Projeto 3: Gravador de Dados Ambientais (*datalogger*)

Prof. Tiago Fernandes Tavares e Prof. Antonio Quevedo
Modificado pelo Prof. Rafael Ferrari e por Erick Leonardo de Sousa Monteiro
Setembro de 2018

Dados ambientais são úteis tanto para fins imediatos (por exemplo, agricultura) quanto para pesquisas de longo prazo (por exemplo, a coleta de dados para análise do aquecimento global). Embora seja possível visitar estações de medição que ficam próximas a lugares habitados (por exemplo, no pátio de uma escola), a visita regular a alguns lugares é impraticável (por exemplo: a Ilha da Queimada Grande - SP, o topo do Pico das Agulhas Negras - RJ ou o fundo da Caverna do Janelão - MG). Uma solução possível para a aquisição de dados ambientais nesses lugares é usar um dispositivo que grava dados ao longo do tempo, e que pode ser recuperado periodicamente. Esse tipo de dispositivo chama-se *datalogger*, e será implementado neste projeto.

1 Objetivos

Ao fim deste projeto, o aluno deverá ser capaz de:

1. Projetar e implementar um sistema complexo tendo em vista a necessidade de compatibilidade com um protocolo.
2. Implementar os circuitos elétricos e algoritmos necessários ao funcionamento de um teclado matricial.
3. Implementar comunicação a dois fios (TWI/I^2C) e um sistema de arquivos uniformizado para preservação de dados em EEPROM.

2 Requisitos do sistema

A dupla deverá apresentar uma prova de conceito do sistema de gravação de dados (*datalogger*) que deve atender aos seguintes requisitos:

• O dispositivo será controlado usando um teclado matricial, de forma que possa ser ativado em campo sem a necessidade de um computador próximo. Seu status será apresentado no *display* LCD Nokia 5110, o mesmo usado no projeto anterior. O teclado permitirá a entrada dos comandos descritos na Tabela 1 para operação do sistema. O comando deve ser seguido da tecla cerquilha (#) para sua confirmação. A tecla asterisco (*) cancela o comando. Por exemplo, para limpar a memória (*Reset*), deve ser pressionada a tecla "1" seguida da tecla "#" para confirmação do comando.

Comando	Função	Observações
1	Reset	Apaga toda a memória, com aviso no *display* (zera o contador de medidas armazenadas)
2	Measure	Apresenta no *display* o valor atual da temperatura, sem gravar na memória.
3	Status	Mostra no *display* o número de dados gravados e o número de medições disponíveis
4	Inicia coleta periódica	Mostra mensagem no *display*
5	Finaliza coleta periódica	Mostra mensagem no *display*
6	Transferência de dados	Envia pela porta serial os dados coletados, mostra mensagem no *display*. Ver descrição detalhada no roteiro.

Tabela 1: Comandos do teclado. O caractere cerquilha (#) confirma o comando e o caractere asterisco (*) cancela. O *display* deve mostrar o nome da função em uma linha quando a tecla é pressionada.

• O comando de transferência via serial, tecla 6, deve transmitir as N ($1 \leq N \leq 1022$) medidas de temperatura mais antigas separadas por quebras de linha (CR + LF) pela porta serial. Após a confirmação do comando ("6" seguido de "#"), deve aparecer no *display* uma mensagem solicitando o número de medidas a serem transferidas. Nesse momento, o usuário deve digitar um valor entre 1 e 1022 e pressionar "#" em seguida para confirmar ou "*" para cancelar. As medidas transferidas podem ser visualizadas através do monitor serial ou do *plotter* serial.

• O sistema usará uma memória EEPROM do tipo AT24C16, com capacidade de armazenamento de 2048 palavras de 8bits. Os dados serão armazenados de forma padronizada, de modo que um CI de memória possa ser retirado do *datalogger* de uma dupla e colocado no de outra dupla mantendo o funcionamento normal do sistema. O padrão de armazenamento é descrito na Seção 5.
• O dado ambiental coletado será a temperatura ambiente, com resolução de décimo de grau, adquirida através de um sensor de temperatura analógico LM35 (http://www.ti.com/lit/ds/symlink/lm35.pdf). Consideraremos somente temperaturas positivas.
• Os dados serão gravados com resolução de 16 bits (2 bytes por registro), formando um número inteiro de 16 bits sem sinal (na ordem MSB – LSB), que representa a temperatura em décimos de grau. Por exemplo, uma temperatura de 37.8ºC é representada pelo número 378 (0x017A), ou pela sequência de bytes 0x01 – 0x7A.
• A periodicidade da coleta da temperatura no estudo de conceito será a cada 2 segundos, para que os testes possam ser realizados em tempo razoável. Em um sistema real, pode-se esperar vários minutos entre coletas, e também pode-se usar EEPROMs de maior capacidade, estendendo assim a autonomia do sistema.
• Neste projeto, em especial, não nos preocuparemos com o consumo de energia nem com o encapsulamento (caixa). Estamos trabalhando com uma prova de conceito.

3 Teclado Matricial

Um teclado matricial é um dispositivo que organiza teclas na forma de uma matriz com linhas e colunas. Há uma conexão externa para cada linha e uma para cada coluna. Em estado de repouso, a impedância entre as conexões das linhas e das colunas é muito alta. Quando a tecla na posição (*i; j*) é pressionada, a impedância entre as conexões da linha *i* e da coluna *j* cai para próximo de zero. Se nenhuma ou apenas uma tecla estiver sendo pressionada, a impedância entre as conexões de duas linhas distintas (assim como de duas colunas distintas) é muito alta.

Usando essa propriedade, é possível executar uma rotina chamada varredura. As conexões das linhas estão ligadas a saídas GPIO de um microcontrolador, e as colunas estão ligadas a entradas. Uma modificação na polaridade do sinal só pode chegar da linha *i* à coluna *j* se a impedância entre *i* e *j* for baixa, ou seja, se a tecla (*i; j*) estiver pressionada.

A rotina de varredura se baseia na ideia de que o não-contato em cada entrada relacionada à coluna *j* implica na leitura de um estado padrão. Para isto, é necessário que um pino de entrada nunca seja deixado flutuando (desconectado). Esse problema pode ser resolvido usando um resistor *pull-up*. Veja que, nesse caso, se a conexão da coluna não existir, então a leitura na GPIO será, forçadamente, um nível alto.

Chaves mecânicas apresentam um problema chamado *bouncing*. Ao ser pressionada, a chave oscila algumas vezes entre as posições aberta e fechada. Portanto, após ser lido um caractere válido, é preciso aguardar até que o transitório da chave tenha terminado antes da leitura do próximo caractere digitado.

Ao final deste roteiro, há alguns *links* para páginas que mostram com mais detalhes o algoritmo de varredura.

4 EEPROM externa com protocolo I^2C

A EEPROM (*Electric Erasable and Programmable Read-Only Memory*) é um dispositivo de memória não-volátil. Nesta seção, discutiremos um componente específico (AT24Cxx) e seu funcionamento. Trata-se de um circuito integrado de memória cujo acesso é realizado através de protocolo I^2C (*InterIntegrated Circuit*). O protocolo I^2C é um protocolo a dois fios e, portanto, também é chamado de *Two-Wire Interface*, TWI. O nome I^2C é uma *trademark* da Philips, e por isso é evitado por outros fabricantes. Apesar disso, o protocolo não é uma patente fechada e pode ser usado livremente, desde que com outro nome.

Nele, há dois fios: SDA é o fio onde trafegam dados e SCL é o fio que leva o sinal de *clock*. É um protocolo síncrono, isto é, um bit de dados só é transmitido quando acompanhado de uma borda de *clock*. Também é um protocolo mestre-escravo. Isso significa que em um barramento há, pelo menos, um dispositivo mestre e uma série de dispositivos escravos. O fluxo de dados em I^2C é bidirecional *half-duplex* e uma transferência de um dispositivo escravo para o dispositivo mestre só pode acontecer mediante requisição do dispositivo mestre.

Quando o dispositivo mestre deseja iniciar o processo de comunicação, ele envia ao barramento o **endereço** do dispositivo que deseja contactar. O dispositivo escravo, então, fica ativo, aguardando comandos. O dispositivo mestre passa a enviar bytes que representam os comandos e/ou dados, implementados no dispositivo específico. Após, a comunicação é encerrada pelo dispositivo mestre, que volta o barramento para a posição parada (trata-se de uma condição de parada, ou *stop condition*).

Embora seja, teoricamente, possível implementar todo esse processo manualmente usando temporizadores e GPIO, microcontroladores modernos já trazem *hardware* e bibliotecas específicas que resolvem as questões de envio de endereços, de dados e condições de parada. Nós utilizaremos a biblioteca **Wire** para realizar a comunicação com a memória. É fundamental que a dupla compreenda o funcionamento do protocolo TWI e como ele é manipulado pela biblioteca Wire.

Em termos computacionais, a EEPROM funciona como um grande vetor não-volátil de bytes. Assim, o acesso à EEPROM é completamente determinado pelas operações de escrita e leitura de dados. É muito importante a leitura do datasheet da memória AT24C16 para o entendimento do mecanismo de endereçamento do dispositivo no barramento TWI e também dos procedimentos de escrita e leitura de dados em endereços específicos da memória.

5 Sistema de Arquivos

Uma vez que o problema de escrever e ler dados da memória não-volátil tenha sido resolvido, é preciso definir como esses dados serão organizados. Uma forma muito simples de organização, descrita nesta seção, é adequada para dispostivos tipo *datalogger*, que recebem dados sequencialmente e não precisam apagar dados acessados aleatoriamente. Na explicação desta solução, a notação $x[n]$ representa a n-ésima palavra armazenada na memória (ou, equivalentemente, o conteúdo do n-ésimo endereço).

O sistema de arquivos se baseia em usar dois bytes no final da memória ($x[2046]$ (MSB) e $x[2047]$ (LSB)) para armazenar o número de registros existentes, assumindo que todos os registros têm o mesmo tamanho (2 bytes). Assim, o procedimento para reiniciar o sistema de arquivos consiste em simplesmente escrever o valor 0 (zero) nessas duas posições.

Ao receber uma instrução de escrever um novo registro de temperatura (2 bytes), o sistema grava o novo registro a partir do endereço **$2 * \{(256 * x[2046]) + x[2047]\}$** da memória. Esta posição é o primeiro byte livre da memória. Após, incrementa $x[2047]$, e caso haja um *rollover* desta posição (retorno ao valor 0), incrementa $x[2046]$.

7 *Links* auxiliares

Biblioteca para manipular o barramento I2C:
https://www.arduino.cc/en/Reference/Wire

Teclado:

http://www.ganssle.com/debouncing.htm

http://www.techsavvy.net76.net/8051%20microcontroller%20keypad%20interfacing%20code.htm

http://extremeelectronics.co.in/avr-tutorials/4x3-matrix-keypad-interface-avr-tutorial/

https://circuitdigest.com/microcontroller-projects/keypad-interfacing-with-8051-microcontroller

http://www.gadgetronicx.com/4x4-keypad-interface-with-8051/