Datalogger de Temperatura com Comunicação Serial

Iury Cleveston[1], Lucas Cielo Borin[1], Miguel Pfitscher[1]

[1] Curso de Engenharia de Computação – Universidade Federal de Santa Maria (UFSM)
Caixa Postal 1000 - 97105- 900 – Santa Maria – RS – Brazil

iurycl@gmail.com, lucas.borin@ecomp.ufsm.br, pfits002@umn.edu

1. Introdução

O objetivo deste trabalho foi projetar um *Datalogger* programável para medições de temperaturas. Conforme solicitado, o código foi estruturado em cima de uma FSM (*finite state machine*) com base em uma estrutura switch-case. Utilizamos memória EEPROM (*Electrically-Erasable Programmable Read-Only Memory*) para armazenar os dados coletados.

O projeto foi desenvolvido com a placa HCS08, utilizando o software *Codewarrior,* ambos da *Freescale*. Para fazer a conexão da placa ao computador foi utilizado um cabo serial.

Ilustração da placa HCS08

2. Arquivos do Projeto

O projeto foi desenvolvido utilizando a linguagem de programação C e separado em arquivos para melhor compreensão. Fizemos com que cada arquivo abordasse um propósito específico a fim de facilitar a manutenção futura do código e a identificação de *bugs*.

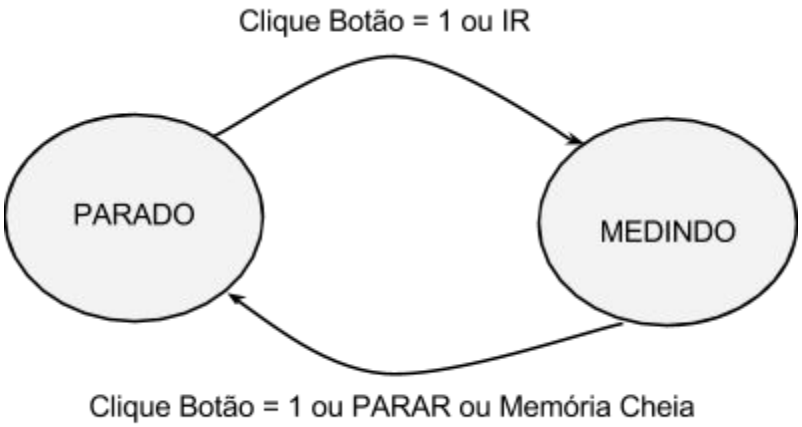
Para cada arquivo *source* (.c), foi criado um arquivo *header* (.h), o qual contém os protótipos das funções e definições de algumas variáveis globais que são utilizadas em outros arquivos. Abaixo são citados os arquivos utilizados:

- ProcessorExpert.c
- eeprom.c
- botaoIRQ.c
- amostragem.c
- timer.c
- serial.c

2.1. ProcessorExpert.c

Arquivo principal do programa, no qual foi implementada a FSM por switch-case. Basicamente, este código faz as configurações iniciais da placa, após isso, entra em um laço infinito no qual está situada a FSM e fica esperando comandos do usuário, via serial ou cliques no botão, para executar as demais funções implementadas.

Abaixo vemos o diagrama de estados do projeto:



A FSM foi estruturada com dois estados: PARADO e MEDINDO. Inicialmente, ela começa no estado MEDINDO, a troca para o estado PARADO ocorre quando o programa detecta um aperto no botão IRQ, quando o usuário envia um comando de parada ou quando a memória encheu.

A troca para o estado MEDINDO ocorre quando o sistema detectar um aperto no botão IRQ ou quando o usuário enviar um comando de retomada.

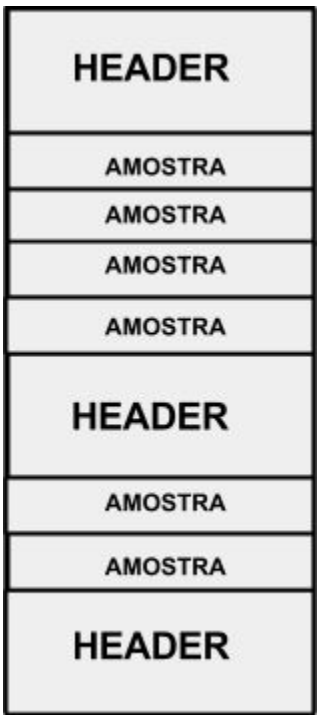
2.2. eeprom.c

Arquivo que contém todas as manipulações necessárias para o funcionamento da memória EEPROM. Aqui definimos as configurações da memória, bem como uma

função *delay* de tempo de operação da memória. Também possui as funções de leitura e escrita, nas quais diferem para salvar um dado amostrado ou um cabeçalho de informações. Por fim, possui uma função de formatação da memória que substitui os dados gravados por um valor padrão.

As informações salvas na memória podem ser de dois tipos diferentes, amostra ou evento. O tipo amostra é salvo quando o programa está amostrando valores de temperatura, já o tipo evento ocorre quando a memória encher ou quando detectar um baixo nível de tensão de alimentação no circuito.

Abaixo vemos como os dados são organizados na memória:



Todos os tipos de informações são salvas de forma contínua na memória através de uma *struct*, que também salva valores como a hora e a data em que a amostras ou eventos ocorreram. Os valores são salvos byte a byte, porém, há dados que possuem 2 bytes de informação. Para isso, foi implementada uma *union*, onde esses valores são quebrados em bytes separados.

2.3. botaoIRQ.c

Neste arquivo estão as manipulações necessárias para utilizar o botão IRQ contido na placa HCS08. Primeiramente, possui uma função de configuração, que gera valores para o registrador IRQSC. Possui uma função contador que detecta quanto tempo o botão ficou pressionado a partir da leitura do registrador PTAD5, necessária quando o usuário deseja fazer a formatação da memória.

Também possui funções que liberam o botão para o usuário poder pressioná-lo quantas vezes desejar sem perda de informação e funções que resetam contadores e registradores após o final do aperto. Todas essas função podem ser executadas graças a uma interrupção contida no vetor de interrupções que é executada quando o botão IRQ é pressionado.

2.4. amostragem.c

Aqui temos as funções necessárias para fazer a amostragem das temperaturas. Possui uma função que configura o conversor analógico/digital para que a temperatura seja encontrada através de um valor de tensão lido pelo sensor. Também há uma função de calibragem do sensor, pois o mesmo usa equações diferentes para medir temperaturas que estão acima e abaixo de 25 °C.

Ademais, quando o usuário envia o comando para realizar a amostragem, a função de inicialização é ativada, salvando a hora, data e a taxa de amostragem. Após esse cabeçalho ser salvo, as amostras são salvas uma após a outra. Cada amostragem é realizada entre um período de 10 milissegundos até 10 segundos, ficando a critério do usuário. Por fim, há uma função de parada, que interrompe o sensor e para a amostragem.

2.5. timer.c

Possui as funções dos três *timers* utilizados durante a execução do programa. Cada *timer* é acionado por uma interrupção diferente do vetor de interrupções, isso permite programá-los separadamente, pois há momentos em que mais de um *timer* é necessário para o funcionamento correto do programa.

2.6. serial.c

Aqui estão contidas as funções que descrevem como é feita a leitura e escrita através da porta serial. Também possui uma função que recebe os comandos enviados pelo usuário ao buffer de dados e os interpreta.

Outra função importante contida neste arquivo é a que faz a autenticação do usuário. Se o usuário não possui a senha correta, ele não será autenticado e não terá acesso ao resto das funções disponíveis. Caso o usuário tenho efetuado *login* e enviado algum comando, por exemplo, o comando de calibragem do sensor, mas por um grande período de tempo deixou de enviar os dados, o próprio sistema envia uma mensagem para o usuário alertando que o programa retornou para a rotina usual do sistema. Isso acontece pois enquanto o sistema espera comandos do usuário, os relógios do arquivo timer.c estão contando o tempo. Se o tempo em que o programa esteve obsoleto estourar, acontece o desligamento da função e retorno.

Caso o programa detecte que o usuário digitou um ‘ctrl-c’, o mesmo limpa o buffer de informações e se encerra.

3. Funcionamento

Primeiramente, é importante verificar se todos os componentes contidos na placa estão funcionando corretamente, se a porta serial foi reconhecida e a placa devidamente alimentada. Tendo isso, ao montar e executar o programa, usamos um *software* auxiliar do Arduíno que serve como terminal serial.

Ao abrir esta interface o programa mostrará uma mensagem padrão para o usuário. Para prosseguir o mesmo deve digitar o caractere ‘+’ três vezes em um curto espaço de tempo para poder ter acesso ao sistema. Ao fazer isso, a interface irá pedir a autenticação do usuário, que deve inserir uma senha. Como mostrado abaixo:

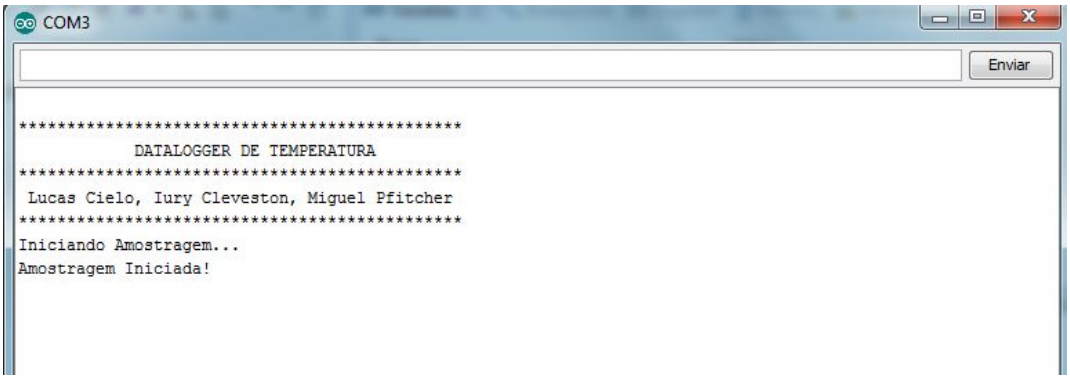


Figura 1: Tela de Início

Estando devidamente autenticado, o usuário tem acesso a todas as funcionalidades do programa. Digitando o comando AJUDA, receberá na tela um catálogo com todas as funcionalidades disponibilizadas pelo programa, como mostrado abaixo:

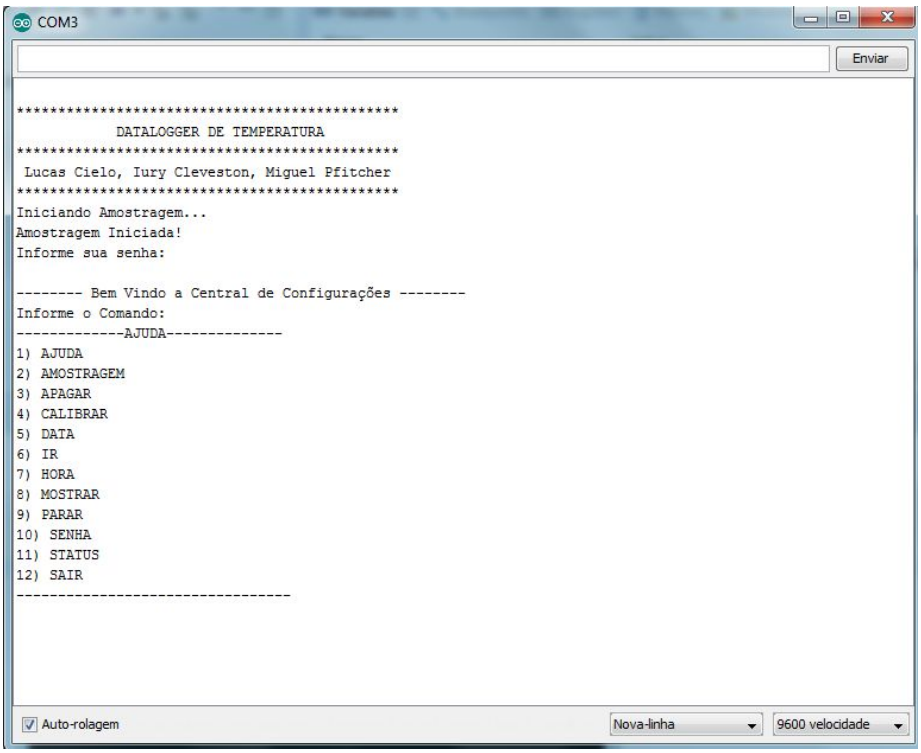


Figura 2: Tela de Ajuda

O comando AMOSTRAGEM permite ao usuário trocar a taxa de amostragem que, como citado anteriormente, pode ser programada entre 10 milissegundos e 10 segundos. O comando APAGAR permite ao usuário limpar todas as informações que estão armazenadas na memória.

CALIBRAR permite ao usuário fazer a calibragem do sensor. Logo após digitar esta opção, o usuário deve entrar com um valor que represente uma temperatura para que a calibragem seja executada. Se o usuário digitar uma temperatura maior do que o programa aceita, o programa retorna uma mensagem na qual devem ser enviadas apenas temperaturas dentro da faixa especificada, de -126 °C até 125 °C. Digitando o comando DATA, o usuário terá, junto com todos os valores amostrados de temperatura, a data em que o mesmo foi efetuado. Similarmente, o comando HORA tem a mesma funcionalidade do comando citado anteriormente.

IR e PARAR permitem ao usuário começar ou cessar a realização da amostragem. O comando MOSTRAR exibe na tela todos os valores amostrados salvos dentro da memória, bem como a hora e a data em que foram efetuados e com qual frequência foram amostrados, bem como na ilustração abaixo:

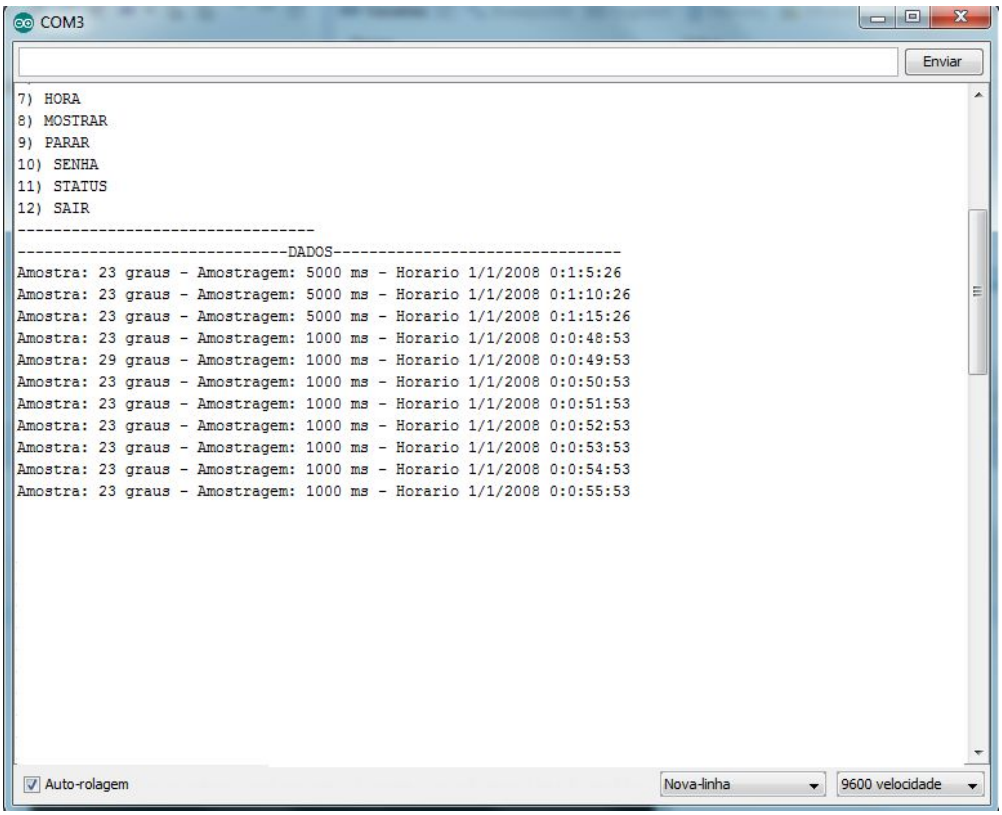


Figura 3: Mostrar os Dados

O comando SENHA permite ao usuário fazer a troca da senha que faz a autenticação do mesmo. STATUS permite ao usuário ter conhecimento de quanto espaço de memória já foi ocupado, a fim de ter um controle da quantidade de dados armazenados na memória EEPROM. O comando SAIR permite ao usuário efetuar *logoff* do sistema.

Por fim, se o usuário estiver com o programa rodando e por algum acidente a alimentação da placa for removida, ao reconectá-la, o programa terá salvo os dados que haviam sido amostrado. Do mesmo modo, os novos dados que serão amostrados terão a mesma taxa de amostragem salva anteriormente, junto com a data e hora que haviam sido programadas.

4. Modificações Futuras: Uso de infravermelho

Como especificado, o relatório deveria conter uma sessão sobre alguma mudança futura que o projeto virá a conter, sendo essa escolhida, o uso de comunicação através de infravermelho. Para isso, é importante conhecer com o que estamos trabalhando, abaixo segue algumas informações e requisitos sobre a comunicação e o dispositivo infravermelho.

4.1 Descrição: tópicos de conhecimento

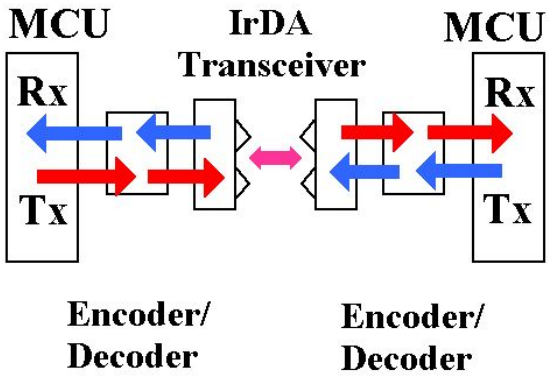
- Falando de hardware, o IrDA (*Infrared Data Associations*) é um dispositivo de conexão USB ou Serial que possua implementado um SIR (*standard Infrared*) e/ou um FIR (*fast Infrared*) que possibilitam haver uma taxa de transferência de dados de até 4 Mbps.

- Fabricantes: IrDA (Infrared Data Associations). Indústria formada em 1990 por mais de 50 empresas, entre elas Canon, Sharp, Actisys e Panasonic.
- Produto de fácil acesso, podendo ser encontrado em sites de vendas de componentes eletrônicos na internet.
- Sua faixa de preço pode variar entre R$ 30,00 até R$ 350,00.
- Geralmente possui dimensões em torno de 50 cm^3 e peso em torno de 60 gramas.
- A conexão pode ser feita via porta Serial ou USB.

4.2 Implementação, o que deve ser programado

- A comunicação com o microcontrolador deve ser feita de forma serial e assíncrona, com transferência de dados com tecnologia half-duplex.
- Funções de configuração do componente, definição da taxa de comunicação e registradores.
- Transmissão e recepção das informações.
- Laço infinito que espera alguma informação vinda do IrDA.

Abaixo temos uma ilustração do uso do IrDA para comunicação entre microcontroladores:



Como podemos ver, o IrDA é facilmente configurável e nos dá uma forma eficiente de comunicação a curtas distâncias.

4.3 Utilidade no sistema embarcado

Uma boa alternativa para o uso do IrDA seria que o mesmo poderia realizar as funções implementadas pelo botão IRQ.

Essas funções passariam a ser enviadas através de um controle que ao ser apertado, mandaria informações através do sinal infravermelho captado pelo componente IrDA.