



# PCT-Pi-Remote - Webserver - Dokumentation

<b>Version:</b>	3.0.2
<b>Datum:</b>	11.11.2023
<b>Autor:</b>	Pascal Christian Thiede
<b>Copyright © PCT-Software</b>	

## Inhaltsverzeichnis / Table of Contents

Bezeichnung / Description	Seite / Page
<b>German</b>	
Installation	3
Webserver manuell ausführen	3
Webserver beim Start immer automatisch ausführen lassen	4
Zugriff auf Webserver über den Browser	5
Was wird ausgelöst, wenn man einen Button betätigt?	7
Zugriff über die iOS- oder Android-App	7
Webserver-Konfiguration über „_config.txt“	8
GPIO-Darstellung	12
Raspberry Pi Relays	13
WiringPi - GPIO	15
<b>English</b>	
Setup	17
Run the web server manually	17
Always start the web server automatically at startup	18
Access to web servers via the browser	19
What is triggered when you press a button?	21
Access via the iOS or Android app	21
Web server configuration via "_config.txt"	22
GPIO illustration	26
Raspberry Pi Relays	27
WiringPi - GPIO	29

**Wir übernehmen keine Haftung für Schäden, die aufgrund der Anwendung der auf unserer Seite genannten Informationen entstehen.**

**We do not assume liability for disadvantages accruing from the appliance of any information provided on our website.**

## Installation

Startet euren Raspberry Pi und meldet euch mit dem Benutzer „root“ oder „pi“ an. In unserem Beispiel verwenden wir immer den Benutzer „pi“.

Zu allererst müssen wir die Bibliothek „mono“ installieren. Um sicherzustellen, dass euer Raspberry Pi auf den aktuellsten Stand ist, gebt folgendes ein:

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

Dann installieren wir „mono“:

```
sudo apt-get install mono-complete
```

Nachdem „mono“ installiert wurde, gebt in der Konsole folgendes ein, um ins Home-Verzeichnis des Benutzers zu gelangen:

```
cd ~
```

Jetzt erstellt Ihr ein neues Verzeichnis mit dem Namen „webserver“:

```
mkdir webserver
```

Wechselt nun in das neue Verzeichnis „webserver“:

```
cd webserver
```

Falls noch nicht installiert, müsst ihr das Tool „wget“ installieren:

```
sudo apt-get install wget
```

Um nun den „PCT-Pi-Remote - Webserver“ in das aktuelle Verzeichnis herunterzuladen, benutzt ihr folgenden Befehl:

```
wget "http://pct.company/downloads/PCT-Pi-Remote/Webserver/PCT-Pi-Remote - Webserver.exe"
```

Nachfolgend müsst ihr die zugehörige „\_config.txt“ downloaden:

```
wget "http://pct.company/downloads/PCT-Pi-Remote/Webserver/_config.txt"
```

## Webserver manuell ausführen

Nach der Installation könnt ihr den Webserver manuell ausführen lassen:

```
sudo mono "PCT-Pi-Remote - Webserver.exe"
```

Mit STRG + C könnt ihr das Programm verlassen. Nach einem Neustart eures Raspberry Pi würde der Webserver nicht automatisch gestartet werden.

## Webserver beim Start immer automatisch ausführen lassen

Damit der Webserver bei jedem Neustart des Raspberry Pi automatisch mit dem System startet, gehen öffnen wir die Datei „/etc/rc.local“:

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

Vor dem „exit 0“ fügt ihr die Zeile „su pi -c 'sudo mono "/home/pi/webserver/PCT-Pi-Remote - Webserver.exe"'“ ein. Am besten kopiert ihr diese so wie sie ist, weil man auch die Hochkommata achten muss. Hier gibt es je nach „automatischer“ Formatierung sonst Probleme. Wichtig ist zudem, dass das Verzeichnis stimmt, falls ihr zuvor einen anderen Benutzer oder Verzeichnis ausgewählt habt.

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

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

su pi -c 'sudo mono "/home/pi/webserver/PCT-Pi-Remote -
Webserver.exe"'

exit 0
```

## Zugriff auf Webserver über den Browser

Über die IP-Adresse eures Raspberry Pi, den Port 18181 und den Suffix /pi, könnt ihr auf den Webserver mit jeden beliebigen Browser zugreifen.

Beispiel: <http://192.168.2.99:18181/pi>

Wenn ihr an euren Raspberry Pi angemeldet seid, kriegt ihr die IP-Adresse über den befehl „ifconfig“ raus.

Die Ausgabe sollte dann in etwa so aussehen:

```
pi@PCT-Pi-Garage:~ $ ifconfig
eth0      Link encap:Ethernet  Hardware Adresse b8:27:eb:30:d2:f4
          inet6-Adresse: fe80::50ca:dc5e:a06a:3edf/64  Gültigkeitsbereich:Verbindung
          UP BROADCAST MULTICAST  MTU:1500  Metrik:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Lokale Schleife
          inet Adresse:127.0.0.1  Maske:255.0.0.0
          inet6-Adresse: ::1/128  Gültigkeitsbereich:Maschine
          UP LOOPBACK RUNNING  MTU:65536  Metrik:1
          RX packets:137 errors:0 dropped:0 overruns:0 frame:0
          TX packets:137 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1
          RX bytes:11551 (11.2 KiB)  TX bytes:11551 (11.2 KiB)

wlan0     Link encap:Ethernet  Hardware Adresse 74:da:38:00:5f:0e
          inet Adresse:192.168.2.99  Bcast:192.168.2.255  Maske:255.255.255.0
          inet6-Adresse: fe80::76da:38ff:fe00:5f0e/64  Gültigkeitsbereich:Verbindung
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metrik:1
          RX packets:4264 errors:0 dropped:152 overruns:0 frame:0
          TX packets:985 errors:0 dropped:2 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:517907 (505.7 KiB)  TX bytes:288917 (282.1 KiB)
```

eth0 steht für den Netzwerkport und wlan0 für das W-LAN.

Wie man den Raspberry Pi mit dem Netzwerk verbindet, findet man im Internet.

Der Webserver wird nach einer Authentifizierung fragen. Standardmäßig ist hier als Benutzername: admin und als Passwort: admin hinterlegt. Ihr könnt beliebig viele Benutzer in der „\_config.txt“ hinterlegen.

**Bitte melden Sie sich an**

Für `http://192.168.2.99:18181` sind ein Nutzernamen und ein Passwort erforderlich.

Die Verbindung zu dieser Website ist nicht sicher

Benutzername:

Passwort:

Wenn ihr erfolgreich mit dem Webserver verbunden seid, dann sieht die Anzeige so aus:



## PCT-Pi-Remote - Webserver



**GPIO\_4**

Touch

**CUSTOM\_1**

Touch

**LinuxCommand\_1**

Touch

**GPIO\_7**

Touch

**CUSTOM\_2**

Touch

**LinuxCommand\_2**

Touch

**GPIO\_8**

Touch

**CUSTOM\_3**

Touch

**LinuxCommand\_3**

Touch

Diese Buttons kann man dann von jedem Browser aus betätigen. Jeder Button ist mit einer Befehlszeile in der „\_config.txt“ verknüpft. Daher kann jeder selbst entscheiden, was bei einer Betätigung passieren soll.

## Was wird ausgelöst, wenn man einen Button betätigt?

Wenn man einen Button betätigt, wird z.B. folgender Befehl ausgelöst:

[http://192.168.2.99:18181/pi/GPIO\\_4](http://192.168.2.99:18181/pi/GPIO_4)

Diese Befehle könnt ihr auch einfach so als Verknüpfung hinterlegen und starten. In Apps von Drittanbietern werden ebenfalls diese Befehle benötigt. Was letzten Endes ausgeführt wird, wenn ihr den Button betätigt, konfiguriert ihr in der „\_config.txt“.

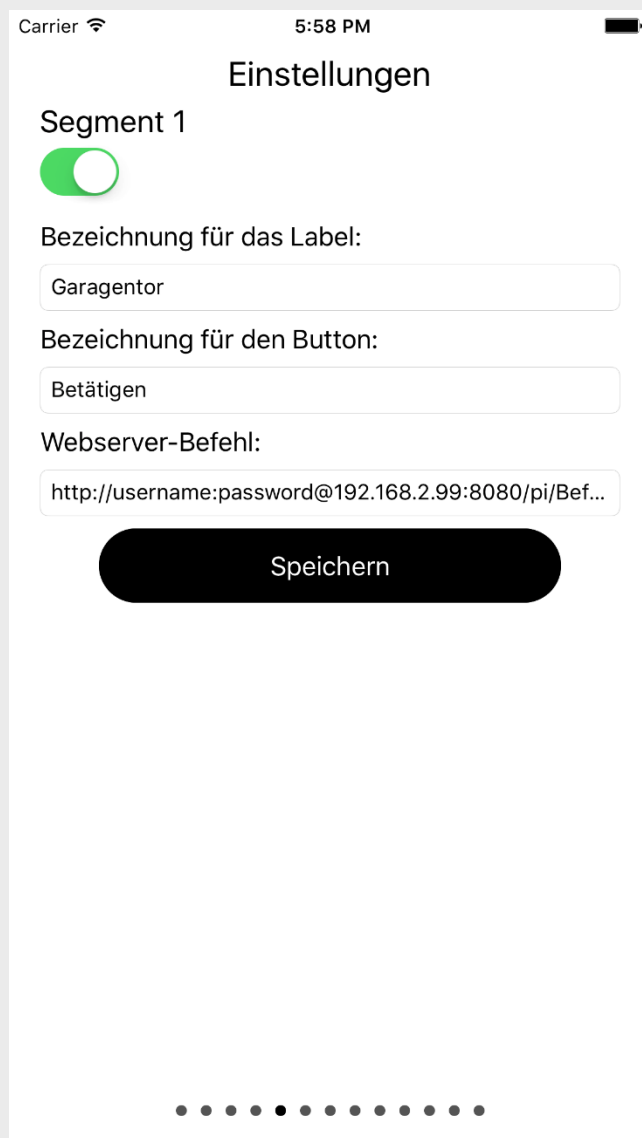
## Zugriff über die iOS- oder Android-App

In den Apps „PCT-Pi-Remote“ gibt es für jedes Segment eine Konfigurationsseite.

Auf dieser Seite gebt ihr unter „Webserver-Befehl“ z.B. folgendes ein:

[http://Benutzername:Passwort@192.168.2.99:18181/pi/GPIO\\_4](http://Benutzername:Passwort@192.168.2.99:18181/pi/GPIO_4)

Den Benutzernamen und das Passwort könnt ihr in der „\_config.txt“ anpassen.



The screenshot shows the 'Einstellungen' (Settings) screen for 'Segment 1' in the PCT-Pi-Remote app. At the top, the status bar shows 'Carrier', signal strength, '5:58 PM', and battery level. The title 'Einstellungen' is centered. Below it, 'Segment 1' is displayed with a green toggle switch that is turned on. There are three text input fields: 'Bezeichnung für das Label:' with the value 'Garagentor', 'Bezeichnung für den Button:' with the value 'Betätigen', and 'Webserver-Befehl:' with the value 'http://username:password@192.168.2.99:8080/pi/Bef...'. A large black button labeled 'Speichern' (Save) is at the bottom. At the very bottom of the screen, there is a row of ten small grey dots, with the fifth dot from the left being filled, indicating the current segment.

## Webserver-Konfiguration über „\_config.txt“

In dieser Datei kann man den Webserver an seine Projekte anpassen. So kann man hier z.B. die Benutzer verwalten und entscheiden, was passiert, wenn man bestimmte Befehle über den Webserver im Browser oder über die Apps auslöst.

Wenn wir mit dem Raspberry Pi verbunden sind, gehen wir ins Homeverzeichnis:

```
cd ~
```

Danach navigieren wir in das zuvor erstellte Verzeichnis „webserver“:

```
cd webserver
```

Anschließend öffnen wir mit nano die „\_config.txt“:

```
sudo nano _config.txt
```

Generell kann diese Datei auch bearbeitet werden, wenn der Webserver gestartet ist. Dieser muss also nicht extra beendet werden.

In diesem Bereich werden die Benutzernamen und die Passwörter hinterlegt. Wichtig ist, dass die Benutzer mit einem Semikolon getrennt werden und die Anzahl der Benutzernamen muss mit der Anzahl der Passwörter übereinstimmen.

```
-----  
// Benutzernamen / Usernames  
// Beispiel / Example  
// admin;admin2;admin3  
admin;PCT  
  
// Passwörter / Passwords  
// Beispiel / Example  
// password;password2;password3  
admin;PCT  
-----
```



Weiter unten kann man für jeden der 13 GPIO-Befehle hinterlegen, was beim Auslösen dieser Befehle passieren soll.

```
// GPIO_4 - PIN_7  
Out;1000;In
```

Die Befehle können generell beliebig lang sein. Out bedeutet, dass ein Port auf HIGH gestellt wird. In bedeutet, dass ein Port auf LOW geschaltet wird. Toggle invertiert die Port-Stellung. Wenn man eine Ganzzahl eingibt, ist das die Wartezeit in Millisekunden.

```
-----  
  
// GPIO-Befehle / GPIO-Commands  
// Beispiel / Example  
// Out;1000;In  
// Der GPIO wird auf HIGH gestellt, dann wird 1000ms gewartet und dann wird d  
// Alle Befehle: "Out": Schaltet einen Port auf HIGH; "Zahlenwert ohne Kommat  
  
// GPIO_4 - PIN_7  
Out;1000;In  
  
// GPIO_7 - PIN_26  
Out;1000;In  
  
// GPIO_8 - PIN_24  
Out;1000;In  
  
// GPIO_9 - PIN_21  
Out;1000;In  
  
// GPIO_10 - PIN_12  
Out;1000;In  
  
// GPIO_11 - PIN_23  
Out;1000;In  
  
// GPIO_14 - PIN_8  
Out;1000;In  
  
// GPIO_15 - PIN_10  
Out;1000;In  
  
// GPIO_18 - PIN_12  
Out;1000;In  
  
// GPIO_22 - PIN_15  
Out;1000;In  
  
// GPIO_23 - PIN_16  
Out;1000;In  
  
// GPIO_24 - PIN_18  
Toggle  
  
// GPIO_25 - PIN_22  
Out;1000;In  
  
-----
```

Damit man in seinem Projekt auch komplexere Sachen umsetzen kann, gibt es noch die Benutzerdefinierten Befehle.

```
-----  
// Benutzerdefinierte-Befehle / Custom-Commands  
// Beispiel / Example  
// GPIO_23: Out;1000;GPIO_24_Out;1000;GPIO_23: I  
// Erst wird der GPIO_23 auf HIGH gestellt, dann  
// Alle Befehle: "GPIO_XY: Out": Schaltet den Po  
  
// CUSTOM_1  
GPIO_24: Toggle;2000;GPIO_24: Toggle  
  
// CUSTOM_2  
GPIO_24: Toggle;2000;GPIO_24: Toggle  
  
// CUSTOM_3  
GPIO_24: Toggle;2000;GPIO_24: Toggle  
  
// CUSTOM_4  
GPIO_24: Toggle;2000;GPIO_24: Toggle
```

Wenn man folgenden Befehl auslöst, wird der Befehl „CUSTOM\_1“ ausgeführt:  
[http://Benutzername:Passwort@192.168.2.99:18181/pi/CUSTOM\\_1](http://Benutzername:Passwort@192.168.2.99:18181/pi/CUSTOM_1)

Man kann hier nur die Ports GPIO\_4, GPIO\_7, GPIO\_8, GPIO\_9, GPIO\_10, GPIO\_11, GPIO\_14, GPIO\_15, GPIO\_18, GPIO\_22, GPIO\_23, GPIO\_24 und GPIO\_25 verwenden. Also genau die Ports, die auch als normale Befehle direkt angesprochen werden können.

Befehle werden auch hier mit einem Semikolon voneinander getrennt. Ganzzahlen sind Wartezeiten. Ansonsten kann man den Port: Befehl nutzen, wie z.B:

```
GPIO_24: Out  
GPIO_24: In  
GPIO_24: Toggle
```

Beispiel welches den Port GPIO\_23 und GPIO\_24 auf HIGH stellt, 10 Sekunden wartet und beide Ports wieder auf LOW stellt:

```
GPIO_23: Out;GPIO_24: Out;10000;GPIO_23: In;GPIO_24: In
```

Seit der Version 2.0.0 kann man nun auch Befehle in der Linux Konsole auslösen lassen. Dies ist z.B. notwendig, wenn man Steckdosen über 433MHz steuern möchte und hierzu eine Fremdsoftware auf dem Raspberry Pi nutzen muss.

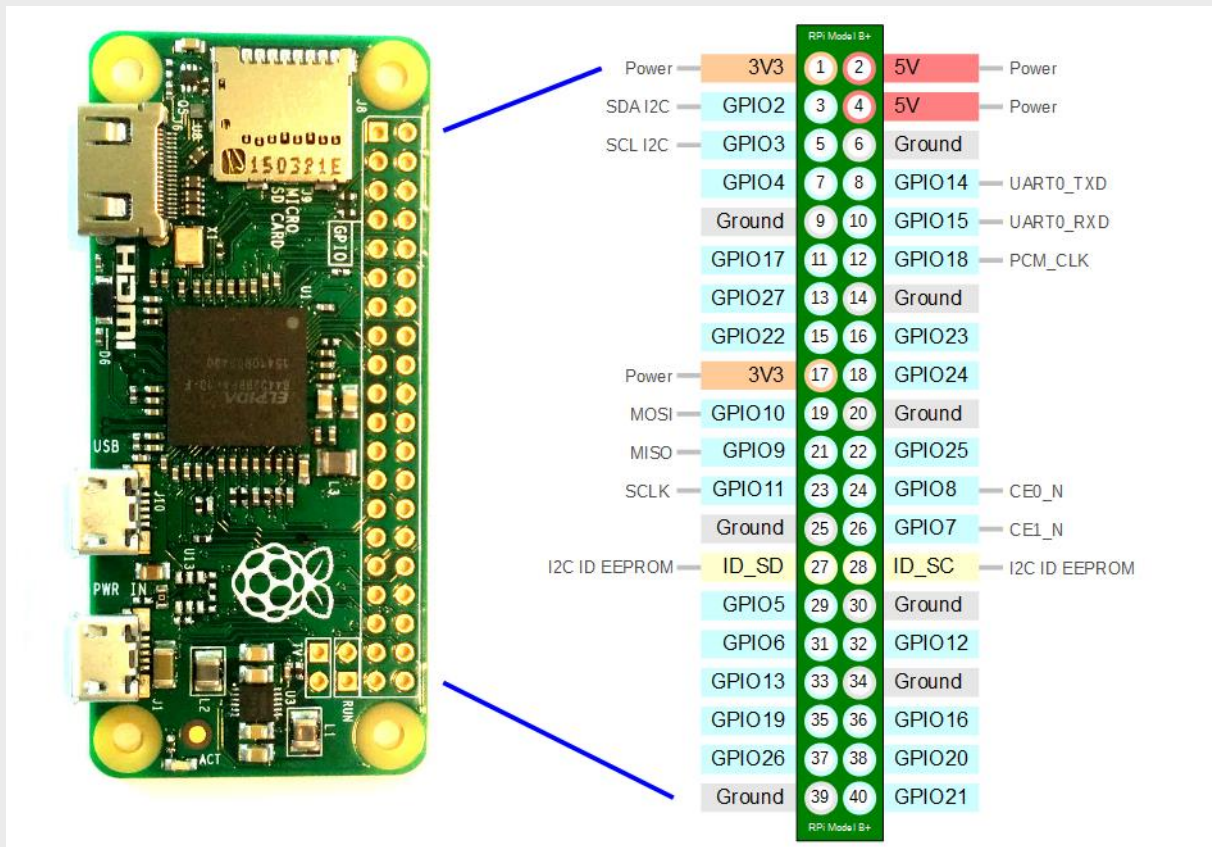
```
-----  
// LinuxCommand  
// Beispiel / Example  
// reboot  
// Löst den Raspberry Pi neu starten. Befehle werden mit sudo ausgeführt!  
  
// LinuxCommand_1  
reboot  
  
// LinuxCommand_2  
shutdown -r -t 0  
  
// LinuxCommand_3  
reboot  
  
// LinuxCommand_4  
reboot  
  
// LinuxCommand_5  
reboot  
  
// LinuxCommand_6  
reboot  
  
// LinuxCommand_7  
reboot  
  
// LinuxCommand_8  
reboot
```

Wenn man folgenden Befehl auslöst, wird der Befehl „LinuxCommand\_1“ ausgeführt:  
[http://Benutzername:Passwort@192.168.2.99:18181/pi/LinuxCommand\\_1](http://Benutzername:Passwort@192.168.2.99:18181/pi/LinuxCommand_1)

In diesem Beispiel wird der Raspberry Pi neu gestartet. Alle Konsolen-Befehle werden mit Adminrechten, also mit „sudo“ ausgeführt.

# GPIO-Darstellung

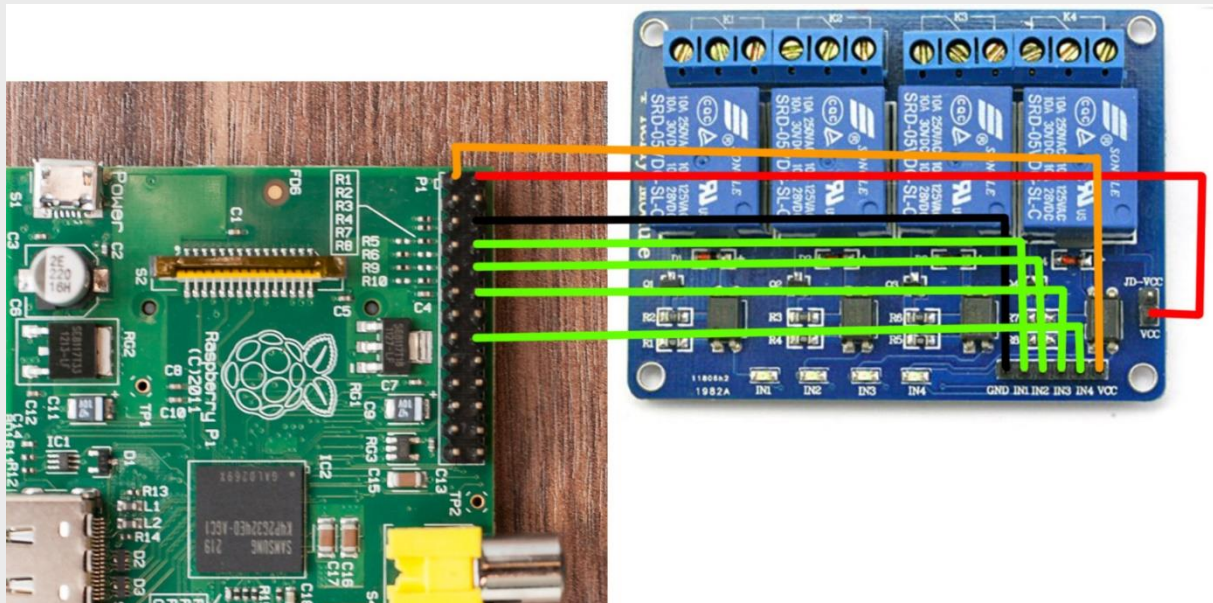
Im Internet bekommt ihr für jedes Raspberry Pi Modell eine GPIO-Darstellung für die Belegung der einzelnen Ports. Anbei ein Beispiel für den Raspberry Pi Zero W. Dieses Beispiel sollte für den Raspberry Pi 2, 3 Zero und Zero W gelten. Für den Raspberry 1 Revision A und B sind Abweichungen vorhanden.



(Quelle: <http://www.forum-raspberrypi.de/Thread-fehlkauf-display-trotzdem-benutzen>)

## Raspberry Pi Relays

Im Internet kann man je nach Anforderungen spezielle Relays für den Raspberry Pi erwerben.



(Quelle: <https://www.raspberrypi.org/forums/viewtopic.php?t=35155>)

Hier in diesem Beispiel wird VCC vom Relays an einen 5V-Pin des Raspberry Pi verbunden. Dasselbe geht für GND des Relays zu einem GND-Pin des Raspberry Pi.

In der vorigen GPIO-Darstellung könnt ihr die PIN-Nummern und dessen Verknüpfung ablesen.

Bei diesem Relays gibt es dann noch pro Relays einen IN-Anschluss. Hier kann man einfach einen beliebigen GPIO des Raspberry Pi auswählen.

Beispiel:

Relays VCC an PIN 2 vom Raspberry.  
 Relays GND an PIN 6 vom Raspberry.  
 Relays IN1 an PIN 16 vom Raspberry.  
 Relays IN2 an PIN 18 vom Raspberry.  
 Relays IN3 an PIN 22 vom Raspberry.  
 Relays IN4 an PIN 24 vom Raspberry.

Nun könnt ihr im Webserver bei den normalen GPIO-Befehlen folgendes hinterlegen:

```
// GPIO_23 - PIN_16
Toggle
```

```
// GPIO_24 - PIN_18
Toggle
```

```
// GPIO_25 - PIN_22
In;1000;Out
```

```
// GPIO_8 - PIN_24  
In;1000;Out
```

Danach speichert ihr die „\_config.txt“ ab.

Wenn man nun die Befehle für GPIO\_23 und GPIO\_24 aufruft, wird das Relays getoggelt. Daher, wenn es vorher auf HIGH war, wird es auf LOW gesetzt und umgekehrt.

GPIO\_25 und GPIO\_8 werden hingehen auf HIGH gestellt, dann wird 1 Sekunde gewartet und dann werden beide GPIOs auf LOW gestellt. Dieses Szenario kann man z.B. für einen Schlüsselschalter eines Garagentorantriebes nutzen. Ein anderer Anwendungsfall wäre hier das Schalten eines Türsummers. Da könnte man dann anstatt der 1 Sekunde auch 5 Sekunden zum Öffnen hinterlegen.

Je nach Relays kann man diese mit einem HIGH oder LOW schalten. Das könnt ihr in die entsprechenden Datenblätter der Relays nachlesen.

### **WICHTIG:**

**Wenn ihr Relays mit 230V nutzen möchtet, dann müsst ihr ganz genau wissen, was ihr da macht! 230V sind lebensgefährlich! Am besten fragt ihr einen fachkundigen Elektriker, wenn ihr Geräte mit 230V schalten möchtet.**

**Bei billigen China-Relays ist hier besondere Vorsicht geboten, da hier meistens die Richtlinien nicht eingehalten werden und es so zu Unfällen kommen kann!**

Selbstverständlich könnt ihr auch die benutzerdefinierten Befehle nutzen. Hinterlegt hierzu z.B. in der „\_config.txt“ folgendes:

```
// CUSTOM_1  
GPIO_23: In;2000;GPIO_24: In; 2000;GPIO_25: In; 2000;GPIO_8: In;GPIO_23:  
Out;2000;GPIO_24: Out; 2000;GPIO_25: Out; 2000;GPIO_8: Out
```

Wenn ihr nun den Befehl CUSTOM\_1 ([http://Benutzername:Passwort@192.168.2.99:18181/pi/CUSTOM\\_1](http://Benutzername:Passwort@192.168.2.99:18181/pi/CUSTOM_1)) ausführt, dann werden die 4 GPIOs nacheinander auf LOW und dann auf HIGH geschaltet. Dazwischen wird immer 2 Sekunden lang gewartet.



## WiringPi - GPIO

Seit 2019 wurde der Support der WiringPi-Bibliothek eingestellt. Es kann also sein, dass diese nicht mehr vorinstalliert ist und man kann diese auch nicht mehr über den Hersteller erhalten.

Es kann also sein, dass der Webserver läuft, aber die Befehle laufen dann ins Leere. In diesem Fall kann man über den Befehl „debug“ im „PCT-Pi-Remote – Webserver“ das Debugging aktivieren. Wenn man jetzt einen Befehl auslöst und es kommt der Fehler: „sudo: gpio: command not found“ oder ähnlich, dann fehlt WiringPi.

Der Quellcode ist derzeit noch über GitHub downloadbar (Stand 11.11.2023):  
<https://github.com/WiringPi/WiringPi>

Wir haben das Projekt aber auch auf unseren Webspace geladen, weil dieses OpenSource ist.

**Hinweis:** Weil WiringPi eingestellt wurde, gibt es kein Support für Raspberry Pi 5.

Über folgende Befehle lässt sich WiringPi selbst herunterladen und installieren:

Ins Home-Verzeichnis des Benutzers zu gelangen:

```
cd ~
```

Jetzt erstellt Ihr ein neues Verzeichnis mit dem Namen „webserver“:

```
mkdir webserver
```

Wechselt nun in das neue Verzeichnis „webserver“:

```
cd webserver
```

Falls noch nicht installiert, müsst ihr das Tool „wget“ installieren:

```
sudo apt-get install wget
```

Um nun den „PCT-Pi-Remote - Webserver“ in das aktuelle Verzeichnis herunterzuladen, benutzt ihr folgenden Befehl:

```
wget "http://pct.company/downloads/PCT-Pi-Remote/Webserver/WiringPi-master.zip"
```

Nun können wir das Projekt entpacken:

```
unzip WiringPi-master.zip
```

In den entpackten Ordner gelangen:

```
cd WiringPi-master
```

Das Projekt installieren:

```
./build
```

Damit wurde nun der Befehl „gpio“ über WiringPi installiert. Jetzt müsste der „PCT-Pi-Remote – Webserver“ laufen.





## Setup

Start your Raspberry Pi and log in with the user "root" or "pi". In our example, we always use the user "pi".

First of all we have to install the library "mono". To make sure your Raspberry Pi is up-to-date, type the following:

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

Then we install "mono":

```
sudo apt-get install mono-complete
```

After "mono" has been installed, enter the following in the console to get into the user's home directory:

```
cd ~
```

Now create your new directory named "webserver":

```
mkdir webserver
```

Now change to the new directory "webserver":

```
cd webserver
```

If you have not already installed, you must install the "wget" tool:

```
sudo apt-get install wget
```

To download the "PCT-Pi remote web server" to the current directory, use the following command:

```
wget "http://pct.company/downloads/PCT-Pi-Remote/Webserver/PCT-Pi-Remote - Webserver.exe"
```

In the following you have to download the corresponding "\_config.txt":

```
wget " http://pct.company/downloads/PCT-Pi-Remote/Webserver/_config.txt"
```

## Run the web server manually

After the installation you can run the web server manually:

```
sudo mono "PCT-Pi-Remote - Webserver.exe"
```

With CTRL + C you can leave the program. After restarting your Raspberry Pi, the web server would not start automatically.

## Always start the web server automatically at startup

In order for the web server to start automatically with the system every time the Raspberry Pi is restarted, we open the file "/etc/rc.local":

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

Before the "exit 0" you insert the line "su pi -c 'sudo mono' / home / pi / webserver / PCT-Pi-Remote - Webserver.exe ""'. The best way to do this is to copy it as it is, because you have to pay attention to the comma. There are other problems depending on the "automatic" formatting. It is also important that the directory is correct if you have previously selected another user or directory.

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

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
    printf "My IP address is %s\n" "$_IP"
fi

su pi -c 'sudo mono "/home/pi/webserver/PCT-Pi-Remote -
Webserver.exe"'

exit 0
```

## Access to web servers via the browser

Via the IP address of your Raspberry Pi, the port 18181 and the suffix / pi, you can access the web server with any browser.

Example: <http://192.168.2.99:18181/pi>

If you are registered with your Raspberry Pi, you get the IP address via the command "ifconfig".

The output should look something like this:

```
pi@PCT-Pi-Garage:~ $ ifconfig
eth0      Link encap:Ethernet  Hardware Adresse b8:27:eb:30:d2:f4
          inet6-Adresse: fe80::50ca:dc5e:a06a:3edf/64  Gültigkeitsbereich:Verbindung
          UP BROADCAST MULTICAST  MTU:1500  Metrik:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo        Link encap:Lokale Schleife
          inet Adresse:127.0.0.1  Maske:255.0.0.0
          inet6-Adresse: ::1/128  Gültigkeitsbereich:Maschine
          UP LOOPBACK RUNNING  MTU:65536  Metrik:1
          RX packets:137 errors:0 dropped:0 overruns:0 frame:0
          TX packets:137 errors:0 dropped:0 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1
          RX bytes:11551 (11.2 KiB)  TX bytes:11551 (11.2 KiB)

wlan0     Link encap:Ethernet  Hardware Adresse 74:da:38:00:5f:0e
          inet Adresse:192.168.2.99  Bcast:192.168.2.255  Maske:255.255.255.0
          inet6-Adresse: fe80::76da:38ff:fe00:5f0e/64  Gültigkeitsbereich:Verbindung
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metrik:1
          RX packets:4264 errors:0 dropped:152 overruns:0 frame:0
          TX packets:985 errors:0 dropped:2 overruns:0 carrier:0
          Kollisionen:0 Sendewarteschlangenlänge:1000
          RX bytes:517907 (505.7 KiB)  TX bytes:288917 (282.1 KiB)
```

eth0 stands for the network port and wlan0 for the wireless LAN.

How to connect the Raspberry Pi to the network can be found on the Internet.

The web server will ask for authentication. Default is here as username: admin and as password: admin. You can add as many users as you want in the "\_config.txt".

**Bitte melden Sie sich an**

Für <http://192.168.2.99:18181> sind ein Nutzernamen und ein Passwort erforderlich.

Die Verbindung zu dieser Website ist nicht sicher

Benutzername:

Passwort:

If you are successfully connected to the web server, the display looks like this:



## PCT-Pi-Remote - Webserver



**GPIO\_4**

Touch

**CUSTOM\_1**

Touch

**LinuxCommand\_1**

Touch

**GPIO\_7**

Touch

**CUSTOM\_2**

Touch

**LinuxCommand\_2**

Touch

**GPIO\_8**

Touch

**CUSTOM\_3**

Touch

**LinuxCommand\_3**

Touch

These buttons can then be activated from any browser. Each button is linked to a command line in the "\_config.txt". Therefore, everyone can decide for themselves what to do with an actuation.

## What is triggered when you press a button?

When a button is pressed, e.g. following command:

[http://192.168.2.99:18181/pi/GPIO\\_4](http://192.168.2.99:18181/pi/GPIO_4)

You can also use these commands as a shortcut and start it. Third-party apps also require these commands. What is ultimately done when you press the button, you configure it in the "\_config.txt".

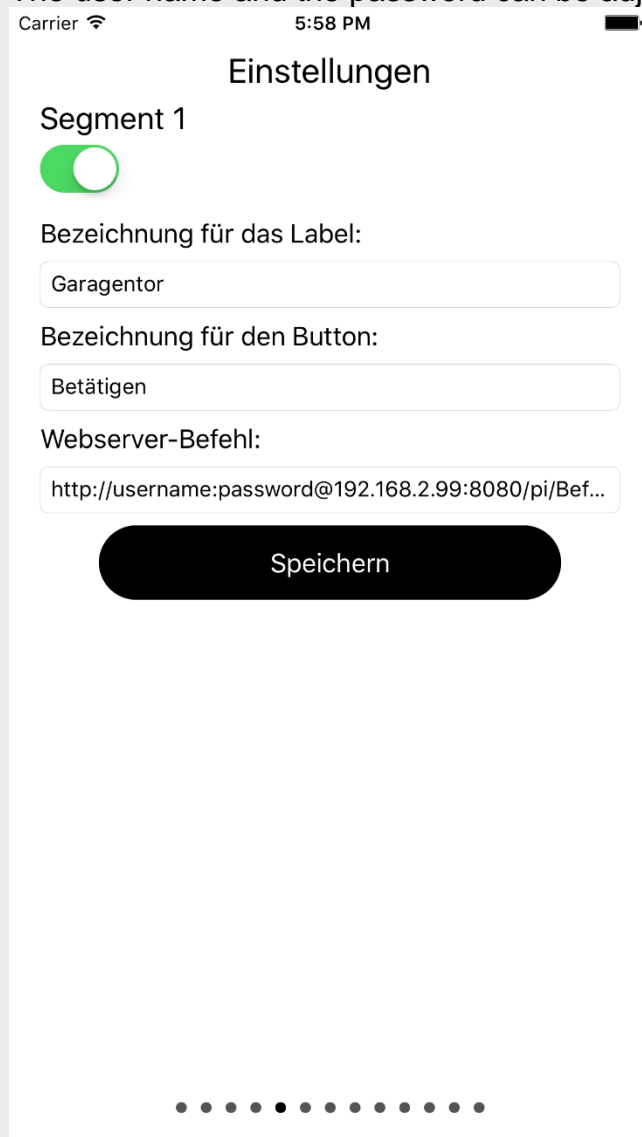
## Access via the iOS or Android app

In the "PCT-Pi-Remote" apps, there is a configuration page for each segment.

On this page, enter "Web-Server command" for example. the following:

[http://Benutzername:Passwort@192.168.2.99:18181/pi/GPIO\\_4](http://Benutzername:Passwort@192.168.2.99:18181/pi/GPIO_4)

The user name and the password can be adjusted in the "\_config.txt".



The screenshot shows the configuration page for Segment 1 in the PCT-Pi-Remote app. The page is titled "Einstellungen" (Settings) and has a status bar at the top showing "Carrier", signal strength, "5:58 PM", and battery level. The segment is labeled "Segment 1" and is currently turned on, indicated by a green toggle switch. Below the toggle, there are three input fields: "Bezeichnung für das Label:" with the value "Garagentor", "Bezeichnung für den Button:" with the value "Betätigen", and "Webserver-Befehl:" with the value "http://username:password@192.168.2.99:8080/pi/Bef...". A large black button labeled "Speichern" (Save) is positioned below the input fields. At the bottom of the screen, there is a row of ten small grey dots, with the fifth dot from the left being filled, indicating the current segment.

## Web server configuration via "\_config.txt"

In this file you can adapt the web server to its projects. Thus, for example, manage users, and decide what happens when you trigger specific commands through the web server in the browser or through the apps.

When we are connected to the Raspberry Pi, we go to the home directory:

```
cd ~
```

Then navigate to the previously created directory "webserver":

```
cd webserver
```

Then we open the "\_config.txt" with nano:

```
sudo nano _config.txt
```

In general, this file can also be edited when the web server is started. This does not have to be finished.

The user names and passwords are stored in this area. It is important that the users are separated by a semicolon, and the number of user names must match the number of passwords.

```
-----  
// Benutzernamen / Usernames  
// Beispiel / Example  
// admin;admin2;admin3  
admin;PCT  
  
// Passwörter / Passwords  
// Beispiel / Example  
// password;password2;password3  
admin;PCT  
-----
```

Below you can define for each of the 13 GPIO commands what should happen when these commands are triggered.

```
// GPIO_4 - PIN_7  
Out;1000;In
```

The commands can generally be of any length. Out means that a port is set to HIGH. In means that a port is set to LOW. Toggle inverts the port position. If you enter an integer, this is the wait time in milliseconds.

```
-----  
// GPIO-Befehle / GPIO-Commands  
// Beispiel / Example  
// Out;1000;In  
// Der GPIO wird auf HIGH gestellt, dann wird 1000ms gewartet und dann wird d  
// Alle Befehle: "Out": Schaltet einen Port auf HIGH; "Zahlenwert ohne Kommat  
  
// GPIO_4 - PIN_7  
Out;1000;In  
  
// GPIO_7 - PIN_26  
Out;1000;In  
  
// GPIO_8 - PIN_24  
Out;1000;In  
  
// GPIO_9 - PIN_21  
Out;1000;In  
  
// GPIO_10 - PIN_12  
Out;1000;In  
  
// GPIO_11 - PIN_23  
Out;1000;In  
  
// GPIO_14 - PIN_8  
Out;1000;In  
  
// GPIO_15 - PIN_10  
Out;1000;In  
  
// GPIO_18 - PIN_12  
Out;1000;In  
  
// GPIO_22 - PIN_15  
Out;1000;In  
  
// GPIO_23 - PIN_16  
Out;1000;In  
  
// GPIO_24 - PIN_18  
Toggle  
  
// GPIO_25 - PIN_22  
Out;1000;In  
-----
```

So that you can implement even more complex things in your project, there are also the user-defined commands.

```
-----  
// Benutzerdefinierte-Befehle / Custom-Commands  
// Beispiel / Example  
// GPIO_23: Out;1000;GPIO_24_Out;1000;GPIO_23: I  
// Erst wird der GPIO_23 auf HIGH gestellt, dann  
// Alle Befehle: "GPIO_XY: Out": Schaltet den Po  
  
// CUSTOM_1  
GPIO_24: Toggle;2000;GPIO_24: Toggle  
  
// CUSTOM_2  
GPIO_24: Toggle;2000;GPIO_24: Toggle  
  
// CUSTOM_3  
GPIO_24: Toggle;2000;GPIO_24: Toggle  
  
// CUSTOM_4  
GPIO_24: Toggle;2000;GPIO_24: Toggle
```

If you run the following command, the command "CUSTOM\_1" is executed:  
[http://Benutzername:Passwort@192.168.2.99:18181/pi/CUSTOM\\_1](http://Benutzername:Passwort@192.168.2.99:18181/pi/CUSTOM_1)

You can only use the ports here GPIO\_4, GPIO\_7, GPIO\_8, GPIO\_9, GPIO\_10, GPIO\_11, GPIO\_14, GPIO\_15, GPIO\_18, GPIO\_22, GPIO\_23, GPIO\_24 and GPIO\_25 use. So exactly the ports, which can also be addressed as normal commands directly.

Commands are also separated by a semicolon. Integers are waiting times. Otherwise you can use the port: command, such as:

```
GPIO_24: Out  
GPIO_24: In  
GPIO_24: Toggle
```

Example that sets port GPIO\_23 and GPIO\_24 to HIGH, waits 10 seconds and returns both ports to LOW:

```
GPIO_23: Out;GPIO_24: Out;10000;GPIO_23: In;GPIO_24: In
```



Since version 2.0.0 you can now trigger commands in the Linux console. This is e.g. necessary, if you want to control sockets over 433MHz and this must use a third-party software on the Raspberry Pi.

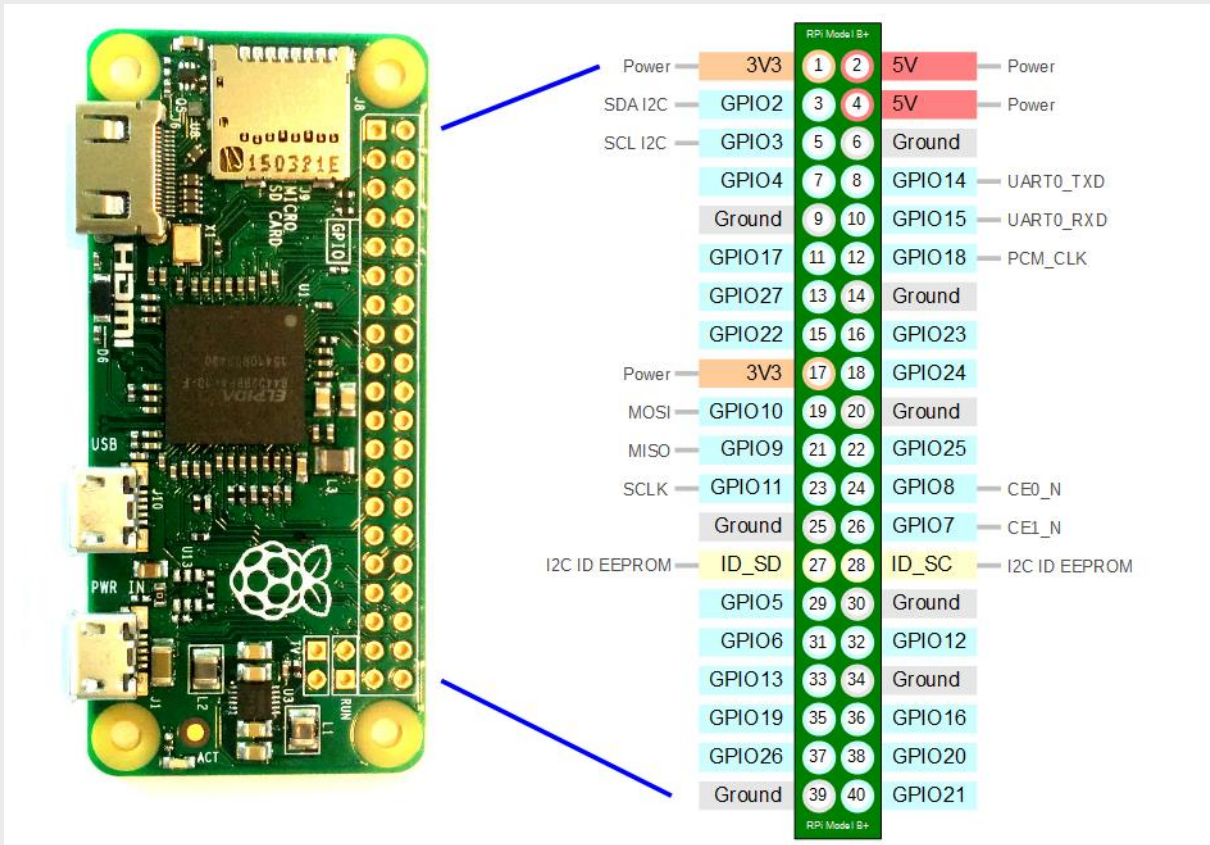
```
-----  
// LinuxCommand  
// Beispiel / Example  
// reboot  
// L sst den Raspberry Pi neu starten. Befehle werden mit sudo ausgef hrt!  
  
// LinuxCommand_1  
reboot  
  
// LinuxCommand_2  
shutdown -r -t 0  
  
// LinuxCommand_3  
reboot  
  
// LinuxCommand_4  
reboot  
  
// LinuxCommand_5  
reboot  
  
// LinuxCommand_6  
reboot  
  
// LinuxCommand_7  
reboot  
  
// LinuxCommand_8  
reboot
```

If you run the following command, the command "LinuxCommand\_1" is executed:  
[http://Benutzername:Passwort@192.168.2.99:18181/pi/LinuxCommand\\_1](http://Benutzername:Passwort@192.168.2.99:18181/pi/LinuxCommand_1)

In this example, the Raspberry Pi restarts. All console commands are executed with admin rights, also with "sudo".

## GPIO illustration

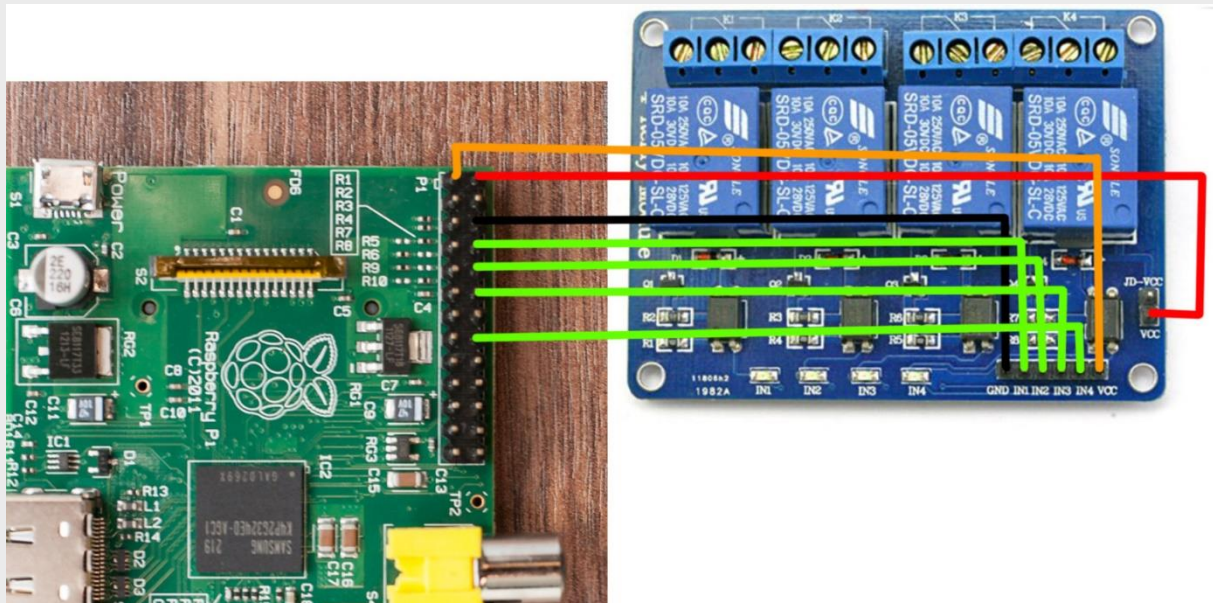
In the Internet, you will get a GPIO representation for each Raspberry Pi model for the assignment of the individual ports. Here is an example of the Raspberry Pi Zero W. This example should apply to the Raspberry Pi 2, 3 Zero and Zero W. There are deviations for Raspberry 1 Revision A and B.



(Source: <http://www.forum-raspberrypi.de/Thread-fehlkauf-display-trotzdem-benutzen>)

## Raspberry Pi Relays

On the Internet you can purchase special relays for the Raspberry Pi, depending on your requirements.



(Source: <https://www.raspberrypi.org/forums/viewtopic.php?t=35155>)

Here, in this example, VCC is connected from the relay to a 5V pin of the Raspberry Pi. The same goes for GND of the relay to a GND pin of the Raspberry Pi.

In the previous GPIO representation you can read the PIN numbers and their linkage.

In this relays, there is then still one relay port per relay. Here you can simply select any GPIO of the Raspberry Pi.

Example:

Relays VCC to PIN 2 from Raspberry.  
 Relays GND to PIN 6 from Raspberry.  
 Relays IN1 to PIN 16 from Raspberry.  
 Relays IN2 to PIN 18 from Raspberry.  
 Relays IN3 to PIN 22 from Raspberry.  
 Relays IN4 to PIN 24 from Raspberry.

Now you can put the following in the web server with the normal GPIO-Commands:

```
// GPIO_23 - PIN_16
Toggle
```

```
// GPIO_24 - PIN_18
Toggle
```

```
// GPIO_25 - PIN_22
In;1000;Out
```

```
// GPIO_8 - PIN_24  
In;1000;Out
```

Then you save the "\_config.txt".

If you now call the commands for GPIO\_23 and GPIO\_24, the relays are toggled. Therefore, if it was previously HIGH, it is set to LOW and vice versa.

GPIO\_25 and GPIO\_8 are set to HIGH, then 1 second is waited and then both GPIOs are set to LOW. This scenario can be e.g. for a key switch of a garage door drive. Another application would be the switching of a door buzzer. Then you could then instead of the 1 second also 5 seconds to open the deposit.

Depending on the relays you can switch them with a HIGH or LOW. This can be read in the corresponding data sheets of the Relays.

### **IMPORTANT:**

**If you want to use relays with 230V, you have to know exactly what you are doing! 230V are life-threatening! The best thing is to ask an expert electrician if you want to switch units with 230V.**

**In the case of cheap China Relays, special caution is advised, as the guidelines are usually not observed here and accidents can occur!**

Of course, you can also use the custom commands. For example, in the "\_config.txt":

```
// CUSTOM_1  
GPIO_23: In;2000;GPIO_24: In; 2000;GPIO_25: In; 2000;GPIO_8: In;GPIO_23:  
Out;2000;GPIO_24: Out; 2000;GPIO_25: Out; 2000;GPIO_8: Out
```

If you have the command CUSTOM\_1 ([http://Username:Password@192.168.2.99:18181/pi/CUSTOM\\_1](http://Username:Password@192.168.2.99:18181/pi/CUSTOM_1)) then the 4 GPIOs are switched to LOW and then to HIGH. In between, it is always waited for 2 seconds.

## WiringPi - GPIO

Since 2019, support for the WiringPi library has been discontinued. It may be that this is no longer pre-installed and you can no longer obtain it from the manufacturer.

So it may be that the web server is running, but the commands then come to nothing. In this case, debugging can be activated using the “debug” command in the “PCT-Pi Remote – Web Server”. If you now trigger a command and you get the error: “sudo: gpio: command not found” or something similar, then WiringPi is missing.

The source code can currently still be downloaded via GitHub (as of November 11, 2023):

<https://github.com/WiringPi/WiringPi>

We also uploaded the project to our web space because it is open source.

**Note:** Because WiringPi has been discontinued, there is no support for Raspberry Pi 5.

You can download and install WiringPi yourself using the following commands:

To get to the user's home directory:

```
cd ~
```

Now create a new directory called “webserver”:

```
mkdir webserver
```

Now switch to the new “webserver” directory:

```
cd webserver
```

If not installed yet, you need to install the “wget” tool:

```
sudo apt-get install wget
```

To download the “PCT-Pi-Remote - Web server” into the current directory, use the following command:

```
wget "http://pct.company/downloads/PCT-Pi-Remote/Webserver/WiringPi-master.zip"
```

Now we can unpack the project:

```
unzip WiringPi-master.zip
```

Get into the unzipped folder:

```
cd WiringPi-master
```

Install the project:

```
./build
```

The “gpio” command has now been installed via WiringPi. The “PCT-Pi-Remote – web server” should now be running.