


# Embedded Web Server Using LWIP Stack


simpleisrobust@gmail.com

## 1. Features

Objective: In a Windows environment, create a virtual device using pcap and lwip, and implement a simple but typical web server with the following features:

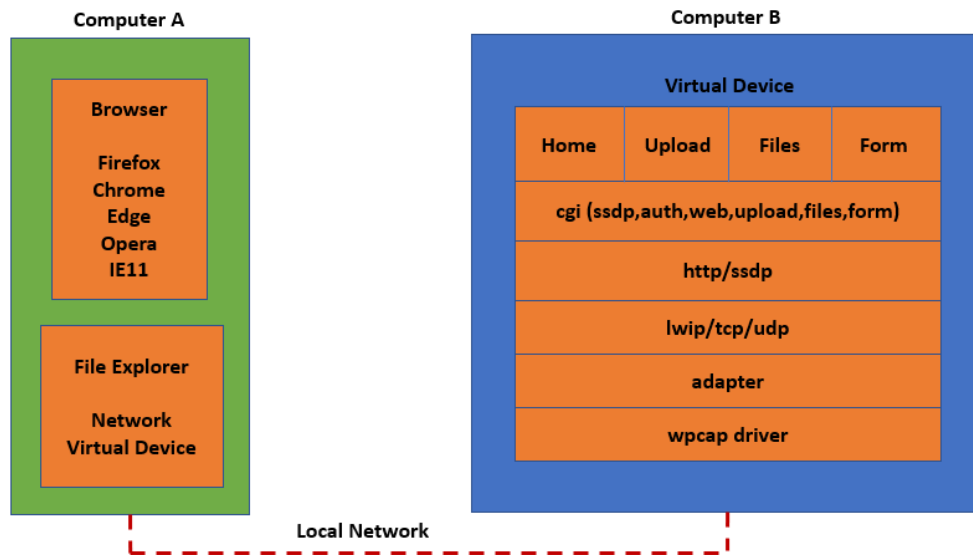
- Support SSDP protocol: neighbours can detect this virtual device in the network.
- Support HTTPS: all HTTP requests will be redirected to HTTPS (except SSDP XML). Both GET and POST are supported.
- Support chunked response: Because the device memory is limited, generally it is not possible to wait for all the data to be ready before replying. When the device begins to return data, it does not know the total length. "**Transfer-Encoding: chunked**" allows you to split the data into chunks of various lengths, so that every chunk is within the device's memory limit.
- Support range request: "**Range: bytes=0-**" allows you to specify a single range. This feature is suitable for paged data, multi-threaded downloading or video fast forwarding.
- Support request header "**If-Modified-Since**" for browsers to cache static files.
- Support simple SSI ("Server Side Includes") with simple syntax: `<!--#TAG_NAME-->`. Note that the syntax does not allow spaces between the "<" and ">". Every tag is placeholder, which corresponds to a variable. The variable can be html content or the initial value of a javascript variable. When the device replies to a request, it will first convert the variable into a string. This string then replaces the corresponding tag. Based on the tag content, the js code can then complete html/DOM operations, such as select, checkbox, and radio control etc.
- Demo pages include:
  - Sign in/out: verify user's username/password, if successful, redirect to "Home", otherwise stay in the login page, the session has 3 minutes timeout after login.
  - Home page: show some device information and the current status of the device.
  - Upload page: multiple files can be selected and queued to be uploaded to a specific device directory. The concurrency is 1, and every upload process can be terminated separately.
  - Files page: you can browse all the files in the specified device directory, enter subdirectories or return to the parent directory.
  - Form page: demonstrates parameter modification and saving.

 Username

 Password

Login

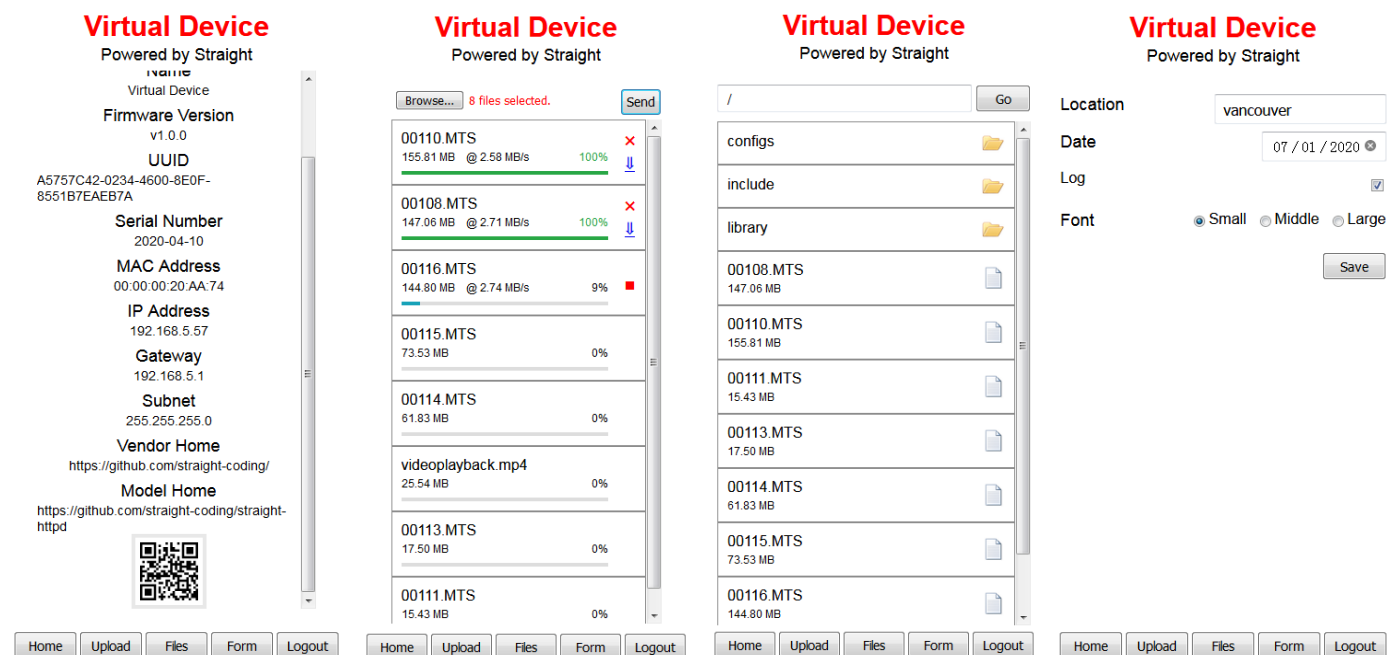
## 2. Diagram



This project creates a virtual device on Computer B. Since the browser on B cannot access this local Virtual Device, another Computer A should be used to access this device remotely. Computer A could be a Windows, Linux, or a mobile device, and Computer B MUST be a Windows device with VS2017 or later installed.

**Computer B:** Use VS2017 to open the project “straight-httpd/straight-httpd.sln” and run it in debug.

**Computer A:** for Windows, open File Explorer and search for network neighbors. There should be a device named “Virtual Device”. Right click on it and select “Properties”, and you will see the device information, including its IP address and URL. Use default user “admin” and password “password” to sign into the device home page.



## 3. Prerequisites

Computer A: only need a browser installed.

Computer B: need to install wpcap driver, or simply install Wireshark that includes wpcap driver.

## 4. Project Directory

Sub folders:

**wpcap**: wpcap C library.

**lwip-port**: porting code for lwip.

Receive ethernet frame from wpcap and send it to lwip stack.

Send ethernet frame from lwip stack via wpcap.

Also provide features: Mutex, Semaphore, Mailbox, System Tick, and File API.

**lwip**: copied from lwip official site with some modifications.

**mbedtls**: copied from mbedtls official git site. The file config.h has been modified.

**straight-httpd**: VS2017 solution file **straight-httpd.sln**, and the project **straight-buildfs** can package all web pages and convert them into source code.

## 5. GET handling

Device HTTP Events	CGI Adapter - Actions	
GET /app/index.shtml HTTP/1.1	CGI_SetCgiHandler(context)	First request line
Connection: keep-alive	http_core.c	Request headers
Cookie: SID=0123ABCD	http_core.c	
Range: bytes=0-	http_core.c	
If-Modified-Since:	http_core.c	
X-Auth-Token: SID=0123ABCD	http_core.c	
other header	CGI_HeaderReceived(context,line)	
all headers received	Check session: GetSession CGI_HeadersReceived(context)	End of the request headers
request received	CGI_RequestReceived(context)	
send response headers	CGI_SetResponseHeaders(context,codeInfo)	Response headers
response with content	CGI_LoadContentToSend(context, caller)	Response body
Response completely sent out	OnAllSent(context)	
When connection disconnected or -500	CGI_Finish(context)	
lwip error, request error, timeout, <=-400	CGI_Cancel(context)	

## 6. POST handling

Device HTTP Events	CGI Adapter - Actions	
POST /auth/login.html HTTP/1.1	CGI_SetCgiHandler(context)	First request line
Content-Type: application/x-form-urlencoded	http_core.c	Request headers
Content-Length: 24	http_core.c	
Connection: keep-alive	http_core.c	
other header	CGI_HeaderReceived(context,line)	
all headers received	Check session: GetSession CGI_HeadersReceived(context)	End of the request headers
request body	CGI_ContentReceived(context, buffer, size)	Request body
post body received	CGI_RequestReceived(context)	
send response headers	CGI_SetResponseHeaders(context,codeInfo)	Response headers
response with content	CGI_LoadContentToSend(context, caller)	Response body
Response completely sent out	OnAllSent(context)	
When connection disconnected or -500	CGI_Finish(context)	
lwip error, request error, timeout, <=-400	CGI_Cancel(context)	

## 7. Self-Signed Certificate for HTTPS

- If you want to create your customized certificate, run "makecert.bat". Both the certificate and the private key will be saved in the file "server.pfx".
- Use OpenSSL or other tools to convert PFX file to PEM format.

```
openssl pkcs12 -in server.pfx -out server.pem
```

- Copy the following content in the pem file to "http\_api.c"

```

const char *privkey_pass = "straight";
const char *privkey = "-----BEGIN ENCRYPTED PRIVATE KEY-----\n\"
"MIIJjJBABgkqhkiG9w0BBQ0wMzAbBgkqhkiG9w0BBQwwDgQIh7VHf699+v0CAgGA\n\"
"-----END ENCRYPTED PRIVATE KEY-----\n";

const char *cert = "-----BEGIN CERTIFICATE-----\n\"
"MIIFEDCCAvigAwIBAgIQ+UjKCIcDNLKcHf+9t1CFjANBgkqhkiG9w0BAQ0FADAR\n\"
"-----END CERTIFICATE-----\n";

```

## 8. Example for SSDP Response

**cgi\_ssdp.c** processes SSDP requests. All information tags can be modified using API functions, e.g. `GetDeviceName()`, `GetVendor()`, `GetModel()`, `GetDeviceUUID()`. All tags are defined and parsed by **cgi\_ssi.c**.

## 9. Example for Login/Sign in UI

`/auth/login.html` is the default page before authentication. The user must provide the username and password. All web pages are physically located in the folder defined by **LOCAL\_WEBROOT**.

**cgi\_auth.c** processes the user's sign-in/sign-out. This module responds to all requests with the prefix `"/auth/*"`.

`/auth/session_check.cgi` checks if the current session has timed out.

`/auth/logout.cgi` signs out the user and frees the session context.

## 10. Example for Status and Information

`/app/index.shtml` is the home page after authentication.

**cgi\_web.c** responds to all requests with prefix `"/app/*"` after authentication.

## 11. Example for Uploading

`/app/upload.shtml` and `/app/plugin/fileTransfer/ fileTransfer.js` provide a demo for uploading files. The destination folder is defined by **UPLOAD\_TO\_FOLDER**.

**cgi\_upload.c** responds to the request URL `"/api/upload.cgi"`.

## 12. Example for File Explorer

`/app/files.shtml` and `/app/plugin/fileList/ fileList.js` provide a demo for file browsing. The directory is defined by **FOLDER\_TO\_LIST**.

**cgi\_files.c** responds to the request with URL `"/api/files.cgi"`

## 13. Example for Form

`/app/form.shtml` is a demo for modifying parameters. All parameters are processed by **cgi\_ssi.c**.

**cgi\_form.c** provides general processing for all forms. All parameters and types are defined in **cgi\_ssi.c**.