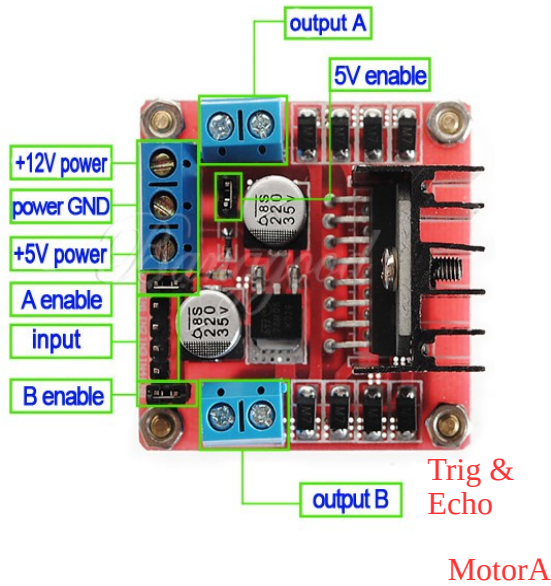


# Workshop ROBOTIK

Knoppixtage September 2018

## Verkabelung und Stromversorgung



Raspberry Pi2 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 (PC ID EEPROM)	(PC 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. 1  
26/01/2014  
<http://www.element14.com>

Line-Sensoren

MotorB

# Verbindung zum Raspberry PI

## 1 IP Adresse des Robo

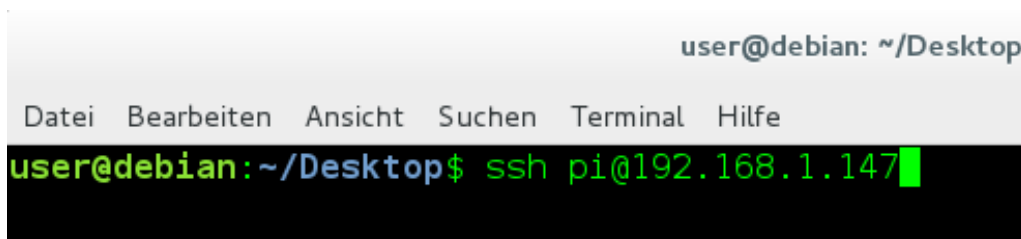
Der PI verbindet sich mit dem WLAN-Router automatisch.

### Active Clients

Hostname	IP-Adresse	MAC-Adresse	Conn. Count	Ratio [4096]
ubuntu-gnome	192.168.2.142	00:22:3F:EB:DC:F2	61	1%
raspberrypi	192.168.2.94	00:E0:4C:0E:3F:E0	0	0%

## 2 Terminalfenster und ssh

Auf dem eigenen Laptop nun ein Terminalfenster öffnen und eine ssh Verbindung aufbauen:



```
user@debian: ~/Desktop
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
user@debian:~/Desktop$ ssh pi@192.168.1.147
```

**ssh pi@192.168.2.xx** → Das nun abgefragte Passwort ist einfach **pi**.

## 3 Verzeichniswechsel

```
pi@misof01 ~ $ cd workshop
```

Das Verzeichnis anzeigen mit **ls** :

```
pi@misof01 ~/workshop $ ls
```

# Der ROBO wird nun mit Python gesteuert

## 1 Grundstruktur

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import sys
import kmmotor

#main

HIER KOMMEN DANN DEINE BEFEHLE FUE DEN RASPBERRY PI

GPIO.cleanup()
```

Alle Programme benötigen nun folgende Eingaben:

## 2 Vorwärts

Robo soll eine **bestimmte Zeit** (hier 2 Sekunden) **vorwärts** fahren und dann anhalten:

```
#main
kmmotor.forward(100,0)
time.sleep(2)
kmmotor.stop()
```

Falls der Robo nicht ganz gerade fährt, kann man versuchen einen positiven oder negativen Trimmfaktor einzugeben.

z. B.:

```
#main
kmmotor.forward(100, -10)
time.sleep(2)
kmmotor.stop()
```

## 3 Rückwärts

Robo soll eine **bestimmte Zeit** (hier 2 Sekunden) **rückwärts** fahren und dann anhalten:

```
kmmotor.back(100,0)
time.sleep(2)
kmmotor.stop()
```

Auch hier kann man auch einen Trimmfaktor verwenden.

```
kmmotor.back(100,15)
time.sleep(2)
kmmotor.stop()
```

#### 4 Linkskurve

Robo soll eine **bestimmte Zeit** (hier 0,5 Sekunden) nach **links drehen** und dann anhalten:

```
kmmotor.left(100)
time.sleep(0.5)
kmmotor.stop()
```

Hier gibt es keine Trimmfunktion. Das **Komma** wird als **PUNKT** eingegeben!!!

#### 5 Rechtskurve

Robo soll eine **bestimmte Zeit** (hier 0,5 Sekunden) nach **rechts drehen** und dann anhalten:

```
kmmotor.right(100)
time.sleep(0.5)
kmmotor.stop()
```

# Der Robo wird mit den Cursortasten gesteuert

## 1 Grundstruktur

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import sys
import kmmotor
import curses

stdscr = curses.initscr()
stdscr.keypad(1)
curses.noecho
```

## 2 Robo wird mit den Pfeiltasten gesteuert

Dazu benötigst du wieder Entscheidungen was der Robo tun sollen wenn eine bestimmte Pfeiltaste gedrückt wird:

Taste	Aktion
curses.KEY_UP	Vorwärtsfahren
curses.KEY_DOWN	Rückwärtsfahren
curses.KEY_RIGHT	Nach rechts drehen
curses.KEY_LEFT	Nach links drehen
q	Programm stoppen (QUIT)
Leertaste	stoppen

Damit obige Entscheidungen immer wieder passieren musst du eine Endlosschleife verwenden:

```
key = ''
while key != ord('q'):
    key = stdscr.getch()
    #stdscr.addch(20,25,key)
    if key == ord(' '):
        kmmotor.all_off()
    if key == curses.KEY_UP:
        kmmotor.forward(100,0)
    if key == curses.KEY_DOWN:
        kmmotor.back(100,0)
    if key == curses.KEY_RIGHT:
        kmmotor.right(100)
    if key == curses.KEY_LEFT:
        kmmotor.left(100)

kmmotor.all_off()
stdscr.keypad(0)
curses.echo()
curses.endwin()
```

# Der Robo erkennt eine Linie

## 1 Verkabelung der Sensoren

**!!!!Bevor man neu verkabelt muss der PI abgeschaltet werden!!!!!!!**

Die beiden Linien Sensoren haben jeweils drei Kontakte und werden mit dem Steckboard nach folgender Tabelle verbunden:

SENSOR LINKS			SENSOR RECHTS		
GND	OUT	VCC	GND	OUT	VCC
Minuspol	BOARD 8	3,2V	Minuspol	BOARD 10	3,2V

## 2 Grundstruktur

Es kommt nun zunächst eine Zeile hinzu:

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
import sys
import kmotor
import kmline

#main

HIER KOMMEN DEINE BEFEHLE

GPIO.cleanup()
```

## 3 Testprogramm für die Liniensensoren

Du kannst nun die Liniensensoren abfragen, ob sie die Linie erkennen:

Du brauchst dafür nun erstmals eine sogenannt Endlosschleife. Die **Einrückung** der Zeilen nach der ersten Zeile ist ganz wichtig!!

```
while True:
    wert=kmline.erkennung()
    print(wert)
```

Am Bildschirm wir dir nun einer der 3 Fälle angezeigt:

BOTH_FALSE	TRUE_LEFT	TRUE_RIGHT
Keine Linie sichtbar	Linie unter dem linken Sensor	Linie unter dem rechten Sensor

Diese Testprogramm kannst du mit der Tastenkombination <STRG+C> abbrechen.

## 4 Der ROBO folgt einer Linien

Du schreibst nun ein ganz kleines Programm, sodass der Robo einer Linie nach fährt. Welche Schritte sind notwendig: Der Robo muss, je nachdem wo die Linie ist, **Entscheidungen** treffen:

Sensormeldung	Aktion
BOTH_FALSE	Robo soll gerade aus fahren
TRUE_LEFT	Robo soll etwas nach links schwenken
TRUE_RIGHT	Robo soll etwas nach rechts schwenken

Diese Entscheidungen wiederholt der Robo immer wieder, darum braucht man auch eine **Endlosschleife**. Das nächste Beispiel zeigt dir die notwendige Programmstruktur:

```
while True:
    try:
        wert=kmline.erkennung()
        if wert == "BOTH_FALSE" :
            kmmotor.forward(100,0)
            time.sleep(0.1)
        if wert == "TRUE_LEFT" :
            kmmotor.left(80)
            time.sleep(0.1)
        .
        .
        .
    except KeyboardInterrupt:
        kmmotor.stop()
        break
```

## ANHANG:

### Funktionen der librarys

1. lib/kmmotor.py:

```
# Motorfunktionen
# forward(tempo,trim)...tenpo von 0-100, trim -50 bis 50
# back(tempo,trim)...tenpo von 0-100, trim -50 bis 50
# right(tempo)...tenpo von 0-100
# left(tempo)...tenpo von 0-100
# backright(tempo)...tenpo von 0-100
# backleft(tempo)...tenpo von 0-100
# stop()
#trim.....geradeausfahrt einstellen
```

2. kmline.py:

```
#####
# Funktionsaufruf
# erkennung() liefert TRUE_RIGHT, TRUE LEFT,BOTH-TRUE.BOTH_FALSE, NOINFO
#####
```

3. kmultrasonic.py:

```
#####
#Funktionsaufruf
# distance() liefert Abstand in cm zurueck
#####
```

4. kmposition.py:

```
#####
# Funktionsaufruf
#position(pos) .....pos in Grad -45 bis 45
#####
```