MOD HTTP-Server

Author: Yannis Plaschko

(TINF20C, SWE I Praxisprojekt 2021/2022)

Project: Websockets in a lwIP HTTP Server

Customer: Rentschler & Holder
Rotebühlplatz 41
70178 Stuttgart

Supplier: Team 4: Laura Reeken, inf20051@lehre.dhbw-stuttgart.de
Benjamin Esenwein, inf20074@lehre.dhbw-stuttgart.de
Yannis Plaschko, inf20093@lehre.dhbw-stuttgart.de
Maximilian Meier, inf20084@lehre.dhbw-stuttgart.de
Lucas Kaczynski, inf20147@lehre.dhbw-stuttgart.de
Isabel Schwalm, inf20085@lehre.dhbw-stuttgart.de
Rotebühlplatz 41
70178 Stuttgart

Version	Date	Author	Comment
0.1	29.04.2021	Maximilian Meier	created
0.9	02.05.2021	Maximilian Meier	Filled in details
1.0	04.05.2022	Maximilian Meier	Finalized document

CONTENTS

1. Scope

This documentation discusses the Graphical User Interface of the lwIP Server to the test the functionality of the restful API.

2. Abbreviations

- GUI – Graphical User Interface
- API – application programming interface
- lwIP – Lightweight IP

3. Module Requirements

3.1 Demo Website

This module serves a website to the User, which can be used to communicate with the lwIP Server.

Following features have been implemented:

- The user can issue a restful API request to “<lwIP-adress>/identification”, to which the server correctly responds in the JSON format.
- A button exists on the demo website, that can forward the user to the “<lwIP-adress> /identification” endpoint.

3.2 Requirements

This module implements the following Project requirements: /NF10/, /F40/, /BUG40/, /BUG50/.

/NF10/, /BUG50/: For this nonfunctional requirement to be met the additional code must be kept to a minimum. This requirement has been met.

/F40/, /BUG40/: This functional requirement is about the test client. The backend of which has been implemented by this Module in the form of an HTTP-Server. An API Endpoint which when called returns a JSON-Object with information including the IP-Address of the Server which provides the endpoint.

3.3 Module Context

This module provides the backend to send a basic website containing explanations and to send a JSON response containing information about the server. It achieves this by implementing two different routes to the API. The first route is the base route. It responsds to a get request made to the “<lwIP-adress>” of the server and responds with a simple HTML page. The second route responds to requests made to “<lwIP-adress>/identification” with said JSON. Both responses can easily be adapted by changing the char array in which they are defined.

4. Analysis

The HTTP-Server simply returns the JSON-Object when the correct endpoint is called. Since lwIP largely does not contain the necessary information needed to fill the object dynamically the servers IP-Address was added to demonstrate that this is possible.

5. Design

```
{ "DeviceClass": "Windows PC", "Manufacturer": "lwIP - A Lightweight TCP/IP stack", "IP-Adress":
"192.168.178.82","Model": null, "ProductCode": null, "HardwareRevision" : 1, "SoftwareRevision" :
"2.1.0", "SerialNumber" : null, "ProductInstanceUri" : "https://savannah.nongnu.org/projects/lwip/",
"applicationSpecificTag" : null, "geolocation": null, "sysUpTime" : null }
```

6. Implementation

The endpoints are implemented in the httpserver_netconn.c file by reading out the buffer and determining the called endpoint from the GET-Request body. Note: an easy way to shorten the code used is by reducing the length of the endpoint to be called. While the IP-Address would be available in this context without getting it from the request itself the length is not provided and cannot be easily determined. Since a correctly formatted JSON-Object is the priority additional code had to be written to get the length of the IP-Address. In order to put the IP-Address into the JSON-Object it must be constructed from multiple char arrays. Depending on how the returned object is supposed to be structured, more or less individual arrays might have to be created and added together into the final object http_index_html2. For the corresponding header the content type was specified to `application/json` to specify that the returned objected is in the JSON format.

```
if (buflen>=7 &&
        buf[0]=='G' &&
        buf[1]=='E' &&
        buf[2]=='T' &&
        buf[3]==' ' &&
        buf[4]=='/' &&
        buf[5]=='i' &&
        buf[6]=='d' &&
        buf[7]=='e' &&
        buf[8]=='n' &&
        buf[9]=='t' &&
        buf[10]=='i' &&
        buf[11]=='f' &&
        buf[12]=='i' &&
        buf[13]=='c' &&
        buf[14]=='a' &&
        buf[15]=='t' &&
        buf[16]=='i' &&
        buf[17]=='o' &&
        buf[18]=='n') {

       […]

         char startBlock[] = "{ \"DeviceClass\": \"Windows PC\", \"Manufacturer\":
\"lwIP - A Lightweight TCP/IP stack\"";
        char ipBlock[] = ", \"IP-Adress\": ";
        char secondBlock[] = ",\"Model\": null, \"ProductCode\": null,
\"HardwareRevision\" : 1, \"SoftwareRevision\" : \"2.1.0\", \"SerialNumber\" : null,
\"ProductInstanceUri\" : \"https://savannah.nongnu.org/projects/lwip/\",
\"applicationSpecificTag\" : null, \"geolocation\": null, \"sysUpTime\" : null";
        char endBlock[] = " }";

       […]

         netconn_write(conn, http_html_hdr2, sizeof(http_html_hdr2) - 1,
       NETCONN_NOCOPY);
```

```
        netconn_write(conn, http_index_html2, size - 1, NETCONN_NOCOPY);
      }
```

For a plain GET-Request to the default route the GUI page is returned as described in MOD Graphical User Interface (GUI).

7. Module Tests

This Section contains Tests, whether the Requirements raised by this module function according to the specifications. Further Descriptions of the Tests can be found in the System test plan.

7.1 Module Testplan

Test-ID	Req. - ID	Functionality
1	LF60: Call API root	This test verifies that calling the Root Endpoint returns a sample website
2	LF70: Call API /something	This test case verifies that calling an undefined Endpoint result in an error Message
3	LF80: Call API /identification directly	This test case verifies that a direct call displays information about the Server.
4	LF90: Call API /identification from root	The test case verifies whether a click on the link redirects to displays information about the Server.

7.2. Module Testreport

Test-ID	PASS / FAIL	When failed: Observation	Tester
1	PASS	-	Y. Plaschko
2	PASS	-	Y. Plaschko
3	PASS	-	Y. Plaschko
4	PASS	-	Y. Plaschko

8. Summary

This module was successfully implemented with all requirements. It is now possible to start LwIP with a basic API. The API provides a root route, which responds with a basic website, and the “/identification” route, which responds with Information about the Server in JSON format.