System Requirements Specification

Author: Maximilian Meier

(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	26.10.2021	Maximilian Meier	created
1.0	8.11.2021	Maximilian Meier	Release Version

CONTENTS

1. Goal

The goal of this project is to address the architectural deficiencies of patch "#9525 (httpd: add websocket support)" [1] in coordination with the project community. This experimental base is to be improved and brought through the approval process in the open-source project.
Furthermore, a demo server is to be designed and implemented in a virtual environment under Windows.
For demonstration and testing purposes of the features, a GUI-based test client shall be designed and implemented.
For development and testing purposes the newest version of lwIP must be installed and made to run on a windows system, due to a lack of embedded systems at our disposal to run it on. From there the Patch #9525 must be implemented with lwIP on a Windows system before the architectural deficiencies can be addressed. To make sure that lwIP and its patch #9525 still work afterwards several tests are supposed be designed to ensure working order.

2. Use Case

2.1. <UC.001>

Related Business Process:	Prozess-ID: UC01
Use Cases Objective:	Customer is developing a CPU and wants to ensure support for TCP/IP and Websockets.
System Boundary:	lwIP allows the hardware to use TCP/IP.
Precondition:	The user can find and download lwIP via a conventional Internet browser. Setting up the software is adequately documented, and users can do it themselves.
Postcondition on success:	lwIP is executable on the CPU of the user.
Beteiligte Nutzer:	Role name: CPU developer
Triggering Event:	Developer gets the order, has own ambitions to extend the CPU with TCP/IP functionalities.

2.2. <UC.002>

Related Business Process:	Prozess-ID: UC02
Use Cases Objective:	Customer works with microcontrollers (e.g. ESP8266) as a hobby and wants to build a TCP/IP stack on top of it.
System Boundary:	lwIP allows the hardware to use TCP/IP.
Precondition:	The user can find and download lwIP via a conventional Internet browser. Setting up the software is adequately documented, and users can do it themselves.
Postcondition on success:	lwIP is executable on the user's microcontroller.
Beteiligte Nutzer:	Role name: Hobby developer

Triggering Event:	Developer gets the order, has own ambitions to extend the microcontroller with TCP/IP functionalities.

3. Non-functional requirements

3.1. /NF10/Keeping additional code to a minimum

lwIP primary focus is on a compact architecture that enables it to run on small embedded systems. If the project is to go through the approval process to become part of the main product, compact code will improve chances for approval.

3.2. /NF20/Efficient Code

For lwIP to run on systems with small processing power the code must be efficient. Efficient Code writing may improve our chances to get through the approval process.

3.3. /NF30/Intuitive GUI for Test-Client

The GUI for the Test-Client should be intuitive to use for any user to test the WebSockets functionality. The GUI should also be designed to be as accessible as possible and convey as much relevant information as necessary.

3.4. /NF40/Pass the Approval Process

If all architectural problems with patch #9525 can be resolved, we will submit our changes for approval to the maintainer of lwIP. It is however not guaranteed that the maintainer even has time to look at our submissions.

3.5. /NF50/Executable Program

At the end of the project it should be possible to use lwIP with patch #9525 as an easy to execute Program to simplify grading.

4. Functional Requirements

4.1. /F10/Extra function for base64 encoding

The code for base64 encoding should become its own function. This is due to the fact that it is used multiple times throughout the Project and implementing it each time is unnecessary and space consuming for a Program specifically designed to be light weight. Or in other words it is supposed to take up as little space and processing power as possible [1].

4.2. /F20/Implementation of WebSocket API

This is the main goal of patch #9525, to enable to usage of websockets. At the end of this project, it should be possible to exchange data through lwIP using websockets. Websockets enable bidirectional connections and keep the connection between server and client open, thus the server can send data without it being specifically requested by the client.

4.3. /F30/Writing of a GUI-based test client

For demonstration and testing purposes a test client shall be designed and build.

4.4. /F40/http Webserver for testing purposes

A http webserver may have to be implemented to enable testing of WebSocket functions and behaviour.

5. Product data

From a hardware point of view lwIP requires no more than tens of kilobytes of free RAM and roughly 40 kilobytes of code ROM. [2] It's small size and memory usage is the main feature of the software and code has to be written accordingly to ensure this remains true.

6. Bug fixes

7. Enhancements

8. References

[1] „lwIP - A Lightweight TCP/IP stack - Patches: patch #9525, httpd: add Websocket support [Savannah],“ [Online]. Available: https://savannah.nognu.org/patch/?9525. [Zugriff am 15 10 21].

[2] „lwIP wiki,“ [Online]. Available: https://lwip.fandom.com/wiki/LwIP_Wiki. [Zugriff am 07 11 2021].