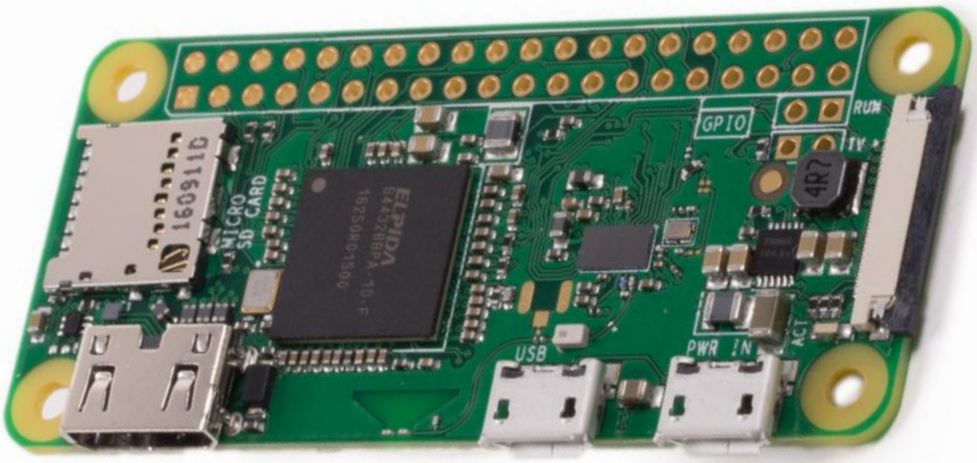
Inhaltsverzeichnis



raspberrypi.org

1 Einführung

Danke das du dich für einen Raspberry Pi aus unseren Shop entschieden hast. Mit dieser Anleitung wollen wir dir den Einstieg zum Raspberry Pi vereinfachen und ein paar Möglichkeiten zeigen.

1.1 Das Betriebssystem installieren und starten

Jeder PC benötigt ein Betriebssystem, so auch der Raspberry. Auf dem Raspberry Pi läuft kein Windows Betriebssystem, sondern ein Linux, genauer gesagt läuft hier Raspbian. Das Raspbian basiert wie auch andere bekannte Linux-Varianten (Ubuntu, Knoppix) auf Debian. Der Name Raspbian ist auch zusammengesetzt aus diesen beiden Worten **Rasp**berry & Deb**ian**. Wie auch bei einem normalen PC benötigt der Raspberry Pi eine Festplatte, wo das Betriebssystem installiert werden muss. Der unterschied hier ist, dass im Raspberry eine SD-Speicherkarte als Festplatte dient.

Die Speicherkarte muss zuerst an einem normalen PC vorbereitet werden. Dazu steckt man die Speicherkarte in einen entsprechenden Kartenleser. HINWEIS: Alle Daten die vorher auf der Speicherkarte gespeichert wurden, sind nach der Raspbian Installation gelöscht und überschrieben. Wir empfehlen die Speicherkarte nun zuerst einmal zu formatieren. Hierfür verwenden kann man die Windows Formatierungsfunktion im Windows Explorer oder das Programm „SDFormatter“ der SD Association. (https://www.sdcard.org/downloads/formatter_4/).

Anschließend laden wir uns die aktuelle Version von Raspbian auf er Raspberry Pi Webseite herunter (https://www.raspberrypi.org/downloads/raspbian/). Die Zip Datei muss auf dem Rechner entpackt werden. Die nun erhaltene Imagedatei wird nun auf unsere vorbereitete SD-Karte mit Win32Diskimager (https://sourceforge.net/projects/win32diskimager/files/latest/download) übertragen. Es gibt auf der Raspberrypi.org Seite 2 Versionen. Einmal eine Light Version und eine „Vollversion“. Wir verwenden die Vollversion, denn diese enthält die Grafische Oberfläche, die Lite hat nur die Konsole zur Verfügung.

Damit wir den Raspberry Pi ohne Display und Tastatur „Headless“ betreiben können, müssen wir vor dem starten auf die SD-Karte Dateien ablegen.

SSH:

Seit der Version 2016-11-25 wurde der SSH-Server deaktiviert. Um SSH zu aktiveren legen wir eine Datei mit „ssh“ als Namen auf der „Boot Partition“ der SD Karte an. Mit dieser Datei sagen wir dem Raspberry Pi, dass er uns SSH aktivieren soll und wir dies explizit wünschen. Dazu öffnen wir im Arbeitsplatz die SD Karte (Boot) und klicken einmal mit der rechten Maustaste auf „Neu > Neues Textdokument“. Das neue Textdokument.txt benennen wir einfach um, mit rechter Maustaste „umbenennen“ nennen wir es einfach ssh. Windows wird dann eine Meldung bringen, dass die Datei unbrauchbar wird, diese Meldung bestätigen wir mit Ja, wir wollen die Datei umbenennen. Die Datei wird dadurch für uns erst brauchbar. (Hinweis: Dateiendungen müssen angezeigt werden; in den Ordneroptionen kann das eingestellt werden!)

WLAN:

Die neuen Raspberry Pi Modelle besitzen WLAN on Board. Damit dies auch gleich zur Verfügung steht, konfigurieren wir uns das auch schon vor dem starten:

Wir legen uns wieder eine neue Datei an, und benennen diese zu „wpa_supplicant.conf“ um. Diese Datei öffnen wir mit einem Texteditor (Notepad++) und schreiben folgende Zeilen hinein:

```
country=DE
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    ssid="WLAN SSID"
    scan_ssid=1
    psk="WLAN PASSWORT"
    key_mgmt=WPA-PSK
}
```

Die Datei muss Linux Zeilenende besitzen, in Notepad++ unter Bearbeiten > Format Zeilenende > UNIX
WLAN SSID und WLAN PASSWORT muss vor dem abspeichern auf deine Daten abgeändert werden!!

Nun können wir die SD-Karte in den Raspberry Pi einschieben und starten.

Ist der Raspberry Pi hochgefahren und mit dem Netzwerk verbunden, können wir mit SSH auf den Raspberry Pi zugreifen. Wir verwenden hier das Tool PuTTY und WinSCP. Mit PuTTY können wir die Kommandozeile Remote auf unseren Computer holen und mit WinSCP später auch Dateien übertragen.

Wir laden uns die Software PuTTY auf http://www.putty.org/ herunter. Gleichzeitig laden wir uns für später auch WinSCP von https://winscp.net/eng/download.php ,

Jetzt brauchen wir noch die IP des Raspberry Pi. Diese bekommen wir von unserem Router. Dazu loggen wir uns auf der Oberfläche des Routers ein. Solltest du z.B. eine Fritz!Box haben, so gehe auf die Website http://fritz.box/ , bei einem SpeedPort von der Telekom lautet die Adresse http://speedport.ip/ .

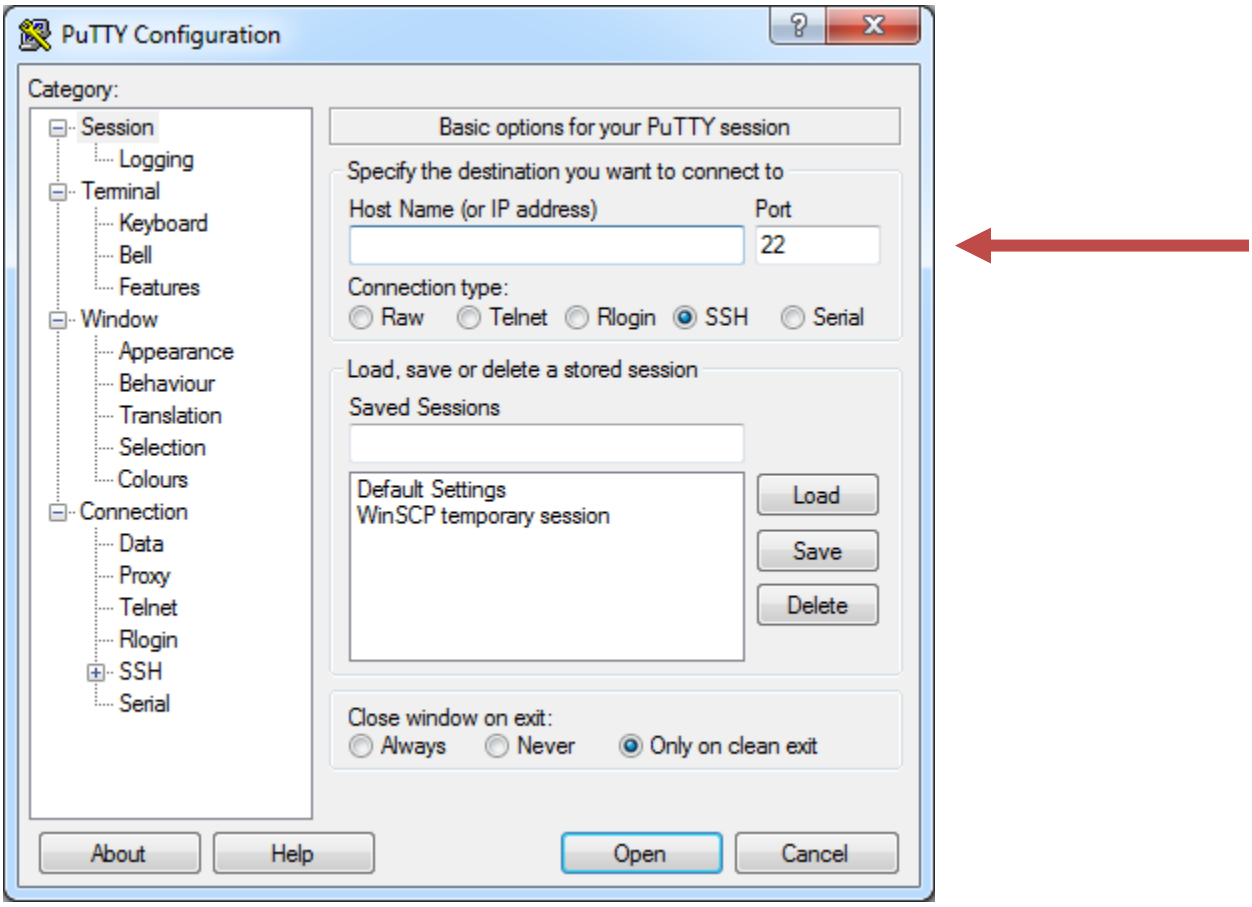
Du kannst aber auch deine IP-Adresse des Computers anzeigen lassen. Starte die Eingabeaufforderung unter Start>Alle Programme>Zubehör>Eingabeaufforderung. Darin geben wir nun „ipconfig“ ein.

```
C:\Users\Florian>ipconfig

Windows-IP-Konfiguration


Drahtlos-LAN-Adapter Drahtlosnetzwerkverbindung:

   Verbindungsspezifisches DNS-Suffix: fritz.box
   Verbindungslokale IPv6-Adresse  . : fe80::cd45:8dae:2a97:dffd%14
   IPv4-Adresse  . . . . . . . . . . : 192.168.31.22
   Subnetzmaske  . . . . . . . . . . : 255.255.255.0
   Standardgateway . . . . . . . . . : 192.168.31.1
```

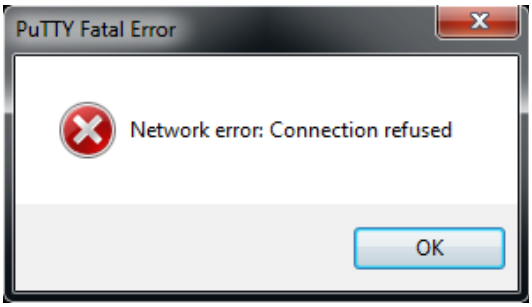
Das Ergebnis sollte so aussehen. Es können aber auch mehrere Netzwerkgeräte angezeigt werden. In der letzten Zeile haben wir das Standardgateway, dieser ist in der Regel unser Router. Geben wir diese IP im Browser ein kommen wir auch auf unseren Router.

1.2.1 Login mit SSH

Wir starten die Software PuTTY:



Bei dem Textfeld „Host Name (or IP address)“ geben wir die IP von unserem Raspberry Pi ein. Der Port (22) bleibt unverändert. Anschließend starten wir mit Open die Verbindung.

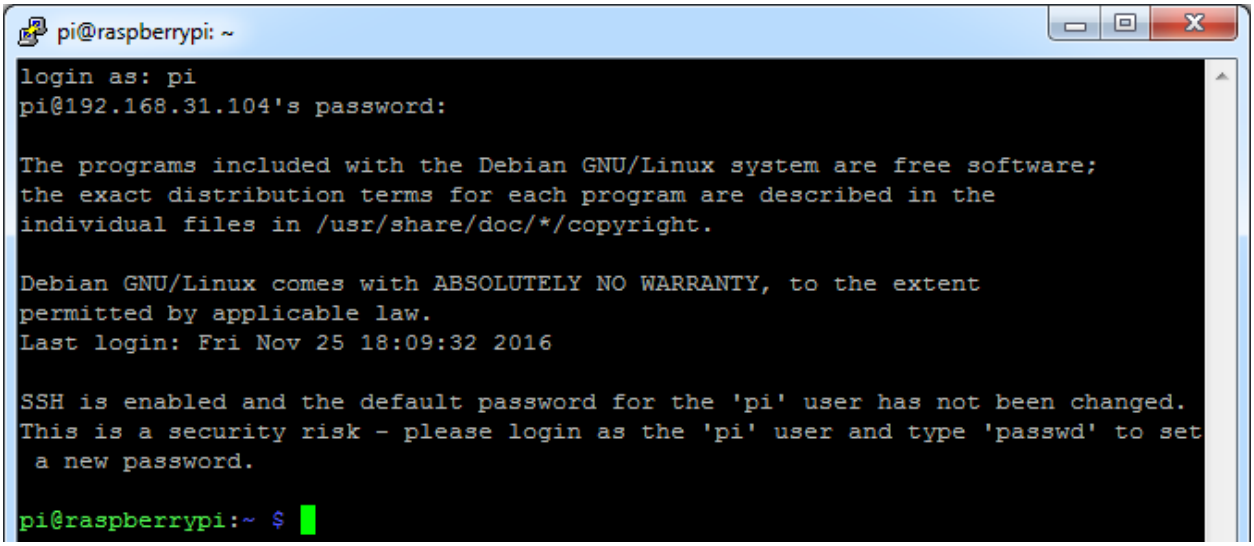


Nachdem wir die SD Karte wieder in den Raspberry Pi gesteckt haben und hochgefahren haben, können wir uns mit PuTTY einloggen.

Erscheint die rechts stehende Fehlermeldung, dann ist der SSH Server nicht aktiv, entweder fehlt die oben beschriebene SSH Datei oder die Datei ist falsch (mit Dateiendung .txt) benannt oder abgelegt.

Login: pi

Password: raspberry



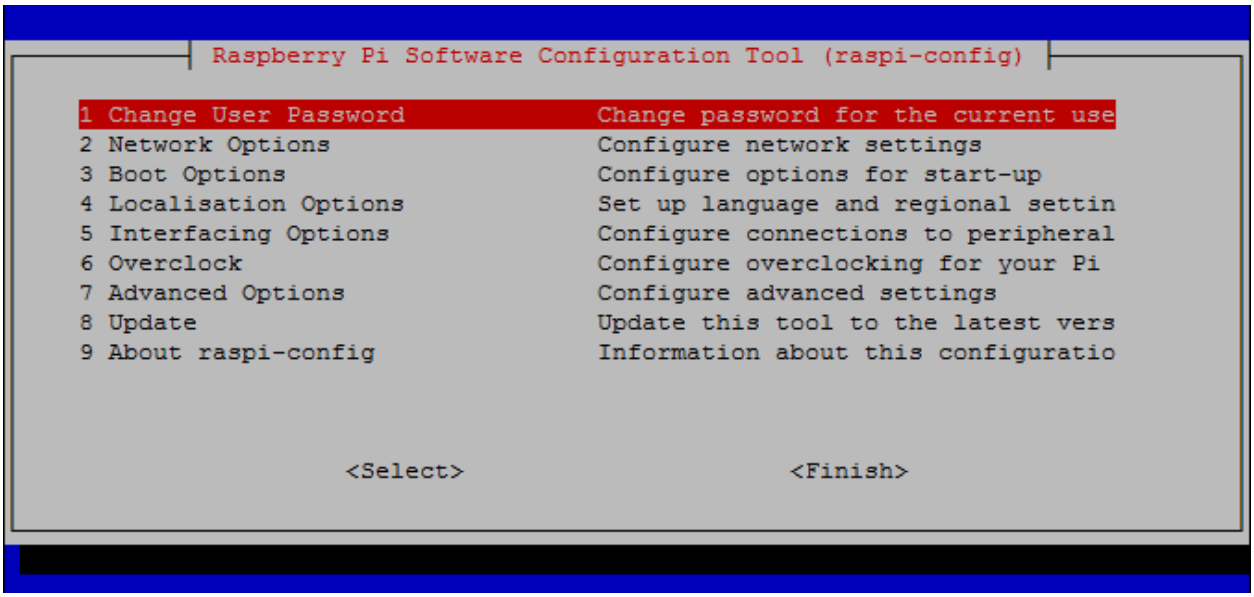
Und schon begrüßt uns der Raspberry und wir sind erfolgreich verbunden. Nun können wir mit dem Raspberry Pi arbeiten.

1.2.2 Grundeinstellungen des Raspberry Pi

Nach dem ersten verbinden mit dem Raspberry sollten wir ein paar Einstellungen vornehmen. Dazu starten wir die Raspberry Konfigurationsoberfläche: `sudo raspi-config`

sudo	**als Administrator ausführen**
raspi-config	**Befehl zum starten des Konfiguration Tools**

Es erscheint folgendes Fenster:



Wir können hier mit den Pfeiltasten ← ↑ ↓ → Navigieren, mit Enter wählen wir aus.

1 Change User Passwort **Das Standardpasswort ist unsicher, hier ändern wir das Passwort**

2 Network Options Netzwerkoptionen
- N1 Hostname — Name des Raspberry Pi im Netzwerk
- N2 Wi-fi — WLAN Netzwerk und Passwort Konfigurieren
- N3 Network interface names — Netzwerkinterface Name ändern

3 Boot Options Hier benötigen wir noch keine Änderungen, zur Auswahl stehen:
- B1 Automatischer Start in Konsole/Desktop
- B2 Warten auf Netzwerk
- B3 Splash Screen aktivieren/deaktivieren

4 Localisation Options
- I1 Change Locale — Wir möchten den Raspberry auf Deutsch verwenden:
  Wir suchen „de_DE.UTF-8 UTF-8“ und wählen es mit der Leertaste an
  Nun noch einmal de auswählen und fertig.
- I2 Change Timezone — Wir befinden uns in der Zeitzone Europe – Berlin
- I3 Keyboard Layout — Tastaturlayout ändern
- I4 Change Wi-fi Country — DE Germany auswählen

5 Interfacing Options
- P1 Camera — Aktivieren des Raspberry Pi Kameramoduls
- P2 SSH — Aktivieren / Deaktivieren des SSH-Servers (alternative zur Datei wie in 1.2.1 beschrieben)
- P3 VNC — Den VNC Server aktiveren wir, damit wir später grafisch arbeiten können
- P4 SPI — Aktiveren der SPI Hardware Schnittstelle

P5	I2C	Aktiveren der I²C Hardware Schnittstelle
P6	Serial	Aktiveren der Kommandozeile über RS232
P7	1-Wire	Aktiveren der 1-Wire Hardware Schnittstelle
P8	Remote GPIO	GPIO Ports für einen Webzugriff freigeben

6 Overclock — alte Raspberry Pis können hier getuned werden (übertakten der CPU)

!! hier nichts Einstellen außer man weiß was man tut !!
!! teilweise sind hier aktive Kühlkomponenten notwendig !!
!! Der Raspberry kann dadurch beschädigt werden !!
!! Aktuelle Raspberry Pis können nicht mehr getuned werden !!

7 Advanced Options

A1	Expand Filesystem	Hier wird die gesamte SD Karte für das System verwendet
A2	Overscan	Deaktivieren, wenn ein schwarzer Rand am Bildschirm angezeigt wird
A3	Memory Split	Speicher für den Grafik-Prozessor, 64MB sollten ausreichen
A4	Audio	Standardausgabegerät (Auto/HDMI/Klinke)
A5	Resolution	Bildschirmauflösung einstellen
A6	Pixel Doubling	2x2 Pixel werden zusammengefasst
A7	GL Driver	Experimental Treiber aktivieren für den Desktop

8 Update — Updates installieren / bei der Erst-Konfiguration ausführen!
Es werden mehrere Systemkomponenten aktualisiert.

Keyboard Layout — können wir remote nicht einstellen, eine Einstellung hat keine Auswirkung
hier könnte man QWERTY auf QWERTZ umstellen

Was müssen wir nun alles einstellen:

1 Change User Passwort **(das Passwort ändern wir auf ein eigenes)**

2 (N1) Hostname **(raspberrypi-NAME)**

4 (I1) Change Locale **(de_DE.UTF-8 UTF-8)**

4 (I2) Change Timezone **(Europe – Berlin)**

4 (I3) Change Wi-Fi Country **(DE Germany)**

5 (P3) VNC **(aktivieren)**

7 (A1) Expand Filesystem **(wir wollen die gesamte Karte verwenden)**

8 Update **(Updates sollten immer installiert werden)**

Achtung nach einem Update kann die Reihenfolge und auch die Unterpunkte anders angeordnet oder nicht mehr vorhanden sein. Wir arbeiten zum Zeitpunkt dieser Anleitung mit der Version vom 18.04.2018.

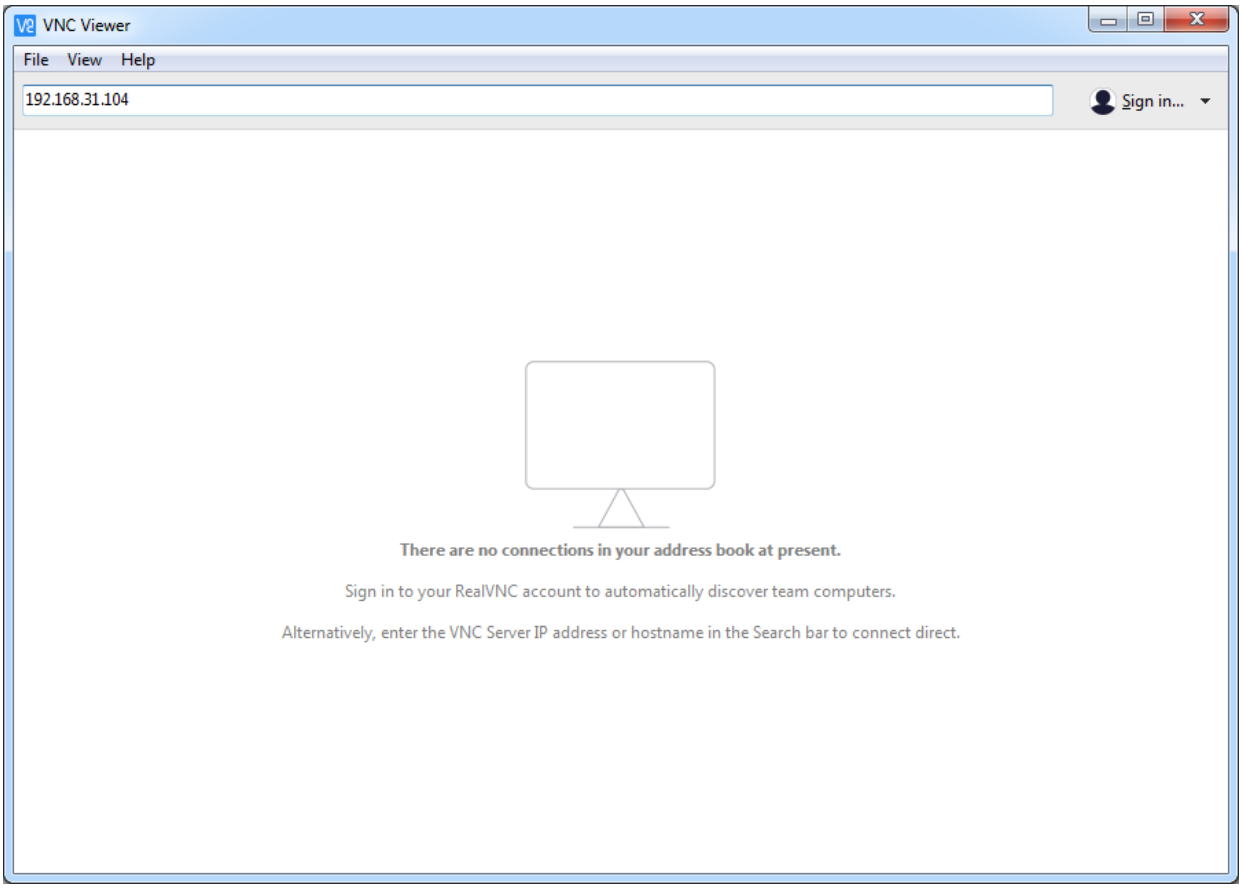
Dies gilt für die gesamte Anleitung. Es ändert sich fast täglich etwas, wenn etwas nicht exakt so funktioniert wie es hier beschrieben ist, liegt das sehr wahrscheinlich an einem neueren Update auf deinem Raspberry. Ebenso gilt das auch für die verwendeten Programme.

Nun beenden wir das Konfiguration Tool mit <Finish> und machen einen Neustart: `sudo reboot`

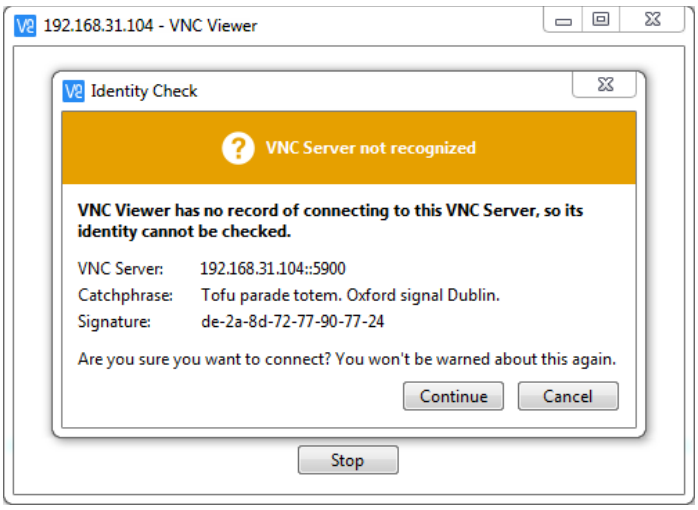
reboot	**Befehl für ein System Neustart**
halt	**Befehl zum System Herunterfahren**

1.2.3 Remote Zugriff auf den Desktop

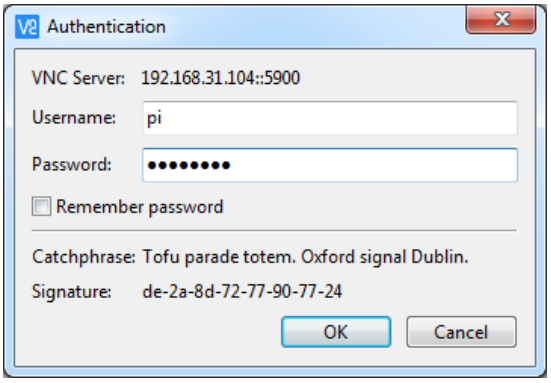
Nach dem Neustart starten wir das Tool VNC Viewer (https://www.realvnc.com/download/viewer/)



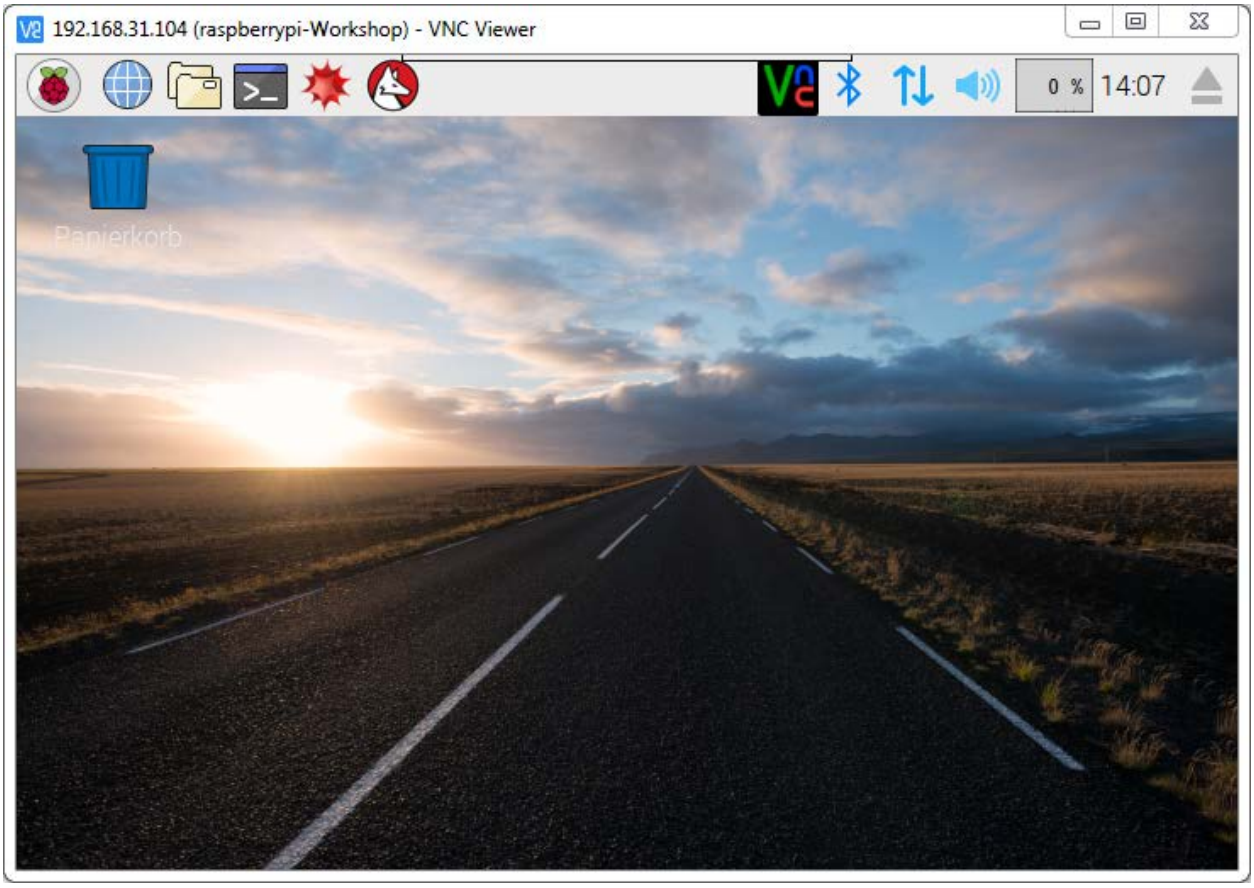
In der ersten Zeile oben geben wir die IP des Raspberry Pi ein und drücken Enter.



Die Sicherheitsabfrage bestätigen wir mit Continue und geben im Anmeldefenster unsere Zugangsdaten ein:



Jetzt können wir zum ersten Mal den Desktop des Raspberry Pi sehen:



1.2.4 Auflösung des Bildschirms ändern

Anschließend stellen wir die Bildschirmauflösung etwas höher ein.

Hierfür öffnen wir die Konsole (4. Symbol von links in der Taskleiste) und geben diesen Befehl ein:

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

nano	**einfacher Texteditor für die Konsole**
/boot/config.txt	**Datei die geöffnet werden soll**

Jetzt suchen wir die Zeile „**# hdmi_force_hotplug=1**“ und entfernen die Raute am Anfang der Zeile. Ebenso bei „**# hdmi_group=1**“ und „**# hdmi_mode=1**“. hdmi_group stellen wir **2** und bei hdmi_mode auf **23 (1280x768 / 60 Hz)**.

hdmi_force_hotplug = 1	**HDMI ohne Monitor verwenden**	
hdmi_mode=1	**CEA (Consumer Electronics Association)**	**Fernseher**
hdmi_mode=2	**DMT (Display Monitor Timing, VESA Standard)**	**PC-Bildschirm**
hdmi_mode=16	**Auflösung: 1024x768 / 60 Hz**	
hdmi_mode=23	**Auflösung: 1280x768 / 60 Hz**	
hdmi_mode=81	**Auflösung: 1366x768 / 60 Hz**	
hdmi_mode=82	**Auflösung: 1920x1080 / 60 Hz (1080p)**	

Nach einem Neustart haben wir nun auch über VNC ein größeres Fenster mit der gewünschten Auflösung.

Weitere Modes siehe http://elinux.org/RPiconfig#Video_mode_options

1.3 Programme auf dem Raspberry

Auf dem Raspberry Pi sind schon viele Programm mitgeliefert, so ist z.B. auf dem Raspberry Pi das LibreOffice installiert. Diese Software ist vergleichbar mit dem Microsoft Office Packet. Somit könnten wir den Raspberry Pi als normalen Arbeitscomputer für Büro und Schule verwenden. Word, Excel und PowerPoint kennt normalerweise jeder, bei LibreOffice nennt man diese Programme, Calc, Writer und Impress. Des Weiteren gibt es bei LibreOffice Programme für Mathematik (Math), Zeichnen (Draw) und für Datenbanken (Base).

Zum Surfen im Internet ist der Chromium-Webbrowser vorinstalliert, dieser ist die Linux Version vom Chrome-Browser.

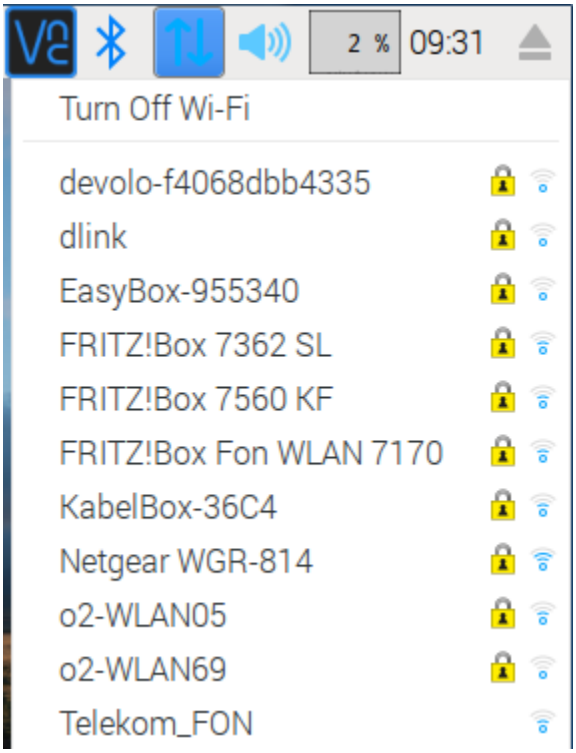
Auch sind einige einfache Spiele mitgeliefert. Diese Spiele sind auf der Basis von Python programmiert. Z.B. gibt es hier Vier gewinnt, Memory, Schieberätsel, Tetris, Snake…)

PDF Viewer, Taschenrechner, Texteditor, Bildanzeige Software und Komprimierungsprogramm gehören inzwischen auf jeden PC und sind ebenfalls vorinstalliert.

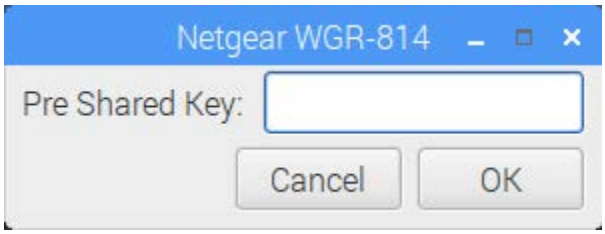
Zu guter Letzt sind auch eine Programme zur Programmentwicklung mit dabei, zu erwähnen wäre hier: Java, Python, SonicPi und Scratch.

1.4 WLAN einrichten/verbinden

Wenn wir oben rechts auf unserem Raspberry Pi sehen, haben wir ein Symbol mit 2 Pfeilen ↑↓ Klicken wir darauf bekommen wir eine ganze Liste mit verfügbaren WLAN Netzwerken in der Umgebung angezeigt. Wählen wir unser WLAN aus:



Es erscheint ein Eingabefenster, in dieses wir unser Passwort eingeben:



Nach dem Bestätigen mit OK und erfolgreicher Verbindung mit dem WLAN, wechselt das Pfeilsymbol zu einem WLAN Symbol.

Der Raspberry ist nun mit dem WLAN verbunden, das Netzwerkkabel kann entfernt werden.

1.5 Raspberry Pi auf aktuellen Stand bringen

Bevor wir nun neue Software installieren können müssen wir unseren Raspberry Pi aktualisieren und evtl. Updates installieren. Dazu gibt es mehrere Befehle die man ausführen muss. Geben wir nun diese Befehle ein:

```
sudo apt-get update
```

```
sudo apt-get upgrade
```

apt-get	**Programm zur Paket- / Softwareverwaltung**
update	**Abgleich der lokalen Paketliste mit verfügbarer Liste aus dem Internet**
upgrade	**Überprüfen der installierten Software und Installation der Updates**

Um auch gleichzeitig mit dem Update das System von nicht mehr benötigten Installationen zu bereinigen verwende ich gerne diese Befehlszeile:

```
sudo apt-get update && sudo apt-get upgrade && sudo apt-get dist-upgrade && sudo apt-get autoremove && sudo apt-get autoclean
```

dist-upgrade	**wie upgrade, nur können Paket auch gelöscht werden**
autoremove	**löschen von ungenutzten Zusatzpaketen (z.B. nach Update von Software ist ein Bestandteil nicht mehr notwendig)**
autoclean	**löschen des Paketcache und löschen nicht mehr vorhandener Software**

Jetzt befindet sich der Raspberry Pi auf dem aktuellsten Stand und wir können neue Software installieren.

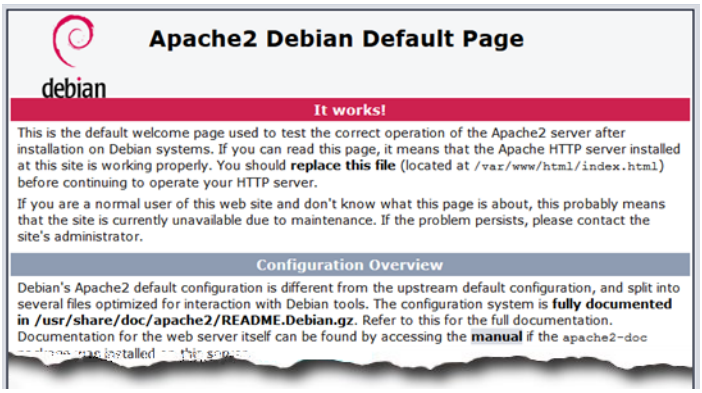
1.6 Webserver installieren

Wenn wir eine Website hosten möchten oder Steuerungsaufgaben durchführen können, können wir dies mit einem Webserver machen.

Dazu installieren wir uns den Apache Webserver (Version 2) und das dazugehörige Pakete PHP5 inkl. der Integration zu Apache. Der Befehl hierzu:

```
sudo apt-get install apache2 php libapache2-mod-php
```

Nachdem die Installation abgeschlossen wurde, können wir die IP von unserem Raspberry Pi im Browser eingeben. Es sollte nun folgende „It works“-Website angezeigt werden:



Legen wir nun in dem neu erstellten Verzeichnis von Apache „/var/www/html/“ eine Datei mit dem Namen index.php an. Leider müssen wir feststellen, dass uns mit dem Benutzer pi die Berechtigungen fehlen.

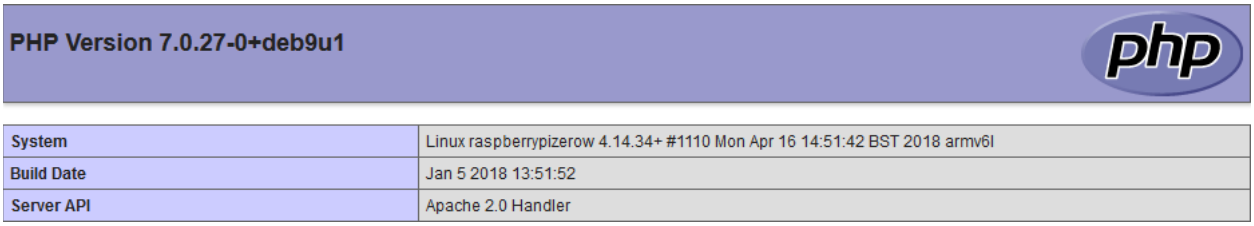
Dazu geben wir in der Konsole folgenden Befehl ein:

```
sudo chown -R pi /var/www
```

Damit setzten wir den Benutzer pi mit als Eigentümer auf den Ordner /var/www/. Jetzt lässt sich die neue Datei anlegen. In die neu erstellte index.php Datei schreiben wir folgendes:

```php
<?php
phpinfo();
?>
```

speichern dies ab und öffnen die Website im Browser: http://[IP-Raspberry]/index.php



System	Linux raspberrypizerow 4.14.34+ #1110 Mon Apr 16 14:51:42 BST 2018 armv6l
Build Date	Jan 5 2018 13:51:52
Server API	Apache 2.0 Handler

Es öffnet sich eine Übersichtseite mit allen PHP Einstellungen. Damit konnten wir überprüfen ob auch PHP richtig installiert wurde.

Jetzt können wir unsere eigenen Homepages erstellen.

2 GPIO

Die eigentliche Besonderheit des Raspberry Pi ist weder seine winzige Größe noch sein Preis – die riesige Faszination des Raspberry Pi geht vielmehr von den 40 Pins aus, die zur Steuerung elektronischer Geräte verwendet werden können. Sowohl Elektronik-Bastler als auch Embedded-Linux-Profis bekommen mit dem Raspberry Pi ein Spielzeug bzw. Werkzeug in die Hand, das die Entwicklung computergesteuerter Geräte so einfach wie selten zuvor macht.

Die Platine des Raspberry Pi enthält in einer Ecke eine Steckerleiste mit 2x20 Kontakten. Der Rasterabstand beträgt 2,54 mm. Diese Steckerleiste stellt die Basis für die weitergehenden Projekte dar. Fast alle Hardware-Basteleien starten hier. Die Steckerleiste stellt neben einigen allgemein verwendbaren Kontakten (General Purpose Input/Output = GPIO) auch zwei Versorgungsspannungen (3,3 V bzw. 5 V) sowie Masse (also 0 V) zur Verfügung.

Leider gibt es unterschiedliche Bezeichnungen der Pins, die in der Praxis viele Verwechslungen verursachen. Darauf werden wir aber später noch stoßen.

Die Pins dürfen zusammen maximal mit 50mA belastet werden. Die Pins werden über eine selbstrückstellende Sicherung (Poly Fuse) geleitet. Fließt hier zu viel Strom, schaltet sich der Raspberry Pi für eine Weile ab. Mit etwas Glück kommt es zu keinen bleibenden Schäden.

Wenn wir GPIO-Kontakte zur Steuerung verwenden (Konfiguration als Output), beträgt die Spannung am betreffenden GPIO-Pin 3,3 V. Der Steuerungsstrom pro Pin sollte 16mA nicht überschreiten bzw. 50mA für alle GPIOs. Wir müssen also geeignete Vorwiderstände verwenden!

Wirklich klare Angaben zum maximal erlaubten GPIO-Strom sind nicht angegeben. Aus Experimenten von Raspberry-Pi Anwendern geht aber hervor, dass der Raspberry Pi auch bei einem etwas höheren Strom nicht gleich beschädigt wird bzw. die Ausgangsspannung dann entsprechend absinkt, um die Leistung zu begrenzen. Wir sollten uns aber die 50mA Gesamtstrom halten, um Schäden zu vermeiden.

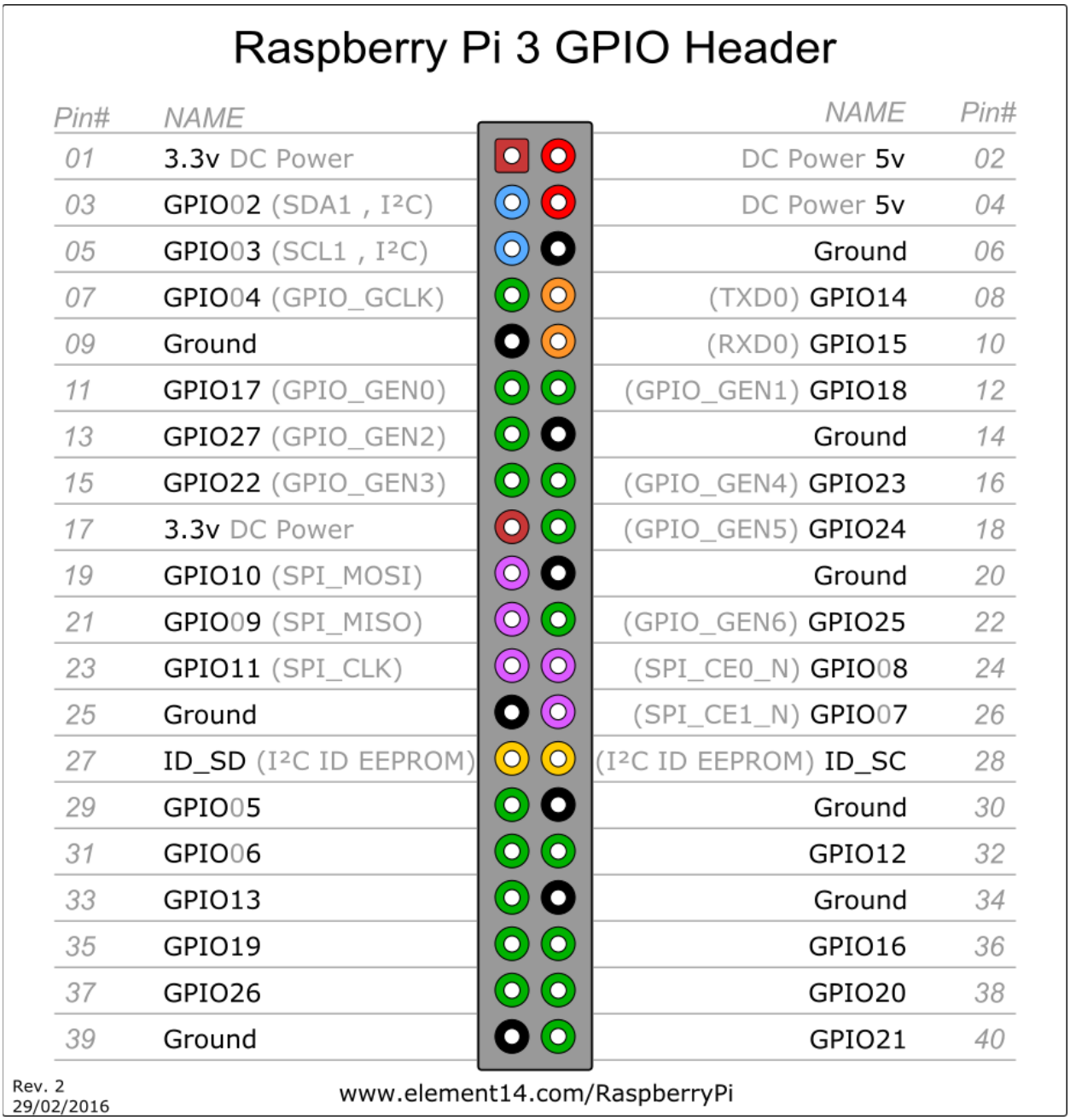
Viele Pins erfüllen je nach Programmierung alternative Funktionen. Vor jedem Projekt müssen wir uns die Frage stellen: Welche der vielen GPIO-Pins verwenden wir? Solange es nur darum geht, nur kleine Experimente durchzuführen und ein paar Leuchtdioden ein- und auszuschalten, können wir die GPIO-Pins frei auswählen. Diverse Spezialfunktionen stehen allerdings auf ausgewählten Pins zur Verfügung und sollten wir diese später einmal Verwenden wollen, sollten wir diese jetzt noch nicht verwenden, erst wenn uns die GPIO-Pins ausgehen. Als Beispiel können die Pins 3 und 5 auch für den I²C-Bus verwendet werden.

Sicherheitshinweise

bei dem Umgang mit den GPIO:

- **Kurzschlüsse und falsche Beschaltung von Pins können den Raspberry Pi beschädigen!**
- **Raspberry Pi ausschalten, wenn am Steckboard die Schaltung geändert wird**
- **Die maximale Spannung an den GPIO-Pins beträgt 3,3 Volt. Niemals 5 Volt mit GPIO verbinden**

Hier eine Übersicht aller GPIOs:



Raspberry Pi 3 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I²C)	DC Power 5v	04
05	GPIO03 (SCL1 , I²C)	Ground	06
07	GPIO04 (GPIO_GCLK)	(TXD0) GPIO14	08
09	Ground	(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)	(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)	Ground	14
15	GPIO22 (GPIO_GEN3)	(GPIO_GEN4) GPIO23	16
17	3.3v DC Power	(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)	Ground	20
21	GPIO09 (SPI_MISO)	(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)	(SPI_CE0_N) GPIO08	24
25	Ground	(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)	(I²C ID EEPROM) ID_SC	28
29	GPIO05	Ground	30
31	GPIO06	GPIO12	32
33	GPIO13	Ground	34
35	GPIO19	GPIO16	36
37	GPIO26	GPIO20	38
39	Ground	GPIO21	40

Rev. 2
29/02/2016

www.element14.com/RaspberryPi

2.1 GPIO als Ausgang

Beginnen wir eine LED an unseren Raspberry Pi anzuschließen und diese auch zu programmieren.

Der maximale Strom, den der Raspberry Pi an den GPIO liefern kann, ist 50mA an allen Pins zusammen! Daher verwenden wir eine LED, die nur wenig Strom verbraucht.

Damit wir den benötigten Vorwiderstand für unsere LED berechnen können, müssen wir die Daten der LED erst einmal im Datenblatt suchen.

Wir verwenden die Rote LED, Kingbright L-53LID.

Suchen wir als nächstes einmal das Datenblatt im Internet für diese LED. Wir benötigen die Spannung und den Strom die diese LED benötigt.

Aus dem Datenblatt können wir entnehmen:

I_F = 2mA V_F = 2,0-2,5V

Mit diesen Werten und dem Ohm'schen Gesetz lässt sich unser Vorwiderstand berechnen.

$$U = R * I \quad => \quad R = \frac{U}{I} \quad => \quad I = \frac{U}{R}$$

Unser Raspberry Pi liefert 3,3V, aber unsere LED benötigt nur 2V. Ziehen wir diese ab:

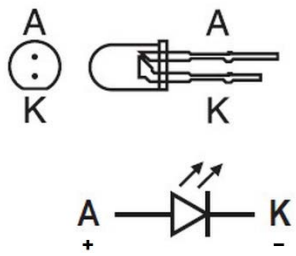
$$3{,}3V - 2V = 1{,}3V$$

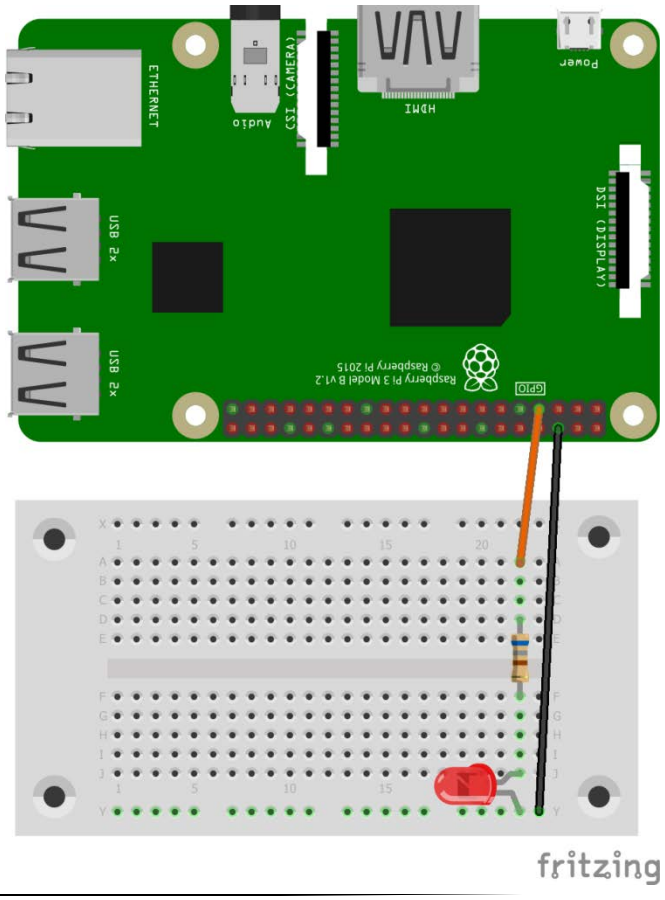
Also muss unser Vorwiderstand 1,3V verbrauchen. Der Strom beträgt 2mA, setzen wir das in die Formel:

$$R = \frac{1{,}3V}{2mA} = \frac{1{,}3V}{0{,}002A} = 650\Omega$$

Somit kommen wir auf einen Vorwiderstand von 650Ω. Diesen Widerstand wird es aber nicht geben und wir suchen uns den nächst höheren Wert heraus, das sind 680Ω.

Bei der LED muss unbedingt auch die Polarität beachtet werden, hier eine kurze Erklärung:





Mit 2 Verbindungsbrücken verbinden wir unser Steckboard mit dem Raspberry Pi. So wie es rechts auf dem Bild zu sehen ist:

Nachdem die Hardware fertig verdrahtet wurde starten wir unseren Raspberry und logen uns wieder mit SSH darauf ein.

Nun versuchen wir die erste einfachste Möglichkeit auf den GPIO-Pin zuzugreifen. Wir haben unsere LED an GPIO4 angeschlossen. Dann geben wir folgende befehle in die Konsole ein:

```
sudo echo "4" > /sys/class/gpio/export

sudo echo "out" > /sys/class/gpio/gpio4/direction

sudo chmod 666 /sys/class/gpio/gpio4/value

sudo chmod 666 /sys/class/gpio/gpio4/direction

echo "1" > /sys/class/gpio/gpio4/value
```

echo “4“ > xx	**schreibe eine 4 in die Datei xx (initialisieren des GPIO 4)**
echo “out“ > xx	**schreibe out in die Datei xx (GPIO 4 als Ausgang definieren)**
chmod	**Berechtigungen ändern**
echo “1“ >	**schreibe eine 1 in die Datei xx (GPIO 4 einschalten)**

Mit dem letzten Befehl (echo "1" > /sys/class/gpio/gpio4/value) schreiben wir in die Datei den Ausgangswert unseres GPIO4. Schreiben wir eine 0, so schalten wir den Pin aus.

```
echo "0" > /sys/class/gpio/gpio4/value
```

Hinweis: Nach einem Neustart des Raspberry Pi muss die Initialisierung erneut durchgeführt werden!

Im nächsten Kapitel werden wir eine andere Möglichkeit der Ansteuerung verwenden und die LED auch blinken lassen.

2.2 LED Blinklicht

Bis jetzt können wir die LED nur manuell Ein oder Aus schalten. Aber die LED blinkt noch nicht.

Um auch eine neue Methode des Zugriffs auf die GPIO zu geben, verwenden wir jetzt wiringPi.

Die wiringPi ist eine Software die uns die Ansteuerung „vereinfacht“.

Installation von wiringPi falls `gpio -v` keine Informationen ausgibt:

`gpio -v`	Version prüfen
`sudo apt-get purge wiringPi`	Deinstallieren von alter Version
`sudo apt-get install git-core`	Installieren von git
`git clone git://git.drogon.net/wiringPi`	wiringPi herunterladen
`cd ~/wiringPi`	in das Verzeichnis wechseln
`git pull origin`	prüfen auf aktuelle Version
`cd ~/wiringPi`	in das Verzeichnis wechseln
`./build`	wiringPi Kompilieren
`sudo ./build`	wiringPi Installieren
`sudo reboot`	Raspberry Pi neustarten

geben wir nun den Befehl:

`gpio readall`

ein, so bekommen wir folgende Ausgabe:

```
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 0 | IN   | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | IN   | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

Eine Übersicht mit allen GPIO inklusive den Portnummern in wiringPi, diese unterscheiden sich zur GPIO Bezeichnung. Unser GPIO4 wird nun in wiringPi der Port 7.

Es wird Zeit das erste Programm zu schreiben.

`touch blink.py` Anlegen einer neuen Datei „blink.py“

`nano blink.py` Datei im Editor öffnen

Dann fügen wir folgenden Code in den Editor ein:

```
#!/usr/bin/python
from time import sleep
import RPi.GPIO as GPIO             # Module einbinden
GPIO.setmode(GPIO.BCM)              # Pin Beschreibung auf BCM
GPIO.setup(4,GPIO.OUT)              # GPIO4 als Ausgang festlegen
while True:                         # Schleife generieren
        GPIO.output(4,GPIO.HIGH)    # LED EIN
        sleep(0.1)                  # warte 0.1s
        GPIO.output(4,GPIO.LOW)     # LED AUS
        sleep(0.1)                  # warte 100ms
```

Das Programm führen mit dem Befehl

`python blink.py`

aus. Die LED blinkt nun solange bis wir das Programm mit STRG+C beenden.

Versuchen wir nun einmal eine 2. LED anzuschließen und die beiden LEDs abwechselnd blinken zu lassen.

Verwenden wir nun die Gelbe LED.

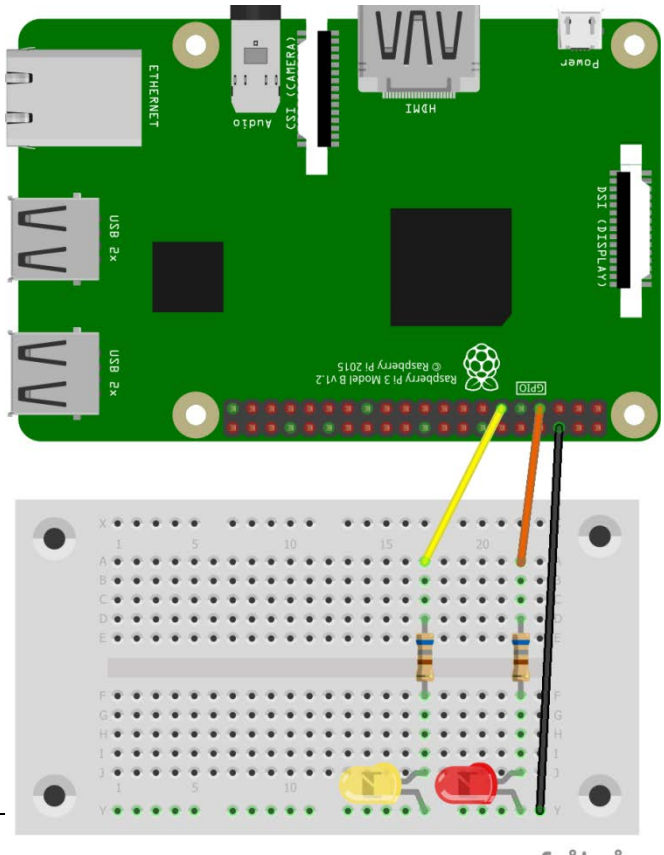
Laut Datenblatt der Kingbright L-53LYD hat die LED diese Werte:

I_F = 2mA V_F = 2,1-2,5V

$$R = \frac{3{,}3V - 2{,}1V}{2mA} = \frac{1{,}2V}{0{,}002A} = 600\Omega$$

Auch hier verwenden wir einen 680Ω Widerstand.

Ergänzen wir nun unsere Schaltung:



Wir erstellen auch eine neue Datei:

`touch blink1.py`

`nano blink2.py`

Mit folgendem Inhalt:

```
#!/usr/bin/python
from time import sleep
import RPi.GPIO as GPIO                 # Module einbinden
GPIO.setmode(GPIO.BCM)                  # Pin Beschreibung auf BCM
GPIO.setup(4,GPIO.OUT)                  # GPIO4 als Ausgang festlegen
GPIO.setup(17,GPIO.OUT)                 # GPIO17 als Ausgang festlegen
while True:                             # Schleife generieren
        GPIO.output(4,GPIO.HIGH)        # LED EIN
        GPIO.output(17,GPIO.LOW)        # LED AUS
        sleep(0.1)                      # warte 0.1s
        GPIO.output(4,GPIO.LOW)         # LED AUS
        GPIO.output(17,GPIO.HIGH)       # LED EIN
        sleep(0.1)                      # warte 100ms
```

3.2 Ampel

Mit 3 LEDs der Farben Rot, Gelb und Grün bauen wir eine Ampel. Die LEDs Rot und Gelb haben wir schon angeschlossen und berechnet. Es fehlt noch die Grüne LED. Sie hat folgende Werte:

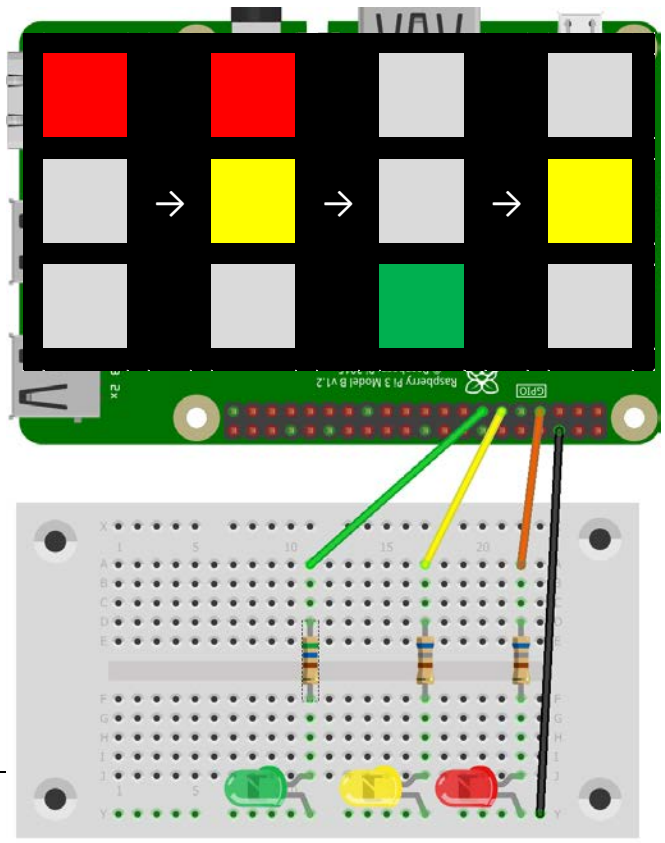
IF = 2mA VF = 2,2-2,5V $R = \frac{3{,}3V-2{,}2V}{2mA} = \frac{1{,}1V}{0{,}002A} = 550\Omega$

Bei der grünen LED benötigen wir also einen anderen Widerstand, nämlich 560Ω. Legen uns ein neues Programm an und Programmieren einen Ampelablauf:

```
touch ampel.py
```

```
nano ampel.py
```

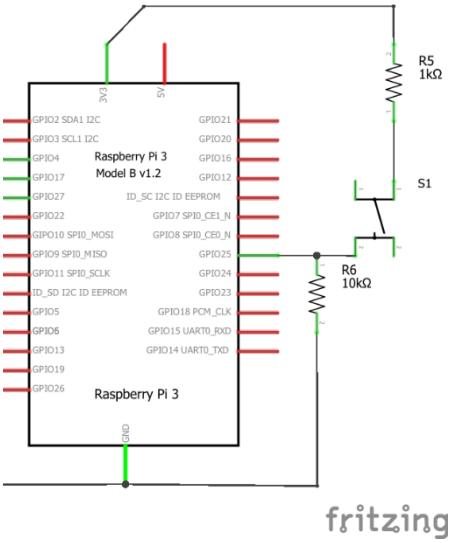
```
#!/usr/bin/python
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setup(4,GPIO.OUT)
GPIO.setup(17,GPIO.OUT)
GPIO.setup(27,GPIO.OUT)
while True:
        GPIO.output(4,GPIO.HIGH)
        GPIO.output(17,GPIO.LOW)
        GPIO.output(27,GPIO.LOW)
        sleep(10)
        GPIO.output(17,GPIO.HIGH)
        sleep(2)
        GPIO.output(4,GPIO.LOW)
        GPIO.output(17,GPIO.LOW)
        GPIO.output(27,GPIO.HIGH)
```



```
sleep(10)
GPIO.output(17,GPIO.HIGH)
GPIO.output(27,GPIO.LOW)
sleep(2)
```

2.3 GPIO als Eingang

Die Ampel steht an einer Nebenstraße und muss nicht immer schalten. In der Straße ist ein Sensor eingebaut der Autos erkennt. Erst wenn ein Auto über die Ampel fahren möchte, soll die Ampel nun umschalten. Den Sensor simulieren wir nun mit einem Taster. Bei Eingängen muss man darauf achten, das ein GPIO-Pin nicht schwebt, das bedeutet, nichts angeschlossen ist. Ein schwebender Anschluss kann durch einen angeschlossenen Draht Signale aus der Luft empfangen und der Raspberry Pi wertet evtl. die Signale als eine Tasterbetätigung aus. Dies führt zu ungewolltem programmverhalten und Fehlersuche. Deswegen verwenden wir folgende Schaltung für unseren Taster:



Die Widerstände vor und nach dem Taster haben eine besondere Bedeutung. Der Widerstand mit 10kΩ ist ein sogenannter Pull-Down Widerstand. Ist der Taster nicht betätigt, liegt die Masse am GPIO an und zieht das Potential auf 0V (Masse). Im Gegensatz dazu gibt es auch einen Pull-Up Widerstand, dieser wäre zwischen GPIO und 3,3V Versorgungsspannung und dieser hebt das Potenzial am GPIO auf die 3,3V.

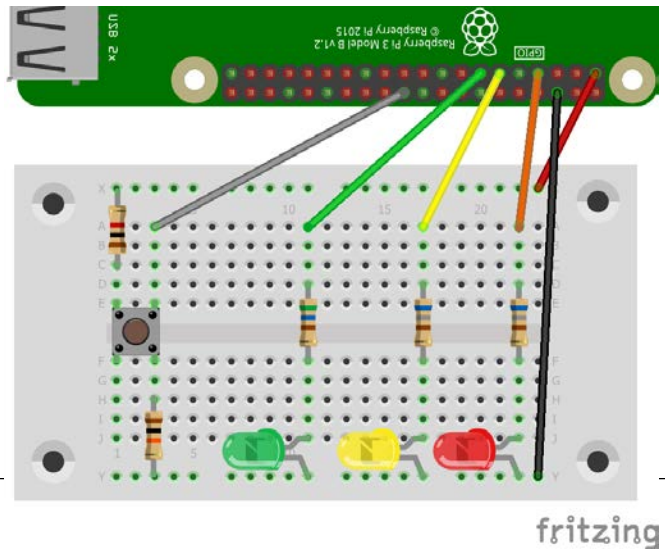
Der 2. Widerstand ist nur ein Schutzwiderstand für unseren Raspberry Pi. Wie oben schon beschrieben wurde, liefert der Raspberry Pi nur 50mA und wenn mehr Strom fließt löst die Poly-Fuse Sicherung aus. Aber im schlechtesten Fall wird der Raspberry Pi bei einem Kurzschluss zerstört. Dieser 1kΩ Widerstand ist im geschlossenem Zustand des Tasters in Reihe mit dem 10kΩ Widerstand. Das ergibt einen gesamten Widerstand von 11kΩ zwischen 3,3V und Masse.

$$I = \frac{3{,}3V}{11k\Omega} = \frac{3{,}3V}{11000\Omega} = 0{,}0003A = 0{,}3mA = 300\mu A$$

Es fließt also gerade mal ein Kurzschlussstrom von 300µA und das ist so gering, dass nichts passiert und trotzdem unsere Eingänge ein sauberes Logisches Signal bekommen.

Der Raspberry Pi wertet die Eingangssignale nur Logisch aus, das bedeutet, ab einer Spannung von ca. 2V wird der Eingang als High (Logisch 1) erkannt. Spannungen unter 2V werden als Low (Logisch 0) ausgewertet. Dies ist wichtig für soeben erklärtes Schweben eines Pins.

Bauen wir unsere Schaltung auf dem Steckboard auf:



```
touch ampel2.py
```

```
nano ampel2.py
```

Bei diesem Projekt verwenden wir dieses Mal nicht die direkte Portzuweisung, sondern arbeiten mit Symbolvariablen. Das bedeutet, wir schreiben nicht GPIO 1 sondern belegen eine Variable mit der GPIO Nummer. Dies vereinfacht später bei größeren Programmen die Änderung der Pinbelegung. Bei Symbolvariablen muss ich nur einmal den Pin ändern, z.B. kommt in der folgenden Schaltung ROT insgesamt 5 x vor. Diese stellen müssten bei Pinänderung alle gefunden und geändert werden.

```
#!/usr/bin/python
from time import sleep
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)                    # Warnungen ausschalten
ROT = 4                                    # GPIO Pin Belegung ROT = GPIO4
GELB = 17                                  # GPIO Pin Belegung GELB = GPIO17
GRUEN = 27                                 # GPIO Pin Belegung GRUEN = GPIO27
TASTER = 25                                # GPIO Pin Belegung TASTER = GPIO25
GPIO.setup(ROT,GPIO.OUT)
GPIO.setup(GELB,GPIO.OUT)
GPIO.setup(GRUEN,GPIO.OUT)
GPIO.setup(TASTER,GPIO.IN)

def ampelschaltung():                      # Funktion ampelschaltung
        GPIO.output(ROT,GPIO.HIGH)
        GPIO.output(GELB,GPIO.HIGH)
        GPIO.output(GRUEN,GPIO.LOW)
        sleep(2)
        GPIO.output(ROT, GPIO.LOW)
        GPIO.output(GELB, GPIO.LOW)
        GPIO.output(GRUEN,GPIO.HIGH)
        sleep(10)
        GPIO.output(GELB, GPIO.HIGH)
        GPIO.output(GRUEN,GPIO.LOW)
        sleep(3)
        GPIO.output(ROT,GPIO.HIGH)
        GPIO.output(GELB,GPIO.LOW)

while True:                                # Schleife
        GPIO.output(ROT,GPIO.HIGH)
        GPIO.output(GELB,GPIO.LOW)
        GPIO.output(GRUEN,GPIO.LOW)
        if GPIO.input(TASTER) == 1:
                ampelschaltung()
```

Schleifen:

Damit man immer wiederkehrende Ereignisse in einer Programmierung abbilden kann, gibt es verschiedene Möglichkeiten. Besonders häufig werden Schleifen verwendet. Wir verwendeten bei uns im Programm bis jetzt immer die While-Schleife. Mit der While Bedingung True, erzeugen wir eine Dauerschleife, denn das While wird nie True werden können, weil wir True nicht beeinflussen können.

Funktionen:

Mit Funktionen kann man seinen Programmcode übersichtlicher gestalten und einzelne wiederkehrende Programmteile auslagern um weniger Speicherplatz zu verbrauchen. Funktionen können in einem Programm beliebig oft aufgerufen werden. Funktionen beginnen mit einem def und einem Funktionsnamen. Die beiden Klammern nach dem Namen sind in unserem Beispiel leer, aber hier könnte man der Funktion Werte übergeben, die in der Funktion verarbeitet werden können.

Bedingungen, if:

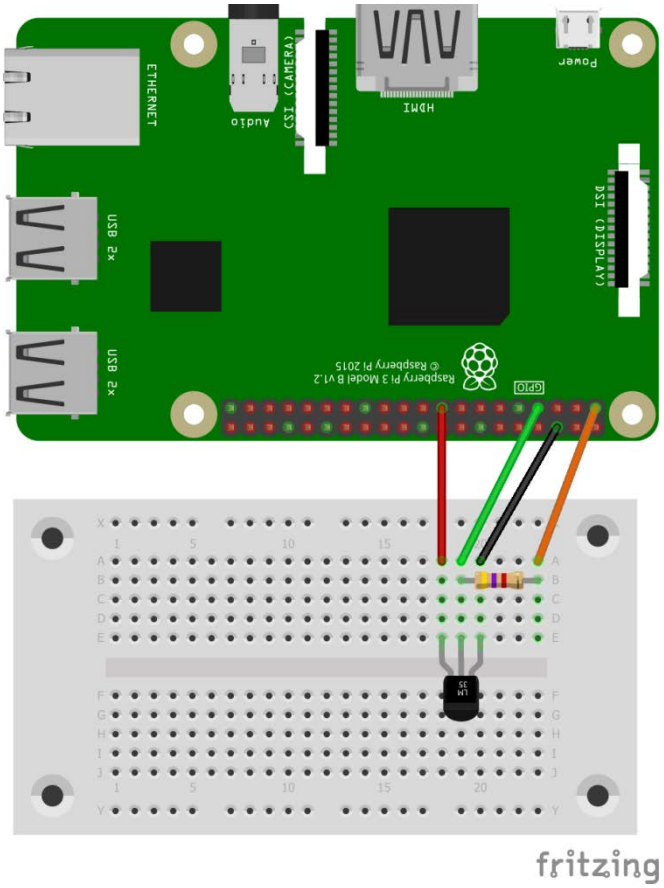
Mit der sogenannten if-then-else Bedingung, kann ich Verzweigungen in einem Programmcode programmieren. Bei uns soll die Ampel erst zum Laufen beginnen, wenn der Taster gedrückt wurde. WENN der TASTER gedrückt, DANN Ampel starten, anSONSTen nichts machen.

Weil es verschieden Programmiermöglichkeiten gibt, wollen wir uns hier die Ampelschaltung auch in einer anderen Programmiersprache, C++, ansehen. C++ ist kein einfaches Skript mehr, sondern ein komplett eigenständiges Programm. Der Programmaufbau ist ähnlich wie PHP, da beide Programmiersprachen von C abstammen.

2.4 Temperatursensor

Nachdem wir nun ein paar LEDs zum Leuchten gebracht haben, wollen wir nun die Temperatur auslesen.

Dazu verwenden wir den Temperatursensor DALLAS DS18B20. Dieser wird mit 1-Wire angesteuert. Das 1-Wire Protokoll wird am Raspberry nur am GPIO4 unterstützt. Deswegen muss der 1-Wire-Anschluss des Temperatursensors (mittlere Anschluss) hier angeschlossen werden.



Bauen wir einmal die Schaltung auf.

Wenn alles entsprechend verkabelt ist, können wir das 1-Wire Protokoll damit aktivieren:

```
sudo modprobe w1-gpio
```

```
sudo modprobe w1-therm
```

Ob es geklappt hat, können wir herausfinden, indem wir folgendes eingeben:

```
lsmod
```

Die Module und müssten nun aufgelistet sein, falls nicht wird ein anderer GPIO benutzt oder es trat ein Fehler beim Aktivieren auf.

Damit nicht bei jedem Start die Module geladen werden, tragen wir sie in die Datei /etc/modules ein:

```
sudo nano /etc/modules
```

und fügen als letztes die folgenden zwei Zeilen ein:

```
w1_gpio
```

```
w1_therm
```

Außerdem muss man noch in der boot.txt den GPIO4 für 1-Wire reservieren:

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

und folgende Zeile am Ende ergänzen:

```
dtoverlay=w1-gpio,gpiopin=4
```

Nach einem Neustart, wechseln wir in das W1-Bus Verzeichnis und lassen uns alle Sensoren anzeigen.

```
cd /sys/bus/w1/devices/
```

```
ls
```

Es sollte in der Ausgabe in etwa so ein Eintrag enthalten sein: **28-02162eb1fbee**

Diese Nummer muss man sich merken, das ist die ID des Temperatur Sensor.

Als nächstes lesen wir den Sensor aus, wir geben diesen Befehl in die Konsole:

```
cat /sys/bus/w1/devices/28-02162eb1fbee/w1_slave
```

Als Antwort bekommen wir folgendes:

```
b8 01 4b 46 7f ff 0c 10 b1 : crc=b1 YES
b8 01 4b 46 7f ff 0c 10 b1 t=27500
```

in der 2. Zeile t=27500 ist unser Temperaturwert in Milligrad. Dieser Wert muss nur durch 1000 geteilt werden, 27000 : 1000 = 27,5. Die gemessene Temperatur ist 27,5°C.

Da diese Anzeige und Umrechnung umständlich ist, schreiben wir ein kleines Skript um nun noch den Befehl „temperatur“ eingeben müssen und diesen dann sofort in Klartext angezeigt bekommen.

```
sudo nano /usr/bin/temperatur
```

Mit dem Inhalt:

```
#! /bin/bash
# Temperatur auslesen
tempread=`cat /sys/bus/w1/devices/28-02162eb1fbee/w1_slave`
# Wert Formatieren
temp=`echo "scale=2; "\`echo ${tempread##*=}\`" / 1000" | bc`
#Ausgabe
echo "Die gemessene Temperatur beträgt" $temp "°C"
```

Nun noch entsprechende Rechte geben

```
sudo chmod +x /usr/bin/temperatur
```

und wenn man nun in die Konsole

```
temperatur
```

eingibt, so erscheint die aktuell gemessene Temperatur. Falls eine Fehlermeldung kommt, so musst du wohl bc nachinstallieren.

```
sudo apt-get install bc
```

Wenn alles funktioniert dann sollte bei “temperatur“ folgendes Ergebnis kommen:

```
Die gemessene Temperatur beträgt 27.5 °C
```

Im nächsten Schritt geben wir die Temperatur noch in einer Webseite aus.

Dazu erstellen wir eine neue PHP Datei:

```
touch temperatur.php
```

```
nano temperatur.php
```

Und geben folgenden Inhalt ein:

```
<?php
$handle = fopen("/sys/bus/w1/devices/w1_bus_master1/w1_master_slaves", "r");
if ($handle) {
    while (($sensors = fgets($handle)) !== false) {
            $sensor = "/sys/bus/w1/devices/".trim($sensors)."/w1_slave";
            $sensorhandle = fopen($sensor, "r");
              if ($sensorhandle) {
                  $thermometerReading = fread($sensorhandle, filesize($sensor));
                  fclose($sensorhandle);
                  // Auslesen der Temperatur nach dem t= in der 2. Zeile
                  preg_match("/t=(.+)/", preg_split("/\n/", $thermometerReading)[1],
$matches);
                  $celsius = $matches[1] / 1000; //umrechnen
                  $fahrenheit = $celsius*9/5+32;
                  print "$sensors = <b>$celsius &deg;C</b> / $fahrenheit &deg;F<br>";
                  $sensors++;
              } else {
                 print "Sensor kann nicht gelesen werden";
              }
    }
    fclose($handle);
} else {
    print "Kein Sensor gefunden";
}
?>
```

Die Webseite zeigt nun beim Aufrufen:

28-02162eb1fbee = **27.437 °C** / 83.1866 °F

Und jetzt noch als Python Programm: temperatur.py

```
#!/usr/bin/python
# coding=utf-8
#import os, sys, time
from time import sleep
def aktuelleTemperatur():
    file = open('/sys/bus/w1/devices/28-02162eb1fbee/w1_slave')
    filecontent = file.read()                              #1-wire Slave Datei lesen
    file.close()
    stringvalue = filecontent.split("\n")[1].split(" ")[9] #Temperaturwerte
    temperature = float(stringvalue[2:]) / 1000            #auslesen und konvertieren
    rueckgabewert = '%6.3f' % temperature                  #Temperatur ausgeben
    return(rueckgabewert)
```

```
while True:
        temperatur = aktuelleTemperatur()
        print "Aktuelle Temperatur : ", temperatur, "°C"
        sleep(2)
```

Du hast es geschafft, du kannst nun deine Projekte mit dem Raspberry Pi verwirklichen!

Ab jetzt heißt es Experimentieren.

Und für mehr Hardware sorgt natürlich dein Online-Shop auf:

https://az-delivery.de

Viel Spaß!
Impressum

https://az-delivery.de/pages/about-us