BIBLIOTECA DE MENUS HIERÁRQUICOS PARA DISPOSITIVOS EMBARCADOS

Igor Borges Tavares, Felipe Adriano da Silva Gonçalves, Carlos Augusto Bissochi Junior, Fernando Cordeiro de Castro, Daniel Pereira de Carvalho, Josué Silva de Morais
Laboratório de Automação, Servomecanismos e Controle (LASEC)
Núcleo de Controle e Automação (NCA)
Faculdade de Engenharia Elétrica (FEELT)
Universidade Federal de Uberlândia (UFU)
Av. João Naves de Ávila, 2160 - Bloco 3N - Campus Santa Mônica CEP: 38400-902
Uberlândia, MG, Brasil
e-mail: igorborgest@gmail.com

Resumo **– Tendo como objetivo o estudo do gerenciamento de uma interface que permita a troca de informações entre dispositivos embarcados e usuários, foi proposta e desenvolvida uma biblioteca que possibilite tal recurso para dispositivos dedicados, que quando contam com sistemas operacionais, são pequenos e não contam com gerenciadores de janelas. Além de desenvolvida, a biblioteca foi testada para várias plataformas por meio de simulações e de testes práticos.**

Palavras-Chave **– Sistemas embarcados, Interface Homem - Maquina, Dispositivos dedicados.**

HIARARCHICAL MENUS LIBRARY FOR EMBEDDED DEVICES

Abstract **– Aiming at the study of a management interface that allows the exchange of information between users and embedded devices a library was proposed and developed which enables this feature for dedicated devices, which rely on small operating systems that do not have a windows managing system. In addition to the development, the library was tested for various platforms through simulations and practical tests.**

Keywords **– Embedded systems, man - machine interface, dedicated devices.**

I. INTRODUÇÃO

É fato que grande parte dos dispositivos embarcados desenvolvidos na atualidade necessita de algum tipo de interface para permitir a troca de informação entre a máquina e o homem. Vários exemplos de tais aparelhos são utilizados diariamente pela maioria dos cidadãos e em número cada vez maior, como os computadores de bordo de automóveis, dispositivos de áudio, aparelhos biomédicos e telefones celulares.

Também há casos de dispositivos que tradicionalmente não dispunham de IHM (Interface Homem - Maquina), mas que devido ao aumento da tecnologia agregada e conseqüentemente da flexibilidade, passaram a oferecer tal interface, como são os casos de geladeiras, máquinas de lavar roupas, e outros eletrodomésticos.

Duas ilustrações dos exemplos citados anteriormente podem ser observadas nas Figuras 1 e 2.

Fig. 1. Computador de bordo do carro Astra.

Fig. 2. Interface de uma Máquina de lavar roupas da Brastemp.

Tendo em vista a necessidade de dispositivos com IHM, muitos sistemas operacionais para sistemas embarcados já são desenvolvidos com gerenciadores de janelas e outros recursos de manipulação de teclados, microfone entre outros. Entretanto tais sistemas operacionais como o *Android* e o *Windows Mobile* são de propósito geral, e muitas vezes não atendem dispositivos dedicados que exigem núcleos mais leves e que muitas vezes até garantam tempo-real. Para tais

Artigo publicado na IX CEEL
03 a 07 de outubro de 2011
Universidade Federal de Uberlândia - UFU
Uberlândia - Minas Gerais - Brasil

aplicações foi proposta e desenvolvida pela empresa de automação Altriz a biblioteca *EmbMenu* que gerencia uma interface simples de menus hierárquicos para *LCD* (*Liquid Crystal Display*) gráfico e ainda aceita interação com botões de navegação.

A implementação foi realizada focando em uma programação modularizada, que possibilita ao usuário da biblioteca construir e manipular sua hierarquia de menus de forma fácil e flexível, não sendo necessário conhecer o funcionamento interno dos algoritmos. A biblioteca foi totalmente desenvolvida em C ANSI (*American National Standards Institute*), o que a torna portável para várias arquiteturas, pouco dependente do hardware a ser utilizado e necessitando apenas de configurações de endereço de alguns registradores de I/O (na parte de *Hardware Abstraction Layer*) fundamentais para a comunicação com a tela.

Um exemplo de *LCD* Gráfico pode ser visualizado a seguir na Figura 3.

Fig. 3. *LCD* Gráfico.

II. DEFINIÇÃO DOS ITENS DO MENU

Para que a utilização da biblioteca EmbMenu em projetos embarcados seja conduzida de maneira fácil e intuitiva, permitindo que o desenvolvedor desses projetos defina e manipule sua própria hierarquia de telas e linhas, a biblioteca foi criada seguindo um modelo geralmente encontrado nas linguagens de programação orientadas a objeto, apesar de a linguagem C ANSI não oferecer diretamente esse recurso. O modelo de desenvolvimento adotado ajuda a abstrair mais ainda a camada de aplicação dos detalhes da implementação da biblioteca.

Como principal "classe" foi criada uma estrutura nomeada de *Item_Struct*, onde cada instância criada é um *Item* (linha) de menu. Cada *Item* pode ser dividido em seis tipos, cada um com seus propósitos que serão descritos em seguida:

- *OpenScreen*, que quando clicado abre uma nova tela com novos respectivos *Items*.
- *CallFunction*, que quando selecionado invoca uma função externa à biblioteca.
- *SelectIdiom*, que representa algum possível idioma para o menu.
- *ShowBool*, aquela linha que mostra algum valor booleano por meio de uma caixa que pode estar marcada ou não.
- *ShowInteger*, aquele que carrega um valor inteiro consigo.
- *ShowFloat* , *Item* que carrega uma variável de ponto flutuante consigo.

E cada novo *Item* criado para compor terá cinco atributos que o caracterizam que podem ser definidos como:

- *Enable,* que informa se o *Item* pode ser alterado via Interface ou não.
- *Value*, variável flexível que carrega o valor da linha independente do tipo de sua variável.
- *ValueAdr*, que armazena o endereço da variável externa que transferirá o seu valor para o *Item* a cada "*MENU_Refresh()*", que atualiza todas as variáveis endereçadas do menu.
- *Pfunc,* que é o ponteiro para função necessário caso o propósito do Item seja *CallFunction*.
- *String*, matriz de caracteres que armazena a frase que o *Item* imprimi na sua linha em todas as línguas que o menu suporte, onde em cada linha da matriz o nome da linha é escrita em uma língua diferente.

A definição da estrutura dos *Items* pode ser observada na imagem seguinte.

```
1  /**
2   * Type thats define the Items information
3   */
4  typedef struct
5  {
6      Item_Purpose Purpose; //!< Purpose
7      uint8_t Enable; //!< Enable
8
9      union
10     {
11         uint8_t Bool;
12         uint16_t Integer;
13         float Float;
14     } Value; //!< Value
15
16     void* ValueAdr; //!< ValueAdr
17     PtrFunction PFunc; //!< PFunc
18     char String[3][16]; //!< String
19
20 }Item_Struct;
```

Fig. 4. Definição da estrutura dos *Items*.

III. ESTRUTURAÇÃO DA HIERARQUIA

Com a biblioteca suportando a criação e caracterização de cada *Item*, foi necessário o desenvolvimento de alguma estratégia para que o usuário tivesse a capacidade de comunicar à biblioteca como esses *Items* devem ser dispostos na tela. A maneira mais eficiente consistiu na elaboração de duas variáveis globais que devem ser manipuladas para descrever a hierarquia do menu que será desenvolvido para o sistema.

Inicialmente é preciso que a lista de *Items* ("*ItemList[ItemNumber]*") seja preenchida com todos os *Items* que serão utilizados para a formação da hierarquia,

onde o índice de cada um seja correspondente à sua posição na lista.

Com a lista de todos os *Items* devidamente preenchida, é possível montar a hierarquia dos mesmos utilizando uma matriz de mapeamento ("*MenuMap [ItemNumber] [LineNumber]*"). Essa matriz é o ponto chave da biblioteca, e é utilizada como um mapa pelos algoritmos da mesma, que por meio dela consegue enxergar como todos os Itens se relacionam.

Cada linha da matriz corresponde ao *Item* de mesmo índice criado na lista de *Items*. E cada coluna da matriz corresponde às opções de seleção que cada *Item* possui. Como apenas os *Items* com propósito *OpenScreen* possuem opções de seleção, as linhas correspondentes a Itens dos demais propósitos devem ser preenchidas com "0".

IV. RECURSOS DA BIBLIOTECA

A estrutura interna da biblioteca é composta por 30 funções, das quais 11 delas são privadas usadas para funcionamento interno e 19 estão disponíveis para a utilização dos usuários.

As funções de uso externo à biblioteca atendem praticamente todas as necessidades possíveis que um usuário pode ter para a manipulação de um menu simples. Elas fornecem recursos como algoritmos para a manipulação do marcador do *Item* atual, para a mudança da janela corrente, cadastramento de variáveis atualizáveis, mudança de idioma, alterar ou ler dados dos *Items*, escrever caixas com mensagens na tela, entre outras que podem ser observadas na figura abaixo com todas as definições de funções globais.

```
//Setup and general functions
void MENU_init (uint8_t Item, uint8_t Idiom);
void MENU_ArrowUp (void);
void MENU_ArrowDown (void);
uint8_t MENU_Enter (void);
void MENU_ESC (void);
void MENU_GoTo (uint8_t Item, uint8_t Line);
void MENU_Refresh (void);
void MENU_MessageBox (char* String);
void MENU_CloseMessageBox (void);

//Get functions
uint8_t MENU_GetBool (uint8_t Item);
uint16_t MENU_GetInt (uint8_t Item);
float MENU_GetFloat (uint8_t Item);
uint8_t MENU_GetIdiom (void);

//Set functions
void MENU_SetBool (uint8_t Item, uint8_t Value);
void MENU_SetInt (uint8_t Item, uint16_t Value);
void MENU_SetFloat (uint8_t Item, float Value);
void MENU_SetIdiom (uint8_t Idiom);

void MENU_RegisterCallBack (uint8_t Item, PtrFunction PFunction);
void MENU_RegisterRefreshVariable (uint8_t Item, void* Value);
```

Fig. 5. Definição das funções globais.

Outro recurso interessante da biblioteca é a versatilidade de poder ter a lista e o mapa de *Items* alocadas na memória RAM, para acessos mais rápidos e para permitir modificações na hierarquia em tempo de execução, ou apenas deixá-las alocadas na memória Flash para economia de memória RAM.

V. TESTES

Para poder testar a biblioteca foi elaborada uma hierarquia de menus que usufruísse de todos os recursos disponíveis. Assim, foi projetada a seguinte estrutura que pode ser visualizada a seguir na figura 6.

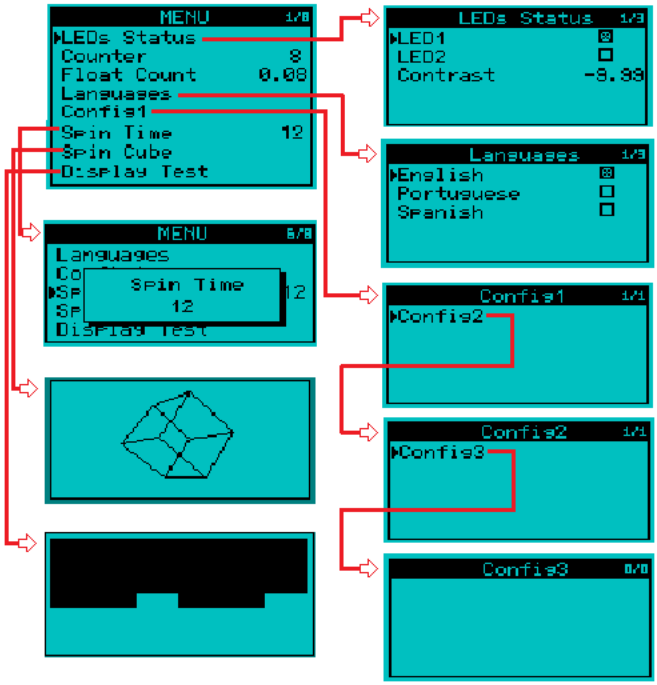

Fig. 6. Hierarquia de menu usada no teste.

Para a elaboração do código que executará essa hierarquia foi elaborada a lista de *Items* da seguinte maneira:

```
//the list of Items
Item_Struct ItemList[ItemNumber] =
{
{OpenScreen,true,{0},NULL,NULL,{{"MENU"},{"MENU"},{"MENU"}}},
{OpenScreen,true,{0},NULL,NULL,{{"LEDs Status"},{"Estado LEDs"},{"Estado LEDs"}}},
{OpenScreen,true,{0},NULL,NULL,{{"Languages"},{"Idiomas"},{"Idiomas"}}},
{OpenScreen,true,{0},NULL,NULL,{{"Config1"},{"Config1"},{"Config1"}}},
{OpenScreen,true,{0},NULL,NULL,{{"Config2"},{"Config2"},{"Config2"}}},
{OpenScreen,true,{0},NULL,NULL,{{"Config3"},{"Config3"},{"Config3"}}},

{ShowBool,true,{0},NULL,NULL,{{"LED1"},{"LED1"},{"LED1"}}},
{ShowBool,true,{1},NULL,NULL,{{"LED2"},{"LED2"},{"LED2"}}},
{ShowInteger,true,{2},NULL,NULL,{{"Spin Time"},{"Tempo de Giro"},{"Tiempo de Giro"}}},
{CallFunction,true,{0},NULL,NULL,{{"Spin Cube"},{"Gira Cubo"},{"Gira Cubo"}}},
{CallFunction,true,{0},NULL,NULL,{{"Display Test"},{"Testa Display"},{"Proba Display"}}},
{SelectIdiom,true,{Idiom_English},NULL,NULL,{{"English"},{"Ingles"},{"Ingles"}}},
{SelectIdiom,true,{Idiom_Portuguese},NULL,NULL,{{"Portuguese"},{"Portugues"},{"Portugues"}}}
{SelectIdiom,true,{Idiom_Spanish},NULL,NULL,{{"Spanish"},{"Espanhol"},{"Espanol"}}},
{ShowInteger,false,{0},NULL,NULL,{{"Counter"},{"Contador"},{"Contador"}}},
{ShowFloat,true,{0},NULL,NULL,{{"Contrast"},{"Contraste"},{"Contraste"}}},
{ShowFloat,false,{0},NULL,NULL,{{"Float Count"},{"Contador Float"},{"Contador Float"}}},
};
```

Fig. 7. Lista de *Items* da hierarquia testada.

E o mapa de *Items* pode ser analisado seguir:

```
//the map of the screens
const uint8_t MenuMap [ItemNumber][LineNumber] =
{
// LINES >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

{ Item_LEDs,Item_Counter,Item_FloatCounter,Item_Idiom,Item_Config1,Item_SpinTime,Item_SpinCube,Item_DisplayTes
{ Item_LED1, Item_LED2, Item_Contrast, 0, 0, 0, 0, 0, 0}, //Item_LEDs
{ Item_English, Item_Portuguese, Item_Spanish, 0, 0, 0, 0, 0, 0}, //Item_Idiom
{ Item_Config2, 0, 0, 0, 0, 0, 0, 0, 0}, //Item_Config1
{ Item_Config3, 0, 0, 0, 0, 0, 0, 0, 0}, //Item_Config2
};
```

Fig. 8. Mapa de *Items* da hierarquia testada.

Os testes foram realizados para os micro-controladores LPC2138/48 da NXP que contam com uma arquitetura ARM7TDMI-S RISC de 32-bits e um clock de CPU máximo de 60MHz. Também foram testados firmwares sem nenhum sistema operacional e com um pequeno sistema operacional de tempo real chamado FreeRTOS.

Foram realizadas simulações no software Proteus e testes no hardware real. Em ambos os casos, foram obtidos resultados muito semelhantes, observando-se uma resposta satisfatória e comprovando o bom desempenho da biblioteca para sistemas com recursos reduzidos, sem sobrecarregar o processador ou consumir muita memória RAM. Essas características permitem ao micro-controlador realizar muitas outras tarefas além de manipular a interface gráfica.

As visualizações da simulação e do teste no hardware real podem ser observadas nas figuras 9 e 10 a seguir.

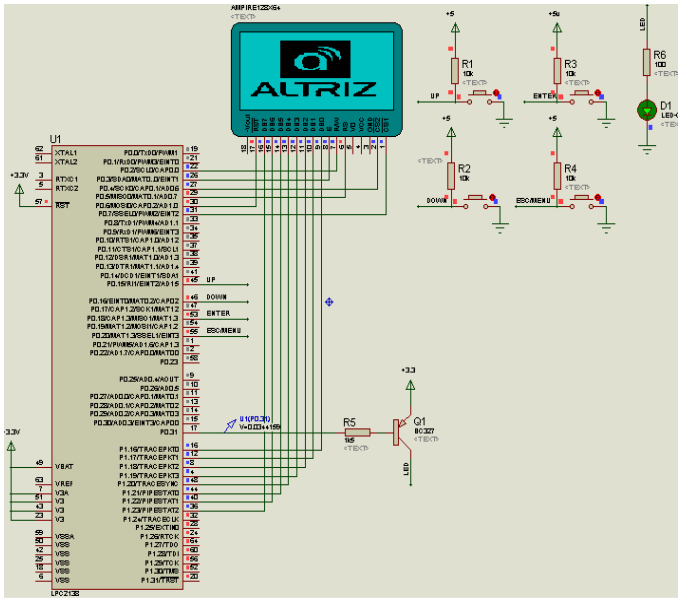

Fig. 9. Simulação do firmware de teste.

Fig. 10. Foto do teste com o equipamento real.

VI. CONCLUSÃO

A interface com o homem é uma parte fundamental em grande parte dos sistemas embarcados desenvolvidos na atualidade. Contudo, sistemas operacionais que oferecem recursos sofisticados de interface homem–máquina muitas vezes não são adequados a projetos com limitações de memória e processamento. Para esses projetos, foi criada e devidamente testada uma biblioteca flexível que pode trabalhar em praticamente qualquer micro-controlador. A utilização dessa biblioteca reduz os custos de desenvolvimento à medida que permite ao desenvolvedor concentrar-se mais na finalidade específica da aplicação sendo criada, em detrimento da programação da interface com o usuário.

AGRADECIMENTOS

Os autores agradecem às suas famílias e amigos pela compreensão nos vários momentos de ausência durante a realização do projeto, assim também como pelo apoio incondicional.

A conclusão do projeto também não seria possível sem a colaboração dos professores do LASEC (Laboratório de Automação, Servomecanismos e Controle), do Núcleo de Controle e Automação da Universidade Federal de Uberlândia, e da empresa Altriz (Altriz Indústria e Comércio de Equipamentos Eletrônicos Ltda) que idealizou e investiu no projeto.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Aubrey Kagan, “Hierarchical Menus in Embedded Systems”, FEATURE ARTICLE - Circuit Cellar, publicado em Novembro de 2003, em: http://www.circuitcellar.com.

[2] ZagGrad (2008). “LPC2148 USB Bootloader Tutorial”. Acedido em 15 de janeiro de 2012, em: http://www.sparkfun.com/tutorials/94.