



Ethernet, TCP klient a server

1 Zadání

- Připojte vývojovou desku NUCLEO-F429ZI do lokální sítě, použijte FreeRTOS a lwIP. Vyzkoušejte ping desky. Implementujte tcpcho využívající netconn API.
- Aktivujte HTTP server pro raw API, vytvořte základní souborový systém webových stránek a ověřte funkci.
- Podle vzoru tcpcho vytvořte nový thread, zajišťující telnet server na portu 23. Implementujte ovládání LD1 až LD3 pomocí textového protokolu podobného jeho v cvičení *UART komunikace s DMA, EEPROM na I2C*.
- Doplňte příkaz CLIENT, který se připojí jako HTTP klient na web <http://www.urel.feec.vutbr.cz/ip.php> a vypíše jeho obsah do terminálu.

Příkazy textového protokolu:

- HELLO
- LED1 ON|OFF
- LED2 ON|OFF
- LED3 ON|OFF
- STATUS
- CLIENT

2 Návod

2.1 Základní seznámení

- Vytvořte si pracovní kopii svého repozitáře z Githubu (Git Clone), příp. aktualizujte repozitář (Git Pull).
- Založte nový projekt přes File / New / STM32 Project / Board Selector / NUCLEO-F429ZI. Budeme využívat HAL knihovny, proto ponechte Targeted Project Type na STM32Cube. Potvrďte inicializaci všech periférií do výchozího nastavení.
- Je vhodné deaktivovat nepoužívané periferie, tj. vypnout USB_OTG_FS (Mode: Disable), aby projekt nebyl zbytečně komplikovaný.

2.2 Konfigurace ethernetu a ping

- Budeme využívat knihovnu lwIP pod systémem FreeRTOS. V CubeMX je třeba povolit FreeRTOS (CMSIS_V1) a podle doporučení přepnout časovou základnu (SYS / Timebase Source = TIM14). Veškeré úlohy budou vytvářeny dynamicky, v konfiguraci FreeRTOS pod CubeMX je tedy nevytváříme.
- Vzhledem k rozsahu projektu je potřeba zvětšit velikost haldy FreeRTOS (FREERTOS / Config Parameters / Memory management settings / TOTAL_HEAP_SIZE = 32768). Dále je vhodné zvětšit velikost zásobníku defaultTasku, v rámci kterého jsou prováděny inicializace (FREERTOS / Tasks and Queues / defaultTask / Stack Size = 1024), a povolit USE_NEWLIB_REENTRANT v Advanced settings.
- Výchozí nastavení ethernetu je správné s výjimkou MAC adresy, ta musí být pro každou vývojovou desku jedinečná a zaregistrovaná v síti FEKT. Zvolte v ETH / Parameter Settings / Ethernet MAC Address adresu **00:80:E1:FE:EC:nn**, kde **nn** je číslo vašeho PC. Např. pro PC-071 tedy bude adresa 00:80:E1:FE:EC:71. Tomu odpovídá DNS adresa VD-STM-071.urel.feec.vutbr.cz.



Mikrokontroléry a embedded systémy – cvičení

- Aktivujte lwIP (Enabled), v nastavení Platform Settings zvolte PHY driver a Solution LAN8742. Vygenerujte kód, přeložte a spusťte.

Platform proposal

BSP

| Name | IPs or Components | Found Solutions | BSP API |
|------------|-------------------|-----------------|----------------------|
| Driver_PHY | LAN8742 | LAN8742 | BSP_COMPONENT_DRIVER |

- Po připojení do sítě se rozsvítí LED kontroly na ethernetovém konektoru, ověřte funkčnost pomocí příkazu ping na adresu VD-STM-0nn.urel.feec.vutbr.cz.
- Získanou IP adresu lze ověřit pauznutím běhu programu a přidáním symbolu gnetif do okna Expressions. Po rozbalení gnetif / ip_addr / addr byste měli najít nenulovou IP adresu přiřazenou DHCP serverem v decimálním vyjádření.

2.3 Příklad tcpecho

- Příklad tcpecho (viz eLearning) je součástí balíku lwIP. Poslouchá na TCP portu 7 a veškerá přijatá data odesílá zpátky. Použitá verze je postavena na netconn API.
- V knihovně lwIP jsou podporována tři základní API (viz https://www.nongnu.org/lwip/2_0_x/raw_api.html):
 - Raw API (někdy se nazývá nativní nebo callback API) je navrženo pro použití bez operačního systému, implementuje zero-copy odesílání a příjem. Toto API také používá jádro lwIP pro interakci mezi různými protokoly. Je to jediné API dostupné při spuštění lwIP bez operačního systému. Použití je komplexní.
 - Sekvenční API (netconn API) poskytuje způsob pro běžné, sekvenční programy. Je podobné BSD socket API. Model běhu je založen na blokujících voláních open-read-write-close. Kód pro TCP/IP a aplikační programy musí být umístěn v rozdílných vláknech. Je programátorsky přívětivé. Obvyklá volba pro vlastní implementace.
 - Socket API je kompatibilní s BSD, je postaveno na netconn API. Je velice rozsáhlé, vzhledem ke své univerzálnosti zabírá hodně paměti RAM i kódu.
- Soubor tcpecho.c vložte mezi zdrojové kódy projektu (Core/Src). V souboru main.c deklarujte prototyp inicializační funkce:

```
extern void tcpecho_init(void);
```
- Tuto funkci zavolejte v rámci inicializace (tj. před nekonečnou smyčkou) ve StartDefaultTask:

```
/* Initialize tcp echo server */  
tcpecho_init();
```
- Ověřte pomocí PuTTY připojení na port 7, veškeré odeslané zprávy bude tcpecho server vracet zpět. Používejte Connection type nastavený na Raw.

2.4 HTTP server

- V konfiguraci lwIP aktivujte HTTP server (LWIP / HTTPD / LWIP_HTTPD = Enabled).
- Tento poměrně komplexní HTTP server je postavený na raw API. Webové stránky v nejjednodušším případě čte z paměti programu, kam jsou vloženy pomocí souboru fsdata.c.
- Pomocí přiloženého programu makefsdata.exe (zdrojový kód je součástí balíku lwIP) zkompilujte HTML soubory a obrázky ve složce Fs do souboru fsdata.c. Tento soubor vložte mezi **hlavičkové** soubory projektu (Core/Inc) a vyřadte z kompilovaných souborů (pravé tlačítko pro kontextové menu / Resource Configurations / Exclude from Build / zaškrtnout vše).



Mikrokontroléry a embedded systémy – cvičení

- Tento postup řeší situaci, kdy soubor chceme mít přístupný v IDE, ale nekompilujeme ho samostatně – je totiž pomocí `#include` vložený do souboru `Middlewares/Third_Party/LwIP/src/apps/http/fs.c`. Musí tedy být v prohledávané cestě, aby fungoval `#include` (proto `Core/Inc`), a zároveň nesmí být samostatně kompilován (jinak by symboly v něm uvedené byly definované vícenásobně).
- V souboru `main.c` includujte příslušný hlavičkový soubor:

```
#include "lwip/apps/httpd.h"
```
- A ve `StartDefaultTasku` inicializujte HTTP server:

```
/* Initialize HTTP server */  
httpd_init();
```
- doplněno 12/2023:** Současná verze STM32CubeIDE obsahuje chybu – po aktivování HTTP serveru přidá celou složku `Middlewares` mezi soubory vynechané při kompilaci. Je třeba ručně je ke kompilaci přidat (`Middlewares` / pravé tlačítko pro kontextové menu / `Resource Configurations` / `Exclude from Build` / odškrtnout vše).
- Projekt nyní obsahuje dva servery – `tcpecho` (port 7) a HTTP (port 80). Oba servery běží současně.
- Ověřte funkci pomocí webového prohlížeče na adrese <http://VD-STM-0nn.urel.feec.vutbr.cz>.

2.5 Telnet server

- Pro implementaci telnet serveru vyjdeme z `tcpecho` příkladu. Soubor `tcpecho.c` okopírujte pro projektu jako `telnet.c`, přejmenujte příslušné funkce (veškeré odkazy na `tcpecho` nahraďte za `telnet`) a změňte port na 23.
- Inicializaci do `main.c` doplníme stejným způsobem jako u `tcpecho` serveru. Projekt tedy obsahuje tři servery – `tcpecho` (port 7), `telnet` (port 23) a HTTP (port 80). Všechny servery běží současně.
- Dále budeme implementovat příkazy pro práci s LED viz zadání. Použijeme stejný postup jako ve cvičení *UART komunikace s DMA, EEPROM na I2C*, tj. budeme znaky zpracovávat jeden po druhém, ignorovat neplatné a na závěr ze složených řetězců parsovat příkazy. Nově je třeba předávat identifikátor spojení (`struct netconn *`), do kterého budeme odpovídat.
- Ve vláknu `telnet_thread()` budeme přijímat data typu `uint8_t *data`, vnitřní smyčku do `{}` `while` tedy přepíšeme například takto:

```
netbuf_data(buf, (void*)&data, &len);  
while (len-->0) telnet_byte_available(*data++, newconn);
```
- Funkce `telnet_byte_available()` bude reagovat na jednotlivé přijaté bajty a skládat je do pole:

```
static void telnet_byte_available(uint8_t c, struct netconn *conn)  
{  
    static uint16_t cnt;  
    static char data[CMD_BUFFER_LEN];  
  
    if (cnt < CMD_BUFFER_LEN && c >= 32 && c <= 127) data[cnt++] = c;  
    if (c == '\n' || c == '\r') {  
        data[cnt] = '\0';  
        telnet_process_command(data, conn);  
        cnt = 0;  
    }  
}
```
- Nakonec je volána funkce `telnet_process_command()`, která provede parsování a vykonání povelů:

```
static void telnet_process_command(char *cmd, struct netconn *conn);
```



Mikrokontroléry a embedded systémy – cvičení

- Není možné používat `printf()` pro standardní výstup, odpovědi je třeba vypisovat pomocí `sprintf()` do textového řetězce a ten následně odeslat pomocí `netconn` volání:
`netconn_write(conn, s, strlen(s), NETCONN_COPY);`
- Funkce `strtok()` není reentrantní a nelze tedy snadno použít ve vícevláknovém prostředí FreeRTOS (pokus skončí hardfaultem). Řešením je použít upravenou funkci `strtok_r()`, která využívá pomocný lokální parametr `char *saveptr`, viz např. https://linux.die.net/man/3/strtok_r. V případě použití `USE_NEWLIB_REENTRANT` se `strtok()` stává reentrantní a problém nenastane.
- Po připojení telnetem na vývojovou desku ověřte funkci pomocí v zadání uvedených příkazů.

2.6 HTTP klient

- Mezi příkazy protokolu doplňte povel `CLIENT`, který bude volat funkci `http_client()`. Funkce má dva parametry: odkaz na textový buffer pro odpověď a velikost tohoto bufferu.
- Funkce otevře TCP spojení na server www.urel.feec.vutbr.cz a vyžádá si stránku `ip.php`:

```
static void http_client(char *s, uint16_t size)
{
    struct netconn *client;
    struct netbuf *buf;
    ip_addr_t ip;
    uint16_t len = 0;

    IP_ADDR4(&ip, 147, 229, 144, 124);
    const char *request = "GET /ip.php HTTP/1.1\r\n"
                          "Host: www.urel.feec.vutbr.cz\r\n"
                          "Connection: close\r\n"
                          "\r\n\r\n";

    client = netconn_new(NETCONN_TCP);
    if (netconn_connect(client, &ip, 80) == ERR_OK) {
        netconn_write(client, request, strlen(request), NETCONN_COPY);

        // Receive the HTTP response
        s[0] = 0;
        while (len < size && netconn_recv(client, &buf) == ERR_OK) {
            len += netbuf_copy(buf, &s[len], size-len);
            s[len] = 0;
            netbuf_delete(buf);
        }
    } else {
        sprintf(s, "Chyba pripojeni\n");
    }
    netconn_delete(client);
}
```

- Během vývoje při ladění budou HTTP požadavky odesílány ze stejného výchozího TCP portu a spojení často nebudou ukončována korektně. Toto chování dokáže zmást firewally ve větších sítích, které neukončené TCP spojení udržují typicky po 24 hodin. Proto je vhodné randomizovat zdrojový port TCP spojení. LwIP obsahuje pro tento účel korektně funkci `rand()`, nicméně v kódu chybí inicializování generátoru funkcí `srand()`.



Mikrokontroléry a embedded systémy – cvičení

- Vlivem USE_NEWLIB_REENTRANT je třeba srand() zavolat v příslušném vlákně, v našem případě ve funkci StartDefaultTask(), ale ještě před voláním MX_LWIP_Init(). Zde bohužel chybí sekce USER CODE, takže kód bude při přegenerování odstraněn.
- Pro inicializaci pseudonáhodného generátoru využijeme periferii generátoru náhodných čísel RNG. V konfigurátoru CubeMX ji aktivujeme a před MX_LWIP_Init() vložíme kód:

```
uint32_t rnd;  
HAL_RNG_GenerateRandomNumber(&hrng, &rnd);  
srand(rnd);
```

- Odpověď na příkaz CLIENT bude pak vypadat např. takto:

```
VD-STM-071.urel.feec.vutbr.cz - PuTTY
client
HTTP/1.1 200 OK
Date: Mon, 12 Dec 2022 15:05:40 GMT
Server: Apache/2.4.6 (CentOS) OpenSSL/1.0.2k-fips PHP/5.6.40 mod_wsgi/3.4 Python/2.7.5
X-Powered-By: PHP/5.6.40
Content-Length: 232
Connection: close
Content-Type: text/html; charset=UTF-8

<!DOCTYPE html><HTML><HEAD><TITLE>ip.php @ UREL</TITLE></HEAD><PRE>
Access from: 172.25.96.201 (VD-STM-071.urel.feec.vutbr.cz)

User agent: Not specified
Server: Apache/2.4.6 at www.urel.feec.vutbr.cz
</PRE></HTML>
```