MOD WebSocket 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	Yannis Plaschko	created
0.9	02.05.2021	Yannis Plaschko	Filled in details
1.0	04.05.2022	Yannis Plaschko	Finalized document
1.1	05.05.2022	Yannis Plaschko	Document format fix

CONTENTS

1. Scope

This Documentation discusses the Implementation of the WebSocket Support Patch and the creation of a Testclient.

2. Abbreviations

- API – application programming interface
- lwIP – Lightweight IP

3. Module Requirements

3.1 Webserver

This module provides a Server, to which a connection can be established via WebSockets. WebSocket Connections can be established using a Software like Postman or using the Testclient.

3.2 Requirements

The following requierments were implemented through this module: /F10/, /F20/, /NF10/, /NF20/, /NF50/, /BUG10/, /BUG20/, /BUG30/

/NF10/, /BF10/: This nonfunctional requirement tries to keep the additional code to a minimum to be as compatible as possible with embedded systems which tend to have limited resources. The implementation of lwIPs own BASE64 function, even after refactoring to make it usable, fulfills this requirement. While refactoring /BUG10/ arose which was fixed as described in the /BUG10/ issue.

/NF20/, /BF10/: This nonfunctional requirement is about efficient code. And has been fulfilled by removing substantial amounts of overhead code as mentioned previously.

/NF50/, /BUG30/: For this requirement the program is supposed to be compiled into an executable file. This however is not practical due to /BUG30/. This also makes it impossible to retroactively switch or upgrade the example apps and modules as needed. As a compromise the network adapter can be chosen manually while the modules are predetermined. For further information refer to /BUG30/.

/F10/, /BUG10/: This functional requirement requires the refactoring of the BASE64 function provided by lwIP and the WebSocket patch as well as rewriting parts of the patch to suit the new BASE64 function. This requirement has been fulfilled after the /BUG10/ had been resolved.

/F20/, /BUG20/: This functional requirement is about the WebSocket server. The WebSocket server should be able to accept and correctly parse WebSocket requests. After the initial handshake it should be possible to exchange messages over this connection. However, WebSocket connections cannot be established due to the program running into an error as described in /BUG20/.

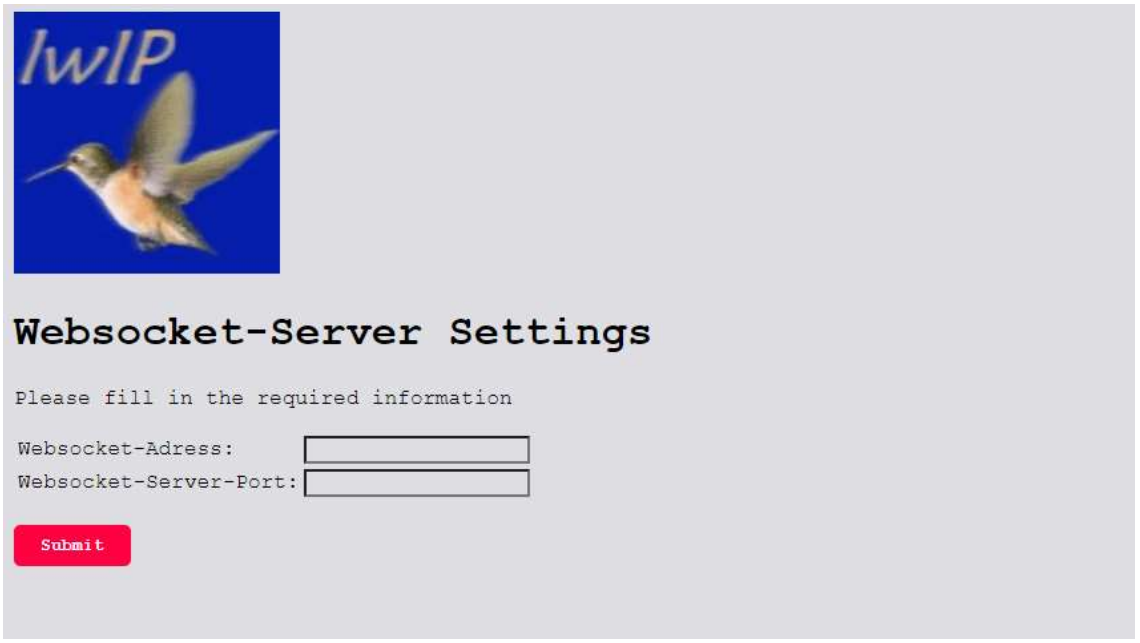
3.3 Module Context

This module introduces the WebSocket support to the core functionality of lwIP. The WebSocket Server is tasked with receiving a connection request from a client, establishing a connection with said client and the correct handling of whatever request the customer implements. If there are any Problems an expressive error message should be logged. This backend focused Part of the Module is further extended by the development of a Testclient used to connect to the backend and demonstrate the functionality of WebSockets by creating a simple echo service. For this the Testclient can connect to a WebSocket by inserting the IP into a text field and clicking a button. Another part of the Testclient is to send and display Messages in a for messengers' typical combination of text field and input field with a button to send messages. Furthermore, the Testclient can destroy a connection by clicking on a button labelled “Disconnect”.

4. Analysis

The WebSocket server must have the ability to receive incoming connection requests from a client and open a connection with this client. When a client-server connection has been established the webserver must be able to receive messages from the client. These messages should somehow be displayed on the server side. To show the bidirectional connection the server can simply echo the send message back to the client. When all tests have been conducted the WebSocket sever needs a function with which a client can close the connection again.

5. Design



Websocket-Server Settings

Please fill in the required information

Websocket-Adress:

Websocket-Server-Port:

6. Implementation

To implement WebSocket functionality the patch #9525 by Sakari Kapanen had to be implemented. From lwIP version 2.1.0 on this cannot be done automatically via the “git am” command after cloning the git repository because of changes to the folder structure of lwIP that came with version 2.1.0. A guide on how to manually implement the WebSocket patch can be found in this document: “Integration of patch #9525 into lwIP”. After implementing the patch, the “LWIP_HTTPD_SUPPORT_WEBSOCKET” in the “httpd_opts.h” file had to be set to 1 to enable all function connected to WebSockets. After the implementation the SHA1 and BASE64 functions provided by the patch were removed and replaced with lwIPs own implementations. To gain access to the BASE64 functionality of lwIP the function, related variables as well as constants had to be refactored into their own files and added to the lwIP library of included functions. To use the BASE64 function for WebSockets the patch had to be altered since the expected variables of the lwIP BASE64 function differ from the patch's own implementation. For further information on this please refer to /BUG10/. Using lwIPs own function has the benefit of already being approved fully by the lwIP community and since it is significantly more compact a lot of code is saved compared to the patch's version.
To use the SHA1 functions of lwIP, decoupled from the ppp-library (the cryptographic function library of lwIP), the original SHA1 header file had to be included. By doing that it is possible to circumvent the multiple layers of chained together headerfiles with additional dependencies which are unnecessary for WebSocket's. After these changes lwIP compiles without errors.

There is however no example implementation for the WebSocket patch. A problem that has previously been raised in the discussion thread about the WebSocket patch. After which the creator of the WebSocket patch left the project. During the implementation process it was discovered that the patch fails to recognize an incoming WebSocket connection request as a WebSocket request and therefor ignores it and crashes since the rest of the application can’t handle a request like this either. The implementation example for the ESP8266 Microcontroller is not compatible with Windows and using the Windows specific library causes several different compatibility issues with other libraries. This makes it impossible to actually test the implementation and improvements.

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	LF10: Choose the correct network adapter	Checks if lwIP starts correctly after choosing the correct network adapter.
2	LF20: Choose the wrong network adapter	Checks if lwIP fails noisy after choosing a wrong network adapter.
3	LF30: Connect to the WebSocket	This test verifies that the Testclient is able to connect to the WebSocket.
4	LF40: Message the Server	This test case verifies that a message sent to the server will be echoed back by it.
5	LF50: Disconnect from Websocket	This test case verifies that the Testclient can destroy an existing connection.

7.2. Module Testreport

Test-ID	PASS / FAIL	When failed: Observation	Tester
1	PASS	-	Y.Plaschko
2	PASS	-	Y.Plaschko
3	FAIL	Couldn’t be tested – no implementation of WebSocket's / Testclient	Y. Plaschko
4	FAIL	Couldn’t be tested – no implementation of WebSocket's / Testclient	Y. Plaschko
5	FAIL	Couldn’t be tested – no implementation of WebSocket's / Testclient	Y. Plaschko

8. Summary

This modules requirements couldn’t be implemented due to time constraints, thus it was impossible to execute the Testplan. The WebSocket patch has been inserted into the LwIP Structure but is not operational. In Accordance with Mr. Rentschler the focus was shifted on different requirements and further work on The WebSocket Support was discontinued.