Inhaltsverzeichnis

[1 Projektumsetzung der Gateway-Website 2](#_Toc64917427)

[1.1 Blockschaltbild 2](#_Toc64917428)

[1.1.1 Anforderungen 2](#_Toc64917429)

[1.2 Entwicklung 2](#_Toc64917430)

[1.2.1 SSH2 2](#_Toc64917431)

[1.2.1.1 Installation 2](#_Toc64917432)

[1.2.1.2 Ergebnisse & Fazit mit SSH2 3](#_Toc64917433)

[1.2.2 jQuery.ajax() 3](#_Toc64917434)

[1.2.2.1 Ergebnisse & Fazit mit jQuery.ajax() 3](#_Toc64917435)

[1.2.3 Ratchet 3](#_Toc64917436)

[1.2.3.1 Übertragen der Nachricht(en) 3](#_Toc64917437)

[1.2.3.1.1 http 3](#_Toc64917438)

[1.2.3.1.2 Ws 4](#_Toc64917439)

[1.2.3.2 Ergebnisse & Fazit 4](#_Toc64917440)

[1.2.3.3 Nächste Schritte 6](#_Toc64917441)

# Projektumsetzung der Gateway-Website

## Blockschaltbild

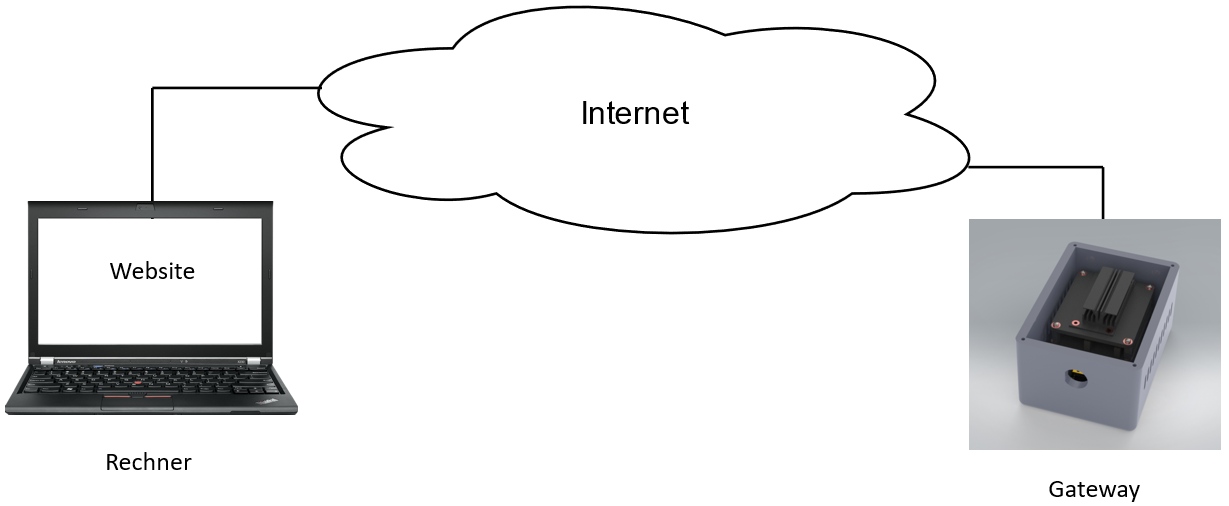


Abbildung 1: Blockschaltbild von Verbindung

### Anforderungen

Der Anwender soll über eine Website sich mit dem Gateway verbinden, um es darüber konfigurieren zu können, außerdem soll es möglich sein bestimmte Eckdaten schnell zu entnehmen. Einmal alles abgeschlossen, kann der Benutzer die Verbindung wieder schließen.

## Entwicklung

Bei der Entwicklung der Seite ist in erster Linie auf php (= eine Scriptsprache die, hauptsächlich zur Erstellung von Webanwendungen verwendet wird) gesetzt worden. Auf dem Weg zur fertigen und voll funktionsfähigen Website, kam es auch zu vielen Iterationen.

Versuche mit einer „ssh2-Library“ (ssh = ein Netzwerkprotokoll, mit dem es möglich ist eine Verbindung zu einem Server herzustellen), über jQuery.ajax() (Asynchronous JavaScript And XML) und schließlich mit der „Ratchet-Library“ (WebSockets für php) sind ausprobiert worden um eine echtzeitfähige bi-direktionale Verbindung zwischen dem Client (Benutzer auf der Website) und dem Server (Raspberry Pi Zero - Gateway) zu gewährleisten, diese werden in den nächsten Unterpunkten behandelt.

### SSH2

libssh2 ist eine client-seitige Library, welche das ssh-Netzwerkprotokoll implementiert. Diese ist jedoch nicht bei dem Standard-php-package dabei und muss deswegen separat installiert werden.

#### Installation

Für die Installation ist es notwendig, das Paket von der <http://libssh2.org> Seite zu installieren. Weiters muss man in das ursprüngliche php-installations-verzeichnis finden und dort das php.ini-file zu konfigurieren. Wichtig ist dort das man „extension=ssh2.so“ hinzufügt. Ist alles richtig gemacht worden, so kann man nun alle dazugehörigen Funktionen verwenden.

Mithilfe etlicher dieser Funktionen wurde erstmal eine Verbindung mit dem Raspberry über ein Heimnetzwerk aufgebaut. Nach dem Aufbau ist durch ein HTML-Form (HTML = Beschreibungssprache zur Strukturierung von Webanwendungen) ein Befehl eingelesen worden und dieses wurde dann zum vorläufigen Gateway geschickt und ausgeführt.

#### Ergebnisse & Fazit mit SSH2

Bei diesem Versuch ist es gelungen einen simplen Command abzusetzen. Pwd beispielsweise, ein Befehl um den aktuellen Pfad auszugeben, wurde richtig eingelesen, ausgeführt und wieder zurückgeschickt, wenn aber zeitsensitiven Befehle (z.B. ping) eingesetzt werden ist keine Antwort erschienen. Zusätzlich kam es nach jedem betätigen zu einem Seiten-refresh, für die Endanwendung unbrauchbar und aus diesen Gründen kam es zum nächsten Lösungsversuch um die Konfigurierung zu gewährleisten.

### jQuery.ajax()

jQuery.ajax()-Methoden sind generell da um Daten mit einem Server auszutauschen und gewisse Teile auf einer Webseite upzudaten ohne gleich die alles neu anzufordern.

#### Ergebnisse & Fazit mit jQuery.ajax()

Bei diesem Lösungsansatz ist nicht viel erreicht worden, da eine Verwirklichung des Ziels mit dieser Methode am besten mit Node.js (Eine JavaScript-Laufzeitumgebung, welche auch außerhalb eines Browsers JS-Code ausführen kann) gelöst hätte werden sollen. Nach langer Recherche ist keine gutes Ende in Sicht gewesen, ohne ganz von php wegzutreten und das war nicht Ziel dieses Teiles der Diplomarbeit.

### Ratchet

Ratchet ist eine php-library die es einem ermöglicht über das WebSockets-Protokoll Daten bi-direktional auszutauschen und das zugleich in Echtzeit, das ist für diese Anwendung sehr wichtig, da die Kommunikation nicht nur einseitig verlaufen darf und da der Port (=Schnittstelle) nach dem Handshake immer offen ist, können auch Zeitsensitive Befehle abgesetzt werden.

#### Übertragen der Nachricht(en)

Beschreibung und Vergleich zwischen dem meist-verwendeten-Protokoll, http und dem für die Anwendung verwendeten Protokoll, ws (WebSockets).

##### http

http steht für hypertext transfer protocol und dient zur Übertragung von Daten. Das auf TCP basierende Protokoll wird hauptsächlich für Webseiten aus dem Internet verwendet, um diese in einem Browser zu laden, es kann jedoch auch auf anderen Anwendungsgebieten verwendet werden.

Bei einer reinen http-Verbindung ist jedoch jede Aktion vom Server nur mit einer davor verschickten Anfrage des Clients möglich.

##### Ws

Das WebSockets-Protokoll ist genauso wie das http-Protokoll auf TCP basierend. Es wurde für eine bidirektionale Verbindung entworfen, damit der Webserver mit dem Client jeder Zeit, ungehindert nach dem obligatorischen Handshake, kommunizieren kann.

#### Ergebnisse & Fazit

Um sich mit WebSockets vertraut zu machen, wurde erstmals ein Chat-Demo-Programm programmiert, welches mehrere Clients mit dem Server verbinden lässt, in der ersten Version war es nur möglich über dem Terminal mit dem Server zu kommunizieren, jedoch funktionierte das hier schon in real time.

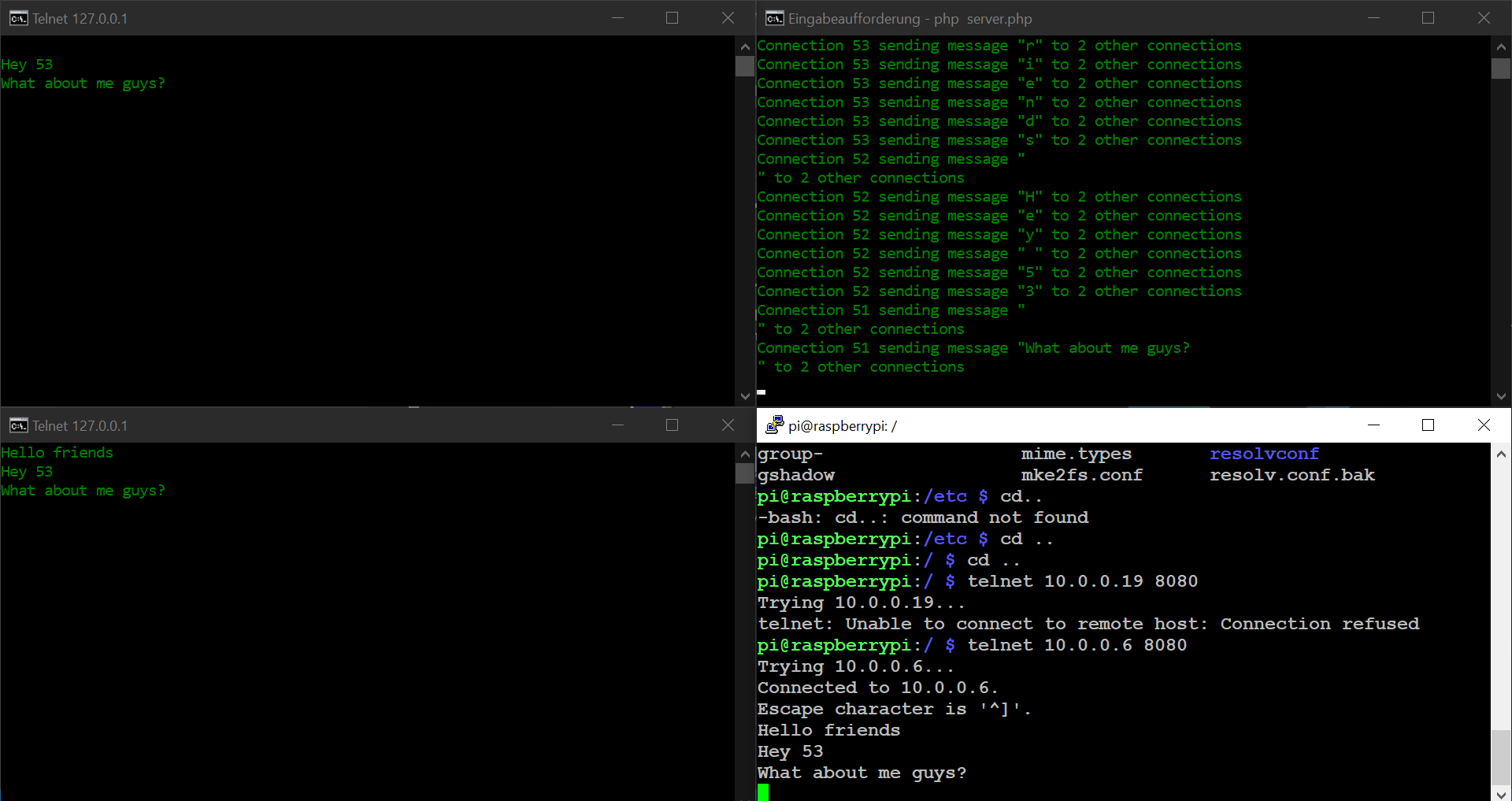


Abbildung 2: Chat-Anwendung V01

Im nächsten Schritt ist eine richtige Client-Seite mit jQuery (= JavaScript Framework) geschrieben worden, diese ermöglicht alle Daten auch im Browser auszugeben.

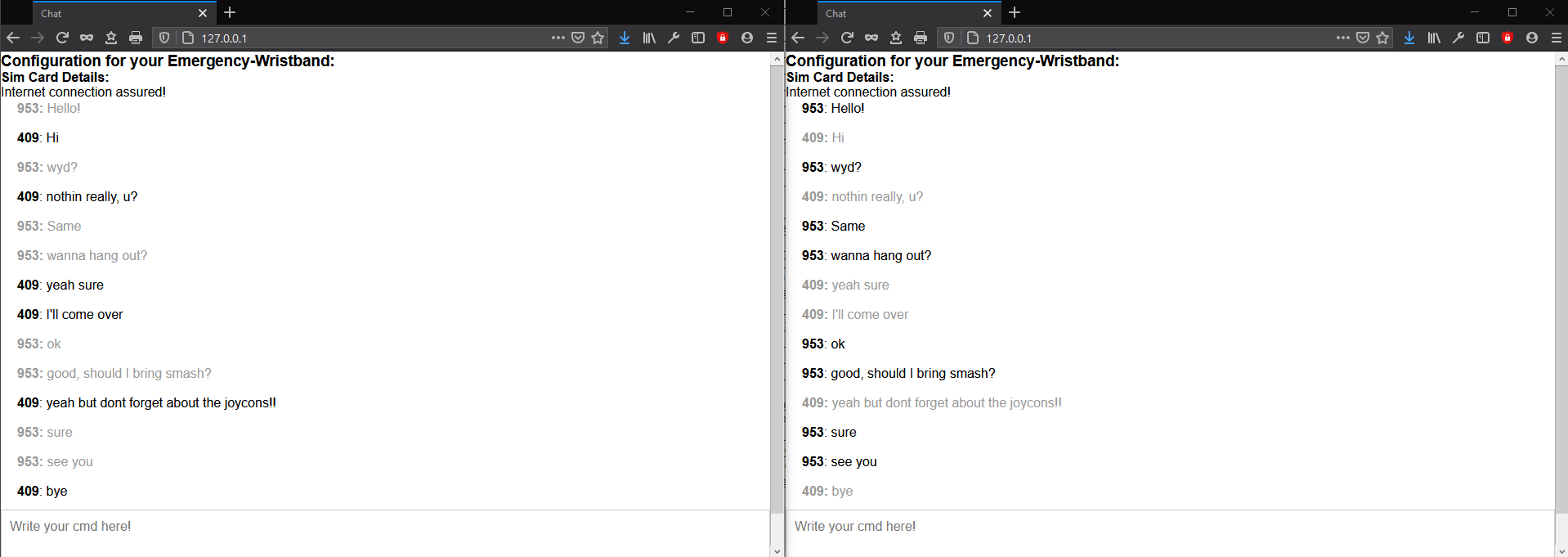


Abbildung 3: Chat-Anwendung V02

Aktuell ist diese Seite umgeschrieben um Skripte und Befehle von der Seite selbst absetzen zu können, diese Skripte sind einfache Shell-Skripte und sie dienen zum Verbindungsaufbau zum Raspberry selber, zur Installation von notwendigen Bestandteilen des Gateways, zum Überprüfen der Verbindung und schließlich für die Schließung der Verbindung. Ein weiterer Unterschied zur Vorgängerversion ist, dass mit dem Design begonnen wurde. Simples css (css = Beschreibungssprache um Weboberflächen zu designen) ist eingebaut, bestimmte Buttons verwenden aber auch in dieser Version schon das Design aus dem weitverbreiteten Google-Framework „Material Design“.

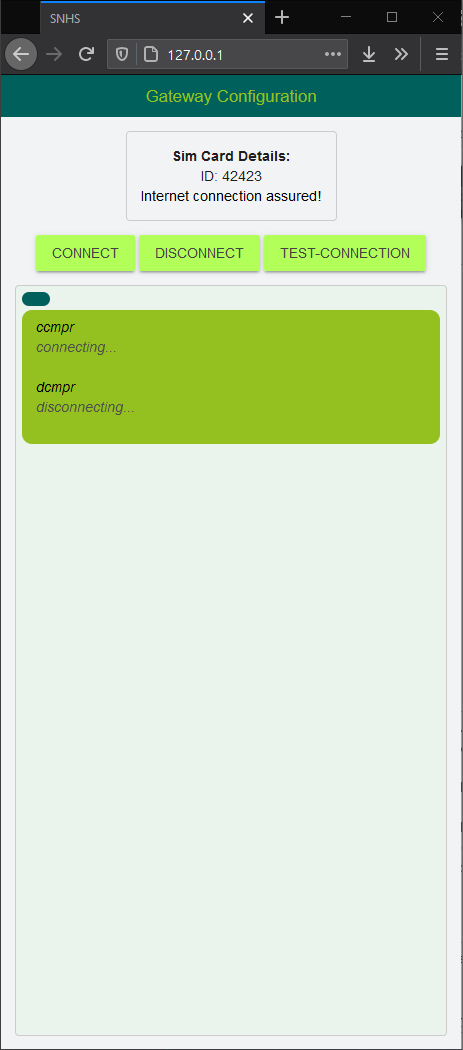


Abbildung 4: Konfigurations-Anwendung V01

#### Nächste Schritte

Die Kommunikation selbst ist fertig und funktionsfähig, wichtig ist jetzt die Server-Anwendung auf dem Gateway zum Laufen zu bringen und auch zu testen. Dort ist natürlich auch zu berücksichtigen, dass es ein Linux-Gerät ist, Anpassungen müssen getroffen werden. Die Client-Seite wird auch neu strukturiert und mithilfe von Material-Design auch optisch komplett erneuert.