uEZ® HTTP Server Quick Start Guide

1. Introduction

This guide covers the HTTP Server in uEZ 2.05 or later. The uEZ HTTP Server is part library and part application level software that runs a standard HTTP web server on a uEZGUI unit. This HTTP Server can serve basic HTML web pages, javascript, graphics, file downloads, and real-time generated information.
Below is a diagram of the HTTP Server.

User Application Software	Application software does background tasks and updates website information (such as sensor info, timestamps, etc) at the top level
Application Level	
Config_Build.h	HTTP server and wired/wireless network turned on in this file
main.c	Starts HTTP Server task
Config_App.h	Configures factory default IP address, MAC address, default gateway, etc
AppHTTPServer.c/.h	Configuration variables for submitting/retrieving information from dynamic web pages, such as dynamic timestamps
uEZ Library Level	
uEZ lwIP v4	Part of uEZ library that handles IPV4 network communications
uEZ TCP	Subset of uEZ lwIP, handles TCP Protocols
HTTPServer.c/.h	Library files for HTTP Server that handle the server activities

The HTTP Servers includes an example demo to demonstrate its various features.

2. Turning the HTTP Server on in the uEZGUI demo project

To turn on the HTTP Server, make sure that the TCPIP stack and either wired or wireless network is turned on in Config_Build.h

```
#define UEZ_ENABLE_TCPIP_STACK          1
#define UEZ_ENABLE_WIRED_NETWORK        1
```

Make sure that the basic web server is turned off, and that the HTTP server is turned on.

```
// Choose one when TCP/IP stack is enabled
#define UEZ_BASIC_WEB_SERVER            0
#define UEZ_HTTP_SERVER                 1
#define DEBUG_HTTP_SERVER               0  // when enabled prints information to console.
```

In “Source/App/uEZDemoCommon/AppHTTPServer.h” there are additional settings for the server port number and which drive to host files from. The default settings are listed below but they can be overridden inside of Config_Build.h.

```
#define HTTP_SERVER_HTTP_PORT           80 // TCP Port to use for HTTP Server
#define HTTP_SERVER_DRIVE               "1:/HTTPROOT" // "1" for SD card, "0" for USB drive
```

Make sure that you select the correct drive for the web server. Right now either the SD card or a flash drive is supported.

3. Setting up the HTTP Server files on a SD card or flash drive

To set up the HTTP Server, place files in this SD card/flash drive folder: (This folder will be the root of the web server directory)

“SDCARDROOT/HTTPROOT”

This HTTPROOT directory can be changed inside of Config_Build.h.
The index file should be at HTTPROOT/index.htm. This can be changed in the code if needed.
This is the index file looked for when the GET request doesn’t specify a file.
A favicon.ico file can be placed inside the same HTTP root folder if desired.

After configuring the settings as needed and adding files to the SD card or USB flash drive, you can then access web pages using the UEZGUI’s IP Address from a web browser.

4. Using the HTTP Server with the example demo

To use the HTTP Server, please refer to the example demo files located on the SD card. These files are also located on SourceForge under the demo SD card files for uEZ 2.05 or above.
The example demo consists of the following files:

uEZ® HTTP Server Quick Start Guide

```
INDEX.HTM                   // The main webpage index.
favicon.ico                 // 32x32 webpage icon, created using GIMP 2.8
IMAGES/uEZ.png              // uEZ logo image file inside of a folder
```

To view the demo on the uEZGUI, setup a network connection on the uEZGUI by setting the IP address, netmask, and gateway using the following commands over the uEZ command line.

```
# ipaddr 192.168.40.50
PASS: 192.168.40.50

# ipgateway 192.168.40.1
PASS: 192.168.40.1

# ipmask 255.255.255.0
PASS: 255.255.255.0
```

By typing in just the base command name, you can verify what setting is used on the uEZGUI.

```
# ipaddr
PASS: 192.168.40.50

# ipgateway
PASS: 192.168.40.1

# ipmask
PASS: 255.255.255.0
```

After verifying the settings, make sure that your PC/Cell phone, etc is on the same sub-network as the uEZGUI.
After rebooting the uEZGUI you will be able to ping the uEZGUI and access the webpage by typing in it's IP address.
The example webpage will look like the picture shown below.

uEZ® HTTP Server Quick Start Guide

uEZ® HTTP Server Quick Start Guide

The first part of the example shows an image displayed on the webpage. This example uses standard HTML to display the image.

```
<img border="0" src="IMAGES/uEZ.PNG" >
```

As the webpage explains, the date and time will be continuously updated on the webpage. If the date or time is changed on the uEZGUI unit, the webpage will refresh to show the updated information.

```
<DIV style="height: 320px; border: 1px solid white; text-align: center">
 <DIV style="margin: 16px">Time: ${time}<BR>Date: ${date}</DIV>
 <DIV class="button"><a href="javascript:document.location.reload();">Update</a></DIV>
</DIV>
```

This part of the example uses the 2 variable names “${time}” and “${date}” in the HTM file. A JavaScript call can reload the webpage automatically without user input.
In “Source/App/uEZDemoCommon/AppHTTPServer.c” is the function shown below:

```
/*---------------------------------------------------------------------------*
 * Routine:  IMainHTTPServerGetValue
 *---------------------------------------------------------------------------*
 * Description:
 *     Retrieves values for dynamic webpages. This example pulls the current
 *     time and date information using the uEZTImeDate API.
 * Outputs:
 *     T_uezError               -- error
 *---------------------------------------------------------------------------*/
static T_uezError IMainHTTPServerGetValue(void *aHTTPState, const char *aVarName){
   T_uezTimeDate TimeDate;
   char line[16];
   if (UEZTimeDateGet(&TimeDate) == UEZ_ERROR_NONE) {
      // success on retriving time from RTC
      // current time value stored in TimeDate
      if (strcmp(aVarName, "time") == 0) {
       if(TimeDate.iTime.iHour == 0){
         sprintf(line, "12:%02u:%02u AM\r\n",
            TimeDate.iTime.iMinute, TimeDate.iTime.iSecond);
       }else if (TimeDate.iTime.iHour < 12){
         sprintf(line, "%02u:%02u:%02u AM\r\n", TimeDate.iTime.iHour,
            TimeDate.iTime.iMinute, TimeDate.iTime.iSecond);
       }else if (TimeDate.iTime.iHour == 12){
         sprintf(line, "%02u:%02u:%02u PM\r\n", TimeDate.iTime.iHour,
            TimeDate.iTime.iMinute, TimeDate.iTime.iSecond);
       }else{
         sprintf(line, "%02u:%02u:%02u PM\r\n", (TimeDate.iTime.iHour-12),
            TimeDate.iTime.iMinute, TimeDate.iTime.iSecond);
       }
         HTTPServerSetVar(aHTTPState, aVarName, line);
      } else if (strcmp(aVarName, "date") == 0) {
         sprintf(line, "%02u/%02u/%04u\r\n", TimeDate.iDate.iMonth,
            TimeDate.iDate.iDay, TimeDate.iDate.iYear);
         HTTPServerSetVar(aHTTPState, aVarName, line);
      } else {
         HTTPServerSetVar(aHTTPState, aVarName, "InsertVariableHere");
      }
   } else { // timedate failure
      HTTPServerSetVar(aHTTPState, aVarName, "InsertVariableHere");
   }
   return UEZ_ERROR_NONE;
}
```

This function replaces the variable names with real-time information from the application using the HTTPServerSetVar function call.

The second part of the example will send a frequency number and time duration back to the uEZGUI. The uEZGUI unit will then play the specified tone for the specified number of seconds. See the following HTML:

```
<form name="input" METHOD="POST">
<table border="1" cellpadding="0" cellspacing="0" style="border-collapse: collapse" bordercolor="#FFFFFF" width="35%">
 <tr>
  <td width="91%"><font face="Verdana" size="2">Tone Frequency (Hz)</font></td>
  <td width="26%"><font face="Verdana"><INPUT NAME="freq" SIZE="4" MAXLENGTH="4" VALUE="1000"></font></td>
 </tr>
 <tr>
  <td width="91%"><font face="Verdana" size="2">Duration (seconds)</font></td>
  <td width="26%"><font face="Verdana"><INPUT NAME="duration" SIZE="4" MAXLENGTH="4" VALUE="2"></font></td>
 </tr>
</table>
<p><font face="Verdana">
<INPUT TYPE="submit" VALUE="Press for tone"> </font></p>
</FORM>
```

This part of the example uses the 2 variable names “freq” and “duration” in the HTM file.

In “Source/App/uEZDemoCommon/AppHTTPServer.c” is the function shown below:

```
/*---------------------------------------------------------------------------*
 * Routine:  IMainHTTPServerSetValue
 *---------------------------------------------------------------------------*
 * Description:
 *     Retrieves values from dynamic webpages.
 * Outputs:
 *     T_uezError               -- error
 *---------------------------------------------------------------------------*/
static T_uezError IMainHTTPServerSetValue(void *aHTTPState, const char *aVarName, const char *aValue){
   static TUInt32 frequency = 1000;
   if (strcmp(aVarName, "freq") == 0) {    // Remember the last know frequency
      frequency = atoi(aValue);
   } else if (strcmp(aVarName, "duration") == 0) {
      PlayAudio(frequency, atoi(aValue) * 1000);
   }
   return UEZ_ERROR_NONE;
}
```

This function plays audio on the uEZGUI from the values submitted from the HTML webpage form.

The final part of the example is the favicon.ico. After creating a suitable icon, just name it favicon.ico and place it in the HTTPROOT folder. The web browser will automatically load it.

5. Configuring MIME Types

The HTTP Server library file is located in uEZ\Source\Library\Web\HTTPServer\HTTPServer.c
Mime types are set in the function “IDetermineMimeTypeFromFilename” inside of the HTTPServer.c file.
Here is the current list of mime types.

```
const T_mimeType G_mimeTypes[] = {
      { "htm", "text/html" },
      { "html", "text/html" },
      { "jpg", "image/jpeg" },
      { "jpeg", "image/jpeg" },
      { "gif", "image/gif" },
      { "png", "image/png" },
      { "js", "text/javascript" },
      { "txt", "text/plain" },
      { "css", "text/stylesheet" },
      { "zip", "application/zip" },
```

```
    { "pdf", "application/pdf" },
    { "swf", "application/x-shockwave-flash" },
    { "mp3", "audio/mpeg" },
    { "ico", "image/vnd.microsoft.icon" },
    { 0, "application/octet-stream" }, // default, always at end
};
```

Additional MIME types can be added to this list. The uEZ library will need to be rebuilt.

6. Advanced Settings

At the beginning of the uEZ\Source\Library\Web\HTTPServer\HTTPServer.c file, the port, max page size, 404 message, and other misc settings can be set by modifying the #defines. Here are the default settings.

```
/* The size of the buffer in which the dynamic WEB page is created. */
#define MAX_PAGE_SIZE         1024

/* Standard GET response. */
#define HTTP_OK               "HTTP/1.0 200 OK\r\nContent-type: text/html\r\n\r\n"

// uEZ 404 not found error message
#define  HTTP_MSG_NOT_FOUND \
    "<HTML>\r\n" \
    "<BODY>\r\n" \
    "<HEAD><TITLE>uEZ File not found</TITLE></HEAD>\r\n" \
    "<H1>uEZ File not found</H1>\r\n" \
    "The requested object does not exist on this server.\r\n" \
    "</BODY>\r\n" \
    "</HTML>\r\n"

// Stack size of the HTTP Server
#define WEB_SERVER_STACK_SIZE  UEZ_TASK_STACK_BYTES(2048)

// Displayed version number of the HTTP Server
#define WEB_SERVER_REPORT_VERSION       0

#define MAX_LINE_LENGTH                 512  // Max length of message to send/receive per packet
#define MAX_HTTP_VERSION                30   // Supported version of HTTP advertised by the server
#define MAX_HTTP_CONTENT_TYPE           120  // Type of content supplied from the HTTP Server
#define MAX_HTTP_CONTENT_BOUNDARY 80    // Content boundary of the HTTP Server
#define MAX_VAR_NAME_LENGTH             20   // Maximum length of variable name
#define HTTP_WRITE_BUFFER_SIZE          512  // Max length of write buffer size
```

If any of these are changed, the uEZ library will need to be rebuilt.