Introdução a Engenharia Elétrica - 323100

Aula S12

Comunicação serial, Acelerômetro, Display LCD

Escola Politécnica da Universidade de São Paulo

Departamentos da Engenharia Elétrica

PCS Computação e Sistemas Digitais

PEA Energia e Automação Elétricas

PSI Sistemas Eletrônicos

PTC Telecomunicações e Controle

V1.4

Outubro de 2014

Comunicação Serial - revisão

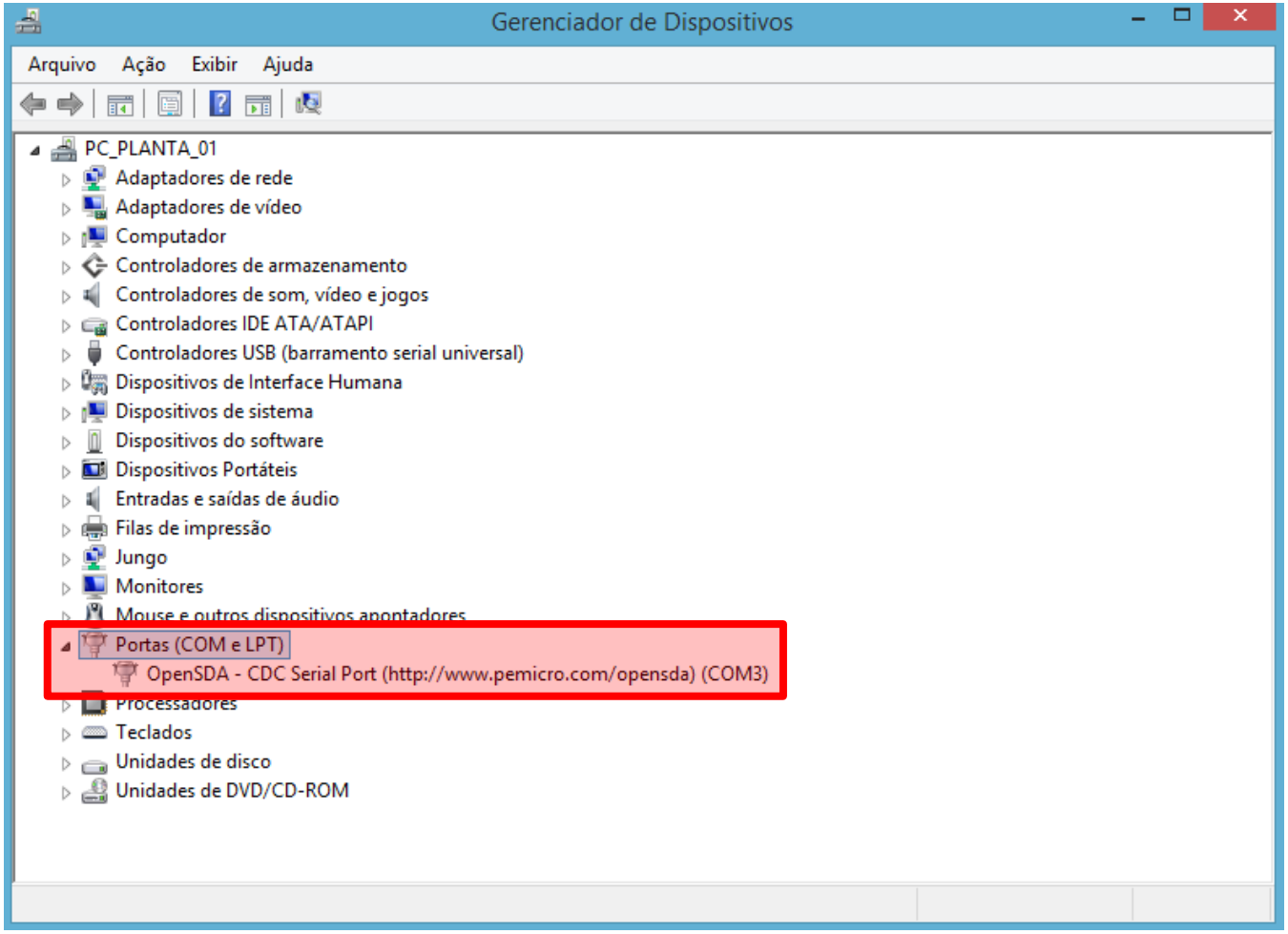
- Comunicação com um PC host através de um "USB Virtual Serial Port" através do mesmo cabo USB que é usado para a programação. Isso permite que você:
  - Imprima mensagens para um terminal host PC (útil para depuração!)
  - Leia a entrada do teclado do PC host
  - Comunique-se com aplicativos e linguagens de programação em execução no PC host que podem se comunicar com uma porta serial, por exemplo, Perl, Python, Java, etc.

No Mbed:

- **`Serial name(USBTX, USBRX)`**: define a comunicação serial com TX e RX via USB DAS. Define o label name.
- **`name.baud(9600)`**: define a taxa de transmissão (baud rate) em 9600 kbps. Aceita outors valotes (19200, 38400, etc).
- **`name.printf`**: imprime *string* KL25Z → PC;
- **`name.putc`**: envia um caractere KL25Z → PC;
- **`name.getc`**: recebe um caractere PC → KL25Z.

Comunicação Serial

- Ao conectar o cabo USB, aparece enumerada no computador uma porta OpenSDA – CDC.

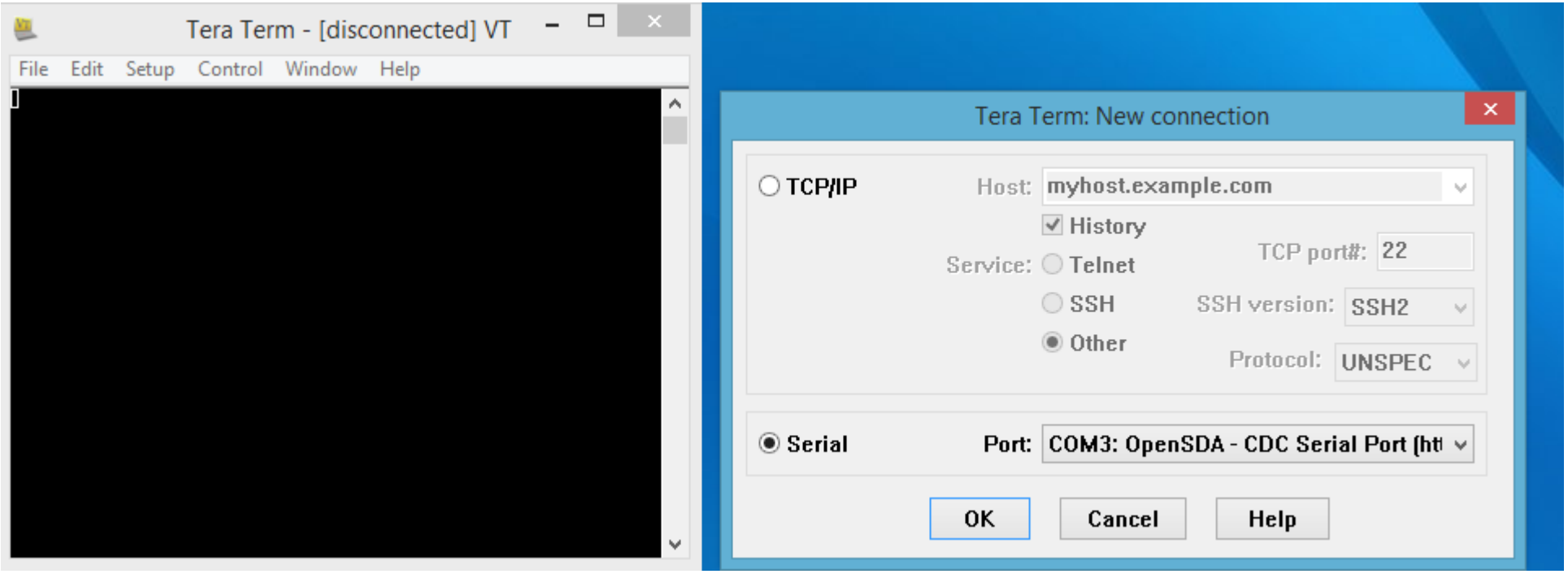


Comunicação Serial

- Caso não apareça no sistema windos, é necessário instalar o driver. Busque por **Download the mbed Windows serial port driver** no *Dashboard* do mbed.
- Para o curso, vamos utilizar um aplicativo emulador de terminal denominado Tera Term (http://ttssh2.sourceforge.jp/index.html.en).

Comunicação Serial

- Escolher a opção serial e a porta enumerada.



Exemplos Básicos no Mbed

- **Hello World!**

```
#include "mbed.h"
Serial pc(USBTX, USBRX); // tx, rx
int main() {
pc.printf("Hello World!\n");
}
```

- ***Papagaio de pirata - Echo back characters you type***

```
#include "mbed.h"
Serial pc(USBTX, USBRX);
int main() {

pc.printf("Echoes back to the screen anything you type\n");
   while(1) {
        pc.putc(pc.getc());
   }
}
```

Outros Exemplos:

- Controla brilho do LED (‘u’ p/ UP e ‘d’ p/ DOWN).

```
#include "mbed.h"
Serial pc(USBTX, USBRX); // tx, rx
PwmOut led(LED1);
int main() {
    char c;
    float brightness = 1.0;
    pc.printf("Press 'u' para aumentar brilho e 'd' para diminuir \n\r");
    while(1) {
        c = pc.getc();
        if((c == 'u') && (brightness > 0.0)) {
            brightness -= 0.1;
            led = brightness;
        }
        if((c == 'd') && (brightness < 1.0)) {
            brightness += 0.1;
            led = brightness;
        }
        pc.printf("Brilho  = %2.0f %% \r", (1.0-led)*100);
     }
}
```

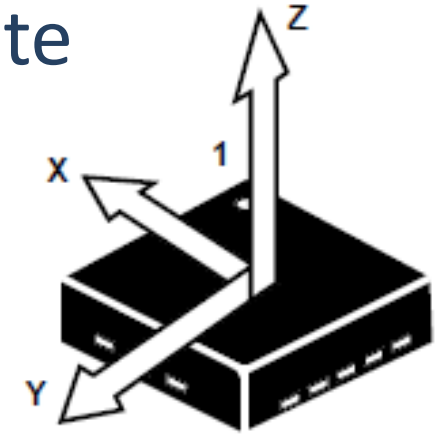
Comunicação Serial

- Muda cor do led RGB (‘r’ p/ RED, ‘g’ p/ GREEN, ‘b’ p/ BLUE. Qualquer outra tecla apaga tudo).

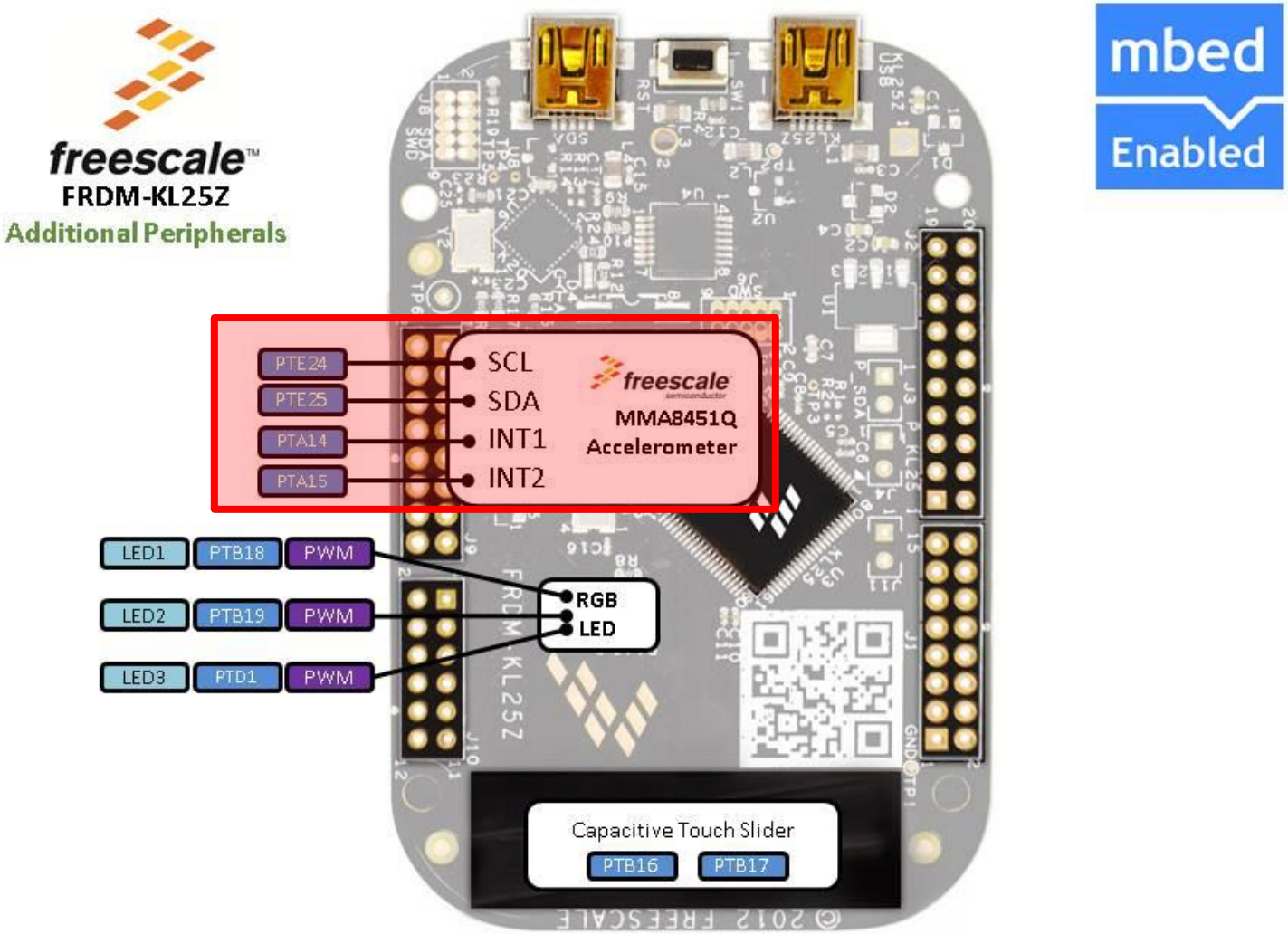
```
#include "mbed.h"
Serial pc(USBTX, USBRX); // tx, rx
DigitalOut r(LED1); DigitalOut g(LED2); DigitalOut b(LED3);
int main() {
    char c;
    r=1; g=1; b=1;
    pc.printf("Pressione a inicial da cor desejada (r ou g ou b)");
    while(1) {
        c = pc.getc();
        switch(c){
        case 'r':
            r=0; g=1; b=1;
            break;
        case 'g':
            r=1; g=0; b=1;
            break;
        case 'b':
            r=1; g=1; b=0;
            break;
        default:
            r=1; g=1; b=1;
            break;
        }   }  }
```

Acelerômetro

- Dispositivo para medição das acelerações às quais está submetido.
- O kit FRDM KL25Z possui um acelerômetro triaxial, do tipo capacitivo, modelo MMA8451Q, capaz de medir a aceleração no eixo X, Y e Z, ortogonais ao chip.
- O MMA8451Q é conectado ao KL05Z via interface de comunicação serial I2C, com a seguinte pinagem:
  - SCL = PTE24
  - SDA = PTE25



Pinos no FRDM KL25Z



mbed Enabled

Exemplo:

```
#include "mbed.h"
#include "MMA8451Q.h" //biblioteca do Acelerometro
// Definição dos pinos de acordo com o modelo do KIT
#define SDA PTE25
#define SCL PTE24
#define MMA8451_I2C_ADDRESS (0x1d<<1)

int main(void)
{   //ativa o periférico e aloca os pinos da com. I2C
    MMA8451Q acc(SDA, SCL, MMA8451_I2C_ADDRESS);
    PwmOut rled(LED1);
    PwmOut gled(LED2);
    PwmOut bled(LED3);
    float x, y, z;
    printf("MMA8451 ID: %d\n", acc.getWhoAmI());
    while (true) {
        x = acc.getAccX(); //valor de acordo com a posição do eixo x
        y = acc.getAccY(); //valor de acordo com a posição do eixo y
        z = acc.getAccZ(); //valor de acordo com a posição do eixo z
        rled = 1.0 - abs(x); //intensidade da componente x na cor vermelha
        gled = 1.0 - abs(y); //intensidade da componente y na cor verde
        bled = 1.0 - abs(z); //intensidade da componente z na cor azul
        wait(0.2);
        printf("X: %1.2f, Y: %1.2f, Z: %1.2f\r", x, y, z);
    }
}
```

Dica:

- Para testar este programa, crie um novo projeto no Mbed e utilize o *template* “Example program for FRDM boards with a Freescale MMA8451Q”.
- Depois, copie o conteúdo do código do exemplo no programa principal.

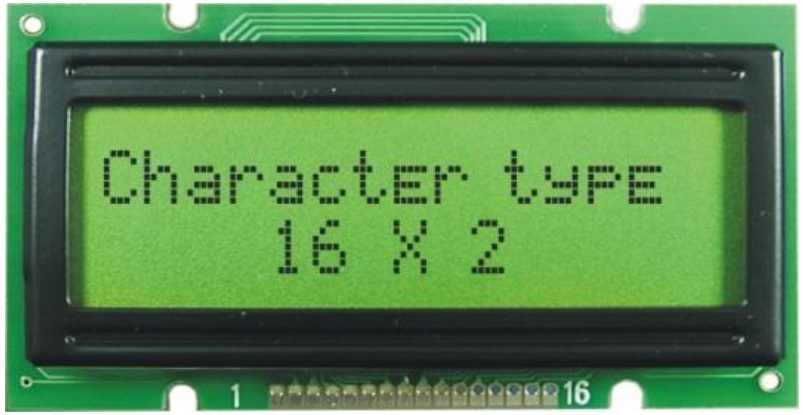
Display LCD

- Há diversos tipos e tamanhos de LCD alfanuméricos disponíveis comercialmente.
- Eles são sempre especificados em número de caracteres exibidos, no formato de colunas e linhas.
- Mais comuns: 08x02 (oito colunas por duas linhas), 16x01 (16 colunas por 1 linha), **16x02 (16 colunas por 2 linhas)**, 16x04 (16 colunas por 4 linhas), 20x01 (20 colunas por 1 linha), 20x02 (20 colunas por 2 linhas) e 20x04 (20 colunas por 4 linhas).

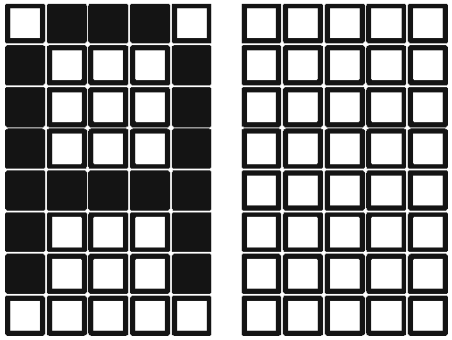
Funcionamento básico do cristal líquido

- São dispositivos que trabalham com luz polarizada com um líquido eletricamente controlável.
- Ao se aplicar ou retirar uma tensão em alguns locais (pixels), os cristais do líquido são polarizados, permitindo ou impedindo a passagem da luz.
- Não são elementos que emitem luz própria, apenas alteram o estado de reflexão da luz ambiente.
- Para funcionamento em locais escuros, precisam de uma fonte de luz na parte traseira, denominada *backlight*.

Exemplo de display LCD 16x02



- Cada “célula”(caractere) do LCD possui 8 pixels na vertical e 5 pixels na horizontal.
- Cada caractere é desenhado com até 35 pixels, pois a linha inferior é normalmente reservada para representar um cursor.



Interface com Display LCD

- Os LCDs mais comuns são os gerenciados por um chip controlador Hitachi HD44780.
- O módulo do display deve ser ligado à placa freedom através de um conjunto de conexões elétricas, denominadas **barramento de dados**.
- O barramento de dados pode ser de quatro bits (modo *nibble*) ou oito bits (modo *byte*).
- No modo *nibble*, são usadas apenas quatro linhas de dados (D4 a D7), simplificando a montagem.

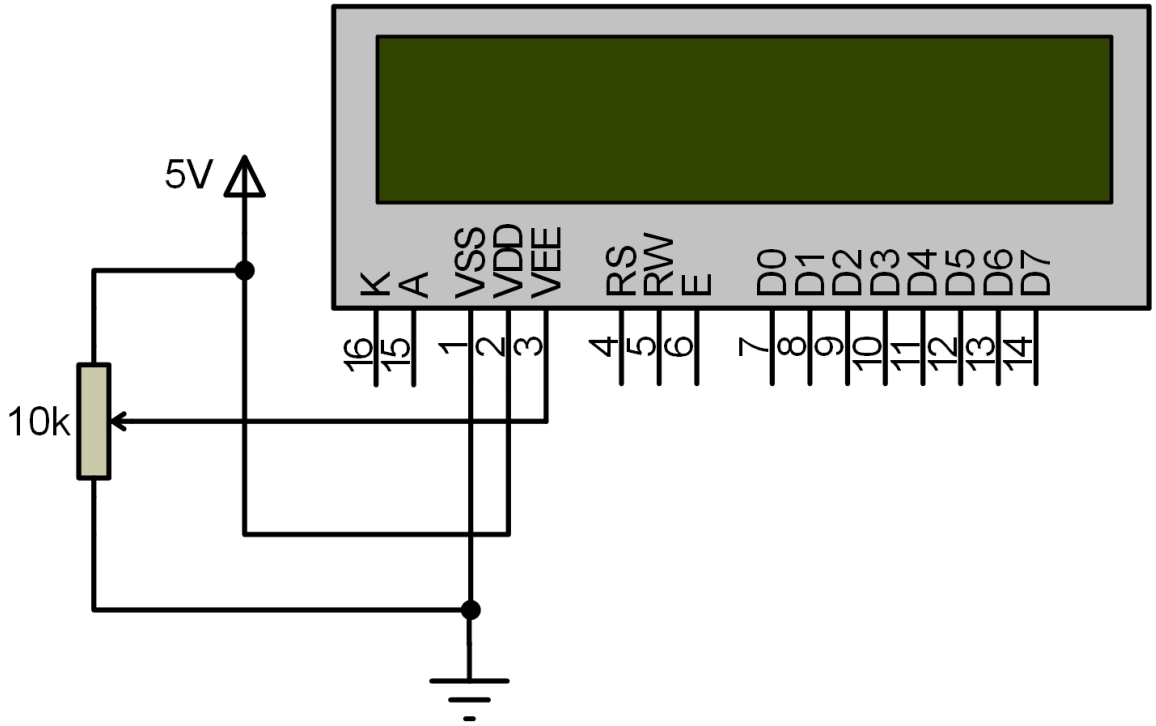
Pinagem

- Na maioria dos displays, a pinagem está impressa.

PINO	SÍMBOLO	FUNÇÃO	
1	VSS	GND (0V)	Alimentação
2	VDD	DC +5V	
3	Vo (VEE)	Ajuste de contraste	Barramento de comunicação
4	D/I (RS)	Dados/Instruções	
5	R/W	Leitura / Escrita	
6	E	Enable	
7	DB0	Linha de dados 0	
8	DB1	Linha de dados 1	
9	DB2	Linha de dados 2	
10	DB3	Linha de dados 3	
11	DB4	Linha de dados 4	
12	DB5	Linha de dados 5	
13	DB6	Linha de dados 6	
14	DB7	Linha de dados 7	
15	A	LED +	Energia para os LEDs de backlight
16	K	LED –	

Interface com Display LCD

- O pino 1 (V_{ss}) é ligado ao terra e o pino 2 (V_{dd}) na tensão de 5V.
- No pino 3 (*Vo*) deve-se ter uma tensão entre 0 e 5V para ajuste do contraste. Isso pode ser feito com um potênciômetro, como abaixo:



Interface com Display LCD

- No modo *nibble*, o display é controlado por meio de comandos e dados, enviados pelos pinos RS, RW, D4, D5, D6 e D7)
- O pino RS designa o tipo de informação enviada:
  - Se RS=0: envio de comandos;
  - Se RS=1: envio de dados.
- O pino RW controla a natureza da operação.
  - Se RW=0: operação de escrita de dados no display;
  - Se RW=1: operação de leitura.
- Também possui um pino de habilitação (ENABLE).
- Em geral, deixa-se permanentemente o pino ENABLE ligado em 3,3V e o pino RW ligado em GND.

Biblioteca TextLCD

- Biblioteca disponível no MBED para utilização do display LDC no kit FR25Z, modo *nibble*.
- A mesma deve ser incluída no cabeçalho do programa principal: `#include "TextLCD.h"`
- O seguinte comando define a pinagem utilizada do kit para interface com o display LCD:

```
TextLCD lcd(RS, Enable, DB4, DB5, DB6, DB7);
```

- Como exemplo, o comando:

```
TextLCD lcd(PTA13, PTD5, PTC9, PTC8, PTA5, PTA4);
```

faz as seguintes atribuições:

Biblioteca TextLCD

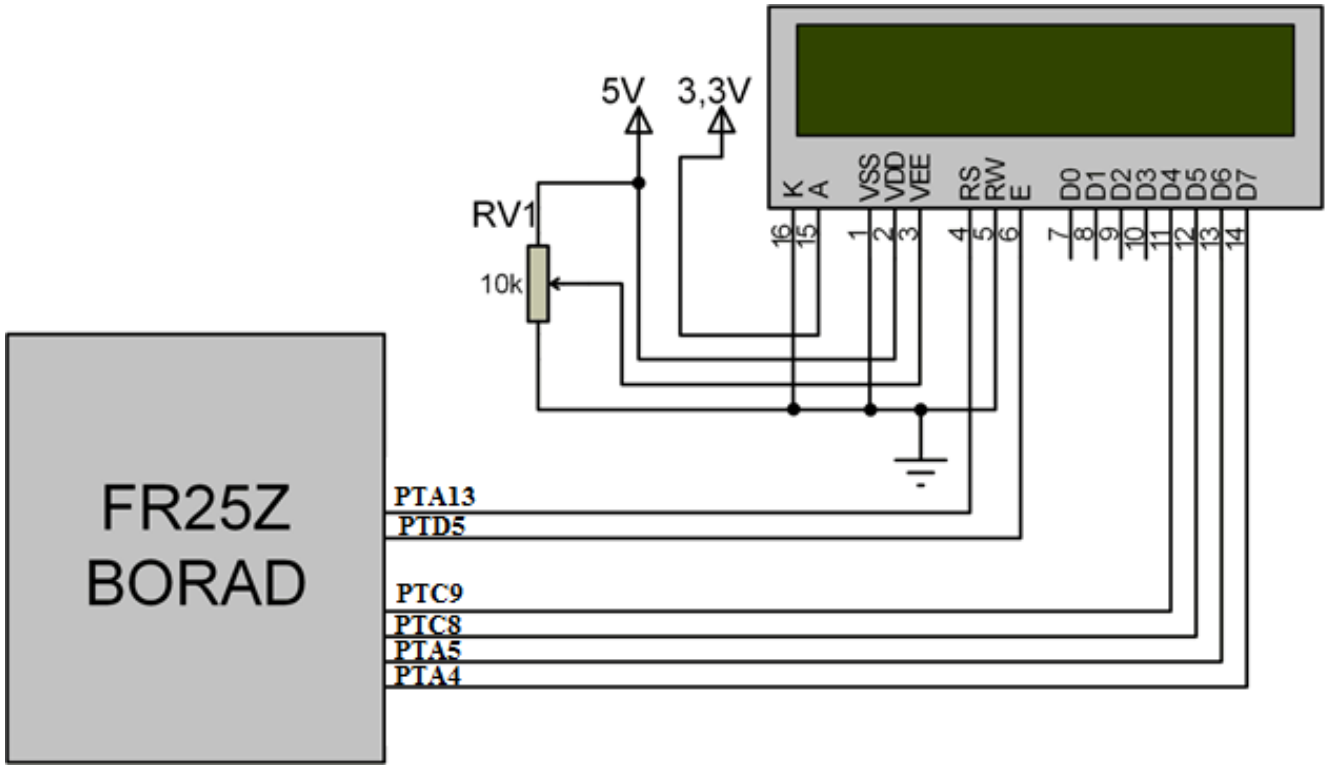
```
/* PINAGEM DA LIGAÇÃO FR25Z - LDC **
-----------------------------------------------
*  - DB4            : d4  -> PTC9
*  - DB5            : d5  -> PTC8
*  - DB6            : d6  -> PTA5
*  - DB7            : d7  -> PTA4
*  - RS             : d8  -> PTA13
*  - Enable         : d9  -> PTD5
*  - RW             : GND (only W mode)
-----------------------------------------------
***/
```

Biblioteca TextLCD

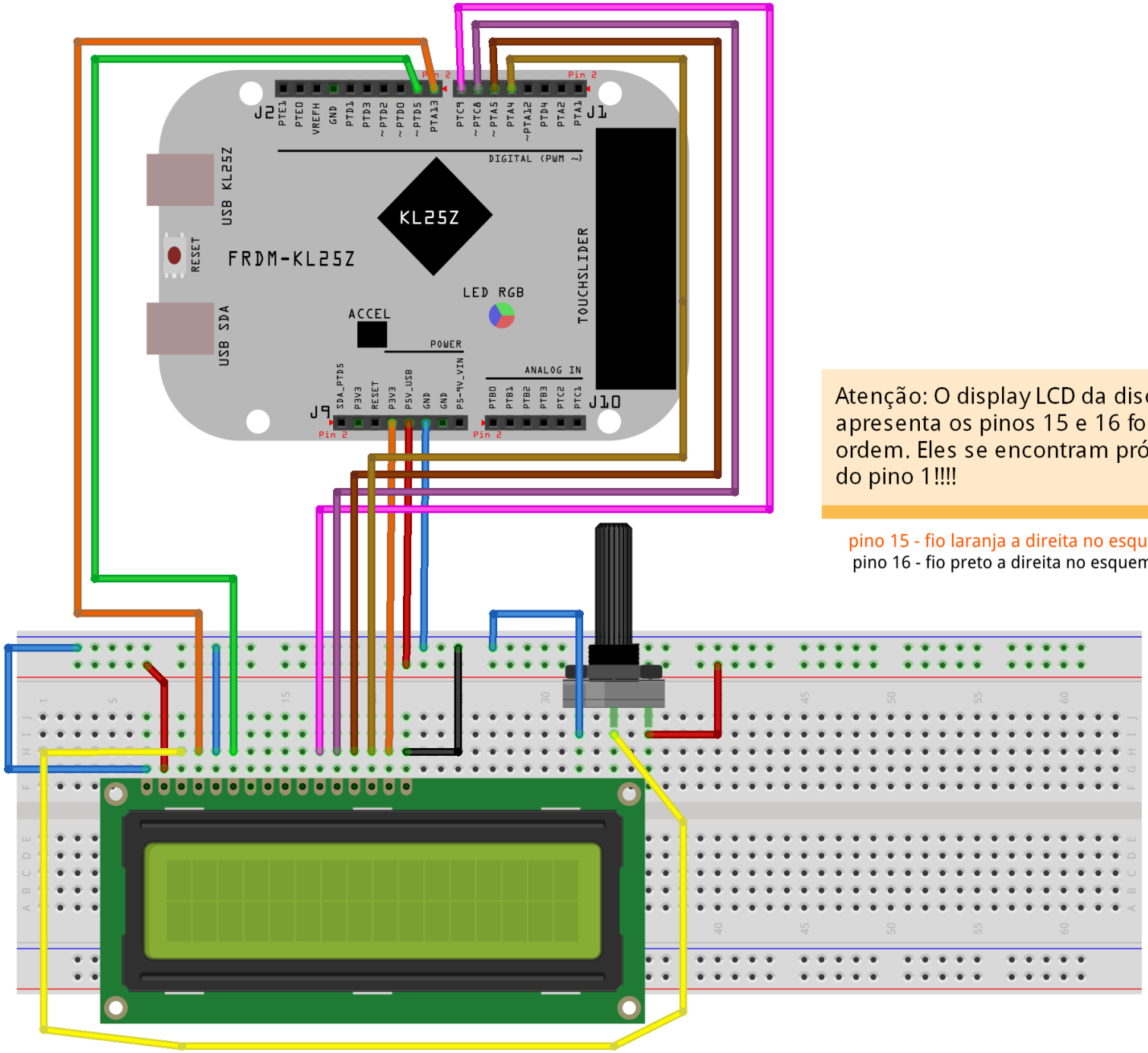
- O comando `lcd.cls()` limpa os caracteres do display;
- O comando `lcd.printf()` imprime *strings* constantes e variáveis no display.
- O comando `lcd.locate(x,y)` posiciona o display na coluna x (0 a 15), linha y (0 ou 1).

Exemplo de Aplicação

- Utilize o display LCD 16x02 para mostrar o valor do acumulador de um contador cíclico de 0 a 59, com incremento a cada 0,5 segundos.



Montagem com Display 7 Segmentos

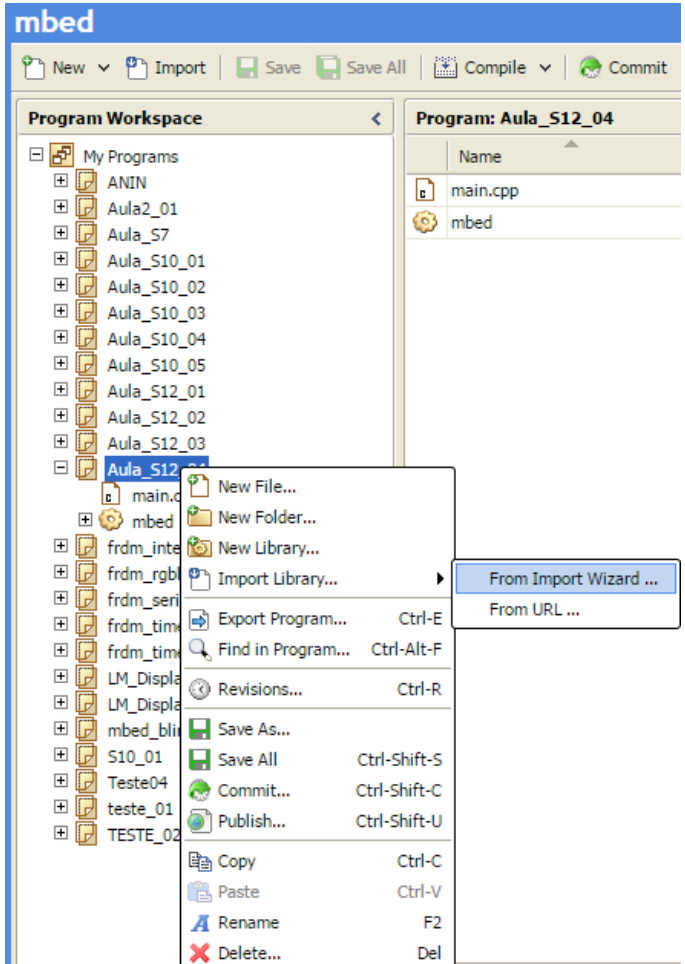
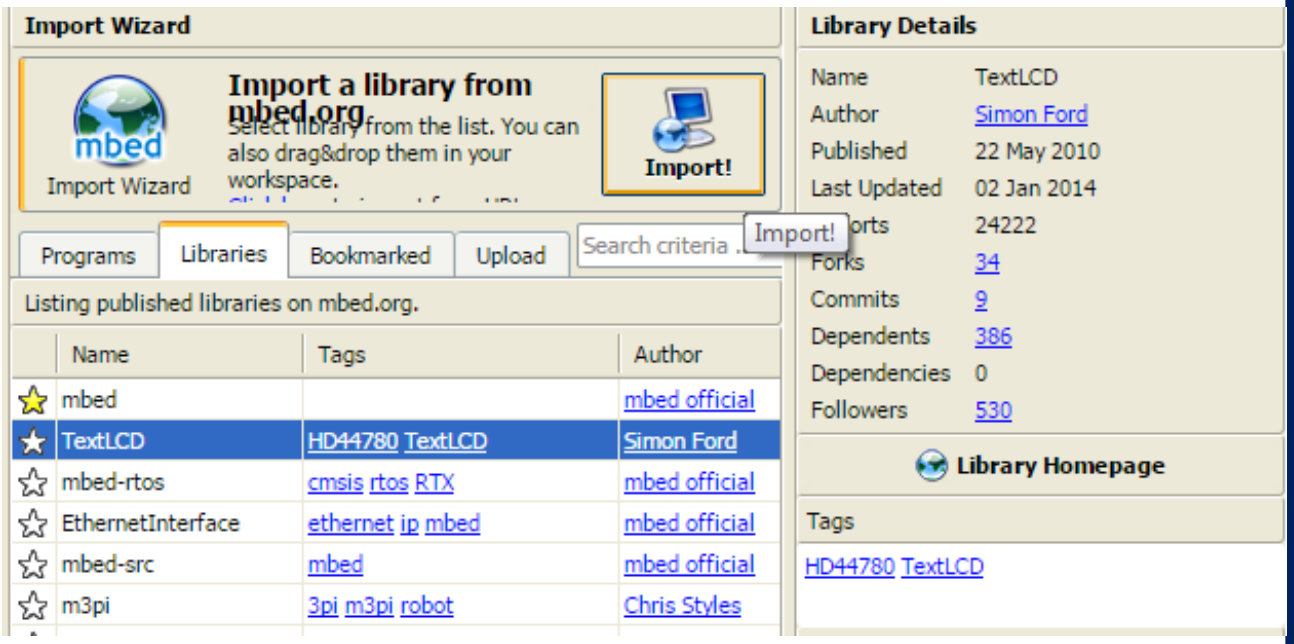


Atenção: O display LCD da disciplina apresenta os pinos 15 e 16 fora de ordem. Eles se encontram próximos do pino 1!!!!

pino 15 - fio laranja a direita no esquema
pino 16 - fio preto a direita no esquema

Código

```
#include "mbed.h"
#include "TextLCD.h"
TextLCD lcd(PTA13, PTD5, PTC9, PTC8, PTA5, PTA4);
int main() {
    int count = 0;
    lcd.cls();
    lcd.printf("CONTA DE 0 A 59");
    wait(0.5);
    while (1) {
        lcd.locate(0,1);
        lcd.printf("CT=%2d", count);
        count = count + 1;
        wait(0.5);
        if (count == 60) count=0;
    }
}
```

Observação:

- A biblioteca TextLCD.h precisa ser importada no projeto.