실시간 운영체제 UbiFOS™에서 멀티미디어 기기를 위한 Graphic User Interface 설계 및 구현

이원용°, 이철훈
충남대학교 컴퓨터공학과
{nissikr°, chlee}@cnu.ac.kr

Design and Implementation of Graphic User Interface for multimedia device on Real-Time Operating System

Won-Yong Lee°, Cheol-Hoon Lee
Dept. of Computer Engineering, Chungnam National University*

요 약

실시간 운영체제(Real Time System)를 탑재한 내장형 시스템(Embedded System)은 수십 년 전부터 다양한 용도로 개발되어 왔다. 그래픽 장치들이 미비했던 초기의 내장형 시스템에서는 사용자 인터페이스가 단순하게 구현되었으나, 기술의 발달로 인하여 사용자가 쉽게 이용할 수 있게 GUI(Graphic User Interface)가 적용될 필요가 있다. 멀티미디어 기기에서 요구되는 포토 뷰, MP3P, 동영상과 같은 기능들을 만족 시키고, 또한 내장형 시스템의 특성상 GUI 가 경량이어야 한다. 본 논문에서는 실시간 운영체제인 UbiFOS™에 멀티미디어 기기를 위한 UbiFOS_GUI 를 설계 및 구현하였다.

1. 서 론

과거 내장형 시스템(Embedded System)은 User Interface(UI)가 간단한 텍스트 기반이었으나 근래에는 시스템과 사용자와의 입출력을 담당하는 인터페이스로서 GUI 가 많이 사용되고 있다.

최근 출시되는 내장형 시스템을 보면 발전된 기술로 인하여 화려하고 미려한 GUI 를 탑재하고 있고 다양한 크기의 TFT-LCD 를 장착하고 있다. 하지만 요즘의 내장형 시스템의 경우 이동성을 중시하는 경향 때문에 크기나 무게뿐만 아니라 사용시간에 있어서도 제약이 있다. 따라서 전력 효율성을 고려하여 하드웨어 성능을 높이지 않는 것이 유리하다. 결국 사용하는 마이크로프로세서의 초래하므로 범용 GUI 를 내장형 시스템에 적용하기에는 적합하지가 않다[3].

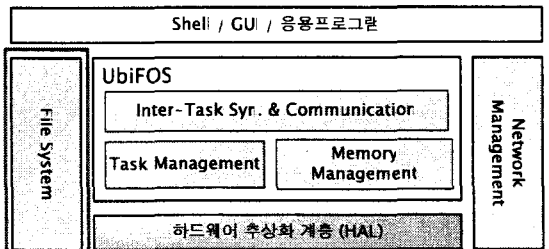
내장형 시스템에서 운영체제로 사용되는 실시간 운영체제의 마이크로 커널 경량화뿐만 아니라 GUI 와 네트워크에 대한 경량화가 이루어져야 한다. 특히 GUI 는 특성상 메모리를 많이 사용하게 되고, 수행시간이 많이 걸리므로 내장형 시스템에서 GUI 가 시스템의 경량화에 큰 요소가 된다[1].

본 논문의 구성은 2 장에서 관련 연구로서 실시간 운영체제인 UbiFOS™ 와 내장형 시스템에서 사용되고 있는 GUI 에 대해 설명하고, 3 장에서는 실시간 운영체제에 UbiFOS_GUI 를 설계 및 구현한 내용을 기술하며, 4 장에는 테스트 환경 및 결과를, 5 장에서는 결론 및 향후 연구과제에 대해서 논하고자 한다.

2. 관련 연구

GUI 를 구현하기 위해 기반이 되는 실시간 운영체제로서 UbiFOS™ 커널은 우선순위 기반의 실시간 운영체제이다. 즉, 실시간 운영체제 커널과 애플리케이션이 통합되어 하나의 큰 프로그램으로 동작하는 구조로써 공통의 메모리 영역을 자유롭게 접근할 수 있다. 또한 태스크들은 각각 우선순위가 부여되고 항상 Ready 상태에서 가장 높은 우선순위의 태스크가 실행된다.

2.1 UbiFOS™ 커널



(그림 2-1) UbiFOS™ 전체 구성도

[그림 2-1]은 UbiFOS™ 의 전체 구성도를 보여주고 있다. 가장 하위단계에 하드웨어와 하드웨어의 초기화 및 관리를 담당하는 디바이스 드라이버가 존재하며, UbiFOS™ 커널과 하드웨어 사이의 추상화 계층

* 본 연구는 정보통신부의 선도기반기술개발사업의 지원으로 수행되었습니다.

(Hardware Abstraction Layer)이 있고, UbiFOS™ 커널과 파일시스템 그리고 GUI 가 위치하게 된다[2][3].

2.2 내장형 시스템 GUI

현재 내장형 시스템에 사용되는 대표적인 GUI 로서 QT/Embedded, MiniGUI, MicroWindows, 등이 있다. QT/Embedded 는 Mac OS, 리눅스, 윈도우등에서 사용가능하고 구현이 C++로 되어있어 클래스에 익숙한 애플리케이션 개발자들에게 편리한 솔루션을 제공한다. 하지만 사이즈(Size)가 다른 GUI 에 비해 크다는 단점을 갖고 있다.

가. MiniGUI

Feynman Software Technology 사에서 개발한 MiniGUI 는 실시간 운영체제에 안정적으로 지원하는 것을 목적으로 개발되었다. 현재 핸드폰, PDA, 셋탑박스(Set Top Box)등 다양한 디바이스에 적용되어 있으며, 멀티 윈도우 메커니즘과 메시징 메커니즘을 제공한다. 또한 경량 윈도우 셋과 다양한 GDI API 를 갖고 있어서 내장형 시스템에 적합하다.

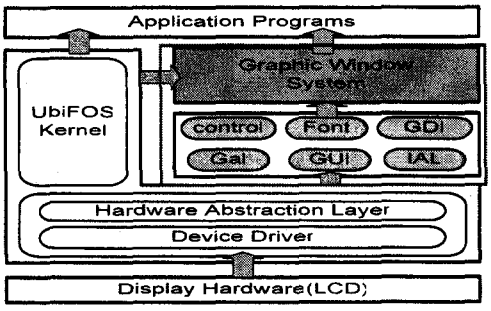
MiniGUI 는 Portable Layer 와 라이브러리 Layer 로 구분되며 Portable Layer 에서 하드웨어 관리를 담당한다. 따라서 이식성이 우수하며 GUI 애플리케이션 개발이 효율적이다[4].

나. MicroWindows

다양한 아키텍처에 포팅하기 위해 MicroWindows 는 3 개의 Layer 로 구성되는데, 가장 하위 레벨에서는 스크린과 마우스, 터치스크린 드라이버를 제공하여 실질적인 화면제어를 지원한다. 중간 계층의 Layer 는 디바이스와 독립적인 그래픽 엔진으로 구성되어 있으며, 핵심엔진은 API 에 포함되지 않고 서버에 독립적으로 존재한다. 가장 상위 Layer 는 사용자의 그래픽 애플리케이션을 위한 다양한 API 가 구현되어 있다[5].

3. 그래픽 윈도우 시스템 설계 및 구현

한정된 하드웨어를 갖는 내장형 시스템에는 범용 그래픽 윈도우 시스템을 사용하기엔 적합하지가 않다. 특히 제한적인 메모리 자원을 이용하여 경량의 그래픽 윈도우 시스템을 구축해야 한다.



(그림 3-1) 실시간 운영체제용 GUI

[그림 3-1]은 본 논문에서 구현한 그래픽 윈도우 시스템으로 LCD 를 위한 디바이스 드라이버, 일반적인 컨트롤 처리를 담당하는 Control, 그래픽처리에 대한 하위 레벨(low level)인 GAL 과 상위 레벨인 GDI, 윈도우 처리에 핵심인 GUI, 사용자 인터페이스 처리를 담당하는 IAL, 이미지를 위한 그래픽 프리미티브와 이를 이용한 그래픽 윈도우 시스템을 Layered Architecture Design 방식으로 설계한 것이다.

3.1 LCD 를 위한 구조체

```
typedef struct_screendevice {
    int  doclip;
    int  clipminx;
    int  clipminy;
    int  clipmaxx;
    int  clipmaxy;
    int  xres;
    int  yres;
    int  xvirtres;
    int  yvirtres;
    int  planes;        /* # planes*/
    int  bpp;          /* # bits per pixel*/
    int  linelen;
    int  size; /* size of memory allocated*/
    gal_pixel gr_foreground;
    gal_pixel gr_background;
    :
```

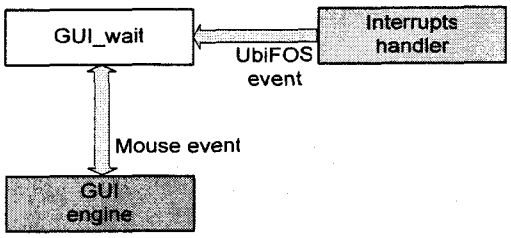
[그림 3-2] LCD 디바이스 드라이버 구조체

[그림 3-2]는 LCD 상의 데이터나 이미지를 출력할 때 사용하는 디바이스 드라이버의 구조체이다.

clipminnx ~ clipmaxy 까지는 x,y 좌표의 최소값과 최대값을 나타내는 필드이고 bpp 는 픽셀당 비트 수를 나타내며 size 는 할당된 메모리의 크기, 다음으로는 현재 foreground color 와 background color 를 나타내는 필드이다.

3.2 Touch Panel Interface

사용자 편의성 증대를 위해 Touch Panel 을 LCD 위에 부착해서 접촉한 위치의 좌표 값을 읽어와서 이벤트를 발생시킨다.

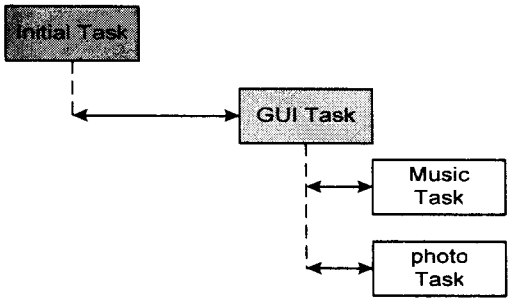


[그림 3-3] Touch Panel Interface

(그림 3-3)은 Touch Panel 에서 인터럽트가 발생하면

UbiFOS_GUI 로 이벤트(event)를 발생시켜 이벤트를 받은 UbiFOS_GUI 는 mouse 이벤트가 발생한 것처럼 처리한다.

3.3 태스크 생성 과정



(그림 3-4) 태스크간의 연관 블록도

[그림 3-4]에서 Initial 태스크는 시스템이 구동되면서 시작되는 태스크로서 시스템 초기화 작업과 GUI 태스크를 생성한다. GUI 태스크는 UbiFOS_GUI 모듈들을 초기화하고, Music 태스크, Photo 태스크를 생성 및 삭제시킨다.

3.4 GUI 태스크(Task)

GUI 태스크는 메인 애플리케이션 태스크로서 메인 윈도우, 헤더 윈도우등 각각의 윈도우 생성과 삭제 및 활성화 시킨다.

```
int UbiFOS_GUI_Main(void)
{
    //메인 윈도우 생성
    hMainWnd = InitMainWnd();
    //헤더 윈도우 생성
    hHeaderWnd =InitHeaderWnd();
    //도움말 윈도우 생성
    hHelpWnd = InitHelpWnd();
    //메인 윈도우 활성화
    SetActiveWindow(hMainWnd);
    //메시지 루프
    while(GetMessage(&Msg, hMainWnd))
    {
        TranslateMessage(&Msg);
        DispatchMessage(&Msg);
    }
    //MiniGUI 종료
    MainWindowThreadCleanup(hMainWnd);
}
```

[그림 3-4] UbiFOS_GUI_Main 함수

[그림 3-4]의 UbiFOS_GUI_Main 함수는 UbiFOS_GUI 모듈에서 가장 먼저 호출 되는 함수로서, 어플리케이션의 top level 윈도우를 생성하고 메시지를 처리한다. 메시지 루프를 빠져나오면 GUI 태스크도 종료되고 시스템에는 idle 태스크만 남게 된다.

```
HWND InitMainWnd(void)
{
    //메인 윈도우 생성
    hWnd = CreateMainWindow(&CreateInfo);
    return hWnd;
}
```

[그림 3-5] InitMainWnd 함수

InitMainWnd 함수는 윈도우와 Movie button, Music button, Photo button 등 서브 컨트롤들을 모두 생성한다.

```
int HeaderWndProc(HWND hWnd, int message, …)
{
    Switch(message){
      Case MSG_CREATE:
         //윈도우가 생성될 때 호출
         //윈도우 서브 컴포넌트들을 생성
        InitHeaderPart(hWnd);
        return 0;
      case MSG_TIMER:
        //타이머에서 발생된 메시지 처리
        DisplayCurTime(hWnd);
        return 0;
      case MSG_DESTROY:
        //윈도우가 삭제된 때 호출
        //윈도우의 서브 컴포넌트들을 삭제
        TimerHeadPart(hWnd);
        return 0;
    }
  //디폴트 메시지 처리 함수
  return DefaultMainWinProc(hWnd,message, …);
}
```

[그림 3-6] HeaderWndProc 함수

HeaderWndProc 함수는 message 에 따라 윈도우를 생성, 타이머 처리, 윈도우 삭제 및 서브 컴포넌트 삭제의 기능을 하는 함수이다.

3.5 Photo 태스크

Photo 태스크는 BMP 포맷의 파일을 디코딩하여 출력하는 태스크로서 내장된 비트맵 이미지 파일을 이동하면서 나타낼 수 있으며 슬라이드 쇼 기능을 포함하여 자동으로 모든 이미지를 나타낸다.

```
void PhotoTask(unsigned long arg, void* data)
{
    //guitask로 notification 전달
    Gem_sem_post((gen_sem_t*)data);
    //윈도우 속성 설정
    hWnd = CreateMainWindow();
    //윈도우 메인 메시지 루프
    while(GetMessage(&Msg, hWnd))
    {
```

```
    TranslateMessage(&Msg);
    DispatchMessage(&Msg);
  }
  //메인 윈도우에 관련된 MIniGUI 삭제
  MainWindowThreadCleanup(hWnd);
}
```

[그림 3-7] PhotoTask 함수

PhotoTask 함수는 이미지 파일을 출력하는 태스크의 메인 윈도우를 생성하여 Photo 태스크의 사용자 인터페이스 화면을 나타낸다.

```
int PhotoWndProc(HWND hWnd, int message, ···)
{
  switch(message){
    case MSG_CREATE:
      LoadCurrentPhotoImage(hWnd, 0);
    case MSG_COMMAND:
      PhotoCommand(hWnd, wParam, lParam);
    case MSG_LBUTTONDOWN:
      DisplayPhotoMenu(hWnd, WParam, lParam);
    case MSG_TIMER:
      LoadNextPhotoImage(hWnd);
    case MSG_PAINT:
      hDC = BeginPaint(hWnd);
  }
}
```

[그림 3-8] PhotoWndProc 함수

PhotoWndProc 함수에 들어온 message 에 따라 MSG_CREATE 이면 윈도우를 생성 후 첫 번째 photo image 를 출력하고, MSG_COMMAND 이면 서브 컨트롤 및 메뉴로부터 전달되는 메시지 처리하며, MSG_LBUTTONDOWN 이면 Popup 메뉴를 화면에 출력한다. 그리고 MSG_TIMER 라면 타이머 메시지를 처리하며 MSG_PAINT 이면 실제 photo 이미지를 화면에 출력한다.

```
void LoadCurrentPhotoImage(HWND hWnd, int flag)
{
  Fs_t* pFs = get_imagedir();
  Imgfile = pFs->files[curPhotoIdx].filename;
  curBitmapLen = get_imagefilesize(imgfile);
   :
```

[그림 3-9] LoadCurrentPhotoIma 함수

LoadCurrentPhotoImage 함수는 photo 이미지 디렉토리 테이블을 얻고 난 후 photo 파일이름과 파일의 크기를 얻어와서 사용자가 선택한 이미지 파일을 읽어 비트맵을 생성한다.

3.6 Music 태스크

Music 태스크는 mp3 파일을 플레이 하기 위한 사용자 인터페이스 윈도우를 생성하고, 사용자 선택에 따라 mp3 파일을 실제로 디코딩하고 플레이하는 mp3play 태스크를 생성 및 삭제 시킨다.

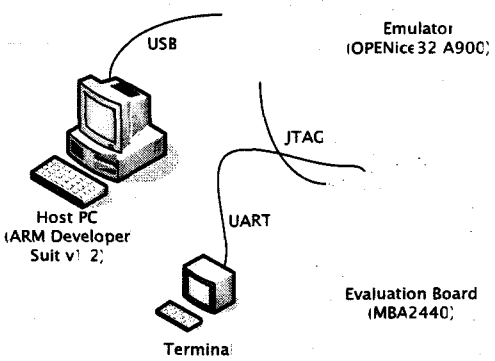
```
void MusicTask(unsigned long arg, void* data)
{
  //guitask로 notification 전달
  gem_sem_post((gen_sem_t*)data);
  //mp3 윈도우 생성
  hWnd = CreateMainWindow();
  //윈도우 메인 메시지 루프
  While(GetMessage(&Msg, hWnd)){
    TranslateMessage(&Msg);
    DispatchMessage(&Msg);
  }
  //메인 윈도우에 관련된 UbiFOS_GUI 리소스 삭
제
  MainWindowThreadCleanup(hWnd);
}
```

[그림 3-10] MusicTask 함수

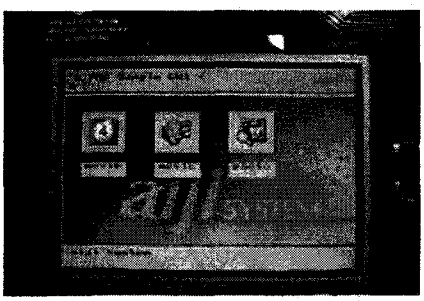
MusicTask 함수는 Mp3 파일을 플레이 하는 태스크의 메인 윈도우를 생성하고 메인 윈도우 리소스를 삭제한다.

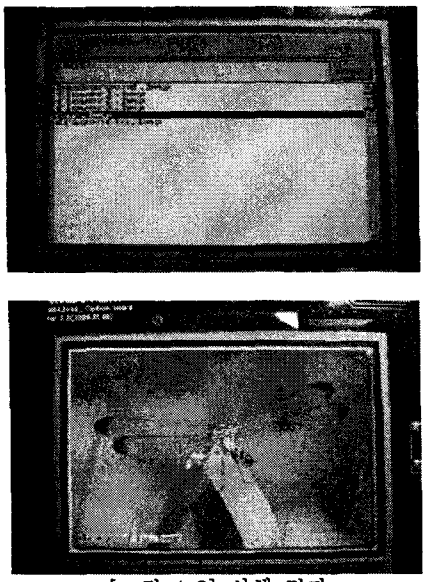
4. 테스트 환경 및 결과

본 논문에서 구현한 UbiFOS_GUI 는 실시간 운영체제 UbiFOS™ 상에서 구현하였고 컴파일러는 ARM ADS 1.2 을 사용하였으며, ARM920T 기반의 MBA2440 보드에 테스트 하였다.



[그림 4-1] 테스트 환경





[그림 4-2] 실행 결과

[그림 4-2]은 본 논문에서 구현한 UbiFOS_GUI 의 실행결과 화면이다. 기본적인 윈도우 생성과 텍스트 출력 및 그림파일 출력, 위젯을 이용한 버튼 출력 등이 터치스크린을 이용하여 정상적으로 수행됨을 단계적으로 확인할 수 있다.

5. 결론 및 향후 과제

본 논문에서는 실시간 운영체제 UbiFOS™ 상에서 다양한 멀티미디어 기기에 적용 가능한 GUI 를 설계 및 구현하였다. GUI 의 근간이 되는 도형, 한글, 영문 및 그림파일 출력기능 등을 제공하며, 또한 터치스크린을 구현함으로써 사용자 편의를 증대시켰다.

향후 연구과제로는 다양한 내장형 시스템을 위해 추가 할 다양한 위젯 구현과 GUI 상에서 오디오 출력을 위한 연구가 필요하다.

참고 문헌

[1] 강희성, 손필창, 이원용, 이철훈, " *The Design and Implementation of Graphic User Interface on Real-Time Operating Systems without File System*" , 한국정보과학회, Vol. 33, No. 1(A), pp.277-279, 2006, 6

[2] 강희성, 이정원, 조문행, 이철훈, " *Design and Implementation of Graphic Window System for Real-Time Operation System*" , 한국정보과학회, Vol, 32. No. 1(A), pp.856-858, 2005, 7

[3] 삼성전자, " *이동형 내장형 시스템에서의 경량 그래픽유저인터페이스의 구현*"

[4] http://www.minigui.com/product/index.html

[5] http://www.microwindows.org/microwindows_architecture.html