Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 5-3: Ethernet – LwIP HTTP Server Netconn RTOS LED

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

Hands-On LwIP HTTP Server Netconn RTOS LED

Objectives

The aims of this hands-on session are to

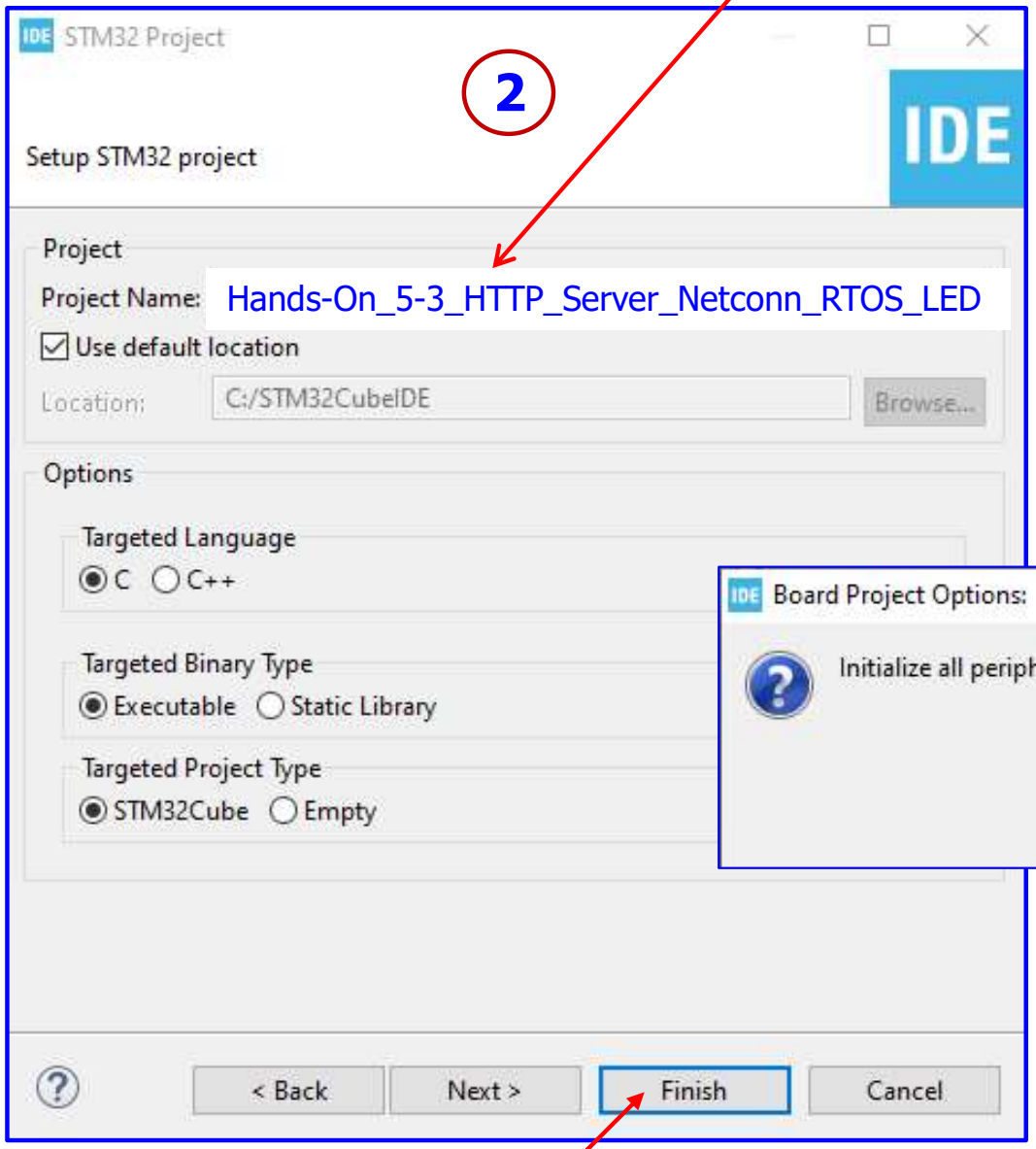
- develop a STM32 (STM32CubeIDE) project
- Implement a web (HTTP) server application based on Netconn RTOS on a STM32F767 microcontroller
- configure and program the Ethernet peripheral to make the microcontroller operating as a HTTP server and connecting web clients for loading a web page
- develop program using the htmlgen.exe software to generate the web page
- test the developed application by opening a web client on a remote PC to interact with the web server
- build up the knowledge of Ethernet application development

Example: The link below shows the use the LwIP stack to create a simple web (HTTP) server running on the Nucleo board. The web-app will allow you to interact with Nucleo LEDs and USER BUTTON, using bootstrap and jQuery.

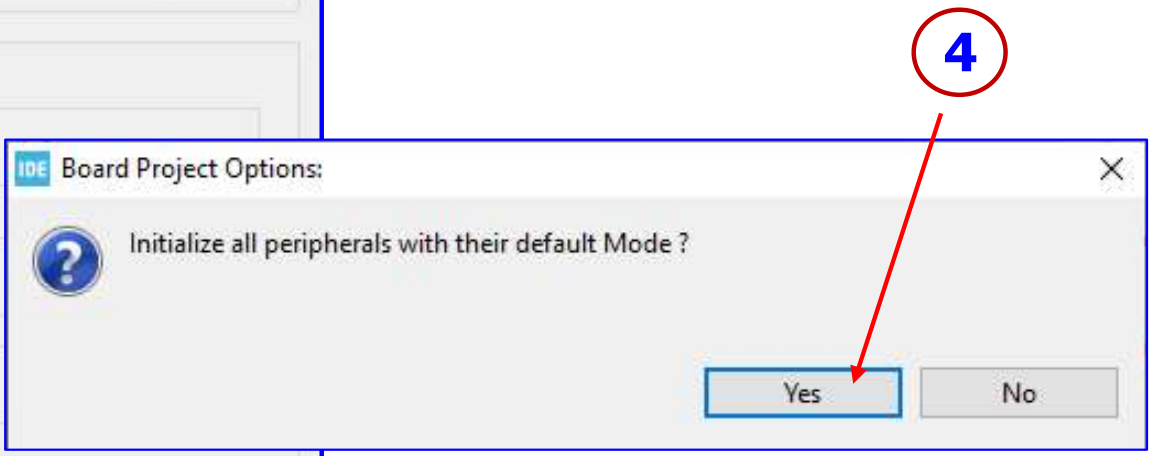
https://www.carminenoviello.com/2016/01/22/getting-started-stm32-nucleo-f746zg/

Hands-On LwIP HTTP Server Netconn RTOS LED

Create the STM32 Project: **Hands-On_5-3_HTTP_Server_Netconn_RTOS_LED**



- Run STM32CubeIDE (1)
- Select workspace: C:\STM32_CS397
- File -> Close All Editors
- Start a New STM32 Project
- Select the Nucleo-F767ZI Board



Follow all the setup steps in (5)

Hands-on_4-1_TCP_Echo_Client

(Pages 4-18)

3

Hands-On LwIP HTTP Server Netconn RTOS LED

Configure LwIP – HTTPD:

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Additional Software | Pinout

Categories | A->Z

- System Core
- Analog
- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware
  - FATFS
  - FREERTOS
  - LIBJPEG
  - LWIP
  - MBEDTLS
  - PDM2PCM
  - USB_DEVICE
  - USB_HOST

LWIP Mode and Configuration

Mode

☑ Enabled

Configuration

Reset Configuration

MDNS/TFTP | Perf/Checks | Statistics | Checksum | Debug | User Constants

General Settings | Key Options | PPP | IPv6 | HTTPD | SNMP | SNTP/SMTP

Configure the below parameters :

Search (Crtl+F) ☑ Show Advanced Parameters

HTTPD Options

LWIP_HTTPD (LwIP HTTPD Support ** CubeMX specific **)	Enabled
LWIP_HTTPD_CGI (HTTP CGI Old Style)	Enabled
LWIP_HTTPD_CGI_SSI (HTTP CGI New Style)	Disabled
LWIP_HTTPD_SSI (HTTP Server Side Includes)	Enabled
LWIP_HTTPD_SSI_RAW (HTTP SSI Tag Handler Callback)	Disabled
LWIP_HTTPD_SSI_BY_FILE_EXTENSION (HTTP SSI By File Extension)	Enabled
LWIP_HTTPD_SUPPORT_POST (HTTP POST)	Disabled
LWIP_HTTPD_MAX_CGI_PARAMETERS (Max Sent Parameters Number for CGI)	16
LWIP_HTTPD_SSI_MULTIPART (Server-Side-Includes Multipart)	Disabled
LWIP_HTTPD_MAX_TAG_NAME_LEN (Max Tag Name String Length)	16
LWIP_HTTPD_MAX_TAG_INSERT_LEN (Max Tag Inserted String Length)	192
LWIP_HTTPD_POST_MANUAL_WND (HTTP POST Manual WND)	Disabled
HTTPD_SERVER_AGENT (HTTP Server)	"lwIP/2.0.0 (http://sa...
LWIP_HTTPD_DYNAMIC_HEADERS (HTTP Dynamic Headers Creation)	Disabled
HTTPD_USE_MEM_POOL (HTTP Use Memory Pool)	Disabled
HTTPD_SERVER_PORT (HTTP Server Port)	80
HTTPD_SERVER_PORT_HTTPS (HTTPS Server Port)	443

Use default settings for other options

Hands-On LwIP HTTP Server Netconn RTOS LED

Enable **FREERTOS** by selecting the interface "**CMSIS_V1**".

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

System Core
Analog
Timers
Connectivity
Multimedia
Security
Computing
Middleware

FATFS
FREERTOS
LIBJPEG
LWIP
MBEDTLS
PDM2PCM
USB_DEVICE
USB_HOST

FREERTOS Mode and Configuration

Mode

Interface CMSIS_V1

Configuration

Reset Configuration

Tasks and Queues | Timers and Semaphores | Mutexes | Events | FreeRTOS Heap Usage

Config parameters | Include parameters | Advanced settings | User Constants

Configure the below parameters :

Search (Ctrl+F)

API	
FreeRTOS API	CMSIS v1
Versions	
FreeRTOS version	10.2.1
CMSIS-RTOS version	1.02
MPU/FPU	
ENABLE_MPU	Disabled
ENABLE_FPU	Disabled
Kernel settings	
USE_PREEMPTION	Enabled
CPU_CLOCK_HZ	SystemCoreClock
TICK_RATE_HZ	1000
MAX_PRIORITIES	7
MINIMAL_STACK_SIZE	1024 Words
MAX_TASK_NAME_LEN	16
USE_16_BIT_TICKS	Disabled

Set Minimal Stack Size: 1024

Hands-On LwIP HTTP Server Netconn RTOS LED

Increase **TOTAL_HEAP_SIZE**

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

System Core
Analog
Timers
Connectivity
Multimedia
Security
Computing
Middleware

FATFS
FREERTOS
LIBJPEG
LWIP
MBEDTLS
PDM2PCM
USB_DEVICE
USB_HOST

FREERTOS Mode and Configuration

Mode

Interface CMSIS_V1

Configuration

Reset Configuration

Tasks and Queues | Timers and Semaphores | Mutexes | Events | FreeRTOS Heap Usage

Config parameters | Include parameters | Advanced settings | User Constants

Configure the below parameters :

Search (Ctrl+F)

Memory management settings	
Memory Allocation	Dynamic / Static
TOTAL_HEAP_SIZE	63488 Bytes
Memory Management scheme	heap_4
Hook function related definitions	
USE_IDLE_HOOK	Disabled
USE_TICK_HOOK	Disabled
USE_MALLOC_FAILED_HOOK	Disabled
USE_DAEMON_TASK_STARTUP_HOOK	Disabled
CHECK_FOR_STACK_OVERFLOW	Disabled
Run time and task stats gathering related definitions	
GENERATE_RUN_TIME_STATS	Disabled
USE_TRACE_FACILITY	Disabled
USE_STATS_FORMATTING_FUNCTIONS	Disabled

Set Total Heap Size: 63488

FreeRTOS Heap Usage

Pinout & Configuration | Clock Configuration | Project M

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware
  - FATFS
  - FREERTOS
  - LIBJPEG
  - LWIP
  - MBEDTLS
  - PDM2PCM

FREERTOS Mode and Configuration

Mode

Interface CMSIS_V1

Configuration

Reset Configuration

Mutexes | Events | FreeRTOS Heap Usage
User Constants | Tasks and Queues | Timers and Semaphores
Config parameters | Include parameters | Advanced settings

Summary	
HEAP STILL AVAILABLE	59280 Bytes
TOTAL HEAP USED	4208 Bytes
Total amount for tasks	4208 Bytes
Total amount for queues	0 Bytes
Total amount for timers	0 Bytes
Total amount for mutexes and semaphores	0 Bytes
Total amount for events	0 Bytes
FreeRTOS tasks	
Idle task (FreeRTOS internal)	0 Bytes
defaultTask	4208 Bytes

For defaultTask only, it is not included the other tasks by the generated code and added code.

LwIP Settings

Hands-On LwIP HTTP Server Netconn RTOS LED

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

- System Core
- Analog
- Timers
- Connectivity
- Multimedia
- Security
- Computing
- Middleware

FATFS
FREERTOS
LIBJPEG
LWIP
MBEDTLS
PDM2PCM
USB_DEVICE
USB_HOST

LWIP Mode and Configuration

Mode

☑ Enabled

Configuration

Reset Configuration

Perf/Checks | Statistics | Checksum | Debug | User Constants | Platform S

General Settings | Key Options | PPP | IPv6 | HTTPD | SNMP | SNTP/SMTP | MDNS/TFTP

Configure the below parameters :

Search (Ctrl+F)

LwIP Version	
LwIP Version (Version of LwIP supported by CubeMX...	2.1.2
IPv4 - DHCP Options	
LWIP_DHCP (DHCP Module)	Disabled
IP Address Settings	
IP_ADDRESS (IP Address)	192.168.001.205
NETMASK_ADDRESS (Netmask Address)	255.255.255.000
GATEWAY_ADDRESS (Gateway Address)	192.168.001.001
RTOS Dependency	
WITH_RTOS (Use FREERTOS ** CubeMX specific **)	Enabled
CMSIS_VERSION (CMSIS API Version used)	CMSIS v1
RTOS_USE_NEWLIB_REENTRANT (RTOS used - 1)	Disabled
Platform Settings	
PHY Driver	Choose/LAN8742/DP83848
Protocols Options	
LWIP_ICMP (ICMP Module Activation)	Enabled
LWIP_IGMP (IGMP Module)	Disabled
LWIP_DNS (DNS Module)	Disabled
LWIP_UDP (UDP Module)	Enabled
MEMP_NUM_UDP_PCB (Number of UDP Connectio...	4
LWIP_TCP (TCP Module)	Enabled
MEMP_NUM_TCP_PCB (Number of TCP Connections)	5

For different router (gateway):

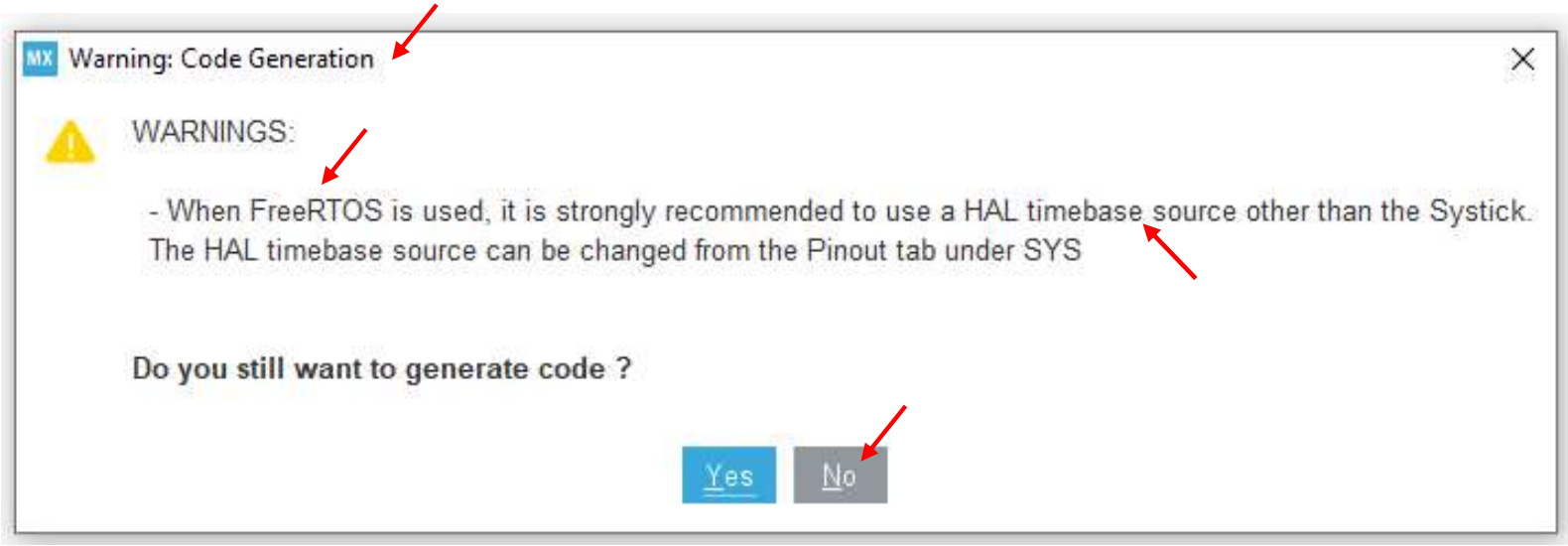
192.168.1.205
255.255.255.0
192.168.1.1

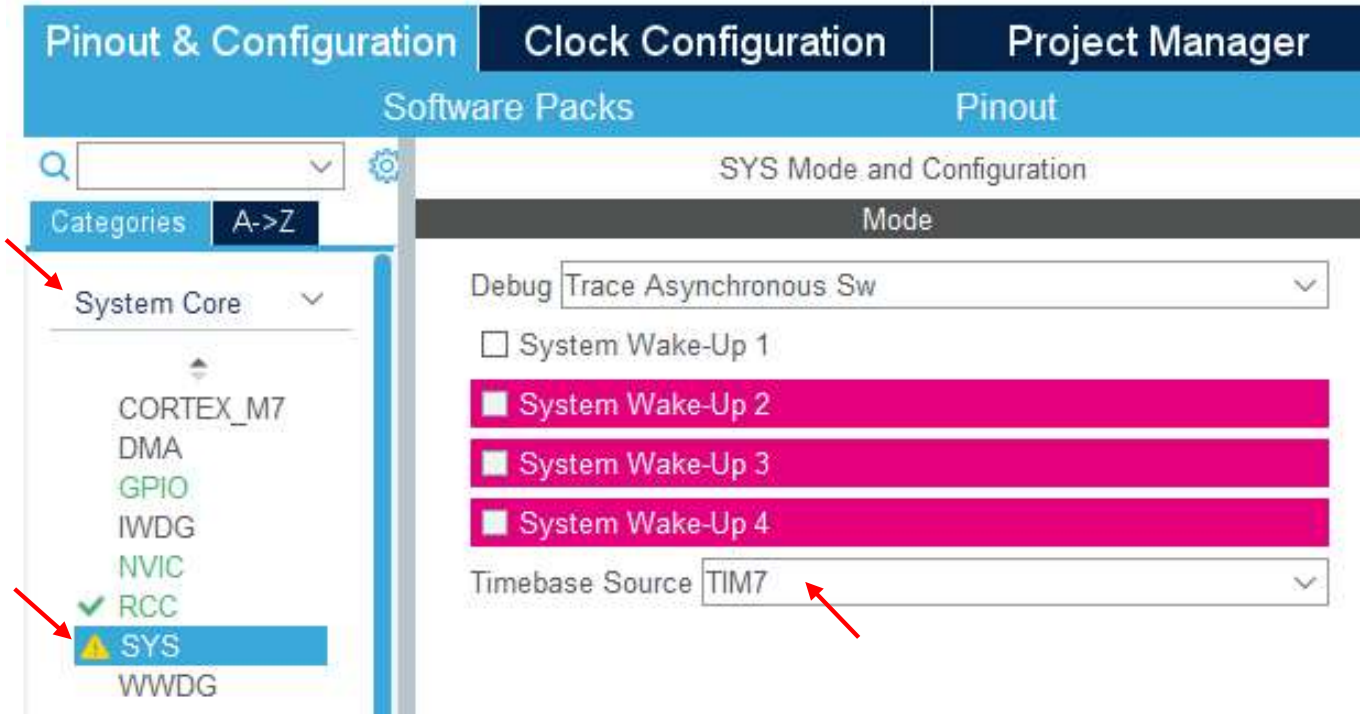
192.168.50.205
255.255.255.0
192.168.50.1

Need to enter:
- IP address
- Netmask address
- Gateway address

With **FREERTOS** selected

With **FREERTOS** selected, the **Timebase Source** is changed to **TIM7** manually.





Configure ADC1, select Temperature Sensor Channel

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

System Core
- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog
- ADC1
- ADC2
- ADC3
- DAC

Timers

Connectivity
- CAN1
- CAN2
- CAN3

ADC1 Mode and Configuration

Mode

IN15

Temperature Sensor Channel

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | DMA Settings

Configure the below parameters :

Search (Ctrl+F)

ADCs_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	PCLK2 divided by 4
Resolution	12 bits (15 ADC Clock cycles)
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Enabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	EOC flag at the end of single channel conversion
ADC_Regular_ConversionMode	
Number Of Conversion	1
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel Temperature Sensor
Sampling Time	480 Cycles

With **FREERTOS** selected, **Ethernet Basic Configuration** is modified.

Pinout & Configuration | Clock Configuration | Project Manager

Software Packs | Pinout

Categories | A->Z

Connectivity

CAN1
CAN2
CAN3
ETH
FMC
I2C1
I2C2
I2C3
I2C4
MDIOS
QUADSPI
SDMMC1
SDMMC2
SPI1
SPI2
SPI3
SPI4

ETH Mode and Configuration

Mode

Mode RMII

Configuration

Reset Configuration

Parameter Settings | User Constants | NVIC Settings | GPIO Settings

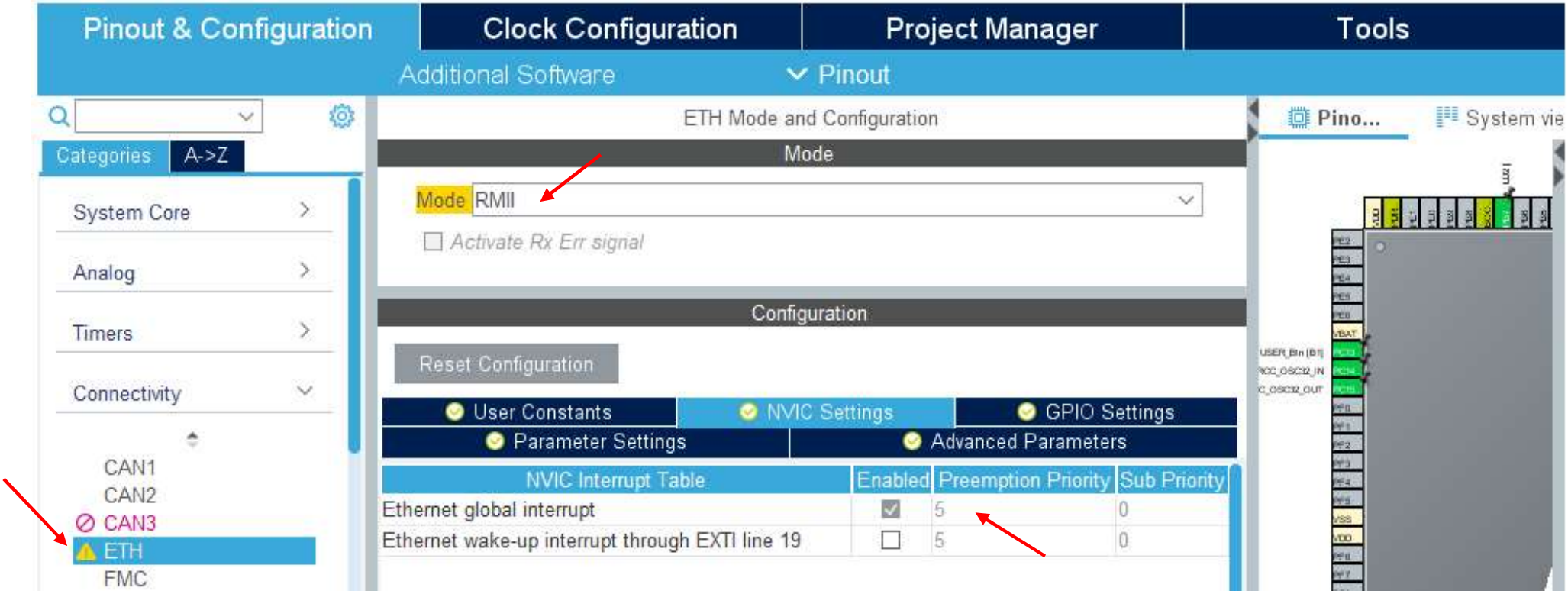
Configure the below parameters :

Search (Ctrl+F)

General : Ethernet Configuration

Warning	The ETH can work only when RAM is pointing at 0x24000000
Note	PHY Driver must be configured from the LwIP 'Platform Settings' ..
Ethernet MAC Address	02:80:E1:00:00:FF
Tx Descriptor Length	4
First Tx Descriptor Address	0x2007c0a0
Rx Descriptor Length	4
First Rx Descriptor Address	0x2007c000
Rx Buffers Length	1536
Rx Mode	Interrupt Mode

With **FREERTOS** selected, **Ethernet Global Interrupt** is enabled and assigned with Preemption Priority.



Hands-On LwIP HTTP Server Netconn RTOS LED

With **FREERTOS** and **Time Base** selections, the NVIC settings are modified automatically

Pinout & Configuration | Clock Configuration | Project Manager | Tools

Software Packs | Pinout

Categories | A->Z

System Core
- CORTEX_M7
- DMA
- GPIO
- IWDG
- NVIC
- RCC
- SYS
- WWDG

Analog

Timers

Connectivity
- CAN1
- CAN2
- CAN3
- ETH
- FMC
- I2C1
- I2C2

NVIC Mode and Configuration

Configuration

NVIC | Code generation

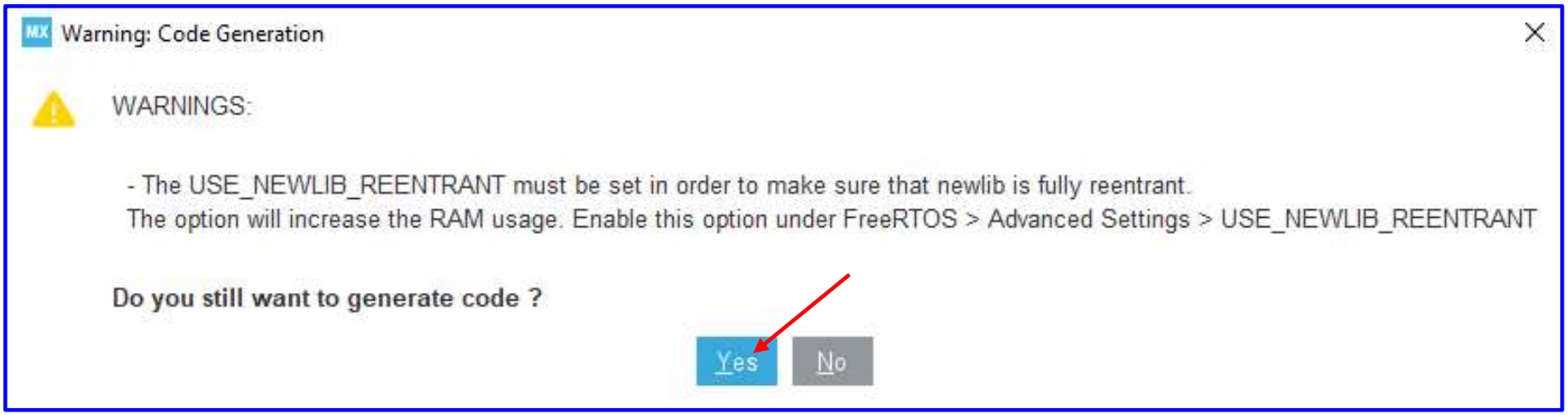
Priority Group: 4 bits for pre-emp... | Sort by Premption Priority and Sub Priority | Sort by interrupts names

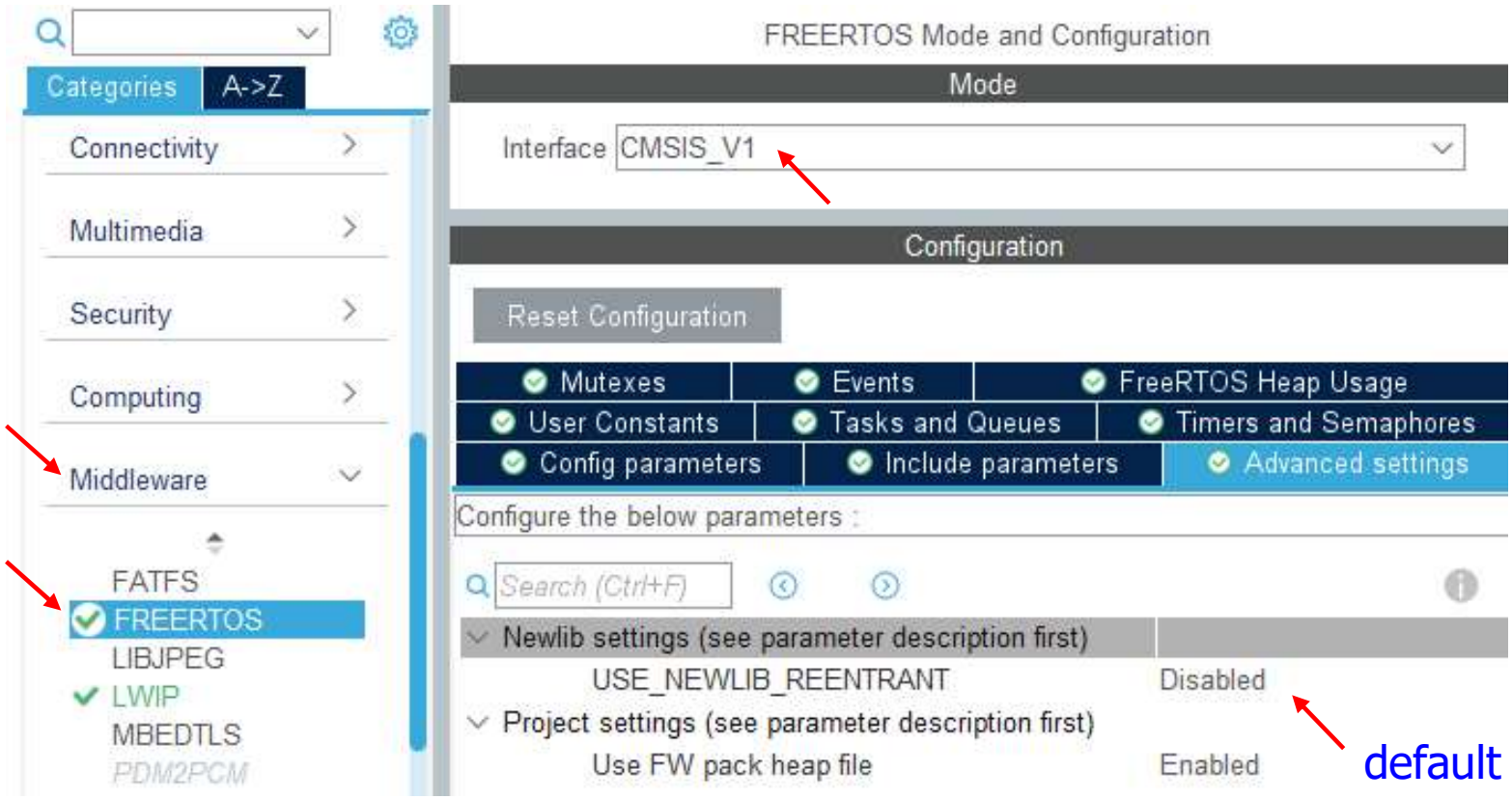
Search: Search (C... | Show: available interrupts | Force DMA channels Interrupts

NVIC Interrupt Table	Enabled	Preemption Priority	Sub Priority	Uses FreeRTOS functions
Non maskable interrupt	☑	0	0	☐
Hard fault interrupt	☑	0	0	☐
Memory management fault	☑	0	0	☐
Pre-fetch fault, memory access fault	☑	0	0	☐
Undefined instruction or illegal state	☑	0	0	☐
System service call via SWI instruction	☑	0	0	☐
Debug monitor	☑	0	0	☐
Pendable request for system service	☑	15	0	☑
System tick timer	☑	15	0	☑
PVD interrupt through EXTI line 16	☐	5	0	☑
Flash global interrupt	☐	5	0	☑
RCC global interrupt	☐	5	0	☑
USART3 global interrupt	☐	5	0	☑
EXTI line[15:10] interrupts	☑	5	0	☑
Time base: TIM7 global interrupt	☑	15	0	☐
Ethernet global interrupt	☑	5	0	☑
Ethernet wake-up interrupt through EXTI line 19	☐	5	0	☑
FPU global interrupt	☐	5	0	☑

Hands-On LwIP HTTP Server Netconn RTOS LED

Code Generation: Do not enable USE_NEWLIB_REENTRANT





default setting

Build warning: Hands-On LwIP HTTP Server Netconn RTOS LED

```
../LWIP/Target/ethernetif.h:36:13: warning: 'ethernetif_input' declared
'static' but never defined [-Wunused-function]
   36 | static void ethernetif_input(void const * argument);
```

Hands-On_5-1_UDP_TCP_Echo_Server_Netc
- Binaries
- Includes
- Core
  - Inc
  - Src
  - Startup
- Drivers
- LWIP
  - App
  - Target
    - ethernetif.c
    - ethernetif.h
    - lwipopts.h
- Middlewares
- Debug

```
23
24 #include "lwip/err.h"
25 #include "lwip/netif.h"
26 #include "cmsis_os.h"
27
28 /* Within 'USER CODE' section, code will be kept by default at each generation */
29 /* USER CODE BEGIN 0 */
30
31 /* USER CODE END 0 */
32
33 /* Exported functions ------------------------------------------------------- */
34 err_t ethernetif_init(struct netif *netif);
35
36 static void ethernetif_input(void const * argument);
37 void ethernet_link_thread(void const * argument);
38
39 void Error_Handler(void);
40 u32_t sys_jiffies(void);
41 u32_t sys_now(void);
```

insert "//"

```
35
36 // static void ethernetif_input(void const * argument);
37 void ethernet_link_thread(void const * argument);
```

- App
- Target
  - ethernetif.c
  - ethernetif.h
  - lwipopts.h
- Middlewares

```
34 /* Within 'USER CODE' section, code will be kept by default at each generation */
35 /* USER CODE BEGIN 0 */
36 static void ethernetif_input(void const * argument);
37 /* USER CODE END 0 */
38
39 /* Private define ------------------------------------------------------------*/
40 /* The time to block waiting for input. */
```

add

Hands-On LwIP HTTP Server Netconn RTOS LED

Generate the `fsdata_custom.c`

1 **Unzip 12_CS397_Hands-On_5-3_LwIP_HTTP_Server_Netconn_RTOS_LED.zip**

Name	Date modified	Type	Size
Fs		File folder	
Fs_Webpage		File folder	
httpserver-netconn.c		C File	7 KB
httpserver-netconn.h		C/C++ Header	1 KB
temp.c		C File	1 KB
temp.h		C/C++ Header	1 KB

4 Copy generated file "fsdata_custom.c" to folder "Fs"

Content of folder "Fs_Webpage"

index.html

2 Copy folder to

`C:\CS397>`

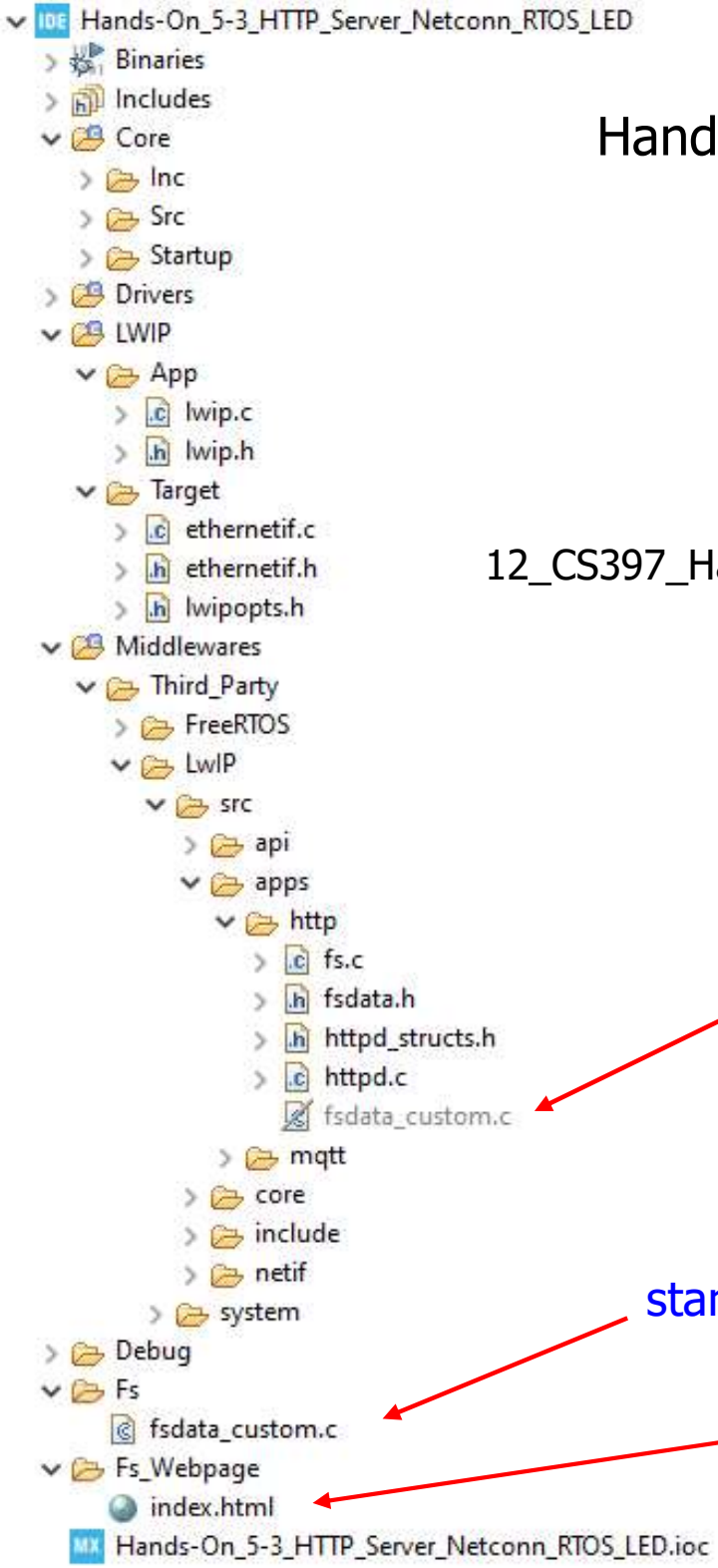
Fs_Webpage	File folder	
echotool.exe	Application	29 KB
fsdata_custom.c	C File	15 KB
htmlgen.exe	Application	106 KB

5 Copy above folders and files to STM32 project

3 Run

```
C:\CS397>htmlgen Fs_Webpage -f:fsdata_custom.c
```

Hands-On LwIP HTTP Server Netconn RTOS LED



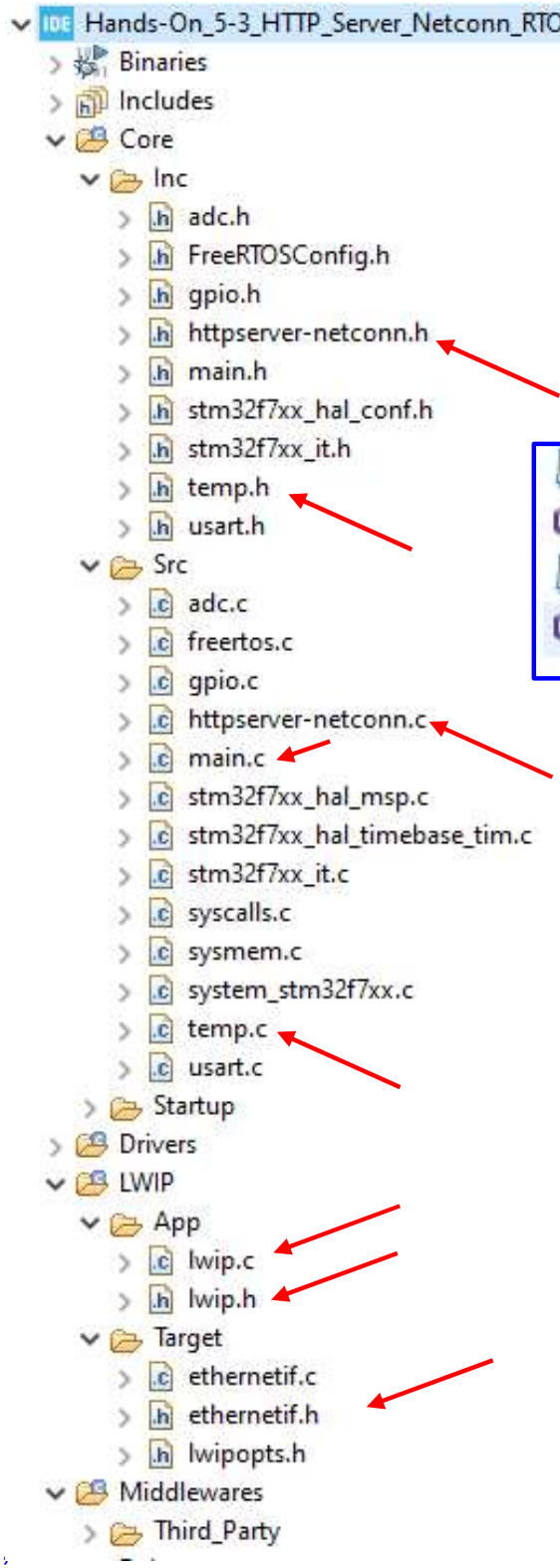
12_CS397_Hands-On_5-2_LwIP_HTTP_Server_Raw_25Jul2022.pptx

Refer to the previous example for setting up these files, pages 10 – 12, and pages 15 – 16.

standby for copying

This file (webpage) is converted to **`fsdata_custom.c`**

Hands-On LwIP HTTP Server Netconn RTOS LED



12_CS397_Hands-On_5-3_LwIP_HTTP_Server_Netconn_RTOS_LED.zip

httpserver-netconn.c	C File	7 KB
httpserver-netconn.h	C/C++ Header	1 KB
temp.c	C File	1 KB
temp.h	C/C++ Header	1 KB

Copy files to this project:

`httpserver-netconn.h`

`temp.h`

`httpserver-netconn.c`

`temp.c`

```c
/* Part of the main.c */
/* Includes */
#include "main.h"
#include "cmsis_os.h"
#include "adc.h"
#include "lwip.h"
#include "usart.h"
#include "gpio.h"

/* Private function prototypes */
void SystemClock_Config(void);
void MX_FREERTOS_Init(void);
int main(void)
{
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();
    /* Configure the system clock */
    SystemClock_Config();

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_USART3_UART_Init();
    MX_ADC1_Init();

    /* Call init function for freertos objects (in freertos.c) */
    MX_FREERTOS_Init();
    /* Start scheduler */
    osKernelStart();

    /* We should never get here as control is now taken by the scheduler */
    /* Infinite loop */
    while (1) { }
}
```

UM1713 User manual

Developing applications on STM32Cube with LwIP TCP/IP stack

Section 6 Using the LwIP applications

6.2.2 Web Server based on Netconn RTOS

Add to **main.c**

```
/* USER CODE BEGIN 4 */

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD1_Pin);
    }
}

int __io_putchar(int ch)
{
    uint8_t c[1];
    c[0] = ch & 0x00FF;
    HAL_UART_Transmit(&huart3, &*c, 1, 10);
    return ch;
}

int _write(int file, char *ptr, int len)
{
    int DataIdx;
    for(DataIdx= 0; DataIdx< len; DataIdx++)
    {
        __io_putchar(*ptr++);
    }
    return len;
}

 /* USER CODE END 4 */
```

Add code (optional)

The freertos.c (1/2)

Add code

```
/* freertos.c */
/* Includes */
#include "FreeRTOS.h"
#include "task.h"
#include "main.h"
#include "cmsis_os.h“

/* Private includes */
/* USER CODE BEGIN Includes */
#include "httpserver-netconn.h"
/* USER CODE END Includes */

osThreadId defaultTaskHandle;
void StartDefaultTask(void const * argument);

extern void MX_LWIP_Init(void);
void MX_FREERTOS_Init(void); /* (MISRA C 2004 rule 8.1) */

/* GetIdleTaskMemory prototype (linked to static allocation support) */
void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t
**ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize );

/* USER CODE BEGIN GET_IDLE_TASK_MEMORY */
static StaticTask_t xIdleTaskTCBBuffer;
static StackType_t xIdleStack[configMINIMAL_STACK_SIZE];

void vApplicationGetIdleTaskMemory( StaticTask_t **ppxIdleTaskTCBBuffer, StackType_t
**ppxIdleTaskStackBuffer, uint32_t *pulIdleTaskStackSize )
{
  *ppxIdleTaskTCBBuffer = &xIdleTaskTCBBuffer;
  *ppxIdleTaskStackBuffer = &xIdleStack[0];
  *pulIdleTaskStackSize = configMINIMAL_STACK_SIZE;
}
/* USER CODE END GET_IDLE_TASK_MEMORY */
```

The freertos.c (2/2)

```
/* @brief  FreeRTOS initialization */
void MX_FREERTOS_Init(void)
{
  /* Create the thread(s) */
  /* definition and creation of defaultTask */
  osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 2048);
  defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);

  /* USER CODE BEGIN RTOS_THREADS */
  /* add threads, ... */
  /* USER CODE END RTOS_THREADS */
}

/* USER CODE BEGIN Header_StartDefaultTask */
/* @brief  Function implementing the defaultTask thread */
/* USER CODE END Header_StartDefaultTask */
void StartDefaultTask(void const * argument)
{
  /* init code for LWIP */
  MX_LWIP_Init();
  /* USER CODE BEGIN StartDefaultTask */

  /* Initialize webserver demo */
  http_server_netconn_init();

  /* Infinite loop */
  for(;;)
  {
      osDelay(500);
      // HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
  }
  /* USER CODE END StartDefaultTask */
}
```

Add code

Modify code

For testing purposes, comment (//) this line after program is running properly.

Add **temp.c** and **temp.h**

```c
/* temp.c */

#include "stm32f7xx_hal.h"

extern ADC_HandleTypeDef hadc1;

float getMCUTemperature()
{
    float temp;

    HAL_ADC_Start(&hadc1);

    HAL_ADC_PollForConversion(&hadc1,HAL_MAX_DELAY);

    HAL_ADC_Stop(&hadc1);

    temp = ((HAL_ADC_GetValue(&hadc1))/4096.0)*3300.0;
    temp = (temp - 760.0)/2.5;
    temp += 25.0;

    return temp;
}
```

```c
/* temp.h */

#ifndef __TEMP_H__
#define __TEMP_H__

float getMCUTemperature();

#endif /* __TEMP_H__ */
```

Part of the **`httpserver-netconn.c`**

```c
/* httpserver-netconn.c */
/* Includes */
#include "lwip/opt.h"
#include "lwip/arch.h"
#include "lwip/api.h"
#include "lwip/apps/fs.h"   //added by Liaw HC
#include "string.h"
#include "httpserver-netconn.h"
#include "cmsis_os.h"
// #include "../webpages/index.h" //removed by Liaw HC

#include "temp.h"

#include <stdio.h>
#include <stdlib.h>

// #define tskIDLE_PRIORITY( ( UBaseType_t ) 0U )

/* Private typedef */
/* Private define  */
// #define WEBSERVER_THREAD_PRIO    ( tskIDLE_PRIORITY + 4 )
#define WEBSERVER_THREAD_PRIO       ( osPriorityAboveNormal )
/* Private macro */
/* Private variables */
// u32_t nPageHits = 0;

/* Private function prototypes */
static void http_server_serve(struct netconn *conn);
static void http_server_netconn_thread(void const *arg);
/* Private functions */
```

`httpserver-netconn.h`

```c
#ifndef __HTTPSERVER_NETCONN_H__
#define __HTTPSERVER_NETCONN_H__

void http_server_netconn_init(void);

#endif /* __HTTPSERVER_NETCONN_H__ */
```

Hands-On LwIP HTTP Server Netconn RTOS LED

Part of the **`httpserver-netconn.c`**

```
/* @brief serve tcp connection */
static void http_server_serve(struct netconn *conn)
{
  struct netbuf *inbuf;
  err_t recv_err;
  char* buf;
  u16_t buflen;
  struct fs_file file;
  float temp;
  int tempInt;

  /* Read the data from the port, blocking if nothing yet there.
   We assume the request (the part we care about) is in one netbuf */
  recv_err = netconn_recv(conn, &inbuf);

  if (recv_err == ERR_OK)
  {
    if (netconn_err(conn) == ERR_OK)
    {
      netbuf_data(inbuf, (void**)&buf, &buflen);

      /* Is this an HTTP GET command? (only check the first 5 chars, since
      there are other formats for GET, and we're keeping it very simple )*/
      if ((buflen >=5) && (strncmp(buf, "GET /", 5) == 0))
      {
         if ((strncmp((char const *)buf, "GET /index.html", 15) == 0) ||
             (strncmp((char const *)buf, "GET / ", 6) == 0))
         {
             /* Load index page */
             fs_open(&file, "/index.html");
             netconn_write(conn, (const unsigned char*)(file.data), (size_t)file.len, NETCONN_NOCOPY);
             fs_close(&file);
         }
```

http://www.cplusplus.com/reference/cstring/strncmp/

Part of the **`httpserver-netconn.c`**

```
        if (strncmp((char const *)buf,"GET /led1", 9) == 0)
        {
            HAL_GPIO_TogglePin(LD1_GPIO_Port, LD1_Pin);
        }
        if (strncmp((char const *)buf,"GET /led2", 9) == 0)
        {
            HAL_GPIO_TogglePin(LD2_GPIO_Port, LD2_Pin);
        }
        if (strncmp((char const *)buf,"GET /led3", 9) == 0)
        {
            HAL_GPIO_TogglePin(LD3_GPIO_Port, LD3_Pin);
        }
        if (strncmp((char const *)buf,"GET /btn1", 9) == 0)
        {
            if(HAL_GPIO_ReadPin(USER_Btn_GPIO_Port, USER_Btn_Pin) == GPIO_PIN_SET)
                netconn_write(conn, (const unsigned char*)"ON", 2, NETCONN_NOCOPY);
            else
                netconn_write(conn, (const unsigned char*)"OFF", 3, NETCONN_NOCOPY);
        }
        if (strncmp((char const *)buf,"GET /adc", 8) == 0)
        {
            temp = getMCUTemperature();
            tempInt = (int)temp;
            sprintf(buf, "%d.%d", tempInt, (int)((temp - (float)tempInt)*100.00) );
            netconn_write(conn, (const unsigned char*)buf, strlen(buf), NETCONN_NOCOPY);
        }
       } //  sprintf(buf, "%f deg C", getMCUTemperature());
      }
    }
    /* Close the connection (server closes in HTTP) */
    netconn_close(conn);
    /* Delete the buffer (netconn_recv gives us ownership,
    so we have to make sure to deallocate the buffer) */
    netbuf_delete(inbuf);
  }
```

Part of the **`httpserver-netconn.c`**

```c
/* @brief  http server thread */
static void http_server_netconn_thread(void const *arg)
{
    struct netconn *conn, *newconn;
    err_t err, accept_err;
    LWIP_UNUSED_ARG(arg);
    /* Create a new TCP connection handle */
    conn = netconn_new(NETCONN_TCP);
    if (conn!= NULL)
    {
        /* Bind to port 80 (HTTP) with default IP address */
        err = netconn_bind(conn, NULL, 80);
        if (err == ERR_OK)
        {
            /* Put the connection into LISTEN state */
            netconn_listen(conn);
            while(1)
            {
                /* accept any incoming connection */
                accept_err = netconn_accept(conn, &newconn);
                if(accept_err == ERR_OK)
                {
                    /* serve connection */
                    http_server_serve(newconn);
                    /* delete connection */
                    netconn_delete(newconn);
                }
            }
        }
    }
}
```

```c
/* @brief  Initialize the HTTP server (start its thread) */
void http_server_netconn_init()
{
    osThreadDef(TASK_HTTP, http_server_netconn_thread, WEBSERVER_THREAD_PRIO, 0,
                                                    configMINIMAL_STACK_SIZE*2 );
    osThreadCreate (osThread(TASK_HTTP), NULL);
}
```

Generated Code in **Lwip.c**

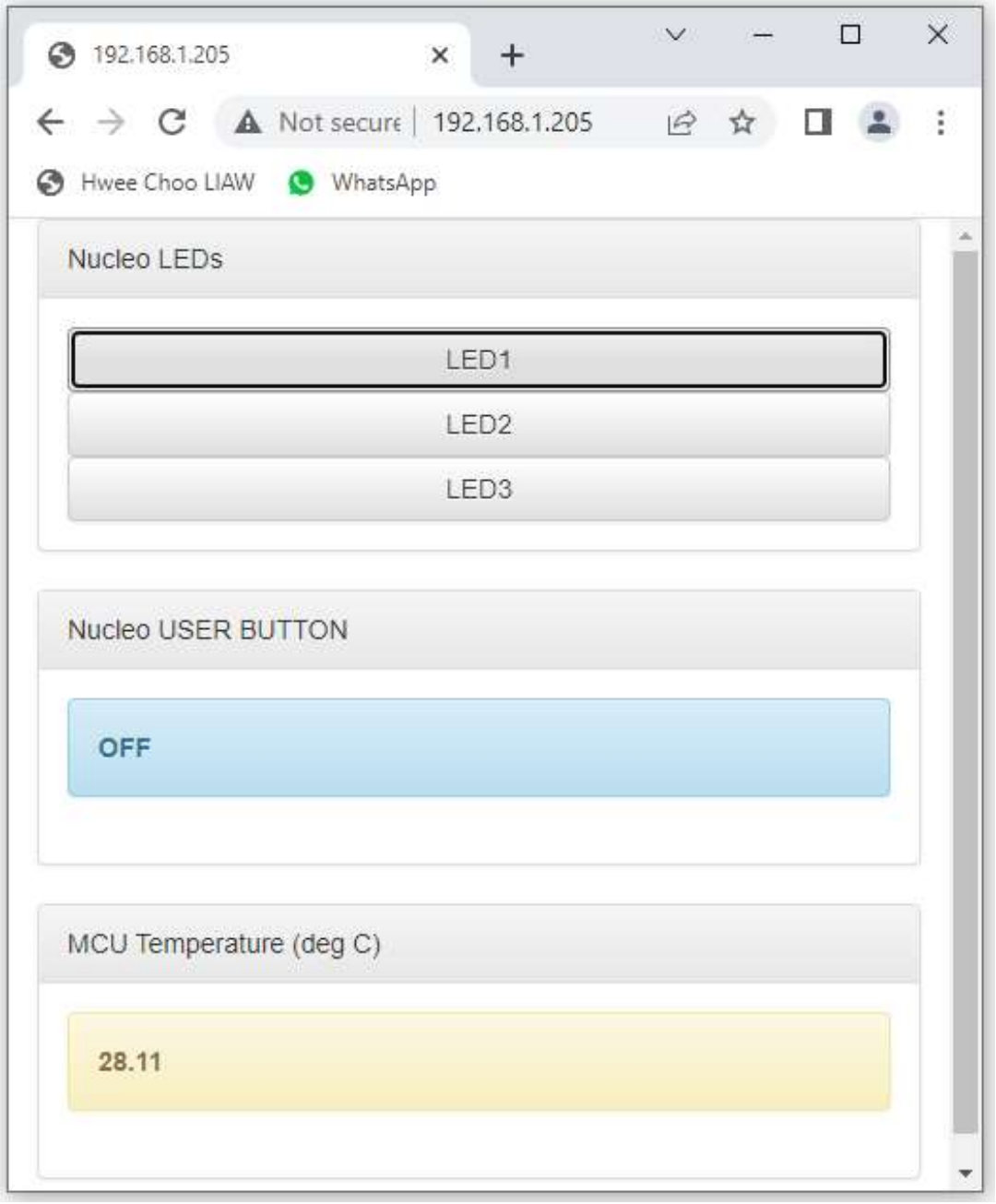
```
/* LwIP initialization function  */
void MX_LWIP_Init(void)
{
  /* IP addresses initialization */
  IP_ADDRESS[0] = 192;
  IP_ADDRESS[1] = 168;
  IP_ADDRESS[2] = 1;
  IP_ADDRESS[3] = 205;
  NETMASK_ADDRESS[0] = 255;
  NETMASK_ADDRESS[1] = 255;
  NETMASK_ADDRESS[2] = 255;
  NETMASK_ADDRESS[3] = 0;
  GATEWAY_ADDRESS[0] = 192;
  GATEWAY_ADDRESS[1] = 168;
  GATEWAY_ADDRESS[2] = 1;
  GATEWAY_ADDRESS[3] = 1;

/* USER CODE BEGIN IP_ADDRESSES */
/* USER CODE END IP_ADDRESSES */

  /* Initilialize the LwIP stack without RTOS */
  lwip_init();

  /* IP addresses initialization without DHCP (IPv4) */
  IP4_ADDR(&ipaddr, IP_ADDRESS[0], IP_ADDRESS[1], IP_ADDRESS[2], IP_ADDRESS[3]);
  IP4_ADDR(&netmask, NETMASK_ADDRESS[0], NETMASK_ADDRESS[1] , NETMASK_ADDRESS[2], NETMASK_ADDRESS[3]);
  IP4_ADDR(&gw, GATEWAY_ADDRESS[0], GATEWAY_ADDRESS[1], GATEWAY_ADDRESS[2], GATEWAY_ADDRESS[3]);

  /* add the network interface (IPv4/IPv6) without RTOS */
  netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &ethernetif_init, &ethernet_input);
```

For a different router (gateway):

```
IP_ADDRESS[0] = 192;
IP_ADDRESS[1] = 168;
IP_ADDRESS[2] = 50;
IP_ADDRESS[3] = 205;
NETMASK_ADDRESS[0] = 255;
NETMASK_ADDRESS[1] = 255;
NETMASK_ADDRESS[2] = 255;
NETMASK_ADDRESS[3] = 0;
GATEWAY_ADDRESS[0] = 192;
GATEWAY_ADDRESS[1] = 168;
GATEWAY_ADDRESS[2] = 50;
GATEWAY_ADDRESS[3] = 1;
```

Hands-On LwIP HTTP Server Netconn RTOS LED

After connected to the web server, http://192.168.1.205, you may control the LEDs of the MCU or Nucleo board, get response from the user-button, and read the MCU temperature.



Hands-On LwIP HTTP Server Netconn RTOS LED

Task List (1/4) – Settings

Pinout & Configuration | Clock Configuration | Project Mana

Software Packs | Pinout

Categories | A->Z

System Core
Analog
Timers
Connectivity
Multimedia
Security
Computing
Middleware

FATFS
FREERTOS
LIBJPEG
LWIP
MBEDTLS
PDM2PCM

FREERTOS Mode and Configuration

Mode

Interface CMSIS_V1

Configuration

Reset Configuration

Timers and Semaphores | Mutexes | Events | FreeRTOS Heap Usage

Config parameters | Include parameters | Advanced settings | User Constants | Tasks and Queues

Configure the below parameters :

Search (Ctrl+F)

TOTAL_HEAP_SIZE	63488 Bytes
Memory Management scheme	heap_4
Hook function related definitions	
USE_IDLE_HOOK	Disabled
USE_TICK_HOOK	Disabled
USE_MALLOC_FAILED_HOOK	Disabled
USE_DAEMON_TASK_STARTUP_HOOK	Disabled
CHECK_FOR_STACK_OVERFLOW	Disabled
Run time and task stats gathering related definitions	
GENERATE_RUN_TIME_STATS	Disabled
USE_TRACE_FACILITY	Enabled
USE_STATS_FORMATTING_FUNCTIONS	Enabled

Enable

Task List (2/4) – Add Code

```
/* freertos.c */

/* Includes */
#include "FreeRTOS.h"
#include "task.h"
#include "main.h"
#include "cmsis_os.h"

/* Private includes */
/* USER CODE BEGIN Includes */
#include "httpserver-netconn.h“
#include <stdio.h>
/* USER CODE END Includes */


/* Private variables */
/* USER CODE BEGIN Variables */
char taskList[512];
/* USER CODE END Variables */

osThreadId defaultTaskHandle;
```

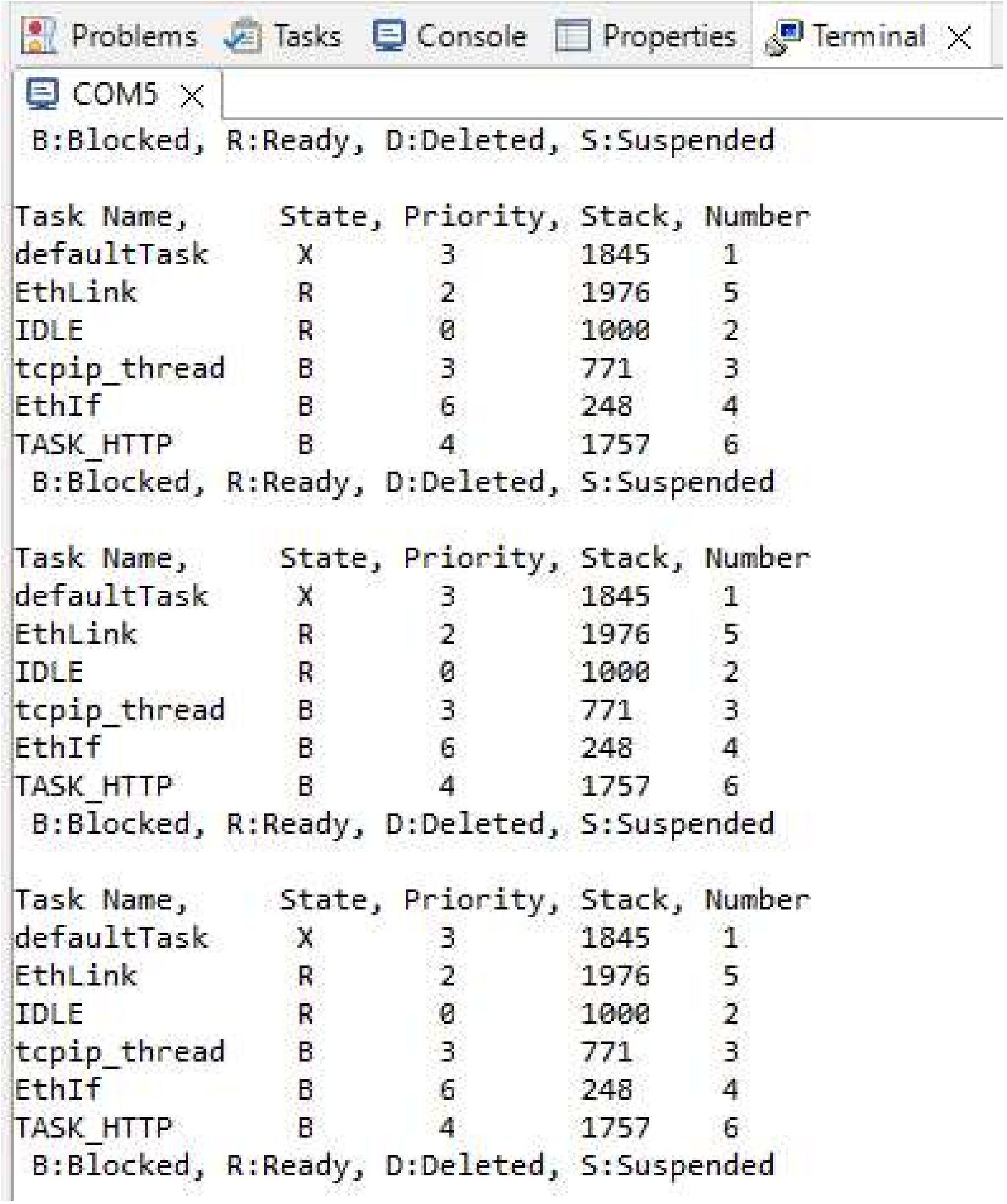
Add code

Task List (3/4) – Add Code

```
/* USER CODE END Header_StartDefaultTask */
void StartDefaultTask(void const * argument)
{
  /* init code for LWIP */
  MX_LWIP_Init();
  /* USER CODE BEGIN StartDefaultTask */

  /* Initialize webserver demo */
  http_server_netconn_init();

  /* Infinite loop */
  for(;;)
  {
      osDelay(500);
      osThreadList((uint8_t *)taskList);
      printf("Task Name,     State, Priority, Stack, Number \r\n");
      printf("%s ",taskList );
      printf("B:Blocked, R:Ready, D:Deleted, S:Suspended \r\n\r\n");
      // Note: the task's stack high water mark
      // (the smaller the high water mark number the closer
      // the task has come to overflowing its stack)
      // NOTE: This function will disable interrupts for its duration.
      // It is not intended for normal application runtime use but as a debug aid.
  }
  /* USER CODE END StartDefaultTask */
}
```

Add code

Hands-On LwIP HTTP Server Netconn RTOS LED Task List (4/4) – Results (TM Terminal Output)



```
 B:Blocked, R:Ready, D:Deleted, S:Suspended

Task Name,     State, Priority, Stack, Number
defaultTask     X        3        1845    1
EthLink         R        2        1976    5
IDLE            R        0        1000    2
tcpip_thread    B        3        771     3
EthIf           B        6        248     4
TASK_HTTP       B        4        1757    6
 B:Blocked, R:Ready, D:Deleted, S:Suspended

Task Name,     State, Priority, Stack, Number
defaultTask     X        3        1845    1
EthLink         R        2        1976    5
IDLE            R        0        1000    2
tcpip_thread    B        3        771     3
EthIf           B        6        248     4
TASK_HTTP       B        4        1757    6
 B:Blocked, R:Ready, D:Deleted, S:Suspended

Task Name,     State, Priority, Stack, Number
defaultTask     X        3        1845    1
EthLink         R        2        1976    5
IDLE            R        0        1000    2
tcpip_thread    B        3        771     3
EthIf           B        6        248     4
TASK_HTTP       B        4        1757    6
 B:Blocked, R:Ready, D:Deleted, S:Suspended
```

- End_ -