Implementação de um Sistema de Salvamento de uma Lista Duplamente Encadeada na EEPROM

Fábio Aurélio de Deus R. Queiroz[1], Marcos Ferreira de M. Sousa[1]

[1]Departamento de Ciência da Computação
Universidade Federal de Catalão (UFCAT) – Catalão, GO – Brasil

{fabio_queiroz, moura.moura}@discente.ufcat.edu.br

Abstract. *This article describes the algorithm developed in the ESP8266 microcontroller language (similar to C++), which receives 20 (twenty) values from the serial input and stores them in EEPROM memory, the non-volatile memory of microcontrollers. From this it is possible to obtain the entered values even after restarting it (reset).*

Resumo. *Este artigo descreve o algoritmo desenvolvido na linguagem do microcontrolador ESP8266 (semelhante à C++), que recebe 20 (vinte) valores da entrada serial e os armazena na memória EEPROM, a memória não-volátil dos microcontroladores. A partir disto é possível obter o os valores inseridos mesmo depois de reiniciá-lo (reset).*

1. Introdução

A Memória de Leitura Apenas Programável e Eletricamente Apagável (EEPROM) é um tipo de memória não volátil que permite a leitura e escrita de dados de forma permanente. É comumente utilizada em sistemas embarcados, para armazenar informações importantes, como configurações e dados de calibração, devido à sua capacidade de retenção de dados mesmo quando não há energia elétrica.

Outrossim, introduzimos o uso de listas duplamente encadeadas, a fim de definir o tamanho destas a nível de execução. Apesar do nosso sistema possuir um tamanho estipulado de 20 valores, as listas são duplamente encadeadas e possuem alocação dinâmica.

1.1. Contexto

O algoritmo desde artigo foi desenvolvido com o intuito de armazenar dados em listas dinâmicas e armazenar uma lista mesclada contendo essas duas na memória EEPROM. Utilizar a memória EEPROM traz grandes benefícios, tendo uma manutenção simples (pode ser substituída por outra de maior capacidade de armazenagem de dados) e facilidade de reescrita, o salvamento de dados do estado atual (se houver queda de energia, a EEPROM voltará a funcionar no estado/momento que ela estava antes da queda de energia).

1.2. Objetivo

Armazenar dados de duas listas duplamente encadeadas, depois de mescladas, na memória EEPROM do microcontrolador ESP8266.

A partir de uma IDE de comunicação entre um notebook e o microcontrolador, é possível realizar entrada de dados para ele.

2. Materiais e Softwares

Estes foram os materiais e softwares que utilizamos no desenvolvimento e implementação do nosso sistema.

2.1. Materiais Utilizados

- NodeMCU 1.0 ESP8266 é uma placa de desenvolvimento baseada no módulo ESP-12E WiFi;
- Cabo USB para recarga e sincronização de dados através da conexão;
- Protoboard;
- Leds;
- Fios para conectar os leds ao NodeMCU;
- Notebook para fazer a conexão com o NodeMCU.

2.2. Softwares Utilizados

- Arduino IDE 2.3.0.

2.2. Metodologia

Método de pesquisa: através de aulas práticas sob a orientação do professor Tercio Aberto dos Santos, consultas em fóruns, documentação do NodeMCU e em exemplos de aplicações dentro do Arduino IDE.

3. Implementação

O programa que desenvolvemos para o ESP8266 permite gerenciar duas listas de 10 elementos cada, mesclá-las e salvá-las na EEPROM. Ele também oferece a opção de carregar a lista mesclada da EEPROM e limpar a memória.

3.1. Lista Duplamente Encadeada

Implantamos a classe LDE como módulo no software Arduino IDE para criação e manipulação das listas dinâmicas. Este se divide em duas partes: o arquivo de extensão *.h* (ver algoritmo 1a) que possui o esqueleto da classe LDE, como declaração de atributos, *structs* e funções; o arquivo *.cpp* (ver algoritmo 1b) que possui a implementação das funções e definição predefinida dos atributos se necessário (ver algoritmo 1). Estes dois arquivos foram colocados em uma pasta e esta, colocada na pasta de bibliotecas do Arduino IDE.

Algoritmo 1a – Esqueleto do módulo LDE

```
struct No {
    int dado;
    No* prox;
```

```
    No* ant;
};
class LDE {
private:
    No* primeiro;
    No* ultimo;
public:
    int size;
[...]
}
```

No Algoritmo 1a temos o *struct* de nome nó que é a estrutura de cada nó da lista dinâmica, ele possui dois ponteiros, sendo que o *ant* aponta para o nó anterior da lista e o *prox* aponta para o próximo. O dado é o valor que armazenamos no nó propriamente dito.

Declaramos os ponteiros primeiro e ultimo com o acesso privado (*private*) para este não ser alterado fora da classe e, o *size* (tamanho da lista) como público. Isto foi necessário, pois a lista mesclada não cria novos nós para armazenar as duas listas, ela apenas aponta para a posição de memória onde elas estão (ver algoritmo 2.3).

Algoritmo 1a – Implementação do módulo LDE

```
#include "LDE.h"
[...]
No* get(int pos) {
    if(pos >= size) return nullptr;
    No *no_temp = primeiro;
    for (int i = 0; no_temp != nullptr && i != pos; i++) {
        no_temp = no_temp -> prox;
    }
    return no_temp;
}
[...]
```

No algoritmo 1b, é importante notar que a função *get(),* que geralmente retorna um valor, está retornando um ponteiro que aponta para o nó desejado. Isto é bastante útil para manipulação dos ponteiros que acompanham os nós (*ant* e *prox*) e neste caso utilizamos para mesclar as listas 1 e 2. Além disso, é necessário incluir o esqueleto *.h* desse módulo neste arquivo, "LDE.h".

3.2. Código-Fonte

Algoritmo 2.1 – Declaração das Listas Duplamente Encadeadas

```
[...]
LDE lista1 = LDE();
LDE lista2 = LDE();
LDE lista_mesclada = LDE();
[...]
```

Algoritmo 2.2 – Declaração do Ponteiro *Lista* para Manipular as Listas 1 e 2

```
void preenche_lista(int numero_lista, uint8_t led) {
  [...]
  LDE *lista = &lista1;
  if(numero_lista) lista = &lista2;
  [...]
}
```

Algoritmo 2.3 – Mesclando as Duas Listas em uma Duplamente Encadeada

```
void mescla_listas(uint8_t led_lista1, uint8_t led_lista2) {
  [...]
  lista_mesclada.size = lista1.getSize() + lista2.getSize();
  lista_mesclada.setPrimeiro(lista1.getPrimeiro());
  lista_mesclada.setUltimo(lista2.getUltimo());
  lista_mesclada.get(9) -> prox = lista2.getPrimeiro();
  lista_mesclada.get(10) -> ant = lista1.getUltimo();
  [...]
}
```

O código completo se encontra neste link. Algumas linhas de código das funções apresentadas foram ocultas a fim de diminuir a excessividade de códigos.

3.3. Explicação do Código

Este programa permite ao usuário realizar as seguintes operações:

1. Preencher duas listas duplamente encadeadas de 10 valores;
2. Mesclar as duas listas em uma lista duplamente encadeada;
3. Salvar a lista mesclada na memória EEPROM;
4. Carregar a lista mesclada da memória EEPROM;
5. Limpar a lista mesclada da memória EEPROM;

O programa inclui um menu interativo que é exibido na porta serial, permitindo ao usuário selecionar a operação desejada digitando um número correspondente à opção do menu. Cada operação é realizada conforme a escolha do usuário, com feedback e mensagens exibidas na porta serial para indicar o progresso e o resultado de cada operação.

3.3.2 Entrada de Dados da Serial

A função *preenche_lista ()* aguarda a disponibilidade de dados na porta serial, onde os valores são lidos e inseridos na lista dinâmica *lista1* ou *lista2*. Durante esse processo, mensagens são enviadas para informar o usuário sobre o progresso da inserção. Se uma entrada inválida for detectada, como uma entrada vazia, um aviso é enviado e o processo de inserção é interrompido. Após o término da inserção, o buffer serial é limpo para evitar resíduos de dados.

3.3.3 Mesclagem de Listas

A função *mescla_listas()*, mostrada no Algoritmo 2.2, aponta para as primeiras 10 posições da lista 1, e para as últimas 10 os da lista 2.

3.3.4 Armazenamento na Memória EEPROM

No Algoritmo 2.3 é mostrado a função de salvamento dos valores da variável *lista_mesclada* na memória do microcontrolador. É importante notar que, apesar da memória do ESP ser de 512 bytes, o algoritmo utiliza apenas as primeiras vinte posições de memória.

4. Resultados

A memória se comportou como o esperado, salvando os valores da lista mesclada de 20 valores de forma que mesmo depois de reiniciar o microcontrolador, este não perdia os dados. A função *Serial.available(),* que retorna à quantidade de bytes da entrada do buffer, não se comportou como o esperado, retornando em todos os casos 0 ou 1.

Isso provavelmente ocorreu por causa da velocidade de frequência do microcontrolador, que foi muito alta para a função retornar uma quantidade correta de bytes. Por este motivo foi necessário a função *Serial.read(),* que executa a leitura do buffer e zera a contagem da função *Serial.available().*

5. Conclusão

O sistema que desenvolvemos armazena na memória EEPROM do ESP8266 20 valores fornecidos pela entrada serial micro-usb por meio da IDE Arduino. Esses valores não são perdidos com a interrupção de energia ao microcontrolador, pelo fato de ser não volátil, assim como as memórias HD, SSD, flash, etc.

A princípio este sistema se aplica apenas a entradas digitais, porém valores lógicos podem ser convertidos em digitais. Por este motivo este artigo se torna relevante a aplicações reais também, como o armazenamento de dados de forma remota em uma fazenda onde se deseja armazenar informações de clima, por exemplo, sem a necessidade de um computador próprio.

Para sugestão de futuros projetos semelhantes a esse, aconselha-se monitorar o funcionamento do microcontrolador de forma frequente (instalar leds na placa é uma opção), pois não há uma forma clara de saber se os dados estão de fato salvos na memória ou não.

Como dito anteriormente, esse sistema não se limita a entrada de dados pela porta serial. Ele também possui uma placa de rede embutida que permite conectá-lo a outro sistema. Isso dá inúmeras possibilidades, desde automatizar aparelhos que usam eletricidade (lâmpadas, televisores) a armazenar variações de uma medida analógica com o passar do tempo.

6. Referências

Eckhardt, A. Automacar: Uma Tecnologia para Carros Populares. *Trabalho de Conclusão de Curso (Ciência da Computação) - UNIFACVEST*, 2020. Disponível em: https://www.unifacvest.edu.br/assets/uploads/files/arquivos/280ca-eckhardt,-a.-automacar.-tcc-ciencia-da-computacao.-defendido-em-dezembro-de-2020..pdf. Acesso em: 21 fev. 2024.

SBC. *Templates para Artigos e Capítulos de Livros*. Disponível em: https://www.sbc.org.br/documentos-da-sbc/category/169-templates-para-artigos-e-capitulos-de-livros. Acesso em: 21 fev 2024.

Random Nerd Tutorials. Getting Started with ESP8266 WiFi Transceiver Review. Disponível em: https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review. Acesso em: 18 fev. 2024.