Universidade Federal do Rio Grande do Sul
Escola de Engenharia
Departamento de Sistemas Elétricos de Automação e Energia
ENG10032 Microcontroladores

Roteiro de Laboratório 10
***Inter Integrated Circuit* (I2C)**

Prof. Walter Fetter Lages

30 de maio de 2019

1 Objetivo

O objetivo deste laboratório é entender o funcionamento do barramento I2C e o acesso aos dispositivos I2C no Linux.

2 Fundamentação Teórica

O I2C é um barramento de comunicação serial utilizado para pequenas distâncias, tipicamente na mesma placa de circuito impresso. Ele opera de forma bidirecional no modo mestre-escravo e suporta vários mestres.

O SMBus, atualmente utilizado em *motherboards* de PCs, é um subconjunto do I2C. Vários dispositivos são compatíveis com ambos barramentos.

O I2C é baseado em dois sinais, com *drivers* em dreno aberto com resistores de *pull-up*:

`SCL:` *clock*, gerado pelo mestre

`SDA:` linha de dados bidirecional

2.1 I2C na Galileo Gen 2

O Quark X1000 possui um barramento I2C que está disponível no conector de *shield* da Galileo Gen2 e é utilizado na própria placa para acessar os expansores de GPIO nos endereços `0x25`, `0x26` e `0x27`, o PWM no endereço `0x47` e uma EEPROM serial de 1kB, ou seja, 8 kb, nos endereços `0x54-0x57`.

O barramento I2C disponível na Galileo Gen 2 é acessado através do dispositivo `/dev/i2c-0`, que suporta as operações `open()`, `close()`, `read()`,

`write()` e `ioctl()`. As operações de `read()` e `write()` são utilizadas para receber e transmitir dados, respectivamente e a operação `ioctl()` é usada para executar outros comandos no barramento. O comando mais importante é o `I2C_SLAVE` (constante definida em `linux/i2c-dev.h`), que é usado para definir o endereço do escravo a ser acessado. As seguintes chamadas IOCTLs estão definidas:

`ioctl(file,I2C_SLAVE,long addr):` Define o endereço do escravo.

`ioctl(file,I2C_TENBIT,long select):` Seleciona endereços de 10 bits se `select` é `0` e endereços de 7 bits se `select` é `1`. O *default* é 7 bits. Esta chamada só é válida se a interface I2C tem a função `I2C_FUNC_10BIT_ADDR`.

`ioctl(file,I2C_PEC,long select):` Seleciona geração e verificação de erros de pacote no SMBus. Só é válida se a interface I2C tem a função `I2C_FUNC_SMBUS_PEC`.

`ioctl(file, I2C_FUNCS unsigned long *funcs):` Obtém as funcionalidades da Interface I2C. As constantes definindo as funcionalidades estão definidas em `linux/i2c.h` e são documentadas no arquivo `Documentation/i2c/functionality` disponível na árvore do código-fonte do *kernel*.

`ioctl(file,I2C_RDWR,struct i2c_rdwr_ioctl_data *msgset):` Realiza operações de leitura e escrita sem parada intermediária. Apenas válida se a interface tem a funcionalidade `I2C_FUNC_I2C`. O argumento é um ponteiro para a estrutura `i2c_rdwr_ioctl_data`:

```
struct i2c_rdwr_ioctl_data
{
        struct i2c_msg *msgs; /* array de mensagens */
        int nmsgs;            /* numero de mensagens */
}
```

e a estrutura `i2c_msg`, também definida em `linux/i2c.h`, é:

```
struct i2c_msg
{
        __u16 addr;     /* endereço do escravo */
        __u16 flags;
        __u16 len;      /* tamanho da mensagem */
        __u8 *buf;      /* ponteiro para mensagem */
}
```

Os dados serão lidos do escravo se a *flag* `I2C_M_RD` estiver setada e escritos caso contrário. A *flag* `I2C_M_TEN` define se o endereço do escravo será de 7 ou 10 bits, sobrescrevendo o valor configurado por outra `ioctl()`.

Este acesso via `/dev` é mais útil para acessar dispositivos I2C conectados no conector de *shield*, já que os dispositivos I2C existentes na Galileo Gen2 (PWM e GPIO) podem ser acessados através das interfaces já abordadas nos laboratórios anteriores.

2.2 Acesso à EEPROM

A EEPROM pode ser acessada através do pseudo-arquivo `/sys/bus/i2c/devices/0-0054/eeprom` e pode ser lida e escita como se fosse um arquivo qualquer.

2.3 Acesso à Dispositivos I2C via `/dev`

Será utilizado o *Grove LCD with RGB Backlight*, mostrado na figura 1.

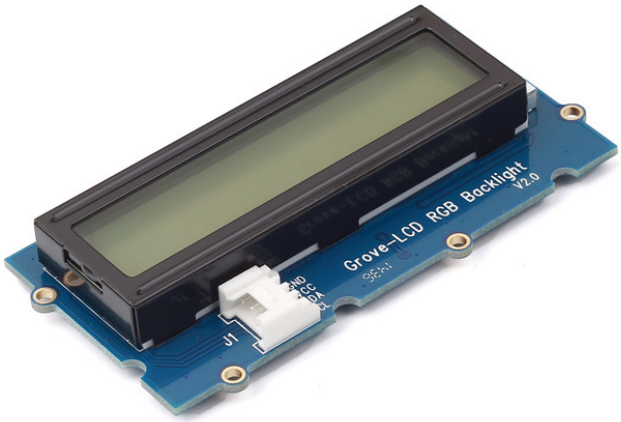

Figura 1: Grove LCD with RGB backlight.

Este módulo aparece no barramento I2C como dois dispositivos. O LCD propriamente dito (JHD1214), no endereço `0x7c` e o *driver* de LED de 4 bits (PCA9633) no endereço `0xc4`, que implementa 3 PWMs para acionamento dos LEDs utilizados no *backlight* do LCD, de forma que se pode controlar a sua cor.

A listagem 1 mostra um programa para inicializar o LCD e o *backlight* e escrever no LCD uma *string* passada como argumento na linha de comando.

Listagem 1: Escrita no LCD com *backlight* vermelho.

```
#include <errno.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <sys/ioctl.h>

#include <linux/i2c-dev.h>

#include <i2cutil.h>
#include <jhdlcd.h>

static void i2c_error(const char *msg)
{
    perror(msg);
    exit(-errno);
}

int main(int argc,char *argv[])
{
        int fd;
        int i;
        int n;

        if(argc != 2)
        {
                printf("writelcd\n");
                printf("\tUsage:\t%s <string>\n",argv[0]);
                return -1;
        }

        if((fd=open("/dev/i2c-0",O_WRONLY)) < 0) i2c_error("Opening /dev/i2c-0");
        usleep(30000);  /* Wait for 30 ms after power on */

        /* LCD initialization */
        if(ioctl(fd,I2C_SLAVE,LCD_ADDR) < 0) i2c_error("ioctl on /dev/i2c-0");

        i2c_write_reg(fd,LCD_C0,LCD_FUNCTIONSET | LCD_2LINE);
        usleep(40);  /* Wait for more than 39 us */

        i2c_write_reg(fd,LCD_C0,LCD_DISPLAYSWITCH | LCD_DISPLAYON |
                LCD_CURSOROFF | LCD_BLINKOFF);
        usleep(40);     /* Wait for more then 39 us */

        i2c_write_reg(fd,LCD_C0,LCD_SCREENCLEAR);
        usleep(1600);   /* Wait for more then 1.53 ms */

        i2c_write_reg(fd,LCD_C0,LCD_INPUTSET | LCD_ENTRYLEFT | LCD_DECREMENT);

        /* Backlight initialization */
        if(ioctl(fd,I2C_SLAVE,BL_ADDR) < 0) i2c_error("ioctl on /dev/i2c-0");
        i2c_write_reg(fd,BL_MODE1,0);
        i2c_write_reg(fd,BL_LEDOUT,BL_RED_GRPPWM | BL_GREEN_GRPPWM |
                BL_BLUE_GRPPWM);
```

```
        i2c_write_reg(fd,BL_MODE2,BL_DMBLNK);

        i2c_write_reg(fd,BL_RED,255);
        i2c_write_reg(fd,BL_GREEN,0);
        i2c_write_reg(fd,BL_BLUE,0);

        /* Write string */
        n=strlen(argv[1]);

        if(ioctl(fd,I2C_SLAVE,LCD_ADDR) < 0) i2c_error("ioctl on /dev/i2c-0");
        for(i=0;i < n;i++) i2c_write_reg(fd,LCD_RS,argv[1][i]);

        close(fd);

        return 0;
}
```

A função `i2c_write_reg()`, mostrada na listagem 2 está declarada no arquivo `i2cutil.h` e foi incorporada na biblioteca `libgalieo2` já utilizada em laboratórios passados.

Listagem 2: Função `i2c_write_reg()`.

```
#include <unistd.h>
#include <i2cutil.h>

int i2c_write_reg(int fd,unsigned char reg,unsigned char data)
{
        unsigned char buf[]={reg,data};

        return write(fd,buf,sizeof buf);
}
```

As constantes definindo os registradores e bits do LCD e do PWM do *backlight* estão definidas no arquivo `jhdlcd.h`, que também foi incluído nos cabeçalhos da biblioteca `libgalileo2`.

3 Experimentos

1. Crie na Galileo os grupos `eeprom` e `i2c` e inclua o seu usuário neles.
2. Faça um *script* de inicialização para ajustar as permissões do arquivo `/sys/bus/i2c/devices/0-0054-eeprom` para leitura e escrita para o grupo `eeprom`.
3. Instale o *script* de inicialização e reinicie a Galileo.

3.1 Uso da EEPROM

4. Faça um programa que escreva uma *string* passada na linha de comando na EEPROM através do pseudo-arquivo `/sys/bus/i2c/devices/0-0054/`

`eeprom`.

5. Faça um programa que mostre o conteúdo da EEPROM, obtido através do pseudo-arquivo `/sys/bus/i2c/devices/0-0054/eeprom`.

6. Teste os programas desenvolvidos em 4 e 5. Para isso, grave a EEPROM, desligue a Galileo Gen2, ligue-a novamente e verifique se os dados que foram gravados da EEPROM continuam lá.

3.2 Uso de Dispositivos I2C Através do `/dev`

7. Baixe do Moodle e instale o *script* para configuração dos pinos do conector de *shield* da galileo.

8. Compile e teste o programa mostrado na listagem 1. Para que o LCD funcione corretamente, é necessário colocar a chave `sw1` do *shield* base na posição de 5 V. Após os experimentos, retorne a chave para a posição de 3.3 V.

9. Faça um programa para implementar um relógio no LCD, fazendo com que a cor do *backlight* seja alterada aleatoriamente a cada segundo.