Programação de Periféricos - Trabalho 1

Objetivo

O objetivo geral do trabalho é desenvolver um sistema embarcado simples, composto por um microcontrolador, alguns periféricos a serem controlados além de uma aplicação externa (a executar no *host*) para o envio de comandos. Os seguintes componentes deverão fazer parte do firmware:

1. 4 pinos de entrada (ligados a chaves) e 4 pinos de saída (ligados a LEDs) que podem ser lidos por ou escritos (programados) por software;
2. Protocolo para o envio de comandos para o sistema embarcado e recebimento de respostas, utilizando a porta serial;

Descrição

O sistema proposto deve permitir a leitura de alguns pinos e escrita de outros pinos, que serão utilizados para viabilizar a conexão com periféricos externos que possuam entradas e saídas digitais (como sistemas industriais, por exemplo). Em uma aplicação real, seria necessário incluir circuitos de proteção, com optoacopladores, transistores e/ou relés, mas isso foge o escopo do trabalho. Esse sistema embarcado será controlado por uma aplicação externa, que enviará e receberá dados por meio de uma interface serial.

Para isso, devem ser implementados no firmware diversos recursos para o gerenciamento, como operações de leitura, escrita e configuração de pinos e um protocolo para o envio e recebimento de comandos e interação com uma aplicação externa. Interrupções deverão ser utilizadas no firmware, gerando o envio de mensagens assíncronas (não requisitadas pela aplicação externa) no caso de gatilhos gerados em função da mudança de estado dos pinos de entrada, caso esses pinos estejam assim configurados.

Além da possibilidade de controlar o estado dos pinos de saída (0 ou 1) e ler o estado dos pinos de entrada, deve ser possível definir eventos de gatilho nos pinos de entrada. Por exemplo, a aplicação externa pode solicitar que um dos pinos de entrada seja sensível a uma mudança de estado (borda de subida) do sinal, e caso esse evento ocorra, o sistema embarcado deve enviar assincronamente uma mensagem à aplicação externa (no host).

Formato de mensagem (quadro) no enlace serial

As mensagens trocadas entre a aplicação de controle externa e o sistema embarcado devem seguir uma codificação com um formato de quadro específico. Nesse formato, são utilizados dois delimitadores, sendo um caracter específico para marcar o início e final de um quadro (contendo uma mensagem de requisição ou resposta) e um caracter utilizado para sinalizar *byte stuffing*. Os quadros de comando (enviado pelo host) e resposta (enviado pelo sistema embarcado) seguem o seguinte formato: o primeiro caracter é o delimitador de um quadro (ASCII 0x7e), seguido dos caracteres que representam os campos do comando / resposta representados com uso de *byte stuffing* (caracter de escape 0x7d, mais detalhes adiante) e por fim mais um caracter delimitador de quadro (ASCII 0x7e).

O conteúdo da mensagem (requisição ou resposta) possui os campos descritos a seguir. É importante observar que, caso os caracteres 0x7e ou 0x7d apareçam no fluxo de bytes, os mesmos deverão ser codificados com uma sequência de escape, que deverá ser tratada para o correto envio e recepção dos dados. Os dados enviados no quadro utilizam uma sequência de escape que segue as seguintes regras:

1. Caso o caracter a ser codificado não seja ASCII 0x7d nem 0x7e, simplesmente usar o caracter na mensagem;

2. Caso o caracter for 0x7d ou 0x7e, adicionar o código de escape ASCII 0x7d e adicionar o caracter, realizando a operação XOR com o valor 0x20 sobre o mesmo.

Por exemplo, em uma mensagem definida pelos caracteres 0xaa, 0xbb, 0xcc, 0x7e, 0x55, 0x7d e 0x23 o seu conteúdo deve ser transmitido como:

```
| 0x7e 0xaa 0xbb 0xcc 0x7d 0x5e 0x55 0x7d 0x5d 0x23 0x7e |
```

Os dados enviados em uma mensagem são representados pelos campos descritos abaixo:

- TID (transaction ID) - é um campo com valor aleatório de 16 bits, gerado no sistema hospedeiro (host). Uma resposta do sistema embarcado deve utilizar o mesmo TID da requisição. Uma mensagem assíncrona deve enviar um TID com valor 0xffff.
- OPER (operation) - é um campo de 8 bits. O valor da operação pode ser 0 (leitura) ou 1 (escrita).
- ADDR (address) - é um campo de 16 bits, identificando o endereço de um recurso. Um recurso (como pino de saída, por exemplo) pode ser mapeado para um endereço qualquer.
- DATA (dados) - é um campo de 16 bits, contendo os dados a serem transferidos. Em uma operação de leitura, esse campo será ignorado pelo sistema embarcado na mensagem enviada pelo host.

Aplicação de controle (no *host*)

Uma aplicação simples deverá ser desenvolvida para realizar o envio comandos ao sistema embarcado e também receber suas respostas. Além disso, essa aplicação será responsável por aceitar comandos do usuário e apresentar resultados ao mesmo, como notificações assíncronas em função de eventos ou respostas à solicitações (notificações síncronas). A aplicação de controle pode ser desenvolvida em qualquer linguagem (são disponibilizados exemplos em C e Python que fazem uso da porta serial).

Este trabalho deverá ser realizado individualmente ou em duplas e apresentado no dia 16/09 (apresentação em torno de 10 a 15 minutos). Para a entrega, é esperado que apenas um dos integrantes envie pelo Moodle um arquivo *.tar.gz* do projeto, contendo:

1. Código da aplicação embarcada (firmware) e *makefile* para geração do binário;
2. Código da aplicação externa.