聯陽半導體
ITE Tech. Inc.

iTE SDK

函式庫開發指南

V0.9

ITE TECH. INC.

修訂記錄

修訂日期	修訂說明	頁次
2014/10/2	初建版本 V0.9	
2015/03/25	对文档提出一些修改建议	
2015/4/13	介绍如何添加新函式库	2,3
2016/04/6	新增 LIBITU Widgets	31

目錄

1. 前言

1.1 編寫目的

介紹相關函式庫的使用方法。

1.2 適用範圍

ITE SDK 軟體系統。

1.3 適用人員

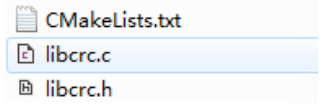
軟體設計人員。

2. 添加新函式庫

2.1 添加函式庫 source code

一般第3方函式庫的source code都擺放在sdk\share下, 如sdk\share\ lwip下是lwip的source code. 這裡以添加 crc 函式庫為例.

第一步, 將 crc 資料夾添加到 sdk\share 下



CMakeLists.txt 內容如下

```
add_library(crc STATIC
    libcrc.c
    libcrc.h
    )
```

第二步, 修改 sdk\build.cmake

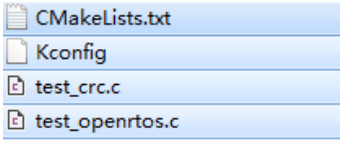
```
if (DEFINED CFG_BUILD_CRC)
    target_link_libraries(${CMAKE_PROJECT_NAME}
        crc
        )
endif()
```

2.2 添加 Test project

第一步, 在 build\openrtos\下參考 test_upgrade.cmd 新建添加一個批次檔 test_crc.cmd, 用來編譯 crc lib 並生成 image. test_crc.cmd 內容如下:

```
@echo off
set CFG_PROJECT=%~n0
call build.cmd
@if not defined NO_PAUSE pause
```

第二步, 在project資料夾下新增test_crc應用程式(裡面的文件可參考其他的test_xxx project):



CMakeLists.txt 內容如下

```
include_directories(
    .
    ${PROJECT_SOURCE_DIR}/sdk/include/
    ${PROJECT_SOURCE_DIR}/sdk/share/crc/
    )
add_executable(${CMAKE_PROJECT_NAME}
    test_openrtos.c
    test_crc.c
```

)
# build
include(${PROJECT_SOURCE_DIR}/sdk/build.cmake)
Kconfig 可參考其他 test_xxx project 內容, 主要是添加下面一項:
config BUILD_CRC
    def_bool y

2.3 Compile 函式庫

執行 build\openrtos\test_crc.cmd, 保存並編譯. 編譯完成後會生成 build\openrtos\test_crc\lib\fa626\libcrc.a

3. 相關函式庫介紹

3.1 Linphone

Linphone 是一套開放原始碼的網路視訊電話專案，使用 SIP 協議，並支援各種影音編/解碼器。我們使用並修改它來應用在可視對講的應用上。

網址: http://www.linphone.org/

3.2 SDL

SDL（Simple DirectMedia Layer）是一套開放原始碼的跨平台多媒體開發函式庫，使用 C 語言寫成。SDL 提供了數種控制圖像、聲音、輸出入的函式，讓開發者只要用相同或是相似的程式碼就可以開發出跨多個平台。我們主要使用它來做 Keypad 與 Touch Panel 的輸入之用。

網址: https://www.libsdl.org/

3.3 libcurl

libcurl 是一套開放原始碼的 URL 傳輸函式庫，支援包括 HTTP 和 FTP 等的協議。我們使用它來與控制伺服器和其它裝置間傳送訊息，以及遠端更新之用。

網址: http://curl.haxx.se/libcurl/

3.4 Libmicrohttpd

Libmicrohttpd 是一套開放原始碼的 HTTP server 函式庫，功能與所佔空間相當精簡。我們使用它來接收控制伺服器與其它裝置傳送來的訊息，以及設定網頁之用。

網址: http://www.gnu.org/software/libmicrohttpd/

3.5 Libxml2

Libxml2 是一個用來解析 XML 文件的函式庫。我們使用它來解析地址簿、解析卡片清單、接收控制伺服器與其它裝置傳送來的 XML 訊息，以及解析設定網頁之用。

網址: http://xmlsoft.org/

3.6 lwIP

lwIP 是一個輕量型的 TCP/IP 協定實作函式庫，並包裝了基本的 socket 相關函式。網路相關的函式庫例如 libcurl、Libmicrohttpd 與 sntp 等都架在本函式庫之上。我們也在 Network 模組中使用它來確認是否與控制伺服器保持連線。在本平台上我們只支援 IPv4 相關的函式。

網址: http://savannah.nongnu.org/projects/lwip/

3.7 sntp

簡單網路時間協議（Simple Network Time Protocol），由 NTP 改編而來，主要用來同步網際網路中的計算機時鐘。我們使用它來與控制伺服器保持時間同步。

網址: http://www-users.cs.york.ac.uk/~pcc/Circuits/mbed/baseboard/code/sntp.c

3.8 iniParser

INI 檔案是一個無固定標準格式的設定檔。它以簡單的文字與簡單的結構組成，常常使用在視窗作業系統，或是其他作業系統上，許多程式也會採用 INI 檔案做為設定程式之用。iniParser 是一個 INI 檔案的解析函式庫，支援讀取與寫入。

網址: http://ndevilla.free.fr/iniparser/html/index.html
http://en.wikipedia.org/wiki/INI_file

3.9 libredblack

紅黑樹是一種自平衡二元搜尋樹，是在計算機科學中用到的一種資料結構。它是複雜的，但它的操作有著良好的最壞情況運行時間，並且在實踐中是高效的: 它可以在 O(log n)時間內做查找，插入和刪除，這裡的 n 是樹中元素的數目。libredblack 是一個提供紅黑樹搜尋與排序序功能的函式庫。我們使用它來對檔案作排序。

網址: http://libredblack.sourceforge.net/
http://en.wikipedia.org/wiki/Red%E2%80%93black_tree

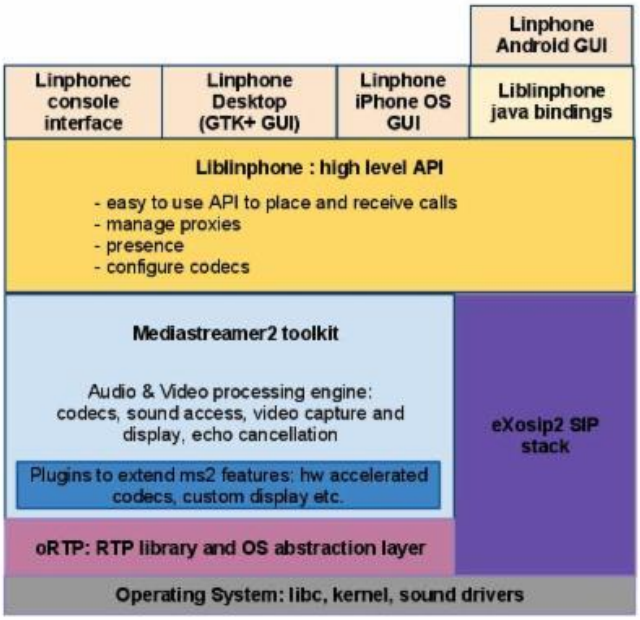
3.10 libitu

我們的 GUI 系統，負責讀取與繪製 GUI Designer 設計出來的 GUI 檔案。另外提供相關的函式來做更細緻的功能。

4. Linphone使用說明

Linphone 本身由獨立的 Task 運作，在桌上型作業系統下支援由命令列輸入指令的方式操作。我們修改了該命令列輸入的方式，改由 Message Queue 的方式對 Linphone 下達指令，並由 Linphone 模組封裝了輸入命令的實作行為。

4.1 Linphone 框架图



4.2 下達命令

除了 Linphone 本身標準的指令外，我們亦添加了自定的命令，以補足可視對講功能上的不足之處。所有的指令可以參考\sdk\share\linphone\console\commands.c 中 Commands table 的部分。例如 castor3-call 用作呼叫入口。

下面程式碼示範了設定鈴聲檔案到 Linphone 的方法，其中使用到了自定的命令 castor3-set-ring。

```
// command buffer
char prompt[PROMPT_MAX_LEN];

// make the command line
strcpy(prompt, "castor3-set-ring " CFG_PRIVATE_DRIVE ":/rings/");
strcat(prompt, theConfig.ringsound);
```

```
// send command to linphone
mq_send(linphoneCastor3.mq, prompt, PROMPT_MAX_LEN, 0);
```

當參數過多，不適合用命令列的方式傳遞時，我們使用一個叫 linphoneCastor3 的全域結構變數來傳遞參數。該結構定義在\sdk\include\linphone\linphone_castor3.h。下面程式碼示範了透過 Linphone 同時呼叫多個裝置的方法，其中使用到了自定的命令 castor3-call，並透多 linphoneCastor3 傳遞多個 IP 位址。

```
// set parameters via linphoneCastor3 structure instance
strcpy(linphoneCastor3.calling_ip_array[0], "192.168.1.20");
strcpy(linphoneCastor3.calling_ip_array[1], "192.168.1.21");
linphoneCastor3.calling_ip_count = 2;

// send command to linphone
mq_send(linphoneCastor3.mq, "castor3-call", PROMPT_MAX_LEN, 0);
```

4.3 回呼函式

當需要由 Linphone 主動通知事情時，我們透過自定的回呼函式來獲取通知。所有的回呼函式定義在 \sdk\include\linphone\linphone_castor3.h 。以下程式碼示範了將 lp_config_sync_notify_callback 回呼函式設成 UpgradeSetFileCrc，也就是當 Linphone 的設定檔 linphonrc 存檔完後，將會回呼 Upgrade 模組來更新 CRC 資料。

```
// called when linphonerc is saved.
lp_config_sync_notify_callback = UpgradeSetFileCrc;
```

5. SDL

藉由 SDL 跨平台的特性，讓我們的可視對講系統不用修改程式，不僅可以在嵌入式系統上執行，也可以在 WIN32 的環境下模擬執行。在 ITE 平台中，主要关注 Keypad 與 Touch Panel 的事件輸入接口。

5.1 進入點

SDL 將作業系統本身的進入點封裝在\sdk\share\sdl\main\openrtos\SDL_openrtos_main.c。WIN32 則是封裝在\sdk\share\sdl\main\windows\SDL_windows_main.c。而在專案下有個統一個進入點 SDL_main()，在<project>\main.c。

5.2 初始化

SDL 的被始化函數如下:

```
#include "SDL/SDL.h"
..

if (SDL_Init(SDL_INIT_VIDEO) < 0)
    printf("Couldn't initialize SDL: %s\n", SDL_GetError());
```

我們在 main.c 中呼叫以上函數。然後需要創造一個 window 物件，讓 SDL 知道輸入與繪製都在這個 window 物件上。

```
window = SDL_CreateWindow("Doorbell Indoor", SDL_WINDOWPOS_CENTERED, SDL_WINDOWPOS_CENTERED,
CFG_LCD_WIDTH, CFG_LCD_HEIGHT, 0);
if (!window)
    printf("Couldn't create window: %s\n", SDL_GetError());
```

5.3 接收事件

我們使用 SDL_PollEvent()來接收實體鍵與 Touch Panel 的輸入。在 WIN32 模擬環境下則是接收鍵盤與滑鼠的輸入。程式碼在<project>\scene.c，架構大致如下:

```
// poll SDL input event
while (SDL_PollEvent(&ev))
{
    switch (ev.type)
    {
     // key events
    case SDL_KEYDOWN:
          ...
        break;
```

```
    case SDL_KEYUP:
          ...
          break;

     // touch panel events
    case SDL_FINGERDOWN:
          ...
        break;

    case SDL_FINGERUP:
          ...
        break;

     // touch panel gesture events
     case SDL_SLIDEGESTURE:
        switch (ev.dgesture.gestureId)
        {
        case SDL_TG_LEFT:
              ...
            break;

        case SDL_TG_UP:
              ...
            break;

        case SDL_TG_RIGHT:
              ...
            break;

        case SDL_TG_DOWN:
              ...
            break;
        }
        break;
    }
}
...
```

6. libcurl

在我們的應用中，大致有三種方式使用 libcurl: 發出 HTTP 需求、發出 HTTP 需求並取得回應，以及 FTP 檔案下載。

6.1 發出 HTTP 需求

下例示範單純發出 HTTP 需求的應用，就像是在網頁瀏覽器中直接鍵入網址一樣。範例中示範了發出開門的命令到別的裝置上。

```
#include "curl/curl.h"
...

CURL* curl = NULL;
CURLcode res = CURLE_OK;

// init curl
curl = curl_easy_init();
if (curl)
{
     // set url
     curl_easy_setopt(curl, CURLOPT_URL, "http://192.168.1.20:80/open_door?ro=01-02-03-04-05-06");

     // output debug logs
     curl_easy_setopt(curl, CURLOPT_VERBOSE, 1L);

     // perform the request, res will get the return code
     res = curl_easy_perform(curl);
     if (CURLE_OK != res)
          printf("curl_easy_perform() fail: %d\n", res);

     // clean up curl
     curl_easy_cleanup(curl);
}
```

6.2 取得 HTTP 回應

Libcurl 利用回呼函式來接收回應的資料，例如定義如下:

```
#define BUFSIZE 8

// buffer definition
typedef struct
{
    uint8_t buf[BUFSIZE];
    size_t size;
} Buffer;

// callback when data received.
```

```
static size_t WriteBufferData(void *ptr, size_t size, size_t nmemb, void *data)
{
    size_t realsize = size * nmemb;
    Buffer* buf = (Buffer*)data;

    if (buf->size + realsize > BUFSIZE)
    {
        printf("out of buffer: %d\n", buf->size + realsize - BUFSIZE);
        realsize = BUFSIZE - buf->size;
    }

    memcpy(&(buf->buf[buf->size]), ptr, realsize);
    buf->size += realsize;
    buf->buf[buf->size] = 0;

    return realsize;
}
```

在使用 libcurl 時把相關的回呼函式與接收用的參數傳給 libcurl，即可接收回應的資料，範例如下。下面示範了要求室內機檢查卡片密碼的要求與取得檢查的回應。

```
CURL* curl = NULL;
CURLcode res = CURLE_OK;
Buffer buf;

buf.size = 0;

// init curl
curl = curl_easy_init();
if (curl)
{
     // set url
     curl_easy_setopt(curl, CURLOPT_URL,
"http://192.168.1.20:80/check_ic?ic=1234567890&ro=01-02-03-04-05-06");

     // set callback function for receiving data
     curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, WriteBufferData);

     // set receiving buffer for callback function
     curl_easy_setopt(curl, CURLOPT_WRITEDATA, &buf);

     // perform the request, res will get the return code
     res = curl_easy_perform(curl);
     if (CURLE_OK == res)
     {
          // the received data will be in buf.buf, size is buf.size
          if (strcmp(buf.buf, "OK") == 0)
               printf("Check Success.\n");
     }
```

```
    // clean up curl
    curl_easy_cleanup(curl);
}
```

6.3 FTP 檔案下載

首先先取得要下載檔案的長度，先準備一個空的 callback 函式:

```
static size_t throw_away(void *ptr, size_t size, size_t nmemb, void *data)
{
    /* we are not interested in the headers itself,
     so we only return the size we would have saved ... */
    return (size_t)(size * nmemb);
}
```

然後發出取得檔案長度的需求，範例如下:

```
CURL* curl = NULL;
CURLcode res = CURLE_OK;
double filesize = 0.0;

// init curl
curl = curl_easy_init();
if (curl)
{
     // set url
     curl_easy_setopt(curl, CURLOPT_URL, "ftp://192.168.1.5/doorbell/ITEPKG03.PKG");

     // do the download request without getting the body
     curl_easy_setopt(curl, CURLOPT_NOBODY, 1L);

     // callback that receives header data
     curl_easy_setopt(curl, CURLOPT_HEADERFUNCTION, throw_away);

     // perform the request, res will get the return code
     res = curl_easy_perform(curl);
     if (CURLE_OK == res)
     {
          // get file size
          res = curl_easy_getinfo(curl, CURLINFO_CONTENT_LENGTH_DOWNLOAD, &filesize);
          if ((CURLE_OK == res) && (filesize > 0.0))
          {
               printf("filesize: %0.0f bytes\n", filesize);
          }
     }

     // clean up curl
     curl_easy_cleanup(curl);
}
```

得到檔案長度後，就可以預先配置好記憶體來接收檔案。與接收 HTTP 回應類似，先準備好接收的 Buffer 與回呼函式如下:

```
// ftp buffer
struct FtpBuf
{
    uint8_t* buf;
    uint32_t pos;
};

// ftp callback when data received.
static size_t FtpWrite(void *buffer, size_t size, size_t nmemb, void *stream)
{
    struct FtpBuf* out = (struct FtpBuf*)stream;
    size_t s;

    assert(out->buf);
    s = size * nmemb;
    memcpy(&out->buf[out->pos], buffer, s);
    out->pos += s;
    return s;
}
```

然後使用 curl 發出 ftp 命令如下。下例示範了到控制伺服器下載更新軟體的代碼。

```
CURL *curl;
CURLcode res;
struct FtpBuf ftpBuf;

// create receiving buffer
ftpBuf.buf = malloc(filesize);
ftpBuf.pos = 0;

// init curl
curl = curl_easy_init();
if (curl)
{
     // set url
     curl_easy_setopt(curl, CURLOPT_URL, "ftp://192.168.1.5/doorbell/ITEPKG03.PKG");

     // set callback function
     curl_easy_setopt(curl, CURLOPT_WRITEFUNCTION, FtpWrite);

     //set receiving buffer as parameter of callback function
     curl_easy_setopt(curl, CURLOPT_WRITEDATA, &ftpBuf);

     // perform the request, res will get the return code
     res = curl_easy_perform(curl);
     if (CURLE_OK == res)
```

```
    {
        // the download file will be in ftpBuf.buf, size is ftpBuf.pos
    }
    // clean up curl
    curl_easy_cleanup(curl);
}
```

7. Libmicrohttpd

在本專案中使用到了接受請求與回傳檔案的功能。和 Curl 的主要区别，Mircohttpd 扮演 Server 端的角色

7.1 初始化與結束

初始化代碼如下:

```
#include "platform.h"
#include "microhttpd.h"
…

struct MHD_Daemon* webServerDaemon;

webServerDaemon = MHD_start_daemon(MHD_USE_SELECT_INTERNALLY, 80, NULL, NULL,
                        &ahc_echo, NULL, NULL, MHD_OPTION_END);
if (webServerDaemon == NULL)
    printf("Start Web Server fail\n");
```

其中 ahc_echo 是接受 HTTP 請求時的回呼函式，架構如下:

```
static int
ahc_echo (void *cls,
          struct MHD_Connection *connection,
          const char *url,
          const char *method,
          const char *version,
          const char *upload_data, size_t *upload_data_size, void **ptr)
{
    ...
}
```

其中 url 參數即是客戶端發送請求的 URL。結束時代碼如下:

```
MHD_stop_daemon(webServerDaemon);
```

7.2 接受請求

在回呼函式中判斷 url 以決定是何種請求。使用本函式庫內建的函式 MHD_lookup_connection_value() 來解析請求的參數。處理完後再回應處理結果。以下示範接受新訊息的通知請求:

```
// if url is http://192.168.1.20/message_new?count=5
if (strcmp(url, "/message_new") == 0)
{
    const char* val;
    int count;

     // get count parameter
```

```
    val = MHD_lookup_connection_value(connection, MHD_GET_ARGUMENT_KIND, "count");
    if (val == NULL)
        return MHD_NO;

     // count should be 5
    count = atoi(val);
     printf("count=%d\n", count);

     // return empty response with code 200
    response = MHD_create_response_from_buffer (0, NULL, MHD_RESPMEM_PERSISTENT);
    ret = MHD_queue_response (connection, MHD_HTTP_OK, response);
    MHD_destroy_response (response);

    return ret;
}
```

7.3 回傳檔案

在回傳檔案時需要使用 2 個回呼函式，一個供讀取檔案用，另一個供結束讀取檔案時，釋放資源用。示範如下:

```
// callback when reading file
static ssize_t
file_reader (void *cls, uint64_t pos, char *buf, size_t max)
{
  FILE *file = cls;

  (void) fseek (file, pos, SEEK_SET);
  return fread (buf, 1, max, file);
}

// callback when finishing read file
static void
file_free_callback (void *cls)
{
  FILE *file = cls;
  fclose (file);
}
```

在遇到回傳檔案的需求時，建立回應示範如下:

```
FILE *file;

file = fopen("C:/test.bin", "rb");
if (file)
{
     // create response
     response = MHD_create_response_from_callback (buf.st_size, 32 * 1024,     /* 32k PAGE_NOT_FOUND size
*/
```

```
                                                &file_reader, file,
                                                &file_free_callback);
    if (response)
    {
        ret = MHD_queue_response (connection, MHD_HTTP_OK, response);
        MHD_destroy_response (response);
    }
}
```

8. Libxml2

開始先設定內容讓 Libxml2 讀取與解析，然後再使用 Libxml2 提供的函式來提取或設定內容。使用的重点在地址簿的解析和获取信息。

8.1 解析內容

解析 XML 檔案的代碼如下:

```
#include "libxml/xpath.h"
…

static xmlDocPtr abDoc;
static xmlXPathContext* abXpathCtx;

// parse addressbook.xml
abDoc = xmlParseFile("B:/addressbook.xml");

// used for xpath api
abXpathCtx = xmlXPathNewContext(abDoc);
```

如果是從記憶體解析 XML，代碼示範如下:

```
static xmlDocPtr weatherDoc;
static xmlXPathContext* weatherXpathCtx;
static const char xml[] = "<weather>...</weather>";

// parse weather xml from memory
weatherDoc = xmlParseMemory(xml, sizeof(xml));

// used for xpath api
weatherXpathCtx = xmlXPathNewContext(weatherDoc);
```

8.2 提取內容

我們使用 XPath 表示法來取得 XML 節點中的內容。XPath 語法可以參考 http://msdn.microsoft.com/en-us/library/ms256086(v=vs.110).aspx。
如下範例中示範了提取地址簿的版本資訊。

```
xmlXPathObject* xpathObj;
char* ret = NULL;

// get value by xpath expression
xpathObj = xmlXPathEvalExpression(BAD_CAST "/AddrList/@ver", abXpathCtx);
if (!xmlXPathNodeSetIsEmpty(xpathObj->nodesetval))
{
    // get xml node
```

```
    xmlNode* node = xpathObj->nodesetval->nodeTab[0];

     // get string of value
    ret = xmlNodeListGetString(abDoc, node->children, 1);
}
xmlXPathFreeObject(xpathObj);

// ret will be the version information

// remember to free ret
xmlFree(ret);
```

8.3 設定內容

設定內容時，使用 XPath 提取出要設定的節點，設定完後再輸出成 XML 字串。下面範例示範了設定地址簿版本，並輸出成字串。

```
xmlChar* xmlbuff = NULL;
int* size = 0;

// get value by xpath expression
xpathObj = xmlXPathEvalExpression(BAD_CAST "/DeviceInfo/addressbook", xpathCtx);
if (!xmlXPathNodeSetIsEmpty(xpathObj->nodesetval))
{
     // get xml node
    xmlNodePtr node = xpathObj->nodesetval->nodeTab[0];

     // set as <addressbook version="2">
    xmlSetProp(node, "version", "2");

    xmlXPathFreeObject(xpathObj);
}

// dump xml tree to buffer
xmlDocDumpFormatMemory(doc, &xmlbuff, size, 1);
```

9. lwIP

我們使用 lwIP 的 socket 相關函式來與控制伺服器之間傳送與接收保持連線的訊號。

9.1 設定連線

設定連線的範例碼如下:

```
#include <sys/socket.h>
…

static int networkSocket;
struct sockaddr_in sockAddr;
int val = 1;

// create socket
networkSocket = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP);
if (networkSocket != -1)
{
     // set to asynchronous mode
     ioctlsocket(networkSocket, FIONBIO, &val);

     // init bind parameters
     memset((char*)&sockAddr, 0, sizeof(sockAddr));
     sockAddr.sin_family = AF_INET;

     // target to any
     sockAddr.sin_addr.s_addr = htonl(INADDR_ANY);

     // port to 49201
     sockAddr.sin_port = htons(49201);

     // bind
     if (bind(networkSocket, (struct sockaddr*)&sockAddr, sizeof(sockAddr)) != -1)
     {
          // binding success
     }
}
```

9.2 傳送與接收

下面程式碼示範了與控制伺服器之間傳受與接收保持連線的訊號的方式。在接收到控制伺服器傳送來的訊號後，再回送一個"*"的字元回控制伺服器。

```
struct sockaddr_in sockAddr;
int slen = sizeof(sockAddr);
int count, diff;
char buf[4];
```

```
// receive 1 byte data from center
count = recvfrom(networkSocket, buf, 1, 0, (struct sockaddr*)&sockAddr, &slen);
if (count > 0)
{
    // send "*" to 192.168.1.5
    sockAddr.sin_family = AF_INET;
    sockAddr.sin_addr.s_addr = inet_addr("192.168.1.5");

    count = sendto(networkSocket, "*", 1, 0, (const struct sockaddr*)&sockAddr, sizeof (sockAddr));
}
```

10. sntp

我們修改了原始程式碼，提供設定同步伺服器位址與同步間隔的功能。

10.1 啟動同步

程式示範如下:

```
#include "sntp.h"
…

// set sntp server address
sntp_set_server_address("192.168.1.5");

// set sync interval (secs)
sntp_set_update_delay(300);

// set port
sntp_set_port(123);

/// init sntp service
sntp_init();
```

11. iniParser

假設存在一個檔案 C:/config.ini，內容如下:

```
[tcpip]
dhcp=n
ipaddr=192.168.191.220

[doorbell]
brightness=8
```

以下示範如何存取這些設定值。

11.1 讀取

程式碼示範如下:

```
#include "iniparser/iniparser.h"
...
static dictionary* cfgIni;

// load config.ini
cfgIni = iniparser_load("C:/config.ini");
if (cfgIni)
{
     int bval, ival;
     char* sval;

     // load boolean value
    bval = iniparser_getboolean(cfgIni, "tcpip:dhcp", 1);
     // bval == 1

     // load string value
    sval = iniparser_getstring(cfgIni, "tcpip:ipaddr", "192.168.1.1");
     // sval == "192.168.191.220"

     // load string value
    sval = iniparser_getstring(cfgIni, "tcpip:netmask", "255.255.255.0");
     // sval == "255.255.255.0"

     ival = iniparser_getint(cfgIni, "doorbell:brightness", 8);
     // ival == 5
}
```

11.2 寫入

程式碼示範如下:

```
#include <stdio.h>
#include "iniparser/iniparser.h"
...

FILE* f;

// set values as string
iniparser_set(cfgIni, "tcpip:dhcp", "y");
iniparser_set(cfgIni, "tcpip:ipaddr", "192.168.1.20");
iniparser_set(cfgIni, "tcpip:netmask", "255.255.255.0");
iniparser_set(cfgIni, "doorbell:brightness", "8");

// save to file
f = fopen("C:/config.ini", "wb");
if (f)
{
     iniparser_dump_ini(cfgIni, f);
     fclose(f);
}
```

執行後 C:/config.ini 的內容會變成

```
[tcpip]
dhcp=y
ipaddr=192.168.1.20
netmask=255.255.255.0

[doorbell]
brightness=8
```

12. libredblack

在我們的專案中，只使用到了排序的功能。

12.1 排序

紅黑樹是在插入節點時就在排序，因此依序拿出時就是排序過的資料。以下代碼示範了排序的功能。

```
#include <string.h>
#include "redblack/redblack.h"
...

static int compare(const void *pa, const void *pb, const void *config)
{
    // compare by characters of string
    return strcmp(pa, pb);
}
...

int i, *ptr;
void *val;
struct rbtree *rb;

// init red-black tree
rb = rbinit(compare, NULL);

if (rb)
{
    // insert strings
    val = rbsearch("a", rb);
    val = rbsearch("b", rb);
    val = rbsearch("c", rb);
    val = rbsearch("d", rb);
    val = rbsearch("e", rb);

    // get sorted strings
    for (val=rblookup(RB_LUFIRST, NULL, rb);
            val!=NULL;
            val=rblookup(RB_LUNEXT, val, rb))
    {
        printf("%s ", val);
    }

    // clean up
    rbdestroy(rb);
}
```

13. libitu

以下描述如何操作相關的函式。

13.1 Widget

這是最基礎的可視元件，其他全部的可視元件都是繼承至此元件。在此元件上提供了一般元件共用的功能，如設定隱藏與致能屬性，取得位置、長度、高度與自訂資料等。

```
#include "ite/itu.h"
...

int x, y, width, height;
ITUText* text;
ITUButton* btn;

// get Text1 widget
text = (ITUText*)ituSceneFindWidget(&theScene, "Text1");

// set Text1 to invisible
ituWidgetSetVisible(text, false);

// set Text1 to visible
ituWidgetSetVisible(text, true);

// whether Text1 is visible or not
if (ituWidgetIsVisible(text))
{
    ...
}

// get position of Text1 on the screen
ituWidgetGetGlobalPosition(text, &x, &y);

// get width of text
width = ituWidgetGetWidth(text);

// get height of text
height = ituWidgetGetHeight(text);

printf("text: x=%d y=%d w=%d h=%d\n", x, y, width, height);

// set x as custom data of Text1
ituWidgetSetCustomData(text, x);

// get back the custom data of Text1. now the y equals x.
y = (int)ituWidgetGetCustomData(text);

// get Button1 widget
btn = (ITUButton*)ituSceneFindWidget(&theScene, "Button1");
```

```
// disable function of Button1
ituWidgetDisable(btn);

// enable function of Button1
ituWidgetEnable(btn);
```

13.2 Layer

Layer 代表每個頁面，提供了切換頁面的功能。

```
#include "ite/itu.h"
...

// get Layer1 widget
ITULayer* layer = ituSceneFindWidget(&theScene, "Layer1");

// change to Layer1
ituLayerGoto(layer);
```

13.3 Icon

Icon 是圖示元件，另外提供了讀取 JPEG 資料的功能。

```
#include <stdio.h>
#include <malloc.h>
#include "ite/itu.h"
...

// get Icon1 widget
ITUIcon* icon1 = (ITUIcon*)ituSceneFindWidget(&theScene, "Icon1");

// try to load jpeg file if exists
FILE* f = fopen("A:/test.jpg", "rb");
if (f)
{
    uint8_t* data;
    int size;

     // get file size
    fseek(f, 0, SEEK_END);
    size = ftell(f);
    fseek(f, 0, SEEK_SET);

     // alloc buffer
    data = malloc(size);
    if (data)
    {
```

```
        // read jpeg file
        size = fread(data, 1, size, f);

        // load jpeg to Icon1
        ituIconLoadJpegData(icon1, data, size);

        // free buffer
        free(data);
    }
    // close file
    fclose(f);
}
...
// release memory
ituIconReleaseSurface(icon1);
```

13.4 Text

Text 是文字元件，提供設定與取得文字的功能。

```
#include "ite/itu.h"
...

char* ptr;

// get Text1 widget
ITUText* text = (ITUText*)ituSceneFindWidget(&theScene, "Text1");

// set string to this widget
ituTextSetString(text, "Hello world!");

ptr = ituTextGetString(text);
// ptr should be "Hello world!"
```

13.5 TextBox

TextBox 是文字輸入元件，提供輸入與操作文字的功能。

```
#include "ite/itu.h"
...

char* ptr;

// get TextBox1 widget
ITUTextBox* textbox = (ITUTextBox*)ituSceneFindWidget(&theScene, "TextBox1");

// set string to this widget
ituTextBoxSetString(textbox, "Hello world!");
```

```
// delete one word from the back, now it should be "Hello world"
ituTextBoxBack(textbox);

// append more words, now it should be "Hello world, guest"
ituTextBoxInput(textbox, ", guest");

ptr = ituTextBoxGetString(textbox);
// ptr should be Hello world, guest"
```

13.6 Button

Button 是按鈕元件，提供設定文字的功能。

```
#include "ite/itu.h"
...

// get Button1 widget
ITUButton* btn = (ITUButton*)ituSceneFindWidget(&theScene, "Button1");

// set string to this widget
ituButtonSetString(btn, "Run");

// clear string of this widget
ituButtonSetString(btn, NULL);
```

13.7 CheckBox

CheckBox 是複選框元件，提供狀態設定與查詢的功能。

```
#include "ite/itu.h"
...

// get CheckBox1 widget
ITUCheckBox* checkbox = (ITUCheckBox*)ituSceneFindWidget(&theScene, "CheckBox1");

// check the CheckBox1 widget
ituCheckBoxSetChecked(checkbox, true);

// un-check the CheckBox1 widget
ituCheckBoxSetChecked(checkbox, false);

// whether CheckBox1 is checked or not
if (ituCheckBoxIsChecked(checkbox))
{
    ...
}
```

13.8 RadioBox

RadioBox 是單選框元件，提供狀態設定與查詢的功能。

```
#include "ite/itu.h"
...

// get RadioBox1 widget
ITURadioBox* radiobox = (ITURadioBox*)ituSceneFindWidget(&theScene, "RadioBox1");

// check the RadioBox1 widget
ituRadioBoxSetChecked(radiobox, true);

// un-check the RadioBox1 widget
ituRadioBoxSetChecked(radiobox, false);

// whether RadioBox1 is checked or not
if (ituRadioBoxIsChecked(radiobox))
{
    ...
}
```

13.9 ListBox

ListBox 是表列框元件，提供重讀與設定選項的功能。

```
#include "ite/itu.h"
...

// get ListBox1 widget
ITUListBox* listbox = (ITUListBox*)ituSceneFindWidget(&theScene, "ListBox1");

// reload the content of the ListBox1 widget
ituListBoxReload(listbox, true);

// select the 3rd list of ListBox1 widget
ituListBoxSelect(listbox, 3);

// un-select the ListBox1 widget
ituListBoxSelect(listbox, -1);
```

13.10 Sprite

Sprite 是動畫元件，提供單格動畫的功能。

```
#include "ite/itu.h"
...

// get Sprite1 widget
```

```
ITUSprite* sprite = (ITUSprite*)ituSceneFindWidget(&theScene, "Sprite1");

// play the sprite from #0 frame
ituSpritePlay(sprite, 0);

// stop playing
ituSpriteStop(sprite);

// goto frame #1
ituSpriteGoto(sprite, 1);
```

13.11 ProgressBar

ProgressBar 是進度條元件，提供顯示進度的功能。

```
#include "ite/itu.h"
...

// get ProgressBar1 widget
ITUProgressBar* bar = (ITUProgressBar*)ituSceneFindWidget(&scene, "ProgressBar1");

// set the value of progress bar to 50%
ituProgressBarSetValue(bar, 50);
```

13.12 Keyboard

Keyboard 是鍵盤元件，提供切換輸入目標(TextBox)的功能。

```
#include "ite/itu.h"
...

ITUTextBox* loginNameTextBox = ituSceneFindWidget(&theScene, "loginNameTextBox");
ITUTextBox* loginPasswordTextBox = ituSceneFindWidget(&theScene, "loginPasswordTextBox");
ITUKeyboard* loginKeyboard = ituSceneFindWidget(&theScene, "loginKeyboard");

// switch input target of loginKeyboard to loginNameTextBox
loginKeyboard->target = (ITUWidget*)loginNameTextBox;

// switch input target of loginKeyboard to loginPasswordTextBox
loginKeyboard->target = (ITUWidget*)loginPasswordTextBox;
```

13.13 TrackBar

TrackBar 是滑桿元件，提供調整刻度的功能。

```
#include "ite/itu.h"
...
```

```
ITUProgressBar* bar = ituSceneFindWidget(&theScene, "TrackBar1");

// set the value of track bar to 50
ituTrackBarSetValue(bar, 50);
```

13.14 Calendar

Calendar 是日曆元件，提供顯示日曆的功能。

```
#include "ite/itu.h"
...

ITUCalendar* cal = ituSceneFindWidget(&theScene, "Calendar1");

// goto today
ituCalendarToday(cal);

// goto last month
ituCalendarLastMonth(cal);

// goto next month
ituCalendarNextMonth(cal);

// goto 2016/05
ituCalendarGoto(cal, 2016, 5);
```

13.15 MeidaFileListBox

MeidaFileListBox 是媒體檔案表列框元件，提供顯示媒體檔案列表與操作的功能。

```
#include "ite/itu.h"
...

ITUMediaFileListBox* box = ituSceneFindWidget(&theScene, "MediaFileListBox1");
ITUScrollText* item;

// set file path to C:/
ituFileListSetPath(box, "C:/");

// repeat all
box-> repeatMode = ITU_MEDIA_REPEAT_ALL;

// enable random play
box-> randomPlay = 1;

// get current item
item = ituMediaFileListPlay(box);

// goto next item
```

```
item = ituMediaFileListNext(box);

// goto previous item
item = ituMediaFileListPrev(box);
```

13.16 CircleProgressBar

CircleProgressBar 是圓形進度條元件，提供顯示圓形進度的功能。

```
#include "ite/itu.h"
...

ITUCircleProgressBar* bar = ituSceneFindWidget(&theScene, "CircleProgressBar1");

// set the value of circle progress bar to 50%
ituCircleProgressBarSetValue(bar, 50);
```

13.17 ScrollBar

ScrollBar 是捲軸元件，提供捲動 ListBox 相關元件的功能。

```
#include "ite/itu.h"
...

ITUScrollListBox* bar = ituSceneFindWidget(&theScene, "ScrollBar1");

// set lenth to 5
ituScrollBarSetLength(bar, 5);

// set position to 1
ituScrollBarSetPosition(bar, 1);
```

13.18 Animation

Animation 是動畫元件，提供內插動畫的功能。

```
#include "ite/itu.h"
...

ITUAnimation* anim = ituSceneFindWidget(&theScene, "Animation0");

// set delay to 1 frame
ituAnimationSetDelay(anim, 1);

// play from keyframe #0
ituAnimationPlay(anim, 0);

// stop playing
```

```
ituAnimationStop(anim);

// goto keyframe #2
ituAnimationGoto(anim, 2);

// reset animation
ituAnimationReset(anim);
```

13.19 Wheel

Wheel 是滾輪元件，提供滾動選擇選項的功能。

```
#include "ite/itu.h"
...

ITUWheel* wheel = ituSceneFindWidget(&theScene, "Wheel1");

// goto index #1
ituWheelGoto(wheel, 1);

// goto previous item
ituWheelPrev(wheel);

// goto next item
ituWheelNext(wheel);
```

13.20 CoverFlow

CoverFlow 是翻頁元件，提供顯示與操作主選單的功能。

```
#include "ite/itu.h"
...

ITUCoverFlow* flow = ituSceneFindWidget(&theScene, "CoverFlow1");

// goto page #0
ituCoverFlowGoto(flow, 0);
```

13.21 ScrollListBox

ScrollListBox 是滑動表列框元件，提供滑動表列框的功能。

```
#include "ite/itu.h"
...

ITUScrollListBox* slistbox = ituSceneFindWidget(&theScene, "ScrollListBox1");
ITCTree* node;
```

```
// get item count per page
int i, count = ituScrollListBoxGetItemCount(slistbox);

// fill last page
node = ituScrollListBoxGetLastPageItem(slistbox);
for (i = 0; i < count; i++)
{
    ITUScrollText* scrolltext = (ITUScrollText*)node;

    ...

    node = node->sibling;
}

// fill current page
node = ituScrollListBoxGetCurrPageItem(slistbox);
for (i = 0; i < count; i++)
{
    ITUScrollText* scrolltext = (ITUScrollText*)node;

    ...

    node = node->sibling;
}

// fill next page
node = ituScrollListBoxGetNextPageItem(slistbox);
for (i = 0; i < count; i++)
{
    ITUScrollText* scrolltext = (ITUScrollText*)node;

    ...

    node = node->sibling;
}
```

13.22 ScrollMeidaFileListBox

ScrollMeidaFileListBox 是滑動媒體檔案表列框元件，提供滑動媒體檔案列表與操作的功能。

```
#include "ite/itu.h"
...

ITUScrollMediaFileListBox* box = ituSceneFindWidget(&theScene, "ScrollMediaFileListBox1");
ITUScrollText* item;

// set file path to C:/
ituFileListSetPath((ITUMediaFileListBox*)box, "C:/");

// repeat all
```

```
box->mflistbox.repeatMode = ITU_MEDIA_REPEAT_ALL;

// enable random play
box->mflistbox.randomPlay = 1;

// get current item
item = ituMediaFileListPlay((ITUMediaFileListBox*)box);

// goto next item
item = ituMediaFileListNext((ITUMediaFileListBox*)box);

// goto previous item
item = ituMediaFileListPrev((ITUMediaFileListBox*)box);
```

13.23 Meter

Meter 是儀表元件，提供滑動操作儀表的功能。

```
#include "ite/itu.h"
...

ITUMeter* meter = ituSceneFindWidget(&theScene, "Meter1");

// set meter value to 50%
ituMeterSetValue(meter, 50);
```

13.24 Scene

Scene 是場景元件，提供讀取、繪製整個 UI 系統，搜尋指定的元件，設定命令函式表、傳送指定事件給元件與執行延遲函式的功能。

```
#include "ite/itu.h"

ITUSurface* screenSurf; // screen surface
ITUScene theScene; // global scene instance

// function to delay execute
static void Function1(void)
{
    printf("Hello, world!\n");
}
...

// initialize scene
ituSceneInit(&theScene, NULL);

// set the functions table of action commands
ituSceneSetFunctionTable(&theScene, actionFunctions);
```

```
// load .itu file
ituSceneLoadFile(&theScene, "A:/test.itu");

...
// get screen surface
screenSurf = ituGetDisplaySurface();

// send timer event, redraw all the scene if necessary
if (ituSceneUpdate(&theScene, ITU_EVENT_TIMER, 0, 0, 0))
{
    // draw the scene to the screen surface
    ituSceneDraw(&theScene, screenSurf);

    // flip the screen surface
    ituFlip(screenSurf);
}

// send custom event "Custom2" with parameter "3"
ituSceneSendEvent(&theScene, ITU_EVENT_CUSTOM + 2, "3");

// try to find Sprite1 widget
ITUSprite* sprite = (ITUSprite*)ituSceneFindWidget(&theScene, "Sprite1");
if (sprite)
{
    ...
}

// delay 6 main loop time (6 * 33ms) to execute Function1
ituSceneExecuteCommand(&theScene, 6, Function1);

// clean up the scene
ituSceneExit(&theScene);
```