

Titel: MQTT mit Raspberry Pi und D1 mini

Autor(en): DI Karl HARTINGER

Datum: 10.7.2017 – 6.11.2017

Inhaltsverzeichnis

1	Allgemeines.....	3
1.1	MQTT.....	3
1.2	Aufgabenstellung „D1mini Taster und OLED-Anzeige“.....	3
2	Raspberry Pi als Access Point.....	4
2.1	Überblick.....	4
2.2	Installation.....	4
3	Raspberry Pi als Broker.....	8
3.1	Installation MQTT Broker und Clients (Mosquitto).....	8
3.2	Test der MQTT Installation.....	8
3.3	Mosquitto konfigurieren.....	9
3.4	Hilfreiches zu Mosquitto.....	9
4	MQTT-Client auf dem D1 mini.....	10
4.1	Möglichkeiten.....	10
4.2	Verwendung der Bibliothek pubsubclient	10
5	D1mini: Taster und OLED-Anzeige.....	13
5.1	Vorbereitung.....	13
5.2	Dimini -Taster.....	15
5.3	Dimini - OLED Anzeige.....	16
6	Raspberry C/C++ Client und Webseite.....	18
6.1	Aufgabenstellung.....	18
6.2	MQTT-Client zur Nachrichten-Abspeicherung (C-Datei).....	19
6.3	PHP-Webseite zum Anzeigen der Topics.....	23
6.4	Automatisches Starten des C-Clients.....	25
7	Webseite mit Echtzeit-Anzeige.....	26
7.1	Einleitung.....	26
7.2	Mosquitto für Websockets konfigurieren.....	26
7.3	MQTT-Webseite mit Real Time Anzeige.....	27
8	Anhang 1.....	28
8.1	Literatur.....	28
8.2	Datei PubSubClient.h	28
8.3	Datei PubSubClient.cpp.....	30
8.4	Datei D1_class_MqttClientKH.h	37
8.5	Datei utils_file.h	42
8.6	Datei utils_file.c	43
9	Anhang 2: RasPi einrichten.....	46
9.1	Betriebssystem auf SD-Card installieren.....	46
9.2	Grundeinstellungen vornehmen (nach dem ersten Start).....	46
9.3	Login und Update der Installation.....	48
9.4	Terminal-Schrift ändern (optional).....	48
9.5	Standard-User-Namen pi ändern (optional).....	48
9.6	Netzwerk konfigurieren beim Raspberry Pi 3.....	50
9.7	Arbeiten über das Netzwerk.....	53

9.8	Installation des 3,5“ HDMI Touch Screen LCD Treibers.....	54
10	Anhang 3: Weitere Raspi-Einstellungen.....	56
10.1	Eigene Autostart-Datei autostart.sh.....	56
10.2	Beispiel für Verzeichnisse und weitere Programme.....	56
10.3	Bearbeiten von Konfigurationsdateien.....	57
10.4	WiringPi installieren.....	58
10.5	I2C installieren.....	59
10.6	Test der Hardware mit eigenen Programmen.....	61
10.7	Netzwerk konfigurieren beim Raspberry Pi 1 und 2.....	63

Hinweis

Die Veröffentlichung dieses Skriptums erfolgt in der Hoffnung, dass es dem Leser von Nutzen sein wird, aber OHNE IRGEND EINE GARANTIE, sogar ohne die implizite Garantie der MARKTREIFE oder der VERWENDBARKEIT FÜR EINEN BESTIMMTEN ZWECK. Details finden Sie in der GNU General Public License.

1 Allgemeines

1.1 MQTT

Quelle: <https://de.wikipedia.org/wiki/MQTT> [9.7.2017]

MQTT (Message Queue Telemetry Transport) ist ein offenes Protokoll für Machine-to-Machine-Kommunikation (M2M), das die Übertragung von Daten in Form von Nachrichten zwischen Geräten ermöglicht. Es ist seit 2013 als Protokoll für das Internet der Dinge standardisiert und verwendet üblicherweise die Ports 1883 und 8883. Jede Nachricht besteht aus Header (2 Bytes), MQTT-Topic (Thema) und Payload (Dateninhalt).

Ein MQTT-System besteht aus einem Server („Broker“, wörtlich Makler) und verschiedenen Clients. Dabei werden Clients, die Daten zur Verfügung stellen, „Publisher“ und Clients, die Daten empfangen wollen, „Subscriber“ genannt. Die Datenübertragung erfolgt häufig über ein WLAN (mit HTTP-Protokoll).

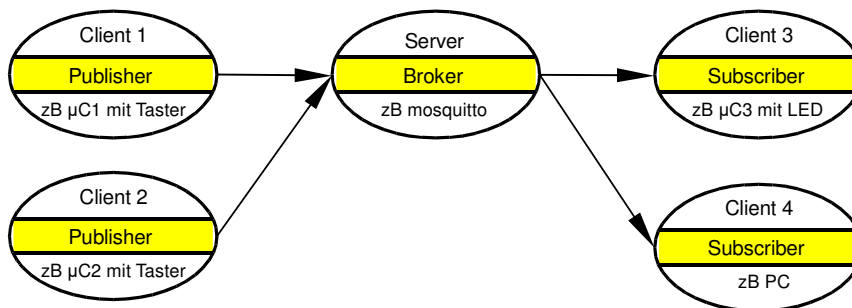


Bild 1: MQTT-System.

1.2 Aufgabenstellung „D1mini Taster und OLED-Anzeige“

Teil 1: Raspberry Pi

Ein Raspberry Pi (RPi) soll als Access Point ein WLAN zur Verfügung stellen, über das ein auf dem RPi installierter MQTT-Broker erreichbar ist. Als Broker soll Mosquitto verwendet werden.

Teil 2: D1 mini Clients

Wird bei einem D1mini eine Taste gedrückt, so soll mittels MQTT-Protokoll bei einem zweiten D1mini ein Zähler erhöht und der Wert auf einer OLED-Anzeige angezeigt werden. Weiters soll zur Kontrolle der Zählerstand zurück an den ersten D1mini gesendet werden.

2 Raspberry Pi als Access Point

2.1 Überblick

Die nachfolgende Installation geht davon aus, dass die RasPi-Grundinstallation (wie zB im Anhang beschrieben) durchgeführt ist. In neueren RasPi-Installationen erfolgt die Interface-Konfiguration über das Service `dhcdd`, allerdings gibt es da Probleme mit `dnsmasq`. Aus diesem Grund erfolgt die Konfiguration wie bisher über `/etc/network/interfaces`, wodurch `dhcpcd` automatisch deaktiviert wird.

Der RasPi als Access Point hat folgende Aufgaben:

- Ersatz für einen WLAN-Router
- Zugang zum Internet für die angeschlossenen Teilnehmer (Accesspoint-Software `hostapd`)
- DHCP- und DNS-Server (zB `dnsmasq`)
- ev. LAMP- und Samba-Server

Zugangsdaten zum WLAN

Netzwerk-Name: `Raspi11`
Netzwerk-Schlüssel: `Raspi11!!`
Verschlüsselung: `WPA2`

Anmerkung: der Chip BCM43438 wird vom open-source Treiber `brcmfmac` unterstützt.

2.2 Installation

Quelle: <https://frillip.com/using-your-raspberry-pi-3-as-a-wifi-access-point-with-hostapd/>
[25.10.2017]

Installation der erforderlichen Packages

```
~ $ sudo apt-get update
~ $ sudo apt-get upgrade
~ $ sudo apt-get install dnsmasq hostapd
```

Statische IP für Interface `wlan0`

Da das Interface `wlan0` eine statische Adresse benötigt, darf man nicht über das WLAN mit dem Raspi verbunden sein. Daher Tastatur + Bildschirm oder LAN + putty verwenden!

- (1) `eth0` und `wlan0` mit einer statischen IP versehen:

```
~ $ sudo nano /etc/network/interfaces
```

Am Ende der Datei folgendes einfügen:

```
auto eth0
iface eth0 inet dhcp
iface eth0 inet static
address    192.168.0.11
netmask    255.255.255.0
network    192.168.0.0
```

```

broadcast 192.168.0.255
gateway   192.168.0.1

auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
address   192.168.1.1
netmask   255.255.255.0
network   192.168.1.0
broadcast 192.168.1.255
#wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

```

Durch das # Zeichen (Doppelkreuz, Nummernzeichen oder hash) wird die wpa-conf-Zeile unwirksam gemacht (Kommentar).

Speichern und beenden durch <Strg>o <Enter> <Strg> x

- (2) In der Konfigurationsdatei zum Service dhcpcd einen eventuellen Eintrag für eth0 (und wlan0) auskommentieren:

```
~ $ sudo nano /etc/dhcpcd.conf
```

Zeichen # setzen:

```

#---added 2017-10-27---
#denyinterfaces wlan0

#-----statische IP-----
#interface eth0
#static ip_address=192.168.0.55/24
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1

```

Access-Point-Software hostapd konfigurieren

- (1) Neue Konfigurationsdatei für hostapd erstellen

```
~ $ sudo nano /etc/hostapd/hostapd.conf
```

Folgendes hineinschreiben:

```

# name of the WiFi interface we want to use
interface=wlan0

# Use the nl80211 driver with the brcmfmac driver
driver=nl80211

# This is the name of the network (change to your values!)
ssid=Raspi11

# Use the 2.4GHz band
hw_mode=g

# Use channel 6
channel=6

# Enable 802.11n
ieee80211n=1

# Enable WMM
wmm_enabled=1

```

```
# Enable 40MHz channels with 20ns guard interval
ht_capab=[HT40][SHORT-GI-20][DSSS_CCK-40]

# Accept all MAC addresses
macaddr_acl=0

# Use WPA authentication
auth_algs=1

# Require clients to know the network name
ignore_broadcast_ssid=0

# Use WPA2
wpa=2

# Use a pre-shared key
wpa_key_mgmt=WPA-PSK

# Network password (passphrase)
wpa_passphrase=Raspi11!!

# Use AES, instead of TKIP
rsn_pairwise=CCMP
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

(2) Test der Konfiguration

```
~ $ sudo /usr/sbin/hostapd /etc/hostapd/hostapd.conf
Configuration file: /etc/hostapd/hostapd.conf
Failed to create interface mon.wlan0: -95 (Operation not supported)
wlan0: Could not connect to kernel driver
Using interface wlan0 with hwaddr xx:xx:xx:xx:xx:xx and ssid "Raspi11"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
```

Das Netzwerk **Raspi11** ist zwar zu sehen, man kann sich aber noch nicht einloggen, da man keine IP erhält.

Abbrechen mit <Strg> c

(3) Konfigurationsdatei für hostapd beim Start verfügbar machen:

```
~ $ sudo nano /etc/default/hostapd
```

Unter der Zeile #DAEMON_CONF=" " einfügen:

```
DAEMON_CONF="/etc/hostapd/hostapd.conf"
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

DHCP- und DNS-Server dnsmasq konfigurieren

(1) Original-Konfigurationsdatei sichern und neue Konfigurationsdatei anlegen:

```
~ $ sudo mv /etc/dnsmasq.conf /etc/dnsmasq.conf.orig
~ $ sudo nano /etc/dnsmasq.conf
```

Inhalt der Datei

```
interface=wlan0          # Use interface wlan0
listen-address=192.168.1.1 # Explicitly specify the address to listen on
bind-interfaces          # Bind to the interface to make sure
                          # we aren't sending things elsewhere
server=8.8.8.8           # Forward DNS requests to Google DNS
```

```
domain-needed          # Don't forward short names
bogus-priv             # Never forward addresses in
                      # the non-routed address spaces.
dhcp-range=192.168.1.150,192.168.1.199,12h # Assign IP addresses in this
                      # range with a 12 hour lease time
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

- (2) Der dnsmasq Server beim Systemstart automatisch starten:

```
~ $ sudo service dnsmasq start
~ $ sudo update-rc.d dnsmasq enable
```

IP4 forwarding ermöglichen

- (1) IP4 forwarding einschalten

```
~ $ sudo nano /etc/sysctl.conf
```

Entfernen des # Zeichens vor der Zeile mit

```
net.ipv4.ip_forward=1
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

- (2) Netzwerkadressübersetzung (NAT) zwischen wlan0 und eth0 einrichten:

```
~ $ sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
~ $ sudo iptables -A FORWARD -i eth0 -o wlan0 -m state --state
RELATED,ESTABLISHED -j ACCEPT
~ $ sudo iptables -A FORWARD -i wlan0 -o eth0 -j ACCEPT
```

- (3) Abspeichern der NAT-Regeln in der Datei /etc/iptables.ipv4.nat:

```
~ $ sudo sh -c "iptables-save > /etc/iptables.ipv4.nat"
```

- (4) Aufruf der NAT-Regeln bei jedem Systemstart:

```
~ $ sudo nano /etc/rc.local
```

VOR der Zeile exit 0 folgende Zeilen einfügen:

```
# -----2017-10-27-----
iptables-restore < /etc/iptables.ipv4.nat
```

Test des Access-Points

- (1) Raspberry neu starten

```
~ $ sudo reboot
```

- (2) Im Terminal (oder Terminal-Programm) den Internetzugang testen

```
~ $ ping 8.8.8.8 -c 4
```

- (3) Auf einem Laptop, der nicht über LAN mit dem Internet verbunden ist, etc. das WLAN-Netzwerk Rasp11 auswählen und den Schlüssel Rasp11!! eingeben.

- (4) Im Browser auf dem Laptop zB die Webseite von Google aufrufen:

```
http://www.google.com
```

3 Raspberry Pi als Broker

Quelle: <http://www.instructables.com/id/Installing-MQTT-BrokerMosquitto-on-Raspberry-Pi/>

3.1 Installation MQTT Broker und Clients (Mosquitto)

Mosquitto installieren

```
~ $ sudo apt-get install mosquitto
```

Mosquitto Clients installieren (Publisher und Subscriber)

```
~ $ sudo apt-get install mosquitto-clients
```

3.2 Test der MQTT Installation

Zum Testen der MQTT Installation muss zuerst der Broker gestartet werden:

```
~ $ sudo /etc/init.d/mosquitto start
[ ok ] Starting mosquitto (via systemctl): mosquitto.service.
```

Als Nächstes wird ein Datenempfänger (Subscriber) eingerichtet, der auf Nachrichten des Typs „Test1“ hört:

```
~ $ mosquitto_sub -d -t Test1
Client mosqsub/3152-raspberryp sending CONNECT
Client mosqsub/3152-raspberryp received CONNACK
Client mosqsub/3152-raspberryp sending SUBSCRIBE (Mid: 1, Topic: Test1, QoS: 0)
Client mosqsub/3152-raspberryp received SUBACK
Subscribed (mid: 1): 0
```

Jetzt soll ein Datensender (Publisher) eine Nachricht senden. Zu diesem Zweck muss ein zweites Konsolenfenster (zB durch Drücken von <Alt><F2>) oder durch eine zweite Datenverbindung mittels putty geöffnet werden:

```
~ $ mosquitto_pub -d -r -t Test1 -m 'Hallo vom Publisher!'
Client mosqpub/3194-raspberryp sending CONNECT
Client mosqpub/3194-raspberryp received CONNACK
Client mosqpub/3194-raspberryp sending PUBLISH (d0, q0, r0, m1, 'Test1', ... (20 bytes))
Client mosqpub/3194-raspberryp sending DISCONNECT
```

Sobald man die Nachricht abschickt, erscheint sie im Subscriber-Fenster :)

Die Option -d -bedeutet „Enable debug messages.“, dh. alle Meldungen werden ausgegeben. Dies bewirkt auch, dass jede Minute die PINGREQ-Nachricht angezeigt wird. Möchte man nur die Nachrichten sehen, so muss man -d weglassen ;)

Der Schalter -r bewirkt, dass die letzte Nachricht im Broker zwischengespeichert wird und so einem Client „nachgeschickt“ werden kann, wenn dieser gerade offline war.

Eine detaillierte Dokumentation zu Mosquitto findet man unter <http://mosquitto.org/man/>

3.3 Mosquitto konfigurieren

*** Passwort verwenden? ***

3.4 Hilfreiches zu Mosquitto

Broker stoppen

```
~ $ sudo /etc/init.d/mosquitto stop
[ ok ] Stopping mosquitto (via systemctl): mosquitto.service.
```

Broker starten

```
~ $ sudo /etc/init.d/mosquitto start
[ ok ] Starting mosquitto (via systemctl): mosquitto.service.
```

Alle Topics und deren Wert anzeigen

```
~ $ mosquitto_sub -t "#" -v
Test1 D1mini message #1971
button 1
counter 21
```

-t "#" bedeutet alle Topics und deren Untertopics, -v zeigt das Topic mit an.

Prüfen, ob Mosquitto läuft

```
~ $ systemctl status mosquitto.service
```

Im Normalfall wird der MQTT-Server nach der Installation so eingerichtet, dass er bei jedem Neustart automatisch gestartet wird. Sollte dies nicht der Fall sein, kann man den Startbefehl Mosquitto bei jedem Systemstart automatisch starten.
Siehe Kapitel „Autostart-Datei“ im Anhang.

4 MQTT-Client auf dem D1 mini

4.1 Möglichkeiten

Für den D1 mini gibt es eine Reihe von Implementierungen von MQTT im Internet:

<https://github.com/knolleary/pubsubclient>

<https://github.com/256dpi/arduino-mqtt>

https://github.com/tuanpmt/esp_mqtt

http://lazyzero.de/elektronik/esp8266/dht_deepsleep/start

https://github.com/adafruit/Adafruit_MQTT_Library

Im Folgenden wird die Verwendung der Bibliothek `pubsubclient` gezeigt.

4.2 Verwendung der Bibliothek `pubsubclient`

Installation

Zuerst muss die Bibliothek installiert werden. Dies geschieht in der Arduino-Oberfläche mit Sketch – Bibliothek einbinden – Bibliotheken verwalten... Eingabe von „PubSubClient“ [Installieren]

Eine vollständige API-Dokumentation findet man unter <https://pubsubclient.knolleary.net/api.html> [18.7.2017]

Alternativ dazu könnte man auch den Quellcode der Dateien herunterladen und zB in einem Unterverzeichnis `libs` ablegen. In diesem Fall muss die Include-Anweisung statt `<>` Hochkomma `""` enthalten:

```
#include "libs/PubSubClient.h"
```

- Vorteil: Eigene Weiterentwicklung der Bibliothek möglich.
Wird die Original-Bibliothek geändert oder gelöscht, kann das Programm weiterhin kompiliert werden.
- Nachteile: Jedes Projekt muss eine Kopie der Bibliotheksdateien enthalten.
Verbesserungen der Bibliothek müssen händisch in allen Projekten nachgezogen werden.

Test

Das folgende Programm entspricht dem Testprogramm https://github.com/knolleary/pubsubclient/tree/master/examples/mqtt_esp8266 [18.7.2017] mit kleinen Adaptierungen.

Beim Start des Publishers (D1mini) wird die Nachricht „(Re)connect: Hello from D1mini :)“ gesendet, danach wird alle 5 Sekunden unter dem Topic „Test1“ die Nachricht „D1mini message #.“ (mit fortlaufender Nummer) an den Broker gesendet. Weiters wird die Nachricht zurückgelesen und über die serielle Schnittstelle (mit 9600Bd) angezeigt.

```

//_____D1_Ex35_mqtt_pubsub1.ino_____170718-170718_____
// Basic ESP8266 MQTT example
// This sketch demonstrates the capabilities of the pubsub
// library in combination with a D1mini or a ESP8266 board.
//
// It connects to a MQTT server then:
// - publishes "D1mini message #.." to the topic "Test1"
//   every 5 seconds
// - subscribes to the topic "Test1", printing out any message
//   (String) it receives.
//
// It will reconnect to the server if the connection is lost
// using a blocking reconnect function.
// See the 'mqtt_reconnect_nonblocking' example for how to
// achieve the same result without blocking the main loop.
//

#include <ESP8266WiFi.h>
#include <PubSubClient.h>

//.....Update these with values suitable for your network.....
const char* ssid = ".....";
const char* password = ".....";
const char* mqtt_server = "192.168.x.x";

#define MESSAGE_MAXLEN      127
WiFiClient espClient;
PubSubClient client(espClient);
long lastMsg = 0;
char msg[1+MESSAGE_MAXLEN];
int value = 0;

//_____connect to the WiFi network_____
void setup_wifi()
{
  delay(10);
  randomSeed(micros());           // start random numbers
  //-----try to connect to WLAN-----
  Serial.println("\nConnecting to "+String(ssid));
  WiFi.begin(ssid, password);
  int i=50;
  while (WiFi.status() != WL_CONNECTED)
  {
    delay(200);
    Serial.print(".");
    i--; if(i<0) { i=50; Serial.println(); }
  }
  //-----success-----
  Serial.print("\nConnected! IP address is ");
  Serial.println(WiFi.localIP());
}

//_____process incoming message_____
void callback(char* topic, byte* payload, unsigned int length)
{
  Serial.print("Message for topic ");
  Serial.print(topic);
  Serial.print(": ");
  for (int i = 0; i < length; i++)
  {
    Serial.print((char)payload[i]);
  }
  Serial.println();
}

```

```

//_____check for connect, loop until reconnected_____
void reconnect()
{
  //-----Loop until reconnected-----
  while (!client.connected())
  {
    Serial.println("Attempting MQTT connection...");
    //-----Create a random client ID-----
    String clientId = "D1mini_Client-";
    clientId += String(random(0xffff), HEX);
    //-----Attempt to connect-----
    if (client.connect(clientId.c_str()))
    {
      Serial.println(clientId+" connected.");
      //-----Once connected, publish an announcement-----
      client.publish("Test1", "(Re)connect: Hello from D1mini :)");
      //.....and resubscribe.....
      client.subscribe("Test1");
    }
    else
    {
      Serial.print("failed, client state rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);          // Wait 5 seconds before retrying
    }
  }
}

//_____setup Serial, WLAN and MQTT clients_____
void setup()
{
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
  client.setCallback(callback);
}

//_____main loop_____
void loop()
{
  if (!client.connected()) reconnect();
  client.loop();
  long now = millis();
  if (now - lastMsg > 5000)
  {
    lastMsg = now;
    value++;
    snprintf(msg, MESSAGE_MAXLEN, "D1mini message #%ld", value);
    Serial.println("Publish message: "+String(msg));
    client.publish("Test1", msg, true);          // true=retain
  }
}

```

5 D1mini: Taster und OLED-Anzeige

5.1 Vorbereitung

Aufgabenstellung

Wird bei einem D1mini eine Taste gedrückt, so soll mittels MQTT-Protokoll bei einem zweiten D1mini ein Zähler erhöht und der Wert auf einer OLED-Anzeige angezeigt werden. Weiters soll zur Kontrolle der Zählerstand zurück an den ersten D1mini gesendet werden.

Benötigte Hardware

- ▶ Raspberry Pi als Broker (Siehe Kap. 2)
- ▶ D1mini mit Taste an D3 als Publisher (Topic „button/02“) und Subscriber (Topic „button/02“)
- ▶ D1mini mit OLED als Subscriber (Topic „button“) und Publisher (Topic „button/xx/ok“)

Klasse `MqttClientKH`

Die Klasse `MqttClientKH` erweitert die Klasse `PubSubClient` zur einfachen Verwendung auf dem D1mini. Die Codierung findet sich im Anhang oder zB auf

https://github.com/khartinger/D1mini_oop/tree/master/D1_oop19_mqtt_V2_buttonD3

<pre>class MqttClientKH (extends PubSubClient) + MqttClientKH(char* ssid, char* pwd, char* mqtt_server, int port) + int getNumSub() + int getNumPub() + void clrSubscribe() + void clrPublish() + void setClientName(String sName) + String getClientName() -- Methoden zum Aufsetzen der WLAN- und MQTT- Verbindung --- + bool setup_wifi() + bool reconnect() + bool isConnected() + bool sendPubSubTopics() + void printPubSubTopics2Serial()</pre>	<p>Name der Klasse (und Datei)</p> <p>Vorgabewerte: <code>mqtt_server="localhost"</code>, <code>port=1883</code></p> <p>Anzahl der abonnierten Nachrichten (Topics)</p> <p>Anzahl der zur Veröffentlichung angemeldeten Nachrichten bei Verbindungsverlust</p> <p>Anzahl abonnierten Nachrichten = Null</p> <p>Anzahl zu veröffentlichender Nachrichten = Null</p> <p>Client Namen setzen</p> <p>Client Namen holen</p> <p>(Versuch zur) Verbindung mit dem WLAN, true bei Erfolg</p> <p>Auf Verbindung prüfen. Wenn nein: Wiederverbindung versuchen</p> <p>Besteht eine MQTT-Verbindung? (Wenn nein: Wiederverbindung nach <code>MQTT_RECONNECT_MS</code> Millisekunden)</p> <p>MQTT: Sende alle PubSub-Topics zum Broker. Funktion wird von <code>reconnect()</code> aufgerufen</p> <p>Ausgabe der Topics über die serielle Schnittstelle (zu Testzwecken)</p>
--	---

--- Methoden zur Bearbeitung der MQTT Topics ---

```

+ bool addSubscribe(String topic)
+ bool delSubscribe(String topic)
+ bool addPublish(String topic, String
payload, bool retain)
+ bool delPublish(String topic)
+ void publishString(String topic, String
payload)
+ void publishString(String topic, String
payload, bool retained)

+ int setSubscribe(String aTopicSub[], int
num)
+ int setPublish(String aTopicPub[],
String aPayload[], int num)
+ void subscribeString(String topic)

```

(String) Topic zur subscribe-Liste hinzufügen

Topic aus subscribe-Liste entfernen

(String) Topic zur publish-Liste hinzufügen

Topic aus publish-Liste entfernen

String-Wert veröffentlichen (ohne in die Liste aufzunehmen)

String-Wert veröffentlichen (ohne in die Liste aufzunehmen). retained=true: Broker soll sich Wert merken.

Feld von Topics zur subscribe-Liste hinzufügen

Feld von Topics zur publish-Liste hinzufügen

String-Topic direkt beim Broker anmelden (ohne Aufnahme in subscribe-Liste)

--- Eigenschaften ---

```

~ char ssid_[SSID_SIZE+1]
~ char pass_[PASS_SIZE+1]
~ char mqtt_[MQTT_SIZE+1]
~ int port_
~ String aTopicSub_[TOPIC_MAX]

~ String aTopicPub_[TOPIC_MAX]
~ String aPayloadPub_[TOPIC_MAX]

~ bool aRetainPub_[TOPIC_MAX]
~ int numSub_
~ int numPub_
~ WiFiClient d1miniClient
~ String sClientName
~ long millis_lastConnected

```

SSID_SIZE = 20

PASS_SIZE = 20

Name des MQTT-Servers (Länge MQTT_SIZE=20)

MQTT-Port (Vorgabe 1883)

Liste (array) mit allen abonnierten Topics (TOPIC_MAX=8)

Liste (array) mit allen Veröffentlichungs-Topics

Liste (array) mit den Werten Veröffentlichungs-Topics

Liste (array) mit den Retain-Werten (true | false)

Anzahl der subscribe-Topics in der Liste

Anzahl der publish-Topics in der Liste

WLAN Client for MQTT

Name des MQTT Clients

Zeitpunkt des letzten Verbindungsaufbaus in Millisekunden

Erstellen einer D1mini MQTT Applikation

- Klasse MqttClientKH einbinden
#include "D1_class_MqttClientKH.h"
- Objekt der Klasse MqttClientKH definieren
MqttClientKH client("..ssid..", "..password..", "mqtt server name");
Nicht vergessen: WLAN-Daten anpassen!
- Callback Function für die angeforderten Nachrichten erstellen. Diese Funktion wird von jedem unterzeichneten Topic aufgerufen.
void callback(char* topic, byte* payload, unsigned int length)
- In der Funktion setup() das MQTT Setup durchführen:
client.addSubscribe("topic");
client.addPublish("topic", "startvalue");
client.setCallback(callback);
client.reconnect();
- In der Hauptschleife loop()
Aufruf von **client.isConnected()** um alle zyklischen Arbeiten durchzuführen (WLAN, MQTT und automatisches Neuverbinden bei Verbindungsabbruch).

5.2 Dimini -Taster

Dieses Programm-Beispiel (Sketch) verbindet sich über WLAN mit einem MQTT Server und macht folgendes:

- Beim Einschalten (Power-On) wird die Nachricht (message) "button/02" mit dem Inhalt (payload) -1 gesendet.
- Wird der Taster D3 gedrückt, so wird dies über die serielle Schnittstelle angezeigt und der Wert 1 unter dem Topic "button/02" veröffentlicht.
- Alle Nachrichten (messages) mit dem Topic "button/02" werden empfangen und über die serielle Schnittstelle angezeigt.

Nicht vergessen: Die WLAN-Daten an das eigene Netzwerk anpassen in der Zeile:

```
MqttClientKH client("..ssid..", "..password..", "mqtt server name");
```

Hardware:

- WeMos D1 mini
- 1-Button Shield D3

```
//_____D1_oop19_mqtt_V2_buttonD3.ino_____170721-171029_____
//
// Simple MQTT-Example (needs a broker!)
// * On Power-On a message "button/02" with payload -1 is sent.
// * When button D3 is pressed, this is printed to Serial and
//   the value 1 is published under topic "button/02".
// * All messages received by topic "button/02" are printed
//   to Serial.
//
// Hardware:
// (1) WeMos D1 mini
// (2) 1-Button shield
#include "D1_class_MqttClientKH.h"
#include "D1_class_Din.h"
#define BUTTON_NAME "button/02"
Din button_(D3);
MqttClientKH client("..ssid..", "..password..", "mqttservername");

//_____process all subscribed incoming messages_____
void callback(char* topic, byte* payload, unsigned int length)
{
//-----convert topic and payload to String, print to Serial-----
String sTopic=String(topic);
String sPayload="";
for (int i=0; i<length; i++) sPayload+=(char)payload[i];
Serial.print("Message received for topic "+sTopic+"=");
Serial.println(sPayload);
}

//_____setup Serial, WLAN and MQTT clients_____
void setup()
{
Serial.begin(9600); Serial.println("");
//-----setup mqtt-----
client.setClientName(String(BUTTON_NAME));
client.addSubscribe(String(BUTTON_NAME)+"/#");
client.addPublish(String(BUTTON_NAME), "-1");
client.setCallback(callback);
client.reconnect();
}
```

```
//_____main loop_____
void loop()
{
  if(client.isConnected())
  {
    if(button_.is_falling_edge())
    {
      client.publish(BUTTON_NAME, "1", false); // true=retain
      Serial.println("\n***** button pressed! *****");
    }
  }
}
```

5.3 Dimini - OLED Anzeige

Dieses Programm (Sketch) verbindet sich mit dem MQTT-Server über das WLAN und macht folgendes

- Der D1 mini abonniert (subscribes) alle Nachrichten vom Typ "button/#".
- Wird die Nachricht "button/xx" mit dem Inhalt (payload) 1 empfangen, wird der Zähler um 1 erhöht, der Wert am OLED-Shield angezeigt und an die serielle Schnittstelle übertragen (xx ist der Wert des Tasters von 00 bis 99). Eine Payload von 0 setzt den Zähler zurück (auf 0).
- Danach wird eine Nachricht "button/xx/ok" mit dem Zählerstand als Inhalt veröffentlicht.

Nicht vergessen: Die WLAN-Daten an das eigene Netzwerk anpassen in der Zeile:

```
MqttClientKH client("..ssid..", "..password..", "mqtt server name");
```

Anmerkung: Zur Darstellung der Zeichen am OLED-Shield wird die Klasse [D1_class_DisplayKH.h](#) verwendet.

Hardware:

- ▶ WeMos D1 mini
- ▶ OLED Shield: SSD1306, 64x48 pixel, I2C

```
//_____D1_oop19_mqtt_V2_counter_oled1.ino_____170721-171029_____
//
// Simple MQTT-Example (needs a broker!)
// * When message "button/xx" (xx=number of button) with
//   payload 1 is received, a counter is incremented,
//   the value is displayed on oled shield and
//   printed to Serial. (Payload 0 resets the counter.)
// * After that, a message "counter/xx/ok" is published with
//   counter value as payload.
//
#include "D1_class_MqttClientKH.h"
#include "D1_class_DisplayKH.h"
DisplayKH display_; //
MqttClientKH client("..ssid..", "..password..", "mqttservername");

unsigned long requestCounter=0;

//_____display counter value on oled_____
void display_counter(String sCounter)
```



```

{
  display_.screen13(1,sCounter,'c',false);
  display_.screen13(2,"MQTT");
  display_.screen13(3,"counter");
  display_.screen13(4,"171029 KH");
  display_.display();
}

//_____process all subscribed incoming messages_____
void callback(char* topic, byte* payload, unsigned int length)
{
  String sTopic=String(topic);
  //-----Convert payload to String, print message to Serial-----
  String sPayload="";
  for (int i=0; i<length; i++) sPayload+=(char)payload[i];
  Serial.print("Message received. Topic '"+sTopic+"', ");
  Serial.println("payload='"+sPayload+"'");
  if(sTopic.indexOf("/ok")==9) Serial.println("");
  //-----process message "button/xx"-----
  String sButtonNr="";
  if(sTopic.startsWith("button"))
  {
    if((sTopic.charAt(6)=='/') && (sTopic.length()==9))
    {
      if(isDigit(sTopic[7]) && isDigit(sTopic[8]))
      {
        sButtonNr=sTopic.substring(7,9);
        if(payload[0]=='0') requestCounter=0;
        if(payload[0]=='1') requestCounter++;
        String sCounter=String(requestCounter);
        Serial.println("requestCounter = "+sCounter);
        display_counter(sCounter);
        client.publishString("button/"+sButtonNr+"/ok", sCounter, true);
      }
    }
  }
}

//_____setup Serial, WLAN and MQTT clients_____
void setup()
{
  Serial.begin(9600); Serial.println("");
  display_counter(String(requestCounter));
  //-----setup mqtt-----
  client.setCallback(callback);
  client.addSubscribe("button/#");
  client.reconnect();
}

//_____main loop_____
void loop()
{
  client.isConnected();
  delay(20);
}

```

6 Raspberry C/C++ Client und Webseite

6.1 Aufgabenstellung

Mit Hilfe der Bibliothek `libmosquitto` soll in C für das Raspberry Pi ein MQTT-Client als Kommandozeilenprogramm geschrieben werden, der in einer Endlosschleife alle Meldungen empfängt (Dateiname `rpi_mqtt_sub2file.c`) und in Dateien abspeichert. Der Name oder die IP des Brokers kann als Parameter angegeben werden (Default: 192.168.1.1)

Für jedes Topic wird eine eigene Datei mit dem Topic-Namen erzeugt, die den Inhalt der letzten Meldung enthält. Enthält ein Topic Leerzeichen, so werden diese durch den Unterstrich (`_`) ersetzt, Schrägstriche `/` werden durch Et-Zeichen `&` ersetzt.

Die Dateien sollen im Verzeichnis `/var/www/html/mqtt/data` gespeichert werden, das dem User `root` gehören und für alle User zum Schreiben, Lesen und Ausführen zugänglich sein soll.

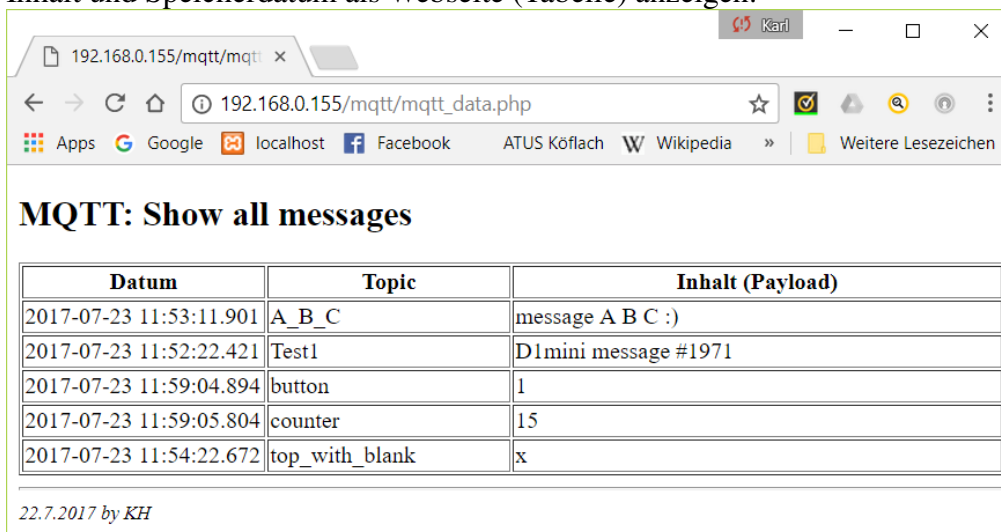
Beispiel

Im Verzeichnis `/var/www/html/mqtt/data` können zB folgende Dateien gespeichert sein:

```
-rw-r--r-- 1 366 2017-07-23 14:44:42.868546050 +0200 _
-rw-r--r-- 1 17 2017-07-23 11:53:11.901837589 +0200 A_B_C
-rw-r--r-- 1 2 2017-07-23 11:59:04.894215568 +0200 button
-rw-r--r-- 1 3 2017-07-23 11:59:05.804221611 +0200 counter
-rw-r--r-- 1 21 2017-07-23 11:52:22.421498641 +0200 Test1
-rw-r--r-- 1 2 2017-07-23 11:54:22.672319922 +0200 top_with_blank
```

(Anzeige durch `ls -ghxG --full-time /var/www/html/mqtt/data`)

Eine PHP-Webseite (`mqtt_data.php`) soll nun auf diese Dateien zugreifen und alle Topics mit Inhalt und Speicherdatum als Webseite (Tabelle) anzeigen:



MQTT: Show all messages

Datum	Topic	Inhalt (Payload)
2017-07-23 11:53:11.901	A_B_C	message A B C :)
2017-07-23 11:52:22.421	Test1	D1mini message #1971
2017-07-23 11:59:04.894	button	1
2017-07-23 11:59:05.804	counter	15
2017-07-23 11:54:22.672	top_with_blank	x

22.7.2017 by KH

Technische Besonderheiten

C-Programm: Leerzeichen im Topic werden in Underline-Zeichen (`_`), Schrägstriche (`/`) in Et-Zeichen (`&`) umgewandelt.

PHP-Script: Die Anzeige des Inhaltsverzeichnis wird mit Hilfe des Linux-Befehls `ls -ghxG --full-time ./data > ./data/_` in eine Datei mit dem Namen `_` (Underline) umgeleitet und aus dieser Datei werden Zeile für Zeile die Informationen für Uhrzeit und Topic herausgeholt.

PHP-Script: Et-Zeichen (`&`) im Topic-Namen werden in Schrägstriche (`/`) (zurück) umgewandelt.

6.2 MQTT-Client zur Nachrichten-Abspeicherung (C-Datei)

Vorbereitung: C-Bibliothek herunterladen

```
~ $ sudo apt-get install libmosquitto-dev
```

Die Datei mosquitto.h wird automatisch nach /usr/include kopiert, damit sie systemweit zur Verfügung steht.

Dokumentation zu Libmosquitto: <https://mosquitto.org/man/libmosquitto-3.html>

Beschreibung aller Datenstrukturen und Funktionsaufrufe:

<https://mosquitto.org/api/files/mosquitto-h.html>

- (1) Arbeitsverzeichnis erstellen (falls dies noch nicht erfolgt ist), dahin wechseln und den Texteditor starten:

```
$ mkdir ~/src_c
$ mkdir ~/src_c/mqtt/
$ cd ~/src_c/mqtt/
$ nano ~/src_c/mqtt/rpi_mqtt_sub2file.c
```

- (2) C-Programm eingeben

Wichtig: Die IP-Adresse des Brokers in main() anpassen in der Zeile

```
char* host="192.168.1.1";
```

Inhalt der C-Datei:

```
//_____rpi_mqtt_sub2file.c_____170722-171106_____
// Subscribe all MQTT messages and copy them to files.
// (One message = one file)
// MQTT Broker : ip = _HOST_ (default 192.168.1.1)
// Directory : _PATH_ (default /var/www/html/mqtt/data/)
// Filename : topic (replacements: ' ' -> '_' and '/' -> '&')
// File content: payload
//
// gcc -o rpi_mqtt_sub2file rpi_mqtt_sub2file.c utils_file.c -lmosquitto
// needs sudo apt-get install libmosquitto-dev

#include <stdio.h> // printf, stdin, stdout
#include <mosquitto.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include "utils_file.h"
#define _HOST_ "192.168.1.1"
#define _PATH_ "/var/www/html/mqtt/data/"
bool prt=true; // true=printf, false=quiet

//_____print log message_____
void mosq_log_callback(struct mosquitto *mosq,
void *userdata, int level, const char *str)
{
    switch(level)
    {
        case MOSQ_LOG_DEBUG:
        case MOSQ_LOG_INFO:
        case MOSQ_LOG_NOTICE:
            break;
        case MOSQ_LOG_WARNING:
        case MOSQ_LOG_ERR:
        default:
            fprintf(stderr, "LOG-Message %i: %s\n", level, str);
    }
}

//_____connect info_____
void mosq_connect_callback(struct mosquitto *mosq,
void *userdata, int result)
{
}
```

```

if(!result)
{ //-----Subscribe all topics on successful connect-----
  mosquitto_subscribe(mosq, NULL, "#", 2);
  //mosquitto_subscribe(mosq, NULL, "$SYS/#", 2); //broker infos
} else {
  fprintf(stderr, "Connect failed\n");
}
}

//_____message call back routine_____
// parameter will be destroyed after return!
void mosq_msg_callback(struct mosquitto *mosq, void *userdata,
const struct mosquitto_message *message)
{
  int ret;
  char filename[FILENAME_MAX];
  //-----replace blank by _, / by &-----
  char* tmp=message->topic;
  while(*tmp)
  {
    if((*tmp)==' ') *tmp='_';
    if((*tmp)=='/') *tmp='&';
    tmp++;
  }
  //-----add path (no extension ;)-----
  sprintf(filename, "%s%s", _PATH_, message->topic);
  if(prt) printf("=> %s | %s\n", filename, message->payload);
  if(message->payloadlen)
    ret=writeline(filename, message->payload);
  else
    ret=writeline(filename, "");
  fflush(stdout);
  if(ret!=0)
    printf("Fehler %d beim Schreiben in die Datei %s\n",ret,filename);
}

//_____print help on stdout_____
void print_help()
{
  fprintf(stdout, "\nSubscribe all messages from MQTT broker and\n");
  fprintf(stdout, "copy them to directory %s\n", _PATH_);
  fprintf(stdout, "      filename = topic, file content = payload\n");
  fprintf(stdout, "      topic replacements: ' ' -> '_' and '/' -> '&'\n");
  fprintf(stdout, "Author: Karl Hartinger, 6.11.2017\n");
  fprintf(stdout, "Usage : rpi_mqtt_sub2file [-q] [ip]\n");
  fprintf(stdout, "      -q ... no output to stdout\n");
  fprintf(stdout, "      ip ... ip of broker (default %s)\n", _HOST_);
  fprintf(stdout, "Needs sudo apt-get install libmosquitto-dev\n");
}

//_____main program_____
int main(int argc, char *argv[])
{
  char host[128];
  int  port=1883;
  int  keepalive=60;
  bool clean_session=true;
  struct mosquitto *mosq = NULL;
  strcpy(host, _HOST_);
  //-----Are there any arguments?-----
  if(argc>3) { print_help(); return 10; };
  if(argc==1) print_help();
  //-----one arg: -q=no_printf or server name-----
  if(argc==2)
  {
    if(argv[1][0]=='-')
    {
      if(argv[1][1]=='h') { print_help(); return(11); }
      if(argv[1][1]=='q') prt=false;
    }
    else
    {
      strcpy(host, argv[1]);
    }
  }
}

```

```

    }
}
//-----two args: servername and quiet-----
if(argc==3)
{
    if(argv[1][0]=='-')
    {
        strcpy(host, argv[2]);
        if(argv[1][1]=='h') { print_help(); return(11); }
        if(argv[1][1]=='q') prt=false;
    }
    else
    {
        strcpy(host, argv[1]);
        if(argv[2][0]=='-')
        {
            if(argv[2][1]=='h') { print_help(); return(11); }
            if(argv[2][1]=='q') prt=false;
        }
    }
}
//-----init network and mqtt-----
mosquitto_lib_init();
mosq=mosquitto_new(NULL, clean_session, NULL);
if(!mosq)
{
    fprintf(stderr, "Error: Out of memory.\n");
    return(1);
}
mosquitto_log_callback_set(mosq, mosq_log_callback);
mosquitto_connect_callback_set(mosq, mosq_connect_callback);
mosquitto_message_callback_set(mosq, mosq_msg_callback);

if(prt) printf("Try to connect to mosquitto...\n");
while(mosquitto_connect(mosq, host, port, keepalive)!=MOSQ_ERR_SUCCESS)
{
    if(prt) { fputc('.', stdout); fflush(stdout); }
    sleep(1);
}
if(prt) printf(" connected!\n");
//-----endless loop-----
mosquitto_loop_forever(mosq, -1, 1);
//-----clean up-----
mosquitto_destroy(mosq);
mosquitto_lib_cleanup();
return 0;
}

```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

(3) Hilfsdateien bereitstellen

Falls sich die Hilfsdateien `utils_file.h` und `utils_file.c` (noch) nicht im Arbeitsverzeichnis befinden, müssen sie dorthin kopiert oder neu erstellt werden.

Neuerstellung durch

```
$ nano ~/src_c/mqtt/utils_file.h
```

Hineinkopieren des Inhalts (siehe Anhang)

Speichern und beenden durch <Strg>o <Enter> <Strg> x

und

```
$ nano ~/src_c/mqtt/utils_file.c
```

Hineinkopieren des Inhalts (siehe Anhang)

Speichern und beenden durch <Strg>o <Enter> <Strg> x

- (4) Datei kompilieren

Eingabe von

```
$ gcc -o rpi_mqtt_sub2file rpi_mqtt_sub2file.c utils_file.c -lmosquitto
```

- (5) Test: Programm starten (ergibt Anzeige der aktuellen Topics und deren Inhalte)

```
$ ./rpi_mqtt_sub2file
=> /var/www/html/mqtt/data/Test1 | D1mini message #1971
=> /var/www/html/mqtt/data/button | 1
=> /var/www/html/mqtt/data/counter | 15
=> /var/www/html/mqtt/data/top_with_blank | x
=> /var/www/html/mqtt/data/A_B_C | message A B C :)
```

Programmende mit <Strg>C

6.3 PHP-Webseite zum Anzeigen der Topics

Annahme: PHP ist auf dem Raspberry installiert - sonst installieren (siehe Anhang) oder durch

```
$ sudo apt-get install apache2
$ sudo apt-get install php
```

- (1) Vorbereitung: Erstellen eines Datenverzeichnisses, das für alle User verwendbar ist

```
$ sudo mkdir /var/www/html/mqtt
$ sudo mkdir /var/www/html/mqtt/data
$ sudo chown root /var/www/html/mqtt/data
$ sudo chmod 777 /var/www/html/mqtt/data
$ sudo chmod u+s /var/www/html/mqtt/data
```

- (2) Ins Arbeitsverzeichnis wechseln und den Texteditor starten

```
$ cd /var/www/html/mqtt/
$ sudo nano /var/www/html/mqtt/mqtt_data.php
```

- (3) PHP-Programm eingeben

```
<?php //mqtt_data.php
// 22.7.2017-1.11.2017 DI Karl Hartinger
define("_START_DATE_ROW_", 15);
define("_START_TOPIC_ROW_", 49);
define("_DATE_LEN_", 23); // max. 29
$s1="<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"\n";
$s1.=" \nhttp://www.w3.org/TR/html4/loose.dtd">\n";
$s1.="<html>\n<head>\n<title>".$title."</title>\n";
$s1.="<meta http-equiv=\"Content-Type\" content=\"text/html; charset=iso-8859-1\">\n";
$s1.="<meta name=\"viewport\" content=\"width=device-width; initial-scale=1.0;\" />\n";
$s1.="</head>\n";
//====html body=====
$s1.="<body>\n";
$s1.="<h2>MQTT: Show all messages</h2>\n";
//-----generate file with topic names-----
$aOutput=Array();
$aLines=Array();
$aTopic=Array();
$aDate=Array();
$aPayload=Array();
$pfad="./data";
$pfaddatei=$pfad."/_";
$cmd="ls -ghxG --full-time ./data > ./data/_";
$x=exec($cmd, $aOutput, $ret);
if($ret==0)
{
    //-----extract date, topic and payload from line(s)-----
    $aLines=getFileContent($pfaddatei);
    $anz=count($aLines);
    $anzTopic=0;
    for($i=0; $i<$anz; $i++)
    {
        $len=strlen($aLines[$i]);
        $iStartDate=strpos($aLines[$i], "20",_START_DATE_ROW_);
        $iStartTopic=strpos($aLines[$i], " ",_START_TOPIC_ROW_);
        if(($len>50)&&($iStartDate>0)&&($iStartTopic>0))
        {
            $topic=trim(substr($aLines[$i], $iStartTopic));
            if(($topic!="_")&&(strlen($topic)>0))
            {
                $aTopic[$anzTopic]=$topic;
                $aDate[$anzTopic]=substr($aLines[$i], $iStartDate,_DATE_LEN_);
                $aTemp=getFileContent($pfad."/".$topic);
                $aPayload[$anzTopic]=$aTemp[0];
                $anzTopic++;
            }
        }
    }
}
```

```

}
//-----show values as table-----
$sl.="Anzahl Topics: ".$anzTopic."<br>\n";
$sl.="<table border='1' width=100%><tr>";
$sl.="<th width='25%'>Datum</th>";
$sl.="<th width='25%'>Topic</th><th>Inhalt (Payload)</th>";
$sl.="</tr>\n";
for($i=0; $i<$anzTopic; $i++)
{
    $temp=str_replace("&", "/", $aTopic[$i]);
    $sl.="<tr><td>".$aDate[$i]."</td><td>";
    $sl.=$temp."</td><td>".$aPayload[$i]."</td></tr>\n";
}
$sl.="</table>\n";
}
else
{
    $sl.="Error: topic list not found!<br>\n";
}
//-----End of html body-----
$version="1.11.2017";
$autor  ="KH";
$sl.="<hr><i><small>".$version." by ".$autor;
$sl.="</small></i></body>\n</html>\n";
echo($sl);

//_____Dateiinhalt als String_____
// Rueckgabe: Array mit Dateiinhalt: 1 Zeile=String
function getFileContent($pfaddatei)
{
    $aa0=Array();
    $index=0;
    if(file_exists($pfaddatei))
    {
        $datei=fopen($pfaddatei, "r");
        while(!feof ($datei))
        {
            $zeile1=fgets($datei, 1024);
            $aa0[$index++]=$zeile1;
        }
        fclose($datei);
    }
    else
    {
        $aa0[0]="File not found";
        return $aa0;
    }
}

?>

```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

(4) Test der Webseite

Zuerst den Client starten

```
$ ~/src_c/mqtt/rpi_mqtt_sub2file
```

Aufruf der Webseite mqtt_data.php im Browser:

```
http://192.168.1.1/mqtt/mqtt_data.php
```

Veröffentlicht man jetzt eine Nachricht, so sollte diese im Brower (nach nochmaligem Aufruf der Seite bzw. Drücken von aktualisieren) sichtbar sein:

```
$ mosquitto_pub -r -t Test1 -m 'Hallo vom Publisher!'
```


6.4 Automatisches Starten des C-Clients

Freigabe des C-Clients für andere User

```
$ sudo cp ~/src_c/mqtt/rpi_mqtt_sub2file /usr/local/bin
$ sudo chown root /usr/local/bin/rpi_mqtt_sub2file
$ sudo chmod u+s /usr/local/bin/rpi_mqtt_sub2file
$ sudo chmod 777 /usr/local/bin/rpi_mqtt_sub2file
```

Soll der C-Client bei jedem Systemstart automatisch aktiviert werden, so kann dies zB durch das Erstellen eines Skripts erfolgen, das bei jedem Systemstart aufgerufen wird:

```
$ sudo nano /usr/local/bin/autostart.sh
```

Inhalt der Datei:

```
#!/bin/bash
#...Farbe der Schrift auf gelb ändern...
echo -e "\033[01;33m"
printf "____autostart.sh____23.7.2017____Karl Hartinger____\n"
printf "RTC-Zeit nach date kopieren (rtc2date)\n"
/usr/local/bin/rtc2date &
printf "MQTT Client cd starten (rpi_mqtt_sub2file)\n"
/usr/local/bin/rpi_mqtt_sub2file -q &
printf "_____\n"
#...Farbe der Schrift wieder auf weiß ändern...
echo -e "\033[00m"
exit 0
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Script ausführbar machen:

```
$ sudo chmod 777 /usr/local/bin/autostart.sh
```

Weiters muss das Script noch am Ende der Datei /etc/rc.local eingebunden werden

```
$ sudo nano /etc/rc.local
```

Vor exit 0 ergänzen:

```
#-----Aufruf eines Scripts mit eigenen Befehlen-----
/usr/local/bin/autostart.sh
exit 0
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Test durch Neustart des Raspberri Pi

```
$ sudo reboot
```

und Aufruf der Webseite mqtt_data.php im Browser:

```
http://192.168.1.1/mqtt/mqtt_data.php
```

Veröffentlicht man jetzt eine Nachricht, so sollte diese im Brower (nach nochmaligem Aufruf der Seite bzw. Drücken von aktualisieren) sichtbar sein:

```
$ mosquitto_pub -r -t Test1 -m 'Hallo vom Publisher!'
```

7 Webseite mit Echtzeit-Anzeige

7.1 Einleitung

Möchte man MQTT-Nachrichten auf einer Webseite „live“ anzeigen, so gibt es mehrere Möglichkeiten:

- ▶ Abspeichern der Nachrichten in einer Datei. Anzeige der Daten mittels Webseite und periodisches Update der Seite.
- ▶ Verwendung von WebSockets
- ▶ Weitere Möglichkeiten, zB
<http://blog.hekkers.net/2012/10/13/realtime-data-with-mqtt-node-js-mqtt-js-and-socket-io/>

7.2 Mosquitto für Websockets konfigurieren

Siehe auch: <http://blog.ithasu.org/2016/05/enabling-and-using-websockets-on-mosquitto/>

- (1) Aktuelle Mosquitto-Version für Raspbian Jessie herunterladen

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key

cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list

sudo apt-get update
sudo apt-get dist-upgrade
```

- (2) Mosquitto für Websockets einrichten

Dazu muss ein weiterer Port (zB 1884) definiert werden, der auf WebSocket-Anfragen wartet. Die Einstellungen können in einer neu zu erstellenden Datei durchgeführt werden:

```
$ sudo nano /etc/mosquitto/conf.d/socket.conf
```

Inhalt der Datei:

```
listener 1883
protocol mqtt
listener 1884
protocol websockets
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

- (3) Mosquitto neu starten

```
$ sudo service mosquitto restart
```

7.3 MQTT-Webseite mit Real Time Anzeige

- (1) Als Vorbereitung legt man ein neues Verzeichnis auf dem Raspberry Pi an:

```
$ md /var/www/html/mqtt
```

- (2) Herunterladen der JavaScript-Datei `mqtt.min.js`, mit der der Zugriff auf die Nachrichten des Brokers ermöglicht wird:

Quelle: <https://unpkg.com/mqtt/dist/mqtt.min.js>

(bzw. <https://unpkg.com/mqtt@2.9.1/dist/mqtt.min.js>)

Abspeichern der Datei im zuvor erstellten Verzeichnis.

```
$ cd /var/www/html/mqtt
$ sudo wget https://unpkg.com/mqtt/dist/mqtt.min.js
```

- (3) Erstellen der Webseite

```
$ sudo nano /var/www/html/mqtt/mqtt_life.html
```

Inhalt der Datei:

```
<html>
<head>
  <title>MQTT Test1</title>
</head>
<body>
<script src="./mqtt.min.js"></script>
<script>
  var client = mqtt.connect({port: 1884 }) // or add a ws:// url here
  client.subscribe("Test1")

  client.on("message", function (topic, payload) {
    var element = document.getElementById("output");
    element.value = [topic, payload].join(": ");
    console.log([topic, payload].join(": "))
  })

  client.publish("Test1", "hello world js-client!")
</script>
<input type="text" id="output" size=30>
</body>
</html>
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Test der Seite durch Aufruf im Browser

```
http://192.168.1.1/mqtt/mqtt_life.html
```

Soll von außerhalb des lokalen Netzwerkes auf die Seite zugegriffen werden, so muss der Port 1884 freigegeben werden. Dies ist allerdings eine Sicherheitslücke, da sich jetzt jeder am Broker einschreiben kann.

8 Anhang 1

8.1 Literatur

- [1] *Trojan, Walter*: Das MQTT-Praxisbuch
1. Aufl., Elektor Verlag GmbH, Aachen 2017. ISBN 978-3-89576-324-3

8.2 Datei PubSubClient.h

```

/*
  PubSubClient.h - A simple client for MQTT.
  Nick O'Leary
  http://knolleary.net
*/

#ifndef PubSubClient_h
#define PubSubClient_h

#include <Arduino.h>
#include "IPAddress.h"
#include "Client.h"
#include "Stream.h"

#define MQTT_VERSION_3_1      3
#define MQTT_VERSION_3_1_1    4

// MQTT_VERSION : Pick the version
// #define MQTT_VERSION MQTT_VERSION_3_1
#ifndef MQTT_VERSION
#define MQTT_VERSION MQTT_VERSION_3_1_1
#endif

// MQTT_MAX_PACKET_SIZE : Maximum packet size
#ifndef MQTT_MAX_PACKET_SIZE
#define MQTT_MAX_PACKET_SIZE 128
#endif

// MQTT_KEEPALIVE : keepAlive interval in Seconds
#ifndef MQTT_KEEPALIVE
#define MQTT_KEEPALIVE 15
#endif

// MQTT_SOCKET_TIMEOUT: socket timeout interval in Seconds
#ifndef MQTT_SOCKET_TIMEOUT
#define MQTT_SOCKET_TIMEOUT 15
#endif

// MQTT_MAX_TRANSFER_SIZE : limit how much data is passed to the network client
// in each write call. Needed for the Arduino Wifi Shield. Leave undefined to
// pass the entire MQTT packet in each write call.
// #define MQTT_MAX_TRANSFER_SIZE 80

// Possible values for client.state()
#define MQTT_CONNECTION_TIMEOUT    -4
#define MQTT_CONNECTION_LOST      -3
#define MQTT_CONNECT_FAILED        -2
#define MQTT_DISCONNECTED          -1
#define MQTT_CONNECTED              0
#define MQTT_CONNECT_BAD_PROTOCOL  1
#define MQTT_CONNECT_BAD_CLIENT_ID  2
#define MQTT_CONNECT_UNAVAILABLE    3
#define MQTT_CONNECT_BAD_CREDENTIALS 4
#define MQTT_CONNECT_UNAUTHORIZED  5

#define MQTTCONNECT      1 << 4 // Client request to connect to Server
#define MQTTCONNACK      2 << 4 // Connect Acknowledgment
#define MQTTPUBLISH      3 << 4 // Publish message
#define MQTTPUBACK      4 << 4 // Publish Acknowledgment
#define MQTTPUBREC      5 << 4 // Publish Received (assured delivery part 1)
#define MQTTPUBREL      6 << 4 // Publish Release (assured delivery part 2)
#define MQTTPUBCOMP      7 << 4 // Publish Complete (assured delivery part 3)
#define MQTTSUBSCRIBE     8 << 4 // Client Subscribe request
#define MQTTSUBACK       9 << 4 // Subscribe Acknowledgment
#define MQTTUNSUBSCRIBE  10 << 4 // Client Unsubscribe request
#define MQTTUNSUBACK     11 << 4 // Unsubscribe Acknowledgment
#define MQTTPINGREQ      12 << 4 // PING Request

```

```

#define MQTTPINGRESP 13 << 4 // PING Response
#define MQTTFDISCONNECT 14 << 4 // Client is Disconnecting
#define MQTTReserved 15 << 4 // Reserved

#define MQTTQOS0 (0 << 1)
#define MQTTQOS1 (1 << 1)
#define MQTTQOS2 (2 << 1)

#ifdef ESP8266
#include <functional>
#define MQTT_CALLBACK_SIGNATURE std::function<void(char*, uint8_t*, unsigned int)> callback
#else
#define MQTT_CALLBACK_SIGNATURE void (*callback)(char*, uint8_t*, unsigned int)
#endif

class PubSubClient {
private:
    Client* _client;
    uint8_t buffer[MQTT_MAX_PACKET_SIZE];
    uint16_t nextMsgId;
    unsigned long lastOutActivity;
    unsigned long lastInActivity;
    bool pingOutstanding;
    MQTT_CALLBACK_SIGNATURE;
    uint16_t readPacket(uint8_t*);
    boolean readByte(uint8_t * result);
    boolean readByte(uint8_t * result, uint16_t * index);
    boolean write(uint8_t header, uint8_t* buf, uint16_t length);
    uint16_t writeString(const char* string, uint8_t* buf, uint16_t pos);
    IPAddress ip;
    const char* domain;
    uint16_t port;
    Stream* stream;
    int _state;
public:
    PubSubClient();
    PubSubClient(Client& client);
    PubSubClient(IPAddress, uint16_t, Client& client);
    PubSubClient(IPAddress, uint16_t, Client& client, Stream&);
    PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
    PubSubClient(IPAddress, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
    PubSubClient(uint8_t *, uint16_t, Client& client);
    PubSubClient(uint8_t *, uint16_t, Client& client, Stream&);
    PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
    PubSubClient(uint8_t *, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);
    PubSubClient(const char*, uint16_t, Client& client);
    PubSubClient(const char*, uint16_t, Client& client, Stream&);
    PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client);
    PubSubClient(const char*, uint16_t, MQTT_CALLBACK_SIGNATURE, Client& client, Stream&);

    PubSubClient& setServer(IPAddress ip, uint16_t port);
    PubSubClient& setServer(uint8_t * ip, uint16_t port);
    PubSubClient& setServer(const char * domain, uint16_t port);
    PubSubClient& setCallback(MQTT_CALLBACK_SIGNATURE);
    PubSubClient& setClient(Client& client);
    PubSubClient& setStream(Stream& stream);

    boolean connect(const char* id);
    boolean connect(const char* id, const char* user, const char* pass);
    boolean connect(const char* id, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage);
    boolean connect(const char* id, const char* user, const char* pass, const char* willTopic, uint8_t willQos, boolean willRetain, const char* willMessage);
    void disconnect();
    boolean publish(const char* topic, const char* payload);
    boolean publish(const char* topic, const char* payload, boolean retained);
    boolean publish(const char* topic, const uint8_t * payload, unsigned int plength);
    boolean publish(const char* topic, const uint8_t * payload, unsigned int plength, boolean retained);
    boolean publish_P(const char* topic, const uint8_t * payload, unsigned int plength, boolean retained);
    boolean subscribe(const char* topic);
    boolean subscribe(const char* topic, uint8_t qos);
    boolean unsubscribe(const char* topic);
    boolean loop();
    boolean connected();
    int state();
};

#endif

```

8.3 Datei PubSubClient.cpp

```

/* PubSubClient.cpp - A simple client for MQTT.
Nick O'Leary
http://knolleary.net
*/

#include "PubSubClient.h"
#include "Arduino.h"

PubSubClient::PubSubClient() {
    this->_state = MQTT_DISCONNECTED;
    this->_client = NULL;
    this->stream = NULL;
    setCallback(NULL);
}

PubSubClient::PubSubClient(Client& client) {
    this->_state = MQTT_DISCONNECTED;
    setClient(client);
    this->stream = NULL;
}

PubSubClient::PubSubClient(IPAddress addr, uint16_t port, Client& client) {
    this->_state = MQTT_DISCONNECTED;
    setServer(addr, port);
    setClient(client);
    this->stream = NULL;
}

PubSubClient::PubSubClient(IPAddress addr, uint16_t port, Client& client, Stream& stream) {
    this->_state = MQTT_DISCONNECTED;
    setServer(addr, port);
    setClient(client);
    setStream(stream);
}

PubSubClient::PubSubClient(IPAddress addr, uint16_t port, MQTT_CALLBACK_SIGNATURE, Client& client) {
    this->_state = MQTT_DISCONNECTED;
    setServer(addr, port);
    setCallback(callback);
    setClient(client);
    this->stream = NULL;
}

PubSubClient::PubSubClient(IPAddress addr, uint16_t port, MQTT_CALLBACK_SIGNATURE, Client& client, Stream& stream) {
    this->_state = MQTT_DISCONNECTED;
    setServer(addr, port);
    setCallback(callback);
    setClient(client);
    setStream(stream);
}

PubSubClient::PubSubClient(uint8_t *ip, uint16_t port, Client& client) {
    this->_state = MQTT_DISCONNECTED;
    setServer(ip, port);
    setClient(client);
    this->stream = NULL;
}

PubSubClient::PubSubClient(uint8_t *ip, uint16_t port, Client& client, Stream& stream) {
    this->_state = MQTT_DISCONNECTED;
    setServer(ip, port);
    setClient(client);
    setStream(stream);
}

PubSubClient::PubSubClient(uint8_t *ip, uint16_t port, MQTT_CALLBACK_SIGNATURE, Client& client) {
    this->_state = MQTT_DISCONNECTED;
    setServer(ip, port);
    setCallback(callback);
    setClient(client);
    this->stream = NULL;
}

PubSubClient::PubSubClient(uint8_t *ip, uint16_t port, MQTT_CALLBACK_SIGNATURE, Client& client, Stream& stream) {
    this->_state = MQTT_DISCONNECTED;
    setServer(ip, port);
    setCallback(callback);
    setClient(client);
    setStream(stream);
}

PubSubClient::PubSubClient(const char* domain, uint16_t port, Client& client) {
    this->_state = MQTT_DISCONNECTED;

```

```

        setServer(domain,port);
        setClient(client);
        this->stream = NULL;
    }
    PubSubClient::PubSubClient(const char* domain, uint16_t port, Client& client, Stream& stream) {
        this->_state = MQTT_DISCONNECTED;
        setServer(domain,port);
        setClient(client);
        setStream(stream);
    }
    PubSubClient::PubSubClient(const char* domain, uint16_t port, MQTT_CALLBACK_SIGNATURE, Client& client) {
        this->_state = MQTT_DISCONNECTED;
        setServer(domain,port);
        setCallback(callback);
        setClient(client);
        this->stream = NULL;
    }
    PubSubClient::PubSubClient(const char* domain, uint16_t port, MQTT_CALLBACK_SIGNATURE, Client& client,
    Stream& stream) {
        this->_state = MQTT_DISCONNECTED;
        setServer(domain,port);
        setCallback(callback);
        setClient(client);
        setStream(stream);
    }
    }

    boolean PubSubClient::connect(const char *id) {
        return connect(id,NULL,NULL,0,0,0,0);
    }

    boolean PubSubClient::connect(const char *id, const char *user, const char *pass) {
        return connect(id,user,pass,0,0,0,0);
    }

    boolean PubSubClient::connect(const char *id, const char* willTopic, uint8_t willQos, boolean willRetain,
    const char* willMessage) {
        return connect(id,NULL,NULL,willTopic,willQos,willRetain,willMessage);
    }

    boolean PubSubClient::connect(const char *id, const char *user, const char *pass, const char* willTopic,
    uint8_t willQos, boolean willRetain, const char* willMessage) {
        if (!connected()) {
            int result = 0;

            if (domain != NULL) {
                result = _client->connect(this->domain, this->port);
            } else {
                result = _client->connect(this->ip, this->port);
            }
            if (result == 1) {
                nextMsgId = 1;
                // Leave room in the buffer for header and variable length field
                uint16_t length = 5;
                unsigned int j;

                #if MQTT_VERSION == MQTT_VERSION_3_1
                    uint8_t d[9] = {0x00,0x06,'M','Q','I','s','d','p', MQTT_VERSION};
                #define MQTT_HEADER_VERSION_LENGTH 9
                #elif MQTT_VERSION == MQTT_VERSION_3_1_1
                    uint8_t d[7] = {0x00,0x04,'M','Q','T','T',MQTT_VERSION};
                #define MQTT_HEADER_VERSION_LENGTH 7
                #endif
                for (j = 0;j<MQTT_HEADER_VERSION_LENGTH;j++) {
                    buffer[length++] = d[j];
                }

                uint8_t v;
                if (willTopic) {
                    v = 0x06|(willQos<<3)|(willRetain<<5);
                } else {
                    v = 0x02;
                }

                if(user != NULL) {
                    v = v|0x80;

                    if(pass != NULL) {
                        v = v|(0x80>>1);
                    }
                }

                buffer[length++] = v;

                buffer[length++] = (MQTT_KEEPAIVE) >> 8);

```

```

        buffer[length++] = (MQTT_KEEPLIVE) & 0xFF);
        length = writeString(id,buffer,length);
        if (willTopic) {
            length = writeString(willTopic,buffer,length);
            length = writeString(willMessage,buffer,length);
        }

        if(user != NULL) {
            length = writeString(user,buffer,length);
            if(pass != NULL) {
                length = writeString(pass,buffer,length);
            }
        }

        write(MQTTCONNECT,buffer,length-5);

        lastInActivity = lastOutActivity = millis();

        while (!_client->available()) {
            unsigned long t = millis();
            if (t-lastInActivity >= ((int32_t) MQTT_SOCKET_TIMEOUT*1000UL)) {
                _state = MQTT_CONNECTION_TIMEOUT;
                _client->stop();
                return false;
            }
        }
        uint8_t llen;
        uint16_t len = readPacket(&llen);

        if (len == 4) {
            if (buffer[3] == 0) {
                lastInActivity = millis();
                pingOutstanding = false;
                _state = MQTT_CONNECTED;
                return true;
            } else {
                _state = buffer[3];
            }
        }
        _client->stop();
    } else {
        _state = MQTT_CONNECT_FAILED;
    }
    return false;
}
return true;
}

// reads a byte into result
boolean PubSubClient::readByte(uint8_t * result) {
    uint32_t previousMillis = millis();
    while(!_client->available()) {
        uint32_t currentMillis = millis();
        if(currentMillis - previousMillis >= ((int32_t) MQTT_SOCKET_TIMEOUT * 1000)){
            return false;
        }
    }
    *result = _client->read();
    return true;
}

// reads a byte into result[*index] and increments index
boolean PubSubClient::readByte(uint8_t * result, uint16_t * index){
    uint16_t current_index = *index;
    uint8_t * write_address = &(result[current_index]);
    if(readByte(write_address)){
        *index = current_index + 1;
        return true;
    }
    return false;
}

uint16_t PubSubClient::readPacket(uint8_t* lengthLength) {
    uint16_t len = 0;
    if(!readByte(buffer, &len)) return 0;
    bool isPublish = (buffer[0]&0xF0) == MQTTPUBLISH;
    uint32_t multiplier = 1;
    uint16_t length = 0;
    uint8_t digit = 0;
    uint16_t skip = 0;
    uint8_t start = 0;

    do {
        if(!readByte(&digit)) return 0;

```



```

        buffer[len++] = digit;
        length += (digit & 127) * multiplier;
        multiplier *= 128;
    } while ((digit & 128) != 0);
    *lengthLength = len-1;

    if (isPublish) {
        // Read in topic length to calculate bytes to skip over for Stream writing
        if(!readByte(buffer, &len)) return 0;
        if(!readByte(buffer, &len)) return 0;
        skip = (buffer[*lengthLength+1]<<8)+buffer[*lengthLength+2];
        start = 2;
        if (buffer[0]&MQTTQOS1) {
            // skip message id
            skip += 2;
        }
    }

    for (uint16_t i = start; i<length; i++) {
        if(!readByte(&digit)) return 0;
        if (this->stream) {
            if (isPublish && len-*lengthLength-2>skip) {
                this->stream->write(digit);
            }
        }
        if (len < MQTT_MAX_PACKET_SIZE) {
            buffer[len] = digit;
        }
        len++;
    }

    if (!this->stream && len > MQTT_MAX_PACKET_SIZE) {
        len = 0; // This will cause the packet to be ignored.
    }

    return len;
}

boolean PubSubClient::loop() {
    if (connected()) {
        unsigned long t = millis();
        if ((t - lastInActivity > MQTT_KEEPA_LIVE*1000UL) || (t - lastOutActivity > MQTT_KEEPA_LIVE*1000UL))
        {
            if (pingOutstanding) {
                this->_state = MQTT_CONNECTION_TIMEOUT;
                _client->stop();
                return false;
            } else {
                buffer[0] = MQTTPINGREQ;
                buffer[1] = 0;
                _client->write(buffer, 2);
                lastOutActivity = t;
                lastInActivity = t;
                pingOutstanding = true;
            }
        }
        if (_client->available()) {
            uint8_t llen;
            uint16_t len = readPacket(&llen);
            uint16_t msgId = 0;
            uint8_t *payload;
            if (len > 0) {
                lastInActivity = t;
                uint8_t type = buffer[0]&0xF0;
                if (type == MQTTPUBLISH) {
                    if (callback) {
                        uint16_t t1 = (buffer[llen+1]<<8)+buffer[llen+2]; /* topic length in bytes */
                        memmove(buffer+llen+2, buffer+llen+3, t1); /* move topic inside buffer 1 byte to
front */
                        buffer[llen+2+t1] = 0; /* end the topic as a 'C' string with \x00 */
                        char *topic = (char*) buffer+llen+2;
                        // msgId only present for QOS>0
                        if ((buffer[0]&0x06) == MQTTQOS1) {
                            msgId = (buffer[llen+3+t1]<<8)+buffer[llen+3+t1+1];
                            payload = buffer+llen+3+t1+2;
                            callback(topic, payload, len-llen-3-t1-2);

                            buffer[0] = MQTTPUBACK;
                            buffer[1] = 2;
                            buffer[2] = (msgId >> 8);
                            buffer[3] = (msgId & 0xFF);
                            _client->write(buffer, 4);
                            lastOutActivity = t;

```

```

        } else {
            payload = buffer+l1en+3+t1;
            callback(topic,payload,len-l1en-3-t1);
        }
    }
    } else if (type == MQTTPINGREQ) {
        buffer[0] = MQTTPINGRESP;
        buffer[1] = 0;
        _client->write(buffer,2);
    } else if (type == MQTTPINGRESP) {
        pingOutstanding = false;
    }
}
}
return true;
}
return false;
}

boolean PubSubClient::publish(const char* topic, const char* payload) {
    return publish(topic, (const uint8_t*)payload, strlen(payload), false);
}

boolean PubSubClient::publish(const char* topic, const char* payload, boolean retained) {
    return publish(topic, (const uint8_t*)payload, strlen(payload), retained);
}

boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength) {
    return publish(topic, payload, plength, false);
}

boolean PubSubClient::publish(const char* topic, const uint8_t* payload, unsigned int plength, boolean
retained) {
    if (connected()) {
        if (MQTT_MAX_PACKET_SIZE < 5 + 2+strlen(topic) + plength) {
            // Too long
            return false;
        }
        // Leave room in the buffer for header and variable length field
        uint16_t length = 5;
        length = writeString(topic,buffer,length);
        uint16_t i;
        for (i=0;i<plength;i++) {
            buffer[length++] = payload[i];
        }
        uint8_t header = MQTTPUBLISH;
        if (retained) {
            header |= 1;
        }
        return write(header,buffer,length-5);
    }
    return false;
}

boolean PubSubClient::publish_P(const char* topic, const uint8_t* payload, unsigned int plength, boolean
retained) {
    uint8_t l1en = 0;
    uint8_t digit;
    unsigned int rc = 0;
    uint16_t tlen;
    unsigned int pos = 0;
    unsigned int i;
    uint8_t header;
    unsigned int len;

    if (!connected()) {
        return false;
    }

    tlen = strlen(topic);

    header = MQTTPUBLISH;
    if (retained) {
        header |= 1;
    }
    buffer[pos++] = header;
    len = plength + 2 + tlen;
    do {
        digit = len % 128;
        len = len / 128;
        if (len > 0) {
            digit |= 0x80;
        }
        buffer[pos++] = digit;
    } while (len > 0);
    len = tlen;
    while (len > 0) {
        digit = payload[len-1];
        len--;
        buffer[pos++] = digit;
    }
    return true;
}

```

```

        llen++;
    } while(len>0);

    pos = writeString(topic,buffer,pos);

    rc += _client->write(buffer,pos);

    for (i=0;i<length;i++) {
        rc += _client->write((char)pgm_read_byte_near(payload + i));
    }

    lastOutActivity = millis();

    return rc == tlen + 4 + length;
}

boolean PubSubClient::write(uint8_t header, uint8_t* buf, uint16_t length) {
    uint8_t lenBuf[4];
    uint8_t llen = 0;
    uint8_t digit;
    uint8_t pos = 0;
    uint16_t rc;
    uint16_t len = length;
    do {
        digit = len % 128;
        len = len / 128;
        if (len > 0) {
            digit |= 0x80;
        }
        lenBuf[pos++] = digit;
        llen++;
    } while(len>0);

    buf[4-llen] = header;
    for (int i=0;i<llen;i++) {
        buf[5-llen+i] = lenBuf[i];
    }

#ifdef MQTT_MAX_TRANSFER_SIZE
    uint8_t* writeBuf = buf+(4-llen);
    uint16_t bytesRemaining = length+1+llen; //Match the length type
    uint8_t bytesToWrite;
    boolean result = true;
    while ((bytesRemaining > 0) && result) {
        bytesToWrite = (bytesRemaining > MQTT_MAX_TRANSFER_SIZE)?MQTT_MAX_TRANSFER_SIZE:bytesRemaining;
        rc = _client->write(writeBuf,bytesToWrite);
        result = (rc == bytesToWrite);
        bytesRemaining -= rc;
        writeBuf += rc;
    }
    return result;
#else
    rc = _client->write(buf+(4-llen),length+1+llen);
    lastOutActivity = millis();
    return (rc == 1+llen+length);
#endif
}

boolean PubSubClient::subscribe(const char* topic) {
    return subscribe(topic, 0);
}

boolean PubSubClient::subscribe(const char* topic, uint8_t qos) {
    if (qos < 0 || qos > 1) {
        return false;
    }
    if (MQTT_MAX_PACKET_SIZE < 9 + strlen(topic)) {
        // Too long
        return false;
    }
    if (connected()) {
        // Leave room in the buffer for header and variable length field
        uint16_t length = 5;
        nextMsgId++;
        if (nextMsgId == 0) {
            nextMsgId = 1;
        }
        buffer[length++] = (nextMsgId >> 8);
        buffer[length++] = (nextMsgId & 0xFF);
        length = writeString((char*)topic, buffer,length);
        buffer[length++] = qos;
        return write(MQTTSUBSCRIBE|MQTTQOS1,buffer,length-5);
    }
    return false;
}

```

```

}

boolean PubSubClient::unsubscribe(const char* topic) {
    if (MQTT_MAX_PACKET_SIZE < 9 + strlen(topic)) {
        // Too long
        return false;
    }
    if (connected()) {
        uint16_t length = 5;
        nextMsgId++;
        if (nextMsgId == 0) {
            nextMsgId = 1;
        }
        buffer[length++] = (nextMsgId >> 8);
        buffer[length++] = (nextMsgId & 0xFF);
        length = writeString(topic, buffer, length);
        return write(MQTTUNSUBSCRIBE|MQTTQOS1, buffer, length-5);
    }
    return false;
}

void PubSubClient::disconnect() {
    buffer[0] = MQTTDISCONNECT;
    buffer[1] = 0;
    _client->write(buffer, 2);
    _state = MQTT_DISCONNECTED;
    _client->stop();
    lastInActivity = lastOutActivity = millis();
}

uint16_t PubSubClient::writeString(const char* string, uint8_t* buf, uint16_t pos) {
    const char* idp = string;
    uint16_t i = 0;
    pos += 2;
    while (*idp) {
        buf[pos++] = *idp++;
        i++;
    }
    buf[pos-i-2] = (i >> 8);
    buf[pos-i-1] = (i & 0xFF);
    return pos;
}

boolean PubSubClient::connected() {
    boolean rc;
    if (_client == NULL) {
        rc = false;
    } else {
        rc = (int)_client->connected();
        if (!rc) {
            if (this->_state == MQTT_CONNECTED) {
                this->_state = MQTT_CONNECTION_LOST;
                _client->flush();
                _client->stop();
            }
        }
    }
    return rc;
}

PubSubClient& PubSubClient::setServer(uint8_t * ip, uint16_t port) {
    IPAddress addr(ip[0], ip[1], ip[2], ip[3]);
    return setServer(addr, port);
}

PubSubClient& PubSubClient::setServer(IPAddress ip, uint16_t port) {
    this->ip = ip;
    this->port = port;
    this->domain = NULL;
    return *this;
}

PubSubClient& PubSubClient::setServer(const char * domain, uint16_t port) {
    this->domain = domain;
    this->port = port;
    return *this;
}

PubSubClient& PubSubClient::setCallback(MQTT_CALLBACK_SIGNATURE) {
    this->callback = callback;
    return *this;
}

```

```

PubSubClient& PubSubClient::setClient(Client& client){
    this->_client = &client;
    return *this;
}

PubSubClient& PubSubClient::setStream(Stream& stream){
    this->stream = &stream;
    return *this;
}

int PubSubClient::state() {
    return this->_state;
}

```

8.4 Datei D1_class_MqttClientKH.h

```

//_____D1_class_MqttClientKH.h_____170721-170721_____
// The class MqttClient extends the class PubSubClient,
// so you can use all commands from this class as well.
// When PubSubClient lib is installed, delete directory /libs !
//
// Hardware: D1 mini
//           Button Shield D3
#ifndef D1_CLASS_MQTTCLIENTKH_H
#define D1_CLASS_MQTTCLIENTKH_H
#include <ESP8266WiFi.h>
#include "libs/PubSubClient.h"          // use with /libs

#define SSID_SIZE      20                // max.len ssid
#define PASS_SIZE      20                // max.len password
#define MQTT_SIZE      20                // max.len mqttservername
#define TOPIC_MAX      8                 // max. topics to sub
#define MQTT_RECONNECT_MS 4000
#define TIMEOUT_WIFI_CONNECT_MS 8000 // wait for WLAN
#define MESSAGE_MAXLEN 127
#ifndef DEBUG1
#define DEBUG1          true // true=Serial output
#endif

class MqttClientKH : public PubSubClient {
    //-----properties-----
protected:
    char ssid_[SSID_SIZE+1];             //
    char pass_[PASS_SIZE+1];             //
    char mqtt_[MQTT_SIZE+1];             //
    int port_;                           // mqtt port (def 1883)
    String aTopicSub_[TOPIC_MAX];        //
    String aTopicPub_[TOPIC_MAX];        //
    String aPayloadPub_[TOPIC_MAX];      // value on (re)start
    int numSub_;                          //
    int numPub_;                          //
    WiFiClient dlminiClient;             //
    long millis_lastConnected;           //
public:
    MqttClientKH(char* ssid, char* pwd, char* mqtt_server, int port);
    int  getNumSub() { return numSub_; };
    int  getNumPub() { return numPub_; };
    void clrSubscribe() { numSub_=0; };
    void clrPublish() { numPub_=0; };
    //-----methods to setup WLAN and mqtt connection-----
    bool setup_wifi();
    bool reconnect();
    bool isConnected();
    //-----methods to define mqtt topics-----
    bool addSubscribe(String topic);
    bool delSubscribe(String topic);
    bool addPublish(String topic, String payload);
    void publishString(String topic, String payload);
    void publishString(String topic, String payload, bool retained);
    //-----seldom used-----

```

```

    int  setSubscribe(String aTopicSub[], int num);
    int  setPublish(String aTopicPub[], String aPayload[], int num);
    void subscribeString(String topic);
};

//_____constructor_____
MqttClientKH::MqttClientKH(char* ssid, char* pwd,
    char* mqtt_server="localhost", int port=1883):PubSubClient(dlminiClient)
{
    strcpy(ssid_, ssid);
    strcpy(pass_, pwd);
    strcpy(mqtt_, mqtt_server);
    port_=port;
    millis_lastConnected = 0;
    numSub_=0;
    numPub_=0;
    setup_wifi();
    setServer(mqtt_, port_);
}

//*****
// methods to setup WLAN and mqtt connection
//*****

//_____connect to the WiFi network_____
bool MqttClientKH::setup_wifi()
{
    if(WiFi.status()==WL_CONNECTED) return true;
    delay(10);
    if(DEBUG1) Serial.println("\nConnecting to "+String(ssid_));
    WiFi.begin(ssid_, pass_);
    //-----try to connect to WLAN (access point)-----
    int i=TIMEOUT_WIFI_CONNECT_MS/200;
    while((WiFi.status()!=WL_CONNECTED) && (i>0))
    {
        delay(200);
        i--;
        if(DEBUG1){Serial.print("."); if(i%50==0) Serial.println("");}
    }
    //-----connected to WLAN (access point)?-----
    if(i<1)
    { //-----not connected to WLAN-----
        if(DEBUG1) Serial.println("No connection - time-out!");
        return false;
    }
    //-----success WiFi new connection/reconnect-----
    if(DEBUG1)Serial.println("\nConnected! IP address is "+WiFi.localIP().toString());
    return true;
}

//_____check for connect, if not: try to reconnect_____
bool MqttClientKH::reconnect()
{
    //-----when connected, return-----
    if(connected()) { return true; }
    //-----WiFi connected?-----
    if(!setup_wifi()) return false;
    //-----WiFi yes, mqtt no-----
    if(DEBUG1)Serial.println("MQTT: Not connected - reconnect...");
    //-----Create a random client ID-----
    randomSeed(micros()); // start random numbers
    String clientId = "Dlmini_Client-";
    clientId += String(random(0xffff), HEX);
    //-----Try to connect-----
    if(connect(clientId.c_str()))
    {
        if(DEBUG1) Serial.println(clientId+" connected.");
        //-----Once connected, publish an announcement-----
        for(int i=0; i<numPub_; i++)
            publishString(aTopicPub_[i], aPayloadPub_[i]);
        //.....and resubscribe.....
        for(int i=0; i<numSub_; i++)
            subscribeString(aTopicSub_[i]);
    }
}

```

```

    return true;
}
if(DEBUG1)Serial.println("failed, client state rc="+String(state()));
return false;
}

//_____is mqtt connection ok? (no: reconnect)_____
bool MqttClientKH::isConnected()
{
    long now = millis();
    //-----check for mqtt connection-----
    if (!connected())
    {
        if (now - millis_lastConnected > MQTT_RECONNECT_MS)
        {
            millis_lastConnected=now;
            if(reconnect()) millis_lastConnected=0;
        }
    }
    //-----if connected to broker, do loop function-----
    if (connected())
    {
        loop();
        return true;
    }
    return false;
}

//*****
// methods to define mqtt topics
//*****

//_____add a (String) topic to subscribe array_____
bool MqttClientKH::addSubscribe(String topic)
{
    //-----is topic already in subscribe array?-----
    for(int i=0; i<numSub_; i++)
    {
        if(topic.equals(aTopicSub_[i])) return true;
    }
    //-----add topic (if enough space)-----
    if(numSub_<TOPIC_MAX)
    {
        aTopicSub_[numSub_++]=topic;
        return true;
    }
    return false;
}

//_____convert String to array and unsubscribe_____
// return: true=unsubscribed, false=not
bool MqttClientKH::delSubscribe(String topic)
{
    //-----is topic in subscribe array?-----
    int i=0;
    for(i=0; i<numSub_; i++)
    {
        if(topic.equals(aTopicSub_[i])) break;
    }
    if(i>=numSub_) return false;
    i--;
    //-----topic in array at index i, delete from array-----
    for(int j=i; j<numSub_; j++)
        aTopicSub_[j]=aTopicSub_[j+1];
    numSub_--;
    //-----unsubscribe from broker-----
    char cTopic[1+MESSAGE_MAXLEN]; // helper array
    topic.toCharArray(cTopic,MESSAGE_MAXLEN);
    unsubscribe(cTopic);
}

//_____add a (String) topic to publish array_____
bool MqttClientKH::addPublish(String topic, String payload)

```

```

{
//-----is topic already in subscribe array?-----
for(int i=0; i<numPub_; i++)
{
    if(topic.equals(aTopicPub_[i])) return true;
}
//-----add topic (if enough space)-----
if(numPub_<TOPIC_MAX)
{
    aTopicPub_[numPub_]=topic;
    aPayloadPub_[numPub_]=payload;
    numPub_++;
    return true;
}
return false;
}

//_____convert String to array and publish_____
void MqttClientKH::publishString(String topic, String payload)
{
    char top[1+MESSAGE_MAXLEN];          // helper array
    char msg[1+MESSAGE_MAXLEN];          // helper array
    topic.toCharArray(top,MESSAGE_MAXLEN);
    payload.toCharArray(msg,MESSAGE_MAXLEN);
    publish(top,msg);
}

//_____convert String to array and publish_____
void MqttClientKH::publishString(
    String topic, String payload, bool retained=false)
{
    char top[1+MESSAGE_MAXLEN];          // helper array
    char msg[1+MESSAGE_MAXLEN];          // helper array
    topic.toCharArray(top,MESSAGE_MAXLEN);
    payload.toCharArray(msg,MESSAGE_MAXLEN);
    publish(top,msg,retained);
}

//_____set array of registered subscribe topics_____
// return: number of registered subscribe topics
int MqttClientKH::setSubscribe(String aTopicSub[], int num)
{
    for(int i=0; i<num; i++)
    {
        //-----check, if topic is already registered-----
        bool isreg=false;
        for(int j=0; j<numSub_; j++)
        {
            if(aTopicSub[i].equals(aTopicSub_[j])) {isreg=true; j=numSub_;}
        }
        //-----if not registered: add topic-----
        if(!isreg)
        {
            if(numSub_<TOPIC_MAX) aTopicSub_[numSub_++]= aTopicSub[i];
        }
        if(numSub_>=TOPIC_MAX) return TOPIC_MAX;
    }
    return numSub_;
}

//_____set array of registered subscribe topics_____
// return: number of registered subscribe topics
int MqttClientKH::setPublish(
    String aTopicPub[], String aPayload[], int num)
{
    for(int i=0; i<num; i++)
    {
        //-----check, if topic is already registered-----
        bool isreg=false;
        for(int j=0; j<numPub_; j++)
        {
            if(aTopicPub[i].equals(aTopicPub_[j])) {isreg=true; j=numPub_;}
        }
    }
}

```



```
//-----if not registered: add topic-----
if(!isreg)
{
    if(numPub_<TOPIC_MAX)
    {
        aTopicPub_[numPub_]= aTopicPub[i];
        aPayloadPub_[numPub_++]= aPayload[i];
    }
}
if(numPub_>=TOPIC_MAX) return TOPIC_MAX;
}
return numPub_;
}

//_____convert String to array and subscribe_____
void MqttClientKH::subscribeString(String topic)
{
    char top[1+MESSAGE_MAXLEN];          // helper array
    topic.toCharArray(top,MESSAGE_MAXLEN);
    subscribe(top);
}

#endif
```

8.5 *Datei utils_file.h*

```

/* utils_file.h, 13.8.2014-1.11.2017, Karl Hartinger          */
#ifndef _FILE_UTILS_H_
#define _FILE_UTILS_H_
#include <stdlib.h>          // system
#ifdef __cplusplus
extern "C" {
#endif
#include <stdio.h>          // printf
#include <string.h>         // sprintf
#include <errno.h>          // errno, strerror

#ifdef FILENAME_MAX
#undef FILENAME_MAX
#endif
#define FILENAME_MAX        256 // LINUX-Default: 4096 !!!
#define LINE_LEN_MAX        896 // date;cmd;par;ttl;ok;com
#define EOL                  "\n"

//====Special data types=====
typedef char type_line[LINE_LEN_MAX+1];
extern type_line last_error;
extern int last_errno;

//_____does file exist?_____
// returns 1=yes, 0=no
int isfile(char *filename_);
//_____delete file_____
// returns 0=OK, <0 on error
int deletefile(char *filename_);
//_____writes a line (string) in a file_____
// old values will be deleted
// returns 0=OK, 1 on error
int writeline(char *filename_, char *line_);
//_____insert a new line in log file_____
// returns 0=OK, 1 on error
int insertline(char *filename_, char *line_);
//_____insert a new FIRST line in log file_____
// returns 0=OK, 1,2 or 3 on error
int insertfirstline(char *filename_, char *line_);
//_____get first line of file_____
// returns 0=OK, 1 on error
int getfirstline(char *filename_, char *line_);
//_____get last line of file_____
// returns 0=OK, 1 on error
int getlastline(char *filename_, char *line_);
//_____show content of (text) file_____
// returns 0=OK, 1 on error
int showfile(char *filename_);
//_____save a line to a file_____
// type "w" makes a new file, "a" appends to a file
// returns 0=OK, 1 on error
int saveline(char *filename_, char* type, char *line_);
//_____save lines to a file_____
// type "w" makes a new file, "a" appends to a file
// returns 0=OK, 1 on error
int savelines(char *filename_, char* type_, int nr_, type_line lines[]);

//_____remove \r \n from end of line, set new end of string_____
int remove_lfcr(char *line_);

#ifdef __cplusplus
}
#endif
#endif // _FILE_UTILS_H_

```

8.6 *Datei utils_file.c*

```

/* utils_file.c, 13.8.2014-8.3.2015, Karl Hartinger          */
#include "utils_file.h"

//====Special data types=====
type_line last_error;
int last_errno;

//_____does file exist?_____
// returns 1=yes, 0=no
int isfile(char *filename_)
{
    FILE *fp;
    if( (fp=fopen(filename_, "r")) == NULL) return 0;
    fclose(fp);
    return 1;
}

//_____delete file_____
// returns 0=OK, <>0 on error
int deletefile(char *filename_)
{
    FILE *fp;
    if( (fp=fopen(filename_, "r")) == NULL) return 0;
    fclose(fp);
    char temp[FILENAME_MAX+4]="rm ";
    strcat(temp, filename_);
    return(system(temp));
}

//_____writes a line (string) in a file_____
// old values will be deleted
// returns 0=OK, 1 on error
int writeline(char *filename_, char *line_)
{
    return (saveline(filename_, "w", line_));
}

//_____insert (append) a new line in log file_____
// returns 0=OK, 1 on error
int insertline(char *filename_, char *line_)
{
    return (saveline(filename_, "a", line_));
}

//_____insert a new first line in log file_____
// returns 0=OK, 1,2 or 3 on error
int insertfirstline(char *filename_, char *line_)
{
    FILE *fp1, *fp2;
    type_line line1;
    char s[255];
    int ret;
    //-----delete temp file-----
    if(isfile("temp.log")) deletefile("temp.log");
    //-----file does not exist yet-----
    if(isfile(filename_)!=1)
    {
        ret=saveline(filename_, "w", line_);
        return ret;
    }
    //-----copy old file to temp.log-----
    sprintf(s,"cp %s temp.log", filename_);
    system(s);
    ret=saveline(filename_, "w", line_);
    //-----open file-----
    fp1=fopen("temp.log", "r");
    if( fp1 == NULL)
    {
        last_errno=errno;
        sprintf(last_error,"Fehler 1 - insertfirstline: %d %s\n",errno,strerror(errno));
        return 1;
    }
    fp2=fopen(filename_, "a");
    if( fp2 == NULL)
    {
        last_errno=errno;
        sprintf(last_error,"Fehler 2 - insertfirstline: %d %s\n",errno,strerror(errno));
        fclose(fp1);
        deletefile("temp.log");
    }
}

```

```

    return 2;
}
//-----copy lines-----
while( fgets(line1_, LINE_LEN_MAX, fp1) != NULL)
{
    ret=fputs(line1_, fp2);
    if(ret<0)
    {
        last_errno=errno;
        sprintf(last_error,"Fehler 3 - insertfirstline: %d %s\n",errno,strerror(errno));
        fclose(fp2);
        fclose(fp1);
        deletefile("temp.log");
        return 3;
    }
}
fclose(fp2);
fclose(fp1);
deletefile("temp.log");
return ret;
}

//_____get first line_____
// returns 0=OK, 1 on error
int getfirstline(char *filename_, char *line_)
{
    FILE *fp;
    if( (fp=fopen(filename_, "r")) == NULL) return 1;
    fgets(line_, LINE_LEN_MAX, fp);
    remove_lfcr(line_);
    fclose(fp);
    return 0;
}

//_____get last line_____
// returns 0=OK, 1 on error
int getlastline(char *filename_, char *line_)
{
    FILE *fp;
    if( (fp=fopen(filename_, "r")) == NULL) return 1;
    while( fgets(line_, LINE_LEN_MAX, fp) != NULL) ;
    remove_lfcr(line_);
    fclose(fp);
    return 0;
}

//_____show content of (text) file_____
// returns 0=OK, 1 on error
int showfile(char *filename_)
{
    FILE *fp;
    type_line line_;
    if( (fp=fopen(filename_, "r+")) == NULL) return 1;
    while( fgets(line_, LINE_LEN_MAX, fp) != NULL)
        printf("%s",line_);
    fclose(fp);
    return 0;
}

//_____save a line to a file_____
// type "w" makes a new file, "a" appends to a file
// returns 0=OK, 1 on error
int saveline(char *filename_, char* type_, char *line_)
{
    int ret;
    FILE *fp;
    type_line line1_;
    //-----check open-type-----
    if( (strcmp(type_, "w")!=0) && (strcmp(type_, "a")!=0) && (strcmp(type_, "a+")!=0) )
    {
        last_errno=-1;
        sprintf(last_error,"writelines-1: Error type ist not \"w\" or \"a\\n\"");
        return 1;
    }
    //-----open file-----
    fp=fopen(filename_, type_);
    if( fp == NULL)
    {
        last_errno=errno;
        sprintf(last_error,"Fehler 1 - saveline: %d %s\n",errno,strerror(errno));
        return 1;
    }
    //-----write line to file-----
    sprintf(line1_, "%s%s", line_, EOL);

```

```

ret=fputs(line1_, fp);
if(ret<0)
{
    last_errno=errno;
    sprintf(last_error,"Fehler 2 - saveline: %d %s\n",errno,strerror(errno));
    fclose(fp);
    return 1;
}
//-----close file-----
ret=fclose(fp);
if(ret<0)
{
    last_errno=errno;
    sprintf(last_error,"Fehler 3 - saveline: %d %s\n",errno,strerror(errno));
    fclose(fp);
    return 1;
}
return 0;
}

//_____save lines to a file_____
// type "w" makes a new file, "a" appends to a file
// returns 0=OK, 1 on error
int savelines(char *filename_, char* type_, int nr_, type_line lines[])
{
    int i,ret;
    FILE *fp1;
    //-----check open-type-----
    if((strcmp(type_,"w")!=0) && (strcmp(type_,"a")!=0) && (strcmp(type_,"a+")!=0))
    {
        last_errno=-1;
        sprintf(last_error,"savelines-1: Error type ist not \"w\" or \"a\"\\n");
        return 1;
    }
    //-----try to open file-----
    fp1=fopen(filename_, type_);
    if(fp1==NULL)
    {
        last_errno=errno;
        sprintf(last_error,"savelines-2: Error %d: %s\n",errno,strerror(errno));
        return 2;
    }
    //-----write lines to file-----
    for(i=0; i<nr_; i++)
    {
        ret=fputs(lines_[i], fp1);
        if(ret<0)
        {
            last_errno=errno;
            sprintf(last_error,"savelines-3-%d Error %d: %s\n",i,errno,strerror(errno));
            fclose(fp1);
            return 3;
        }
    }
    //-----close file-----
    ret=fclose(fp1);
    if(ret<0)
    {
        last_errno=errno;
        printf("savelines-4 Error %d: %s\n",errno,strerror(errno));
        return 4;
    }
    return 0;
}

//_____remove \\r \\n from end of line, set new end of string_____
int remove_lfcr(char *line_)
{
    char* p=strrchr(line_,'\0');
    p--;
    while(*p=='\\r' || *p=='\\n') { *p='\\0'; p--; }
    return 0;
}

```

9 Anhang 2: RasPi einrichten

9.1 Betriebssystem auf SD-Card installieren

1. Herunterladen des Hilfsprogramms zum Beschreiben der SD-Card unter Windows
Link „Win32DiskImager“ auf <http://www.raspberrypi.org/downloads>
2. Herunterladen des Betriebssystems von <http://www.raspberrypi.org/downloads>,
Auf Kachel „Raspbian“ klicken =>

```
Raspbian Stretch with desktop
Version: September 2017
Release date: 2017-09-07
Kernel version: 4.9
```

Auf [Download Zip] klicken, Image herunterladen (ca. 1GByte!!)

3. DiskImager und Betriebssystem entpacken (ergibt ca. eine 4,8-GByte-Datei)
4. DiskImager starten, Betriebssystem-Imagedatei und Ziellaufwerk (!) auswählen und Datei übertragen [Write]

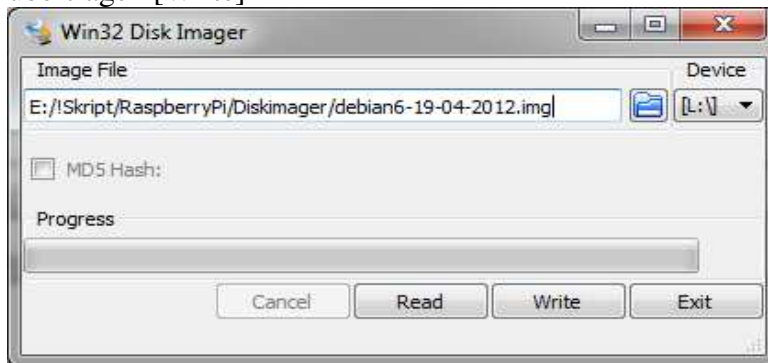


Bild 2: SD-Card DiskImager (im Bild mit einem alten Image-File ;)

9.2 Grundeinstellungen vornehmen (nach dem ersten Start)

SD-Karte einlegen, Bildschirm und Tastatur anstecken, Stromversorgung anlegen. Das RasPi startet und es erscheint nach einiger Zeit der graphische Bildschirm (Stand 24.10.2017). Die Konfiguration kann entweder in der grafischen Oberfläche erfolgen oder in der Konsole (bei älteren Systemen nur in der Konsole).

Konfiguration in der grafischen Oberfläche

```
[Menu] links oben anklicken – Preferences (Einstellungen) – Raspberry Pi Configuration –
[System]
  Password:                               [Change Password...] zB ras!!11 [OK] [OK]
  Hostname:                               Rasp11
  Boot:                                   Ⓐ Zum Desktop
  Automatische Anmeldung: [x] Als Benutzer 'pi' anmelden
  [OK]
  Would you like to reboot now [No]

[Menu] – Preferences (Einstellungen) – Raspberry Pi Configuration – [Interfaces]
  SSH                                   Ⓐ Enabled
```

I2C:	⊙	Enabled
[OK]		
[Menu] – Preferences (Einstellungen) – Raspberry Pi Configuration – [Localisation]		
[Set Locale...] Language: de (Austrian German), Country: AT (Austria), Character Set: UTF-8 [OK]		
[Set Timezone] Area: Europe, Location: Vienna [OK]		
[Set Keyboard] Country: Austria, Variant: German (Austria) [OK]		
[Set WiFi Country...] AT Austria [OK]		
Would you like to reboot now [Yes]		

Konfiguration mittels Konsole (optional)

Aus der grafischen Oberfläche heraus kann auch eine Konsole öffnen durch <Strg><Alt><F2> und Eingabe von Username (default raspberry login: **pi**) und Password (default): **raspberrry** (Eingabe wird nicht angezeigt). Sollte die Ausgabe „Login incorrect“ und eine nochmalige Eingabe des Passwortes notwendig sein: **raspberrrz** eingeben (Englische Tastatur).

Danach ruft man das Konfigurationsmenü auf: (Bei älteren Versionen wird es automatisch beim ersten Start angezeigt):

```
pi@raspberrypi ~ $ sudo raspi-config
```

(Bei englischer Tastatur erhält man den Bindestrich – mit der Taste ß!!!)

```
Raspi-config - Setup Options
1 Expand Filesystem          Ensures that all of the SD card storage is available to the OS
2 Change User Password       Change password for the default user (pi)
3 Enable Boot to Desktop/Scratch Choose whether to boot into desktop, Scratch or command-line
4 Internationalisation Options Setup language and regional settings to match your location
5 Enable Camera              Enable this Pi to work with the Raspberry Pi Camera
6 Add to Rastrack            Add this Pi to the online Raspberry Pi Map (Tastrack)
7 Overclock                  Configure overclocking for your Pi
8 Advanced Options           Configure advanced settings
9 About raspi-config          Information about this configuration tool

                                <Select>                                <Finish>
```

Beispiele für Änderungen der Konfiguration (Auswahl mit Hilfe der Cursor-Tasten):

- ▶ 1 Expand Filesystem markieren, mit Tabulator auf <Select> <Enter>
- ▶ 2 Change User Password Passwort ändern (**ras!!11** oder wie bisher **raspberrry**)
- ▶ 3 Enable Boot.. (Console Text console (default) wählen <OK>)
- ▶ 4 Internationalisation - I1 Change Locale
[*] de_AT.UTF8 UTF8 wählen **und** [] en_GB.UTF-8 abwählen! <Ok>
Environment de_AT.UTF-8
- ▶ 4 Internationalisation - I2 Time zone - Europe - Vienna<Ok>
- ▶ 4 Internationalisation - I3 Keyboard
Generic 105-key (Intl) PC, Other, German (Austria), AltGr= The default for the keyboard layout, No compose key, Control+Alt+Backspace **<Yes>!**)

Abschließend mit Tabulator <Finish> wählen, <Enter>

Would you like to reboot now **<Yes>** <Enter>

Das Rebooten kann auch mit folgendem Befehl erreicht werden:

```
pi@raspberrypi ~ $ sudo reboot
```

9.3 Login und Update der Installation

Nach dem neuerlichen Hochfahren kann das Einloggen mit den neuen Werten erfolgen:

```

rasberry login: pi
Password: ras!!11
(oder, wenn nichts geändert wurde, raspberry)
pi@raspberrypi ~ $

```

Sollte das Einloggen nicht erfolgreich sein und das Passwort nicht geändert worden sein, dann mit dem Passwort `rasberrz` probieren, da möglicherweise die deutsche Tastatur nicht eingestellt ist (dann ist z und y vertauscht, bzw. ist das Minuszeichen auf der ß-Taste).

Danach ein Terminal-Fenster öffnen und folgendes eingeben:

```

~ $ sudo apt-get update
~ $ sudo apt-get upgrade

```

9.4 Terminal-Schrift ändern (optional)

Falls man kleinere, grafische Displays verwenden möchte (zB 3,5“), so sollte man die Schrift im LXTerminal etwas vergrößern. Dies erfolgt durch Aufruf des Programms LXTerminal (in der Menüleiste oben), Menü Bearbeiten – Einstellungen [Stil] Terminal-Schriftart: Monospace – Stil Bold – Größe 12 [OK] [OK]

9.5 Standard-User-Namen pi ändern (optional)

Aus Sicherheitsgründen sollte der User `pi` umbenannt werden, zB auf `pi11`. Dies geschieht wie unten beschrieben. (Quelle: <https://jankarres.de/2013/09/raspberry-pi-standard-benutzername-pi-aendern/> [24.10.2017])

Anmerkung: Bereits erstellte User-Crontabs werden beim Ändern des Benutzernamens gelöscht – daher zuvor sichern!

- (1) Terminal starten durch Anklicken des Symbols (LXTerminal) in der Leiste oben
- (2) Hilfs-User `temp` anlegen

```

~ $ sudo adduser --no-create-home temp
Geben Sie ein neues UNIX-Passwort ein: ras!!11
Geben Sie das neue UNIX-Passwort erneut ein: ras!!11

```

Alle anderen Eingaben leer lassen. (Es werden die Standardwerte übernommen)

- (3) Dem User `temp` sudo-Rechte einräumen

```

~ $ sudo visudo

```

Am Ende der Datei einfügen

```
temp ALL=(ALL) NOPASSWD: ALL
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

- (4) Beim nächster Start in die Konsole booten und das Passwort abfragen
 [Menu] – Preferences (Einstellungen) – Raspberry Pi Configuration – [System]
 Boot: ☒ Zum CLI
 Automatische Anmeldung: ☐ Als Benutzer 'pi' anmelden
 [OK]
 Would you like to reboot now [Yes]

- (5) Raspberry Pi neu starten und als User temp einloggen

```
Raspill login: temp
password: ras!!11
```

- (6) Benutzer pi auf pill umbenennen

```
~ $ sudo usermod --move-home --login pill --home /home/pill pi
```

- (7) Den Namen der Gruppe pi ändern

```
~ $ sudo groupmod -n pill pi
```

- (8) Raspberry Pi neu starten und als User pill einloggen

```
Raspill login: pill
password: ras!!11
```

- (9) Dem User temp die sudo-Rechte entziehen:

```
~ $ sudo visudo
```

Kommentarzeichen einfügen:

```
#temp ALL=(ALL) NOPASSWD: ALL
```

Speichern und beenden durch <Strg>o <Enter> <Strg> xh <Strg>o <Enter> <Strg> x

- (10) Löschen des Users temp

```
~ $ sudo deluser temp
```

- (11) In der Datei /etc/lightdm/lightdm.conf den User für ein automatisches Login ändern:

```
autologin-user=pill
```

- (12) Grafik-Bildschirm aufrufen und Zurückstellen auf Start mit Grafik-Bildschirm (falls gewünscht)

```
~ $ startx
```

[Menu] – Preferences (Einstellungen) – Raspberry Pi Configuration – [System]

Boot: ☒ Zum Desktop

Automatische Anmeldung: ☒ Als Benutzer 'pi' anmelden

Anmerkung – Fehler in der Grafik-Oberfläche: Es wird immer der Benutzer 'pi' angezeigt, obwohl der User pill verwendet wird.

9.6 Netzwerk konfigurieren beim Raspberry Pi 3

Die Netzwerkkonfiguration ist von der Raspberry-Version abhängig, da zB das Raspberry 3 bereits eine Wireless LAN- und Bluetooth-Schnittstelle direkt Onboard hat. Die Netzwerkkonfiguration Für die Versionen 1 und 2 findet sich weiter hinten im Anhang.

Die aktuellen Netzwerkeinstellungen können im Terminal folgendermaßen angezeigt werden:

```
~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 192.168.0.102  netmask 255.255.252.0  broadcast 192.168.3.255
    inet6 fe80::a7a1:3a43:307a:1734  prefixlen 64  scopeid 0x20<link>
    ether b8:27:eb:02:b3:9d  txqueuelen 1000  (Ethernet)
    RX packets 916  bytes 64946 (63.4 KiB)
    RX errors 0  dropped 72  overruns 0  frame 0
    TX packets 90  bytes 13266 (12.9 KiB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
...
```

Raspberry 3: DHCP LAN-/WLAN-Konfiguration

Im für das RasPi3 benötigten Betriebssystem (Raspian Jessie ab Februar 2016) ist standardmäßig bereits ein DHCP-Client-Dämon aktiv. So kann das WLAN einfach in der grafischen Oberfläche konfiguriert werden. Die vorhandenen Netzwerke werden direkt nach dem Anklicken des Netzwerk-Symbols rechts oben auf dem Bildschirm angezeigt. Danach muss nur noch der Schlüssel des Netzwerkes (Passphrase) eingegeben werden.

Anmerkung: Die Netzwerkeinstellungen werden in der Datei `/etc/wpa_supplicant/wpa_supplicant.conf` gespeichert und können folgendermaßen angezeigt werden:

```
~ $ sudo cat /etc/wpa_supplicant/wpa_supplicant.conf
```

Raspberry 3: Statische IP-Adressen ¹

Will man keine automatische Zuweisung der IP-Adresse (DHCP), so sollte die Konfiguration über den DHCP-Client-Dämon erfolgen und **nicht** mehr über die Datei `/etc/network/interfaces`! Diese sollte sich im Originalzustand befinden:

```
~ $ cat /etc/network/interfaces
```

Inhalt:

```
# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d# interfaces(5) file used by ifup(8) and ifdown(8)

# Please note that this file is written to be used with dhcpcd
# For static IP, consult /etc/dhcpcd.conf and 'man dhcpcd.conf'

# Include files from /etc/network/interfaces.d:
source-directory /etc/network/interfaces.d
```

¹ Quelle: <https://www.elektronik-kompodium.de/sites/raspberry-pi/1912151.htm>

In früheren Versionen der Datei waren zB noch folgende Zeilen enthalten:

```
auto lo
iface lo inet loopback

iface eth0 inet manual

allow-hotplug wlan0
iface wlan0 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf

allow-hotplug wlan1
iface wlan1 inet manual
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Nun kontrolliert man, ob der Dämon aktiv ist:

```
~ $ sudo service dhcpd status
dhcpd.service - dhcpd on all interfaces
Loaded: loaded (/lib/systemd/system/dhcpd.service; enabled; vendor preset: enabled)
Active: active (running) since Fri 2017-10-27 16:20:50 CEST; 17min ago
Process: 332 ExecStart=/usr/lib/dhcpd5/dhcpd -q -b (code=exited, status=0/SUCCESS)
Main PID: 365 (dhcpd)
CGroup: /system.slice/dhcpd.service
        └─365 /sbin/dhcpd -q -b
          └─453 wpa_supplicant -B -c/etc/wpa_supplicant/wpa_supplicant.conf -iwlan0
...

```

Sollte bei Active: **failed** stehen, so muss das Service gestartet werden:

```
~ $ sudo service dhcpd start
~ $ sudo systemctl enable dhcpd
```

Die folgenden Änderungen bewirken, dass die LAN-Verbindung auf die statische IP 192.168.0.55 eingestellt wird und das WLAN weiterhin eine dynamische IP erhält. Soll auch das WLAN eine statische IP erhalten, braucht man nur die Kommentarzeichen (#) entfernen und die entsprechenden Werte eintragen.

Sichern der Originaldatei und bearbeiten der Datei /etc/dhcpd.conf:

```
~ $ sudo cp /etc/dhcpd.conf /etc/dhcpd.conf.original
~ $ sudo nano /etc/dhcpd.conf
```

Datei ändern:

```
# A sample configuration for dhcpd.
# See dhcpd.conf(5) for details.

# Allow users of this group to interact with dhcpd via the control socket.
#controlgroup wheel

# Inform the DHCP server of our hostname for DDNS.
hostname

# Use the hardware address of the interface for the Client ID.
clientid
# or
# Use the same DUID + IAID as set in DHCPv6 for DHCPv4 ClientID as per RFC4361.
# Some non-RFC compliant DHCP servers do not reply with this set.
# In this case, comment out duid and enable clientid above.
#duid

# Persist interface configuration when dhcpd exits.
persistent

# Rapid commit support.
# Safe to enable by default because it requires the equivalent option set
# on the server to actually work.
```

```
option rapid_commit

# A list of options to request from the DHCP server.
option domain_name_servers, domain_name, domain_search, host_name
option classless_static_routes
# Most distributions have NTP support.
option ntp_servers
# Respect the network MTU. This is applied to DHCP routes.
option interface_mtu

# A ServerID is required by RFC2131.
require dhcp_server_identifier

# Generate Stable Private IPv6 Addresses instead of hardware based ones
slaac private

# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8

# It is possible to fall back to a static IP if DHCP fails:
# define static profile
#profile static_eth0
#static ip_address=192.168.1.23/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1

# fallback to static profile on eth0
#interface eth0
#fallback static_eth0

#denyinterfaces wlan0
#denyinterfaces eth0

#-----statische IP-----
interface eth0
static ip_address=192.168.0.11/24
static routers=192.168.0.1
static domain_name_servers=192.168.0.1

#interface wlan0
#static ip_address=192.168.1.1/24
#static routers=192.168.1.1
#static domain_name_servers=192.168.1.1
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Test der erfolgreichen Umstellung von LAN auf eine statische IP

- (1) Raspberry Pi neu starten

```
~ $ sudo reboot
```

- (2) In der grafischen Oberfläche LXTerminal starten und folgendes eingeben:

```
~ $ ifconfig eth0
```

WLAN-Einstellungen im Terminal bearbeiten

Sichern der alten Netzwerkeinstellungen und eintragen der neuen Daten in die Datei
/etc/wpa_supplicant/wpa_supplicant.conf

```
~ $ sudo cp /etc/wpa_supplicant/wpa_supplicant.conf  
/etc/wpa_supplicant/wpa_supplicant.conf.original  
  
~ $ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
```

Datei ändern:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev  
update_config=1  
country=AT  
  
network={  
    ssid="***WLAN name***"  
    psk="***password***"  
    key_mgmt=WPA-PSK  
}
```

Jetzt können die Netzwerk-Interfaces neu gestartet werden...

```
~ $ sudo ifdown eth0  
~ $ sudo ifup eth0  
~ $ sudo ifdown wlan0  
~ $ sudo ifup wlan0
```

oder besser, das RasPi neu gestartet werden:

```
~ $ sudo reboot
```

Nach dem Neustart kann man kontrollieren, ob die Interfaces die richtige IP haben und das RasPi ins Internet kommt:

```
~ $ ifconfig  
~ $ ping 8.8.8.8 -c 2
```

9.7 Arbeiten über das Netzwerk

Statt Tastatur und Monitor am RasPi anzustecken, kann man auch über das Netzwerk auf dem RasPi arbeiten. Auf der PC-Seite sind dazu sinnvollerweise zwei Programme erforderlich:

- Terminal-Programm zur Eingabe von Befehlen, zB **putty**
- File-Transfer-Programm zum Hin- und Herschicken von Dateien, zB **WinSCP**

Nach der Installation der Programme auf dem PC und dem Verbinden des RasPi über ein Netzkabel kann dieses über die entsprechende IP-Adresse (zB 192.168.0.55) erreicht werden.

9.8 Installation des 3,5“ HDMI Touch Screen LCD Treibers

Hat man auf einer microSD-Karte ein Raspian-System bereits installiert, so kann man wie unten stehend den 3,5“ HDMI Touch Screen LCD Treiber installieren. (Unterstützte Betriebssysteme: Raspbian und Ubuntu). Quelle:

<http://osoyoo.com/2017/01/18/install-3-5-hdmi-touch-screen-linux-driver-on-raspberry-pi/>

Alternativ dazu kann man auch ein Raspian-Image herunterladen, das die Treiber bereits enthält:

<http://osoyoo.com/2016/11/20/raspberry-pi-3-5inch-hdmi-touchscreen/>

(Stand: 23.10.2017)

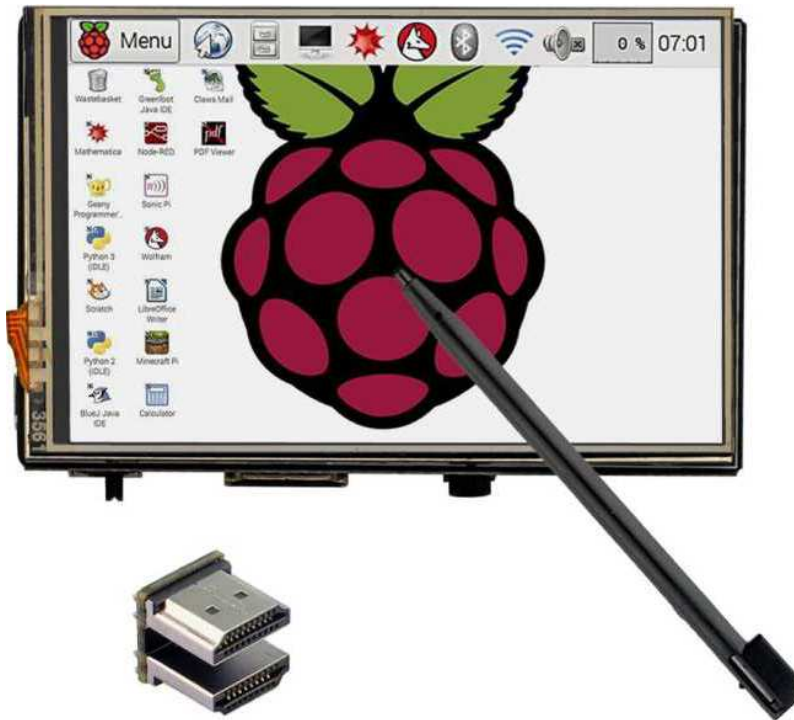


Bild 3: RasPi mit 3,5“ Touch Screen

- (1) Micro-SD-Karte mit installiertem Betriebssystem ins RasPi einstecken, RasPi mittels Kabel mit dem Netzwerk verbinden, Spannungsversorgung einschalten.
- (2) Falls die grafische Oberfläche erscheint: Ein Terminalfenster starten.
- (3) Ins Home-Verzeichnis des Users wechseln und 3.5“ HDMI touch screen Treiber herunterladen.

```
cd ~
wget http://osoyoo.com/driver/LCD_show_35hdmi.tar.gz
```

- (4) Rechte der Datei ändern:

```
sudo chmod 777 LCD_show_35hdmi.tar.gz
```

- (5) Datei entpacken:

```
tar -xvzf LCD_show_35hdmi.tar.gz
```

- (6) Ins entstandene Treiber-Verzeichnis wechseln:

```
cd ~/LCD_show_35hdmi
```

- (7) System updaten (optional):

```
sudo apt-get update  
sudo apt-get upgrade
```

- (8) Treiber sichern (optional):

```
sudo ./LCD_backup
```

- (9) Raspberry hinunterfahren, Spannung abstecken.

```
sudo shutdown -h 0
```

- (10) 3.5" HDMI touch screen auf das RasPi aufstecken, eventuell in ein vorhandenes Gehäuse einbauen, HDMI-Verbindung anstecken, Spannung einschalten.

- (11) Treiber einstellen durch Eingabe von **einem der drei** Befehle (die Zahlen geben die gewünschte Auflösung an):

```
sudo ~/LCD_show_35hdm/LCD35_480*320  
sudo ~/LCD_show_35hdm/LCD35_720*480  
sudo ~/LCD_show_35hdm/LCD35_810*540
```

Nach einiger Zeit ist der Treiber installiert und das Display startet neu.

Einige Tipps

Will man die vorherige Installation wieder herstellen, so muss man folgendes eingeben:

```
sudo ~/LCD_show_35hdm/LCD_restore
```

Zum Ändern der Schriftgröße des Konsolenfensters (zB von 10pt auf 12pt) muss, zumindest während des Einstellens, die Auflösung auf zB den doppelten Wert der physikalischen Auflösung eingestellt werden (960x640), da man sonst den [OK] Button nicht betätigen kann.

Eine Änderung der Auflösung kann in der Datei `/boot/config.txt` erfolgen (im Beispiel eingestellte Auflösung ist 480 x 320):

```
sudo nano /boot/config.txt
```

Folgende Zeile zB anpassen auf 960 640 oder 720 480

```
hdmi_cvt 480 320 60 6 0 0 0
```

Nach einer Änderung in der Datei muss das RasPi mittels `sudo reboot` neu gestartet werden.

Andere Einstellungen, wie zB die Tastaturbelegung, kann man auch in der Konsole durch Aufruf der Konfigurationsdatei vornehmen.

```
sudo raspi-config
```

10 Anhang 3: Weitere Raspi-Einstellungen

10.1 Eigene Autostart-Datei `autostart.sh`

In der Datei `/etc/rc.local` vor `exit 0` folgende Zeilen einfügen
(zB durch Eingabe von `sudo nano /etc/rc.local`):

```
#-----Aufruf eines Scripts mit eigenen Befehlen-----  
/usr/local/bin/autostart.sh
```

Speichern und beenden durch <Strg> o <Enter> <Strg> x

Datei (`sudo nano`) `/usr/local/bin/autostart.sh` erzeugen

```
#!/bin/bash  
#...Farbe der Schrift auf gelb ändern...  
echo -e "\033[01;33m"  
printf "____autostart.sh____24.10.2017____Karl Hartinger____\n"  
printf "_____\n"  
#...Farbe der Schrift wieder auf weiß ändern...  
echo -e "\033[00m"  
exit 0
```

Tipp: Soll beim Start einer Datei nicht auf das Ende eines Programmes gewartet werden, so muss am Ende der Aufruf-Zeile ein `&` stehen!

Script ausführbar machen:

```
$ sudo chmod 777 /usr/local/bin/autostart.sh
```

10.2 Beispiel für Verzeichnisse und weitere Programme

Folgende Befehle eingeben (und dazwischen unter Umständen einige Minuten warten... ;).

```
mkdir /home/pi/src_c  
mkdir /home/pi/src_cpp  
mkdir /home/pi/src_python  
mkdir /home/pi/hardwaretest  
  
sudo apt-get install bash-completion  
sudo apt-get install openssh-server  
sudo apt-get install python  
sudo apt-get install g++  
sudo apt-get install apache2  
sudo apt-get install php  
  
sudo apt-get update  
sudo apt-get upgrade  
sudo apt-get autoremove  
sudo reboot
```


10.3 Bearbeiten von Konfigurationsdateien

- (1) Automatisches Vervollständigen von Eingaben.
In der Datei `/etc/bash.bashrc` die Kommentarzeichen `#` entfernen
(zB durch Eingabe von `sudo nano /etc/bash.bashrc`):

```
# enable bash completion in interactive shells
if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
    . /etc/bash_completion
fi
```

Speichern und beenden durch `<Strg> o` `<Enter>` `<Strg> x`

- (2) PHP-Zugriff auf die Hardware mit dem PHP-Befehl `shell_exec()` ermöglichen.
(Falls `pi` geändert wurde, die Befehle entsprechend anpassen)

```
pi@raspberrypi:~$ sudo chown -R pi:www-data /var/www
pi@raspberrypi:~$ sudo chown -R pi:www-data /var/www/html
pi@raspberrypi:~$ sudo adduser www-data gpio
pi@raspberrypi:~$ sudo adduser www-data sudo
```

10.4 WiringPi installieren

WiringPi ist eine Bibliothek für den einfachen Zugriff auf die Ein-/Ausgabepins des Raspberrys (incl. I2C-Bus usw.) Der Dateiname ist je nach aktueller Version unterschiedlich.

- (1) GIT installieren

```
pi@raspberrypi ~ $ sudo apt-get install git-core
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
```

- (2) WiringPi mittels GIT herunterladen:

```
pi@raspberrypi ~ $ git clone git://git.drogon.net/wiringPi
```

- (3) WiringPi installieren:

```
pi@raspberrypi ~ $ cd wiringPi
pi@raspberrypi ~/wiringPi $ ./build
```

2. Möglichkeit:

Funktioniert die Installation auf diese Weise nicht (weil zB eine Firewall das git-Protokoll blockiert), kann man die wiringPi-Datei auch mit einem Browser herunterladen und manuell entpacken.

- (1) Falls man sich nicht in der grafische Oberfläche befindet => diese starten:

```
pi@raspberrypi ~ $ startx
```

- (2) In der grafischen Oberfläche den Browser öffnen (Button links oben) und folgenden Link eingeben:

```
https://git.drogon.net/?p=wiringPi;a=summary
```

Auf der rechten Seite befinden sich Links mit dem Namen „snapshot“ → Auf den obersten dieser Links klicken, startet den Download. Die Datei steht im Verzeichnis /home/pi/Downloads. Dateiname am 3.4.2016: wiringPi-b0a60c3

- (3) In eine Konsole wechseln (zB <Strg>>Alt><F2>) und die Installation der Bibliothek durchführen:

```
~ $ cd ~
~ $ cp ~/Downloads/wiringPi-b0a60c3.tar.gz ~
~ $ tar xzf wiringPi-b0a60c3.tar.gz
~ $ cd wiringPi-b0a60c3
~/wiringPi-b0a60c3 $ ./build
```

Nicht vergessen: Der aktuelle Dateiname wird unterschiedlich sein – bitte anpassen ;)

Eine Alternative zu wiringPi ist die Bibliothek **pigpio**, die wesentlich mehr Möglichkeiten zur Verfügung stellt.

10.5 I2C installieren

Ab Raspbian Februar 2016 kann der I2C-Bus auch über die graphische Oberfläche freigeschaltet werden: Am Bildschirm links oben [Menu] anklicken – Einstellungen – Raspberry Pi Configuration auswählen, Reiter „Interfaces“ I2C: ☒ Enable.

Natürlich können die Einstellungen auch - wie bei älteren Versionen - in der Konsole durchgeführt werden (siehe unten).

Ältere Raspbian-Versionen

Von Raspbian Version 3.18 (2015) bis Februar 2016 kann der I2C-Bus über das Konfigurations-Tool freigeschaltet werden:

```
pi@raspberrypi ~ $ sudo raspi-config
```

► 8 Advanced Options markieren, mit Tabulator auf <Select> <Enter>
A7 I2C Enable/Disable automatic loading of I2C kernel module
mit Tabulator auf <Select> <Enter>, Would you like the ARM I2C Interface to be enabled
mit <Ja> abschließen, <OK> <Ja> <OK>.

Danach das RasPi neu booten:

```
pi@raspberrypi ~ $ sudo reboot
```

Für noch ältere Versionen muss in der Blacklist `/etc/modprobe.d/raspi-blacklist.conf` den für den I2C-Bus benötigten Kernel-Modul auskommentiert werden (# vor folgende Zeile setzen):

```
#blacklist i2c-bcm2708
```

Speichern und beenden durch <Strg>o <Enter> <Strg>x

Module bei jedem Neustart automatisch laden durch Eintrag in `/etc/modules`

```
# /etc/modules: kernel modules to load at boot time.
#
# This file contains the names of kernel modules that should be loaded
# at boot time, one per line. Lines beginning with "#" are ignored.
# Parameters can be specified after the module name.
snd-bcm2835
i2c_bcm2708
i2c-dev
```

Installation von I2C-Zusatztools

Die Installation erfolgt von der Kommandozeile aus (eventuell <Strg><Alt><F2> drücken und einloggen):

- (1) Das folgende Skript installiert i2c-Zusatz-Tools. Dazu das Skript mit nano eingeben (oder mittels **WinSCP** auf das RasPi kopieren)

```
pi@raspberrypi ~ $ nano ~/myInstall_i2c.sh
```

Skript-Inhalt:

```
#!/bin/bash
#_____myInstall_i2c.sh_____
sudo modprobe i2c_bcm2708
sudo modprobe i2c-dev
sudo chmod 666 /dev/i2c-1
sudo apt-get install python-smbus
sudo apt-get install i2c-tools
sudo apt-get install libi2c-dev
sudo adduser pi i2c
printf "Wurden die beiden Module i2c_bcm2708 und i2c-dev geladen?...\\n"
lsmod
printf "Ist user pi Mitglied der i2c-Gruppe?...\\n"
sudo groups pi
exit 0
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Anmerkung: i2c-1 ab Revision 2, sonst i2c-0

- (2) Skript ausführbar machen und starten.

```
pi@raspberrypi ~ $ sudo chmod 777 ~/myInstall_i2c.sh
pi@raspberrypi ~ $ sudo ./myInstall_i2c.sh
```

Frage mit J beantworten.

- (3) Das RasPi aktualisieren und neu starten

```
pi@raspberrypi ~ $ sudo apt-get update
pi@raspberrypi ~ $ sudo apt-get upgrade
pi@raspberrypi ~ $ sudo reboot
```

- (4) Kontrolle, ob es das Interface i2c-1 gibt:

```
pi@raspberrypi ~ $ ls /dev/i2c*
/dev/i2c-1
```

- (5) Feststellen, ob ein I2C-Device an I2C-1 angeschlossen ist:

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
   0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
10:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
20: 20  --  --  --  --  --  --  --  --  --  --  --  --  --  --
30:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
40:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
50:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
60:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
70:  --  --  --  --  --  --  --  --  --  --  --  --  --  --  --
```

Es wurde ein IC mit der hexadezimalen 7-Bit-Adresse 0x20 (= 0100 000x) entdeckt. Dies entspricht beim Schreiben dem Adressbyte 0x40 bzw. beim Lesen 0x41 (!).

10.6 Test der Hardware mit eigenen Programmen

- (1) Mit Hilfe der folgenden Programme kann die Installation und die Hardware überprüft werden. Zuerst muss das Skript `make_hardwaretest.sh` erstellt werden:

```
pi@raspberrypi ~ $ sudo rm ~/hardwaretest/make_hardwaretest.sh
pi@raspberrypi ~ $ sudo nano ~/hardwaretest/make_hardwaretest.sh
```

Inhalt des Skripts:

```
#!/bin/bash
printf "____make_hardwaretest.sh____7.10.2015____Karl Hartinger____\n"
cd ~/hardwaretest
gcc rpi_out_dio.c -o rpi_out_dio -l wiringPi
gcc rpi_out_dio16.c -o rpi_out_dio16 -l wiringPi
gcc rpi_out_8574.c -o rpi_out_8574 -l wiringPi
gcc rpi_rs232_sendrec_dev.c -o rpi_rs232_sendrec_dev -l wiringPi
sudo cp ./rpi_out_dio /usr/local/bin
sudo cp ./rpi_out_dio16 /usr/local/bin
sudo cp ./rpi_out_8574 /usr/local/bin
sudo cp ./rpi_rs232_sendrec_dev /usr/local/bin
sudo chown root /usr/local/bin/rpi_out_dio
sudo chmod u+s /usr/local/bin/rpi_out_dio
sudo chown root /usr/local/bin/rpi_out_dio16
sudo chmod u+s /usr/local/bin/rpi_out_dio16
sudo chown root /usr/local/bin/rpi_out_8574
sudo chmod u+s /usr/local/bin/rpi_out_8574
sudo chown root /usr/local/bin/rpi_rs232_sendrec_dev
sudo chmod u+s /usr/local/bin/rpi_rs232_sendrec_dev
sudo cp whoami.php /var/www/html
sudo cp rpi_out_dio.php /var/www/html
printf "fertig :)\n"
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Nach Eingabe der ersten Zeile kann man den restlichen Text auch unter Verwendung von putty und der rechten Maustaste in den nano-Editor hineinkopieren und dann speichern.

- (2) Kopieren Sie den Quellcode der Dateien mit Hilfe von **WinSCP** ins Verzeichnis `hardwaretest`
- (3) Machen Sie das Skript ausführbar und starten Sie es.

```
pi@raspberrypi ~ $ sudo chmod 777 ~/hardwaretest/make_hardwaretest.sh
pi@raspberrypi ~ $ sudo ~/hardwaretest/make_hardwaretest.sh
____make_hardwaretest.sh____14.3.2015____Karl Hartinger____
fertig :)
```

Die Dateien werden kompiliert, ins Verzeichnis `/usr/local/bin` kopiert und dort für alle Benutzer freigegeben. Die php-Dateien werden ins Verzeichnis `/var/www/html` kopiert. Die Codierung der Programme befindet sich im Anhang.

GPIO mit wiringPi

Empfohlene Hardware: LEDs an den GPIOs des 26poligen/40poligen Steckers.

Das Kommandozeilenprogramm `rpi_out_dio` ist für RasPis mit 26poligem Stecker gedacht und ermöglicht die Ausgabe eines 8- oder 10-Bit-Wertes auf die GPIO des RasPis. Als Parameter wird eine 2- oder 3-stellige Hex-Zahl erwartet.

Beispiel: `rpi_out_dio 000` schaltet alle Ausgangspins aus.

Das Kommandozeilenprogramm `rpi_out_dio16` ermöglicht die Ausgabe eines 16-Bit-Wertes auf die GPIO eines RasPis mit 40poligem Stecker.

I2C mit wiringPi

Benötigte Hardware: Ein am I2C-Bus angeschlossener I/O-Expander PCF8574P. Voreingestellt ist die 7-Bit-Adresse 0x20, es kann aber auch eine andere Adresse gewählt werden.

Das Kommandozeilenprogramm `rpi_out_8574` ermöglicht die Ausgabe eines Bytes auf einen am I2C-Bus angeschlossenen I/O-Expander PCF8574P.

Beispiel: `rpi_out_8574 00` schaltet alle Ausgangspins beim 8574 Adresse 0x20 aus.
`rpi_out_8574 AA 21` schaltet jeden 2. Pin beim 8574 mit Adresse 0x21 ein.

PHP-Web-Seite abrufen

Benötigte Hardware: PC mit Browser

Die Webseite `whoami.php` zeigt den WWW-Username am Raspberry an. Dazu ist im Browser folgende Adresse einzugeben:

`192.168.0.155/whoami.php`

Die Antwort sollte „WWW-Username auf dem RasPi: www-data“ sein.

PHP-Web-Seite mit Hardware-Zugriff

Die Webseite `rpi_out_dio.php` ermöglicht die Ausgabe eines 8- oder 10-Bit-Wertes auf die GPIO des RasPis Version 1 (26poliger Stecker). Als Parameter wird eine 2- oder 3-stellige Hex-Zahl erwartet. Zur Ansteuerung der Hardware wird das Programm verwendet. Der Aufruf im Browser erfolgt durch Eingabe von

`192.168.0.155/rpi_out_dio.php`

Auf neuere RasPi-Modelle mit 40poligem GPIO-Stecker kann ein 16-Bit-Wert durch folgende Eingabe im Browser ausgegeben werden:

`192.168.0.155/rpi_out_dio16.php`

Serielle Schnittstelle

Benötigte Hardware: Ein am USB angeschlossener USB-RS232-Adapter, ausgekreuztes Kabel und ein PC mit serieller Schnittstelle (oder USB-RS232-Adapter ;) mit laufendem Terminal-Programm. Das Programm `rpi_rs232_sendrec_dev.c` zählt von 10 Sekunden sekundenweise herunter und sendet die verbleibende Zeit als String „10 second(s) left to end of program.“ über die serielle Schnittstelle. Ist Null erreicht, beendet sich das Programm. Während des Programmlaufs werden ankommende Zeichen am RasPi-Bildschirm dargestellt.

Voreingestellt sind die Schnittstelle (Device) `/dev/ttyUSB0` sowie die Baudrate 9600.

Aufruf: `rpi_rs232_sendrec_dev`

oder `rpi_rs232_sendrec_dev /dev/ttyUSB0 19200`

10.7 Netzwerk konfigurieren beim Raspberry Pi 1 und 2

Raspberry 1 und 2: Netzwerk-Konfiguration

Nach dem ersten Systemstart ist das RasPi auf DHCP (Dynamic Host Configuration Protocol) eingestellt, das heißt, es erwartet von einem DHCP-Server eine IP-Adresse.

Die Einstellung einer IP-Adresse erfolgt in der Datei `/etc/network/interfaces`, die nach der Original-Installation folgendermaßen aussieht (Linux-Befehl `cat /etc/network/interfaces`)

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface default inet dhcp
```

Da die Lampensteuerung **nur über WLAN** betrieben werden soll, sollte nur die Schnittstelle `wlan0` mit einer fixen IP in der `interfaces`-Datei aktiviert werden. Dazu ist die bestehende Konfigurationsdatei durch kopieren zu sichern. Danach kann sie mit dem Texteditor `nano` bearbeitet werden. Die folgende Musterdatei enthält zusätzliche (mit # auskommentierte) mögliche LAN-Konfigurationen.

```
pi@raspberrypi ~ $ cd /etc/network
pi@raspberrypi /etc/network $ sudo cp interfaces interfaces_original
pi@raspberrypi /etc/network $ sudo nano interfaces
```

Um den neuerlichen WLAN-Verbindungsaufbau nach einem Router-Ausfall sicherzustellen, sollten die WLAN-Netzwerkeigenschaften in einer eigenen Datei stehen, dh, es werden die Daten auf die zwei Dateien `/etc/network/interfaces` (IP-Adressen, ...) und `wpa_supplicant.conf` (Name des WLANs, Passwort, ...) aufgeteilt.

Hat nur der Root Leserechte bei der 2. Datei, ist das WLAN-Passwort, das in dieser Datei unverschlüsselt steht, auch vor unerlaubtem Zugriff geschützt.

Raspberry 1 und 2: WLAN-Konfiguration mit `wpa_supplicant.conf`

Mit `nano` in die Datei `/etc/network/interfaces` eingeben:

```
#_____/etc/network/interfaces_____1.12.2015_____KH_____
# ---Loop back interface---
auto lo
iface lo inet loopback

# ---LAN = primary network interface---
#auto eth0
##iface eth0 inet dhcp
#iface eth0 inet static
#address 192.168.0.55
#netmask 255.255.255.0
#gateway 192.168.0.1
#network 192.168.0.0
#broadcast 192.168.0.255
# ---Wireless network interface---
auto wlan0
```

```
allow-hotplug wlan0
iface wlan0 inet manual
wpa-roam /etc/wpa_supplicant/wpa_supplicant.conf
iface Wifi inet static
address 192.168.0.155
netmask 255.255.255.0
gateway 192.168.0.1
#iface default inet dhcp
```

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Mit nano in die Datei /etc/wpa_supplicant/wpa_supplicant.conf eingeben:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
network={
    id_str="Wifi"
    ssid= "meine_ssid_netzwerk"
    proto=RSN
    key_mgmt=WPA-PSK
    pairwise=CCMP TKIP
    psk="mein_passwort"
}
```

Nicht vergessen: Die Angaben bei ssid und psk an das eigene WLAN anpassen!

Speichern und beenden durch <Strg>o <Enter> <Strg> x

Mit nano in die Datei /etc/rc.local vor exit 0 eingeben:

```
ifup wlan0
```

Jetzt noch eine Sicherheitskopie der neuen Netzwerkkonfiguration anlegen, den USB-WLAN-Stick anstecken und das Raspberry neu starten, damit die Netzwerkadresse wirksam wird.

```
pi@raspberrypi /etc/network $ sudo cp interfaces interfaces_155
pi@raspberrypi /etc/network $ sudo reboot
```

Kontrolle der IP-Adresse nach dem Reboot und Einloggen:

```
pi@raspberrypi ~ $ ifconfig
eth0      ...
wlan0     Link encap:Ethernet  Hardware Adresse: 80:1f:xx:xx:xx:xx
          inet addr:192.168.0.155 Bcast:192.168.0.255 Mask:255.255.255.0
```

Raspberry 1 und 2: Bevorzugte Netzwerkverbindung einstellen

Benötigt man beide Schnittstellen, so muss man die Kommentarzeichen # vor den LAN-Konfigurationszeilen entfernen. Die bevorzugte Verbindung (wlan0 oder eth0) kann mit Hilfe des Befehls route angezeigt und auch geändert werden:

```
pi@raspberrypi ~ $ route
Kernel-IP-Routentabelle

```

Ziel	Router	Genmask	Flags	Metric	Ref	Use	Iface
default	192.168.0.1	0.0.0.0	UG	0	0	0	wlan0
192.168.0.0	*	255.255.255.0	U	0	0	0	eth0
192.168.0.0	*	255.255.255.0	U	0	0	0	wlan0

Default-Verbindung von eth0 auf wlan0 umstellen:

```
pi@raspberrypi ~ $ sudo ifdown eth0
pi@raspberrypi ~ $ sudo ifup wlan0
```



```
pi@raspberrypi ~ $ sudo route del default
pi@raspberrypi ~ $ sudo route add default gw 192.168.0.1 wlan0
```

Kontrolle der Umstellung:

```
pi@raspberrypi ~ $ route
pi@raspberrypi ~ $ ping -c 1 8.8.8.8
pi@raspberrypi ~ $ ping -c 1 www.google.com
```

Domain Name Server (DNS)

Die Einstellung der Domain Name Server (DNS), die verwendet werden sollen, erfolgt durch

```
$ sudo nano /etc/resolv.conf
```

Inhalt der Datei (zB mit Google-Server) zB

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

oder

```
domain fritz.box
search fritz.box
nameserver 192.168.0.1
```