Hochschule Konstanz
Fakultät Elektrotechnik
und Informationstechnik

Verteilte Systeme

Übung zum Webserver

Mouaad Gssair

Andreas Baur

Daniel Grotz

Andreas Roth

6. Juni 2019

Table of Contents

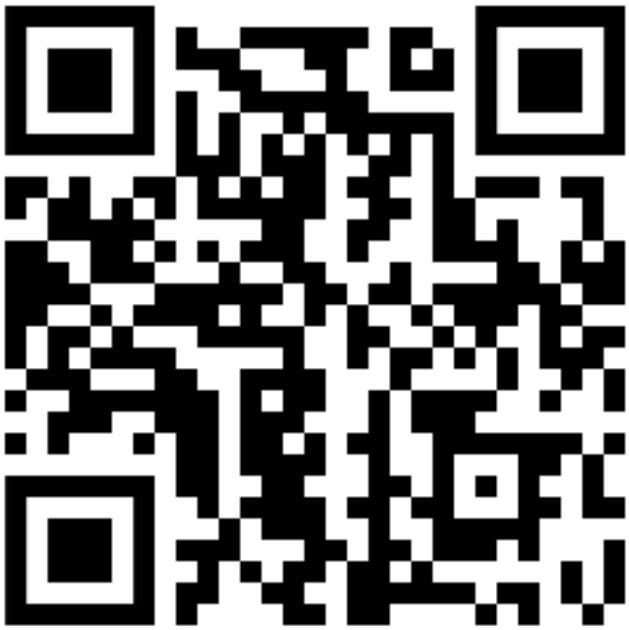
1. Einleitung

In der folgenden Laboraufgabe ist ein Embedded Webserver mit einer SD-Karteplatine für das Filesystem auf dem TM4C129EXL LaunchPad zu implementieren. Dabei soll das unvollständige Grundprojekt zuerst importiert werden um dann die noch notwendigen Konfigurationen gemäß den Vorgaben anzupassen. Eine entsprechende Anleitung zur genauen Vorgehensweise bei der Implementierung und Konfiguration ist in den folgenden Kapiteln zu finden.

2. Einrichten/Vorbereitung

2.1. Projektordner herunterladen

Link zur Git Repository
https://github.com/GMouaad/Webserver_SD-Card_Uebung



3. Implementierung

3.1. Interrupt Vektor-Tabelle anpassen

In diesem Schritt müssen in der Interrupt Vektor Tabelle folgende Interrupt-Handler angelegt werden.

lwIPEthernetIntHandler: Wird aufgerufen bei empfangen einer gültigen Nachricht über den Ethernetport.
SysTickIntHandler: Wird für den periodischen Aufruf des lwIP-Protokoll-Stacks benötigt.
disk_timerproc: Wird von FatFS benötigt um Zeitüberschreitungen zu signalisieren und Pausen zu generieren.

```
59 extern void lwIPEthernetIntHandler(void);
60 extern void SysTickIntHandler(void);
61 extern void disk_timerproc(void);
```

Anschließend muss die Interrupt-Service-Routine für den FatFS-Timer in der Datei **mmc-ek-tm4c1294xl.c** (im **Projekt-Ordner** unter \third_party\fatfs\port zu finden) implementiert werden.

```
void disk_timerproc (void)
{
//    BYTE n, s;
    BYTE n;


    n = Timer1;                         /* 100Hz decrement timer */
    if (n) Timer1 = --n;
    n = Timer2;
    if (n) Timer2 = --n;

    TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
}
```

1.1. Timer Initialisierung

Die zuvor beschriebene Interrupt-Service-Routine disk_timerproc soll alle 10ms aufegrufen werden. Hierfür wird ein Timer benötigt. Er wird so eingestellt, dass alle 10ms ein Timer-Overflow-Interrupt generiert wird. Die folgende Initialisierung des Timers erfolgt in der Datei **webserv_appl_fcns.c.**

```
void InitTimer0Int(void)
{
    uint32_t ui32_Period;

    SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
    TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
    ui32_Period = (g_ui32SysClock/100);   //120MHz/100 => 1.2Mticks -> Period of 0.01s

    TimerLoadSet(TIMER0_BASE, TIMER_A, ui32_Period - 1);     //-1 zero-count

    TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
    TimerEnable(TIMER0_BASE, TIMER_A);
    IntEnable(INT_TIMER0A);
}
```

3.2. FatFs Port-Anpassungen

Um die SD-Karte nutzen zu können, müssen die Ports für die SPI-Schnittstelle an das verwendete Boosterpack angepasst werden. Folgend sind die Port-Definitionen aufgelistet, welche in die Datei **mmc-ek-tm4c1294xl.c** (im **Projekt-Ordner** unter \third_party\fatfs\port zu finden) eingefügt werden müssen.

```
/* Definition of the Boosterpack is used */
#define VS_Webserver_Boost

#elif defined(VS_Webserver_Boost)
/* Peripheral definitions for EK-TM4C1294XL and EK-TM4C129EXL boards */
/* SSI port */
#define SDC_SSI_BASE            SSI3_BASE
#define SDC_SSI_SYSCTL_PERIPH   SYSCTL_PERIPH_SSI3

/* GPIO for SSI pins */
/* CLK pin */
#define SDC_SSI_CLK_SYSCTL_PERIPH    SYSCTL_PERIPH_GPIOQ
#define SDC_SSI_CLK_GPIO_PORT_BASE   GPIO_PORTQ_BASE
#define SDC_SSI_CLK             GPIO_PIN_0
/* TX pin */
#define SDC_SSI_TX_SYSCTL_PERIPH    SYSCTL_PERIPH_GPIOQ
#define SDC_SSI_TX_GPIO_PORT_BASE   GPIO_PORTQ_BASE
#define SDC_SSI_TX              GPIO_PIN_2
/* RX pin */
#define SDC_SSI_RX_SYSCTL_PERIPH    SYSCTL_PERIPH_GPIOQ
#define SDC_SSI_RX_GPIO_PORT_BASE   GPIO_PORTQ_BASE
#define SDC_SSI_RX              GPIO_PIN_3
/* CS pin */
#define SDC_SSI_FSS_SYSCTL_PERIPH    SYSCTL_PERIPH_GPIOA
#define SDC_SSI_FSS_GPIO_PORT_BASE   GPIO_PORTA_BASE
#define SDC_SSI_FSS             GPIO_PIN_7
```

3.3. Lange Filenamen aktivieren

Standardmäßig lassen sich beim FatFs-Modul nur kurze Filenamen im 8.3 Zeichen-Namensformat verwenden. Hierbei bestehen Dateinamen aus acht Buchstaben oder Ziffern, gefolgt von einem Punkt und der aus drei Zeichen bestehenden Namenserweiterung. Um lange File und Verzeichnisnamen bis zu einer Zeichenlänge von 255 Zeichen verwenden zu können, müssen folgende Anpassungen gemacht werden:

1) **Anpassungen in der Datei ffconfig.h** (im **SW_ROOT-Ordner** unter \third_party\fatfs\src zu finden)

Zunächst muss der richtige Zeichensatz eingestellt werden. In unserem Fall ist das der Latin1 Zeichensatz mit der CODE_PAGE 1252.

```
62
63 #define _CODE_PAGE  1252
64 /* The _CODE_PAGE specifies the OEM code page to be used on the target system.
65 /  Incorrect setting of the code page can cause a file open failure.
66 /
67 /   932  - Japanese Shift-JIS (DBCS, OEM, Windows)
```

Anschließend muss die Unterstützung für lange Filenamen durch die Definition von ENABLE_LFN freigeschaltet werden.

```
 95 #define ENABLE_LFN
 96 #ifdef ENABLE_LFN
 97 #define _USE_LFN    1       /* 0 to 3 */
 98 #else
 99 #define _USE_LFN    0       /* 0 to 3 */
100 #endif
101 #define _MAX_LFN    255     /* Maximum LFN length to handle (12 to 255) */
```

Festlegen der Sektorgröße mit 512 Byte.

```
138 #define _MAX_SS     512     /* 512, 1024, 2048 or 4096 */
```

2) **Anpassungen in der Datei mmc-ek-tm4c1294xl.c** (im **Projekt-Ordner** unter \third_party\fatfs\port zu finden)

Hier müssen die folgenden zwei Funktionen implementiert werden. Sie sind für die Konvertierung der langen Filenamen zuständig.

```
/*-----------------------------------------------------------------------*/
/*  Needed no convert 8x3 filenames in long filenames                    */
/*-----------------------------------------------------------------------*/
WCHAR ff_convert (WCHAR wch, UINT dir)
{
          if (wch < 0x80) {
                    /* ASCII Char */
                    return wch;
          }

          /* I don't support unicode it is too big! */
          return 0;
}
/*-----------------------------------------------------------------------*/
/*   Needed no convert 8x3 filenames in long filenames                   */
/*-----------------------------------------------------------------------*/
WCHAR ff_wtoupper (WCHAR wch)
{
          if (wch < 0x80) {
                    /* ASCII Char */
                    if (wch >= 'a' && wch <= 'z') {
                              wch &= ~0x20;
                     }
                      return wch;
          }

          /* I don't support unicode it is too big! */
          return 0;
}
```

3.4. File-APIs einfügen

Das Lesen gewünschter Files aus der SD-Karte erfolgt über die Funktionen f_open, f_read und f_close. Diese müssen wie folgt in die Datei **enet_fs.c** implementiert werden.

1) f_open

Die Funktion f_open öffnet eine Datei und erstellt ein Dateiobjekt. Das Dateiobjekt wird für die Identifikation und für nachfolgende Lese und Schreibvorgänge in die Datei verwendet. Die geöffnete Datei sollte nach dem Dateizugriff mit der Funktion f_close geschlossen werden.

```
if (ustrncmp(pcName, "/", 1) == 0)
{
    fresult = f_open(psFatFile, pcName + 1, FA_READ);
}
else
{
    fresult = f_open(psFatFile, pcName, FA_READ);
}
```

2) f_read

Die Funktion beginnt an der Position auf die der Lese-/ Schreibzeiger zeigt Daten aus der Datei zu lesen.

```
fresult = f_read(psFile->pextension, pcBuffer, iCount, &uiBytesRead);
```

3) f_close

Die Funktion schließt ein geöffnetes Dateiobjekt. Nachdem die Funktion erfolgreich ausgeführt wurde, ist das Dateiobjekt nicht mehr gültig und kann verworfen werden.

```
f_close(psFile->pextension);
```

3.5. HTML Dateien ergänzen und speichern

Es muss ein Index.htm oder Index.html Datei vorhanden sein. Beispielseiten sind im Ordner „fs“ zu finden. Speichern Sie die Dateien auf einer SD-Karte, und stecken Sie diese in das Booster Pack ein.

3.6. LwIP Optionen anpassen

3.6.1. DHCP

Um automatisch eine IP-Adresse zugewiesen zu bekommen, braucht man ein DHCP Server. Diese Option ist in LwIP vorhanden, man muss es allerdings aktivieren. Setzen Sie dafür den Wert von LWIP_DHCP in lwipopts.h auf 1.

```
#define LWIP_DHCP                       1           // default is 0
```

3.6.2. AutoIP

Eine weitere Option, die in LwIP vorhanden ist, ist AutoIP. Diese ermöglicht die Zuweisung von einer statische IP-Adresse, die aus der MAC-Adresse berechnet wird.

Setzen Sie den Wert von LWIP_AUTOIP auf 1, um die Option zu aktivieren.

```
#define LWIP_AUTOIP                     1           // default is 0
```

3.6.3. Weitere Optionen

Die Applikation erwartet eine Zuweisung der IP-Adresse, entweder vom DHCP Server oder von AutoIP. Um diese Wartezeit zu reduzieren, kann man die Wiederholungen der DHCP Server Proben verkleinern, indem man den Wert von LWIP_DHCP_AUTOIP_COOP_TRIES in lwipopts.h ändert.

```
#define LWIP_DHCP_AUTOIP_COOP_TRIES     1           // default is 9
```

4. Test & Lösung

4.1. Schritte

- Firmware auf dem MCU programmieren
- Ethernet Kabel und SD-Karte Adapter einstecken.

- Auf die vom DHCP oder AutoIP zugewiesene IP-Adresse warten. Die IP-Adresse wird nach einiger Sekunden auf dem Display gezeigt werden.
- IP-adresse im Browser eingeben.

5. Optionale Features

Um zusätzliche Funktionalitäten in dem Projekt hinzuzufügen, Beispielsweise eine LED Steuerung, befolgen Sie folgende Schritte.

5.1. GPIO Port konfig

Konfigurieren Sie die Ports, an der die LED angeschlossen ist. Dies sollte am Anfang des Programablaufs geschehen.

5.2. HTML Datei anpassen

Fügen Sie HTML Tags in der gpio.htm datei hinzu, und definieren Sie die Tags mit id="smth", so dass Sie später nit Javascript darauf zugreifen können.

Die Ergebnisse sollten wie folgt aussehen:

```
<div class="card">
      <p id="ledState">LED State</p>
      <div id="img">
           <img src="led_off.png" alt="data graph" width="150px" height="150">
      </div>
      <br>
      <br>
      <button onclick="toggleLED();">Toggle LED</button>
      <br>
      <br>
            LED ansteuern
 </div>
```

5.3. HTTP-GET Request (javascript)

Implementieren Sie eine Javascript Funktion, die ein HTTP-GET Request generieren kann. Die Funktion muss in einer verbundenen Javascript-Datei implementiert werden und würde wie folgt aussehen:

```
function toggleLED() {
    var req = false;
    function callbackFunction() {
```

```
        if (req.readyState == 4) {
            if (req.status == 200) {
                // TODO: Handel Response Here
               // Hint: Access HTML Tags by id and change the Content
            }
        }
    }

    if (window.XMLHttpRequest) {
        req = new XMLHttpRequest();
    }
    else if (window.ActiveXObject) {
        req = new ActiveXObject("Microsoft.XMLHTTP");
    }
    if (req) {
        // TODO: Set the appropriate URI for the GET Request
        req.open("GET", "/toggle_led?id=" + Math.random(), true);
        req.onreadystatechange = callbackFunction;
        req.send(null);
    }
}
```

5.4. HTTP-GET Handler (fs-open)

Fügen Sie eine weitere Funktion zur Verarbeitung Ihres generierten HTTP-GET Requestes ein, indem Sie eine weitere IF-Anweisung in der Funktion fs_open() hinzufügen. Diese ist in der Datei enet_fs.c zu finden.

```
// TODO: Change the URI accordingly to what you wrote in the GET Request
if (ustrncmp(pcName, "/toggle_led", 11) == 0)
        {
            char pcBuf[5];

            // TODO: toggle LED depending on the previous state
            if(!ledState){
                strcpy(pcBuf, "ON");
                ledState = 1;
                MAP_GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0,
                                 (MAP_GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_0)^
                                  GPIO_PIN_0));
            }
            else{
                strcpy(pcBuf, "OFF");
                ledState = 0;
                MAP_GPIOPinWrite(GPIO_PORTN_BASE, GPIO_PIN_0,
```

```
                                (MAP_GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_0)^
                                        GPIO_PIN_0));
            }

            psFile->data = pcBuf;
            psFile->len = strlen(pcBuf);
            psFile->index = psFile->len;
            psFile->pextension = NULL;
            return (psFile);
        }
```

Lösung

https://github.com/GMouaad/Webserver_SD-Card