



Berner Fachhochschule  
Haute école spécialisée bernoise  
Bern University of Applied Sciences

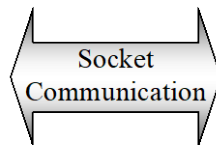
# Projektarbeit Webhaus

---

## Socket-Programmierung



PC-client :  
Browser mit  
HTML-Benutzeroberfläche  
Und JavaScript



Webhaus:  
Webserver  
C-Programm

© 2014 BFH-TI / E. Firouzi

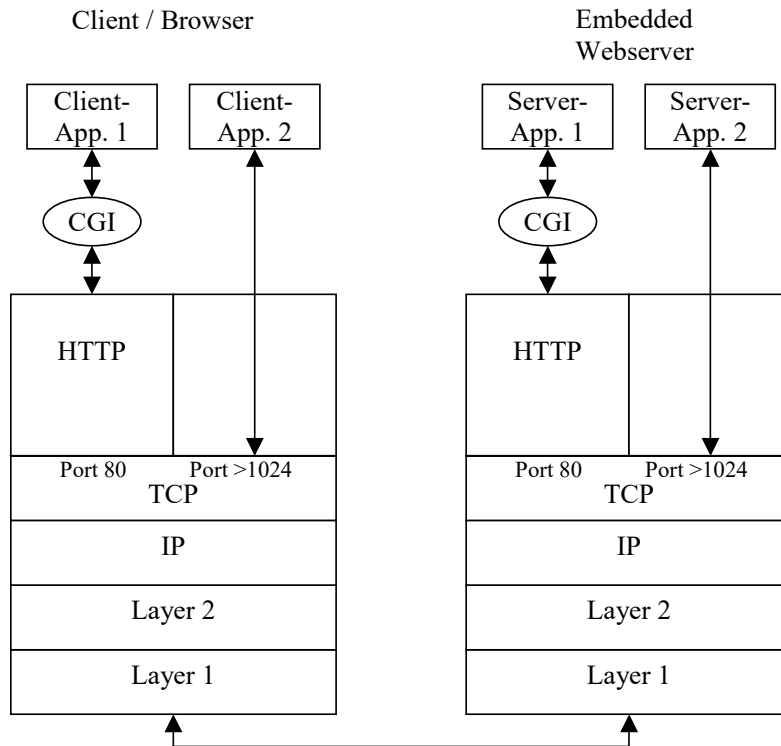
Letze Änderung:	Dezember 2021
Autor:	Elham Firouzi
Version:	1,1

# Inhaltsverzeichnis

1	Ziel des Projekts.....	3
1.1	Randbedingung .....	3
1.2	Aufgabenstellung .....	3
2	Erstellen der HTML-Benutzeroberfläche für das Webhaus.....	4
2.1	Endziel .....	4
3	Programmierung des Servers für das Webhaus in C .....	5
4	Startup Code in Linux .....	7
4.1	Script für das Starten der Serverapplikation Webhaus.....	7
4.2	Script für das Senden der IP-Adresse.....	8
5	Fernbedingung des Raspberry-Pi-Kit mit VNC-Viewer .....	12
5.1	Installation vom VNC-Viewer auf dem PC.....	12
5.2	Aktivierung vom VNC-Viewer auf dem Raspberry Pi .....	12
5.3	Festlegen der Bildschirmauflösung für den VNC Viewer-Modus .....	13
5.4	Start vom VNC-Viewer.....	14
6	Austausch von Dateien mit WinSCP .....	16
6.1	Installation von WinSCP auf dem PC .....	16
6.2	Starten vom WinSCP .....	16
7	Bibliothek « Webhouse . h ».....	18
7.1	Installieren der Peripherie-Bibliothek « bcm2835 . h ».....	19
7.2	Projektcompilation .....	19
8	jansson.....	21
8.1	Installieren der Bibliothek jansson.....	21

# 1 Ziel des Projekts

Das Ziel von dieser Projektarbeit ist eine grafische Benutzeroberfläche in HTML zu programmieren, um mehrere Ein- bzw. Ausgänge von einem Embedded System zu steuern. Dafür muss sich ein PC (mit einem Browser, um die HTML-Seite der Benutzeroberfläche darstellen zu können) mit einem Raspberry-Pi-Kit (welches ein Webhaus modelliert) über die Ethernet-Schnittstelle verbinden können. Die folgende Abbildung stellt die ISO/OSI-Schichten der Client-Server-Kommunikation dar.



## 1.1 Randbedingung

Diese Arbeit ergibt 25% von einem Teil (80%) der Modulnote, und sie sollte in Zweiergruppen realisiert werden. Am Ende der Arbeit müssen Sie den Quellcode der gesamten HTML-Benutzerschnittstelle und den C-Code des Webservers per E-Mail zu kommen lassen.

## 1.2 Aufgabenstellung

Definieren Sie auf der Clientseite die Benutzeroberfläche in HTML, um Anträge ans Webhaus zu senden; und auf der Serverseite das C-Programm, um diese Anträge auf dem Webhaus auszuführen. Dabei sollte die Benutzeroberfläche alle Lampen, den Fernseher, die Heizung und die Alarmanlage ein- bzw. ausschalten können. Zusätzlich muss auch die Raumtemperatur im Webhaus kontinuierlich angezeigt werden können. Optional sollten auch die Lichtintensität gedimmt oder die Raumtemperatur reguliert werden können.

## 1.3 Bewertungskriterien

Die Gewichtung der Bewertungskriterien ist wie folgt:

- 25% für die grafische Benutzeroberfläche in HTML.
- 25% für die vom System angebotenen Funktionalitäten
- 25% für die Struktur des server- und benutzerseitigen Codes
- 25% für die Qualität des server- und benutzerseitigen Codes

## 2 Erstellen der HTML-Benutzeroberfläche für das Webhaus

Erstellen Sie die HTML-Seite mit dem JavaScript-Programm, welche die Bedienung des Hauses aus einem Browser ermöglicht.

Einige Tipps:

- Definieren Sie ein Protokoll für die Kommunikation zwischen der HTML-Seite und des Webservers. Idealerweise übertragen Sie für jeden Befehl einen String, den Sie zu Testzwecken auch ausgeben können. Beispielsweise könnte der Befehl die Lampe 1 einzuschalten wie folgt definiert werden: `"<L1on>"`.
- In JavaScript kann die Verbindung mit dem Embedded Webserver mit wie folgt aufgebaut werden: `var websocket = new WebSocket ("ws://192.168.1.100:8000")`. Dabei entspricht `192.168.1.100` an der IP-Adresse des Embedded Webservers und `8000` an die Portnummer der Serverapplikation. Die Ereignisse `onopen` oder `onerror` des Objekts `websocket` erlauben zu testen, ob die Verbindung mit dem Server ohne Fehler aufgebaut werden konnte oder nicht. Das Kapitel 25.6 vom Skript enthält zusätzliche Informationen über die Klasse `WebSocket`.
- Die zu sendenden Meldungen können mit der Methode `send()` von Objekt `websocket` wie folgt gesendet werden: `websocket.send("<L1on>")` ;
- Die angetroffenen Meldungen können mit Hilfe vom Socketereignis `onmessage` des Objekts `websocket` wie folgt eingelesen werden.  

```
websocket.onmessage = function (message) {
    var received_msg = message.data;
}
```
- Wir empfehlen Ihnen in der ersten Version die Steuerung durch einige einfache Kontroll-Elemente zu realisieren. Sie können eine etwas aufwendigere Grafik-Version mit Bildern in einem zweiten Schritt realisieren.

### 2.1 Endziel

Die grafische Benutzeroberfläche des Webhauses muss über ihre IP-Adresse im Internetbrowser heruntergeladen werden können. Dazu muss noch einen Internetserver auf Ihrem Raspberry-Pi-Kit installiert werden. Mit den folgenden Kommandozeilen-Befehlen können Sie beispielsweise den Apache-Internetserver installieren<sup>1</sup>:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt install apache2 -y
```

Anschließend müssen Sie alle Dokumente von Ihrer grafischen Benutzeroberfläche (HTML, CSS und JavaScript) auf dem virtuellen Laufwerk des Internetserver installiert werden. Die HTML-Hauptseite muss dabei noch in `«index.html»` umbenannt werden. Bei Apache befindet sich das virtuelle Laufwerks standardmässig an der folgenden Stelle `«/var/www/html»`. Dieser Speicherort kann jedoch in der Apache-Konfigurationsdatei `«/etc/apache2/sites-enabled/000-default»` geändert werden.

<sup>1</sup> <https://pimylifeup.com/raspberry-pi-apache/>

### 3 Programmierung des Servers für das Webhaus in C

Schreiben Sie ein C-Programm auf dem Webserver, welches mit der HTML-Seite kommuniziert und die I/O's ansteuert.

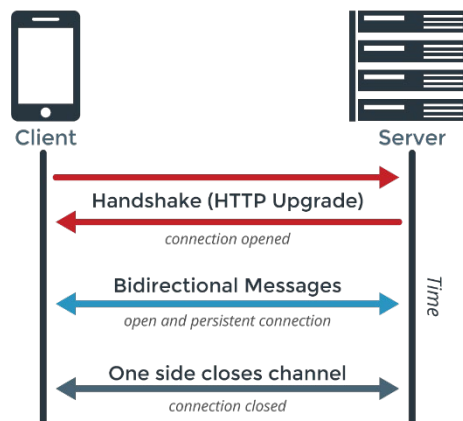
Zur Ansteuerung der I/O's können Sie die Funktionen von der Bibliothek "**Webhouse.h**" verwenden (siehe Kapitel 6 Bibliothek « **Webhouse.h** »).

Mögliches Vorgehen:

- Implementation der Socket-Verbindung:
  - Die Socketverbindung soll Schritt für Schritt mit den Funktionen der Standardbibliotheken `<sys/types.h>` und `<sys/socket.h>` aufgebaut bzw. abgebaut werden. Eine genaue Beschreibung von diesen Funktionen finden Sie im Kapitel 25.4 des Skriptes.
  - Bitten Sie den Serversocket mit der Portnummer **8000**, damit die Client sich über diese Nummer mit dem Server verbinden können.
  - Nach der Verbindungsaufbau zum Client mit der Funktion **accept()**, muss eine Handshake-Operation durchgeführt werden (siehe untere Abbildung). Während dieses Verfahrens erhält der Server einen Handshakeantrag über die Funktion **recv()**. Dieser Handshakeantrag enthält einen Schlüssel (**Web-Socket-Key**), welcher durch den gesicherten Hashalgorithmus 1 (secured hash algorithm 1) bearbeitet werden muss<sup>2</sup>. Das Ergebnis von dieser Verarbeitung (**Web-Socket-Accept**) muss anschliessend an dem Client mit der Funktion **send()** zurückgeschickt werden. Um diese Handshakeantwort zu erzeugen, wird die folgende Funktion der Bibliothek "**handshake.h**" zur Verfügung gestellt:

```
int get_handshake_response(char request[], char response[]);
```

Diese Funktion wandelt den erhaltenen Handshakeantrag in der zusendenden Handshakeantwort um. Dabei entspricht der Parameter «**request**» an dem erhaltenen Handshakeantrag und der Parameter «**response**» an der zu sendenden Handshakeantwort. **Achtung:** das Array «**response**» muss mindestens 130 Bytes gross sein.



- Nach dem Handshake Verfahren können die Daten zwischen dem Client und Server mit den Funktionen «**send**» und «**recv**» vertauscht werden. Um die Übertragungssicherheit zu verbessern, verwendet WebSocket ein Verschlüsselungsverfahren<sup>3</sup>. Für dieses Verschlüsselungsverfahren wurden die folgenden Funktionen der Bibliothek "**handshake.h**" zur Verfügung gestellt:

```
int decode_incoming_request(char coded_request[], char request[]);
```

```
int code_outgoing_response(char response[], char coded_response[]);
```

Die Funktion «**decode\_incoming\_request**» erlaubt den erhaltenen verschlüsselten Antrag zu decodieren; und die Funktion «**code\_outgoing\_response**» die zusendende Antwort zu codieren. **Achtung:** das Array «**request**» muss gleich gross sein wie das Array «**coded\_request**»; und das Array «**coded\_respons**» muss zwei Byte grösser sein als das Array «**respons**» sein.

<sup>2</sup> [https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API/Writing\\_WebSocket\\_servers](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers)

<sup>3</sup> [https://developer.mozilla.org/en-US/docs/Web/API/WebSockets\\_API/Writing\\_WebSocket\\_servers](https://developer.mozilla.org/en-US/docs/Web/API/WebSockets_API/Writing_WebSocket_servers)

Das folgende Beispiel stellt das Verfahren dar, welches in C implementiert werden muss, um die empfangenen WebSocket-Meldungen zu verarbeiten.

```
char rxBuf[RX_BUFFER_SIZE];
int rx_data_len = recv (com_sock_id, (void *)rxBuf,
                        RX_BUFFER_SIZE, MSG_DONTWAIT);
// Has a new WebSocket message have been received
if (rx_data_len > 0) {
    rxBuf[rx_data_len] = '\0';
    // Is the message a handshake request
    if (strncmp(rxBuf, "GET", 3) == 0) {
        // Yes -> create the handshake response and send it back
        char response[WS_HS_ACCLLEN];
        get_handshake_response(rxBuf, response);
        send(com_sock_id, (void *)response, strlen(response), 0);
    }
    else {
        /* No -> decode incoming message,
           process the command and
           send back an acknowledge message */
        char command[rx_data_len];
        decode_incoming_request(rxBuf, command);
        command[strlen(command)] = '\0';
        processCommand (command);
        char response[] = "<Command executed>";
        char codedResponse[strlen(response)+2];
        code_outgoing_response(response, codedResponse);
        send(com_sock_id, (void *)codedResponse,
              strlen(codedResponse), 0);
    }
}
```

- Ansteuerung der I/O's auf dem Webserver
  - Werten Sie die decodierte Telegramme, welche vom JavaScript-Programm empfangen werden, aus.
  - Implementieren Sie die Funktionalität der Ausgänge (Lampen, TV, Heizung).
  - Lesen Sie die Temperatur ein und senden Sie diese Informationen codiert dem JavaScript-Programm zurück.

## 4 Startup Code in Linux

Linux erlaubt Applikation direkt nach seinem Startverfahren zu starten. Dafür muss der Startup-Skript «`/etc/rc.local`» angepasst werden. Das folgende Beispiel zeigt, wie die Applikation Webhaus nach einer gültigen IP-Adresse gestartet werden kann. Im diesem Beispiel startet `rc.local` zwei weitere Skripts, nämlich «`emailNotify.sh`» und «`/root/startTemplate.sh`». Der erster Skript schickt ein E-Mail mit der IP-Adresse vom Raspberry-Pi-Kit und der zweite Skript startet den Webserver.

```
#!/bin/sh -e
#
# /etc/rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
printf "run emailNotify.sh\n"
sudo bash /root/emailNotify.sh &
printf "run startTemplate\n"
sudo bash /root/startTemplate.sh &

exit 0
```

### 4.1 Script für das Starten der Serverapplikation Webhaus

Das erste Shell-Skript «`/root/tartTemplate.sh`» startet die C-Applikation Webhaus. Am Anfang vom diesem Skripts wird gewartet, bis eine der Ethernet- oder Wifi-Schnittstellen einsatzbereit ist. Danach startet man die Anwendung Webhaus als Superuser.

```
#!/bin/bash
#
# /root/startTemplate.sh
#
# This script will be called from rc.local

# Wait for a valid IP-address during 1 minutes
count=0
while ! ifconfig | grep -F "10.19." > /dev/null && # wlan0 BFH bfh-open
      ! ifconfig | grep -F "147.87." > /dev/null && # eth0 BFH Quellgasse 21
      ! ifconfig | grep -F "172.10." > /dev/null && # wlan0 SIP sip-guest
      ! ifconfig | grep -F "172.31." > /dev/null && # eth0 SIP class room
      ! ifconfig | grep -F "192.168.1" > /dev/null ; # wlan0 home Swisscom
do
    count=$((count+1))
    if [ $count -eq 60 ]; then
        break
    fi
    sleep 1
done

# Start the WebSocket server
sudo /home/pi/webhouse/local/Firouzi/Template
exit 0
```

## 4.2 Skript für das Senden der IP-Adresse

Das zweite Shell-Skript «[/root/emailNotify.sh](#)» sendet eine E-Mail mit den IP-Adressen. Am Anfang vom diesem Skript wird wieder gewartet, bis eine der Schnittstellen Ethernet (**eth0**) oder Wifi (**wlan0**) einsatzbereit ist. Dann wird eine Textdatei erstellt, welche die IP-Adressen der Ethernet- und Wifi-Schnittstelle enthält. Schliesslich wird der Inhalt der Textdatei mit dem Python-Skript **mailing.py** als E-Mail gesendet.

```
#!/bin/bash
#
# /root/emailNotify.sh
#
# This script will be called from rc.local

# Wait for a valid IP-address during 1 minutes
count=0
while ! ifconfig | grep -F "10.19." > /dev/null && # wlan0 BFH bfh-open
      ! ifconfig | grep -F "147.87." > /dev/null && # eth0 BFH Quellgasse 21
      ! ifconfig | grep -F "172.10." > /dev/null && # wlan0 SIP sip-guest
      ! ifconfig | grep -F "172.31." > /dev/null && # eth0 SIP class room
      ! ifconfig | grep -F "192.168.1" > /dev/null ; # wlan0 home Swisscom
do
    count=$((count+1))
    if [ $count -eq 60 ]; then
        break
    fi
    sleep 1
done

# Define the email message
HOSTNAME='hostname -f'
EIP='hostname -I'
LIP='hostname -i'
echo "$HOSTNAME online" >> /root/email.txt
echo >> /root/email.txt
echo "Ethernet IP address: $EIP" >> /root/email.txt
echo "Local IP address: $LIP" >> /root/email.txt
echo >> /root/email.txt
date >> /root/email.txt
echo >> /root/email.txt

# Send the email message with mailutils
# mail -s "$HOSTNAME online" elham.firouzi@bfh.ch < /root/email.txt

# Send the email with python
sudo python /root/mailling.py

# Remove the email message
cat /root/email.txt
rm -rf /root/email.txt
exit 0
```

Das Python-Skript «[/root/mailling.py](#)» sendet die zuvor erstellte Textdatei elektronisch. Dazu sucht es unter den gängigsten Domains nach E-Mail-Austausch-Server. Anschliessend sendet es für jeden gefundenen Server eine E-Mail an die Adresse «[elham.firouzi@bfh.ch](mailto:elham.firouzi@bfh.ch)», welche die IP-Adressen enthält. **Achtung:** Die maximale Anzahl der zu versendenden E-Mails kann mit der Variablen «[max\\_successful\\_email](#)» begrenzt werden.

Damit Sie die IP-Adresse von Ihrem Raspberry-Pi-Kits per E-Mail erhalten können, müssen Sie die E-Mail-Adressen «[elham.firouzi@bfh.ch](mailto:elham.firouzi@bfh.ch)» durch Ihre eigene E-Mailadresse im Skript «[/root/mailling.py](#)» ersetzen.



```
#!/usr/bin/python3
# /root/mailling.py

import subprocess, sys, smtplib, nslookup

from email.mime.multipart import MIMEMultipart
from email.mime.text import MIMEText

#####
# to change: recipient & max_successful_email
#####
recipient = "fuel@bfh.ch"
max_successful_email = 3

successful_email = 0

# list of the 21 most popular mail server
# (https://domar.com/pages/smtp_pop3_server)
array_domain = []
array_domain.append('gmail.com')
array_domain.append('live.com')
array_domain.append('office365.com')
array_domain.append('yahoo.com')
array_domain.append('o2.ie')
array_domain.append('ntlworld.com')
array_domain.append('btconnect.com')
array_domain.append('btopenworld.com')
array_domain.append('btinternet.com')
array_domain.append('orange.net')
array_domain.append('wanadoo.co.uk')
array_domain.append('o2online.de')
array_domain.append('t-online.de')
array_domain.append('landl.com')
array_domain.append('lundl.de')
array_domain.append('comcast.net')
array_domain.append('verizon.net')
array_domain.append('zoho.com')
array_domain.append('mail.com')
array_domain.append('gmx.com')

for domain in array_domain:
    originator = "webhouse@" + domain

    print("\r\nFrom: " + originator)
    print("To:" + recipient)
    print("Domain: " + domain)

    mx_records = []
    mx_values = {'pref' : 0, 'serv' : ''}

    # search every mail exchange server for a given domain
    p = subprocess.Popen('nslookup -type=mx ' + domain + ' 8.8.8.8',
        shell=True, stdout=subprocess.PIPE, stderr=subprocess.STDOUT)
```

```

for line in p.stdout.readlines():
    line = line.decode().lower()
    if line.find("mail exchange") != -1 :
        for char in line:
            if str(char) in "\r\n\t":
                line = line.replace(char, ' ')
        if line.find("mx preference") != -1 :
            mx_parse = line.replace(' ', '').split(",")
            mx_values['pref'] = int(mx_parse[0].split("=")[1])
            mx_values['serv'] = mx_parse[1].split("=")[1]
        else:
            mx_parse = line.split(" = ")[1].split(" ")
            mx_values['pref'] = int(mx_parse[0])
            mx_values['serv'] = mx_parse[1]
        mx_records.append(mx_values.copy())
        print("\nline = " + line)

retval = p.wait()

if len(mx_records) == 0 :
    continue

# sort the mail exchange server upon their priority
print("\r\nSearch first priority mail exchange server for \"" +
      domain + "\"")

def mx_pref_sortvalue(record):
    return record['pref']
mx_records=sorted(mx_records, key=mx_pref_sortvalue)

# take the mail exchange server with the highest priority
server = mx_records[0]['serv']

# generate the mail message to send
msg = MIMEMultipart()
msg['From'] = originator
msg['To'] = recipient
msg['Subject'] = "IP-Address Webhouse"

body = open('/root/email.txt', 'r').read()
msg.attach(MIMEText(body, 'plain'))

# send the email with the mail exchange server
print("\r\nSending mail to: \"" + recipient + "\" via server: \"" +
      server + "\"")

try:
    smtp_send = smtplib.SMTP(server, 25)
    try:
        smtp_send.sendmail(originator, recipient, msg.as_string())
        successful_email += 1
    except:
        print("\r\nSending mail to: \"" + recipient +
              "\" via server: \"" + server + "\" has failed!")

```

```
        finally:
            smtp_send.quit()

    except:
        print("\r\nConnection to the server \"" + server +
              "\" has failed")

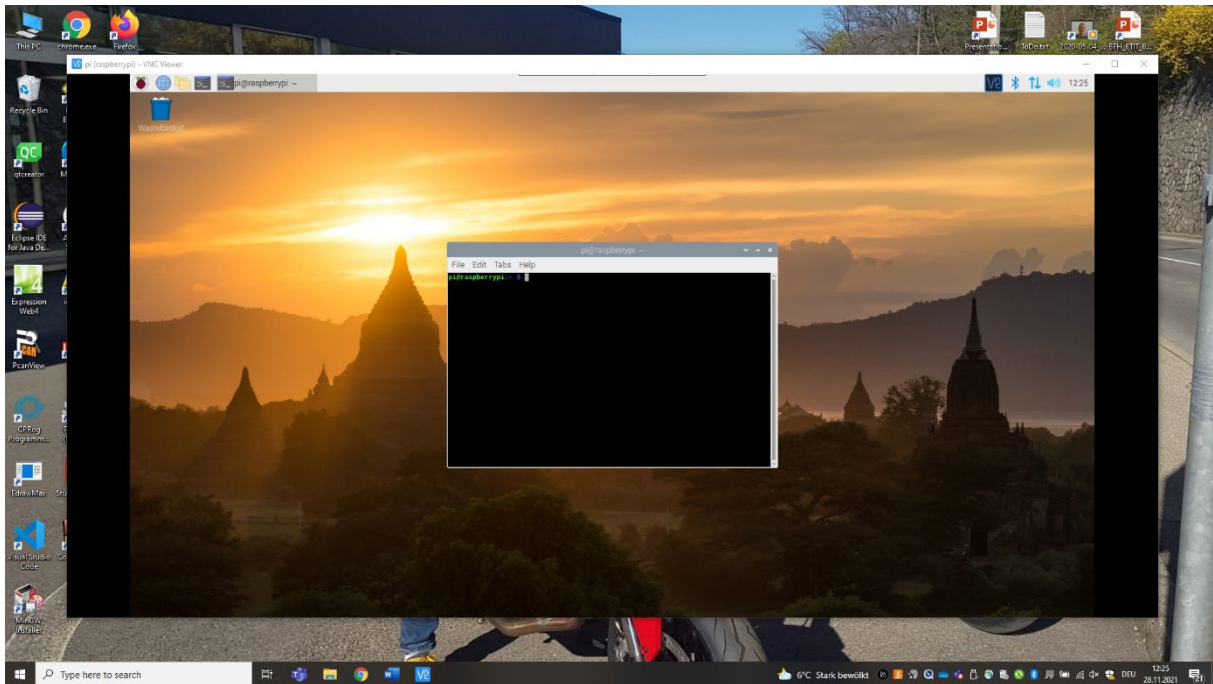
    if successful_email == max_successful_email:
        sys.exit(0);
```

Das obigen Python-Skript verwendet die Bibliothek **nslookup**. Diese Bibliothek kann mit den folgenden Shell-Befehle auf die Raspberry-Pi-Kits installiert werden.

```
sudo apt-get update
sudo apt-get install bind-utils
sudo pip install nslookup
```

## 5 Fernbedingung des Raspberry-Pi-Kit mit VNC-Viewer

VNC Viewer ist ein Softwaretool, mit dem Sie auf Ihr Raspberry Pi Kit von aussen zugreifen können. In diesem Fall können Sie Ihr PC als Bildschirm, Tastatur und Maus für Ihr Raspberry-Pi-Kit verwenden.

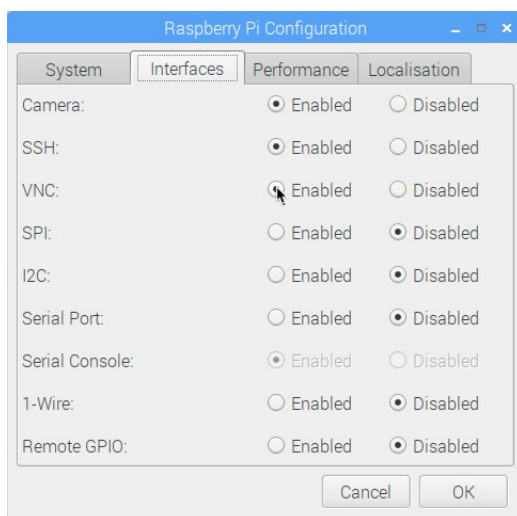


### 5.1 Installation vom VNC-Viewer auf dem PC

Mit der Anwendungs-Installer «**tightvnc-2.8.63-gpl-setup-64bit.msi**» können Sie den VNC-Viewer auf Ihrem PC installieren. Diese kann entweder von meiner Moodle-Seite oder von der offiziellen VNC-Website heruntergeladen werden<sup>4</sup>.

### 5.2 Aktivierung vom VNC-Viewer auf dem Raspberry Pi

Um den VNC-Viewer auf Ihrem Raspberry-Pi verwenden zu können, müssen Sie die VNC-Option im seinem Schnittstellenkonfigurationsfenster aktivieren<sup>5</sup>.



<sup>4</sup> <https://www.realvnc.com/en/connect/download/viewer/>

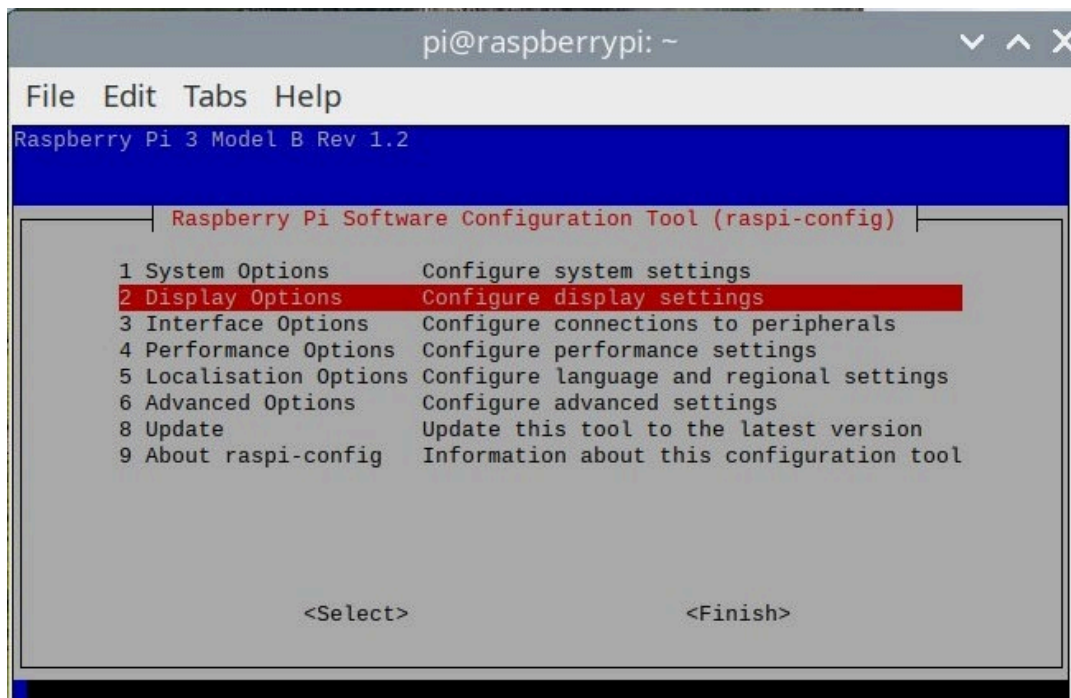
<sup>5</sup> <https://magpi.raspberrypi.com/articles/vnc-raspberry-pi>

### 5.3 Festlegen der Bildschirmauflösung für den VNC Viewer-Modus

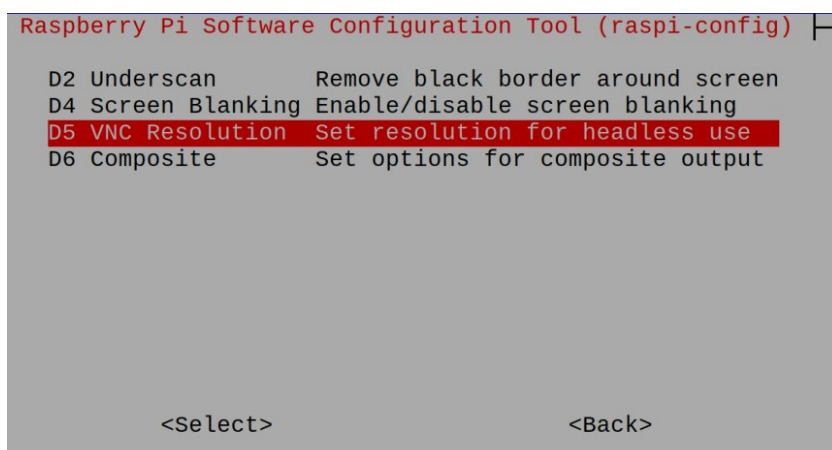
Nachdem Sie die VNC Viewer Option auf Ihrem Raspberry Pi Kit aktiviert haben, müssen Sie noch die Bildschirmauflösung für diesen Modus auf Ihrem Raspberry Pi Kit festlegen. Dazu müssen Sie den folgenden Befehl in eine Shell eingeben.

```
sudo raspi-config
```

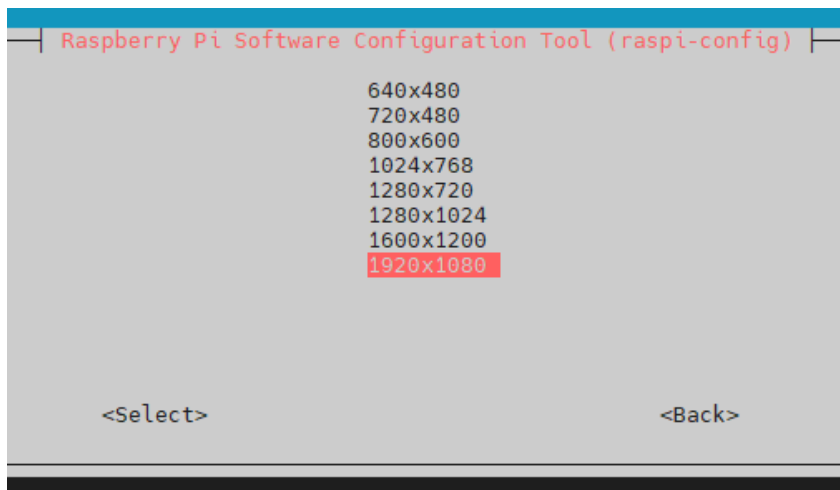
Dieser Befehl öffnet ein Konfigurationsfenster. In diesem Fenster müssen Sie die Option "2 Display Options" auswählen.



Diese Optionswahl öffnet ein weiteres Konfigurationsfenster, in dem Sie die Option "D5 VNC Resolution" auswählen müssen.



Schließlich können Sie im letzten Konfigurations-Fenster die Bildschirmauflösung für den VNC-Viewer-Modus auswählen.

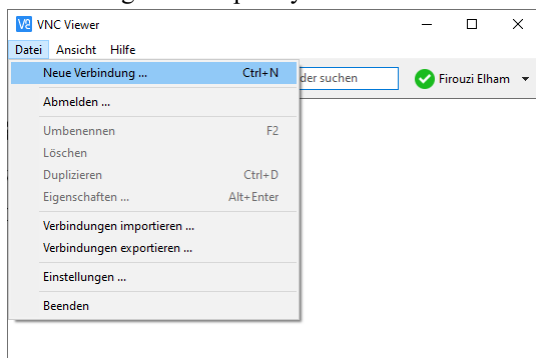


## 5.4 Start vom VNC-Viewer

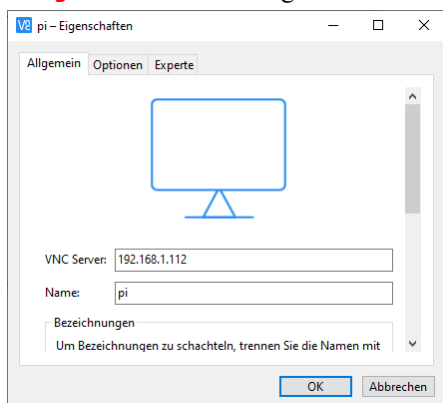
Nachdem VNC-Viewer vollständig installiert wurde, können Sie VNC-Viewer mit der folgenden Anwendungssikone auf Ihrem PC starten:




Nach dem Start des VNC-Viewers können Sie über das Menü «**Datei/Neue Verbindung ...**» eine neue Verbindung zum Raspberry-Pi-Kit herstellen.




Während des Verbindungsaufbaus müssen Sie zunächst die IP-Adresse Ihres Raspberry-Pi-Kits - welche Sie normalerweise per E-Mail erhalten haben - und den Benutzernamen «**pi**» in das Dialogfenster «**Eigenschaften**» eingeben.



Danach müssen Sie den Benutzernamen (**pi**) und das Benutzerpasswort (**pi**) im Dialogfenster «**Authentifizierung**» eingeben.

 Authentifizierung

✕

 **Authentifizierung bei VNC Server**  
192.168.1.112::5900 (TCP)


VNC Server-Anmeldeinformationen eingeben  
(Hinweis: NICHT Ihre RealVNC-Kontodaten)

Benutzername:

pi

Kennwort:

••



☐ Kennwort speichern

[Kennwort vergessen?](#)

Schlagwort:

Herman chief explain. Popular economy nice.

Signatur:

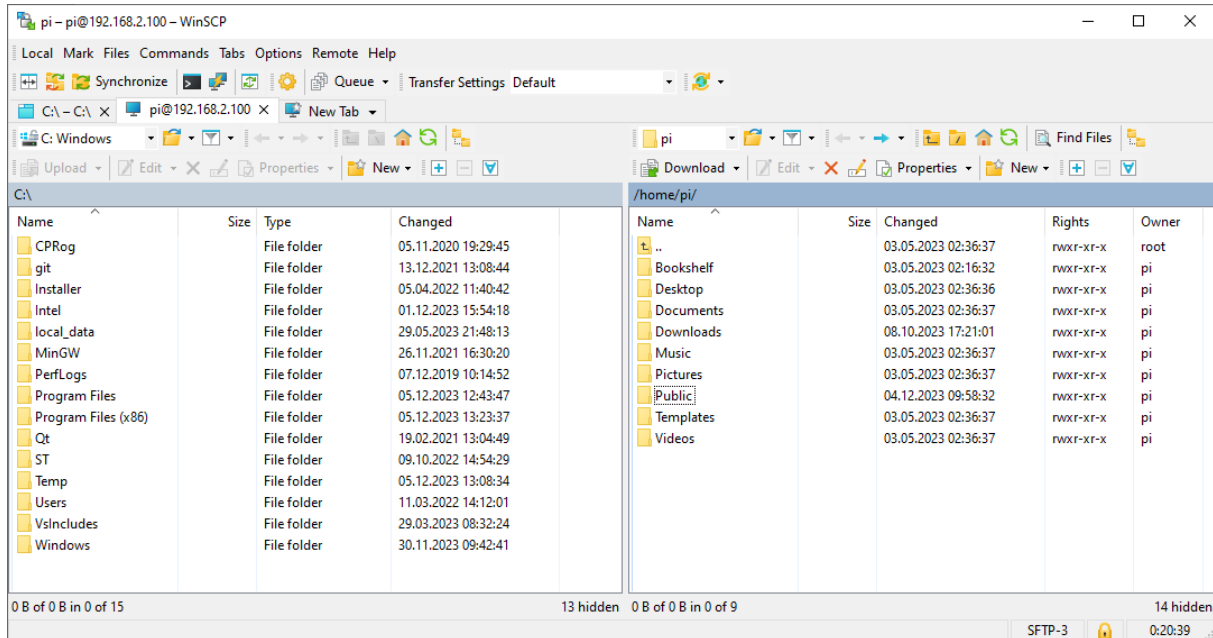
f3-8e-cf-d3-4d-01-c9-e5

OK

Abbrechen

## 6 Austausch von Dateien mit WinSCP

VNC Viewer erlaubt den Austausch von Dateien zwischen Ihrem PC und Ihrem Raspberry-Pi-Kit nur mit einer Lizenz. Deshalb müssen Sie Ihre Dateien entweder mit einem USB-Stick oder mit «WinSCP» auf dem Raspberry-Pi-Kit Herauf- bzw. Herunterladen.



### 6.1 Installation von WinSCP auf dem PC

Mit der Anwendungs-Installer «**WinSCP-6.1.2-Setup.exe**» können Sie den WinSCP auf Ihrem PC installieren. Diese kann entweder von meiner Moodle-Seite oder von der offiziellen VNC-Website heruntergeladen werden<sup>6</sup>.

### 6.2 Starten vom WinSCP

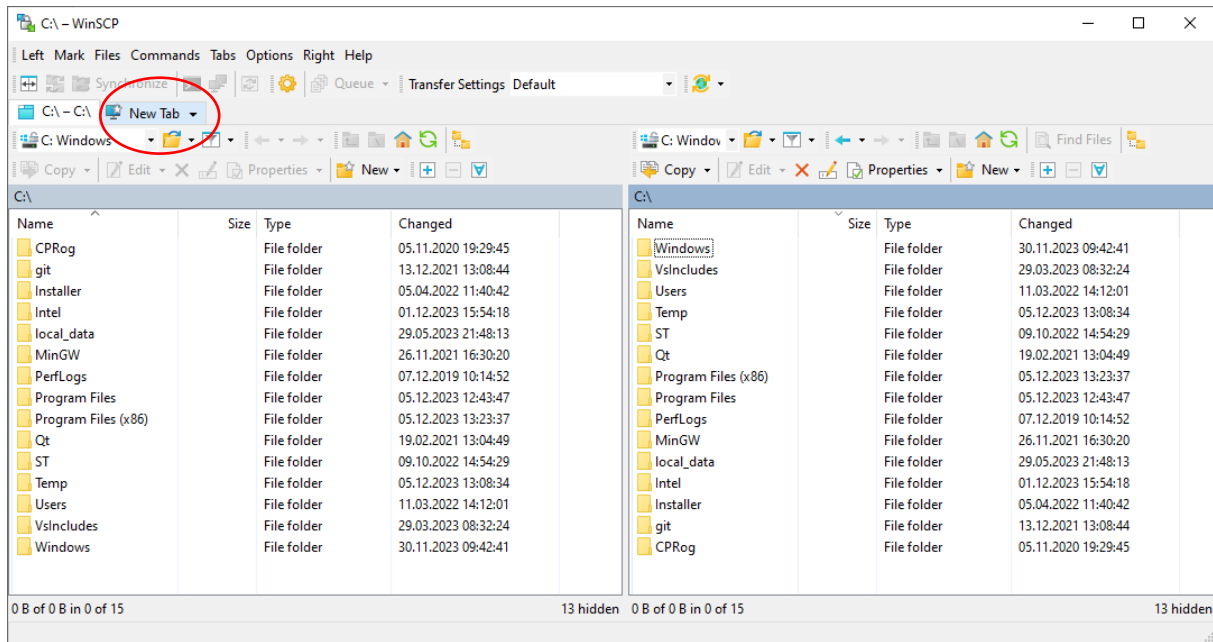
Nachdem WinSCP vollständig installiert wurde, können Sie WinSCP mit der folgenden Anwendungssikone auf Ihrem PC starten:



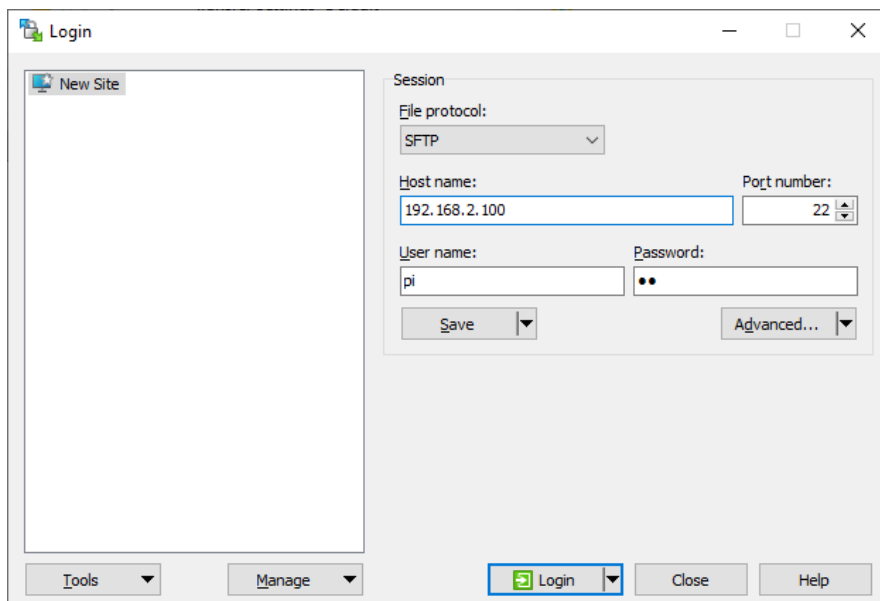
Nach dem Start von WinSCP können Sie über den Tab «**New Tab**» eine neue Verbindung zum Raspberry-Pi-Kit herstellen.

<sup>6</sup> <https://winscp.net/download/WinSCP-6.1.2-Setup.exe>





Dieser Tab öffnet ein Login-Fenster, mit dem Sie eine Verbindung des Typs «SFTP» (Secure File Transfer Protocol) mit dem Raspberry-Pi-Kit herstellen können. Dazu müssen Sie zunächst die IP-Adresse des Kits (192.168.2.100), den Benutzernamen (pi) und das Passwort (pi) eingeben. Anschließend müssen Sie die Taste "Login" drücken.



## 7 Bibliothek « Webhouse .h »

Die Bibliothek "**Webhouse.h**" stellt die folgenden Funktionen zur Verfügung, um die Ein- und Ausgänge des Webhauses zu steuern.

```
//-----Macros-----
#define PWM

#define GPIO_TV          RPI_BPLUS_GPIO_J8_03          //Pin 3
#define GPIO_dimSLamp    RPI_BPLUS_GPIO_J8_32          //Pin 32
#define GPIO_dimRLamp    RPI_BPLUS_GPIO_J8_33          //Pin 33
#define GPIO_Heat        RPI_BPLUS_GPIO_J8_05          //Pin 5
#define GPIO_Alarm        RPI_BPLUS_GPIO_J8_15          //Pin 15
#define GPIO_LED1        RPI_BPLUS_GPIO_J8_07          //Pin 7
#define GPIO_LED2        RPI_BPLUS_GPIO_J8_11          //Pin 11

//-----Function prototypes-----
extern void initWebhouse(void);
extern void closeWebhouse(void);

extern void turnTVOn(void);
extern void turnTVOff(void);
extern int getTVState(void);

extern void dimRLamp(uint16_t Duty_cycle);
extern void dimSLamp(uint16_t Duty_cycle);

extern void turnLED1On(void);
extern void turnLED1Off(void);
extern int getLED1State(void);

extern void turnLED2On(void);
extern void turnLED2Off(void);
extern int getLED2State(void);

extern void turnHeatOn(void);
extern void turnHeatOff(void);
extern int getHeatState(void);
extern float getTemp(void);

extern int getAlarmState(void);
```

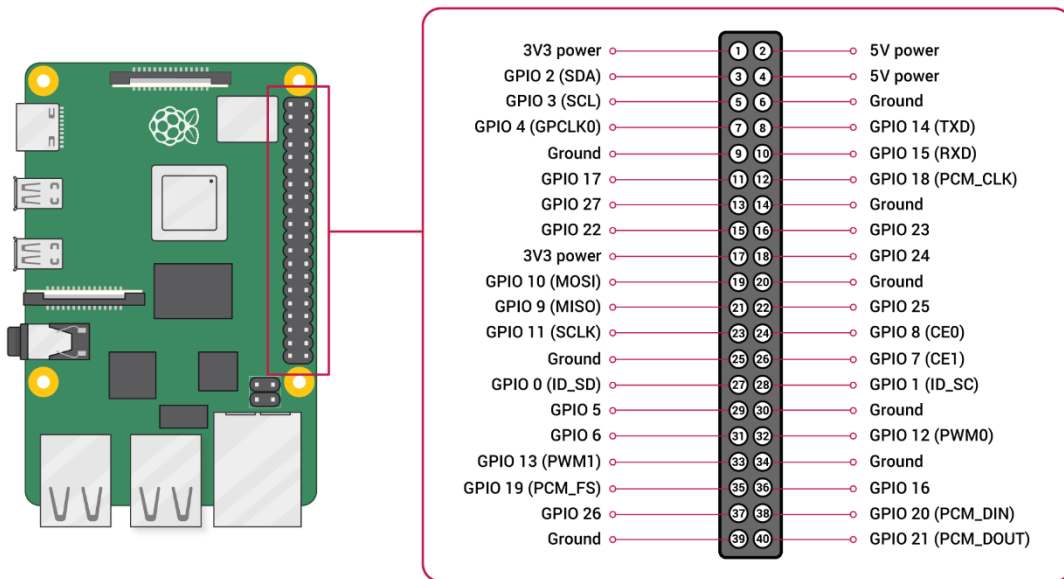
Die Funktionen «**initWebhouse()**» und «**closeWebhouse()**» erlauben die Steuerungsumgebung des Webhauses zu initialisieren und zu schliessen. Sobald die Steuerungsumgebung erfolgreich initialisiert wurde, können die anderen Steuerungsfunktionen jederzeit aufgerufen werden.

Das Makro «**#define PWM**» erlaubt festzulegen, dass die PWM-Schnittstelle für die Helligkeitsregelung der beiden Lampen verwendet werden sollte. **Achtung:** Wenn das Programm mit diesem Makro kompiliert wird, muss es systematisch im Superuser-Modus wie folgt gestartet werden:

**sudo ./Template**

Die PWM-Schnittstelle funktioniert nämlich nur im Superuser-Modus.

Die anderen Makros geben an, wie die Input- und Outputpins auf dem Raspberry Pi Kit verwendet werden. Die Nummerierung dieser Pins entspricht der Nummerierung des Models «Raspberry Pi 3 model B».



## 7.1 Installieren der Peripherie-Bibliothek « bcm2835.h »

Die Bibliothek "[Webhouse.h](#)" basiert auf den API-Funktion der Peripherie-Bibliothek «bcm2835.h». Es könnte sehr wahrscheinlich Sein, dass Sies diese Bibliothek auf Ihrem Raspberry-Pi-Kit noch nicht vorhanden ist. In diesem Fall müssen Sie zuerst irgendeinen Temporär-Ordner wählen. Anschliessend können die Bibliothek **bcm2835.h** mit den folgenden Kommandozeile-Befehle innerhalb von diesem Ordners installieren:

```
wget http://www.airspayce.com/mikem/bcm2835/bcm2835-1.68.tar.gz
tar xvfz bcm2835-1.68.tar.gz
cd bcm2835-1.68
./configure
make
sudo make install
```

## 7.2 Projektcompilation

Das Kompilieren des Projekts kann mit dem Linux-Befehl **make** im Projektverzeichnis durchgeführt werden. Dieser Befehl startet das Kompilieren entsprechend dem Kompilierungsskript **Makefile**. Dieser Skript enthält die folgenden direktiven:

```
Template.o: main.o Webhouse.o handshake.o
    gcc -o Template main.o Webhouse.o handshake.o base64.o sha1.o
    -lbcm2835 -lpthread -ljansson -lm

main.o: main.c Webhouse.h handshake.h
    gcc -c main.c

Webhouse.o: Webhouse.c Webhouse.h
    gcc -c Webhouse.c

handshake.o: handshake.c handshake.h base64.h sha1.h
    gcc -c handshake.c

base64.o: base64.c base64.h
    gcc -c base64.c

sha1.o: sha1.c sha1.h
```

```
gcc -c sha1.c
```

```
clean:
```

```
rm Template main.o Webhouse.o handshake.o base64.o sha1.o
```

## 8 jansson

### 8.1 Installieren der Bibliothek jansson

Es könnte sehr wahrscheinlich Sein, dass Sies die Bibliothek «**jansson.h**» auf Ihrem Raspberry-Pi-Kit noch nicht vorhanden ist. In diesem Fall müssen Sie zuerst irgendeinen Temporär-Ordner wählen. Anschliessend können die Bibliothek **jansson.h** mit den folgenden Kommandozeile-Befehle innerhalb von diesem Ordners installieren:

```
wget https://github.com/akheron/jansson/releases/download/v2.14/jansson-2.14.tar.gz
tar xvfz jansson-2.14.tar.gz
cd jansson-2.14
./configure
make
sudo make install
```