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 por grupos formados por no máximo três pessoas. Neste trabalho será utilizada a raspberry PI, rodando o SO Raspbian, e a placa de expansão GertBoard. O objetivo do trabalho é desenvolver duas aplicativos em C/C++, uma para a Raspberry (piSW) e outra para o Arduíno (ArSW) capazes de interagir através da interface UART. O arSW deve ser capaz de se comunicar com o piSW e capturar interrupções dos 3 botões diretamente da GertBoard.

A conectividade da placa deve ser tal que o piSW possa capturar diretamente os 3 botões e se comunicar via UART com o arSW. Adicionalmente, a conectividade deve permitir que o arSW também se comunicaque via UART com o piSW e que interaja com 5 leds.

O piSW será responsável pelo envio de comandos para o arSW e a captura de informações vindas interface UART do arSW. Tais comandos devem que uma nova sequência (pré-definida) de ligar/desligar de LEDS seja assumida, ou que a sequência anterior seja retomada ou que o número de mudanças de padrão seja informado. Os comandos a serem enviados pelo piSW devem ser feitos pelo usuário através dos botões da gertboard. Sempre que o botão sw1 for pressionado, o comando de nova sequência deve ser enviado do piSW para o arSW. Quando o botão sw2 for pressionado, o padrão anterior deve ser reassumido. Quando o botão sw3 for pressionado, o piSW deve enviar um comando ao arSW solicitando o número de vezes que cada padrão foi usado. Este resultado deve ser apresentado na tela. O protocolo de comunicação entre o piSW e o arSW deve ser definido pelo grupo e descrito em relatório. Ao todo, 5 LEDs da Gertboard devem ser controlados na gertboard, de acordo com o desejo do grupo. Um exemplo código fonte em c que manipula a uart e que pode ser usado como base para o desenvolvimento deste trabalho pode ser encontrado no final deste documento. Cabe reforçar que aquele código serve como modelo base, e não é a solução final.

O arSW deve continuamente monitorar a interface UART, responder as requisições do piSW e atuar sobre os LEDS. O arSW deve ter o controle da contagem dos padrões de acionamento de LED. O número de sequências e os padrões devem ser predefinidos pelos projetistas.

A configuração da placa de expansão, 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, a configuração 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;
}
```

No Raspbian, para compilar um código em C:

gcc mycode.c -o my_code