ISEL
ADEETC
Área Departamental de Engenharia Electrónica e de Telecomunicações e de Computadores

Licenciatura em Engenharia Informática e de Computadores

Jogo da Roleta

Keyboard Reader
(Roulette Game)

A46378 Berto Barata
A44787 Gonçalo Garcia
A44776 João Gomes

Professores:
Pedro Miguens Matutino (pedro.miguens@isel.pt)
Nuno Sebastião (nuno.sebastiao@isel.pt)

Projeto de
Laboratório de Informática e Computadores
2020 / 2021 inverno

22 de outubro de 2020

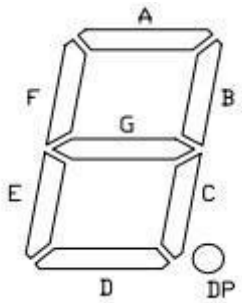
Índice

Introdução

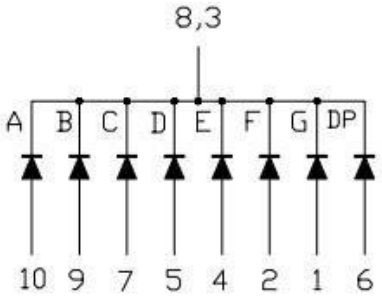
O projeto semestral desta unidade curricular como já apresentado anteriormente consiste no desenvolvimento de um jogo da roleta. O seu modo de jogo também foi explicado anteriormente.
O sistema que implementa o jogo será constituído por um computador (módulo de controlo), um teclado de doze teclas, um moedeiro, um mostrador *LCD* de duas linhas com dezasseis caracteres cada, um mostrador da roleta e uma chave de manutenção (para colocar o sistema em modo de manutenção).
O módulo ***Roullette Display*** é precisamente o mostrador da roleta que depois de realizar uma animação a rodar, depois de serem efetuadas todas as apostas, acaba por apresentar um valor. É importante mencionar que esta animação é não bloqueante daí a existência de 2 métodos (*startAnimation* e *animation*).
O módulo utilizado neste projeto é um «display de 7 segmentos» e o dispositivo utilizado é ***RD47seg***

Display

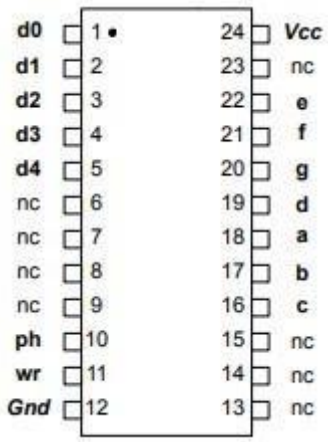
O módulo ***Roullette Display*** é composto pelo ***RD47seq*** e por uma ***ATF750C*** programada para se poder controlar o display de forma programada posteriormente em JAVA.
O ***RD47seq*** aceita cinco linhas de dados de entrada (d0.. 4), que é armazenado num registo interno em cada borda ascendente do sinal do ***wr***. Os dados descodificados estão ligados aos 7 pinos de saída (a g) que conduzir os LEDs do ecrã de 7 segmentos. A fase (ph) entrada é usado para inverter a fase de tabela de funções.



a) *RD47seg*



b) Diagrama do Circuito Interno



c) PIN Description de ***ATF750C***

Figura 1 – Bloco ***Roullette Display***

Outputs

As saídas de cada registo são descodificadas como mostrado na figura abaixo e apresentados no display.

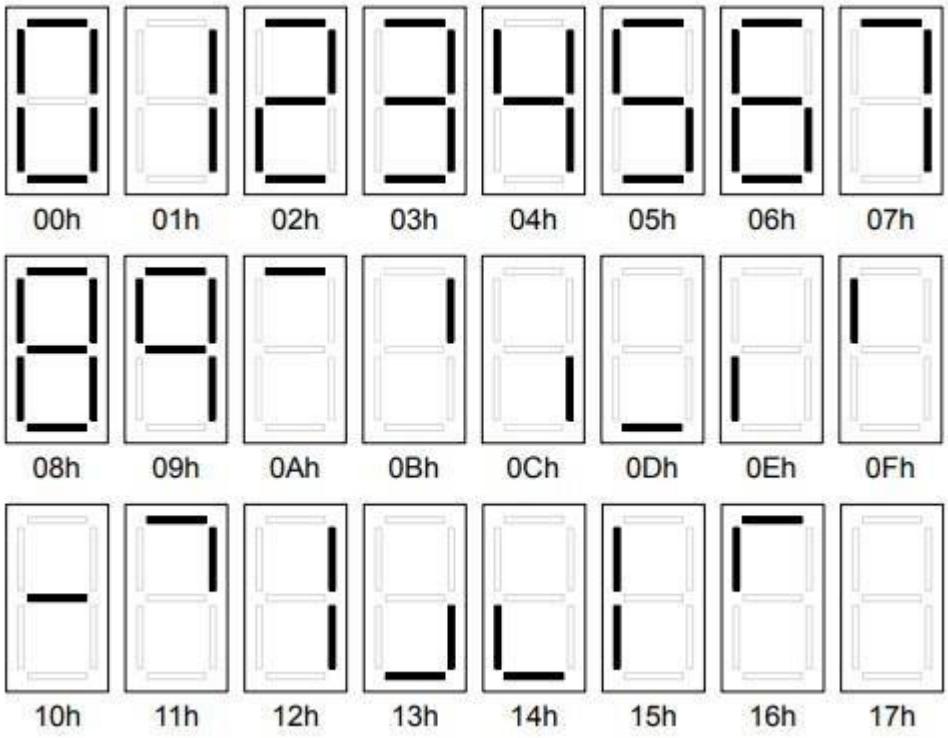
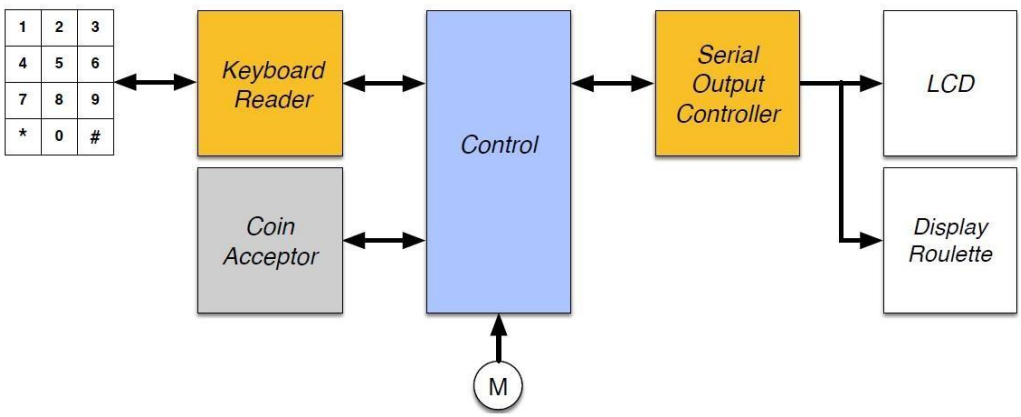


Figura 2 – Caracteres possíveis de apresentar no ***RD47seg*** e respetivos códigos

Interface

Implementou-se o módulo *Roulette Display* em *software*, recorrendo a linguagem Java.
Nesta fase do trabalho, realizou-se apenas uma das classes: ***Roulette Display.***



Classe RouletteDisplay

A essência desta classe reside no facto de ela criar uma animação não bloqueante, daí o facto de existirem 2 metodos diferentes em que quando queremos fazer a animação (apenas o rodar da roleta) faz-se o *startAnimation* e depois dentro de um *loop while* chama-se diversas vezes o método animation deixando assim de ser bloqueante pois podemos correr outros métodos dentro do mesmo *loop while.* Desta forma não estamos apenas a fazer percorrer a animação roleta.

Conclusões

Os objetivos solicitados foram alcançados. Percorrendo todos os passos enumerados neste relatório, desenvolvemos os conhecimentos previamente adquiridos sobre lógica e sistemas digitais, descrevendo hardware e vários módulos funcionais. Aprendemos ainda a elaborar esquemas elétricos e melhorámos as nossas capacidades de programação em Java. Algumas etapas foram mais trabalhosas que outras, nomeadamente a análise e compreensão descrições hardware que, tal como muitas outras partes deste trabalho, passaram por várias versões e processos de simplificação e interpretação.

A.1. Descrição CUPL do bloco Roulette Display

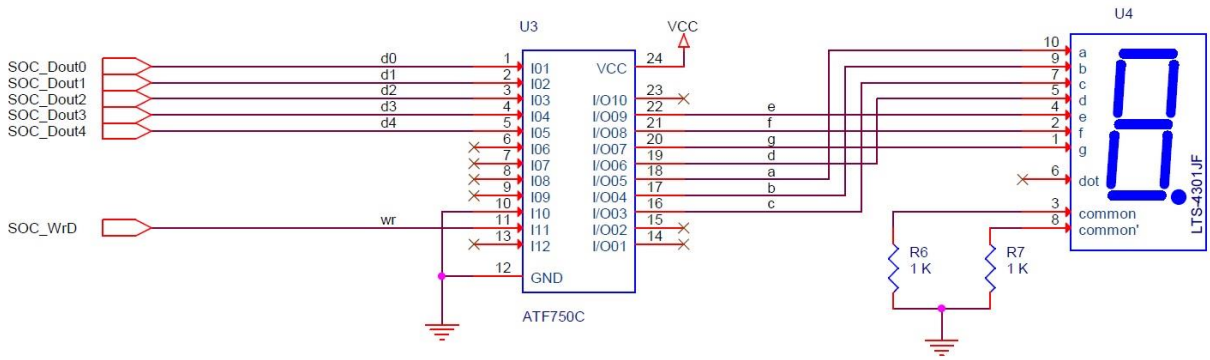
```
Name     rouletteDisplay ;
PartNo   00 ;
Date     14-10-2009 ;
Revision 01 ;
Designer Engineer ;
Company  CCISEL ;
Assembly None ;
Location  ;
Device    v750c ;

/* *************** INPUT PINS *********************/
PIN  [1..5] = [d0..4]; pin  10 = ph; pin  11
= wr;
/* *************** OUTPUT PINS *********************/ PIN
[18,17,16,19,22,21,20] = [a,b,c,d,e,f,g]; pinnode
[14,15,25,26,35] = [q0..4];

[q0..4].d = [d0..4];
[q0..4].ck = wr;
[q0..4].ar ='b'0;
[q0..4].sp ='b'0;

field number =[q0..4];
field segments = [ina,inb,inc,ind,ine,inf,ing]; table
number => segments{
0=>'b'1111110; 4=>'b'0110011; 8=>'b'1111111; C=>'b'0010000;
1=>'b'0110000; 5=>'b'1011011; 9=>'b'1111011; D=>'b'0001000;
2=>'b'1101101; 6=>'b'1011111; A=>'b'1000000; E=>'b'0000100;
3=>'b'1111001; 7=>'b'1110000; B=>'b'0100000; F=>'b'0000010;
10=>'b'0000001; 12=>'b'0110000; 14=>'b'0001100; 16=>'b'1000010;
11=>'b'1100000; 13=>'b'0011000; 15=>'b'0000110; 17=>'b'0000000;
}   a = ina $ ph; b
= inb $ ph; c = inc
$ ph; d = ind $ ph;
e = ine $ ph; f =
inf $
ph; g = ing $
ph;
```

A.2. Esquema elétrico do módulo *Roulette Display*



A.3. Código Java da classe Roulette Display

```
public class RouletteDisplay {
    private static final int CLEAR_DISPLAY = 0x17;
    private static final char ANIMATION[] = {0x0A, 0x0B, 0x0C, 0x0D, 0x0E,
0x0F};
    private static final int ANIMATION_MAX_INDEX = ANIMATION.length -
1;     private static final long ANIMATION_TIMEOUT_MS = 300;
private static final boolean SERIAL_INTERFACE = true;     private
static final int DATA_MASK_OUT_PORT = 0x1F;     private static final
int WR_MASK_OUT_PORT = 0x40; // Write.
    private static int animationIndex;
private static long timeout;
    public static void main(String[] args) {
        HAL.init();

SerialEmitter.init();
init();     }
    public static void init() {
clear();
    }
    public static void clear() {
showNumber(CLEAR_DISPLAY);
    }
    private static void showNumberParallel(int number) {
        HAL.writeBits(DATA_MASK_OUT_PORT, number);
        HAL.setBits(WR_MASK_OUT_PORT);
        HAL.clrBits(WR_MASK_OUT_PORT);
    }
    private static void showNumberSerial(int number) {
        SerialEmitter.send(SerialEmitter.Destination.RDisplay, number);
    }
    public static void showNumber(int number) {
if (SERIAL_INTERFACE) {
showNumberSerial(number);
        } else {
            showNumberParallel(number);

}     }
    public static void startAnimation() {
animationIndex = 0;
        showNumber(ANIMATION[animationIndex]);
        timeout = Time.getTimeInMillis() + ANIMATION_TIMEOUT_MS;
    }
    public static void animation() {         if (timeout <
Time.getTimeInMillis()) {             animationIndex =
animationIndex < ANIMATION_MAX_INDEX ?
animationIndex + 1 : 0;
            showNumber(ANIMATION[animationIndex]);
timeout += ANIMATION_TIMEOUT_MS;
        }
    }
}
```