Embedded Systems

CS 397

TRIMESTER 3, AY 2021/22

Hands-On 5-2: Ethernet –

1. LwIP HTTP Server Raw
2. LwIP HTTP Server Raw CGI
3. LwIP HTTP Server Raw CGI SSI

Dr. LIAW Hwee Choo

Department of Electrical and Computer Engineering

DigiPen Institute of Technology Singapore

HweeChoo.Liaw@DigiPen.edu

CGI (Common Gateway Interface)
SSI (Server Side Includes)

Hands-On LwIP HTTP Server Raw

Objectives

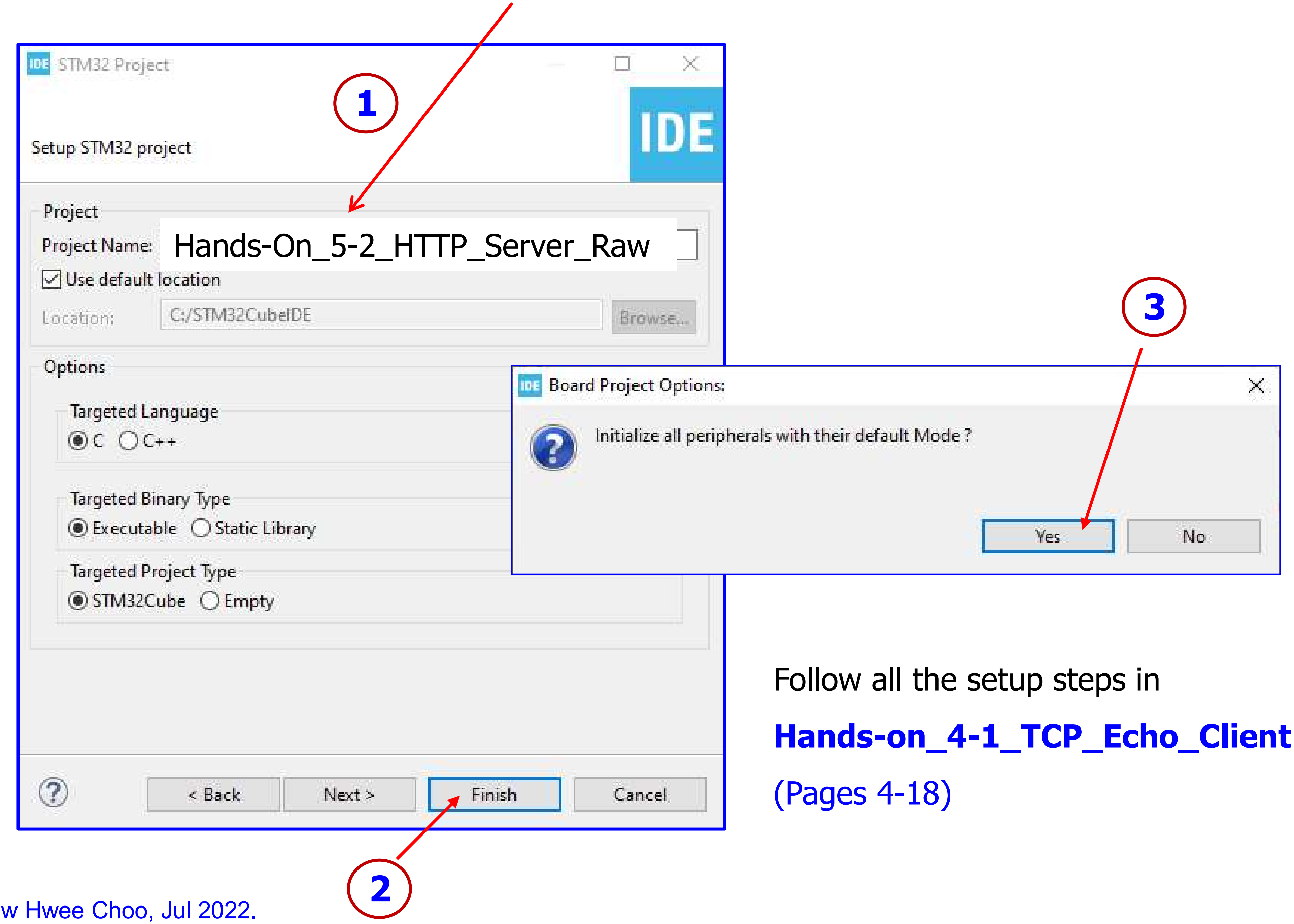
The aims of this hands-on session are to

- develop a STM32 (STM32CubeIDE) project
- implement a web (HTTP) server application based on LwIP raw API using STM32F767 microcontroller
- configure and program the Ethernet peripheral to make the microcontroller operating as a HTTP server and connecting web clients for loading of HTML pages
- develop program using the htmlgen.exe software to generate the web pages
- 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

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

Hands-On LwIP HTTP Server Raw

Enter Project Name: Hands-On_5-2_HTTP_Server_Raw



STM32 Project

Setup STM32 project

Project

Project Name: Hands-On_5-2_HTTP_Server_Raw

☑ Use default location

Location: C:/STM32CubeIDE

Options

Targeted Language: ● C ○ C++

Targeted Binary Type: ● Executable ○ Static Library

Targeted Project Type: ● STM32Cube ○ Empty

Board Project Options:

Initialize all peripherals with their default Mode ?

Follow all the setup steps in

Hands-on_4-1_TCP_Echo_Client

(Pages 4-18)

Hands-On LwIP HTTP Server Raw

Web Server Application Based on Raw API

In this hands-on session, applications are created to implement a web server, which is based solely on the LwIP raw API.

These applications will be used to connect web clients to the STM32 MCU to load web (HTML) pages stored in the MCU as well as access other web-sites on the Internet.

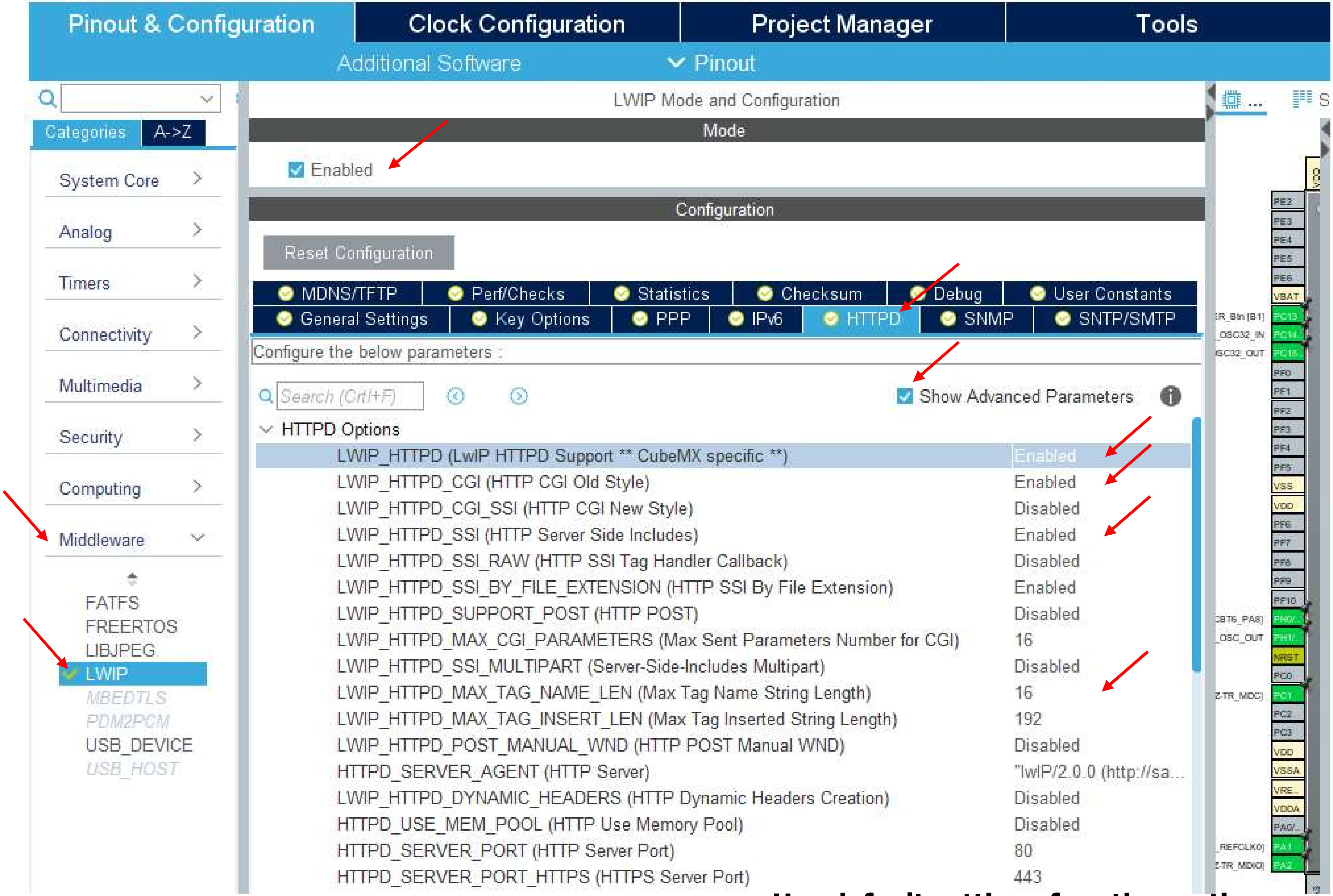
The web server applications implement the following features:

- URL (Uniform Resource Locator) parsing
- CGI (Common Gateway Interface)
- SSI (Server Side Includes)
- Dynamic Header generation
- HTTP Post request

A URL is an address that shows where a particular page can be found on the World Wide Web.

Hands-On LwIP HTTP Server Raw

Add in LwIP – HTTPD:



☑ Enabled

Configure the below parameters :

☑ Show Advanced Parameters

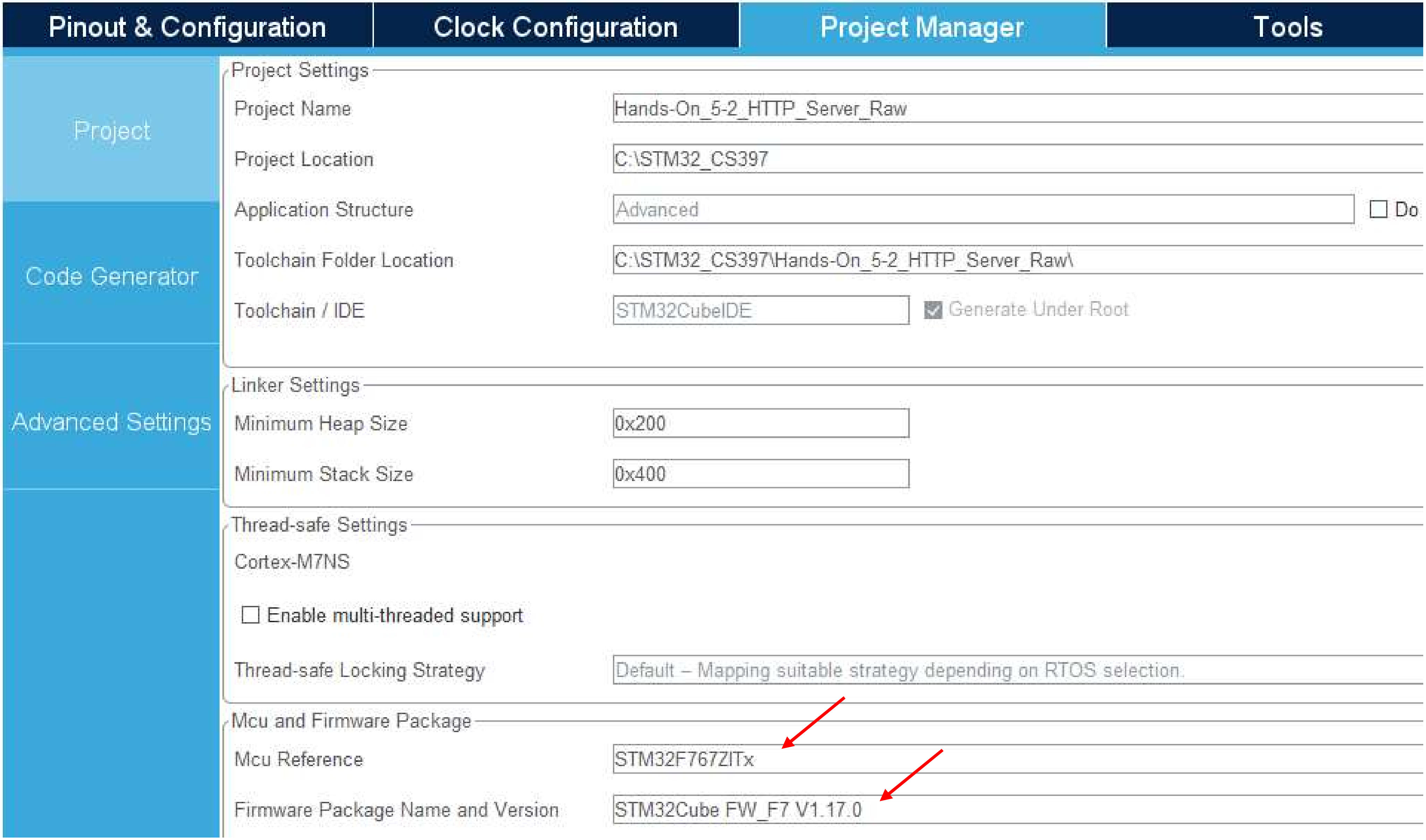
HTTPD Options

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 Raw

Information: Firmware Package Name and Version



Project Settings

Project Name: Hands-On_5-2_HTTP_Server_Raw

Project Location: C:\STM32_CS397

Application Structure: Advanced □ Do

Toolchain Folder Location: C:\STM32_CS397\Hands-On_5-2_HTTP_Server_Raw\

Toolchain / IDE: STM32CubeIDE ☑ Generate Under Root

Linker Settings

Minimum Heap Size: 0x200

Minimum Stack Size: 0x400

Thread-safe Settings

Cortex-M7NS

□ Enable multi-threaded support

Thread-safe Locking Strategy: Default – Mapping suitable strategy depending on RTOS selection.

Mcu and Firmware Package

Mcu Reference: STM32F767ZITx

Firmware Package Name and Version: STM32Cube FW_F7 V1.17.0

Hands-On LwIP HTTP Server Raw

Add Code to **main.c**

```c
/* Private includes */
/* USER CODE BEGIN Includes */

#include "lwip/apps/httpd.h"

/* USER CODE END Includes */

/* USER CODE BEGIN 2 */

/* Httpd Init */
httpd_init();

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    MX_LWIP_Process();

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Purpose and Test procedure:

UM1713 User manual

Developing applications on STM32Cube

with LwIP TCP/IP stack

Section 6.2 Features Applications

6.2.1 Web Server Based on Raw API

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);
```

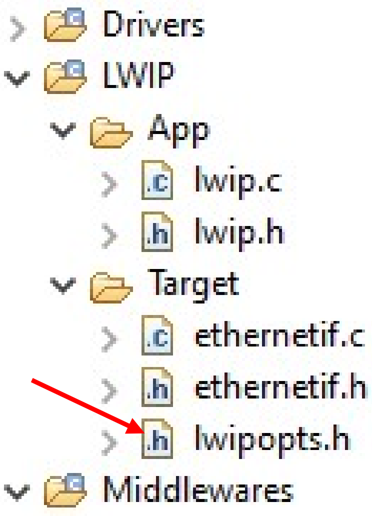
For a different router (gateway):

```
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;
```

The Settings in **lwipopts.h**

Line 70:

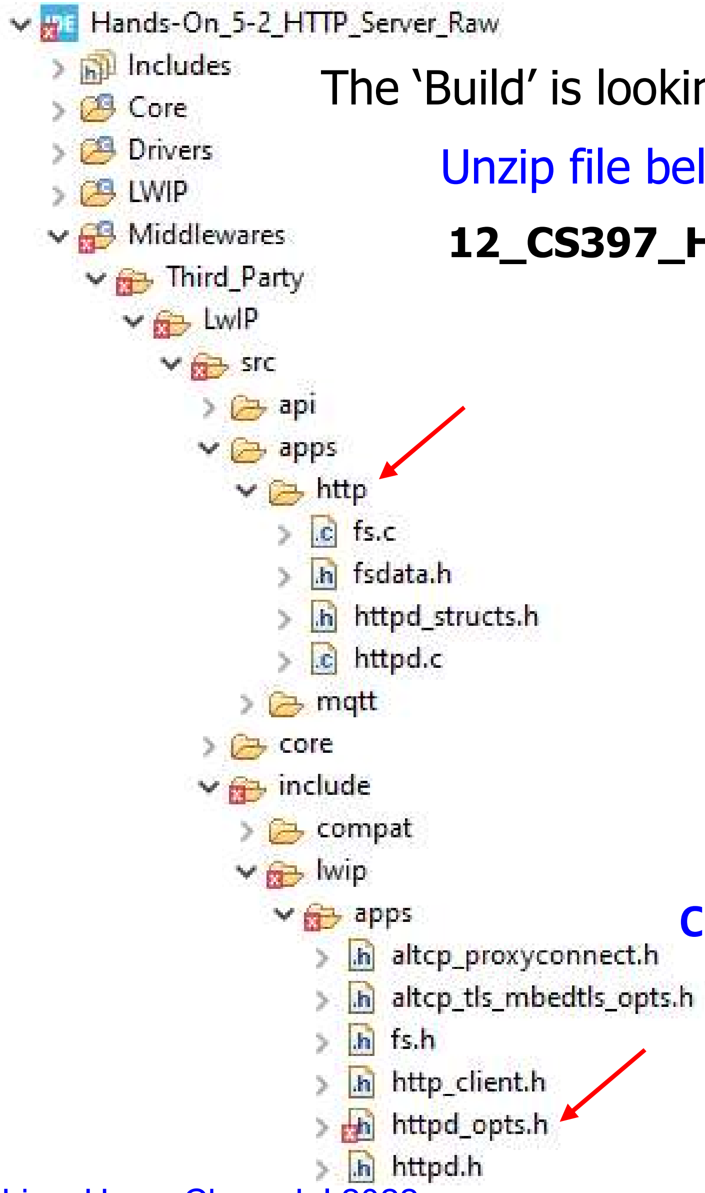
```
/*----- Value in opt.h for LWIP_NETCONN: 1 -----*/
#define LWIP_NETCONN 0
/*----- Value in opt.h for LWIP_SOCKET: 1 -----*/
#define LWIP_SOCKET 0
/*----- Value in opt.h for RECV_BUFSIZE_DEFAULT: INT_MAX -----*/
#define RECV_BUFSIZE_DEFAULT 2000000000
/*----- Default Value for LWIP_HTTPD: 0 ---*/
#define LWIP_HTTPD 1
/*----- Default Value for LWIP_HTTPD_CGI: 0 ---*/
#define LWIP_HTTPD_CGI 1
/*----- Default Value for LWIP_HTTPD_SSI: 0 ---*/
#define LWIP_HTTPD_SSI 1
/*----- Default Value for LWIP_HTTPD_MAX_TAG_NAME_LEN: 8 ---*/
#define LWIP_HTTPD_MAX_TAG_NAME_LEN 16
/*----- Value in opt.h for HTTPD_USE_CUSTOM_FSDATA: 0 -----*/
#define HTTPD_USE_CUSTOM_FSDATA 1
/*----- Value in opt.h for LWIP_STATS: 1 -----*/
#define LWIP_STATS 0
/*----- Value in opt.h for CHECKSUM_GEN_IP: 1 -----*/
#define CHECKSUM_GEN_IP 0
/*----- Value in opt.h for CHECKSUM_GEN_UDP: 1 -----*/
#define CHECKSUM_GEN_UDP 0
/*----- Value in opt.h for CHECKSUM_GEN_TCP: 1 -----*/
#define CHECKSUM_GEN_TCP 0
/*----- Value in opt.h for CHECKSUM_GEN_ICMP: 1 -----*/
#define CHECKSUM_GEN_ICMP 0
/*----- Value in opt.h for CHECKSUM_GEN_ICMP6: 1 -----*/
#define CHECKSUM_GEN_ICMP6 0
/*----- Value in opt.h for CHECKSUM_CHECK_IP: 1 -----*/
#define CHECKSUM_CHECK_IP 0
```



Missing File: fsdata_custom.c Hands-On LwIP HTTP Server Raw

With the code added to **main.c** and **Lwip.c,** the 'Build' will report an error.

```
../Middlewares/Third_Party/LwIP/src/include/lwip/apps/httpd_opts.h:386:27: fatal error:
fsdata_custom.c: No such file or directory
  386 | #define HTTPD_FSDATA_FILE "fsdata_custom.c"
```



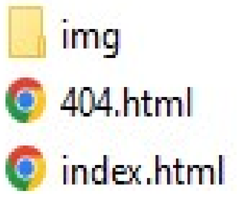
```
v Hands-On_5-2_HTTP_Server_Raw
  > Includes
  > Core
  > Drivers
  > LWIP
  v Middlewares
    v Third_Party
      v LwIP
        v src
          > api
          v apps
            v http
              > fs.c
              > fsdata.h
              > httpd_structs.h
              > httpd.c
            > mqtt
          > core
          v include
            > compat
            v lwip
              v apps
                > altcp_proxyconnect.h
                > altcp_tls_mbedtls_opts.h
                > fs.h
                > http_client.h
                > httpd_opts.h
                > httpd.h
```

The 'Build' is looking for **fsdata_custom.c** (web pages) defined in **httpd_opts.h**

Unzip file below to obtain the folder "Fs_HTTP_Server_Raw"

12_CS397_Hands-On_5-2_LwIP_HTTP_Server_Raw_CGI_SSI.zip

C:\CS397\Fs_HTTP_Server_Raw



img
404.html
index.html

To generate the web pages, we need to:

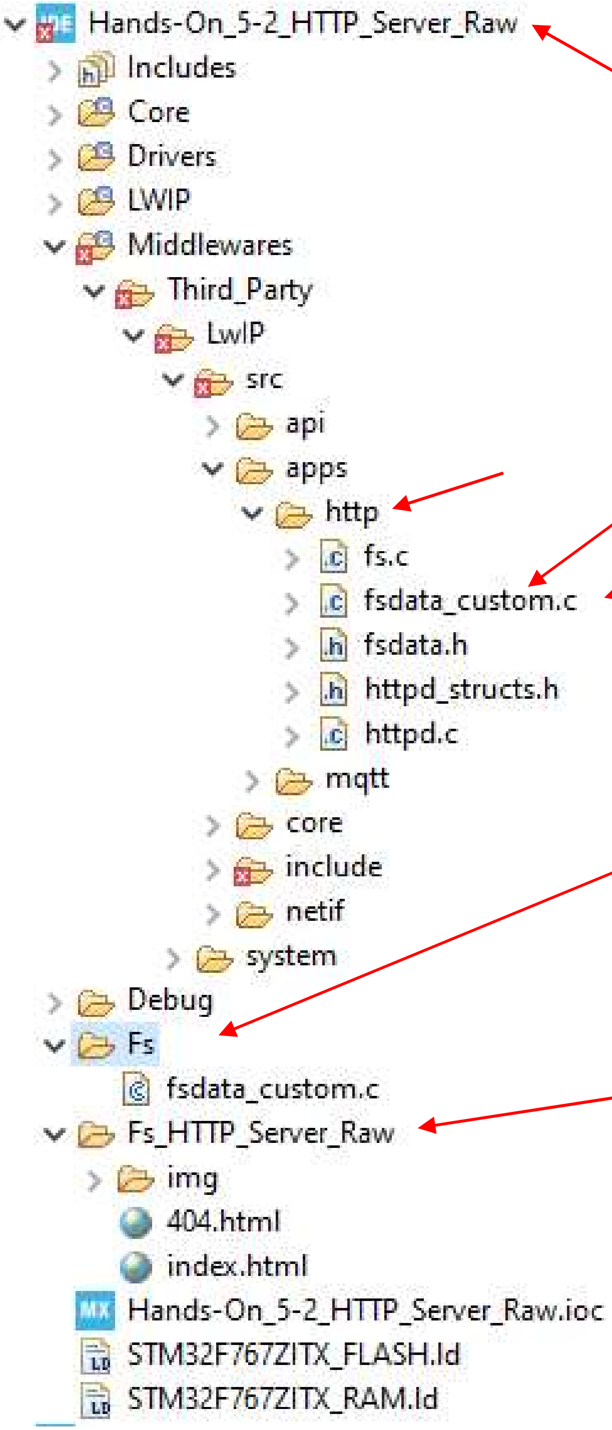
1. Copy the **htmlgen.exe** to **c:\CS397\**
2. Copy the folder, **Fs_HTTP_Server_Raw**, that contains the html and image files, to **c:\CS397\**
3. Open a command prompt window and go to **c:\CS397**
4. At the command prompt, enter: (folder name)

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

5. Copy the generated file **fsdata_custom.c** to

   ...**\Middlewares\Third_Party\LwIP\src\apps\http\**

Hands-On LwIP HTTP Server Raw



```
Hands-On_5-2_HTTP_Server_Raw
  Includes
  Core
  Drivers
  LWIP
  Middlewares
    Third_Party
      LwIP
        src
          api
          apps
            http
              fs.c
              fsdata_custom.c
              fsdata.h
              httpd_structs.h
              httpd.c
            mqtt
          core
          include
          netif
        system
  Debug
  Fs
    fsdata_custom.c
  Fs_HTTP_Server_Raw
    img
    404.html
    index.html
  Hands-On_5-2_HTTP_Server_Raw.ioc
  STM32F767ZITX_FLASH.ld
  STM32F767ZITX_RAM.ld
```

The generated file, **`fsdata_custom.c`**

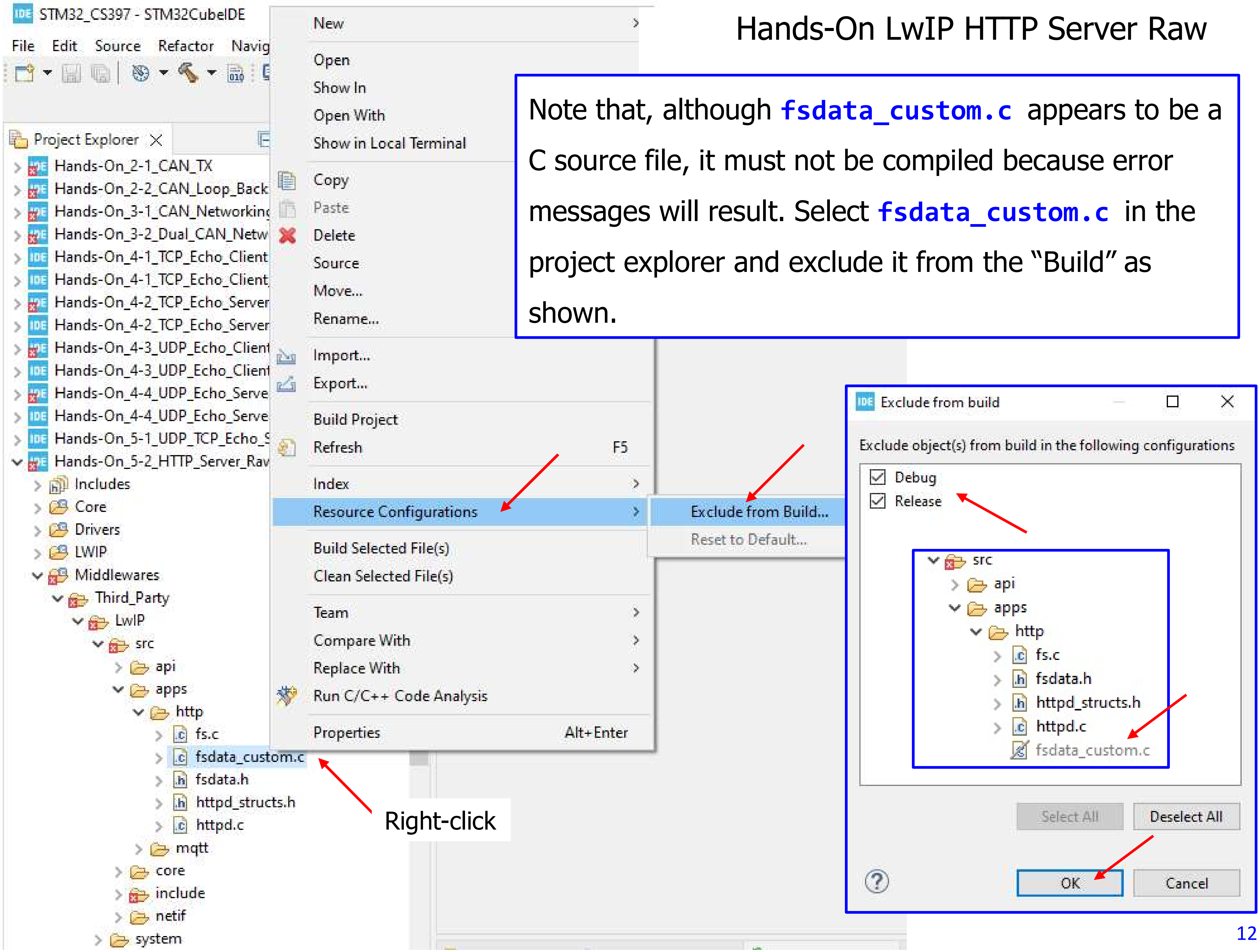
Note that, this file will be deleted when STM32CubeIDE is generated a new set of code via STM32CubeMX.

Create a folder "**`Fs`**" to keep the **`fsdata_custom.c`**, so that this file can be copied to the "**`http`**" folder when needed.

Copy the folder "**`Fs_HTTP_Server_Raw`**" and its contents here for reference to associate with the **`fsdata_custom.c`**

Hands-On LwIP HTTP Server Raw

Note that, although **fsdata_custom.c** appears to be a C source file, it must not be compiled because error messages will result. Select **fsdata_custom.c** in the project explorer and exclude it from the "Build" as shown.



Exclude object(s) from build in the following configurations

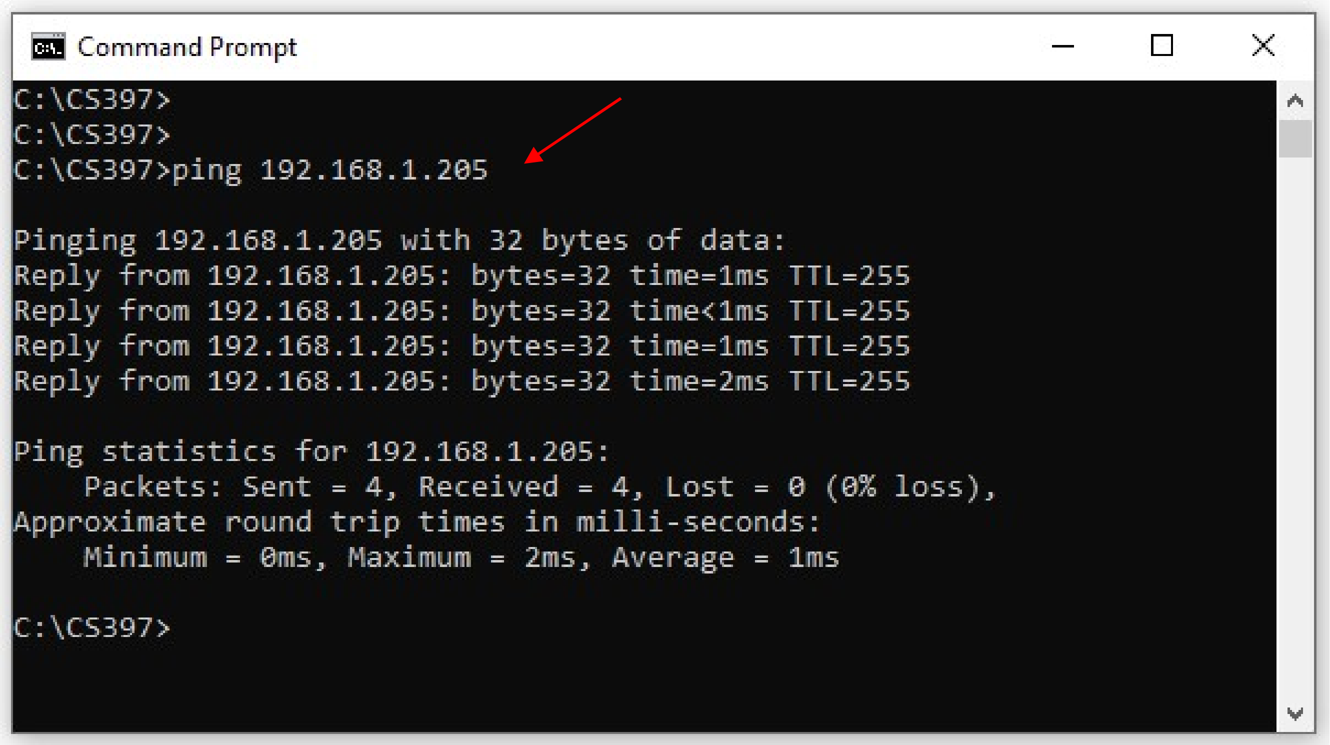
☑ Debug

☑ Release

Right-click

Hands-On LwIP HTTP Server Raw

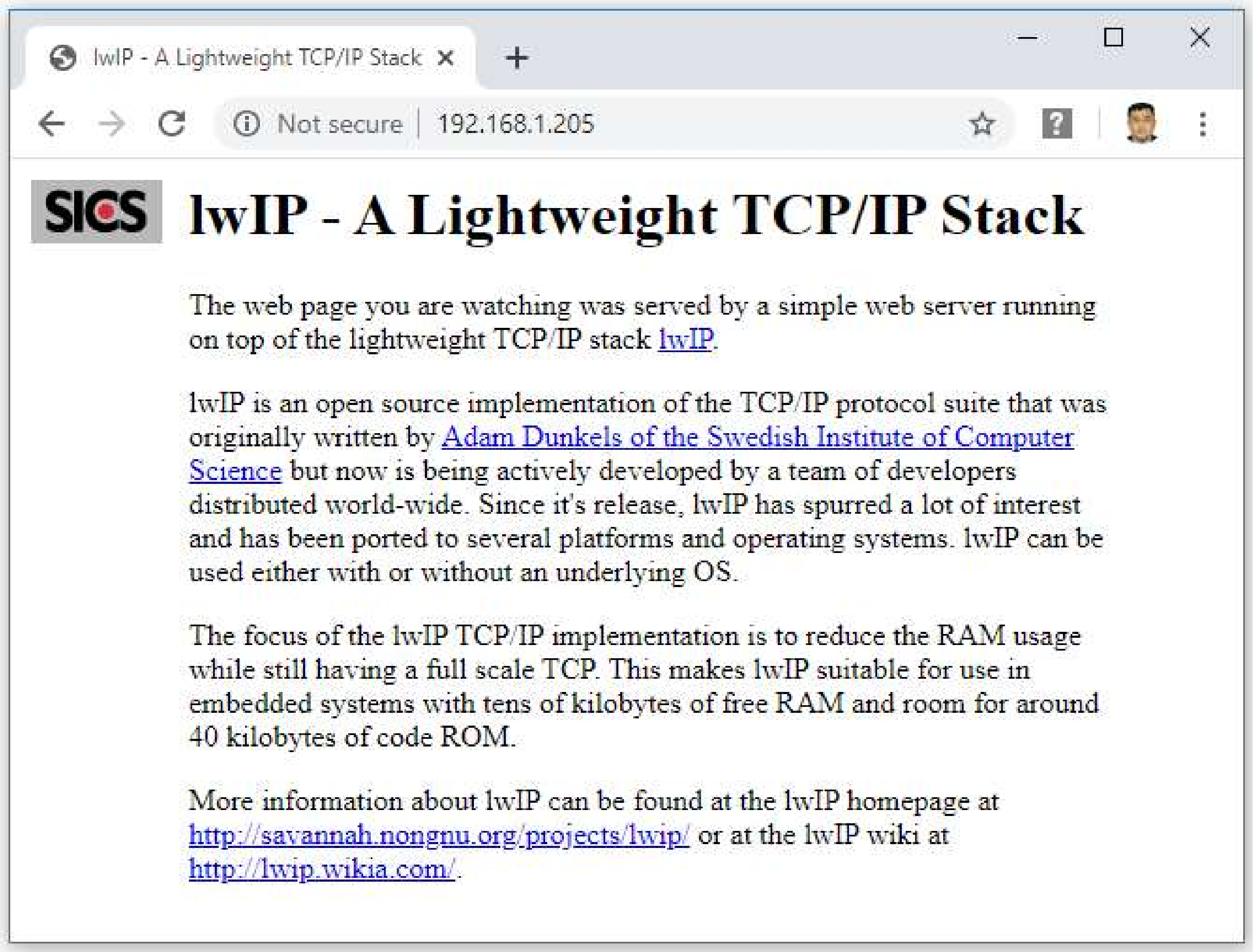
Build and program the project code into the STM32F767ZI Flash. Reset the MCU board power and connect the board to local network, you should be able to ping the board as shown in the figure below.



```
C:\CS397>
C:\CS397>
C:\CS397>ping 192.168.1.205

Pinging 192.168.1.205 with 32 bytes of data:
Reply from 192.168.1.205: bytes=32 time=1ms TTL=255
Reply from 192.168.1.205: bytes=32 time<1ms TTL=255
Reply from 192.168.1.205: bytes=32 time=1ms TTL=255
Reply from 192.168.1.205: bytes=32 time=2ms TTL=255

Ping statistics for 192.168.1.205:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 2ms, Average = 1ms

C:\CS397>
```

Hands-On LwIP HTTP Server Raw

When the project code is running on the Nucleo board, you should be able to access the web page by typing http://192.168.1.205 onto a web browser. At this time, you should be able to view the web page on the browser.



lwIP - A Lightweight TCP/IP Stack

The web page you are watching was served by a simple web server running on top of the lightweight TCP/IP stack lwIP.

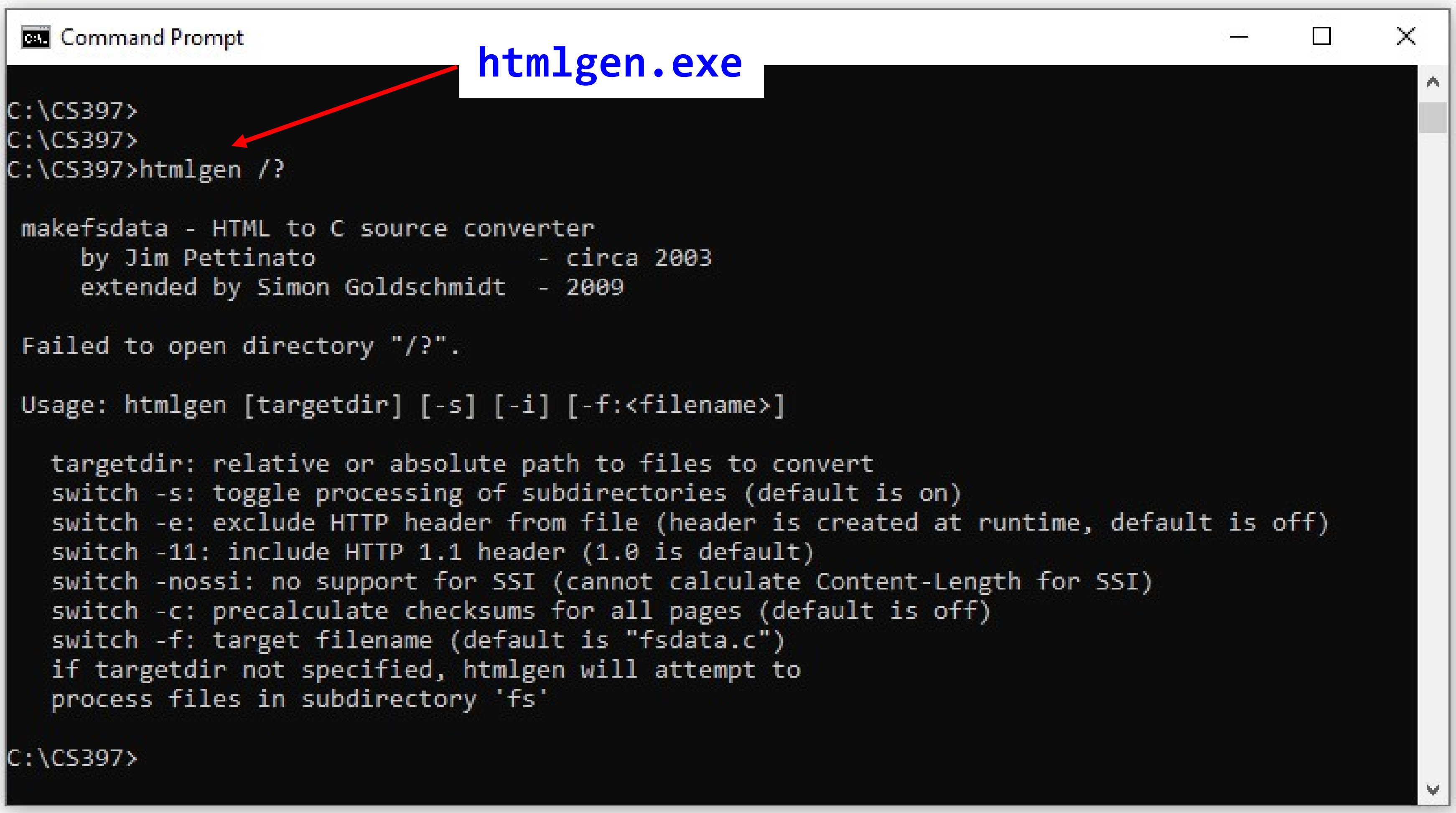
lwIP is an open source implementation of the TCP/IP protocol suite that was originally written by Adam Dunkels of the Swedish Institute of Computer Science but now is being actively developed by a team of developers distributed world-wide. Since it's release, lwIP has spurred a lot of interest and has been ported to several platforms and operating systems. lwIP can be used either with or without an underlying OS.

The focus of the lwIP TCP/IP implementation is to reduce the RAM usage while still having a full scale TCP. This makes lwIP suitable for use in embedded systems with tens of kilobytes of free RAM and room for around 40 kilobytes of code ROM.

More information about lwIP can be found at the lwIP homepage at http://savannah.nongnu.org/projects/lwip/ or at the lwIP wiki at http://lwip.wikia.com/.

Making a Web Page

In the development of a web server application, web pages are needed in the MCU. No file system is implemented, and the web pages are converted into a single file named as **fsdata_custom.c**, which is included during compiling. This is achieved by using the command line utility **htmlgen.exe** in the DOS command line.

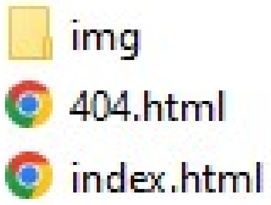


```
C:\CS397>
C:\CS397>
C:\CS397>htmlgen /?

 makefsdata - HTML to C source converter
     by Jim Pettinato               - circa 2003
     extended by Simon Goldschmidt  - 2009

 Failed to open directory "/?".

 Usage: htmlgen [targetdir] [-s] [-i] [-f:<filename>]

   targetdir: relative or absolute path to files to convert
   switch -s: toggle processing of subdirectories (default is on)
   switch -e: exclude HTTP header from file (header is created at runtime, default is off)
   switch -11: include HTTP 1.1 header (1.0 is default)
   switch -nossi: no support for SSI (cannot calculate Content-Length for SSI)
   switch -c: precalculate checksums for all pages (default is off)
   switch -f: target filename (default is "fsdata.c")
   if targetdir not specified, htmlgen will attempt to
   process files in subdirectory 'fs'

C:\CS397>
```

Making a Web Page

For example, if the **html** files are contained in the folder "**Fs_HTTP_Server_Raw**",

```
C:\CS397\Fs_HTTP_Server_Raw
```



img
404.html
index.html

(folder name)

the command is

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

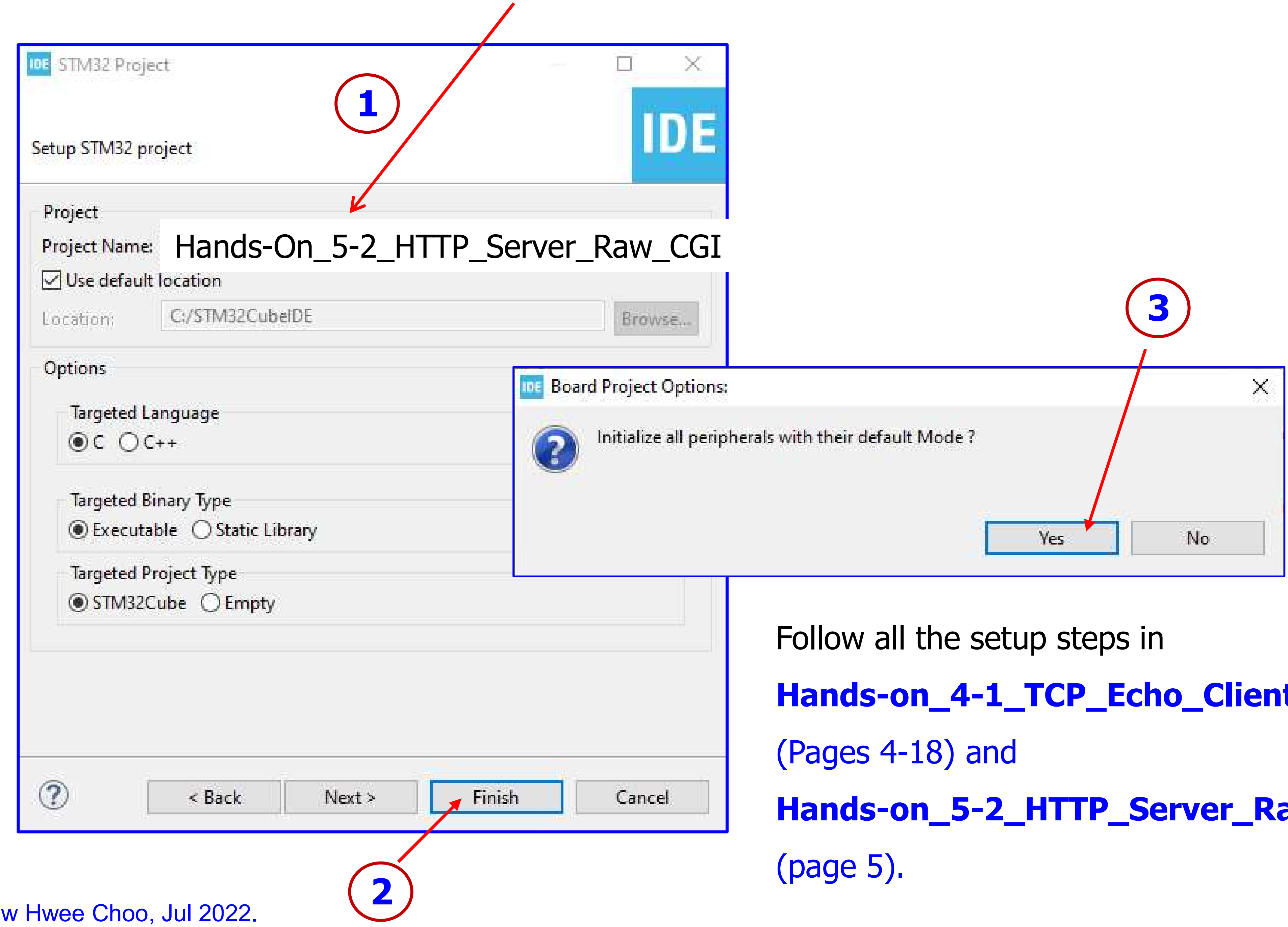
If **htmlgen.exe** is at **c:\CS397.** The folder should contain an **index.html** file and a **404.html** file at a minimum. The **404.html** file is helpful if you make a mistake as it will tell the server to load a non-existent webpage. Otherwise, the web page would just sit there like a dump, and you won’t know what is wrong.

Note that you need to copy the generated file **fsdata_custom.c** to the project folder:

...**\Middlewares\Third_Party\LwIP\src\apps\http\**

Hands-On LwIP HTTP Server Raw CGI

Enter Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI



1

2

3

STM32 Project

Setup STM32 project

Project

Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI

☑ Use default location

Location: C:/STM32CubeIDE

Options

Targeted Language

● C ○ C++

Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

Board Project Options:

Initialize all peripherals with their default Mode ?

Follow all the setup steps in **Hands-on_4-1_TCP_Echo_Client** (Pages 4-18) and **Hands-on_5-2_HTTP_Server_Raw** (page 5).

Hands-On LwIP HTTP Server Raw CGI

Add Code to **main.c**

CGI = Common Gateway Interface

```
/* Private includes */
/* USER CODE BEGIN Includes */
#include "lwip/apps/httpd.h"
#include <string.h>
/* USER CODE END Includes */

/* Private variables */
/* USER CODE BEGIN PV */
// prototype CGI handler for the LED control
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[]);

// this structure contains the name of the LED CGI and corresponding handler for the LEDs
const tCGI LedCGI={"/leds.cgi", LedCGIhandler};

// table of the CGI names and handlers
tCGI theCGItable[1];
/* USER CODE END PV */

/* Private user code */
/* USER CODE BEGIN 0 */
// Initialize the CGI handlers
void myCGIinit(void)
{
    // add LED control CGI to the table
    theCGItable[0] = LedCGI;

    // give the table to the HTTP server
    http_set_cgi_handlers(theCGItable, 1);
} // myCGIinit
```

Purpose and Test procedure:

UM1713 User manual

Developing applications on STM32Cube

with LwIP TCP/IP stack

Section 6.2 Features Applications

6.2.1 Web Server Based on Raw API

Hands-On LwIP HTTP Server Raw CGI

Add Code to **main.c**

```c
/**** CGI handler for controlling the LEDs ****/
// the function pointer for a CGI script handler is defined in httpd.h as tCGIHandler
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[])
{
    uint32_t i = 0;
    // index of the CGI within the theCGItable array passed to http_set_cgi_handlers
    // Given how this example is structured, this may be a redundant check.
    // Here there is only one handler iIndex == 0
    if (iIndex == 0)
    {
        // turn off the LEDs
        HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
```

Add Code to **main.c**

```
        // Check the cgi parameters, e.g., GET /leds.cgi?led=1&led=2&led=3
        for (i=0; i<iNumParams; i++)
        {
            // if pcParmeter contains "led", then one of the LED check boxes has been set on
            if (strcmp(pcParam[i], "led") == 0)
            {
                // see if checkbox for LED 1 has been set
                if(strcmp(pcValue[i], "1") == 0)
                {
                    // switch led 1 ON if 1
                    HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_SET);
                }
                // see if checkbox for LED 2 has been set
                else if(strcmp(pcValue[i], "2") == 0)
                {
                    // switch led 2 ON if 2
                    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
                }
                // see if checkbox for LED 3 has been set
                else if(strcmp(pcValue[i], "3") == 0)
                {
                    // switch led 3 ON if 3
                    HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
                }
            } //if
        } //for
    } //if
    // uniform resource identifier to send after CGI call, i.e., path and filename of the response
    return "/index.html";
} // LedCGIhandler
/* USER CODE END 0 */
```

Add Code to **main.c**

```
/* @brief  The application entry point */
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_LWIP_Init();
  /* USER CODE BEGIN 2 */
  // start the web server
  httpd_init();

  // initialise the CGI handlers
  myCGIinit();
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      MX_LWIP_Process();
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

The `index.html`

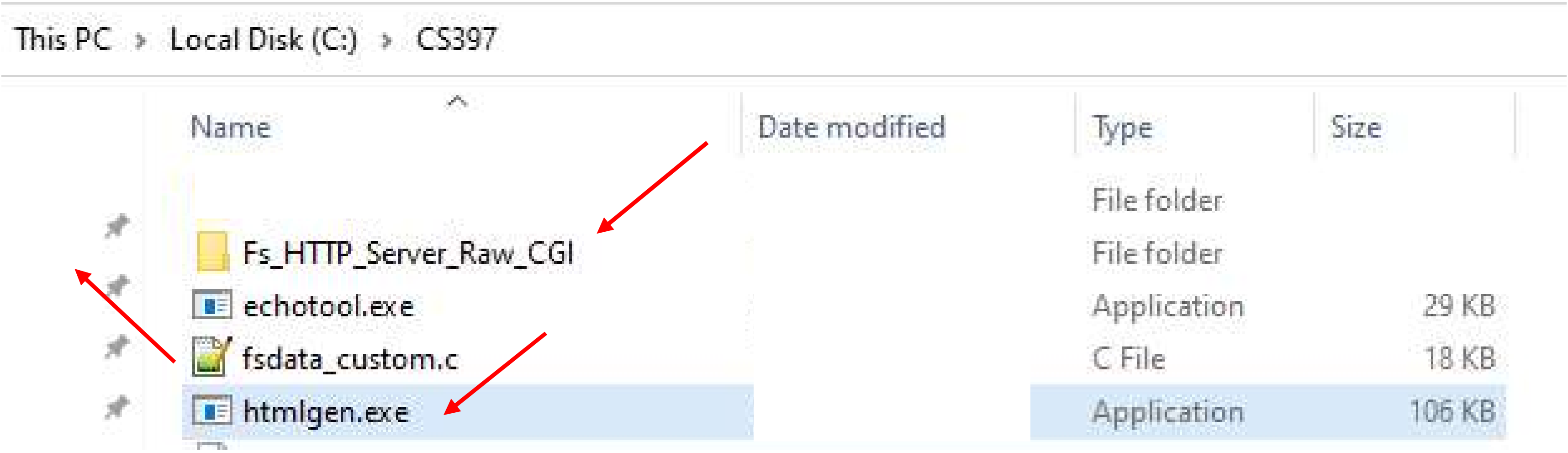
```html
<!DOCTYPE html>
<html><head>
<title>LED Test</title></head>
<p>This program allows you to control the LEDs: LED1, LED2 and LED3.</p>
<p>You must select and click on "Send" button to change the LEDs.</p>
<form method="get" action="/leds.cgi">
<input value="1" name="led" type="checkbox">LED1<br>
<input value="2" name="led" type="checkbox">LED2<br>
<input value="3" name="led" type="checkbox">LED3<br>
<br>
<input value="Send" type="submit"> </form>
<p>Modified by Liaw Hwee Choo, June 2022</p>
</html>
```

Hands-On LwIP HTTP Server Raw CGI

Generate the **fsdata_custom.c**

1 **Unzip 12_CS397_Hands-On_5-2_LwIP_HTTP_Server_Raw_CGI_SSI.zip**

2 Copy folder “Fs_HTTP_Server_Raw_CGI” to c:\CS397



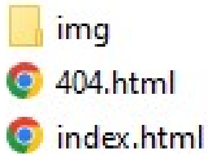
This PC › Local Disk (C:) › CS397

Name	Date modified	Type	Size
		File folder	
Fs_HTTP_Server_Raw_CGI		File folder	
echotool.exe		Application	29 KB
fsdata_custom.c		C File	18 KB
htmlgen.exe		Application	106 KB

3 Run

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

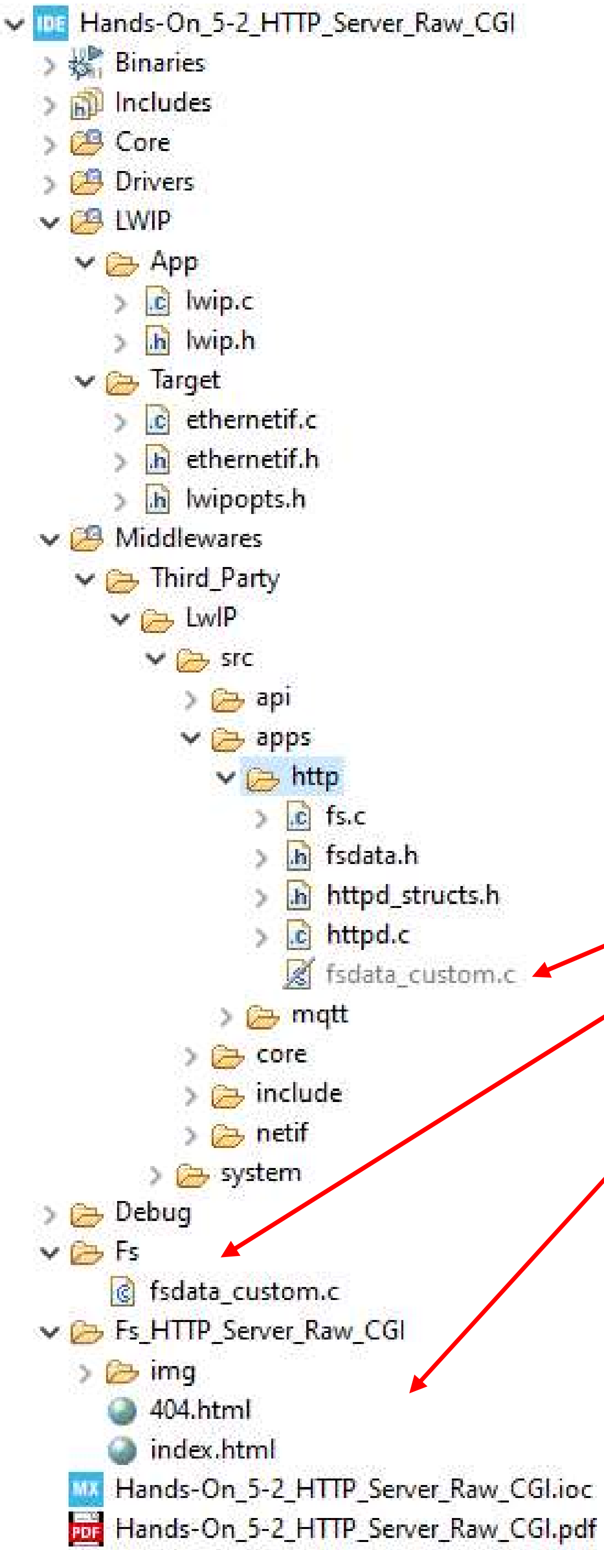
C:\CS397\Fs_HTTP_Server_Raw_CGI



img
404.html
index.html

4 Copy folder and generated file “fsdata_custom.c” to STM32 project

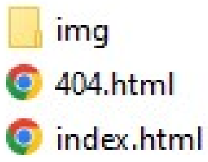
Hands-On LwIP HTTP Server Raw CGI



```
Hands-On_5-2_HTTP_Server_Raw_CGI
  Binaries
  Includes
  Core
  Drivers
  LWIP
    App
      lwip.c
      lwip.h
    Target
      ethernetif.c
      ethernetif.h
      lwipopts.h
  Middlewares
    Third_Party
      LwIP
        src
          api
          apps
            http
              fs.c
              fsdata.h
              httpd_structs.h
              httpd.c
              fsdata_custom.c
            mqtt
          core
          include
          netif
        system
  Debug
  Fs
    fsdata_custom.c
  Fs_HTTP_Server_Raw_CGI
    img
    404.html
    index.html
  Hands-On_5-2_HTTP_Server_Raw_CGI.ioc
  Hands-On_5-2_HTTP_Server_Raw_CGI.pdf
```

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

C:\CS397\Fs_HTTP_Server_Raw_CGI

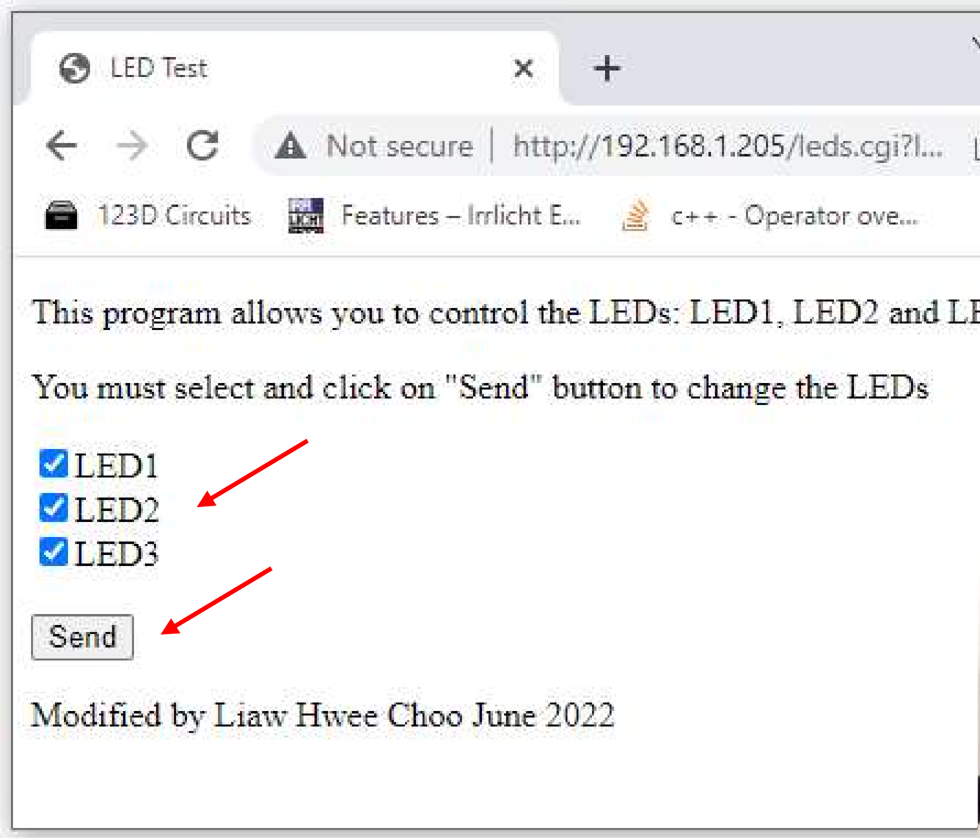


```
img
404.html
index.html
```

Hands-On LwIP HTTP Server Raw CGI

When the project code is running on the Nucleo board, you should be able to access the web page by typing http://192.168.1.205 onto a web browser. At this time, you should be able to view the web page on the browser and perform the implemented function.

Note: Remember to reset the MCU board



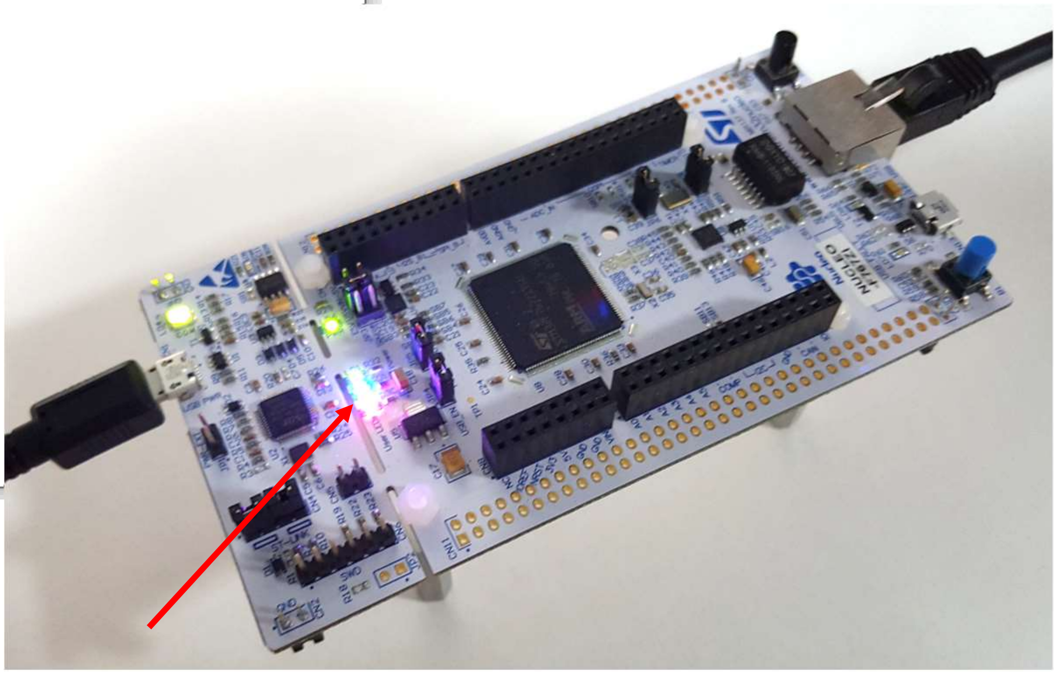
This program allows you to control the LEDs: LED1, LED2 and LED3

You must select and click on "Send" button to change the LEDs

☑ LED1
☑ LED2
☑ LED3

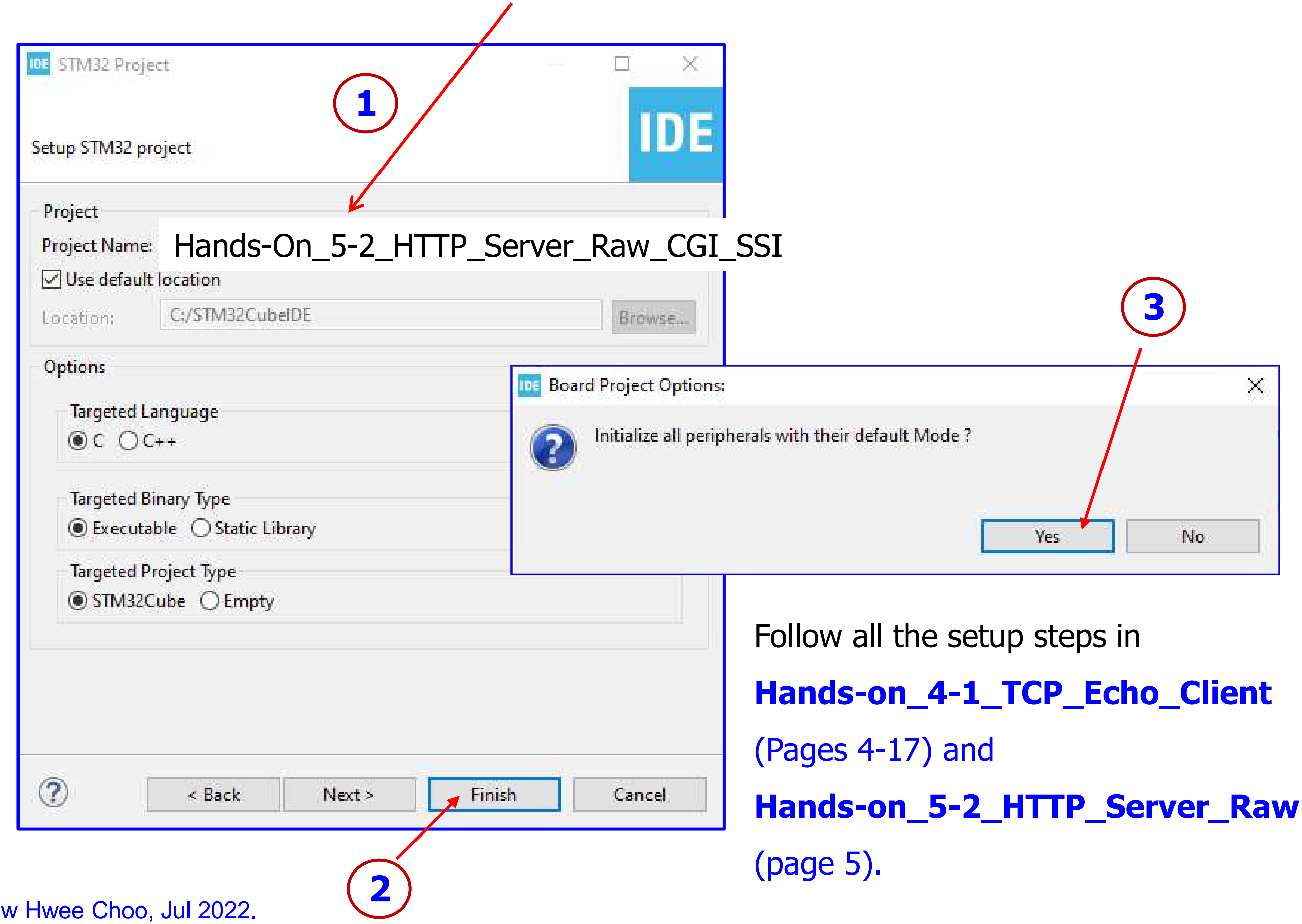
Send

Modified by Liaw Hwee Choo June 2022



Hands-On LwIP HTTP Server Raw CGI SSI

Enter Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI_SSI



STM32 Project

Setup STM32 project

Project

Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI_SSI

☑ Use default location

Location: C:/STM32CubeIDE Browse...

Options

Targeted Language

● C ○ C++

Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

< Back | Next > | Finish | Cancel

Board Project Options:

Initialize all peripherals with their default Mode ?

Yes | No

Follow all the setup steps in **Hands-on_4-1_TCP_Echo_Client** (Pages 4-17) and **Hands-on_5-2_HTTP_Server_Raw** (page 5).

Hands-On LwIP HTTP Server Raw CGI SSI

Add Code to **main.c**

```
/* Private includes */
/* USER CODE BEGIN Includes */
#include "lwip/apps/httpd.h"
#include <string.h>
#include <stdlib.h>
/* USER CODE END Includes */

/* Private typedef */
/* USER CODE BEGIN PTD */
// array of tags for the SSI handler
// these are the tags <!--#tag1--> contained in the shtml file
#define numSSItags   2
/* USER CODE END PTD */

/* Private variables */
/* USER CODE BEGIN PV */
char const *theSSItags[numSSItags] = {"tag1","tag2"};

// prototype CGI handler for the LED control
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[]);

// this structure contains the name of the LED CGI and corresponding handler for the LEDs
const tCGI LedCGI={"/leds.cgi", LedCGIhandler};

// table of the CGI names and handlers
tCGI theCGItable[1];
/* USER CODE END PV */
```

CGI = Common Gateway Interface
SSI = Server Side Includes

Purpose and Test procedure:

UM1713 User manual

Developing applications on STM32Cube

with LwIP TCP/IP stack

Section 6.2 Features Applications

6.2.1 Web Server Based on Raw API

Add Code to **main.c**

```
/* Private user code */
/* USER CODE BEGIN 0 */
// Initialize the CGI handlers
void myCGIinit(void)
{
    // add LED control CGI to the table
    theCGItable[0] = LedCGI;

    // give the table to the HTTP server
    http_set_cgi_handlers(theCGItable, 1);
} // myCGIinit

/**** CGI handler for controlling the LEDs ****/
// the function pointer for a CGI script handler is defined in httpd.h as tCGIHandler
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[])
{
    uint32_t i = 0;
    // index of the CGI within the theCGItable array passed to http_set_cgi_handlers
    // Given how this example is structured, this may be a redundant check.
    // Here there is only one handler iIndex == 0
    if (iIndex == 0)
    {
        // turn off the LEDs
        HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
```

Add Code to **main.c**

```
        // Check the cgi parameters, e.g., GET /leds.cgi?led=1&led=2&led=3
        for (i=0; i<iNumParams; i++)
        {
            // if pcParmeter contains "led", then one of the LED check boxes has been set on
            if (strcmp(pcParam[i], "led") == 0)
            {
                // see if checkbox for LED 1 has been set
                if(strcmp(pcValue[i], "1") == 0)
                {
                    // switch led 1 ON if 1
                    HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_SET);
                }
                // see if checkbox for LED 2 has been set
                else if(strcmp(pcValue[i], "2") == 0)
                {
                    // switch led 2 ON if 2
                    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
                }
                // see if checkbox for LED 3 has been set
                else if(strcmp(pcValue[i], "3") == 0)
                {
                    // switch led 3 ON if 3
                    HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
                }
            } // if
        } // for
    } // if
    // uniform resource identifier to send after CGI call, i.e., path and filename of the response
    return "/index.shtml";
} // LedCGIhandler
```

Hands-On LwIP HTTP Server Raw CGI SSI

```
/**** SSI handler ****/
 // This function is called each time the HTTPD server detects a tag of the form
 // <!--#name--> in a .shtml, .ssi or .shtm file
 // It won't work if the file has a .html extension.
 u16_t mySSIHandler(int iIndex, char *pcInsert, int iInsertLen)
 {
     // see which tag in the array theSSItags to handle
     if (iIndex == 0) // is “tag1”
     {
         char myStr1[] = "Liaw, Hello from Tag #1!"; // string to be displayed on web page
         // copy the string to be displayed to pcInsert
         strcpy(pcInsert, myStr1);
         // return number of characters that need to be inserted in html
         return strlen(myStr1);
     }
     else if (iIndex == 1) // is “tag2”
     {
         char myStr2[] = "Hwee Choo, Hello from Tag #2!"; //string to be displayed on web page
         // copy string to be displayed
         strcpy(pcInsert, myStr2);
         // return number of characters that need to be inserted in html
         return strlen(myStr2);
     }
     return 0;
 } // mySSIHandler

 /**** Initialize SSI handlers ****/
  void mySSIinit(void)
  {
      // configure SSI handler function
      // theSSItags is an array of SSI tag strings to search for in SSI-enabled files
      http_set_ssi_handler(mySSIHandler, (char const **)theSSItags, numSSItags);
  } // mySSIinit
/* USER CODE END 0 */
```

Add Code to **main.c**

```c
/* @brief  The application entry point */
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_LWIP_Init();
    /* USER CODE BEGIN 2 */
    //start the web server
    httpd_init();

    //initialise the CGI handlers
    myCGIinit();

    // initialize the SSI handlers
    mySSIinit();
    /* USER CODE END 2 */
    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        MX_LWIP_Process();
       /* USER CODE END WHILE */
       /* USER CODE BEGIN 3 */
    }
    /* USER CODE END 3 */
}
```

The `index.shtml`

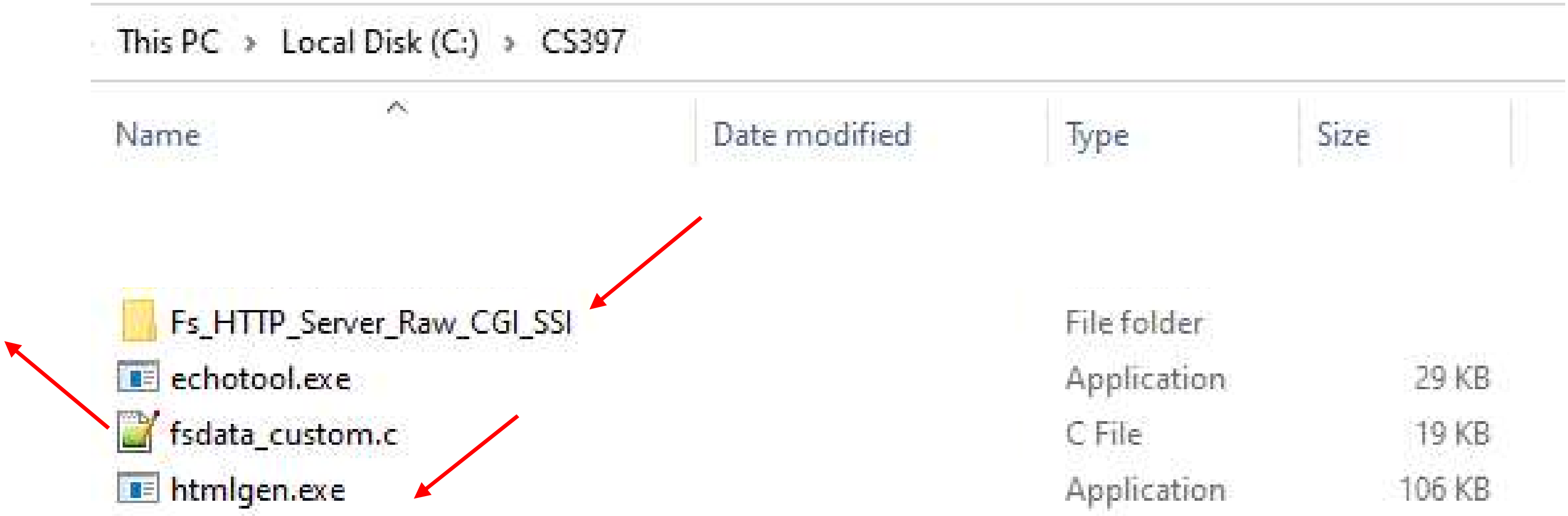
```html
<!DOCTYPE html>
<html><head>
<title>LED Test</title></head>
<body>
<p>This web page allows you to control the LEDs: LED1, LED2 and LED3.</p>
<p>You must select and click on "Send" button to change the LEDs.</p>
<form method="get" action="/leds.cgi">
<input value="1" name="led" type="checkbox">LED1<br>
<input value="2" name="led" type="checkbox">LED2<br>
<input value="3" name="led" type="checkbox">LED3<br>
<br>
<p>text for tag1: <!--#tag1--></p>
<p>text for tag2: <!--#tag2--></p>
<br>
<input value="Send" type="submit"> </form>
<p>Modified by Liaw Hwee Choo, June 2022</p>
</body></html>
```

Hands-On LwIP HTTP Server Raw CGI SSI

Generate the `fsdata_custom.c`

1 **Unzip 12_CS397_Hands-On_5-2_LwIP_HTTP_Server_Raw_CGI_SSI.zip**

2 Copy folder "Fs_HTTP_Server_Raw_CGI_SSI" to c:\CS397



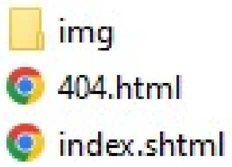
This PC › Local Disk (C:) › CS397

Name	Date modified	Type	Size
Fs_HTTP_Server_Raw_CGI_SSI		File folder	
echotool.exe		Application	29 KB
fsdata_custom.c		C File	19 KB
htmlgen.exe		Application	106 KB

3 Run

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

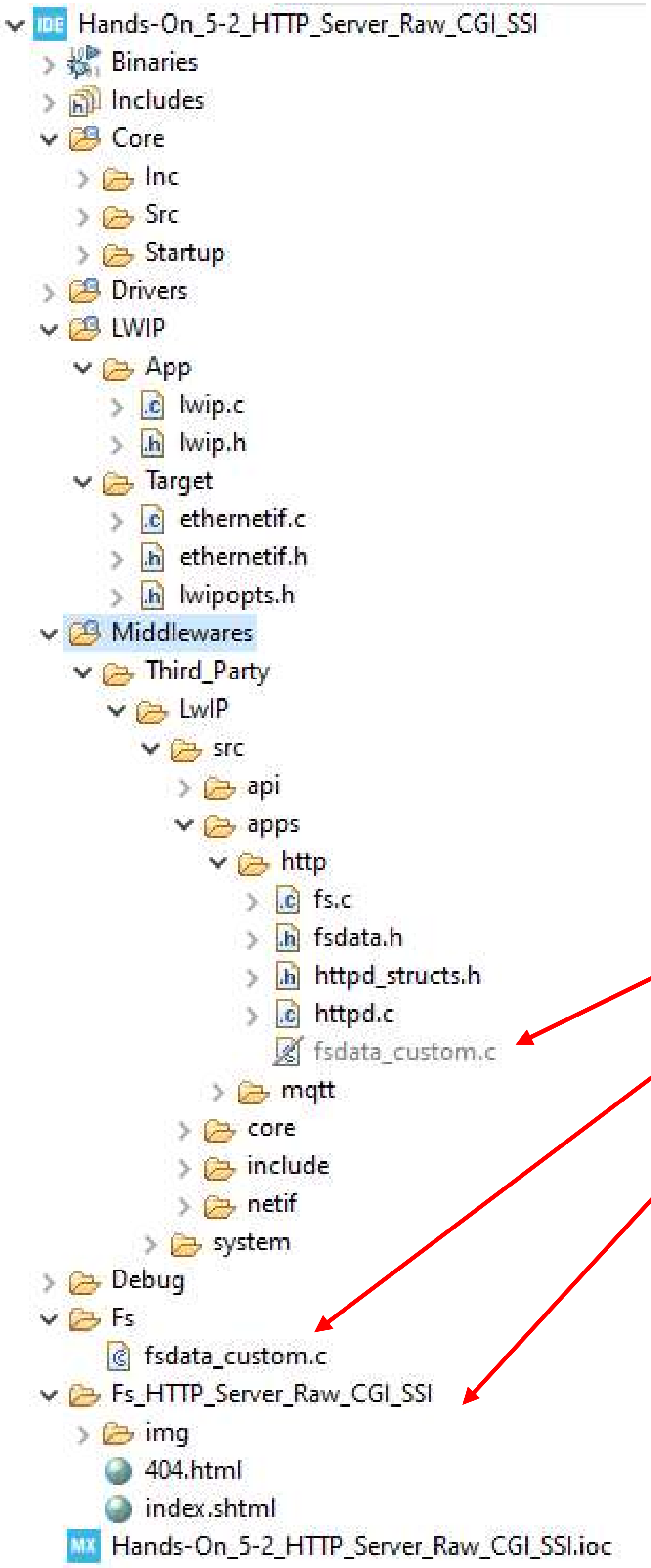
```
C:\CS397\Fs_HTTP_Server_Raw_CGI_SSI
```



img
404.html
index.shtml

4 Copy folder and generated file "fsdata_custom.c" to STM32 project

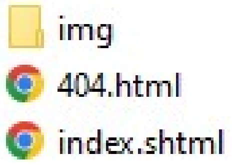
Hands-On LwIP HTTP Server Raw CGI SSI



```
v Hands-On_5-2_HTTP_Server_Raw_CGI_SSI
  > Binaries
  > Includes
  v Core
    > Inc
    > Src
    > Startup
  > Drivers
  v LWIP
    v App
      > lwip.c
      > lwip.h
    v Target
      > ethernetif.c
      > ethernetif.h
      > lwipopts.h
  v Middlewares
    v Third_Party
      v LwIP
        v src
          > api
          v apps
            v http
              > fs.c
              > fsdata.h
              > httpd_structs.h
              > httpd.c
                fsdata_custom.c
            > mqtt
          > core
          > include
          > netif
        > system
  > Debug
  v Fs
      fsdata_custom.c
  v Fs_HTTP_Server_Raw_CGI_SSI
    > img
      404.html
      index.shtml
  Hands-On_5-2_HTTP_Server_Raw_CGI_SSI.ioc
```

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

C:\CS397\Fs_HTTP_Server_Raw_CGI_SSI

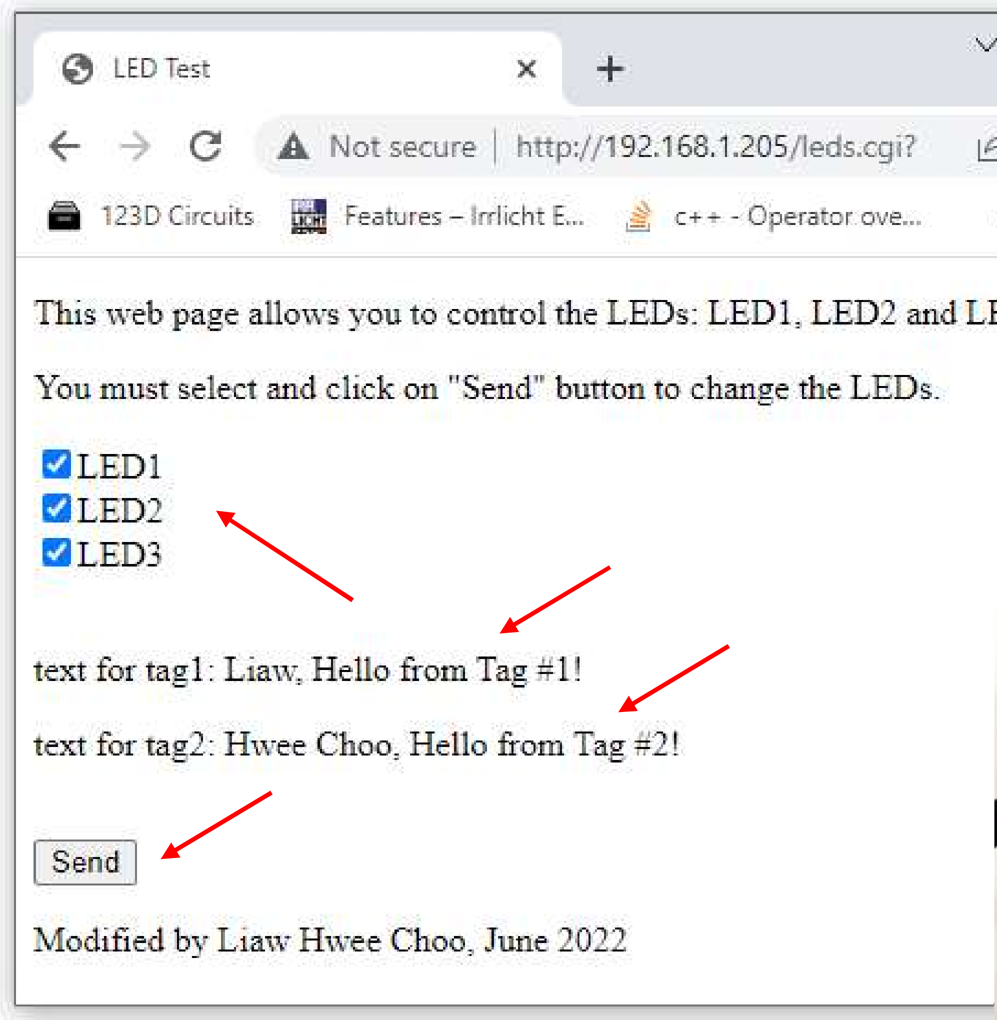


```
img
404.html
index.shtml
```

Hands-On LwIP HTTP Server Raw CGI SSI

When the project code is running on the Nucleo board, you should be able to access the web page by typing http://192.168.1.205 onto a web browser. At this time, you should be able to view the web page on the browser and perform the implemented function.

Note: Remember to reset the MCU board



This web page allows you to control the LEDs: LED1, LED2 and LED3.

You must select and click on "Send" button to change the LEDs.

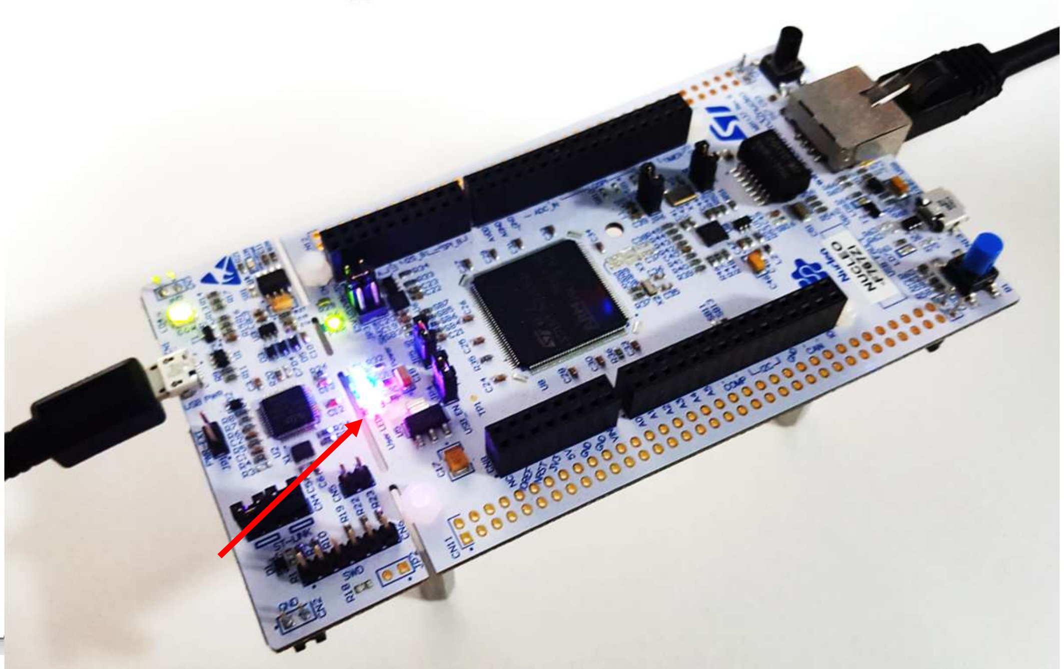
☑ LED1
☑ LED2
☑ LED3

text for tag1: Liaw, Hello from Tag #1!

text for tag2: Hwee Choo, Hello from Tag #2!

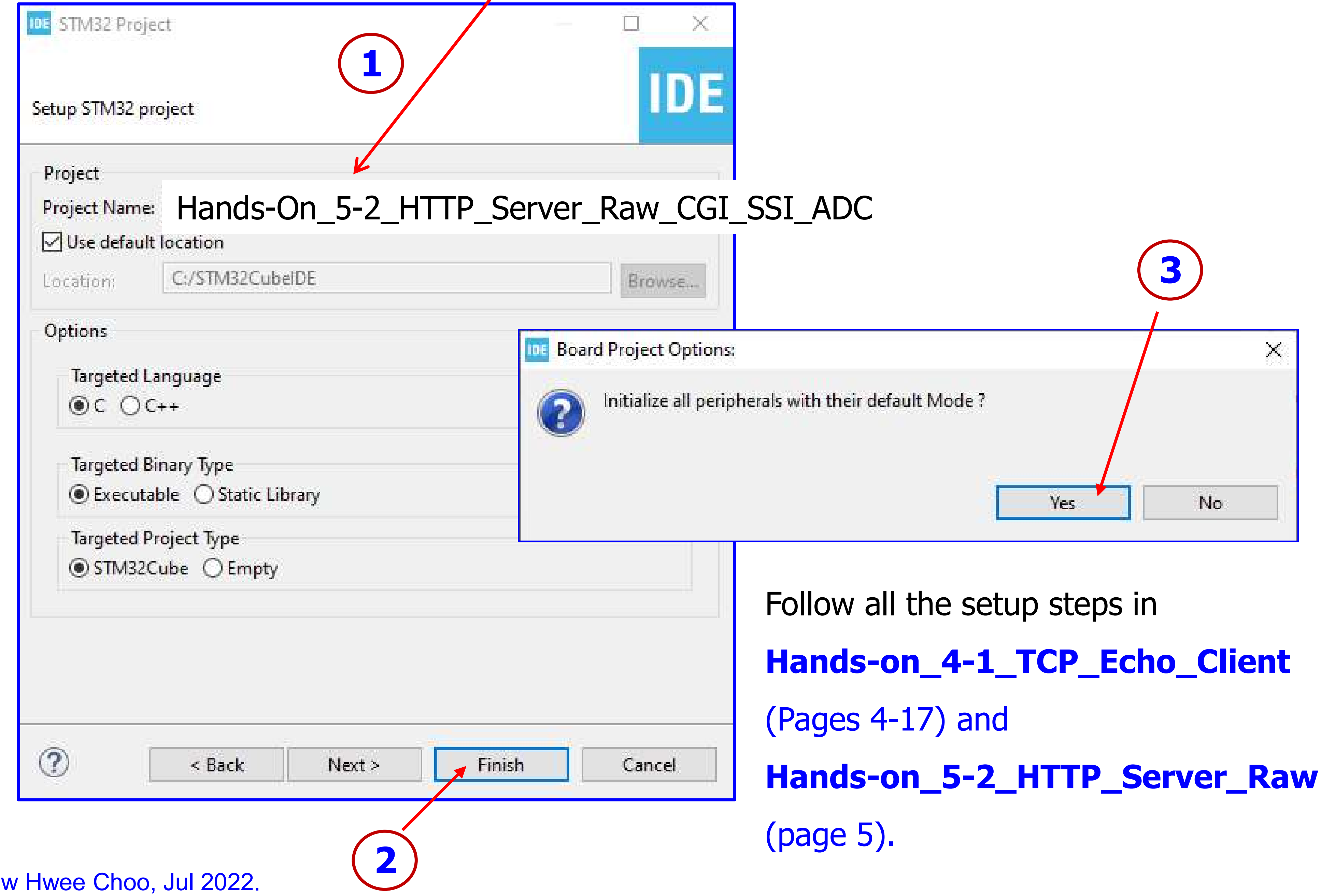
Send

Modified by Liaw Hwee Choo, June 2022



Hands-On LwIP HTTP Server Raw CGI SSI ADC

Enter Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC



STM32 Project

Setup STM32 project

Project

Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC

☑ Use default location

Location: C:/STM32CubeIDE

Options

Targeted Language

● C ○ C++

Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

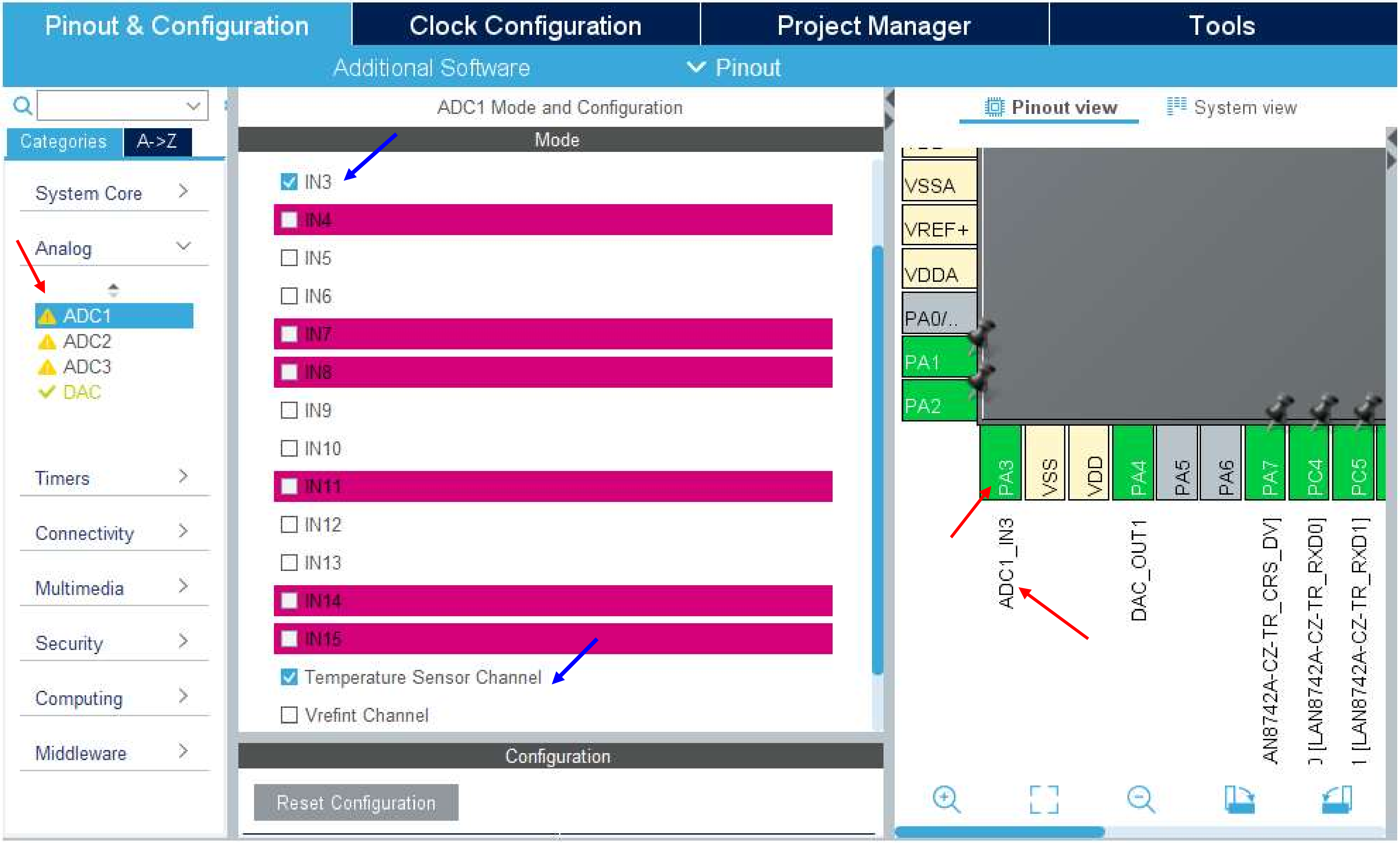
Board Project Options:

Initialize all peripherals with their default Mode ?

Follow all the setup steps in **Hands-on_4-1_TCP_Echo_Client** (Pages 4-17) and **Hands-on_5-2_HTTP_Server_Raw** (page 5).

Hands-On LwIP HTTP Server Raw CGI SSI ADC

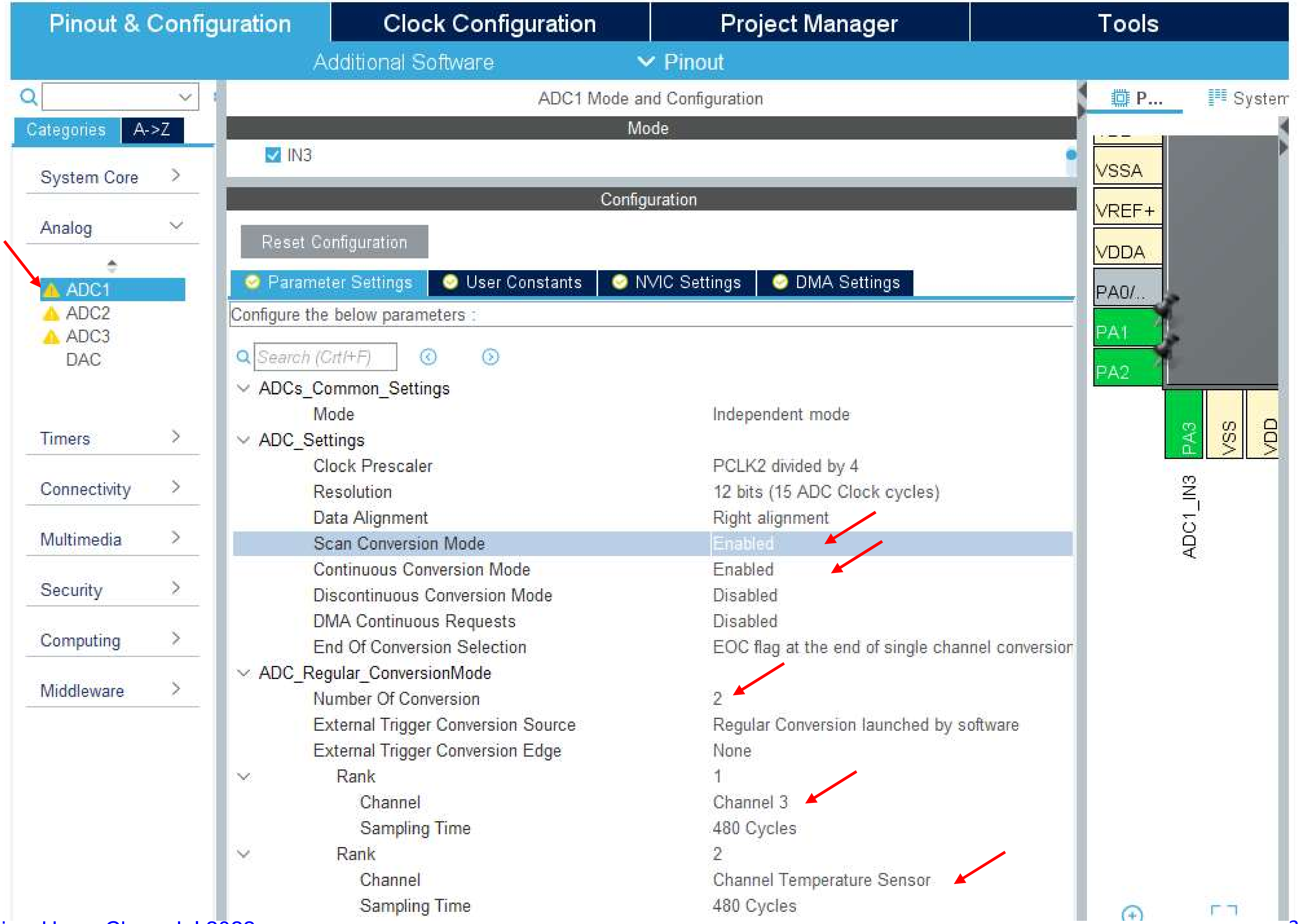
Add ADC1, enabled IN3 and Temperature Sensor Channel



☑ IN3
□ IN4
□ IN5
□ IN6
□ IN7
□ IN8
□ IN9
□ IN10
□ IN11
□ IN12
□ IN13
□ IN14
□ IN15
☑ Temperature Sensor Channel
□ Vrefint Channel

Liaw Hwee Choo, Jul 2022.

Hands-On LwIP HTTP Server Raw CGI SSI ADC



ADC1 Mode and Configuration

Mode

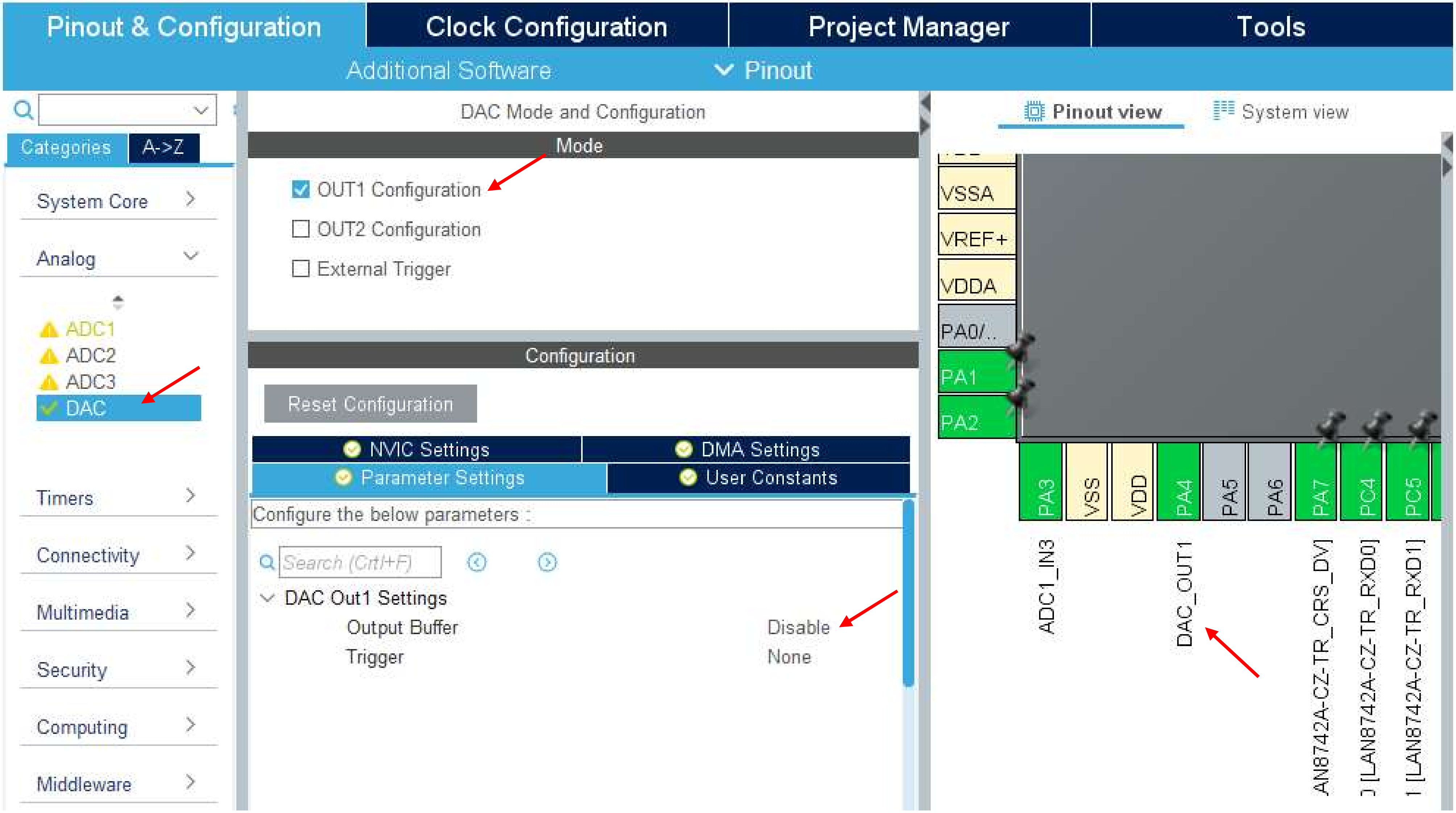
☑ IN3

Configuration

Configure the below parameters :

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	2
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 3
Sampling Time	480 Cycles
Rank	2
Channel	Channel Temperature Sensor
Sampling Time	480 Cycles

Add DAC, enabled OUT1, and disabled Output Buffer



☑ OUT1 Configuration
☐ OUT2 Configuration
☐ External Trigger

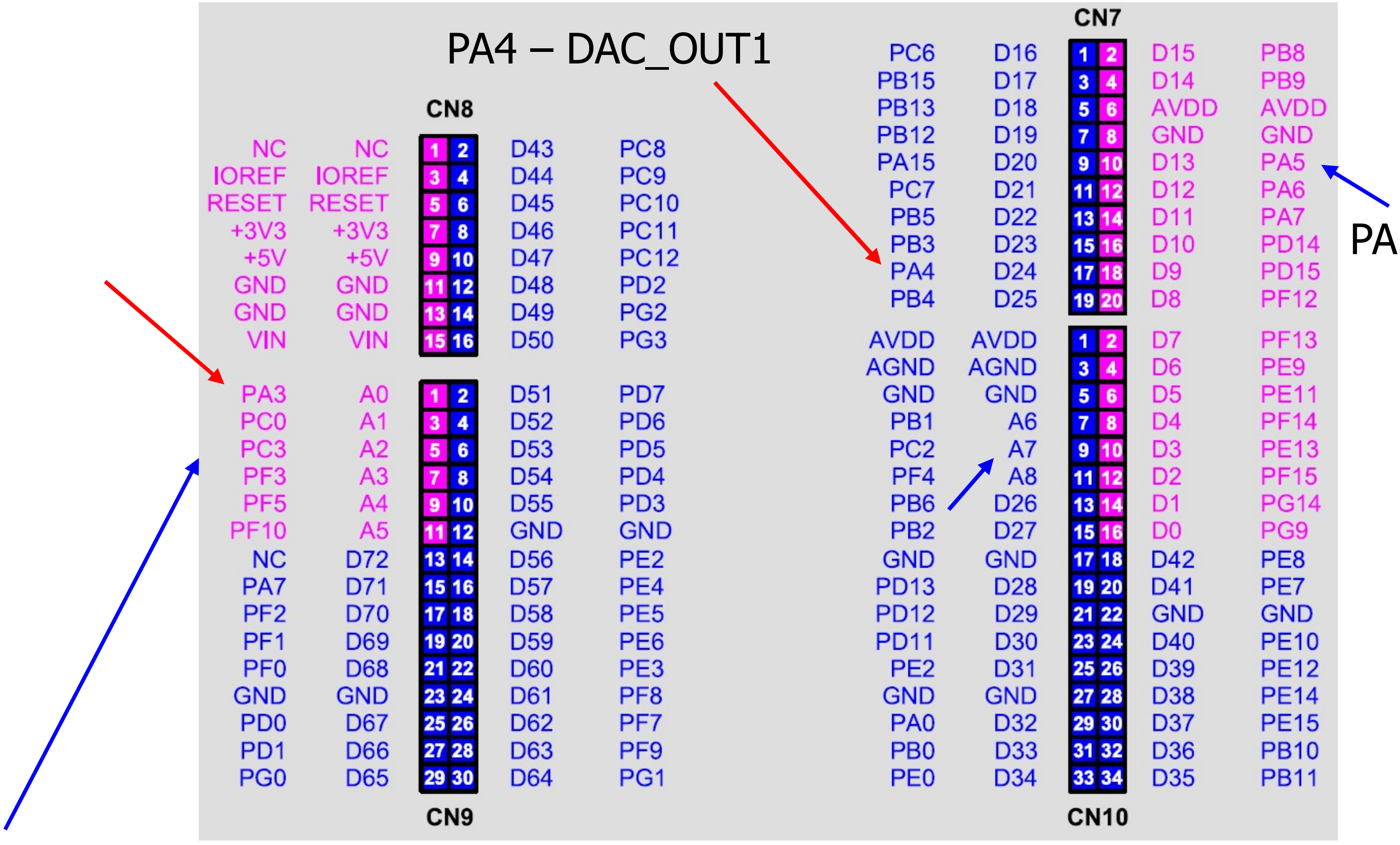
Configure the below parameters :

DAC Out1 Settings

Output Buffer	Disable
Trigger	None

Hands-On LwIP HTTP Server Raw CGI SSI ADC

Nucleo-F767ZI Board Pinout – DAC and ADC Pins on ST Zio



PA4 – DAC_OUT1

PA5 – DAC_OUT2

CN8

NC	NC	1	2	D43	PC8
IOREF	IOREF	3	4	D44	PC9
RESET	RESET	5	6	D45	PC10
+3V3	+3V3	7	8	D46	PC11
+5V	+5V	9	10	D47	PC12
GND	GND	11	12	D48	PD2
GND	GND	13	14	D49	PG2
VIN	VIN	15	16	D50	PG3

PA3	A0	1	2	D51	PD7
PC0	A1	3	4	D52	PD6
PC3	A2	5	6	D53	PD5
PF3	A3	7	8	D54	PD4
PF5	A4	9	10	D55	PD3
PF10	A5	11	12	GND	GND
NC	D72	13	14	D56	PE2
PA7	D71	15	16	D57	PE4
PF2	D70	17	18	D58	PE5
PF1	D69	19	20	D59	PE6
PF0	D68	21	22	D60	PE3
GND	GND	23	24	D61	PF8
PD0	D67	25	26	D62	PF7
PD1	D66	27	28	D63	PF9
PG0	D65	29	30	D64	PG1

CN9

CN7

PC6	D16	1	2	D15	PB8
PB15	D17	3	4	D14	PB9
PB13	D18	5	6	AVDD	AVDD
PB12	D19	7	8	GND	GND
PA15	D20	9	10	D13	PA5
PC7	D21	11	12	D12	PA6
PB5	D22	13	14	D11	PA7
PB3	D23	15	16	D10	PD14
PA4	D24	17	18	D9	PD15
PB4	D25	19	20	D8	PF12

AVDD	AVDD	1	2	D7	PF13
AGND	AGND	3	4	D6	PE9
GND	GND	5	6	D5	PE11
PB1	A6	7	8	D4	PF14
PC2	A7	9	10	D3	PE13
PF4	A8	11	12	D2	PF15
PB6	D26	13	14	D1	PG14
PB2	D27	15	16	D0	PG9
GND	GND	17	18	D42	PE8
PD13	D28	19	20	D41	PE7
PD12	D29	21	22	GND	GND
PD11	D30	23	24	D40	PE10
PE2	D31	25	26	D39	PE12
GND	GND	27	28	D38	PE14
PA0	D32	29	30	D37	PE15
PB0	D33	31	32	D36	PB10
PE0	D34	33	34	D35	PB11

CN10

PA3 (A0) – ADC1_IN3, ADC2_IN3, ADC3_IN3
PC0 (A1) – ADC1_IN10, ADC2_IN10, ADC3_IN10
PC3 (A2) – ADC1_IN13, ADC2_IN13, ADC3_IN13
PF3 (A3) – ADC3_IN9
PF5 (A4) – ADC3_IN15
PF10 (A5) – ADC3_IN8

PB1 (A6) – ADC1_IN9, ADC2_IN9
PC2 (A7) – ADC1_IN12, ADC2_IN12, ADC3_IN12
PF4 (A8) – ADC3_IN14

Hands-On LwIP HTTP Server Raw CGI SSI ADC

Add Code to **main.c**

```c
/* Private includes */
/* USER CODE BEGIN Includes */
#include "lwip/apps/httpd.h"
#include <string.h>
#include <stdlib.h>
/* USER CODE END Includes */

/* Private typedef */
/* USER CODE BEGIN PTD */
// array of tags for the SSI handler
// these are the tags <!--#tag1--> contained in the shtml file
#define numSSItags   2
/* USER CODE END PTD */

/* Private variables */
/* USER CODE BEGIN PV */
uint32_t adc[2];
uint32_t value_dac = 0;
float vsense = 3.3 / 4096.0;
float temperature = 0.0;
char  str[50];
char const *theSSItags[numSSItags] = {"tag1","tag2"};

// prototype CGI handler for the LED control
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[]);
// this structure contains the name of the LED CGI and corresponding handler for the LEDs
const tCGI LedCGI={"/leds.cgi", LedCGIhandler};

// table of the CGI names and handlers
tCGI theCGItable[1];
/* USER CODE END PV */
```

Purpose and Test procedure:

UM1713 User manual

Developing applications on STM32Cube with LwIP TCP/IP stack

Section 6.2 Features Applications

6.2.1 Web Server Based on Raw API

CGI = Common Gateway Interface
SSI = Server Side Includes

Hands-On LwIP HTTP Server Raw CGI SSI ADC

Add Code to **main.c**

```c
/* Private user code */
/* USER CODE BEGIN 0 */
// Initialize the CGI handlers
void myCGIinit(void)
{
    // add LED control CGI to the table
    theCGItable[0] = LedCGI;

    // give the table to the HTTP server
    http_set_cgi_handlers(theCGItable, 1);
} // myCGIinit

/**** CGI handler for controlling the LEDs ****/
// the function pointer for a CGI script handler is defined in httpd.h as tCGIHandler
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[])
{
    uint32_t i = 0;
    // index of the CGI within the theCGItable array passed to http_set_cgi_handlers
    // Given how this example is structured, this may be a redundant check.
    // Here there is only one handler iIndex == 0
    if (iIndex == 0)
    {
        // turn off the LEDs
        HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
```

Add Code to **main.c**

```
        // Check the cgi parameters, e.g., GET /leds.cgi?led=1&led=2&led=3
        for (i=0; i<iNumParams; i++)
        {
            // if pcParmeter contains "led", then one of the LED check boxes has been set on
            if (strcmp(pcParam[i], "led") == 0)
            {
                // see if checkbox for LED 1 has been set
                if(strcmp(pcValue[i], "1") == 0)
                {
                    // switch led 1 ON if 1
                    HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_SET);
                }
                // see if checkbox for LED 2 has been set
                else if(strcmp(pcValue[i], "2") == 0)
                {
                    // switch led 2 ON if 2
                    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
                }
                // see if checkbox for LED 3 has been set
                else if(strcmp(pcValue[i], "3") == 0)
                {
                    // switch led 3 ON if 3
                    HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
                }
            } // if
        } // for
    } // if
    // uniform resource identifier to send after CGI call, i.e., path and filename of the response
    return "/index.shtml";
} // LedCGIhandler
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC

Add Code to **main.c**

```
/**** SSI handler ****/
 // This function is called each time the HTTPD server detects a tag of the form
 // <!--#name--> in a .shtml, .ssi or .shtm file
 // It won't work if the file has a .html extension.
 u16_t mySSIHandler(int iIndex, char *pcInsert, int iInsertLen)
 {
     HAL_ADC_Start(&hadc1);

     HAL_ADC_PollForConversion(&hadc1,100);
     adc[0] = HAL_ADC_GetValue(&hadc1);

     HAL_ADC_PollForConversion(&hadc1,100);
     adc[1] = HAL_ADC_GetValue(&hadc1);

     HAL_ADC_Stop(&hadc1); // stop ADC is important

     temperature = (( adc[1] * vsense - 0.76 ) / 0.0025) + 25;

     sprintf(str, "%4.2f",temperature); // need  -u _printf_float

     // see which tag in the array theSSItags to handle
     if (iIndex == 0) // is “tag1”
     {
         char Digit1 = 0, Digit2 = 0, Digit3 = 0, Digit4 = 0;
         uint32_t ADCVal = 0;

         /* convert to voltage, 12 bits, step = 3.3V / 4096 = 0.8056640625 mV */
         ADCVal = (uint32_t)(adc[0] * 0.8056640625);
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC

Add Code to **main.c**

```
         /* get digits to display */
         Digit1 =  ADCVal / 1000;
         Digit2 = (ADCVal-(Digit1*1000)) / 100;
         Digit3 = (ADCVal-((Digit1*1000)+(Digit2*100))) / 10;
         Digit4 =  ADCVal-((Digit1*1000)+(Digit2*100)+(Digit3*10));

         /* prepare data to be inserted in html */
         *pcInsert       = (char)(Digit1+0x30);  // ascii 0 = 48 (dec) or 0x30
         *(pcInsert + 1) = (char)(Digit2+0x30);
         *(pcInsert + 2) = (char)(Digit3+0x30);
         *(pcInsert + 3) = (char)(Digit4+0x30);

         /* 4 characters need to be inserted in html*/
         return 4;
     }
     else if (iIndex == 1) // is “tag2”
     {
         strcpy(pcInsert, str);
         return strlen(str);
     }
     return 0;
 } // mySSIHandler

 /**** Initialize SSI handlers ****/
  void mySSIinit(void)
  {
      // configure SSI handler function
      // theSSItags is an array of SSI tag strings to search for in SSI-enabled files
      http_set_ssi_handler(mySSIHandler, (char const **)theSSItags, numSSItags);
  } // mySSIinit
/* USER CODE END 0 */
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC

Add Code to **main.c**

```c
/* @brief  The application entry point */
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_LWIP_Init();
    MX_ADC1_Init();
    MX_DAC_Init();
    /* USER CODE BEGIN 2 */
    // start DAC and output value
    HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, value_dac);

    // start the web server
    httpd_init();

    // initialise the CGI handlers
    myCGIinit();

    // initialize the SSI handlers
    mySSIinit();
  /* USER CODE END 2 */

  /* Infinite loop */
```

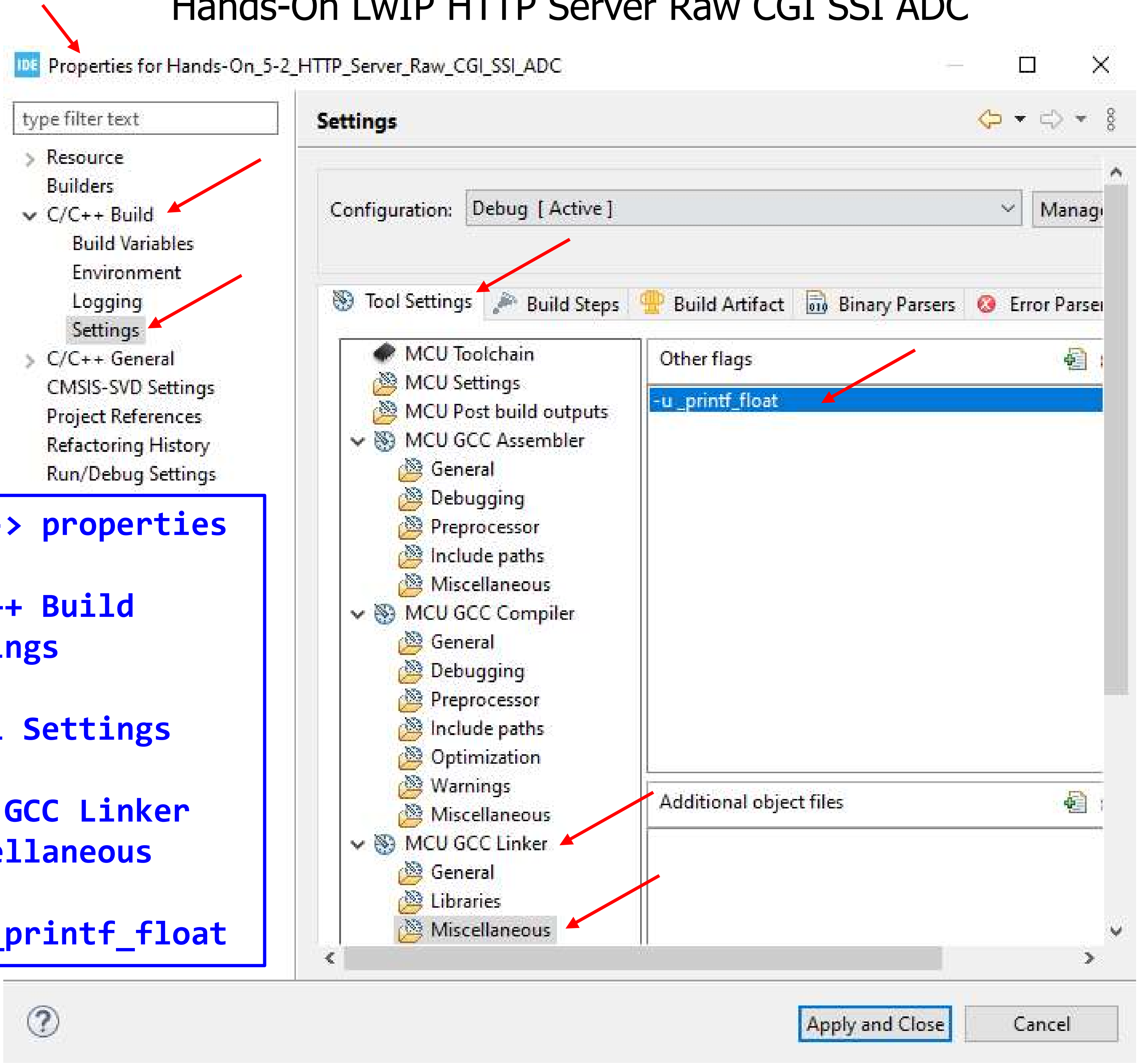
Hands-On LwIP HTTP Server Raw CGI SSI ADC

Add Code to **main.c**

For testing purposes

```
 /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
       MX_LWIP_Process();

       // HAL_GPIO_TogglePin(GPIOB, LD3_Pin);

       HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, value_dac);

       value_dac = value_dac + 1;

       if(value_dac > 4095)
       {
          value_dac = 0;
       }
       //  project -> properties -> C/C++ Build -> Settings - Tool Settings -> MCU GCC Linker
       //  - Miscellaneous
       //  Add: -u _printf_float

       printf("DAC: %ld  ADC: %ld  %ld  Temp: %4.2f degC \n\r",value_dac,adc[0],adc[1],temperature);

       HAL_Delay(100);

    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC



-u _printf_float

```
project -> properties

  -> C/C++ Build
  - Settings

  -> Tool Settings

  -> MCU GCC Linker
  - Miscellaneous

add: -u _printf_float
```

Add Code to **main.c**

```c
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_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 */
```

The index.shtml

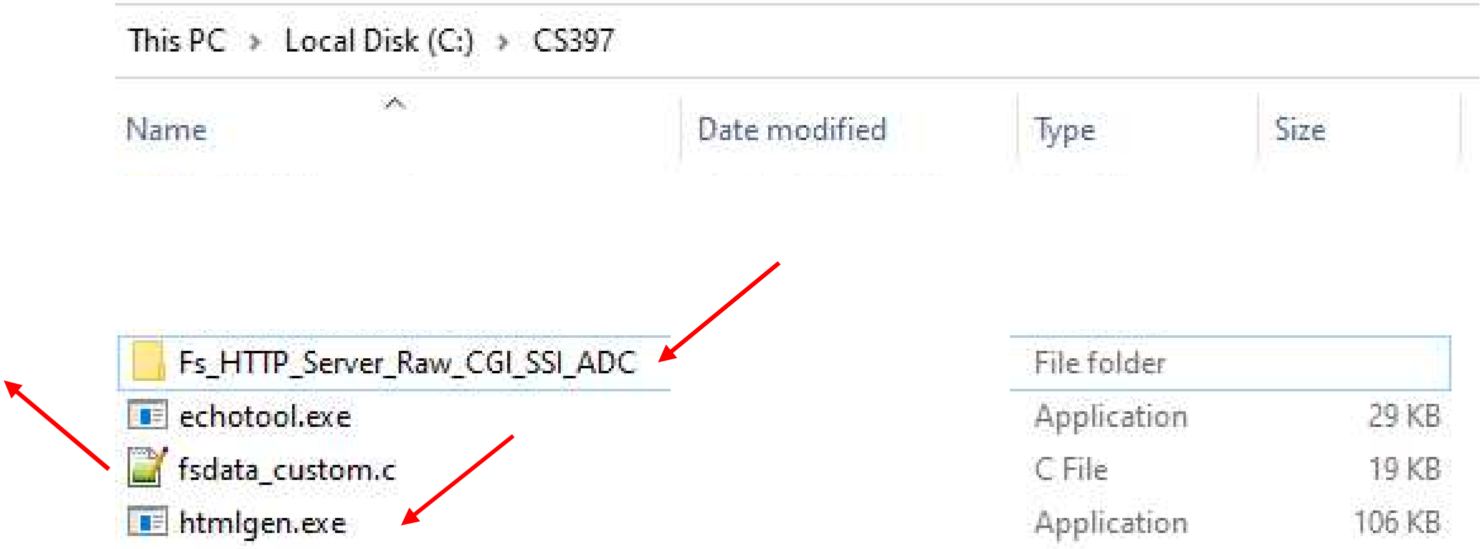
```
<!DOCTYPE html>
<html><head>
<title>LED Test</title></head>
<body>
<p>This web page allows you to control the LEDs: LED1, LED2 and LED3.</p>
<p>You must select and click on "Send" button to change the LEDs.</p>
<form method="get" action="/leds.cgi">
<input value="1" name="led" type="checkbox">LED1<br>
<input value="2" name="led" type="checkbox">LED2<br>
<input value="3" name="led" type="checkbox">LED3<br>
<br>
<p>text for tag1: <!--#tag1--></p>
<p>text for tag2: <!--#tag2--></p>
<br>
<input value="Send" type="submit"> </form>
<p>Modified by Liaw Hwee Choo, June 2022</p>
</body></html>
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC

Generate the `fsdata_custom.c`

1 **Unzip 12_CS397_Hands-On_5-2_LwIP_HTTP_Server_Raw_CGI_SSI.zip**

2 Copy folder “Fs_HTTP_Server_Raw_CGI_SSI_ADC” to c:\CS397



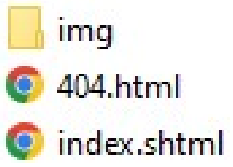
This PC › Local Disk (C:) › CS397

Name	Date modified	Type	Size
Fs_HTTP_Server_Raw_CGI_SSI_ADC		File folder	
echotool.exe		Application	29 KB
fsdata_custom.c		C File	19 KB
htmlgen.exe		Application	106 KB

3 Run

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

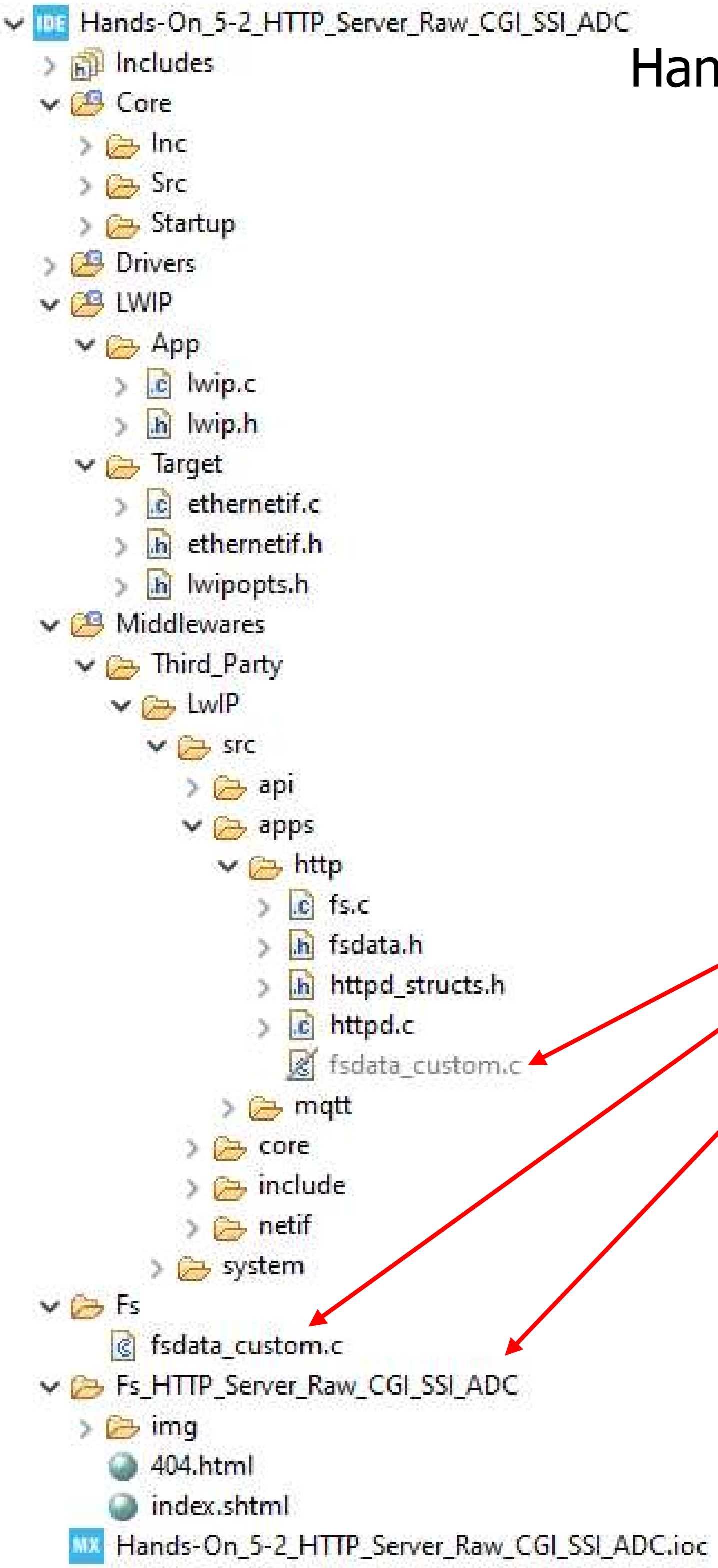
```
C:\CS397\Fs_HTTP_Server_Raw_CGI_SSI_ADC
```



img
404.html
index.shtml

4 Copy folder and generated file “fsdata_custom.c” to STM32 project

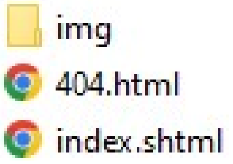
Hands-On LwIP HTTP Server Raw CGI SSI ADC



```
Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC
  Includes
  Core
    Inc
    Src
    Startup
  Drivers
  LWIP
    App
      lwip.c
      lwip.h
    Target
      ethernetif.c
      ethernetif.h
      lwipopts.h
  Middlewares
    Third_Party
      LwIP
        src
          api
          apps
            http
              fs.c
              fsdata.h
              httpd_structs.h
              httpd.c
              fsdata_custom.c
            mqtt
          core
          include
          netif
        system
  Fs
    fsdata_custom.c
  Fs_HTTP_Server_Raw_CGI_SSI_ADC
    img
    404.html
    index.shtml
  Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC.ioc
```

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

`C:\CS397\Fs_HTTP_Server_Raw_CGI_SSI_ADC`



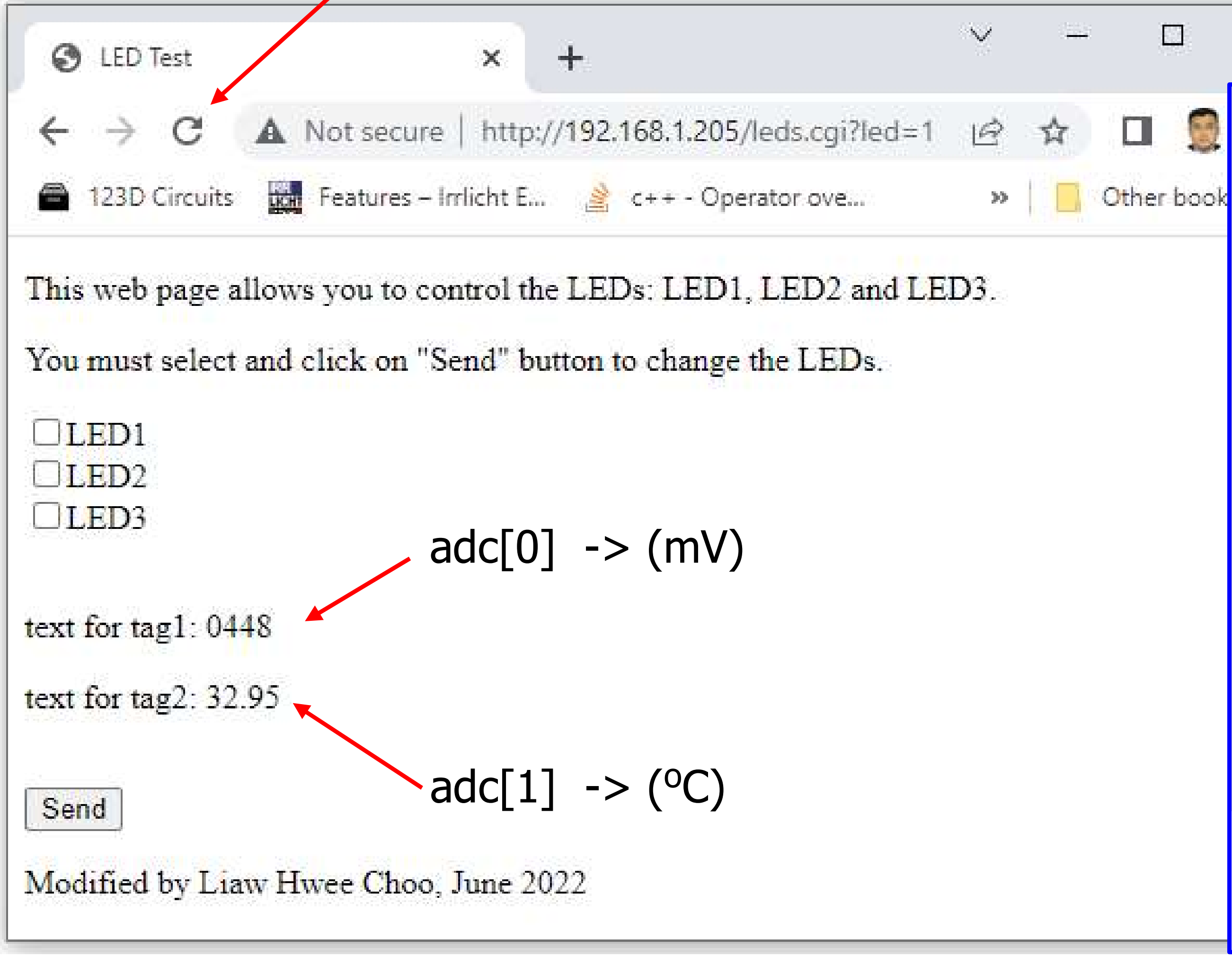
```
img
404.html
index.shtml
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC

When the project code is running on the Nucleo board, you should be able to access the web page by typing http://192.168.1.205 onto a web browser. At this time, you should be able to view the web page on the browser and perform the implemented function.

Note: Remember to reset the MCU board

Need to reload web page (F5) to update adc



This web page allows you to control the LEDs: LED1, LED2 and LED3.

You must select and click on "Send" button to change the LEDs.

□LED1
□LED2
□LED3

text for tag1: 0448

text for tag2: 32.95

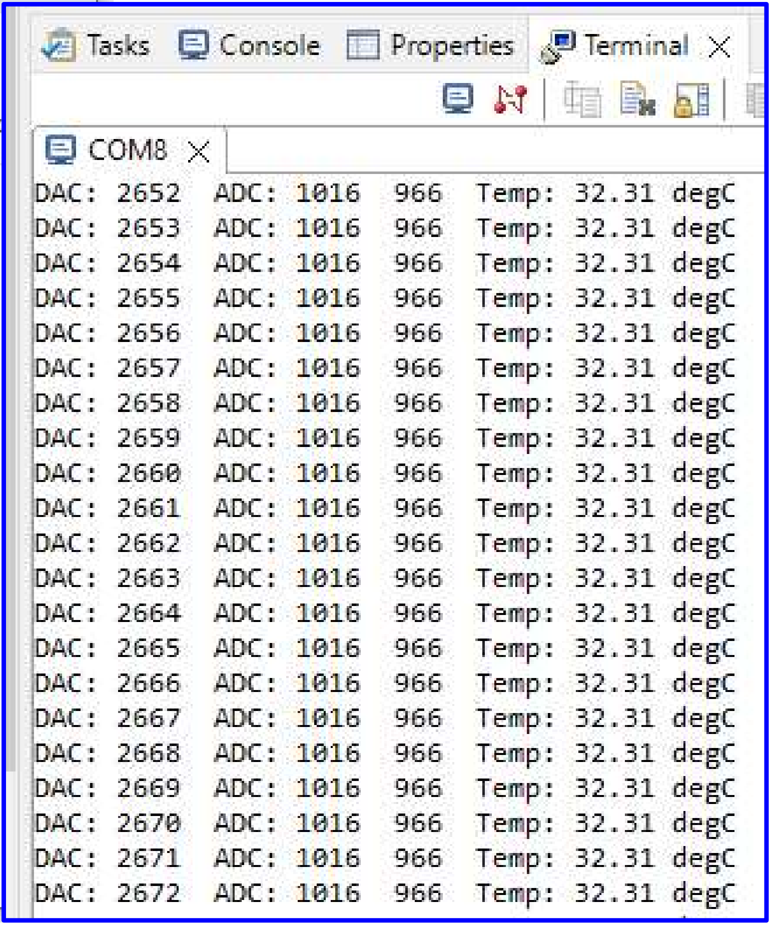
Send

Modified by Liaw Hwee Choo, June 2022

adc[0] -> (mV)

adc[1] -> (°C)

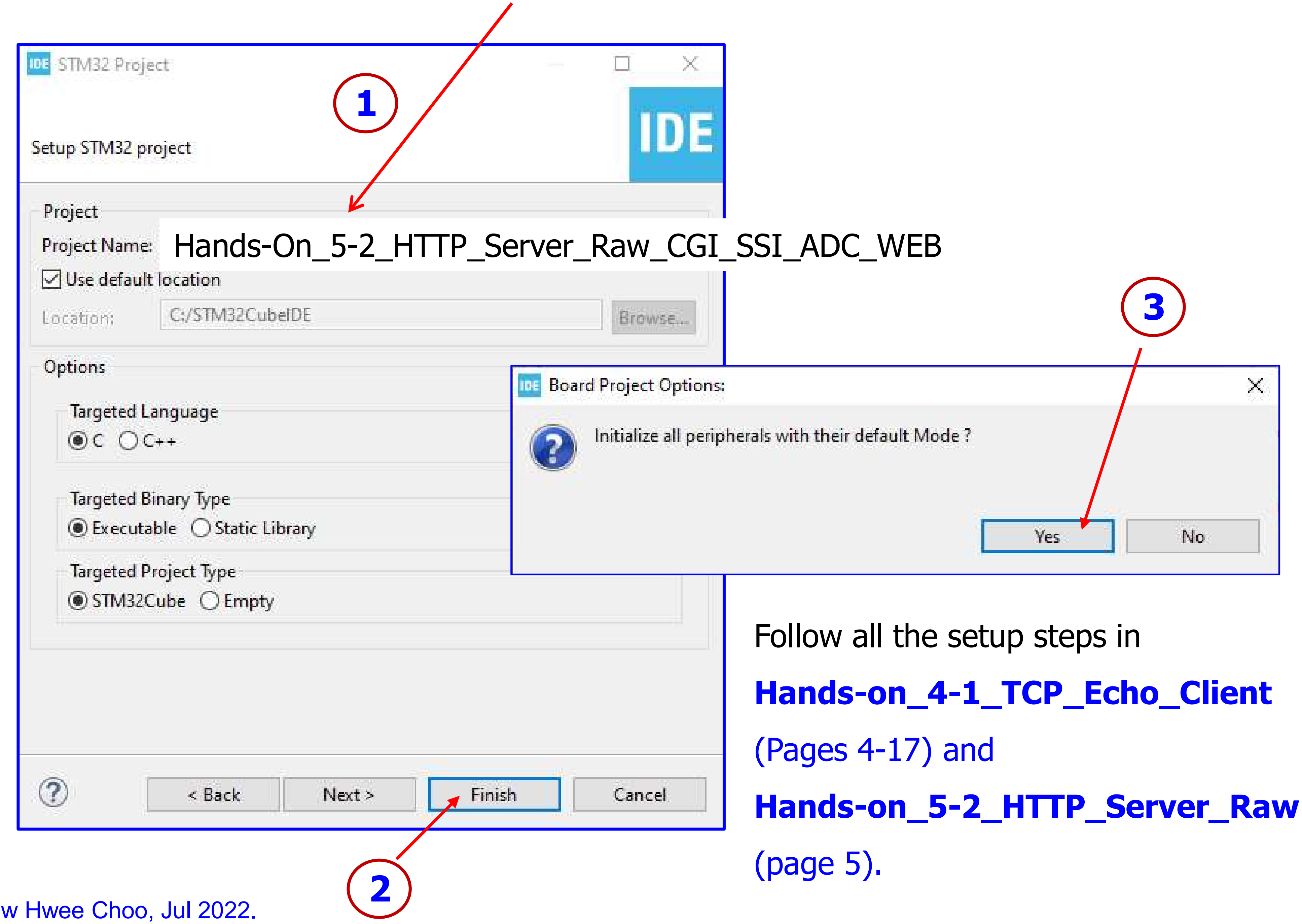
TM Terminal



```
DAC: 2652  ADC: 1016  966  Temp: 32.31 degC
DAC: 2653  ADC: 1016  966  Temp: 32.31 degC
DAC: 2654  ADC: 1016  966  Temp: 32.31 degC
DAC: 2655  ADC: 1016  966  Temp: 32.31 degC
DAC: 2656  ADC: 1016  966  Temp: 32.31 degC
DAC: 2657  ADC: 1016  966  Temp: 32.31 degC
DAC: 2658  ADC: 1016  966  Temp: 32.31 degC
DAC: 2659  ADC: 1016  966  Temp: 32.31 degC
DAC: 2660  ADC: 1016  966  Temp: 32.31 degC
DAC: 2661  ADC: 1016  966  Temp: 32.31 degC
DAC: 2662  ADC: 1016  966  Temp: 32.31 degC
DAC: 2663  ADC: 1016  966  Temp: 32.31 degC
DAC: 2664  ADC: 1016  966  Temp: 32.31 degC
DAC: 2665  ADC: 1016  966  Temp: 32.31 degC
DAC: 2666  ADC: 1016  966  Temp: 32.31 degC
DAC: 2667  ADC: 1016  966  Temp: 32.31 degC
DAC: 2668  ADC: 1016  966  Temp: 32.31 degC
DAC: 2669  ADC: 1016  966  Temp: 32.31 degC
DAC: 2670  ADC: 1016  966  Temp: 32.31 degC
DAC: 2671  ADC: 1016  966  Temp: 32.31 degC
DAC: 2672  ADC: 1016  966  Temp: 32.31 degC
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Enter Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC_WEB



STM32 Project

Setup STM32 project

Project

Project Name: Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC_WEB

☑ Use default location

Location: C:/STM32CubeIDE

Options

Targeted Language

● C ○ C++

Targeted Binary Type

● Executable ○ Static Library

Targeted Project Type

● STM32Cube ○ Empty

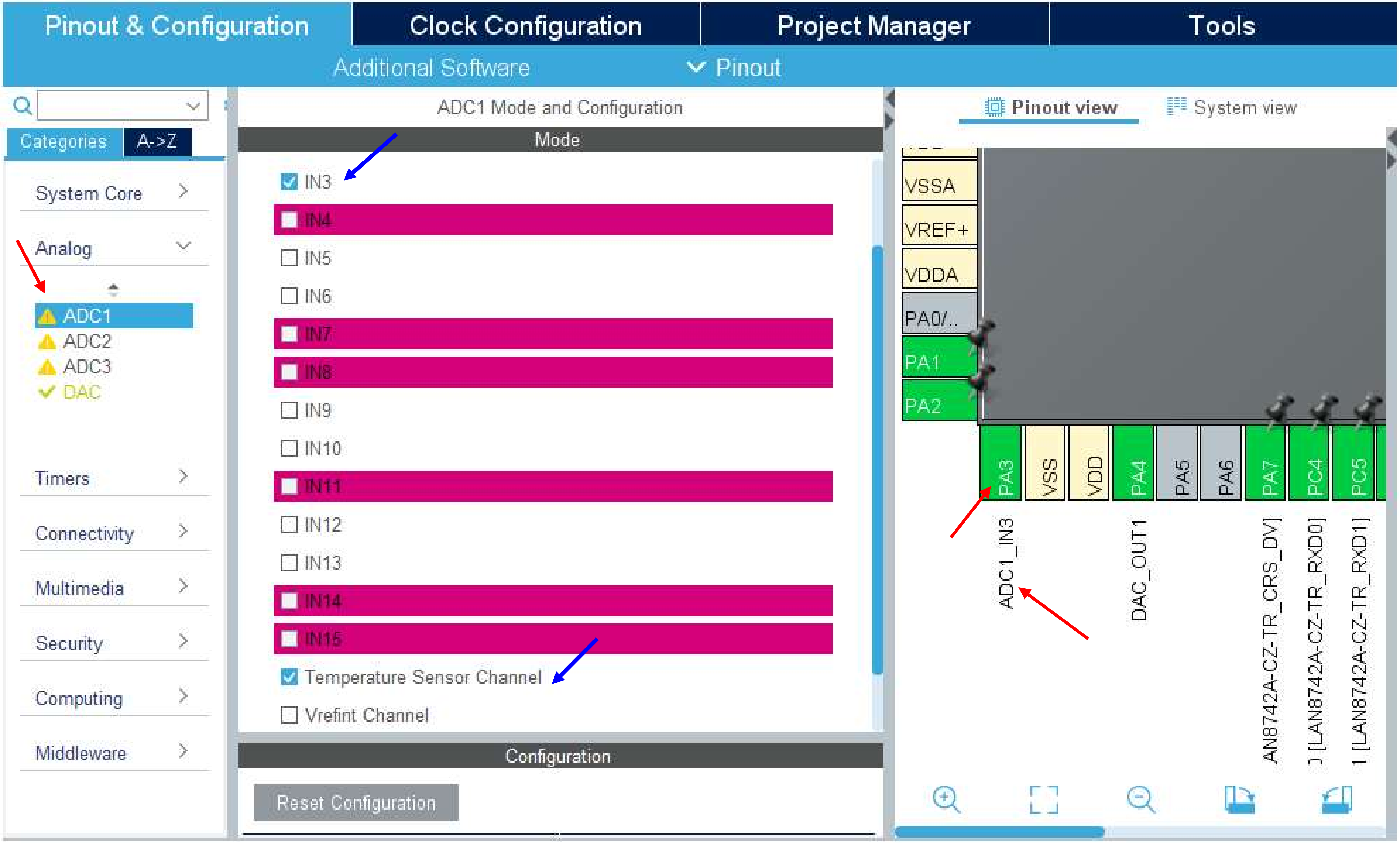
Board Project Options:

Initialize all peripherals with their default Mode ?

Follow all the setup steps in **Hands-on_4-1_TCP_Echo_Client** (Pages 4-17) and **Hands-on_5-2_HTTP_Server_Raw** (page 5).

Liaw Hwee Choo, Jul 2022.

Add ADC1, enabled IN3 and Temperature Sensor Channel

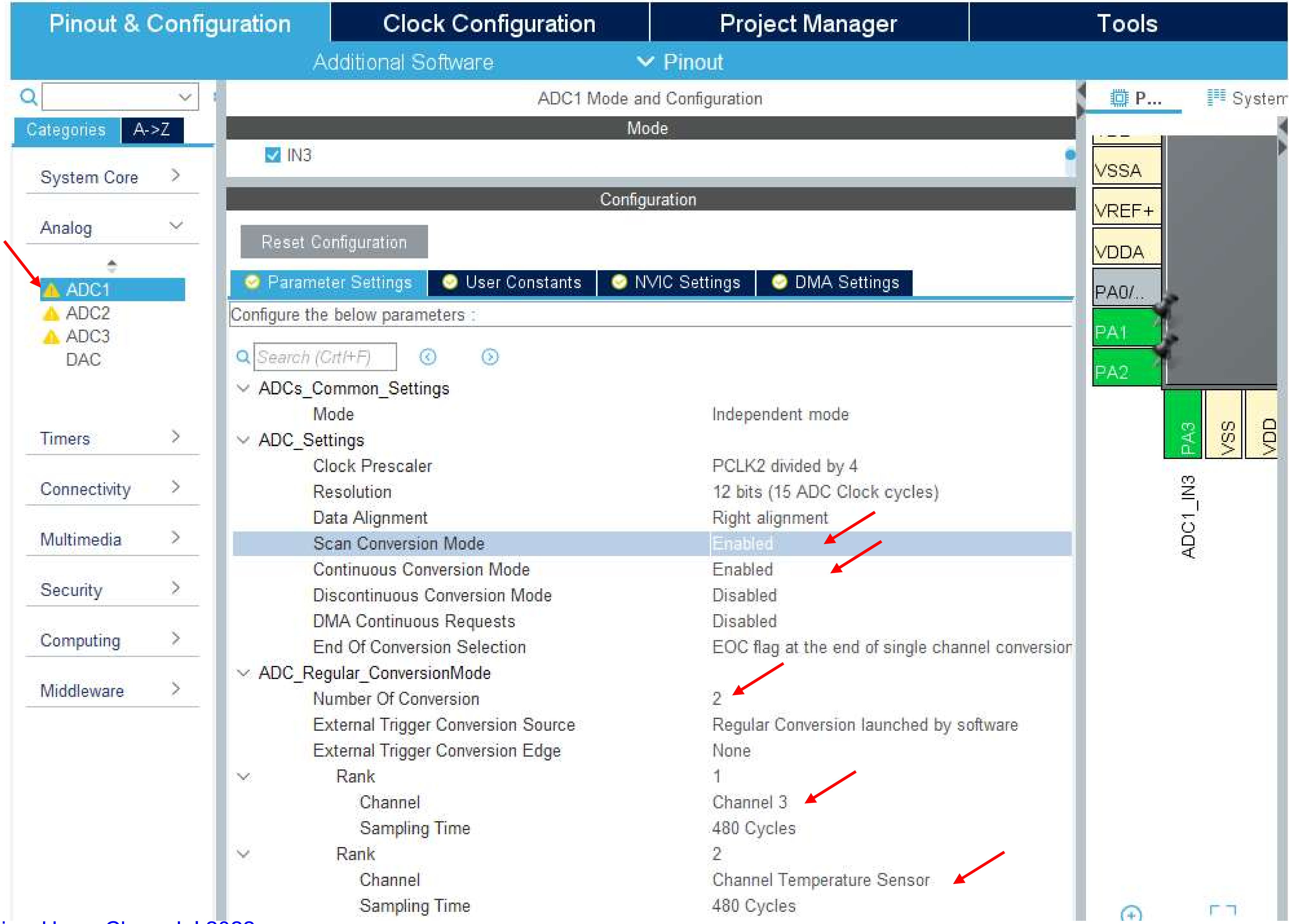


ADC1 Mode and Configuration

Mode

☑ IN3
□ IN4
□ IN5
□ IN6
□ IN7
□ IN8
□ IN9
□ IN10
□ IN11
□ IN12
□ IN13
□ IN14
□ IN15
☑ Temperature Sensor Channel
□ Vrefint Channel

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB



ADC1 Mode and Configuration

Mode

☑ IN3

Configuration

Reset Configuration

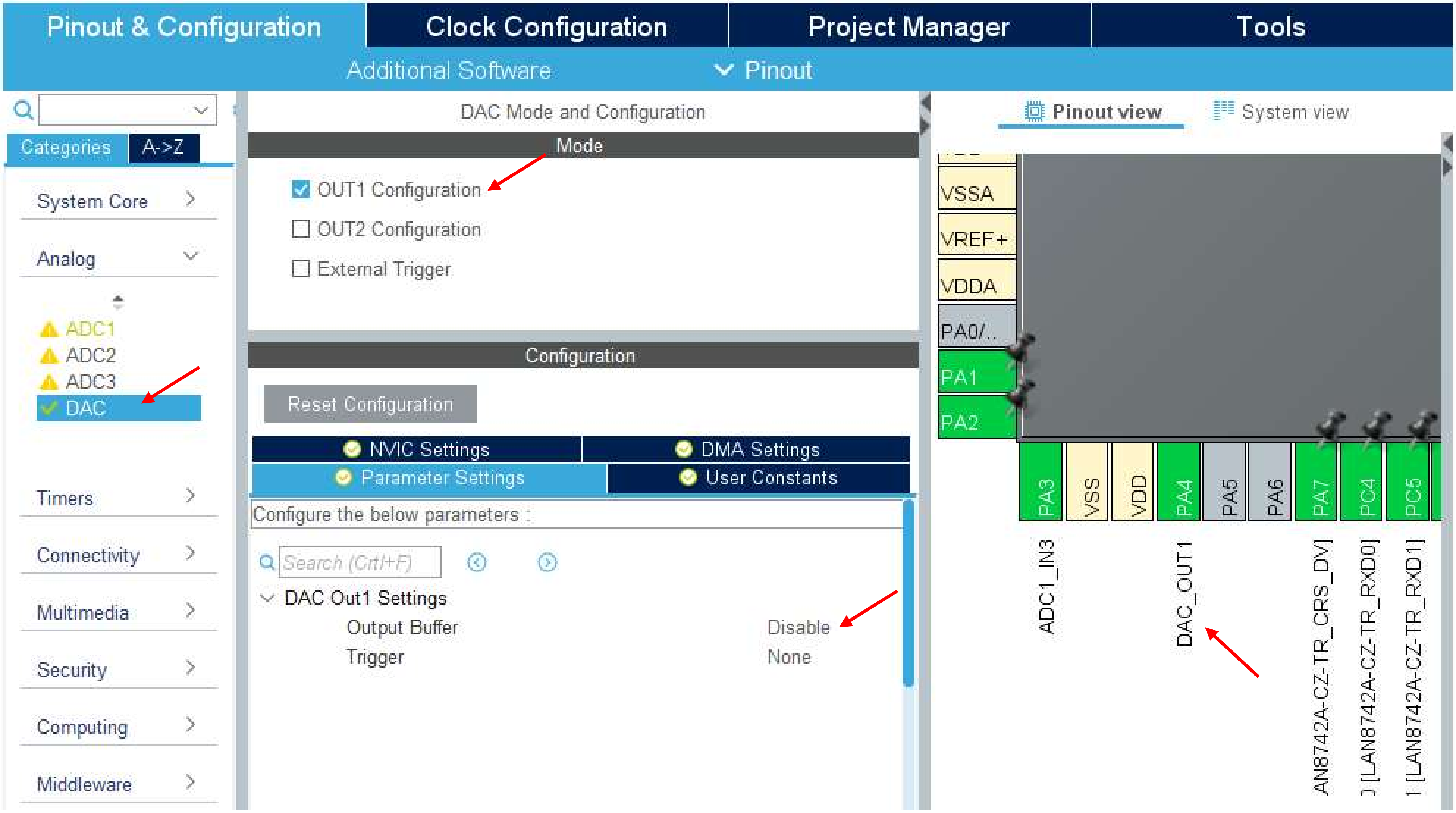
Parameter Settings | User Constants | NVIC Settings | DMA Settings

Configure the below parameters :

Search (Crtl+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	2
External Trigger Conversion Source	Regular Conversion launched by software
External Trigger Conversion Edge	None
Rank	1
Channel	Channel 3
Sampling Time	480 Cycles
Rank	2
Channel	Channel Temperature Sensor
Sampling Time	480 Cycles

Add DAC, enabled OUT1, and disabled Output Buffer



DAC Mode and Configuration

Mode

☑ OUT1 Configuration

□ OUT2 Configuration

□ External Trigger

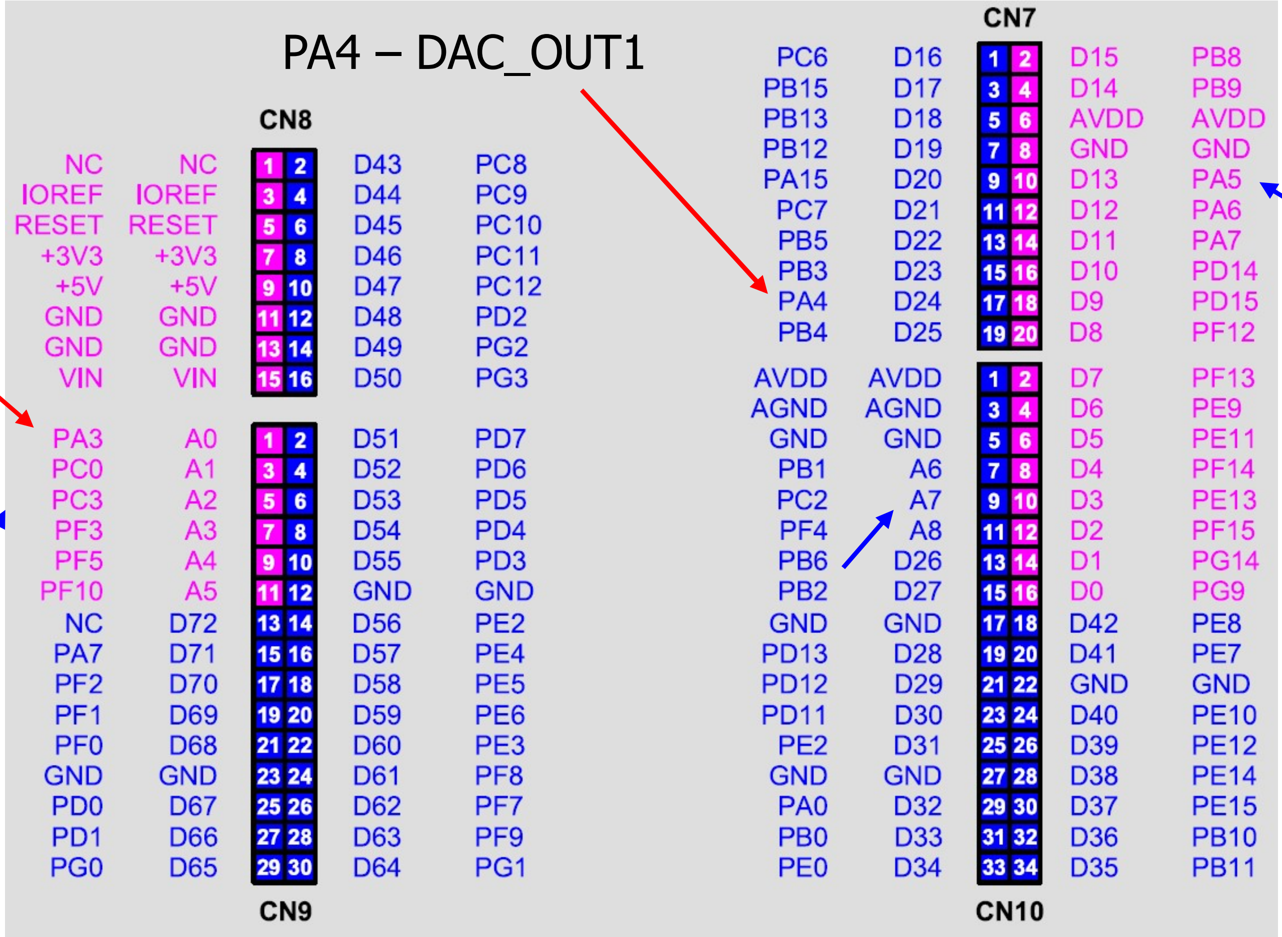
Configuration

Configure the below parameters :

DAC Out1 Settings

Output Buffer	Disable
Trigger	None

Nucleo-F767ZI Board Pinout – DAC and ADC Pins on ST Zio



PA4 – DAC_OUT1

CN8

NC	NC	1	2	D43	PC8
IOREF	IOREF	3	4	D44	PC9
RESET	RESET	5	6	D45	PC10
+3V3	+3V3	7	8	D46	PC11
+5V	+5V	9	10	D47	PC12
GND	GND	11	12	D48	PD2
GND	GND	13	14	D49	PG2
VIN	VIN	15	16	D50	PG3

PA3	A0	1	2	D51	PD7
PC0	A1	3	4	D52	PD6
PC3	A2	5	6	D53	PD5
PF3	A3	7	8	D54	PD4
PF5	A4	9	10	D55	PD3
PF10	A5	11	12	GND	GND
NC	D72	13	14	D56	PE2
PA7	D71	15	16	D57	PE4
PF2	D70	17	18	D58	PE5
PF1	D69	19	20	D59	PE6
PF0	D68	21	22	D60	PE3
GND	GND	23	24	D61	PF8
PD0	D67	25	26	D62	PF7
PD1	D66	27	28	D63	PF9
PG0	D65	29	30	D64	PG1

CN9

CN7

PC6	D16	1	2	D15	PB8
PB15	D17	3	4	D14	PB9
PB13	D18	5	6	AVDD	AVDD
PB12	D19	7	8	GND	GND
PA15	D20	9	10	D13	PA5
PC7	D21	11	12	D12	PA6
PB5	D22	13	14	D11	PA7
PB3	D23	15	16	D10	PD14
PA4	D24	17	18	D9	PD15
PB4	D25	19	20	D8	PF12

AVDD	AVDD	1	2	D7	PF13
AGND	AGND	3	4	D6	PE9
GND	GND	5	6	D5	PE11
PB1	A6	7	8	D4	PF14
PC2	A7	9	10	D3	PE13
PF4	A8	11	12	D2	PF15
PB6	D26	13	14	D1	PG14
PB2	D27	15	16	D0	PG9
GND	GND	17	18	D42	PE8
PD13	D28	19	20	D41	PE7
PD12	D29	21	22	GND	GND
PD11	D30	23	24	D40	PE10
PE2	D31	25	26	D39	PE12
GND	GND	27	28	D38	PE14
PA0	D32	29	30	D37	PE15
PB0	D33	31	32	D36	PB10
PE0	D34	33	34	D35	PB11

CN10

PA5 – DAC_OUT2

PA3 (A0) – ADC1_IN3, ADC2_IN3, ADC3_IN3
PC0 (A1) – ADC1_IN10, ADC2_IN10, ADC3_IN10
PC3 (A2) – ADC1_IN13, ADC2_IN13, ADC3_IN13
PF3 (A3) – ADC3_IN9
PF5 (A4) – ADC3_IN15
PF10 (A5) – ADC3_IN8

PB1 (A6) – ADC1_IN9, ADC2_IN9
PC2 (A7) – ADC1_IN12, ADC2_IN12, ADC3_IN12
PF4 (A8) – ADC3_IN14

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Add Code to **main.c**

```
/* Private includes */
/* USER CODE BEGIN Includes */
#include "lwip/apps/httpd.h"
#include <string.h>
#include <stdlib.h>
/* USER CODE END Includes */

/* Private typedef */
/* USER CODE BEGIN PTD */
// array of tags for the SSI handler
// these are the tags <!--#tag1--> contained in the shtml file
// one tag <!--#t-->
#define numSSItags   1
/* USER CODE END PTD */

/* Private variables */
/* USER CODE BEGIN PV */
uint32_t adc[2];
uint32_t value_dac = 0;

char const *theSSItags[numSSItags] = {"t"};

// prototype CGI handler for the LED control
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[]);

// this structure contains the name of the LED CGI and corresponding handler for the LEDs
const tCGI LedCGI={"/leds.cgi", LedCGIhandler};

// table of the CGI names and handlers
tCGI theCGItable[1];
/* USER CODE END PV */
```

Purpose and Test procedure:

UM1713 User manual

Developing applications on STM32Cube

with LwIP TCP/IP stack

Section 6.2 Features Applications

6.2.1 Web Server Based on Raw API

CGI = Common Gateway Interface
SSI = Server Side Includes

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Add Code to **main.c**

```c
/* Private user code */
/* USER CODE BEGIN 0 */
// Initialize the CGI handlers
void myCGIinit(void)
{
    // add LED control CGI to the table
    theCGItable[0] = LedCGI;

    // give the table to the HTTP server
    http_set_cgi_handlers(theCGItable, 1);
} // myCGIinit

/**** CGI handler for controlling the LEDs ****/
// the function pointer for a CGI script handler is defined in httpd.h as tCGIHandler
const char * LedCGIhandler(int iIndex, int iNumParams, char *pcParam[], char *pcValue[])
{
    uint32_t i = 0;
    // index of the CGI within the theCGItable array passed to http_set_cgi_handlers
    // Given how this example is structured, this may be a redundant check.
    // Here there is only one handler iIndex == 0
    if (iIndex == 0)
    {
        // turn off the LEDs
        HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_RESET);
```

Add Code to **main.c**

```
    // Check the cgi parameters, e.g., GET /leds.cgi?led=1&led=2&led=3
    for (i=0; i<iNumParams; i++)
    {
      // if pcParmeter contains "led", then one of the LED check boxes has been set on
      if (strcmp(pcParam[i], "led") == 0)
      {
        // see if checkbox for LED 1 has been set
        if(strcmp(pcValue[i], "1") == 0)
        {
          // switch led 1 ON if 1
          HAL_GPIO_WritePin(LD1_GPIO_Port, LD1_Pin, GPIO_PIN_SET);
        }
        // see if checkbox for LED 2 has been set
        else if(strcmp(pcValue[i], "2") == 0)
        {
          // switch led 2 ON if 2
          HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_SET);
        }
        // see if checkbox for LED 3 has been set
        else if(strcmp(pcValue[i], "3") == 0)
        {
          // switch led 3 ON if 3
          HAL_GPIO_WritePin(LD3_GPIO_Port, LD3_Pin, GPIO_PIN_SET);
        }
      } // if
    } // for
  } // if
  // uniform resource identifier to send after CGI call, i.e., path and filename of the response
  return "/STM32F767LED.html";
} // LedCGIhandler
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Add Code to **main.c**

```
/**** SSI handler ****/
 // This function is called each time the HTTPD server detects a tag of the form
 // <!--#name--> in a .shtml, .ssi or .shtm file
 // It won't work if the file has a .html extension.
 u16_t mySSIHandler(int iIndex, char *pcInsert, int iInsertLen)
 {
     /* We have only one SSI handler iIndex = 0 */
     if (iIndex == 0)
     {
         char Digit1=0, Digit2=0, Digit3=0, Digit4=0;
         uint32_t ADCVal = 0;

         HAL_ADC_Start(&hadc1);

         HAL_ADC_PollForConversion(&hadc1,100);
         adc[0] = HAL_ADC_GetValue(&hadc1);

         HAL_ADC_PollForConversion(&hadc1,100);
         adc[1] = HAL_ADC_GetValue(&hadc1);

         HAL_ADC_Stop(&hadc1); // stop ADC is important

         /* convert to voltage,  12 bits, step = 3.3V / 4096 = 0.8056640625 mV */
         ADCVal = (uint32_t)(adc[0] * 0.8056640625);

         /* get digits to display */
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Add Code to **main.c**

```
         /* get digits to display */
         Digit1 =  ADCVal / 1000;
         Digit2 = (ADCVal-(Digit1*1000)) / 100;
         Digit3 = (ADCVal-((Digit1*1000)+(Digit2*100))) / 10;
         Digit4 =  ADCVal-((Digit1*1000)+(Digit2*100)+(Digit3*10));

         /* prepare data to be inserted in html */
          *pcInsert      = (char)(Digit1+0x30); // ascii 0 = 48 (dec) or 0x30
         *(pcInsert + 1) = (char)(Digit2+0x30);
         *(pcInsert + 2) = (char)(Digit3+0x30);
         *(pcInsert + 3) = (char)(Digit4+0x30);

         /* 4 characters need to be inserted in html*/
         return 4;
     }
     return 0;
 } // mySSIHandler

 /**** Initialize SSI handlers ****/
  void mySSIinit(void)
  {
      // configure SSI handler function
      // theSSItags is an array of SSI tag strings to search for in SSI-enabled files
      http_set_ssi_handler(mySSIHandler, (char const **)theSSItags, numSSItags);
  } // mySSIinit
/* USER CODE END 0 */
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Add Code to **main.c**

```
/* @brief  The application entry point */
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_LWIP_Init();
    MX_ADC1_Init();
    MX_DAC_Init();
    /* USER CODE BEGIN 2 */
    // start DAC and output value
    HAL_DAC_Start(&hdac, DAC_CHANNEL_1);
    HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, value_dac);

    // start the web server
    httpd_init();

    // initialise the CGI handlers
    myCGIinit();

    // initialize the SSI handlers
    mySSIinit();
  /* USER CODE END 2 */

  /* Infinite loop */
```

Add Code to **main.c**

```
 /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
       MX_LWIP_Process();

       // HAL_GPIO_TogglePin(GPIOB, LD3_Pin);

       HAL_DAC_SetValue(&hdac, DAC_CHANNEL_1, DAC_ALIGN_12B_R, value_dac);

       value_dac = value_dac + 1;

       if(value_dac > 4095)
       {
           value_dac = 0;
       }

       printf("DAC: %ld  ADC: %ld  %ld  \n\r", value_dac, adc[0], adc[1]);

       HAL_Delay(100);
    /* USER CODE END WHILE */

    /* USER CODE BEGIN 3 */
  }
  /* USER CODE END 3 */
}
```

Add Code to **main.c**

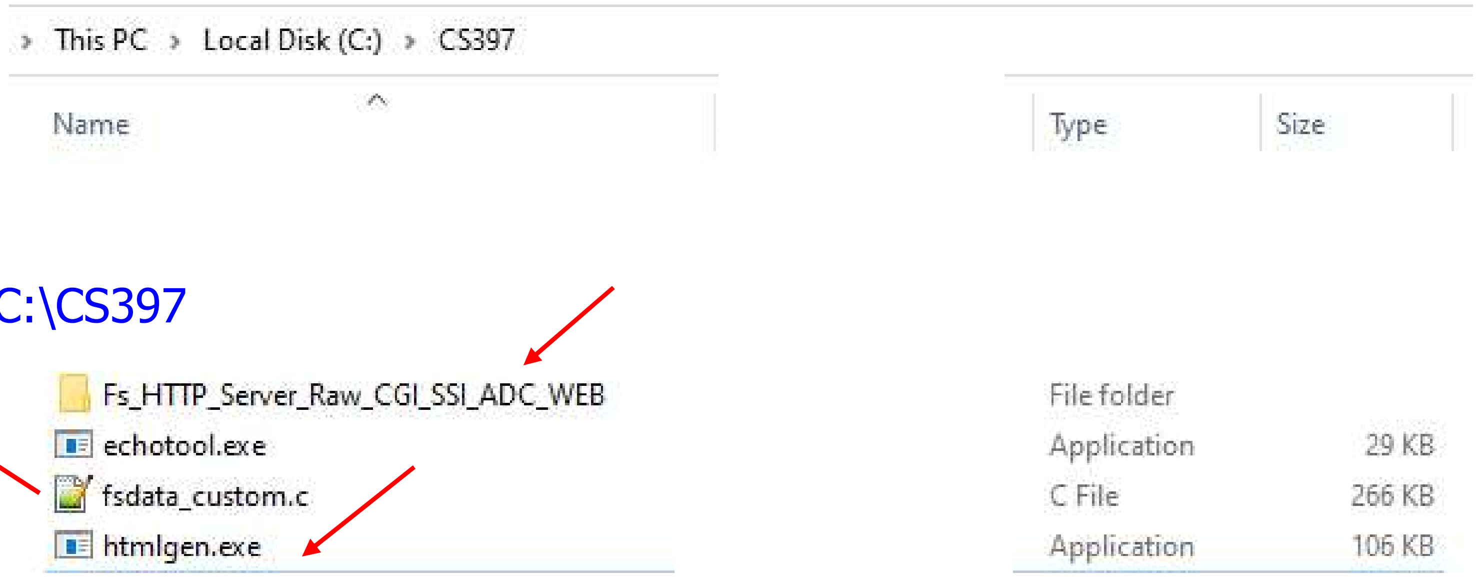
```
/* USER CODE BEGIN 4 */
void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
    if(GPIO_Pin == GPIO_PIN_13)
    {
        HAL_GPIO_TogglePin(GPIOB, LD2_Pin);
    }
}

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 */
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Generate the `fsdata_custom.c`

1 **Unzip 12_CS397_Hands-On_5-2_LwIP_HTTP_Server_Raw_CGI_SSI.zip**

2 Copy folder “Fs_HTTP_Server_Raw_CGI_SSI_ADC_WEB” to c:\CS397



› This PC › Local Disk (C:) › CS397

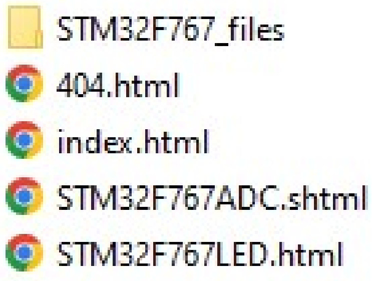
C:\CS397

Name	Type	Size
Fs_HTTP_Server_Raw_CGI_SSI_ADC_WEB	File folder	
echotool.exe	Application	29 KB
fsdata_custom.c	C File	266 KB
htmlgen.exe	Application	106 KB

3 Run

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

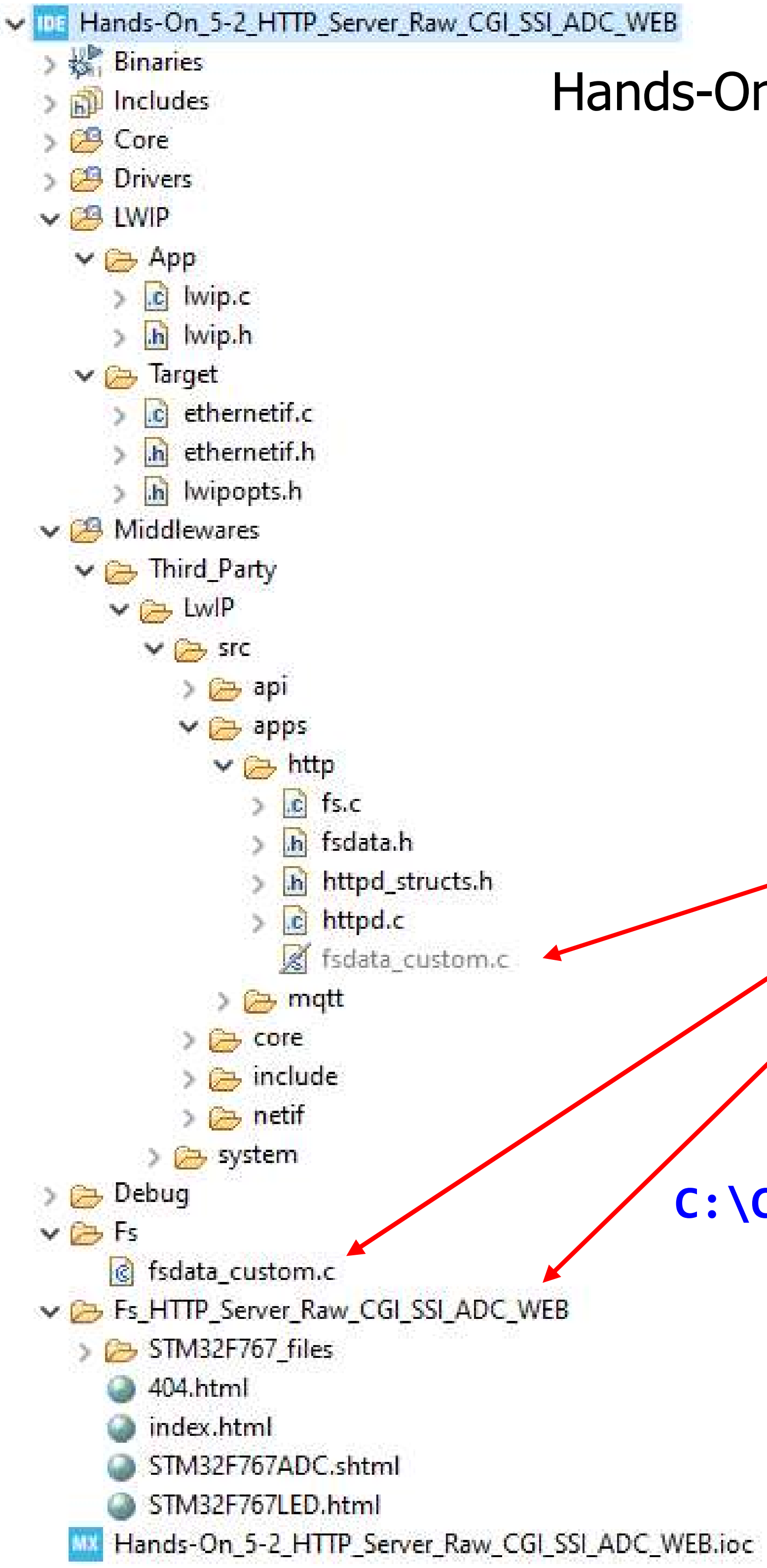
`C:\CS397\Fs_HTTP_Server_Raw_CGI_SSI_ADC_WEB`



STM32F767_files
404.html
index.html
STM32F767ADC.shtml
STM32F767LED.html

4 Copy folder and generated file “fsdata_custom.c” to STM32 project

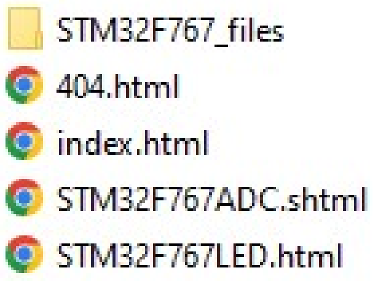
Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB



```
Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC_WEB
  Binaries
  Includes
  Core
  Drivers
  LWIP
    App
      lwip.c
      lwip.h
    Target
      ethernetif.c
      ethernetif.h
      lwipopts.h
  Middlewares
    Third_Party
      LwIP
        src
          api
          apps
            http
              fs.c
              fsdata.h
              httpd_structs.h
              httpd.c
              fsdata_custom.c
            mqtt
          core
          include
          netif
        system
  Debug
  Fs
    fsdata_custom.c
  Fs_HTTP_Server_Raw_CGI_SSI_ADC_WEB
    STM32F767_files
    404.html
    index.html
    STM32F767ADC.shtml
    STM32F767LED.html
  Hands-On_5-2_HTTP_Server_Raw_CGI_SSI_ADC_WEB.ioc
```

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

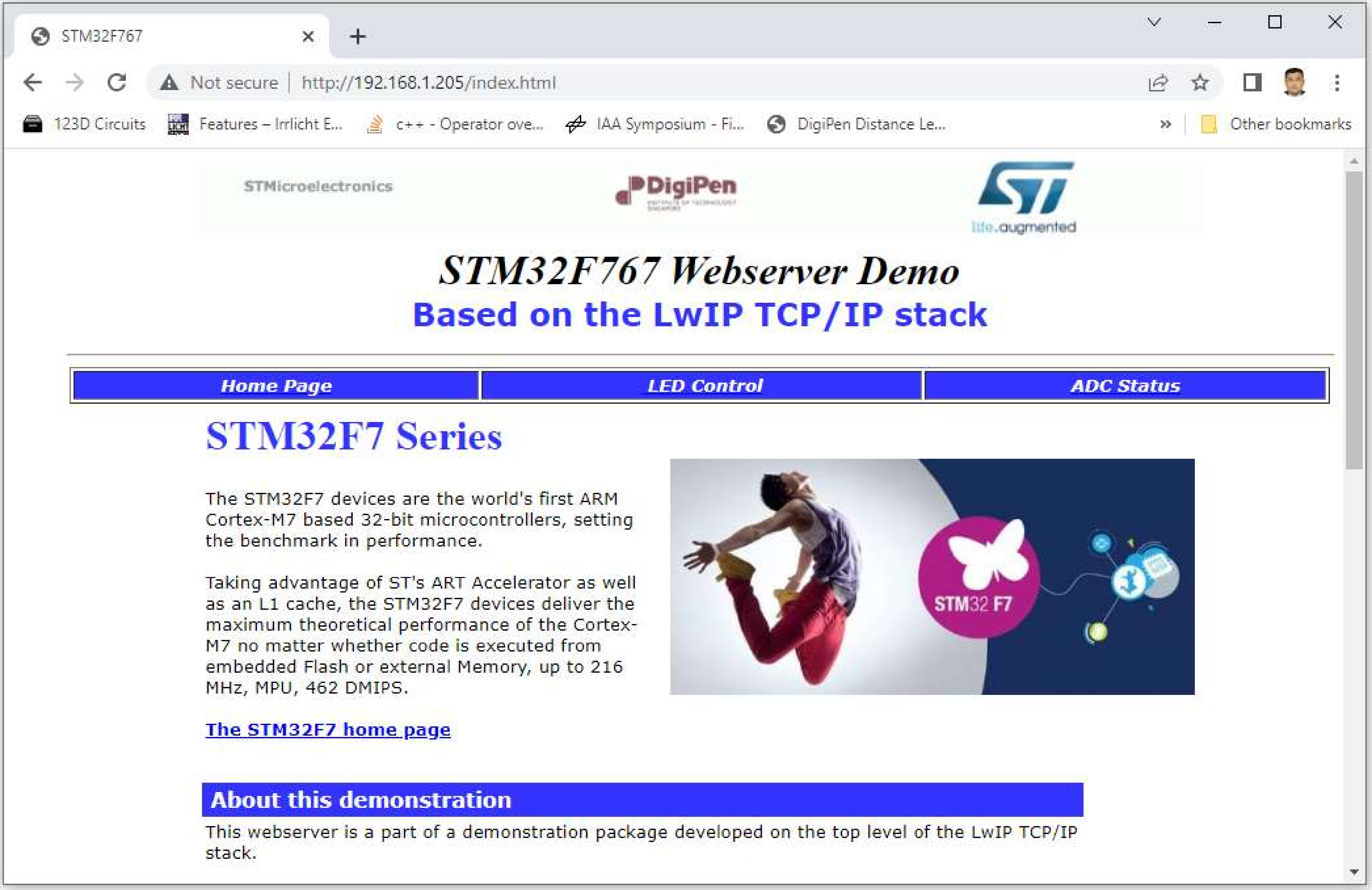
C:\CS397\Fs_HTTP_Server_Raw_CGI_SSI_ADC_WEB



```
STM32F767_files
404.html
index.html
STM32F767ADC.shtml
STM32F767LED.html
```

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB

Access the web page by typing http://192.168.1.205 onto a web browser



STM32F767 Webserver Demo

Based on the LwIP TCP/IP stack

Home Page | LED Control | ADC Status

STM32F7 Series

The STM32F7 devices are the world's first ARM Cortex-M7 based 32-bit microcontrollers, setting the benchmark in performance.

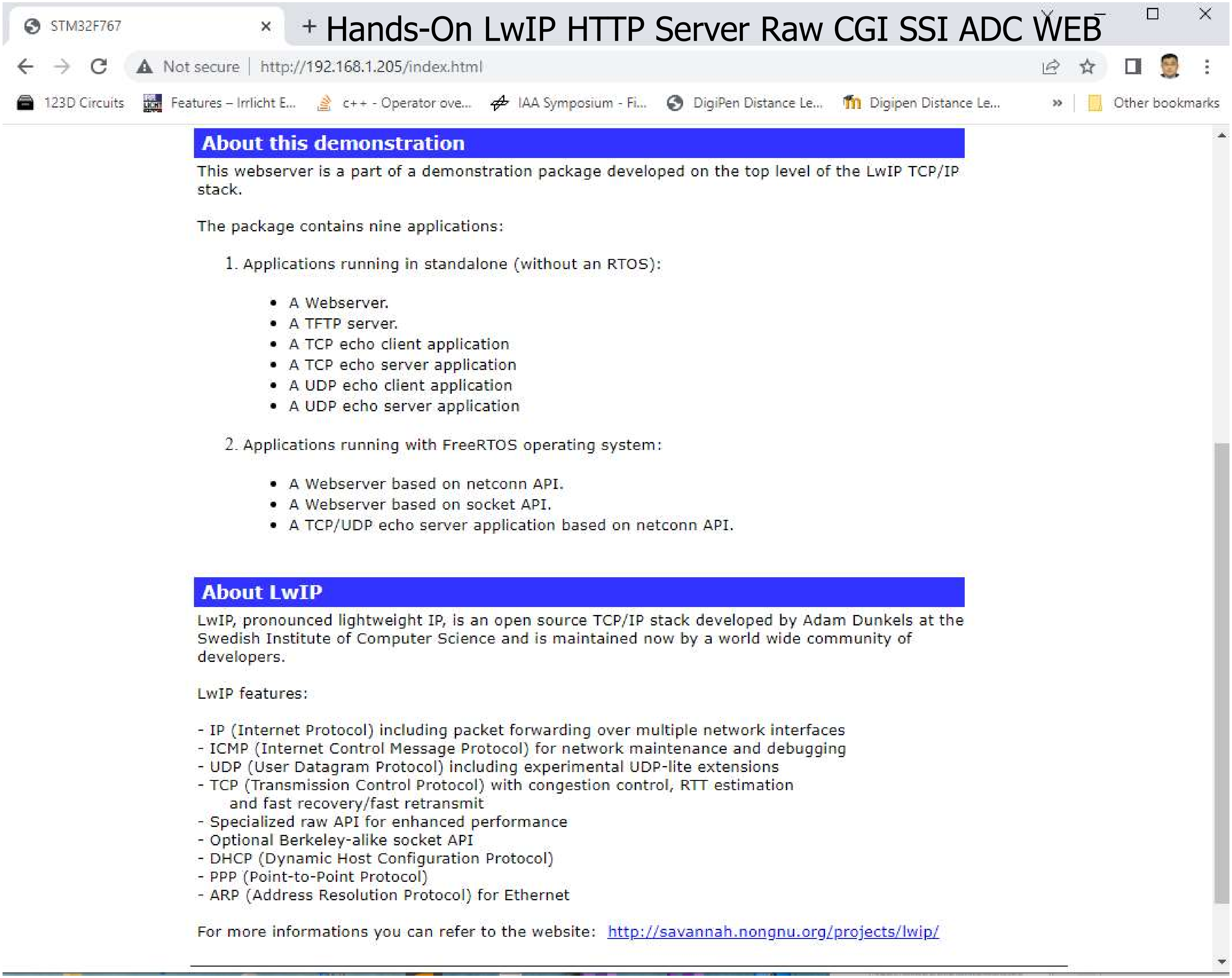
Taking advantage of ST's ART Accelerator as well as an L1 cache, the STM32F7 devices deliver the maximum theoretical performance of the Cortex-M7 no matter whether code is executed from embedded Flash or external Memory, up to 216 MHz, MPU, 462 DMIPS.

The STM32F7 home page

About this demonstration

This webserver is a part of a demonstration package developed on the top level of the LwIP TCP/IP stack.

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB



About this demonstration

This webserver is a part of a demonstration package developed on the top level of the LwIP TCP/IP stack.

The package contains nine applications:

1. Applications running in standalone (without an RTOS):
   - A Webserver.
   - A TFTP server.
   - A TCP echo client application
   - A TCP echo server application
   - A UDP echo client application
   - A UDP echo server application
2. Applications running with FreeRTOS operating system:
   - A Webserver based on netconn API.
   - A Webserver based on socket API.
   - A TCP/UDP echo server application based on netconn API.

About LwIP

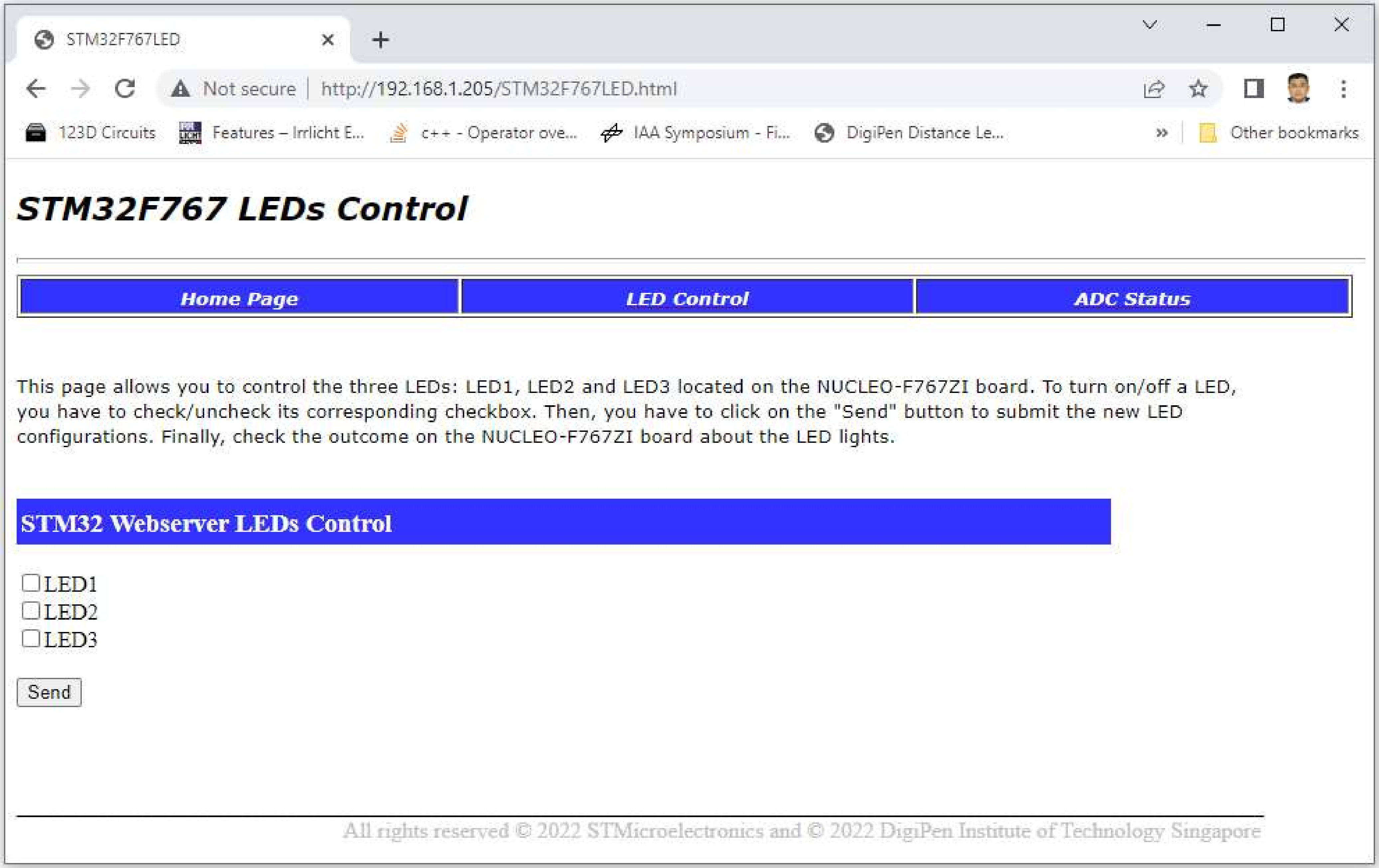
LwIP, pronounced lightweight IP, is an open source TCP/IP stack developed by Adam Dunkels at the Swedish Institute of Computer Science and is maintained now by a world wide community of developers.

LwIP features:

- IP (Internet Protocol) including packet forwarding over multiple network interfaces
- ICMP (Internet Control Message Protocol) for network maintenance and debugging
- UDP (User Datagram Protocol) including experimental UDP-lite extensions
- TCP (Transmission Control Protocol) with congestion control, RTT estimation
  and fast recovery/fast retransmit
- Specialized raw API for enhanced performance
- Optional Berkeley-alike socket API
- DHCP (Dynamic Host Configuration Protocol)
- PPP (Point-to-Point Protocol)
- ARP (Address Resolution Protocol) for Ethernet

For more informations you can refer to the website: http://savannah.nongnu.org/projects/lwip/

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB



STM32F767 LEDs Control

Home Page	LED Control	ADC Status

This page allows you to control the three LEDs: LED1, LED2 and LED3 located on the NUCLEO-F767ZI board. To turn on/off a LED, you have to check/uncheck its corresponding checkbox. Then, you have to click on the "Send" button to submit the new LED configurations. Finally, check the outcome on the NUCLEO-F767ZI board about the LED lights.

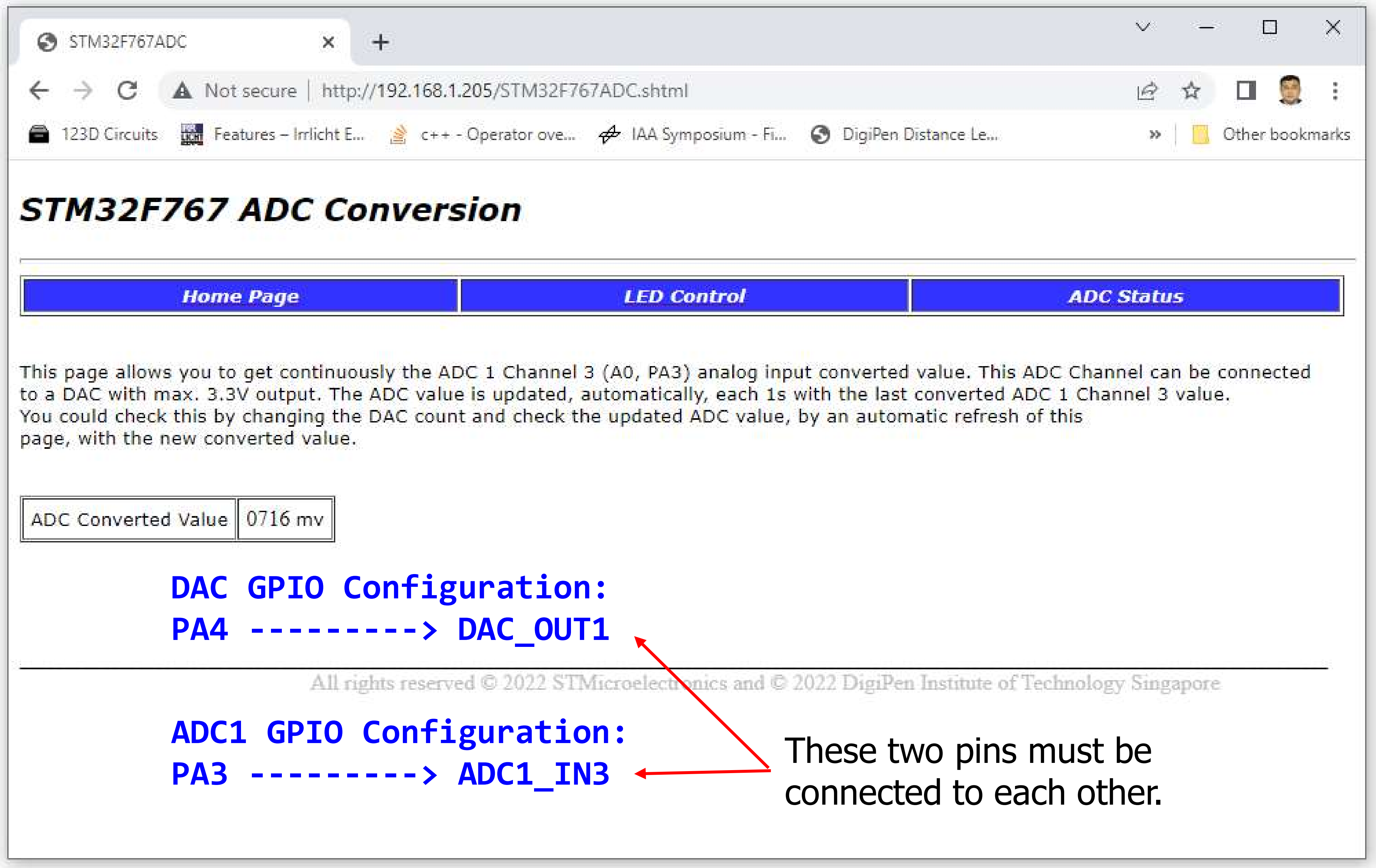
STM32 Webserver LEDs Control

☐ LED1
☐ LED2
☐ LED3

Send

All rights reserved © 2022 STMicroelectronics and © 2022 DigiPen Institute of Technology Singapore

Hands-On LwIP HTTP Server Raw CGI SSI ADC WEB



STM32F767 ADC Conversion

Home Page	LED Control	ADC Status

This page allows you to get continuously the ADC 1 Channel 3 (A0, PA3) analog input converted value. This ADC Channel can be connected to a DAC with max. 3.3V output. The ADC value is updated, automatically, each 1s with the last converted ADC 1 Channel 3 value.
You could check this by changing the DAC count and check the updated ADC value, by an automatic refresh of this page, with the new converted value.

ADC Converted Value	0716 mv

DAC GPIO Configuration:
PA4 ---------> DAC_OUT1

All rights reserved © 2022 STMicroelectronics and © 2022 DigiPen Institute of Technology Singapore

ADC1 GPIO Configuration:
PA3 ---------> ADC1_IN3

These two pins must be connected to each other.

Remark: Common Gateway Interface (CGI)

- The **CGI** is a standard web technique used to execute a request coming from a client to the server and return a response to the client.
- In **LwIP,** the **CGI** offered works only with **GET method** requests and can handle up to **16 parameters** encoded in a URI.
- The CGI handler function executed on the server side returns a HTML file that the HTTP server sends to the clients.

- End -