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

Licenciatura em Engenharia Informática e de Computadores

Jogo da Roleta

LCD

(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 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).
Nesta apresentação iremos explicar o funcionamente e desenvolvimento do módulo LCD. Este LCD é um mostrador de duas linhas com dezasseis caracteres cada, que apresenta os créditos obtidos numa jogada como também o saldo restante do jogador.
É Interessante menciona que tanto no LCD como no RoulletteDisplay estamos apenas a enviar informação em paralelo e não em série algo que ocorrerá apenas na ultima fase do trabalho.

LCD

O módulo LCD é uma estrutura independente contralado pelo módulo de *Control*, mais própriamente pelo módulo *Serial Output Controller*, conforme ilustrado na Figura 1. Neste caso o módulo LCD, possui uma classe própria do mesmo nome implementada em software. Nesta classe LCD, a função *init* acaba por ser a mais relevante visto que ao olhar e interiorizar o datasheet deste aparelho realizou-se uma série de códigos com o objetivo da sua fácil utilização. Implementa-se também funções que nos permitem escrever, alterar e remover informação deste dispositivo, bem como a colocação destas mesmas mensagens.

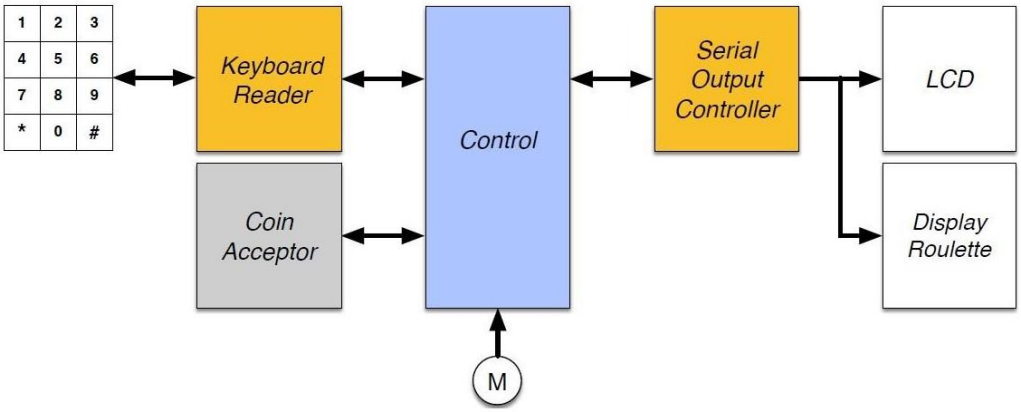


Figura 1 – Diagrama de blocos do módulo de implementação de Roulette Game.

Interface com o *Control*

Implementou-se o módulo *LCD* em *software*, recorrendo a linguagem Java e seguindo a arquitetura lógica apresentada na Figura 2 .

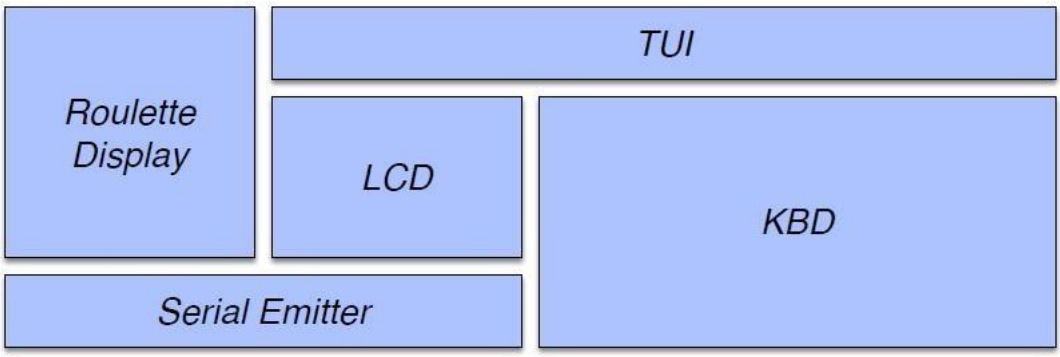
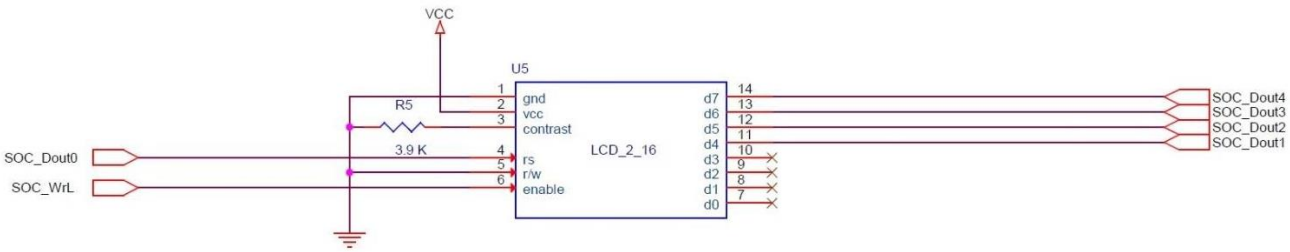


Figura 2 – Diagrama lógico do módulo *Control* de interface com o módulo

Conclusões

Os objetivos solicitados foram alcançados. Percorrendo todos os passos necessário e imprescindíveis para a conclusão com sucesso deste módulo. Melhorámos as nossas capacidades de programação em Java bem como na elaboração de esquemas elétricos.
Algumas etapas foram mais trabalhosas que outras, nomeadamente a análise e compreensão da classe ***init*** que passou por várias versões e processos de simplificação.

A.1. Esquema elétrico do módulo *LCD*



A.2. Código Java da classe *LCD*

```
import isel.leic.utils.Time;
// Liquid Cristal Display.
public class LCD {
    public static final int LINES = 2, COLS = 16;     private static final boolean
SERIAL_INTERFACE = true;     private static final int DATA_MASK_OUT_PORT = 0x0F;
private static final int RS_MASK_OUT_PORT = 0x10; // Register Select.     private
static final int E_MASK_OUT_PORT = 0x20; // Enable.
    public static void main(String[] args) {
        HAL.init();
        SerialEmitter.init();         init();
    }
    private static void writeNibbleParallel(boolean rs, int data) {         if (rs) {
            HAL.setBits(RS_MASK_OUT_PORT);
        } else {
            HAL.clrBits(RS_MASK_OUT_PORT);
        }
        HAL.setBits(E_MASK_OUT_PORT);
        HAL.writeBits(DATA_MASK_OUT_PORT, data);
        HAL.clrBits(E_MASK_OUT_PORT);
    }
    private static void writeNibbleSerial(boolean rs, int data) {
        SerialEmitter.send(SerialEmitter.Destination.LCD, data << 1 | (rs ? 1 :
0));
    }
    private static void writeNibble(boolean rs, int data) {         if
(SERIAL_INTERFACE) {             writeNibbleSerial(rs, data);
        } else {
            writeNibbleParallel(rs, data);
        }     }
    private static void writeByte(boolean rs, int data) {         writeNibble(rs,
data >> 4);         writeNibble(rs, data);
    }
    private static void writeCMD(int data) {         writeByte(false, data);
    }
    private static void writeDATA(int data) {         writeByte(true, data);
    }
    public static void init() {         writeNibble(false, 3);
        Time.sleep(5); // // 4,1 ms until next write.
        writeNibble(false, 3);
        Time.sleep(1); // 100 us until next write.
        writeNibble(false, 3);         writeNibble(false, 2);         writeCMD(0x2C);
writeCMD(0x08);         writeCMD(0x01);         writeCMD(0x06);
writeCMD(0x0F);         writeCMD(0x0C); // Hide the cursor.
    }
    public static void write(char c) {         writeDATA(c);
    }
```

```
    public static void write(String txt) {          for (int i = 0; i < txt.length();
i++) {              writeDATA(txt.charAt(i));
        }
    }
    // Cells: 0-7.     public static void setCustomChar(int cell, int c[]) {
writeCMD(0x40 + (cell * 8)); // Set the CGRAM address.
        for (int i : c) {              writeDATA(i);
        }
    }
    // Lines and columns start at 1.
    public static void cursor(int line, int col) {
        writeCMD(0x80 | (1 == line ? col - 1 : col - 1 + 0x40));
    }
    public static void clear() {          writeCMD(0x01);
    }
}
```