

[Text eingeben]

# Projekt Dokumentation

## RaspBox

## Inhalt

**Contents**

1	Allgemein.....	4
2	Systemausstattung.....	4
3	Pinbelegung.....	5
4	Verteilboard.....	6
5	Grundkonfiguration.....	7
5.1	Betriebssystem.....	7
5.2	Apache Webserver.....	7
5.3	PHP installieren.....	7
5.4	PHP sudo rechte vergeben.....	8
5.5	FTP Server installieren und einrichten.....	9
5.5.1	Konfiguration.....	10
5.5.2	Einen User anlagen, mit welchem man über FTP zugang zu /var/www hat:.....	11
5.6	Python installieren.....	12
5.7	Wiring Pi installieren.....	12
5.7.1	Benutzung.....	12
5.7.2	Module Load Commands.....	13
5.7.3	/sys/class/gpio mode commands.....	13
6	Systemkonfiguration.....	15
6.1	Scripte bei Systemstart starten.....	15
6.2	Scripte als Systemdienste Starten.....	15
6.3	Python Scripte Startbar machen.....	16
7	Analogeingänge.....	17
7.1	Anschluss MCP3008.....	17
7.2	Vorbereitungen.....	18
7.3	Python Script.....	19
8	Serielle Schnittstelle.....	20
8.1	Python Bibliothek für RS232/UART installieren.....	20
8.2	Terminalprograminstallieren.....	20
8.3	Python Scripte.....	20
9	GPIO Ausgänge.....	21
10	CAN BUS Interface.....	21



## 1 Allgemein

Dieses Dokument beschreibt den Aufbau und die Konfiguration der PiBox.

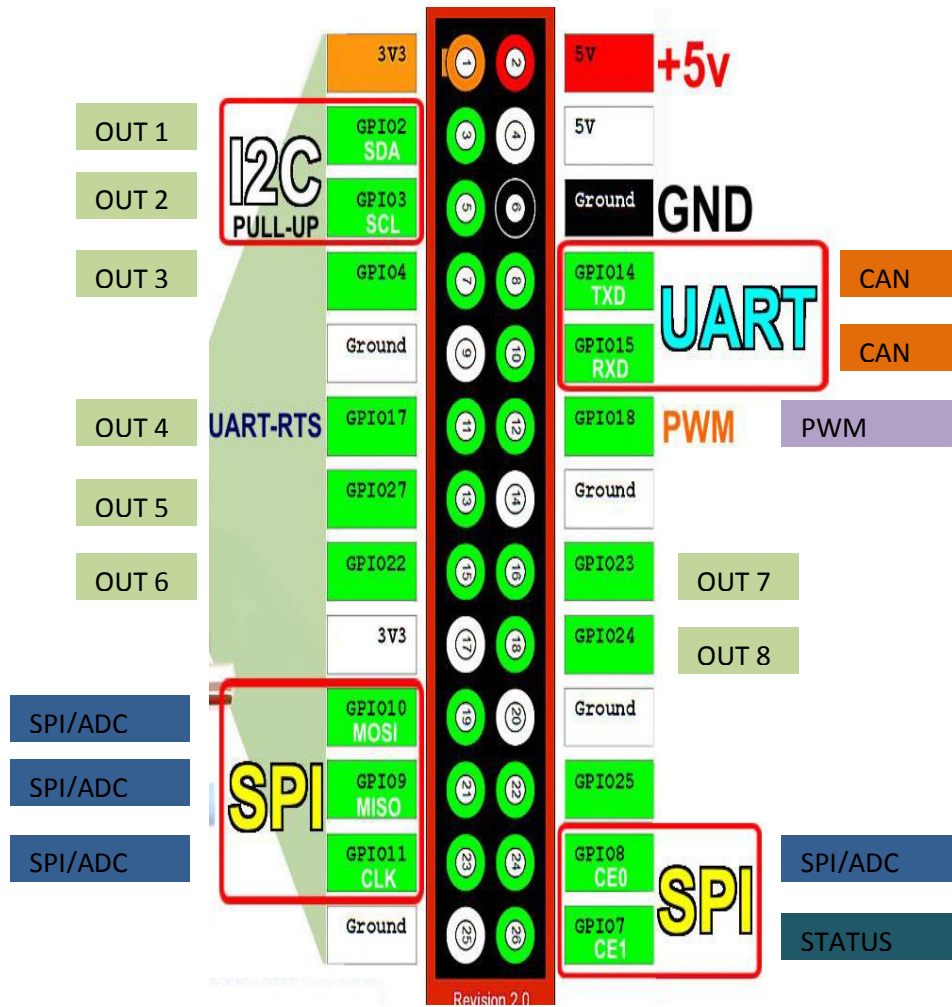
Auf Grund einer einfachen Konfiguration, und einfaches und ziehorientiertes Arbeiten zu ermöglichen, wurden manche Linux-Schutzmechanismen, bewusst, außer Kraft gesetzt.

Sollte der Raspberry von fremden Rechnern zu erreichen Sein, sollten mnche hier gezeigten Konfigurationen überdacht und den gegebenen Bedingungen angepasst werden.

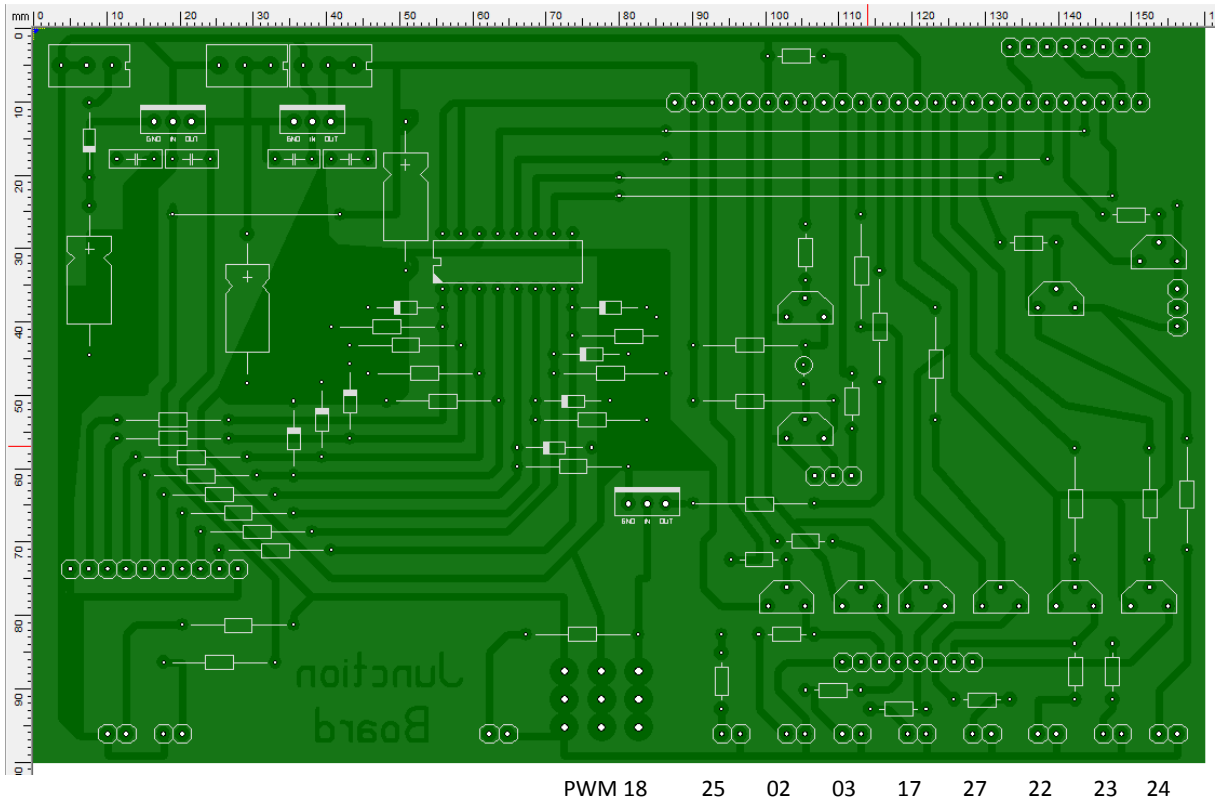
## 2 Systemausstattung

<b>Funktion</b>	<b>Beschreibung</b>	<b>Status</b>
8 potentialfreie Ausgänge	8 potentialfreie Schließer-Kontakte	Im Test
8 analoge Eingänge	8 Messebezogene Spannungsmesseingänge 0-20 V mit Überspannungsschutz	Im Test
CAN Bus adaption	Anschlussmöglichkeit für CAB-Bus	In Planung
Web Interface	Alle Schalt und Konfigurationen per http-Zugang möglich	Im Test

### 3 Pinbelegung



## 4 Verteilboard



## 5 Grundkonfiguration

### 5.1 Betriebssystem

Betriebssystem	<a href="http://2014-01-07-wheezy-raspbian.zip">2014-01-07-wheezy-raspbian.zip</a>
Download Link	<a href="http://downloads.raspberrypi.org/raspbian_latest">http://downloads.raspberrypi.org/raspbian_latest</a>
Install Tool	Win32DiskImager

Sollte etwas bei der Installation schief gehen und die SD-Karte nicht mehr in voller Größe zur Verfügung stehen, hilft folgendes Windows Tool:

- Windows Command Line öffnen
- Diskpart (mit ENTER bestätigen)
- list disk (mit ENTER bestätigen) -> nun werden dir die vorhandenen disk aufgelistet
- select disk <USB-Stick> (mit ENTER bestätigen) -> gib die nummer ein für die sd... Scau auf die gröesse!
- clean (mit ENTER bestätigen)
- create partition primary (mit ENTER bestätigen)
- select partition=1 (mit ENTER bestätigen)
- active (mit ENTER bestätigen)
- format fs=fat32 QUICK (mit ENTER bestätigen)
- assign (mit ENTER bestätigen)
- Console Schliessen und Fertig

### 5.2 Apache Webserver

Installation	<code>sudo apt-get update</code>
	<code>sudo apt-get install apache2</code>

Bei eventuell auftretenden Problemen mit `sudo apt-get update` bei Schaft folgendes Abhilfe:  
[Sudo diskpart](#) ausführen

### 5.3 PHP installieren

Installation	<code>sudo apt-get install php5</code>
--------------	--

## 5.4 PHP sudo rechte vergeben

Das über PHP Skripte ALLE SUDO Befehle ausgeführt werden können muss folgende getan werden

Mit `sudo visudo` das Editieren starten

Wie unten abgebildet, die Zeile `www-data ALL=(ALL) NOPASSWD: ALL` hinzufügen

```
GNU nano 2.2.6      File: /etc/sudoers.tmp

# Allow members of group sudo to execute any command
%sudo  ALL=(ALL:ALL) ALL

# See sudoers(5) for more information on "#include" directives:

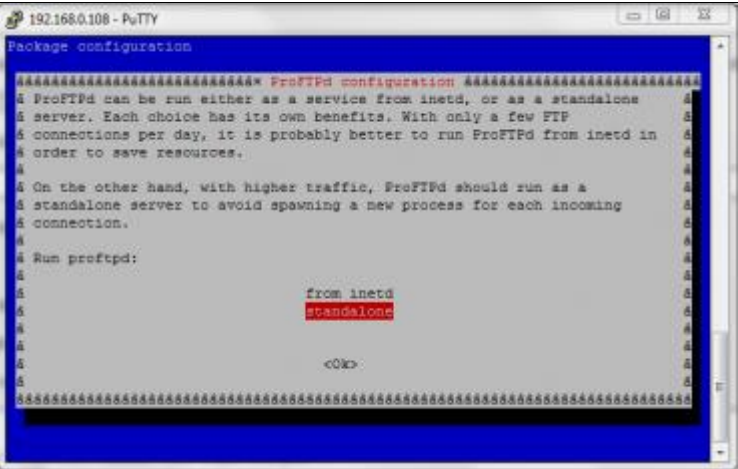
#include_dir /etc/sudoers.d
pi ALL=(ALL) NOPASSWD: ALL
www-data ALL=(ALL) NOPASSWD: ALL
#www-data ALL=(root) NOPASSWD: /usr/bin/python

^G Get Help  ^O WriteOut  ^R Read File ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^J Justify   ^W Where Is  ^V Next Page ^U UnCut Text ^T To Spell
```

Somit hat der User `www-data` (unter dem PHP läuft) die gleichen `sudo`-Rechte wie der Standarduser `pi`



## 5.5 FTP Server installieren und einrichten

Installation	<code>sudo apt-get install proftpd</code>
	 <p>The screenshot shows a terminal window titled "192.168.0.108 - PuTTY" with a blue background. The text displayed is the ProFTPD configuration dialog. It starts with "Package configuration" and "ProFTPD configuration". It explains that ProFTPD can run as a service from inetd or as a standalone server. It asks the user to choose between "from inetd" and "standalone". The "standalone" option is highlighted in red. At the bottom, there is a prompt "&lt;Ok&gt;".</p>

## 5.5.1 Konfiguration

Die Datei `/etc/proftpd/proftpd.conf` entsprechend konfigurieren

```
#
# /etc/proftpd/proftpd.conf -- This is a basic ProFTPD configuration file.
# To really apply changes, reload proftpd after modifications, if
# it runs in daemon mode. It is not required in inetd/xinetd mode.
#

# Includes DSO modules
Include /etc/proftpd/modules.conf

IdentLookups                                off

ServerName                                  "raspi"
ServerType                                 standalone
DeferWelcome                                off

MultilineRFC2228                            on
DefaultServer                                on
ShowSymlinks                                on

TimeoutNoTransfer                            600
TimeoutStalled                               600
TimeoutIdle                                  1200

DisplayLogin                                welcome.msg
DisplayChdir                                .message true
ListOptions                                  "-l"

DenyFilter                                   \*.*

RootLogin                                    on
# Use this to jail all users in their homes
DefaultRoot                                  ~

Port                                       21

<IfModule mod_dynmasq.c>
# DynMasqRefresh 28800
</IfModule>

MaxInstances                                 30

# Set the user and group that the server normally runs at.
User                                         proftpd
Group                                        nogroup

# Umask 022 is a good standard umask to prevent new files and dirs
# (second parm) from being group and world writable.
Umask                                        022 022
# Normally, we want files to be overwriteable.
AllowOverwrite                               on

TransferLog /var/log/proftpd/xferlog
SystemLog /var/log/proftpd/proftpd.log

<IfModule mod_quotatab.c>
QuotaEngine off
</IfModule>

<IfModule mod_ratio.c>
Ratios off
```

```
</IfModule>

# Delay engine reduces impact of the so-called Timing Attack described in
# http://www.securityfocus.com/bid/11430/discuss
# It is on by default.
<IfModule mod_delay.c>
DelayEngine on
</IfModule>

<IfModule mod_ctrls.c>
ControlsEngine    off
ControlsMaxClients 2
ControlsLog       /var/log/proftpd/controls.log
ControlsInterval  5
ControlsSocket    /var/run/proftpd/proftpd.sock
</IfModule>

<IfModule mod_ctrls_admin.c>
AdminControlsEngine off
</IfModule>

Include /etc/proftpd/conf.d/
```

### 5.5.2 Einen User anlagen, mit welchem man über FTP Zugang zu /var/www hat:

#### User mit dem namen **web** anlagen:

```
sudo useradd -m web -g 1000 -G
adm,dialout,cdrom,sudo,audio,video,plugdev,games,users,netdev,inp
ut
```

#### Den eben angelegten User das Passwort **web** zuordnen

```
sudo passwd web
```

#### Dem user **web** das Verzeichnis /var/www/ als home Verzeichnis zuordnen

```
usermod -d /var/www/ web
```

## 5.6 Python installieren

## 5.7 Wiring Pi installieren

Um die GPIO Ausgänge direct von der Shell oder mittels PHP Skripte zu starten wird WiringPi benötigt.

Installation	<code>sudo apt-get install git-core</code>
Testen der Installation	<pre> gpio readall pi@raspberrypi ~ \$ ls /etc/rc.local /etc/rc.local pi@raspberrypi ~ \$ gpio readall -----+-----+-----+-----+-----+-----+   wiringPi   GPIO   Phys   Name    Mode   Value    -----+-----+-----+-----+-----+-----+   0   17   11   GPIO 0   IN   Low     1   18   12   GPIO 1   IN   Low     2   27   13   GPIO 2   IN   Low     3   22   15   GPIO 3   IN   Low     4   23   16   GPIO 4   IN   Low     5   24   18   GPIO 5   IN   Low     6   25   22   GPIO 6   IN   Low     7   4   7   GPIO 7   IN   Low     8   2   3   SDA      OUT   High     9   3   5   SCL      OUT   Low      10   8   24   CEO      ALT0   High     11   7   26   CE1      ALT0   High     12   10   19   MOSI     ALT0   Low      13   9   21   MISO     ALT0   High     14   11   23   SCLK     ALT0   Low      15   14   8   TxD      ALT0   High     16   15   10   RxD      ALT0   High     17   28   3   GPIO 8   ALT2   Low      18   29   4   GPIO 9   ALT2   Low      19   30   5   GPIO10   ALT2   Low      20   31   6   GPIO11   ALT2   Low     -----+-----+-----+-----+-----+-----+ pi@raspberrypi ~ \$ █ </pre>

### 5.7.1 Benutzung

- `gpio [-g] mode <pin> in/out/pwm/up/down/tri`

This sets the mode of a pin to be input, output or pwm and additionally can set the internal pull-up/down resistors to pull-up, pull-down or none.

- `gpio [-g] write <pin> 0/1`

This sets an output pin to high (1) or low (0)

- `gpio [-g] pwm <pin> <value>`

Set the pin to a PWM value (0-1023 is supported)

- **gpio [-g] read <pin>**

Reads and prints the logic value of the given pin. It will print 0 (low) or 1 (high).

- **gpio readall**

This reads all the normally accessible pins and prints a table of their numbers (both wiringPi and BCM\_GPIO, so makes for a handy cross-reference chart), along with their modes and current values.

### 5.7.2 Module Load Commands

- **gpio load spi [buffer size in KB]**

This loads the SPI kernel modules and optionally sets the internal buffer to the given size in KB (multiples of 1024). The default is 4KB and is usually more than enough for most application which only exchange a byte or 2 at a time over the SPI bus.

The /dev/spi\* entries are set to be owned by the person using the **gpio** program, so there is no need to run subsequent programs as root (unless they use other wiringPi functions)

- **gpio load i2c [baud rate in Kb/sec]**

This loads the I2C kernel modules and optionally sets the baud rate to the given speed in Kb/sec (multiples of 1000). The default is 100Kb/sec.

The /dev/I2c\* entries are set to be owned by the person using the **gpio** program, so there is no need to run subsequent programs as root (unless they use other *wiringPi* functions)

### 5.7.3 /sys/class/gpio mode commands

- **gpio export <pin> in/out**

This exports the given pin (BCM-GPIO pin number) as an input or output and makes it available for a user program running as the same user to use.

- **gpio unexport <pin>**

Removes the export of the given pin.

- **gpio unexportall**

Removes all /sys/class/gpio exports.

- **gpio exports**

This prints a list of all gpio pins which have been exported via the /sys/class/gpio interface and their modes.

- **gpio edge <pin> rising/falling/both/none**

This enables the given pin for edge interrupt triggering on the rising, falling or both edges.  
(Or none which disables it)

**Note:** The pin numbers in the sys mode are *always* BCM-GPIO pin numbers.

## 6 Systemkonfiguration

### 6.1 Scripte bei Systemstart starten

Das bei Systemstart zu startende Script muss in die datei `/etc/rc.local`, ans Ende eingetragen werden

```
#!/bin/sh -e
#
# rc.local
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi
#Testscript g starten
/home/pi/g &
#Script zum Auslesen der ADCs starten
/home/pi/radc &
exit 0
```

### 6.2 Scripte als Systemdienste Starten

Um ein Script als Systemdiens zu starten muss im Ordner `/etc/init.d/` die entsprechende datei angelegt werden.

Zum Starten des Auslesens der Analog Digital Converter zum Beispiel die Datei: `/etc/init.d/readadc` mit dem Inhalt:

```
#!/bin/bash
# Description:      Start read ADC
### END INIT INFO

case "$1" in
  start)
    echo "Starting Reading ADC"
    /home/pi/radc &
    ;;
  stop)
    echo "Stopping Reading ADC"
    pkill -f /home/pi/radc
    ;;
  restart)
    pkill -f /home/pi/radc
    echo "STOPPED....."
    /home/pi/radc &
    echo "Restarted Read ADCs!!!!!!!!!"
    ;;
  *)
    Echo "Fehlerhafter Parameter!"
    Exit 1
    ;;
esac
exit 0
```

## 6.3 Python Scripte Startbar machen

Im Kopf des Scriptes muss folgendes stehen:

```
#!/usr/bin/python
# Description:      Bemerkungen zum Script
```

Danach das Script mit folgendem Befehl ausführbar machen:

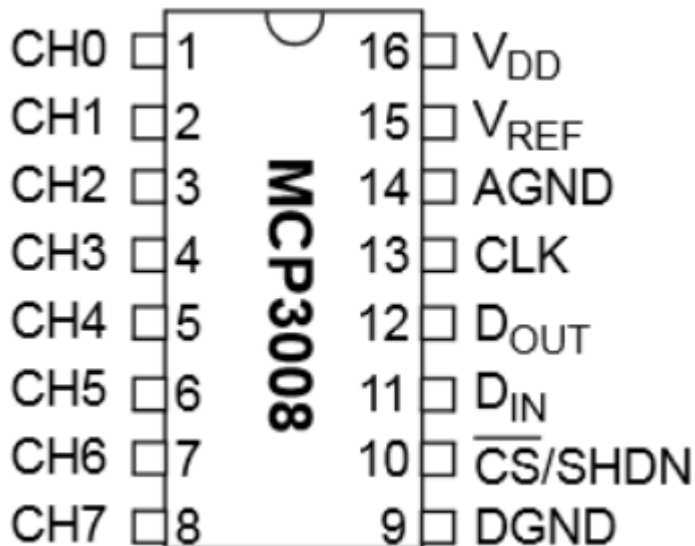
```
Sudo chmod +x SCRIPTNAME
```



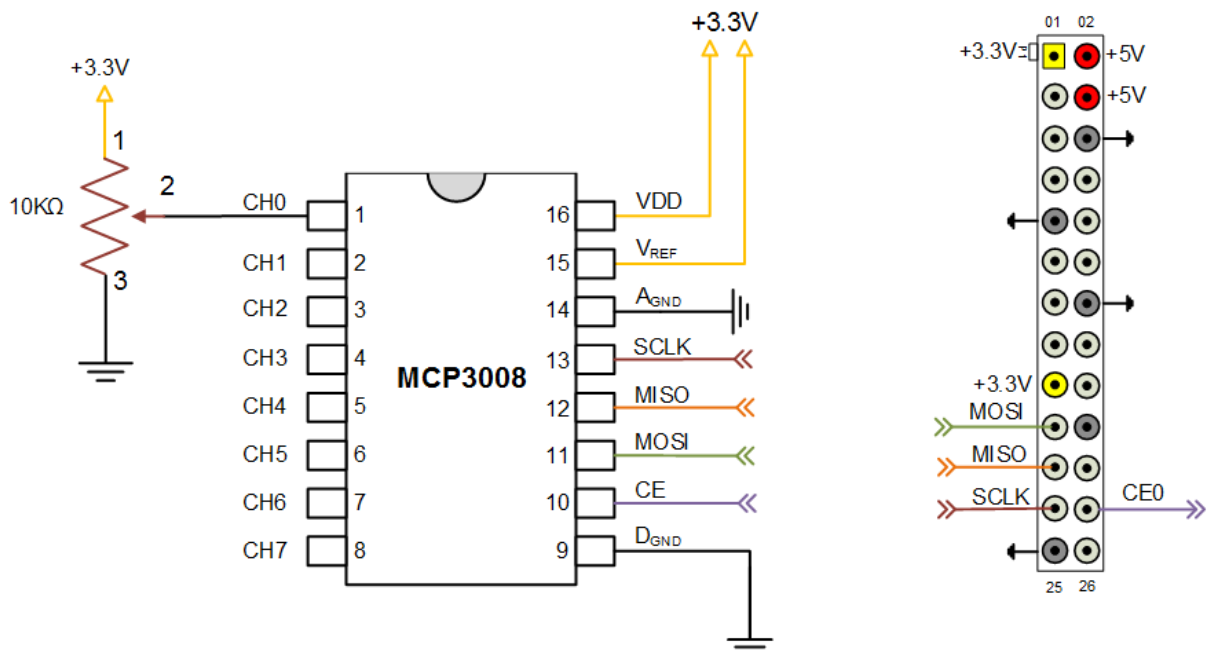
## 7 Analogeingänge

Die 8 analogen Eingänge wurden per MCP3008 an SPI realisiert, wobei die Hardware SPI zum Einsatz kam.

### 7.1 Anschluss MCP3008



VDD	3.3V
VREF	3.3V
AGND	GROUND
CLK	GPIO11 (P1-23)
DOUT	GPIO9 (P1-21)
DIN	GPIO10 (P1-19)
CS	GPIO8 (P1-24)
DGND	GROUND



## 7.2 Vorbereitungen

Als erstes muss die Hardware SPI des PI's eingeschaltet werden, dazu muss folgende Datei editiert werden:

```
sudo nano /etc/modprobe.d/raspi-blacklist.conf
```

In dieser Datei folgende Zeile einfügen:

```
Blacklist spi-bcm2708
```

Danach das System neustarten :

```
Sudo reboot
```

Danach mit `lsmod` überprüfen ob SPI eingeschaltet ist:

```

leds_gpio          2059  0
led_class          3688  1 leds_gpio
snd                61291  7 snd_bcm2835,snd_soc_core,snd_timer,snd_pcm,snd_seq,snd_seq_device,snd_com
press
spi_bcm2708        4728  0
pi@raspberrypi ~$ lsmod
Module              Size  Used by
snd_bcm2835         16165  0
snd_soc_bcm2708_i2s  5474  0
regmap_mmio         2806  1 snd_soc_bcm2708_i2s
snd_soc_core        131268  1 snd_soc_bcm2708_i2s
regmap_spi          1897  1 snd_soc_core
snd_pcm             81593  2 snd_bcm2835,snd_soc_core
snd_page_alloc      5156  1 snd_pcm
regmap_i2c           1645  1 snd_soc_core
snd_compress        8076  1 snd_soc_core
snd_seq             53769  0
snd_timer           20133  2 snd_pcm,snd_seq
snd_seq_device      6473  1 snd_seq
leds_gpio           2059  0
led_class           3688  1 leds_gpio
spi_bcm2708         4728  7 snd_bcm2835,snd_soc_core,snd_timer,snd_pcm,snd_seq,snd_seq_device,snd_compress
pi@raspberrypi ~$

```

Danach müssen die Treiber installiert werden:

```

mkdir /py-spidev
cd /py-spidev
wget https://raw.githubusercontent.com/doceme/py-spidev/master/setup.py
wget https://raw.githubusercontent.com/doceme/py-spidev/master/spidev_module.c
sudo python setup.py install

```

## 7.3 Python Script

Das Python Script liest alle 8 ADCs aus und schreibt die Wert in eine Datei, so dass diese Später mit PHP ausgewertet werden kann. Das Script könnte noch deutlich vereinfacht werden und wird nur der Beispielhaftigkeit auf diese Weise dargestellt.

```
#!/usr/bin/python

import spidev
import time
import os

# Open SPI bus
spi = spidev.SpiDev()
spi.open(0,0)

# Function to read SPI data from MCP3008 chip
# Channel must be an integer 0-7
def ReadChannel(channel):
    adc = spi.xfer2([1, (8+channel)<<4,0])
    data = ((adc[1]&3) << 8) + adc[2]
    return data

# Function to convert data to voltage level,
# rounded to specified number of decimal places.
def ConvertVolts(data,places):
    volts = (data * 3.3) / 1023
    volts = round(volts,places)
    return volts

while True:

    # Read all 8 channels
    CH1d= ReadChannel(0)
    CH1v= ConvertVolts(CH1d,2)
    CH2d= ReadChannel(1)
    CH2v= ConvertVolts(CH2d,2)
    CH3d= ReadChannel(2)
    CH3v= ConvertVolts(CH3d,2)
    CH4d= ReadChannel(3)
    CH4v= ConvertVolts(CH4d,2)
    CH5d= ReadChannel(4)
    CH5v= ConvertVolts(CH5d,2)
    CH6d= ReadChannel(5)
    CH6v= ConvertVolts(CH6d,2)
    CH7d= ReadChannel(6)
    CH7v= ConvertVolts(CH7d,2)
    CH8d= ReadChannel(7)
    CH8v= ConvertVolts(CH8d,2)

    fADC= open("/var/www/adc.txt", "w")
    fADC.write(str(CH1v)+ ": " + str(CH2v)+ ": " + str(CH3v)+ ": " +
str(CH4v)+ ": " + str(CH5v)+ ": " + str(CH6v)+ ": " + str(CH7v)+ ": " +
str(CH8v) )
    fADC.close
    time.sleep(0.5)
```

## 8 Serielle Schnittstelle

### 8.1 Python Bibliothek für RS232/UART installieren

Installation	<code>sudo apt-get install python-serial</code>
--------------	---

#### Achtung!

Zur Nutzung der Seriellen Schnittstelle ist das Debugging des Rasperrys zu deaktivieren!

- Öffnen der Datei `/etc/inittab`
- Am Ende der Datei die Zeile:
  - `T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100`
- auskommentieren

### 8.2 Terminalprograminstallieren

Installation	<code>sudo apt-get install minicom</code>
Starten mit	<code>minicom -b 9600 -o -D /dev/ttyAMA0</code>

### 8.3 Python Scripte

Script welches die ihm übergebenen Parameter auf der UART ausgibt:

```
#!/usr/bin/python

import serial
import sys

UART = serial.Serial("/dev/ttyAMA0", 9600)
UART.open()

UART.write(sys.argv[1]+ "\n\r")
UART.close()

exit(0)
```

Script welches die Wenn über die Serielle Schnittstelle Daten ankommen und diese mit „\n“ abgesendet wurden diese in der SHELL ausgibt

```
#!/usr/bin/python

import serial
import sys

UART = serial.Serial("/dev/ttyAMA0", 9600)
UART.open()
Line=""
# Wenn exit gesendet wird, wird das Script beendet
while (Line.find("exit")):
    Line=UART.readline(None)
    print Line

UART.close()
print "EXIT"
exit(0)
```

## 9 GPIO Ausgänge

## 10 CAN BUS Interface