Programação de periféricos

Trabalho prático 2

Prof. Edson Moreno

1. Introdução

O presente trabalho tem por objetivo a exploração de recursos previamente apresentados em sala de aula, os quais exploram o uso de periféricos básicos (GPIO/LEDs) e a comunicação entre os diferentes microcontroladores presentes nos recursos computacionais disponíveis (ARM + ATMega).

2. Desenvolvimento

O trabalho deve ser desenvolvido no máximo em duplas. Neste trabalho será utilizada a raspberry PI com o SO Raspbian e a placa de expansão GertBoard. O objetivo do trabalho é desenvolver uma aplicação para a Raspberry e outra para o Arduíno capazes de interagir através da UART. A aplicação da Raspberry deve ser capaz de, além de se comunicar com o Arduíno capturar interrupções dos 3 botões disponíveis na GertBoard.

A interação consiste no envio de comandos a partir do raspberry para o arduíno. Tais comandos devem requisitar o acender/apagar de alguns LEDS. Assim, aplicativo executado na Raspberry recebe como entrada qual LED da Gertboard deve ser acionado e envia um comando para o Arduíno que executará a ação. A entrada na raspberry deve ser feita a partir do usuário e sua interação com um programa que captura as solicitações do usuário e as transfere via interface serial. O protocolo de comunicação entre a Raspberry e o Arduíno deve ser definido pelo grupo e descrito em relatório conforme solicitado no item 3. Ao todo, 4 LEDs da Gertboard devem ser controlados pelo Arduíno/Raspberry. Um código fonte em c para ser rodado na raspberry pode ser encontrado no final deste documento. Aquele código serve como modelo base, e não é a solução final.

O programa em c a ser desenvolvido deve continuamente monitorar o pressionamento dos botões contabilizando quantas vezes cada botão foi pressionado. O monitoramento deve ser realizado pelo raspberry a partir dos pinos de gpio que não estiverem sendo mapeados para a comunicação serial. O programa deve ter o controle da contagem de cliques em cada um dos botões. Sempre que uma alteração no número de cliques for realizada, o Raspberry deve imprimir na tela a informação de qual botão foi pressionado por último e quantos cliques foram contabilizados até então. Esta mesma representação deve acontecer nos leds da placa de expansão, sinalizando o número de cliques a partir de piscadas no led. Estas piscadas devem ser comandadas pelo arduíno, a partir de um comando

encaminhado a partir do raspberry que deve informar o botão pressionado e a quantidade de cliques . Exemplos de configuração/programação dos pinos de gpio, bem como sua manipulação para leitura e ou escrita podem ser encontrados no final deste documento.

A configuração da placa para a exploração, bem como o entendimento da pinagem deverão ser exploradas pelos alunos. Para tanto, a exploração do material disponibilizado nas aulas anteriores, tal como códigos, manuais e datasheets das placas devem ser usados.

3. Entrega

O grupo deve apresentar ao professor a integração Arduíno/Raspberry funcionando e postar todos os códigos fontes gerados (Arduíno e Raspberry) no Moodle até o início da aula da data de entrega do trabalho. Além disso, cada grupo deve escrever um relatório de no máximo 2 páginas descrevendo brevemente o protocolo de comunicação e o funcionamento da aplicação de maneira geral.

Cada elemento do grupo será avaliado individualmente. Alunos ausentes no dia da apresentação receberão grau zero. Também receberá grau zero todo o trabalho que for considerado plágio. O grau zero será atribuído tanto para o grupo que copiou quanto para aquele que forneceu o código original.

Suporte:

Para carregar o código no AVR:

avrdude -p atmega328p -c gpio -U arquivo.hex

Arquivo fonte para comunicação serial em C (link):

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

// device configuration function
int config_serial(char * device, unsigned int baudrate){
    struct termios options;
    int fd;

    fd = open(device, O_RDWR | O_NOCTTY | O_NDELAY );
    if (fd < 0)
    {
        /*
        * Could not open the port.
        */

        perror("config_serial: Não pode abrir a serial - ");
        return -1;
    }
```

```
    fcntl(fd, F_SETFL, 0);

    /*
    * Get the current options for the port...
    */
    tcgetattr(fd, &options);

    /* sets the terminal to something like the "raw" mode */
    cfmakeraw(&options);

    /*
    * Set the baudrate...
    */
    cfsetispeed(&options, baudrate);
    cfsetospeed(&options, baudrate);

    /*
    * Enable the receiver and set local mode...
    */
    options.c_cflag |= (CLOCAL | CREAD);

    /*
     * No parit, 1 stop bit, size 8
     */
    options.c_cflag &= ~PARENB;
    options.c_cflag &= ~CSTOPB;
    options.c_cflag &= ~CSIZE;
    options.c_cflag |= CS8;


    /*
     * Clear old settings
     */
    options.c_cflag &= ~CRTSCTS;
    options.c_iflag &= ~(IXON | IXOFF | IXANY);

    /* non-caninical mode */
    options.c_lflag &= ~ICANON;

    /*
    * Set the new options for the port...
    */
    tcsetattr(fd, TCSANOW, &options);

    /* configura a tty para escritas e leituras não bloqueantes */
    //fcntl(fd, F_SETFL, fcntl(fd, F_GETFL) | O_NONBLOCK);

    return fd;
}


int main(int argc, char** argv){
        int fd;
        char a;

        if(argc<2){
                printf("Usage: ./serial <char>");
                return 0;
        }
```

```
        fd = config_serial("/dev/ttyAMA0", B9600);
        if(fd<0){
                return 0;
        }

        // send a byte/char received to the configured interface (serial)
        a = argv[1][0];
        write(fd, &a, 1);

        // receive the byte from this interface
        read(fd, &a, 1);

        printf("%c\n", a);
        close(fd);

        return 0;
}
```

Para definir um pino (em C):

```
p_file = fopen ("/sys/class/gpio/export" , "w");

strcpy(buff,"_NUM_PINO_");

fputs (buff, p_file);

fclose (p_file);
```

Para liberar um pino (em C):

```
p_file = fopen ("/sys/class/gpio/unexport" , "w");

strcpy(buff,"_NUM_PINO_");

fputs (buff, p_file);

fclose (p_file);
```

Para definir um pino como saída (em C):

```
p_file = fopen ("/sys/class/gpio/gpio_NUM_PINO_/direction" , "w");

strcpy(buff,"out");

fputs (buff, p_file);

fclose (p_file);
```

Para definir escrever 0 em um pino (em C):

```
p_file = fopen ("/sys/class/gpio/gpio_NUM_PINO_/value" , "w");
strcpy(buff,"0");
fputs (buff, p_file);
fclose (p_file);
```

Para definir escrever 1 em um pino (em C):

```
p_file = fopen ("/sys/class/gpio/gpio_NUM_PINO_/value" , "w");
strcpy(buff,"1");
fputs (buff, p_file);
fclose (p_file);
```

Para definir um pino como entrada (em C):

```
p_file = fopen ("/sys/class/gpio/gpio_NUM_PINO_/direction" , "w");
strcpy(buff,"in");
fputs (buff, p_file);
fclose (p_file);
```

Para ler de um pino (em C):

```
p_file = fopen ("/sys/class/gpio/gpio_NUM_PINO_/value" , "r");
fgets (buff, 10, p_file);
fclose (p_file);
```

No Raspbian, para compilar um código em C:

gcc mycode.c -o my_code