1.1 Enunciado

Projete um circuito lógico para detetar portas lógicas avariadas, AND e OR, de duas entradas. O circuito deve acender um LED sempre que a porta evidenciar um comportamento anómalo. Desenhe o modelo de entradas e saídas do circuito

4.1.2 Entradas e Saídas

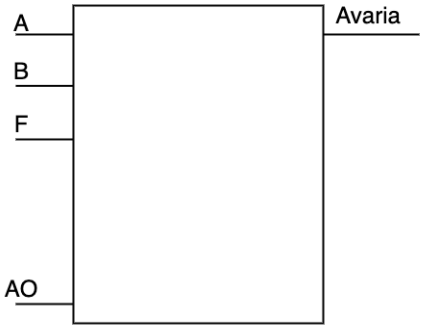


Figura 1: Modelo Entradas e Saídas — Detetor de Avarias em Portas Lógicas

As nossas entradas são: os botões AO, A, B e F, em que o resultado da operação lógica entre A e B com o símbolo lógico AO (se AO = 1 a operação é um AND, se AO = 0 a operação é um OR) vai ser comparado ao valor de F. A saída é um LED de nome Avaria que vai ter como valor o resultado dessa comparação.

4.1.3 Tabela de Verdade

Realize a tabela de verdade que descreve o funcionamento pretendido

AO	A	B	F	AVARIA
0	0	0	0	0
1	0	0	0	0
0	0	0	1	1
1	0	0	1	1
0	0	1	0	1
1	0	1	0	0
0	0	1	1	0
1	0	1	1	1
0	1	0	0	1
1	1	0	0	0
0	1	0	1	0
1	1	0	1	1
0	1	1	0	1
1	1	1	0	1
0	1	1	1	0
1	1	1	1	0

Figura 2: Tabela de Verdade — Detetor de Avarias em Portas Lógicas

4.1.4 Mapa de Karnaugh

Obtenha a expressão lógica simplificada, por intermédio de mapas de Karnaugh **Avaria:**

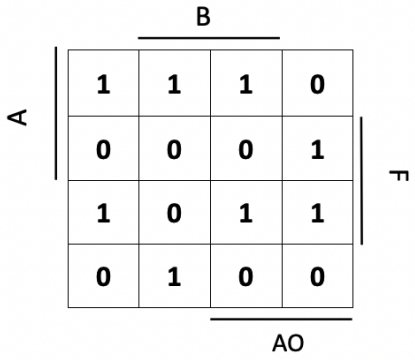


Figura 3: Mapa de Karnaugh — Detetor de Avarias em Portas Lógicas

$$Avaria = A \cdot F/ \cdot AO/ + A \cdot B \cdot F/ + B \cdot AO/ \cdot F/ + A/ \cdot AO \cdot F + A/ \cdot F.B/ + F \cdot AO \cdot B/$$

4.1.5 Expressão lógica por simplificação algébrica

$$Avaria = A/ \cdot B/ \cdot F \cdot AO/ + A/ \cdot B/ \cdot F \cdot AO + A/ \cdot B \cdot F/ \cdot AO/ + A/ \cdot B \cdot F \cdot AO + A \cdot B/ \cdot F/ \cdot AO/ + A \cdot B/ \cdot F \cdot AO + A \cdot B \cdot F/ \cdot AO/ + A \cdot B \cdot F/ \cdot AO$$

$$Avaria = A/ \cdot B/ \cdot F + A/ \cdot B \cdot F/ \cdot AO/ + A/ \cdot B \cdot F \cdot AO + A \cdot B \cdot F/ + A \cdot AO \cdot F/ + A \cdot B/ \cdot F \cdot AO$$

$$Avaria = A/ \cdot B/ \cdot F + A/ \cdot F \cdot AO + A/ \cdot B \cdot F/ \cdot AO/ + A \cdot B \cdot F/ + A \cdot AO \cdot F/ + A \cdot B/ \cdot F \cdot AO$$

$$Avaria = A/ \cdot B/ \cdot F + B/ \cdot F \cdot AO + A/ \cdot F \cdot AO + A \cdot B \cdot F/ + B \cdot F/ \cdot AO/ + A \cdot AO/ \cdot F/$$

4.1.6 Circuito Lógico Simplificado

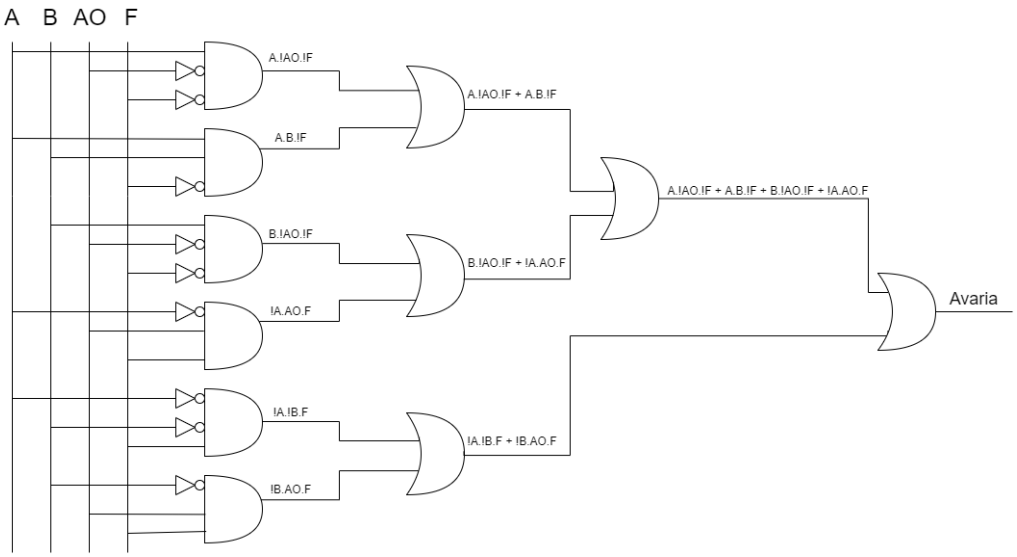


Figura 4: Circuito Lógico Simplificado — Detetor de Avarias em Portas Lógicas

4.1.7 Situações de Testes e os Resultados Obtidos

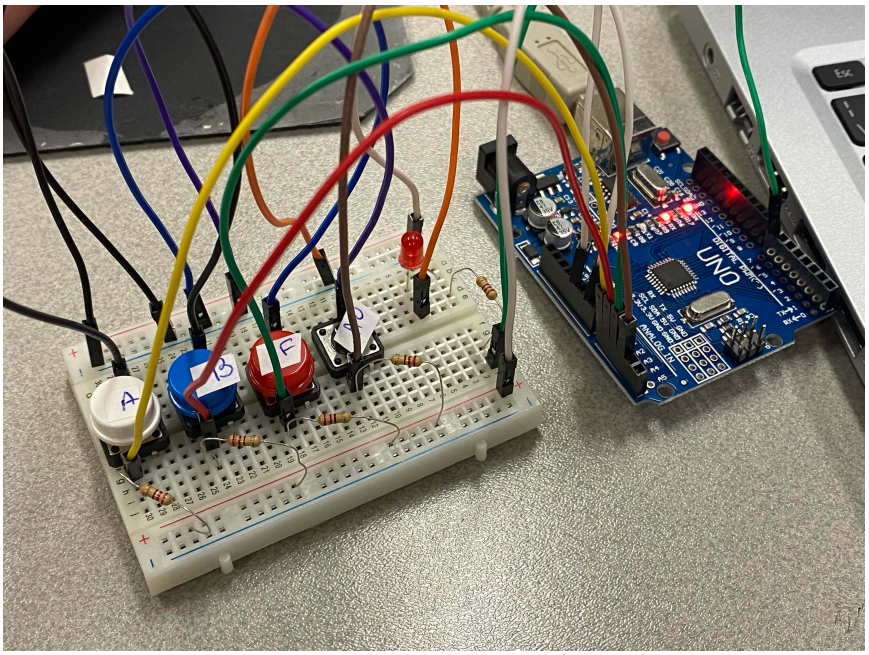


Figura 5: Montagem no Arduino — Detetor de Avarias em Portas Lógicas

O funcionamento do circuito é o esperado, por exemplo quando se activa apenas o botão F, logo $F = 0$ o LED Avaria acende pois A.B é igual a 1 e não a 0, isto não acontece se nenhum botão estiver activo pois $A \cdot B = 1$ e $F = 1$. Para testar o botão AO podemos activar apenas o botão A fazendo com que o LED acenda pois $A/ \cdot B = 0$ o que é diferente de F=1, mas ao activar o botão AO o símbolo lógico da operação muda de $'\cdot'$ para $'+'$, ou seja, fica $A/ + B = 1$ que é igual a F, logo não ocorre Avaria.

4.2 Somador/Subtrator Binário

4.2.1 Enunciado

Projete um circuito somador (S = A + B) e subtrator (S = A - B), de dois números binários de três bits, ou seja, A = [A2 A1 A0], B = [B2 B1 B0]. O resultado também é a três bits: S = [S2 S1 S0]. O utilizador deve poder selecionar qual é a operação que pretende realizar. Para além do resultado da operação aritmética a três bits (S), o sistema deverá fornecer o resultado das flags de **Carry** (Cy) para a soma e **Borrow** (Bw) para a subtração, e ainda **Overflow** (Ov) e **Zero** (Z) para qualquer uma das duas.

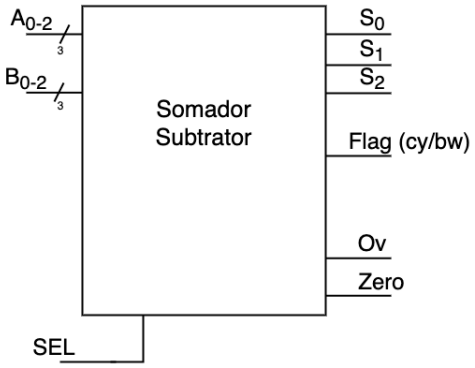


Figura 6: Modelo Entradas e Saídas — Somador/Subtrator

4.2.2 Somador

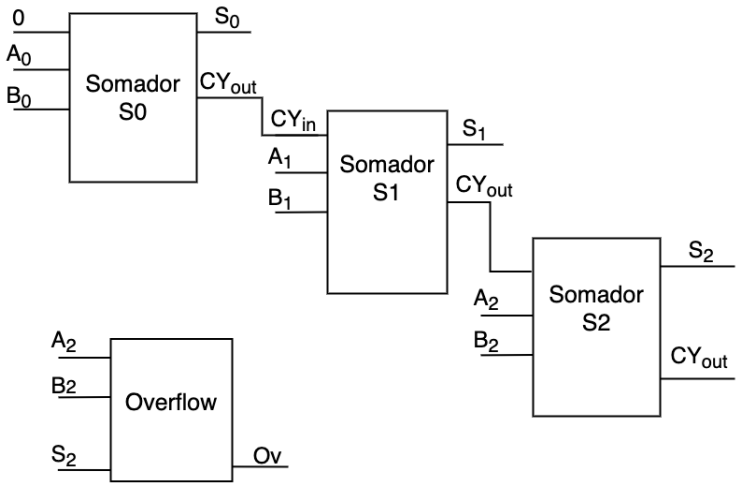


Figura 7: Abordagem modular do Somador

4.2.3 Subtrator

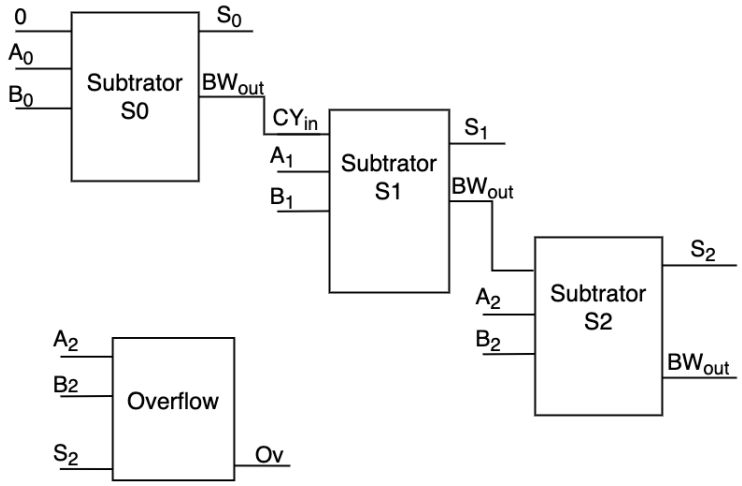


Figura 8: Abordagem modular do Subtrator

4.2.4 Tabela de verdade Somador - S e Carry out

Cyin	A	B	S	Cyout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Figura 9: Tabela de verdade de S e Carry out

4.2.5 Mapas de Karnaugh Somador - S e Carry out

S:

Cyin \ A,B	00	01	11	10
0	0	1	0	1
1	1	0	1	0

Figura 10: Mapa de Karnaugh - S

$$S = A \oplus B \oplus Cyin$$

Cyout:

A \ B,Cyin	00	01	11	10
0	0	0	1	0
1	0	1	1	1

Figura 11: Mapa de Karnaugh - Cyout

$$Cyout = A \cdot B + B \cdot Cyin + A \cdot Cyin$$

4.2.6 Tabela de verdade Subtrator - S e Borrow out

Bwin	A	B	S	Bwout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	1

Figura 12: Tabela de verdade de S e Borrow out

4.2.7 Mapas de Karnaugh Subtrator - S e Borrow out

S:

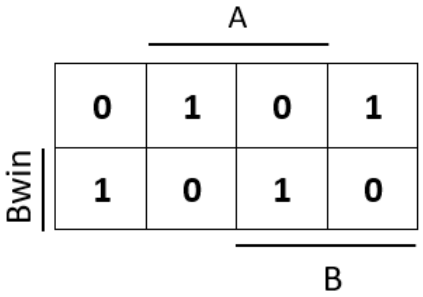


Figura 13: Mapa de Karnaugh - S

Bwout:

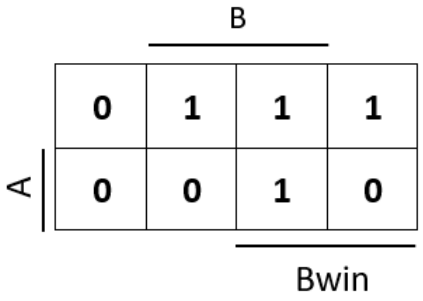


Figura 14: Mapa de Karnaugh - Cyout

$S = A \oplus B \oplus Bwin$

$Cyout = A/ \cdot B + B \cdot Bwin + A/ \cdot Bwin$

Devido às semelhanças das expressões lógicas obtidas através dos mapas de Karnaugh concluímos que é possível fazer um somador e subtrator na mesma máquina. A nossa entrada select, que permite a troca entre a soma e subtracção, vai diferenciar os resultados de BWout e CYout. A expressão de S vai ser igual para ambos os casos.

4.2.8 Tabela de verdade Overflow

Sel	A2	B2	S2	OV
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Figura 15: Tabela de verdade de Overflow

4.2.9 Mapa de Karnaugh Overflow

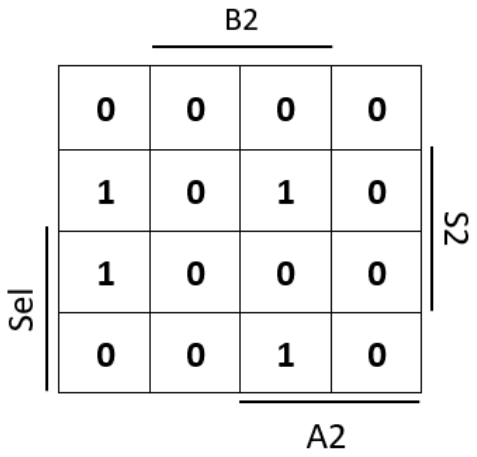


Figura 16: Mapa de Karnaugh de Overflow

$$S = S2 \cdot A2/ \cdot B2/ + S2 \cdot B2 \cdot A2 \cdot Sel/ + Sel \cdot A2 \cdot B2 \cdot S2/$$

4.2.10 Circuito lógico simplificado

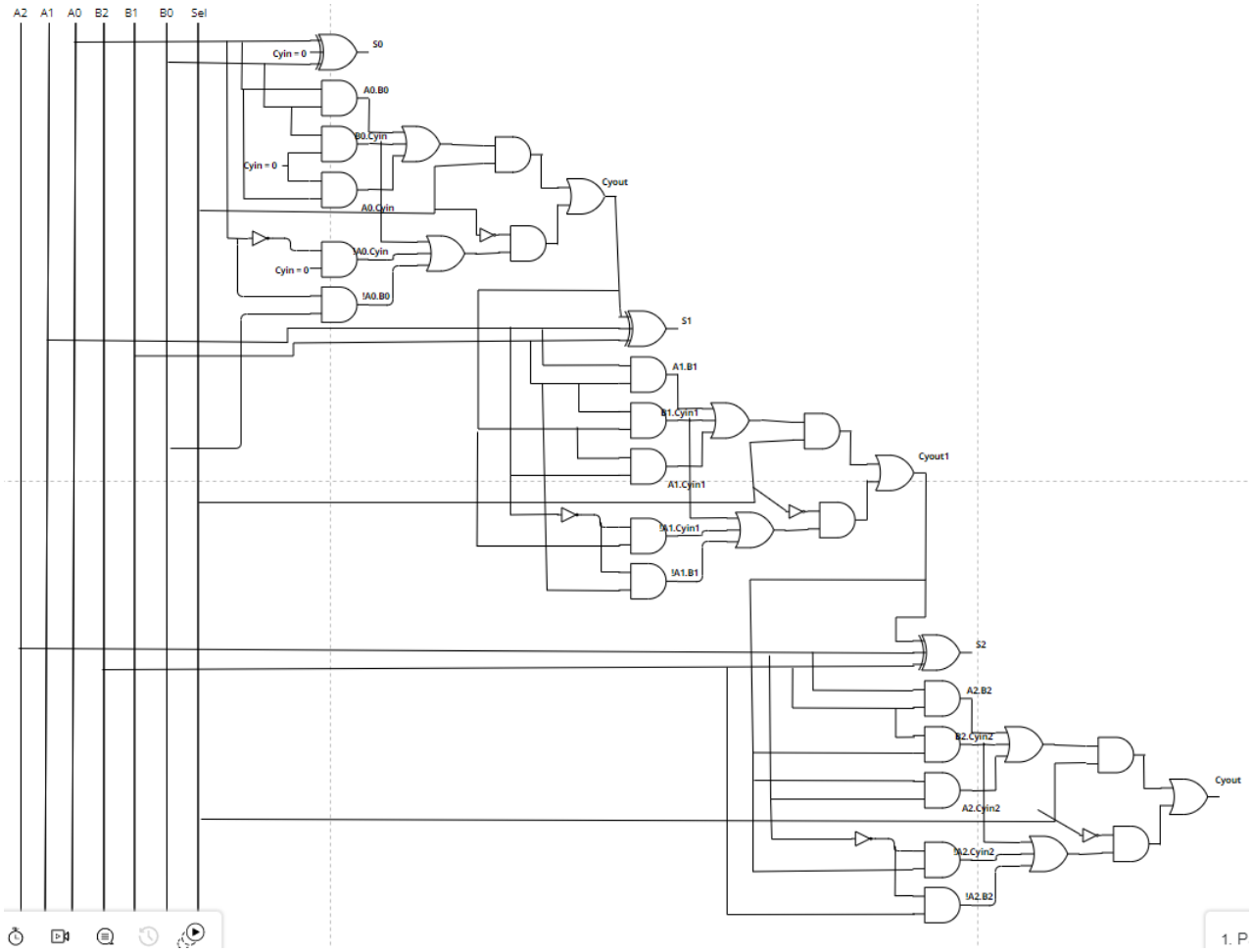


Figura 17: Circuito lógico simplificado Somador/Subtrator

4.2.11 Situação de testes e resultados obtidos

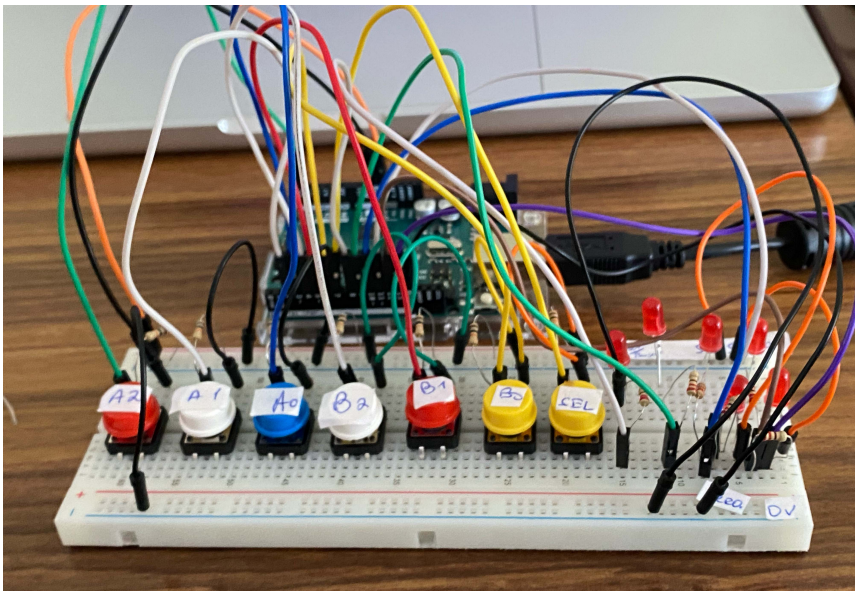


Figura 18: Montagem física do Somador/Subtrator

As contas da soma e subtração de 3 bits têm que ser feitas bit a bit. Ou seja, efectua-se a soma/subtração do bit de menor peso (com o Cyin/Bwin a false), logo a seguir efectua-se a operação do próximo bit, mas tendo em consideração que o Cyout do primeiro bit, entra no Cyin do segundo bit, e assim, sucessivamente até 3 bits.

A máquina tem 7 botões N.O. como inputs (A2,A1,A0,B2,B1,B0 e SEL),6 LEDs vermelhos (Cyout/Bwout, S2, S1, S0, Overflow, Zero). Cada botão vem equipado com uma resistência de 10k Ω e cada LED com uma de 220Ω.

O Select vai definir se estamos a efectuar uma soma ou subtração. Os botões A2, A1, A0 e B2, B1, B0 são, respetivamente, os bits de cada número para o qual queremos efectuar a operação. Os LEDS S2, S1, S0 são os resultados dos bits mencionados acima. (Ver exemplo abaixo)

$$\begin{array}{rr} & A2\ A1\ A0 \\ \pm & B2\ B1\ B0 \\ \hline Cy/Bw & S2\ S1\ S0 \end{array}$$

Figura 19: Exemplo de uma soma/subtração

O Overflow acende quando temos erro na conta, por exemplo, uma soma de 2 números positivos nunca pode dar um resultado negativo. Caso o resultado for negativo, o Overflow acende.

Efectuamos inúmeros testes de somas e subtrações, seja números positivos/-negativos, somas/subtrações. Fomos à procura de erros para testar o overflow, e efectuamos cálculos, cujo resultado é zero, para testarmos o Zero.

4.3 Contador Binário

4.3.1 Enunciado

Projete um contador binário configurável, que através de uma variável externa, realiza uma contagem crescente (Up) ou decrescente (Down). O conjunto de configurações de contagem que este contador deve fornecer na sua saída é o seguinte: [-1, -3, -5, -7].

4.3.2 Modelo Entradas e Saídas

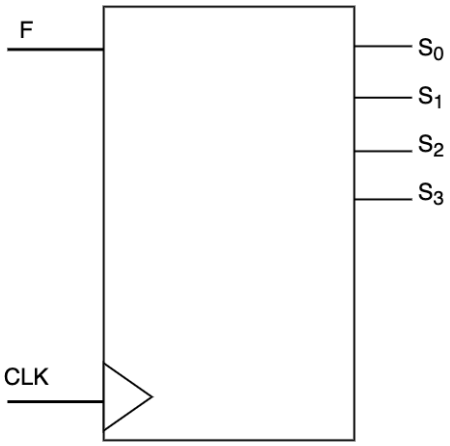


Figura 20: Modelo Entradas e Saídas — Contador

As nossas entradas são: a variável F que vai ter o propósito de selecionar entre a contagem crescente e decrescente, e o Clock que vai fazer a contagem. As saídas S0 a S3 representam os nossos 4 algarismos.

Desenhe modelo de Moore-Mealey, explicitando os blocos da função de estado seguinte (FES) e da função de saída (FS)

4.3.3 Modelo Moore-Mealey

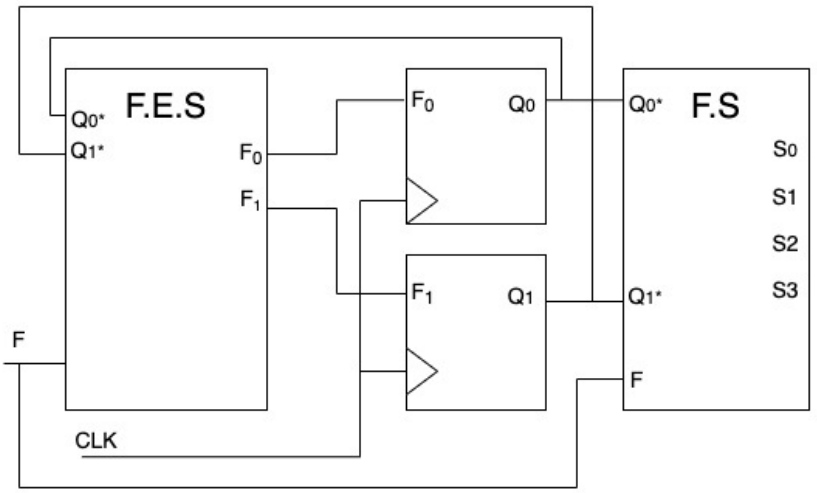


Figura 21: Modelo Entradas e Saídas — Contador

4.3.4 Tabelas de Verdade - F.S e F.E.S

4.3.5 Função de Saída

F	Q1*	Q0*	S3	S2	S1	S0
0	0	0	1	0	1	1
0	0	1	1	0	0	1
0	1	0	1	1	1	1
0	1	1	1	1	0	1
1	0	0	1	1	1	1
1	0	1	1	1	0	1
1	1	0	1	0	1	1
1	1	0	1	0	0	1

Figura 22: Tabela de Verdade — Função FS

4.3.6 Mapas de Karnaugh

S2:

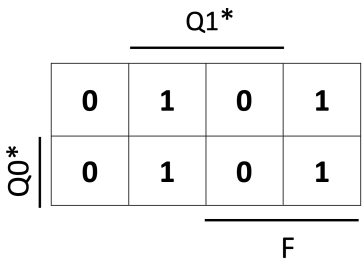


Figura 23: Mapa de Karnaugh-S2

S1:

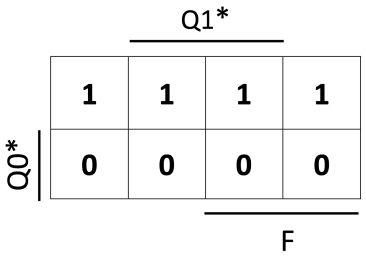


Figura 24: Mapa de Karnaugh-S1

$S2 = F/ \cdot Q1 + F/ \cdot Q1$ $S1 = Q0/$

A partir do modelo Moore-Mealey fizemos as tabelas de verdade que vão definir o comportamento do nosso contador. A F.S (função de saída) vai ser a função que, a partir do código de estado em que nos encontramos, vai definir quais das saídas estão ativas.

4.3.7 Função de Estado Seguinte

F	Q1*	Q0*	F0	F1
0	0	0	1	1
0	0	1	0	0
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	1
1	1	1	0	0

Figura 25: Tabela de Verdade — Função FES

4.3.8 Mapas de Karnaugh

F1:

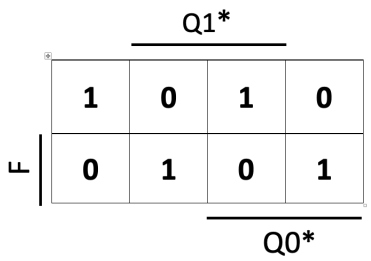


Figura 26: Mapa de Karnaugh-F1

F0:

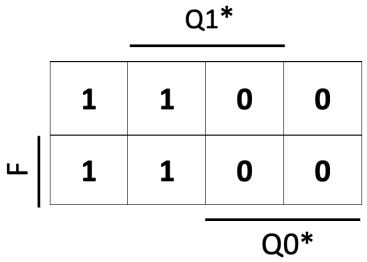


Figura 27: Mapa de Karnaugh-F0

$F1 = F \oplus Q1 \oplus Q0$ $F0 = Q0/$

A F.E.S. (função de estado seguinte) vai definir qual vai ser o nosso próximo estado, consoante o input da nossa variável F que funciona como um select entre contagem crescente e decrescente. Os valores que F1 e F0 vão tomar formam os códigos do estado.

A partir das tabelas de verdade utilizamos os Mapas de Karnaugh para obter a expressão simplificada de cada função.

4.3.9 A.S.M.

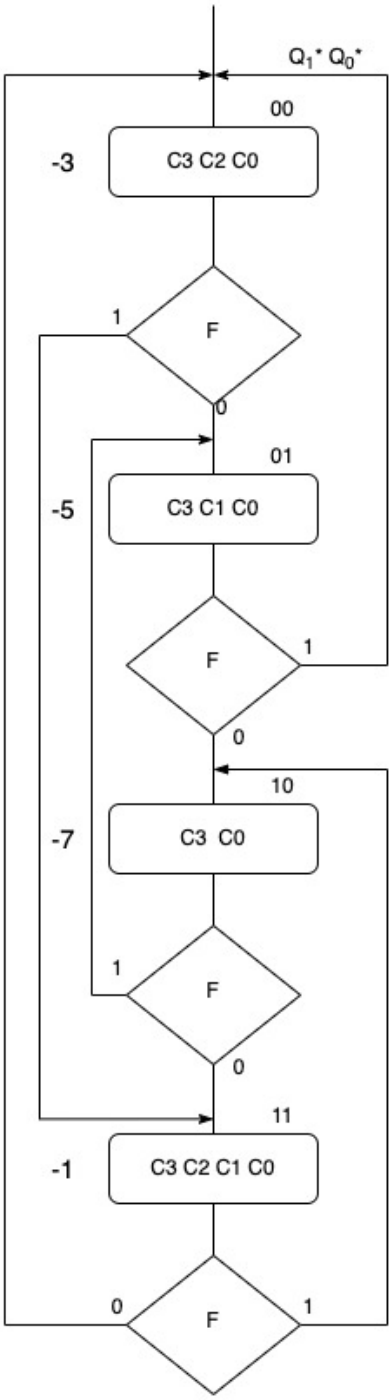


Figura 28: Modelo Entradas e Saídas — Contador

O modelo A.S.M. (Algorithmic State Machine) vai definir o comportamento da nossa maquina. Temos os estados, dentro dos rectângulos, com o seu nome do lado esquerdo, no topo superior direito temos o código que nos permite distinguir entre os vários estados e no seu interior temos as saídas que se encontram ativas. Nos losangos temos as condições, neste caso estamos a fazer a leitura da variável F. Caso F esteja a 0 a contagem decorre do modo decrescente, quando F se encontra a 1 um a contagem passa a crescente.

Desenho do circuito lógico simplificado

4.3.10 Circuito Lógico Simplificado

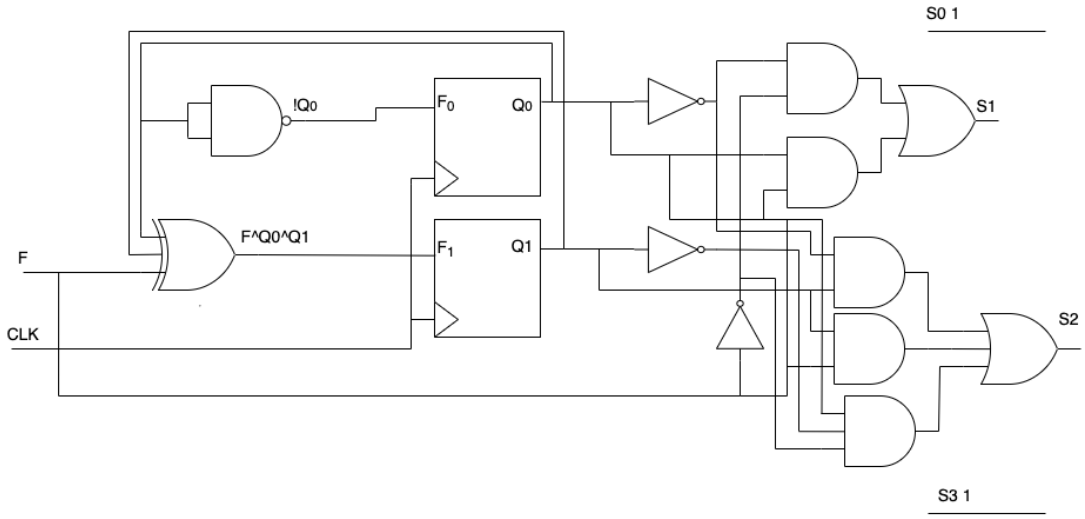


Figura 29: Modelo Entradas e Saídas — Contador

4.3.11 Situações de Testes e os Resultados Obtidos

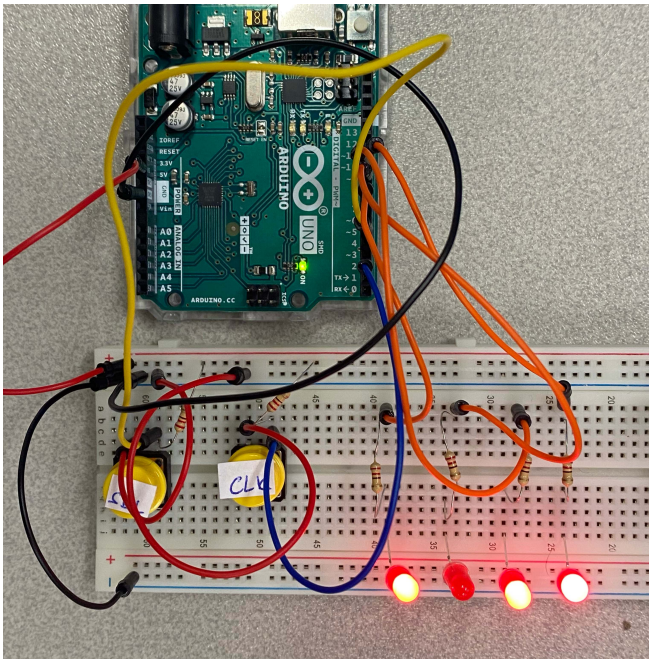


Figura 30: Montagem no Arduino — Contador

O funcionamento do contador é o esperado, quando se activa no botão que simula o clock vemos a contagem crescente representada nos atuadores LED ($1001 - 1011 - 1101 - 1111$). Quando se activa o sensor de select a contagem passa é feita do modo decrescente ($1111 - 1011 - 1101 - 1001$).

5 Conclusões

O objectivo deste primeiro trabalho prático é aprender a trabalhar com funções lógicas, utilizando o microcontrolador Arduíno.

O primeiro exercício foi a projeção de um circuito lógico que detecta avarias numa porta lógica 'AND' ou 'OR'. Tem 4 entradas (A,B,AO,F) e uma saída (Avaria). A seleção AO, serve para o utilizador selecionar, se pretender testar o 'AND' ou o 'OR'. O circuito lógico simplificado requer 6 'ANDs' de 3 entradas, e 5 'ORs' de 2 entradas. Foi uma boa experiência de introdução às funções lógicas e não ocorreram grandes problemas neste exercício.

O segundo exercício foi a projeção de um somador/subtrator de 3 bits. A operação é efectuada bit a bit. Todos os inputs e outputs têm de ser lidos através de Código Complemento, com o bit de maior força sendo o sinal (+ ou -). Ao efectuarmos a montagem do circuito no Arduíno, deparamos nos com alguns problemas que não tínhamos no mundo ideal da simulação em Tinkercad. Foi uma situação de aprendizagem importante que nos permite perceber as diferenças que há entre os dois mundos e a importância de testar fisicamente as nossas máquinas.

O último exercício foi a projeção de um contador binário crescente e decrescente. Foi nos introduzido os Sistemas Sequenciais e os flip-flops que são células de memória capazes de armazenar estados. No circuito estão presentes 2 flip flops do tipo D edge triggered. Para que haja sincronismo entre eles, introduzimos um Clock. Neste caso o nosso Clock é simulado por um botão. A representação dos bits, dado que são números negativos foi feita em Código Complemento. A montagem em física correu bem.

Finalmente, os trabalhos laboratoriais foram um sucesso, pois todos os componentes funcionaram como esperado.

6 Bibliografia

Para a realização deste projecto apenas utilizamos o material fornecido pela Unidade Curricular

- Folhas de Computação Física - Jorge Pais

7 Código Arduíno

7.1 Circuito detetor de Avarias em Portas Lógicas

```
#define A A0
#define B A1
#define F A2
#define AO A3
#define LED 7
bool a,b,f,ao,avaria;           //definição das variáveis de entrada
                                //e saída
void setup()
{
  pinMode(A, INPUT);
  pinMode(B, INPUT);
  pinMode(F, INPUT);
  pinMode(AO, INPUT);
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

void loop()                     //chamada, por ordem, de todos os métodos
{
  lerEntradas();
  funcaoDependente();
  ativarSaidas();
}

void lerEntradas(){             //leitura das entradas
  a=digitalRead(A);
  b=digitalRead(B);
  f=digitalRead(F);
  ao=digitalRead(AO);
}

void funcaoDependente(){        //escrita da expressão lógica da saída
                                //'avaria' obtida nos mapas de karnaugh

   avaria = (a && !f && !ao) || (a && b && !f) ||
        (b && !ao && !f) || (!a && ao && f)||
        (!a && f && !b)  || (f && ao && !b);
}

void ativarSaidas(){            //ativação da saída em função de
  digitalWrite(LED,avaria);     //'avaria'
}
```

7.2 Somador/Subtrator

```
#define A2 A5
#define A1 2
#define A0 3
#define B2 4
#define B1 5
#define B0 6
#define SEL 7
#define LED_Cyout 8
#define LED_S2 9
#define LED_S1 10
#define LED_S0 13
#define LED_Flag 12
#define LED_Zero 11
bool a2,a1,a0,b2,b1,b0,cyout,s2,s1,s0,zero,sel,flag,cyin;

//defenição das variáveis de entrada e saída

void setup()
{
  //pinMode(A2, INPUT);
  pinMode(A1, INPUT);
  pinMode(A0, INPUT);
  pinMode(B2, INPUT);
  pinMode(B1, INPUT);
  pinMode(B0, INPUT);
  pinMode(SEL, INPUT);
  pinMode(LED_Cyout, OUTPUT);
  pinMode(LED_S2, OUTPUT);
  pinMode(LED_S1, OUTPUT);
  pinMode(LED_S0, OUTPUT);
  pinMode(LED_Flag, OUTPUT);
  pinMode(LED_Zero, OUTPUT);
  Serial.begin(9600);

}

void loop()                 //chamada, por ordem, de todos os métodos
{
  lerEntradas();
  somador3bits(a2,a1,a0,b2,b1,b0,false,&cyout,&s2,&s1,&s0);
  zer0();
  escreverSaidas();
}


void lerEntradas(){         //leitura das entradas
  a2=analogRead(A2);
  a1=digitalRead(A1);
  a0=digitalRead(A0);
```

```
  b2=digitalRead(B2);
  b1=digitalRead(B1);
  b0=digitalRead(B0);
  sel=digitalRead(SEL);
}

void escreverSaidas(){      //escrita das saídas em função das funções
                            //de saída da máquina

  digitalWrite(LED_Cyout,cyout);
  Serial.println(cyout);
  digitalWrite(LED_S2,s2);
  digitalWrite(LED_S1,s1);
  digitalWrite(LED_S0,s0);
  digitalWrite(LED_Flag,flag);
  digitalWrite(LED_Zero,zero);

}

void somador1bit(bool cyin, bool a, bool b, bool *cyout, bool *s){

//escrita das funções lógicas 'cyout' e 's' obtidas nos mapas de karnaugh

 *cyout=(((b && cyin) || (a && cyin) || (a && b)) && sel)
        || (((b && cyin) || (!a && cyin) || (!a && b)) && !sel);

 *s= a^b^cyin;
}

void somador3bits(bool a2, bool a1, bool a0, bool b2, bool b1, bool b0,
                  bool cyin, bool *cyout, bool *s2, bool *s1, bool *s0){

  bool cy0,cy1;

  //chamada do método somador1bit 3 vezes

  somador1bit(false,a0,b0,&cy0,s0);
  somador1bit(cy0,a1,b1,&cy1,s1);
  somador1bit(cy1,a2,b2,cyout,s2);

}
void zer0(){

//escrita das funções lógicas 'zero' e 'flag' obtidas nos mapas de karnaugh

    zero =  !s2 && !s1 && !s0 && !cyout;
    flag =  (s2 && !a2 && !b2) ||
            (s2 && a2 && b2 && !sel) ||
            (sel && a2 && b2 && !s2);
  }
```

7.3 Contador Binário

```
#define tRuido 100
#define f 5
#define LED_S3 8
#define LED_S2 9
#define LED_S1 10
#define LED_S0 11
bool F;                              //F=0 crescente F=1 decrescente
                                     //botao pressionado é 0
bool D1,D0,Q1,Q0;
bool S3,S2,S1,S0;                    //saidas
long unsigned int t0,t=millis();     //inicialização do tempo

void setup(){
    pinMode(f, INPUT);
    pinMode(LED_S3, OUTPUT);
    pinMode(LED_S2, OUTPUT);
    pinMode(LED_S1, OUTPUT);
    pinMode(LED_S0, OUTPUT);
    Serial.begin(9600);
  attachInterrupt(0, circuitoSequencial, RISING);

    // o @param1 pode ser 0 ou 1 (pin D2, D3)
    // quando o @param3 for RISING (ascendente),
    //executa o @param2 circuitoSequencial (é onde chamamos os flipflops)

  interrupts();
}
void loop () {                       //chamada, por ordem, de todos
                                     //os métodos
  lerEntradas();
  noInterrupts();
  circuitoCombinatorio();
  interrupts();
  escreverSaidas();

}

bool filtro(){                       //caso o tempo já tenha passado o tRuido
  t0=t;                              //devolve true
  t=millis();
  return ((t - t0) > tRuido);

}

void circuitoSequencial(){           //chamada dos métodos flip flop
  if(filtro()){                      //filtro do ruido
  flipflopD(D1, &Q1);
  flipflopD(D0, &Q0);
  }
```

```
}


void circuitoCombinatorio(){        //Chamada da Função Estado Seguinte,
  FES();                            //e da Função saída
  FS();
}

void flipflopD(bool D, bool *Q){    //Armazenamento da entrada
  *Q=D;
}

void FES(){                         //Função Estado Seguinte
  D1= F^Q1^Q0;                      //escrita das expressões lógicas
  D0= !Q0;                          //obtidas nos mapas de karnaugh


}

void FS(){                          //Função Estado Saída
                                    //escrita das expressões lógicas
  S3 = 1;                           //obtidas nos mapas de karnaugh
  S2 = (!F && Q1) ||(F && !Q1);
  S1 = !Q0;
  S0 = 1;

}

void lerEntradas(){                 //leitura da entrada
  F=digitalRead(f);
}

void escreverSaidas(){              //escrita das saídas em função das
                                    //funções de saída do contador
  digitalWrite(LED_S3,S3);
  digitalWrite(LED_S2,S2);
  digitalWrite(LED_S1,S1);
  digitalWrite(LED_S0,S0);

}
```