AN3120

使用 SAM E54 的嵌入式 Web 服务器应用

简介

嵌入式 Web 服务器是一种基于单片机的服务器，可采用 HTTP 或 HTTPS 协议进行通信，允许通过网络访问用户，从而控制和监视与其连接的设备。例如，在发电厂中，可以通过连接到 MCU（具有嵌入式 Web 服务器）的传感器来监视电力输出和温度。如果电力输出和温度不满足需求，还可以通过调整连接到 MCU 的输入和冷却系统执行器来加以控制。

以下是嵌入式 Web 服务器的优点：

- 经济高效
- 离线监视
- 随时随地访问

本文档介绍了使用三个 LwIP API 的基本 Web 服务器实现，以及基于可获取实时传感器数据和事件的 SAM E54 Xplained Pro 评估板的高级 Web 服务器应用实现。可以使用 Web 页面访问和配置相关数据，从而监视和控制板载功能。本文档还提供了相关的软件包，其中包含用于实现嵌入式 Web 服务器的应用程序代码。基本 Web 页面实现所用的示例代码在 Atmel START 中以示例形式提供。

本文档将主要侧重于 TCP 服务器连接，而非 HTTP Web 服务器。通过演示简单的 HTTP Web 服务器来解释相应功能。关于 Web 页面设计所需的 HTML、CSS 和 JavaScript 文件，本文档中并未详细介绍。

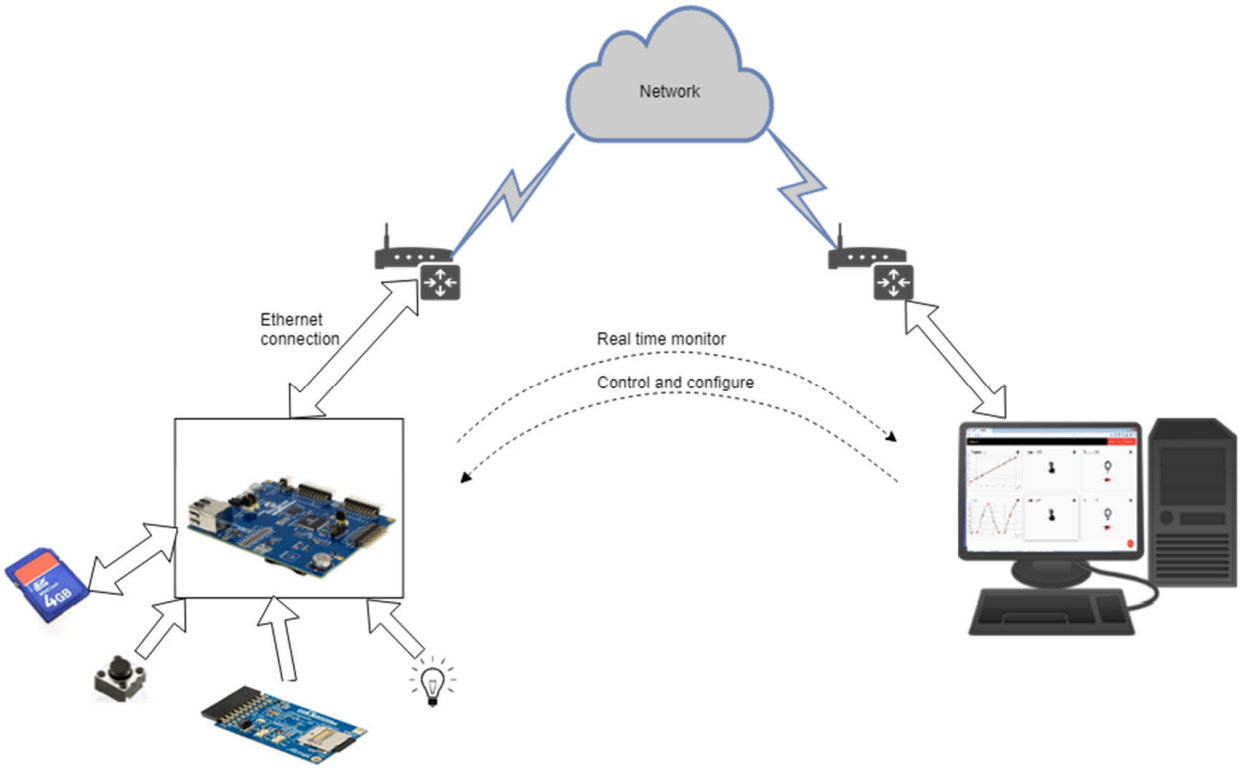
目录

1. 应用概览

基本 Web 服务器需要实现以太网接口和 HTTP 服务器。下图所示为应用概览，图中通过以太网将单片机连接到网络，并使用 GMAC 外设通过网络进行通信。PC 用于从单片机访问所需的全部数据，这些数据可用于实时监视和控制连接到单片机的设备。本应用演示使用 LwIP v1.4.0 和 FreeRTOS v8.2.3 来创建基本 Web 服务器和高级 Web 服务器。

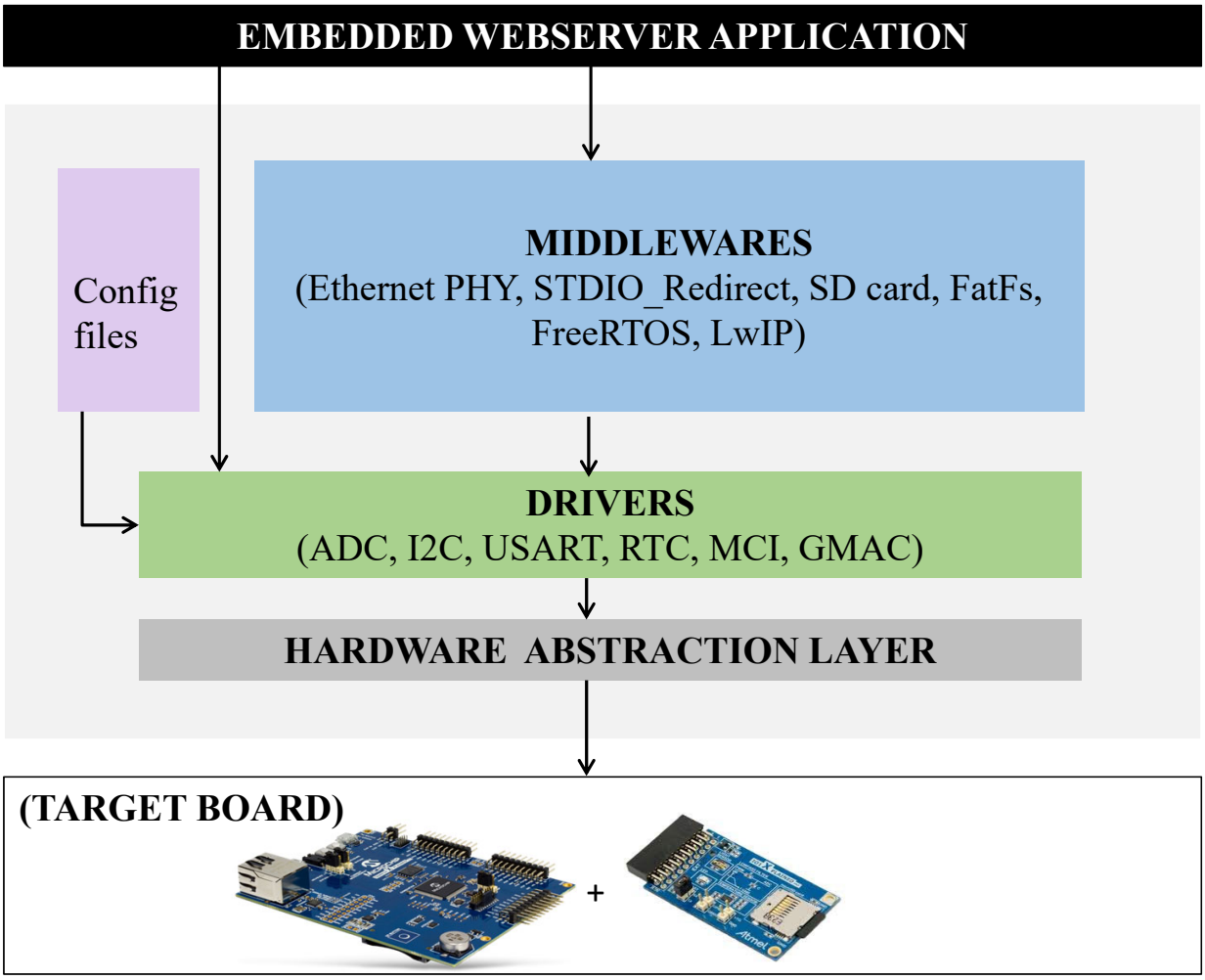
在基本 Web 服务器实现中，为用户简单介绍了三个 LwIP API：原始 API、Netconn API 和套接字 API。在高级实现中，嵌入式 Web 服务器将借助 Netconn API 向 Web 页面发送温度、光和按钮状态等数据。本应用还会将数据记录到 SD 卡中。此外，用户可根据需要来控制 LED 和配置报警。

图 1-1. 应用概览



下图所示为高级 Web 服务器应用中所使用的驱动程序和中间件的软硬件概览。

图 1-2. 软件和硬件组件概览



2. LwIP

轻量级 Internet 协议（Light Weight Internet Protocol，LwIP）是 TCP/IP 协议套件的小型独立实现。下表列出了 LwIP 的主要功能。

表 2-1. LwIP 的功能

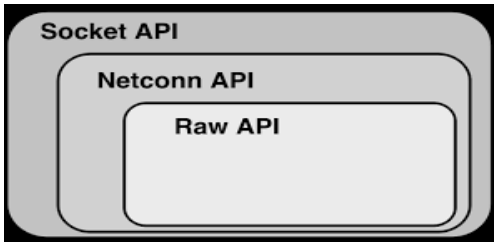
LwIP 的功能	说明
IP	Internet 协议，IPv4 和 IPv6。软件分配的逻辑地址，用于与其他协议进行网络连接（**注 1**）。
TCP	面向连接的协议，用于实现可靠的数据通信（**注 1**）。
DHCP	网络管理协议，用于为 DHCP 服务器的节点分配 IP 地址。
ICMP	错误报告协议，用于诊断网络连接。
IGMP	多播管理。
UDP	最少协议机制的无连接通信模型。
DNS	URL 的动态名称解析。
PPPoE	用于将 PPP 帧封装在以太网帧内的网络协议。

注：

1. 该功能在演示应用中使用。

LwIP 提供了三个应用程序接口（Application Program Interface，API），供程序与 TCP/IP 进行通信，如下图所示：

图 2-1. LwIP API



- **原始 API**——LwIP 的核心 API。该 API 旨在使用最少代码实现最佳性能。它使用回调处理异步事件。
- **Netconn API**——旨在使协议栈更易于使用（与事件驱动的原始 API 相比），同时仍保留零复制功能。由于该 API 需要使用线程，因此使用该 API 时需要一个操作系统。协议栈核心中的所有数据包处理（输入和输出）均在专用线程（tcp_thread）内完成。使用 Netconn API 的应用程序线程通过消息框和信号量与该核心线程进行通信。
- **套接字 API**——进程间通信（Inter-Processing Communication，IPC）编程接口，最初作为 Berkeley UNIX 操作系统的一部分提供。它是网络通信路径的本地端点的抽象表示（句柄），在 Unix 原理中以文件描述符（文件句柄）表示，用于提供输入和输出的通用接口。与其他 API 相比，套接字 API 的主要优势在于它还兼容其他 TCP/IP 协议栈。

使用 LwIP 协议栈时，关键在于根据应用对其进行配置。我们必须将 LwIP 配置为使用套接字 API、Netconn API 或原始 API 中的任何一个，并根据应用需求设置连接。

2.1 LwIP 配置

下表列出了三个 LwIP API 的配置：

表 2-2. LwIP 配置（在 Config\lwipopts.h 中可用）

选项	要配置的宏		
	NO_SYS	LWIP_NETCONN	LWIP_SOCKET 和 LWIP_COMPAT_SOCKETS
原始 API	1	0	0
Netconn API	0	1	0
套接字 API	0	1	1

注： 这些 API 适用于 LwIP 1.4.0。

2.2 静态和动态 IP 配置

LwIP 可配置为用于静态和动态 IP 配置。用户必须根据所用的 IP 配置来选用下表中提及的 LwIP 配置。

表 2-3. 准备工作

选项	静态 IP 值	DHCP 值	说明
CONF_TCPIP_STACK_INTERFACE_0_STATIC_IP（Config/lwip_macif_config.h）	1	0	使能静态 IP 配置
CONF_TCPIP_STACK_INTERFACE_0_DHCP（Config/lwip_macif_config.h）	0	1	使能动态配置
LWIP_DHCP（Config/lwipopts.h）	0	1	使能 DHCP 模块

在动态主机配置协议（Dynamic Host Configuration Protocol，DHCP）中，DHCP 服务器会自动为设备分配 IP 地址。DHCP 的 `tcpip_init_done()` 回调函数中必须包含以下代码。

```
//netif_set_up(&TCPIP_STACK_INTERFACE_0_desc); //为静态 IP 使能该代码

/* DHCP 模式。*/
if (ERR_OK != dhcp_start(&TCPIP_STACK_INTERFACE_0_desc)) {
    LWIP_ASSERT("ERR_OK != dhcp_start", 0);
}
```

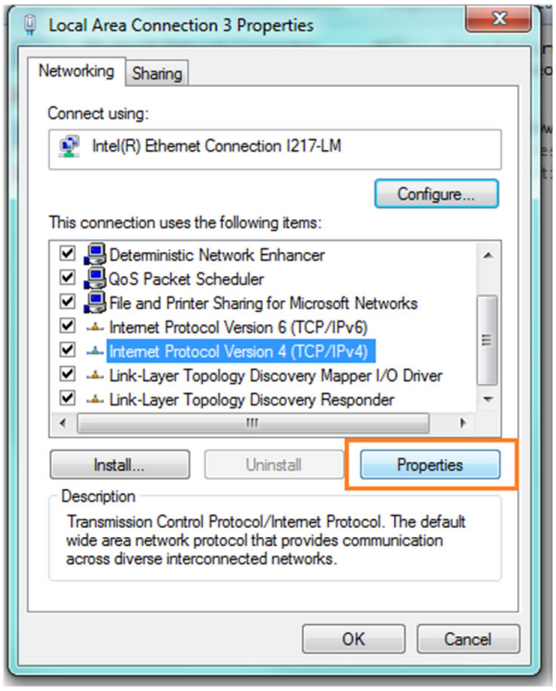
在静态 IP 中，要与 Web 服务器进行通信，应将 PC 的 IP 地址与服务器的 IP 地址配置在同一网络中。在本演示应用程序中，嵌入式 Web 服务器的 IP 地址配置为 192.168.1.100。 程序中配置的网络掩码地址为 255.255.255.0。因此，所有 192.168.1.xx 格式的 IP 地址均与服务器位于同一子网中。必须在 `tcpip_init_done()` 回调函数中使能以下代码才能使能静态 IP。

```
netif_set_up(&TCPIP_STACK_INTERFACE_0_desc);  /* 为静态 IP 使能该代码 */
```

要在装有 Windows 7 的 PC 上配置静态 IP，请按照以下步骤操作：

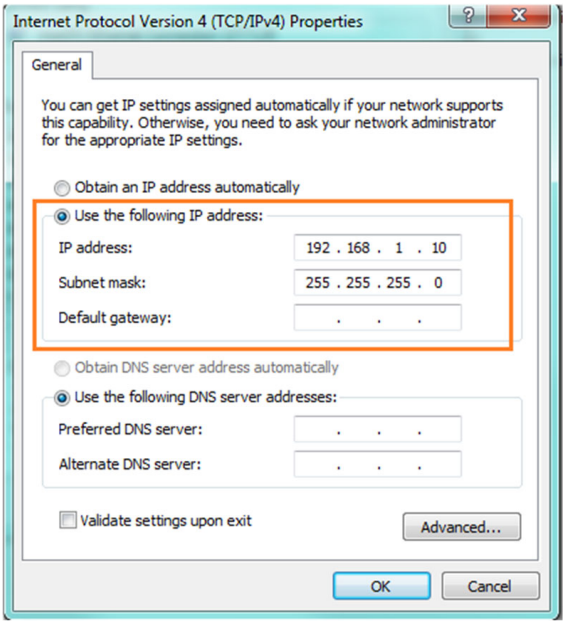
- 打开 Network and Sharing Center（网络和共享中心）。
- 单击左窗格中的 **Change Adapter Setting**（更改适配器设置）。
- 右键单击 **Local Area Connection**（局域网连接），然后选择 **Properties**（属性）。
- 选择 **Internet Protocol Version 4**（Internet 协议版本 4），然后单击 **Properties**。

图 2-2. LAN 属性选项卡



- 将 IP 地址配置为子网掩码，然后单击 **OK**（确定）。

图 2-3. TCP/IPv4 属性选项卡



3. 基本 Web 服务器实现

本章介绍如何使用全部三个 LwIP API 实现简单的基本 Web 服务器应用，帮助用户熟悉 LwIP API 及其用法，并提供每个 API 实现的详细信息。

基本 Web 服务器实现的输出将是显示文本的静态 Web 页面。分配的 IP 地址将打印在控制台中。每当用户在浏览器中输入该 IP 地址时，浏览器就会向服务器发送连接请求。建立连接后，将请求要显示的文件。收到该请求后，服务器将发送要显示的数据。

注： 基本 Web 服务器实现的示例代码作为 Atmel START 的一部分提供，可帮助用户了解三个 LwIP API。原始、Netconn 和套接字三种 API 实现的示例名称分别为 LwIP raw API example、LwIP netconn API example 和 LwIP socket API example。可以从以下位置访问这些示例：https://start.atmel.com/的 BROWSE EXAMPLES（浏览示例）选项下。

3.1 原始 API

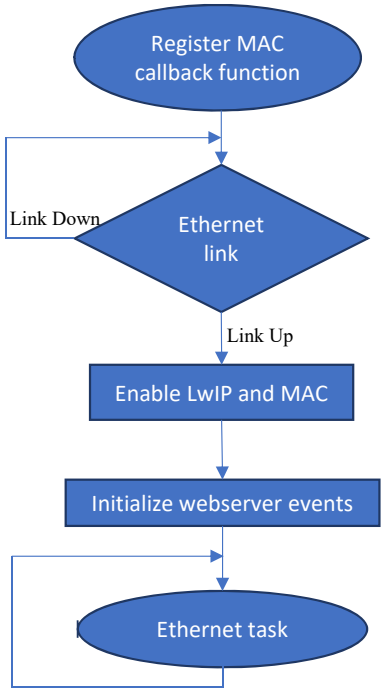
原始 API 是使用最低级别 LwIP 编程的直接接口。它是事件驱动的 API，设计为在未安装可实现零复制和接收的操作系统的情况下使用。原始 API 以一组回调函数的形式实现，随后在发生与应用程序相关的活动时由 LwIP 核心调用。本应用程序演示了一个在 Web 页面上显示文本消息的 Web 服务器实现。请参见应用笔记 AT04055：Using the lwIP Network Stack，了解有关 TCP 原始 API 函数的更多信息。

表 3-1. TCP 原始 API 函数

函数	API 名称	API 说明
TCP 连接设置	`tcp_new()`	创建一个新的 TCP PCB
	`tcp_bind()`	将 PCB 绑定到本地 IP 地址或端口
	`tcp_listen()`	使 PCB 侦听传入连接
	`tcp_accept()`	设置用于新传入连接的回调
发送 TCP 数据	`tcp_write()`	将待发送的数据排队
接收 TCP 数据	`tcp_recv()`	设置传入数据的回调
应用程序轮询	`tcp_poll()`	设置应用程序轮询回调
关闭连接和中止连接	`tcp_close()`	关闭连接
	`tcp_abort()`	中止连接

主函数用于初始化单片机、驱动程序和中间件，并按下图所示的步骤执行操作：

图 3-1. 使用原始 API 的基本 Web 服务器的应用程序流程




```
int main(void)
{
    int32_t ret;

    atmel_start_init();
    systick_enable();

    printf("\r\nRaw API implementation\r\n");
    mac_async_register_callback(&COMMUNICATION_IO, MAC_ASYNC_RECEIVE_CB,
(FUNC_PTR)mac_receive_cb);

    eth_ipstack_init();
    do {
        ret = ethernet_phy_get_link_status(&ETHERNET_PHY_0_desc, &link_up);
        if (ret == ERR_NONE && link_up) {
            break;
        }
    } while (true);
    printf("Ethernet link up\n");
    TCPIP_STACK_INTERFACE_0_init((u8_t *)MAC_ADDRESS);
    #if CONF_TCPIP_STACK_INTERFACE_0_DHCP
    /* DHCP 模式。*/
    if (ERR_OK != dhcp_start(&TCPIP_STACK_INTERFACE_0_desc)) {
        LWIP_ASSERT("ERR_OK != dhcp_start", 0);
    }
    #else
    netif_set_up(&TCPIP_STACK_INTERFACE_0_desc); //为静态 IP 使能该代码
    #endif

    /* 处理 Web 服务器事件 */
    lwip_raw_api_init();

    while (true) {
        if (recv_flag) {
            recv_flag = false;
            ethernetif_mac_input(&TCPIP_STACK_INTERFACE_0_desc);
        }
        /* LWIP 定时器——ARP、DHCP 和 TCP 等 */
        sys_check_timeouts();
```

```
        /* 打印 IP 地址信息 */
        if (link_up && TCPIP_STACK_INTERFACE_0_desc.ip_addr.addr) {
            link_up = false;
            print_ipaddress();
        }
    }
}
```

实际的 TCP 服务器初始化通过从 `main()` 函数调用 `lwip_raw_api_init()` 函数来实现。该函数将实例化新的 TCP 协议控制块（Protocol Control Block，PCB），并绑定到任何 IP 地址和端口 80。PCB 将侦听 HTTP 端口 80 的传入连接。

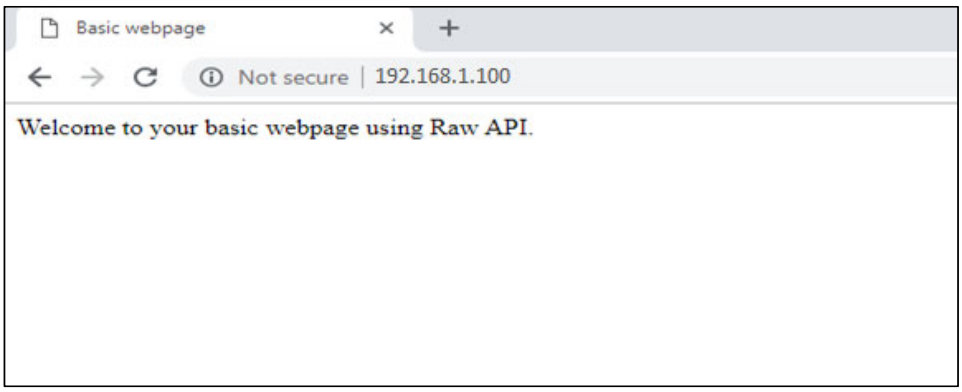
```
void lwip_raw_api_init(void)
{
    struct tcp_pcb *pcb;

    pcb = tcp_new();
    tcp_bind(pcb, IP_ADDR_ANY, HTTP_PORT);
    pcb = tcp_listen(pcb);
    if (pcb != NULL) {
        tcp_accept(pcb, http_accept);
    }
}
```

注： 有关 `lwip_raw_api_init()` 的更多信息，请参见应用笔记 AT04055：Using the lwIP Network Stack 的第 5.1.1 节 `httpd_init()`。但请注意，本文档中将 `httpd_init()` 重命名为 `lwip_raw_api_init()`。

如果使用静态 IP，则可通过 http://192.168.1.100 访问服务器主页。如果使用动态 IP，则相应的 IP 地址（`TCPIP_STACK_INTERFACE_0_desc > ip`）将显示在控制台中并且结果可供浏览，如下图所示。

图 3-2. 使用原始 API 的基本 Web 页面



3.2 Netconn API

Netconn API 是在原始 API 的基础之上构建的顺序 API。与原始 API 相比，使用更加方便，但性能有所降低且占用的存储空间增大。以下部分演示了如何使用 LwIP Netconn API 开发可同时处理多个请求的服务器。

基于 Netconn API 的程序通常使用以下线程：

- Tcpip 线程：使用原始 API 的 LwIP 核心线程。
- GMAC：负责将以太网帧从 GMAC IP 传送到 tcpip 线程的 net_if 驱动程序线程。
- 对 Netconn 连接执行打开、读取、写入和关闭操作的一个或多个用户应用程序线程。

以上线程使用报文传送（完全由 Netconn API 处理）进行通信。请参见应用笔记 AT04055：Using the lwIP Network Stack，了解有关 TCP Netconn API 函数的更多信息。下图所示为使用 Netconn API 的基本 TCP 连接的应用程序流程。

图 3-3. 使用 Netconn API 的基本 TCP 连接流程

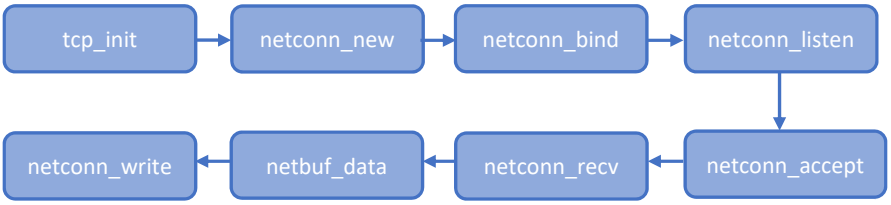


表 3-2. TCP Netconn API 函数

API 函数	说明
netconn_new()	创建新的 Netconn 连接结构
netconn_bind()	将 Netconn 结构绑定到本地 IP 地址或端口号
netconn_listen()	将 TCP Netconn 连接设置为侦听模式
netconn_accept()	基于所侦听的 TCP Netconn 连接接受传入连接
netconn_connect()	使用 IP 地址和端口号连接到远程 TCP 主机
netconn_recv()	从 Netconn 连接接收数据
netbuf_data()	指向第一个 netbuf 中的数据
netconn_write()	基于已建立的 TCP Netconn 连接发送数据
netconn_close()	关闭 TCP Netconn 连接但不删除该连接
netconn_delete()	删除现有 Netconn 连接

basic_main.c 中的 main() 调用 basic_netconn() 来实现基本 Web 服务器应用程序。它执行以下初始化：

- 创建 LED 任务和以太网任务
- 启动 FreeRTOS 调度程序

```
void basic_netconn()
{
    /* 创建 LED 任务 */
    task_led_create();

    /* 创建以太网任务 */
    if (xTaskCreate(netconn_basic_ethernet, "Ethernet_basic", TASK_ETHERNETBASIC_STACK_SIZE,
NULL, (TASK_ETHERNETBASIC_STACK_PRIORITY-1), &xCreatedEthernetBasicTask) != pdPASS) {
        while (1);
    }

    /* 启动 FreeRTOS 调度程序 */
    vTaskStartScheduler();
}
```

在 Netconn 以太网任务中，将调用 tcpip_init() 函数来初始化 LwIP 协议栈。sys_sem_wait() 函数用于阻止进程，直到协议栈完成初始化为止。新的连接结构使用 netconn_new() 创建，netconn_bind() 将连接绑定到任何 IP 地址的端口 80。netconn_listen() 侦听所有传入连接请求。

```
void netconn_basic_ethernet(void *p)
{

    struct netbuf *inbuf;
    char *rq;
    unsigned portSHORT len;
    int conn_check;
    sys_sem_t sem;
    err_t err_sem;
```

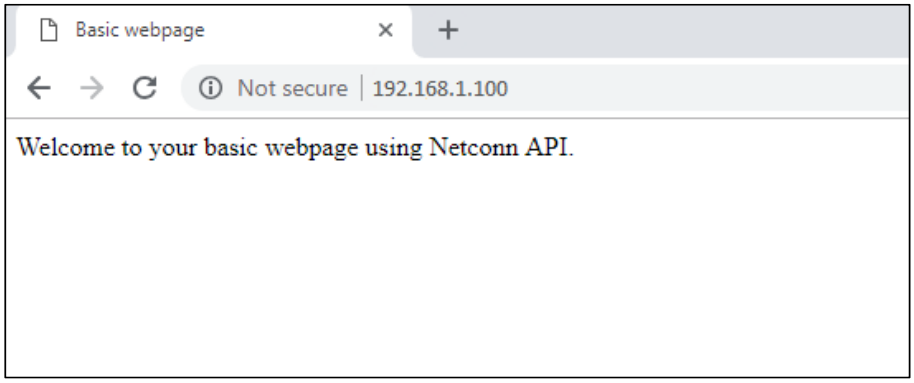
```
    err_sem = sys_sem_new(&sem, 0); /* 创建新的信号量。*/
    tcpip_init(tcpip_init_done, &sem);
    sys_sem_wait(&sem);      /* 阻止，直到 lwIP 协议栈完成初始化为止。*/
    sys_sem_free(&sem);      /* 释放信号量。*/

    print_ipaddress();

    struct netconn *conn_1, *newconn_1;
    /* 创建连接结构 */
    conn_1 = netconn_new(NETCONN_TCP);
    /* 将连接绑定到任何 IP 地址的端口 */
    conn_check = netconn_bind(conn_1, NULL, HTTP_PORT);
    while( conn_check!= ERR_OK)
    {
        LWIP_DEBUGF(LWIP_DBG_ON, ("Bind error=%d\n",conn_check));
        goto conn_close;
    }
    /* 告知连接侦听传入连接请求 */
    netconn_listen(conn_1);
    for( ;; ){
         conn_check = netconn_accept(conn_1, &newconn_1);
         while(conn_check != ERR_OK){
             LWIP_DEBUGF(LWIP_DBG_ON, ("Connection accept error=%d\n",conn_check));
             goto conn_close;
         }
         if(newconn_1 != NULL){
             conn_check = netconn_recv( newconn_1, &inbuf);
            while( conn_check != ERR_OK){
                LWIP_DEBUGF(LWIP_DBG_ON, ("Receive error=%d\n",conn_check));
                goto conn_close;
            }
            if( inbuf != NULL ){
                /* 获取指向第一个 netbuf 片段中的数据的指针
                （我们希望其中包含请求）。*/
                netbuf_data(inbuf,( void * ) &rq, &len);
                /*检查请求是否为“GET /\r\n”。*/
                if(( NULL != rq)&& ( !strncmp( rq, "GET", 3 ) )){
                    /* 发送标题。*/
                    conn_check = netconn_write(newconn_1, http_html_hdr,
sizeof(http_html_hdr), NETCONN_NOCOPY);
                    if( conn_check != ERR_OK){
                    LWIP_DEBUGF(LWIP_DBG_ON, ("Write error=%d\n",conn_check));
                    goto conn_close;
                    }
                    /* 发送实际 Web 页面。*/
                    conn_check = netconn_write(newconn_1, netconn_webpage,
sizeof(netconn_webpage), NETCONN_NOCOPY);
                    if( conn_check != ERR_OK){
                    LWIP_DEBUGF(LWIP_DBG_ON, ("Write error=%d\n",conn_check));
                    goto conn_close;
                }
            }
            netbuf_delete(inbuf);
            }
conn_close:     /*关闭连接。*/
            netconn_close(newconn_1);
            netconn_delete(newconn_1);
        }
    }
}
```

如果建立连接时出错，则将以调试消息的形式显示相应的错误代码。从 TCPIP_STACK_INTERFACE_0_desc 获知的相应 IP 地址将显示在控制台中。

图 3-4. 使用 Netconn API 的基本 Web 页面



3.3 套接字 API

LwIP 套接字 API 主要用于编程基于 Internet 的分布式应用程序。请参见应用笔记 AT04055：Using the lwIP Network Stack，了解有关 TCP 套接字 API 函数的更多信息。下图所示为使用套接字 API 的 TCP 连接的应用程序流程。

图 3-5. 使用套接字 API 的 TCP 连接流程

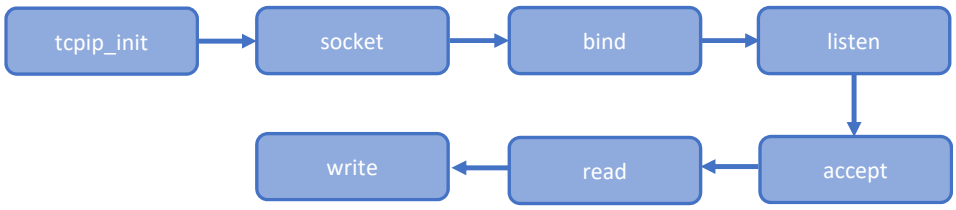


表 3-3. 套接字 API 函数

API 函数	说明
socket()	指定通信协议的类型，它将返回一个套接字描述符。
setsockopt()	设置与套接字相关的选项。
bind()	将本地协议地址分配给套接字。
listen()	将未连接的套接字转换为被动套接字，指示内核应接受传入的连接请求。
accept()	为连接等待队列中的客户端连接返回新的套接字描述符。
read()和 write()	用于与已连接的套接字通信。
send()	与 write()相似，但允许指定一些选项。
recv()	与 read()相似，但允许指定一些选项来控制数据的接收方式。
close()	关闭套接字并终止 TCP 套接字。

初始化单片机后，主函数将调用 basic_socket()。basic_socket()与 Netconn API 部分介绍的 basic_netconn()相同，但 socket_basic_ethernet()除外。在套接字 Web 服务器任务中，LwIP 协议栈初始化过程与 Netconn API 相同。不同之处在于调用的 API。AF_INET 宏用于定义地址系列。htonl 和 htons 是用于将长数据和短数据转换为大尾数格式的 API，与系统是小尾数格式还是大尾数格式无关。IP 地址将显示在控制台中，显示方式与 Netconn API 部分所述相同。

```
void socket_basic_ethernet(void *p)
{
```

```
    struct sockaddr_in address;
    int s_create, new_socket;
    int addrlen = sizeof(address);
    int opt = 1;
    int socket_check;

    sys_sem_t sem;
    err_t err_sem;
    err_sem = sys_sem_new(&sem, 0); /* 创建新的信号量。*/
    tcpip_init(tcpip_init_done, &sem);
    sys_sem_wait(&sem);     /* 阻止，直到 lwIP 协议栈完成初始化为止。*/
    sys_sem_free(&sem);     /* 释放信号量。*/

    print_ipaddress();

    /* 创建套接字 */
    s_create = socket(AF_INET, 1, 0);

    setsockopt(s_create, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,&opt, sizeof(opt));

    address.sin_family = AF_INET;
    address.sin_addr.s_addr = htonl(IPADDR_ANY);
    address.sin_port = htons( HTTP_PORT );

    /* 将连接绑定到端口 */
    socket_check = bind(s_create, (struct sockaddr *)&address,sizeof(address));
    if(socket_check<0){
        LWIP_DEBUGF(LWIP_DBG_ON, ("Bind error=%d\n",socket_check));
        goto socket_close;
    }
    /* 告知连接侦听传入的连接请求 */
    listen(s_create, 3);
    for(;;){
        new_socket = accept(s_create, (struct sockaddr *)&address,(socklen_t*)&addrlen);
        if(new_socket<=0){
            LWIP_DEBUGF(LWIP_DBG_ON, ("Connection error=%d\n",new_socket));
            goto socket_close;
        }
        socket_check = read( new_socket ,buffer, 1024);
        if(socket_check<=0){
            LWIP_DEBUGF(LWIP_DBG_ON, ("Read error=%d\n",socket_check));
            goto socket_close;
        }
        /* 检查请求是否为 HTTP“GET /\r\n”。*/
        if( !strncmp( buffer, "GET", 3 )){
            socket_check = write(new_socket , http_html_hdr , strlen(http_html_hdr));
            if(socket_check<=0){
                LWIP_DEBUGF(LWIP_DBG_ON, ("Write error=%d\n",socket_check));
                goto socket_close;
            }
            /* 发送实际 Web 页面 */
            socket_check = write(new_socket , socket_webpage , strlen(socket_webpage));
            if(socket_check<=0){
                LWIP_DEBUGF(LWIP_DBG_ON, ("Write error=%d\n",socket_check));
                goto socket_close;
            }
        }
        /* 关闭连接 */
        socket_close:
        close(new_socket);
    }
}
```

图 3-6. 使用套接字 API 的基本 Web 页面

4. 高级 Web 服务器应用

嵌入式 Web 服务器上运行的控制面板应用程序用于配置和管理系统设置。SAM E54 Xplained Pro 充当嵌入式 Web 服务器。任何客户端（Web 浏览器）均可以连接到该服务器，从而控制或监视该应用。实现此类 Web 服务器主要需要管理 LwIP 协议栈以及处理传入的连接和请求，请参见第 4.2 节.实现。

该应用的用户界面以动态 Web 页面（动态 Web 页面意味着 Web 页面将根据应用状态自动更新）的形式提供。该 Web 页面允许监视温度和光传感器状态，并根据需要控制板上 LED 和设置报警。与演示应用相对应的动态 Web 页面文件（HTML、CSS 和 JavaScript）存储在 SD 卡中，请参见第 4.2 节.实现。

注： 如果安装的 SD 卡中包含 Web 页面所需的全部文件（如第 4.2 节.实现中所述），预计所提供的应用可以正常工作。该应用已针对装有 Google Chrome v73.0.3683.86、Firefox Quantum v66.0.2 和 Internet Explorer v11.0.9600.19230 的 Windows 7 操作系统进行了测试。

4.1 应用设置

4.1.1 所需组件

硬件准备工作：

- SAM E54 Xplained Pro 评估工具包
- I/O1 Xplained Pro 扩展工具包
- SD 卡
- 以太网电缆（RJ45）
- USB 电缆

软件准备工作：

- Atmel START
- Atmel Studio 7（v7.0.1931）
- SAME54_DFP（v1.0.87）

4.1.2 设置详细信息

硬件设置详细信息如下：

- 必须将 I/O1 Xplained Pro 扩展模块连接到 SAM E54 Xplained Pro 评估板的 EXT1 扩展插座。
- 使用网络线缆将 PC 的以太网端口连接到 SAM E54 Xplained Pro 板。
- 插入 SD 卡。
- 将 USB 线缆连接到 DEBUG USB 插座，为 SAM E54 Xplained Pro 板上电。

软件设置详细信息如下：

表 4-1. 所需的 Atmel START 软件模块

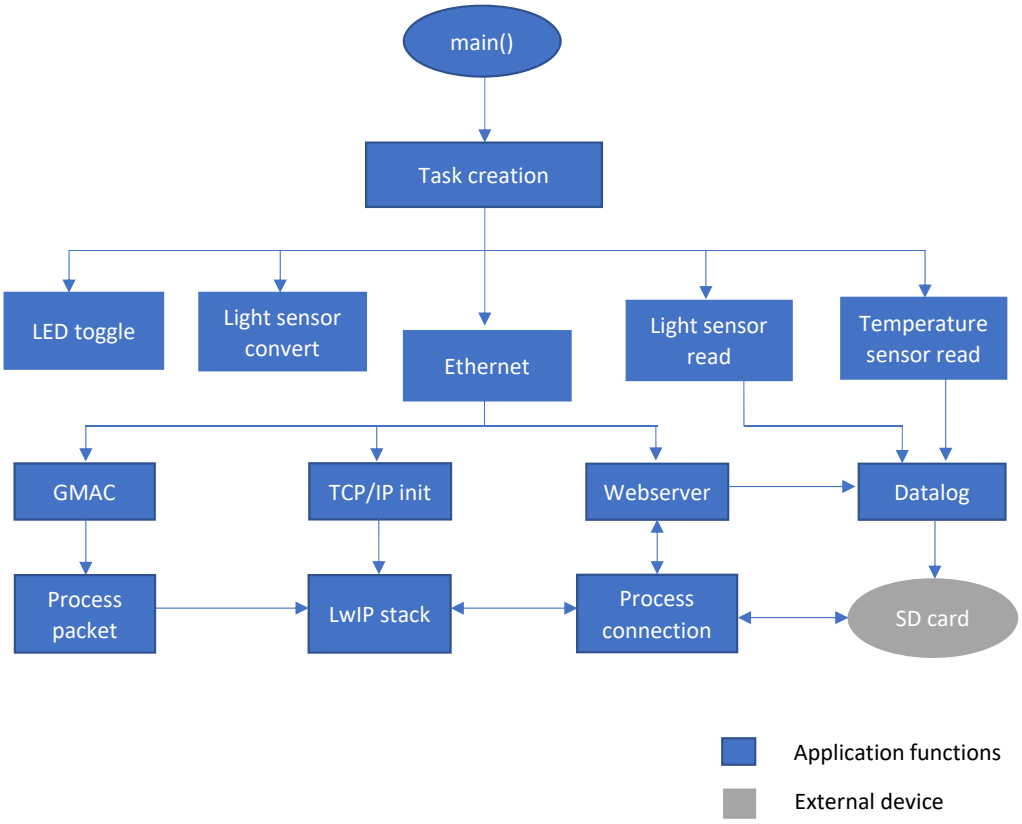
模块	说明
以太网 PHY	实现物理层功能所需的驱动程序。
TCP/IP 协议栈接口	将以太网驱动程序与 TCP/IP 协议栈接口。所用 LwIP 模块的版本为 1.4.0。
FreeRTOS v8.2.3	为应用程序提供任务、调度和任务间通信。
FatFs	小型嵌入式系统的通用 FAT 文件系统模块。
STDIO Redirect	提供一种将标准输入/输出重定向到 HAL I/O 的方法。
ADC 和 I^2C	应用中使用的传感器所需的驱动程序。

注： 如果编程`.elf`文件时出现*闪存验证失败*问题，则使用`.hex`文件或`.bin`文件来完成 SAM E54 Xplained Pro 板上的器件编程。

4.2 实现

下图所示为高级 Web 服务器的应用程序流程：

图 4-1. 应用程序流程



应用程序流程概述

开发应用程序涉及以下步骤：

- 单片机初始化
- 应用程序任务创建

下表列出了要创建的五个不同任务。

表 4-2. 应用程序任务

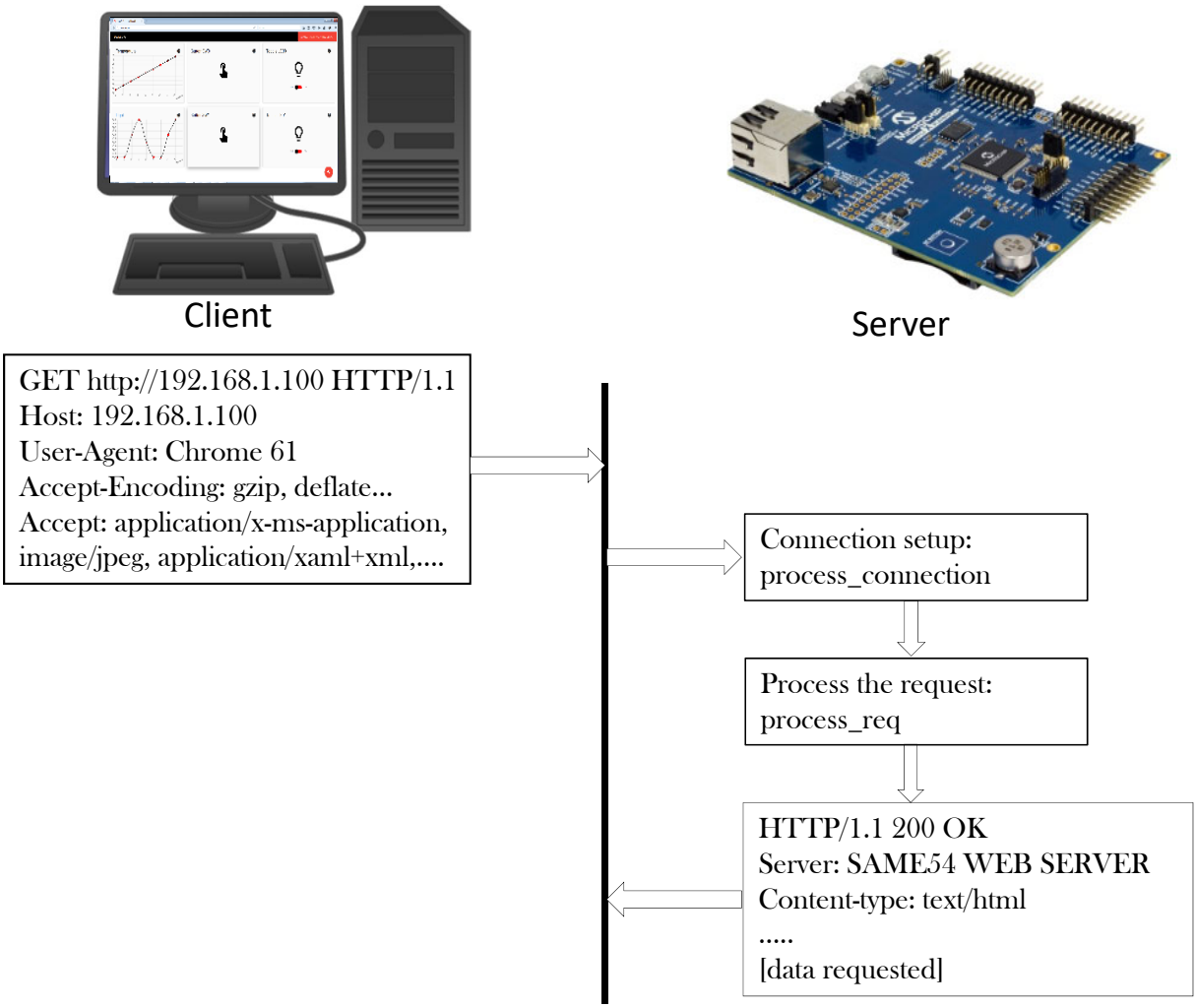
任务	说明
LED 任务	LED 以 500 ms 的间隔闪烁
光传感器转换	对 TEMT6000 光传感器输出进行 ADC 转换
光传感器读取	读取 ADC 通道并将输出转换为光传感器范围内的百分比
温度传感器读取	通过 I^2C 读取 AT30TE758 温度传感器数据
以太网任务	创建用于与 Web 页面通信的 TCP/IP 平台

- **TCP/IP 连接设置：**
  `task_ethernet` 任务执行以下功能：
  - 初始化 LwIP 模块，以太网缓冲区将已处理的数据包传输到 LwIP 协议栈。

- 使用 sys_thread_new 创建新线程 vBasicWEBServer。该任务将检查传入连接并进行处理。TCP 连接的设置步骤与使用 Netconn API 的基本 Web 服务器部分所述的步骤相同，只是所接收请求的处理方式有所不同。

- **客户端-服务器通信：**
  process_connection() 函数用于客户端和服务器之间的通信。Web 浏览器发出请求时，服务器会从 SD 卡读取数据（将其发送到 Web 浏览器），然后将数据写入 SD 卡，以用于 Web 浏览器的各种命令。下图所示为客户端和服务器之间的通信流程。

图 4-2. 客户端服务器通信流程



process_req() 函数用于解析请求以及搜索首次出现的空格字符、\t、\r 或\n。假设传感器数据没有这些令牌。字符串中包含要再次搜索“？”（用户定义的符号）的路径，如果发现该符号，则认为相应请求为获取请求或设置请求，否则认为该请求为文件请求。

- 如果获得的请求为：*GET /?get_sensor_data HTTP/1.1\r\n*，则可将该请求归类为获取请求或设置请求。
- 如果获得的请求为：*GET /logo.png HTTP/1.1\r\n*，则可将该请求归类为文件请求。

下图所示为 process_req() 函数内执行的函数调用。

图 4-3. 函数调用图

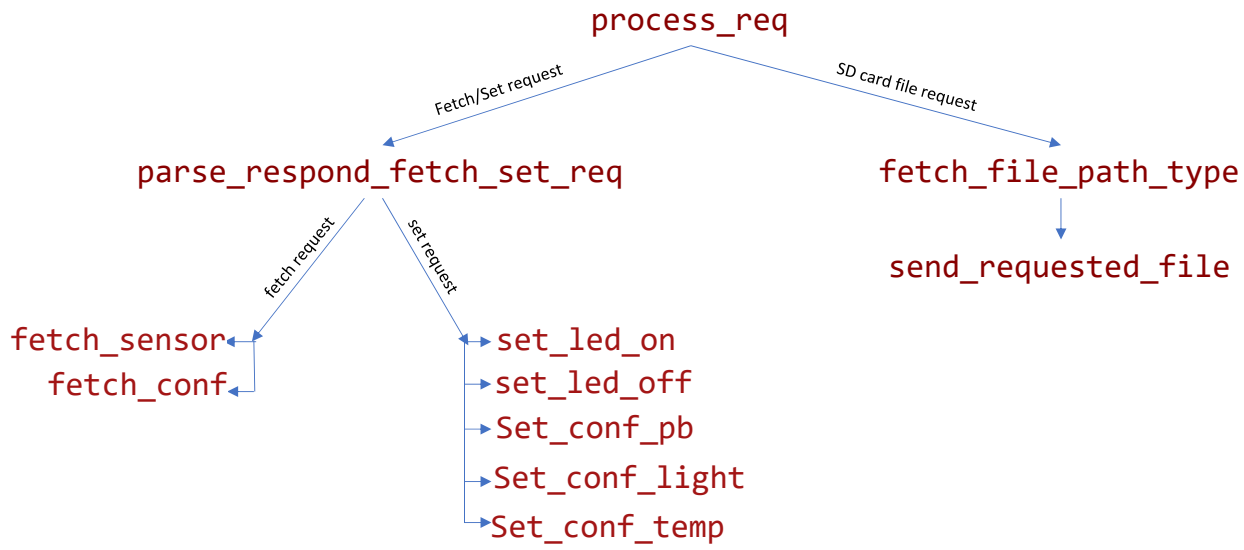


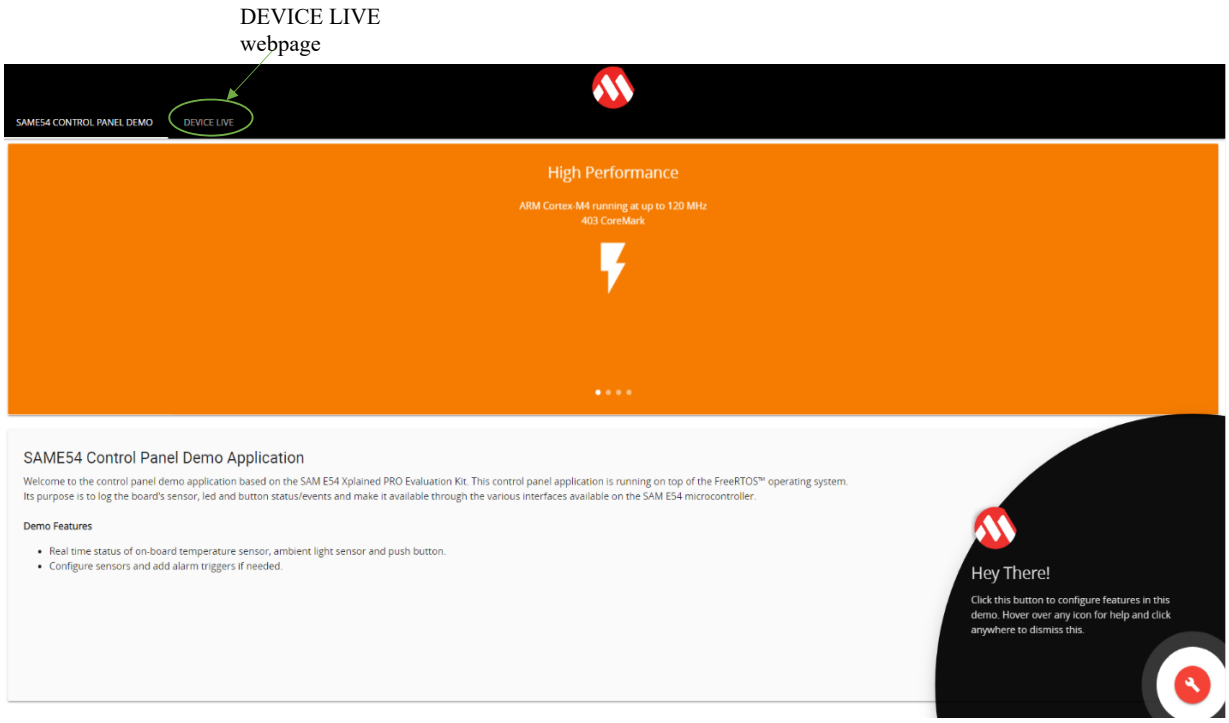
表 4-3. SD 卡文件布局

文件名	说明
index.htm	html 主页。包含页面的元素，并引用所需的支持文件。
core.css	定义 Web 页面上基本元素的样式。
core.js	用于将数据发送到服务器、从服务器请求数据以及处理输入/输出服务器的数据。
pack.css	定义其他元素的样式。
pack.js	用于渲染图形（图表和图标发光效果等）
logo.png	Microchip 徽标图标。
log.txt	数据记录文件。
config.txt	存储最小值和最大值等配置数据。

当用户请求将 `index.htm` 文件显示在浏览器上时，浏览器将请求与 Web 服务器建立连接。建立连接后，服务器将收到.html 文件的请求。索引页最初会运行图像 `pack.js`、`pack.css` 和 `core.css`，它们用于定义页面和其他元素样式。

```
<link rel = “shortcut icon” type = “image/png” href = “logo.png” >
<script type = “application/javascript” src = “pack.js” ></script>
<link href = “pack.css” type = “text/css” rel = “stylesheet” >
<link href = “core.css” type = “text/css” rel = “stylesheet” >
```

Web page display of Index page



- **应用程序用户操作：**如果服务器获得*获取*或*设置*请求，则将调用 `parse_respond_fetch_set_req()`（`Webserver/Web.c`）函数、解析请求并做出适当的响应。获取请求部分主要包括以下两个选项：
  - `fetch_sensor`：用于更新光和温度传感器值的状态以及按钮状态
  - `fetch_conf`：用于在收到请求时发送配置文件

  set 请求部分将检查以下选项：
  - `set_led_on`：点亮 LED
  - `set_led_off`：熄灭 LED
  - `set_conf_pb`：配置按钮报警
  - `set_conf_light`：配置光最小值或最大值以及报警
  - `set_conf_temp`：配置温度最小值或最大值以及报警

`core.js` 文件将检查用户请求并执行必要操作。它被放置在 `index.htm` 下进行相关处理，将在用户使用浏览器等 Web 客户端请求 URL 时触发。

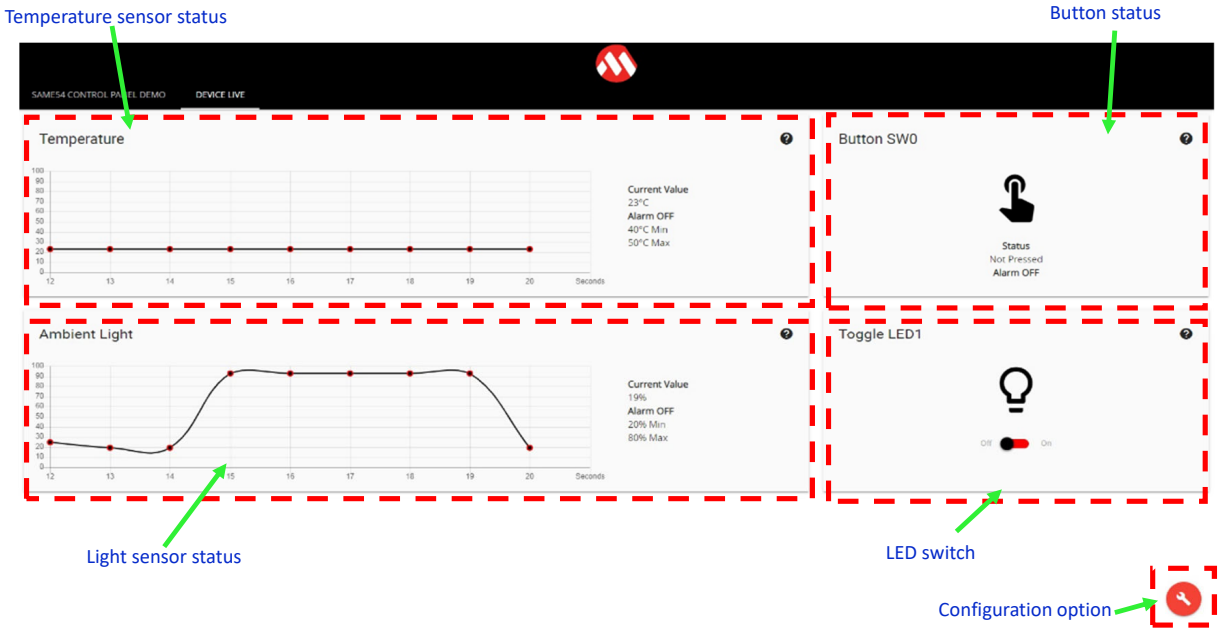
```
<script type = “application/javascript” src = “core.js”>
</script>
```

- **用户界面：**SAM E54 控制面板演示页面显示关于 SAM E54 Xplained Pro 评估工具包和演示应用程序的简要介绍。单击 Web 页面上的 DEVICE LIVE（实时设备）选项可修改视图。该页面会定期从服务器请求传感器数据，以及用户指定的控制和配置命令。
  DEVICE LIVE 页面上的状态更新如下：
  - 温度传感器状态框显示每秒更新的温度值，还会显示当前温度值（文本形式）、报警状态、最小值和最大值。
  - 同样，光传感器值也会每秒进行更新（光传感器状态图形式，如下图所示），还会显示当前光传感器值、报警状态、最小值和最大值。
  - 按钮状态框显示单片机中 SW0 开关的状态。按下按钮时，图标变为红色。

  DEVICE LIVE 页面上的控制和配置选项如下：

- 当将按钮图标切换为 ON 状态时，LED 开关盒会将 LED“点亮”；当将按钮图标切换为 OFF 状态时，LED 开关盒会将 LED“熄灭”。
- 凭借配置选项，用户可以设置传感器可以达到的最小值和最大值以及开启和关闭报警。

图 4-5. 控制面板 Web 页面



- 收集的数据以文本格式存储在 `log.txt` 文件中以供将来参考（可从 SD 卡访问）。存储数据时所采用的格式为：Sensor_idname | 日期 | 时间 | 值 | 最大值/最小值。

  注： 仅当传感器值超出设置的配置时，最后一个字段的最大值或最小值才适用。

5. 相关文章和资源

- 有关 SAM E54 Xplained Pro 评估工具包的详细信息：
  https://www.microchip.com/design-centers/32-bit/sam-32-bit-mcus/sam-e-mcus
- FreeRTOS 入门：
  http://ww1.microchip.com/downloads/en/appnotes/atmel-42382-getting-started-with-freertos-on-atmel-sam-flash-mcus_applicationnote_at04056.pdf
- Using the LwIP Network Stack：
  https://www.microchip.com/wwwAppNotes/AppNotes.aspx?appnote=en591731
- Use of the Ethernet on SAM4E-EK：
  http://ww1.microchip.com/downloads/en/AppNotes/Atmel-42134-Use-of-Ethernet-on-SAM4E-EK_AT02971_Application-Note.pdf
- TCP/IP Server-Client with CycloneTCP：
  http://ww1.microchip.com/downloads/en/AppNotes/Atmel-42738-TCPIP-Server-Client-with-CycloneTCP_AT16287_ApplicationNote.pdf

第三方链接：

- LwIP 源代码：
  https://savannah.nongnu.org/projects/lwip/
- Web 开发资源：
  https://www.w3schools.com/whatis/default.asp

6. 缩略语列表

下列缩略语适用于本文档：

- MCU——单片机（Microcontroller Unit）
- PC——个人计算机（Personal Computer）
- DHCP——动态主机配置协议（Dynamic Host Configuration Protocol）
- PCB——协议控制块（Protocol Control Block）
- TCP/IP——传输控制协议/Internet 协议（Transmission Control Protocol/Internet Protocol）
- ICMP——Internet 控制报文协议（Internet Control Message Protocol）
- UDP——用户数据报协议（User Datagram Protocol）
- ROM——只读存储器（Read Only Memory）
- LwIP——轻量级 Internet 协议（Light Weight Internet Protocol）
- HTML——超文本标记语言（Hyper Text Markup Language）

Microchip 网站

Microchip 网站（http://www.microchip.com/）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。我们的网站提供以下内容：

- **产品支持**——数据手册和勘误表、应用笔记和示例程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及归档软件
- **一般技术支持**——常见问题解答（FAQ）、技术支持请求、在线讨论组以及 Microchip 设计伙伴计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

产品变更通知服务

Microchip 的产品变更通知服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

欲注册，请访问 http://www.microchip.com/pcn，然后按照注册说明进行操作。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（ESE）
- 技术支持

客户应联系其代理商、代表或 ESE 寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http://www.microchip.com/support 获得网上技术支持。

Microchip 器件代码保护功能

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下，Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿意与关心代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是“牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了《数字器件千年版权法案（Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

法律声明

提供本文档的中文版本仅为了便于理解。请勿忽视文档中包含的英文部分，因为其中提供了有关 Microchip 产品性能和使用情况的有用信息。Microchip Technology Inc.及其分公司和相关公司、各级主管与员工及事务代理机构对译文中可能存在的任何差错不承担任何责任。建议参考 Microchip Technology Inc.的英文原版文档。

本出版物中所述的器件应用信息及其他类似内容仅为您提供便利，它们可能由更新之信息所替代。确保应用符合技术规范，是您自身应负的责任。Microchip 对这些信息不作任何明示或暗示、书面或口头、法定或其他形式的声明或担

保，包括但不限于针对其使用情况、质量、性能、适销性或特定用途的适用性的声明或担保。Microchip 对因这些信息及使用这些信息而引起的后果不承担任何责任。如果将 Microchip 器件用于生命维持和/或生命安全应用，一切风险由买方自负。买方同意在由此引发任何一切伤害、索赔、诉讼或费用时，会维护和保障 Microchip 免于承担法律责任，并加以赔偿。除非另外声明，否则在 Microchip 知识产权保护下，不得暗中或以其他方式转让任何许可证。

商标

ISBN：978-1-5224-5495-3

质量管理体系

有关 Microchip 的质量管理体系的信息，请访问 http://www.microchip.com/quality。

全球销售及服务网点

美洲	亚太地区	亚太地区	欧洲
公司总部 2355 West Chandler Blvd. Chandler, AZ 85224-6199 电话：480-792-7200 传真：480-792-7277 技术支持： http://www.microchip.com/support 网址： http://www.microchip.com **亚特兰大** 德卢斯，佐治亚州 电话：678-957-9614 传真：678-957-1455 **奥斯汀，德克萨斯州** 电话：512-257-3370 **波士顿** 韦斯特伯鲁，马萨诸塞州 电话：774-760-0087 传真：774-760-0088 **芝加哥** 艾塔斯卡，伊利诺伊州 电话：630-285-0071 传真：630-285-0075 **达拉斯** 阿迪森，德克萨斯州 电话：972-818-7423 传真：972-818-2924 **底特律** 诺维，密歇根州 电话：248-848-4000 **休斯顿，德克萨斯州** 电话：281-894-5983 **印第安纳波利斯** 诺布尔斯维尔，印第安纳州 电话：317-773-8323 传真：317-773-5453 电话：317-536-2380 **洛杉矶** 米慎维荷，加利福尼亚州 电话：949-462-9523 传真：949-462-9608 电话：951-273-7800 **罗利，北卡罗来纳州** 电话：919-844-7510 **纽约，纽约州** 电话：631-435-6000 **圣何塞，加利福尼亚州** 电话：408-735-9110 电话：408-436-4270 **加拿大 - 多伦多** 电话：905-695-1980 传真：905-695-2078	**澳大利亚 - 悉尼** 电话：61-2-9868-6733 **中国 - 北京** 电话：86-10-8569-7000 **中国 - 成都** 电话：86-28-8665-5511 **中国 - 重庆** 电话：86-23-8980-9588 **中国 - 东莞** 电话：86-769-8702-9880 **中国 - 广州** 电话：86-20-8755-8029 **中国 - 杭州** 电话：86-571-8792-8115 **中国 - 香港特别行政区** 电话：852-2943-5100 **中国 - 南京** 电话：86-25-8473-2460 **中国 - 青岛** 电话：86-532-8502-7355 **中国 - 上海** 电话：86-21-3326-8000 **中国 - 沈阳** 电话：86-24-2334-2829 **中国 - 深圳** 电话：86-755-8864-2200 **中国 - 苏州** 电话：86-186-6233-1526 **中国 - 武汉** 电话：86-27-5980-5300 **中国 - 西安** 电话：86-29-8833-7252 **中国 - 厦门** 电话：86-592-2388138 **中国 - 珠海** 电话：86-756-3210040	**印度 - 班加罗尔** 电话：91-80-3090-4444 **印度 - 新德里** 电话：91-11-4160-8631 **印度 - 浦那** 电话：91-20-4121-0141 **日本 - 大阪** 电话：81-6-6152-7160 **日本 - 东京** 电话：81-3-6880-3770 **韩国 - 大邱** 电话：82-53-744-4301 **韩国 - 首尔** 电话：82-2-554-7200 **马来西亚 - 吉隆坡** 电话：60-3-7651-7906 **马来西亚 - 槟榔屿** 电话：60-4-227-8870 **菲律宾 - 马尼拉** 电话：63-2-634-9065 **新加坡** 电话：65-6334-8870 **台湾地区 - 新竹** 电话：886-3-577-8366 **台湾地区 - 高雄** 电话：886-7-213-7830 **台湾地区 - 台北** 电话：886-2-2508-8600 **泰国 - 曼谷** 电话：66-2-694-1351 **越南 - 胡志明市** 电话：84-28-5448-2100	**奥地利 - 韦尔斯** 电话：43-7242-2244-39 传真：43-7242-2244-393 **丹麦 - 哥本哈根** 电话：45-4450-2828 传真：45-4485-2829 **芬兰 - 埃斯波** 电话：358-9-4520-820 **法国 - 巴黎** 电话：33-1-69-53-63-20 传真：33-1-69-30-90-79 **德国 - 加兴** 电话：49-8931-9700 **德国 - 哈恩** 电话：49-2129-3766400 **德国 - 海尔布隆** 电话：49-7131-72400 **德国 - 卡尔斯鲁厄** 电话：49-721-625370 **德国 - 慕尼黑** 电话：49-89-627-144-0 传真：49-89-627-144-44 **德国 - 罗森海姆** 电话：49-8031-354-560 **以色列 - 若那那市** 电话：972-9-744-7705 **意大利 - 米兰** 电话：39-0331-742611 传真：39-0331-466781 **意大利 - 帕多瓦** 电话：39-049-7625286 **荷兰 - 德卢内市** 电话：31-416-690399 传真：31-416-690340 **挪威 - 特隆赫姆** 电话：47-72884388 **波兰 - 华沙** 电话：48-22-3325737 **罗马尼亚 - 布加勒斯特** 电话：40-21-407-87-50 **西班牙 - 马德里** 电话：34-91-708-08-90 传真：34-91-708-08-91 **瑞典 - 哥德堡** 电话：46-31-704-60-40 **瑞典 - 斯德哥尔摩** 电话：46-8-5090-4654 **英国 - 沃金厄姆** 电话：44-118-921-5800 传真：44-118-921-5820