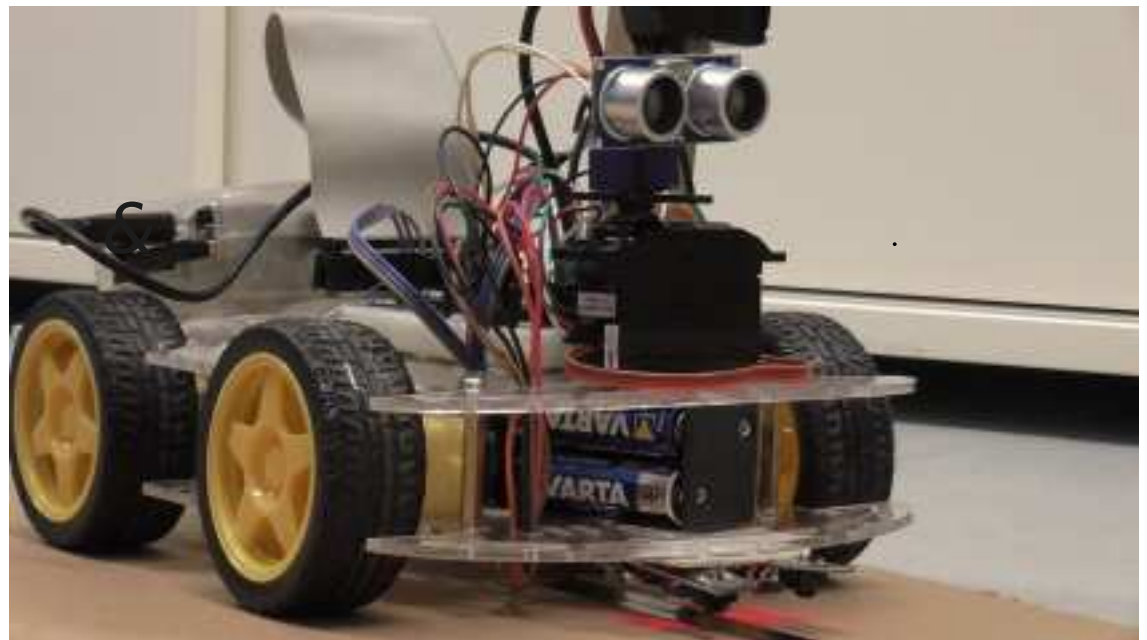


Raspberry PI 2 in der 9. Schulstufe

Mag. Dr. Klaus Misof

Mag. Martin Gruber

BRG Kremszeile

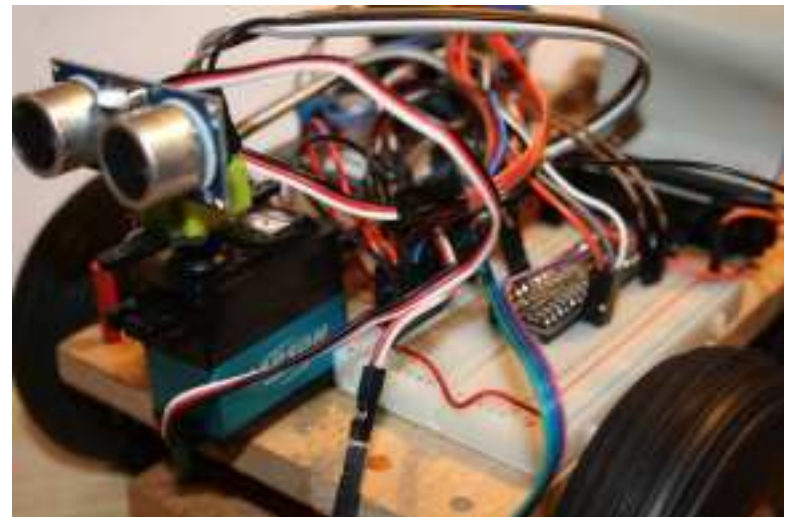


Neues Fach in der 9. Schulstufe AHS

- **Kombination aus Physik, Informatik & Mechanik**
- **Team-Teaching**



- **Pflichtfach Informatik mit Praxis ergänzen**
- **Fortsetzung der Kenntnisse aus Physik, Informatik und Technischen Werken aus der Unterstufe**
- **Förderung des kreativen Gestaltens**
- **Selbständiges Arbeiten an einem konkretem Projekt**



Auswahl der Hardware

- **Raspberry Pi 2 ?**
- **Arduino ?**
- **Chassis ?**



Robots & Sensoren steuern und verstehen



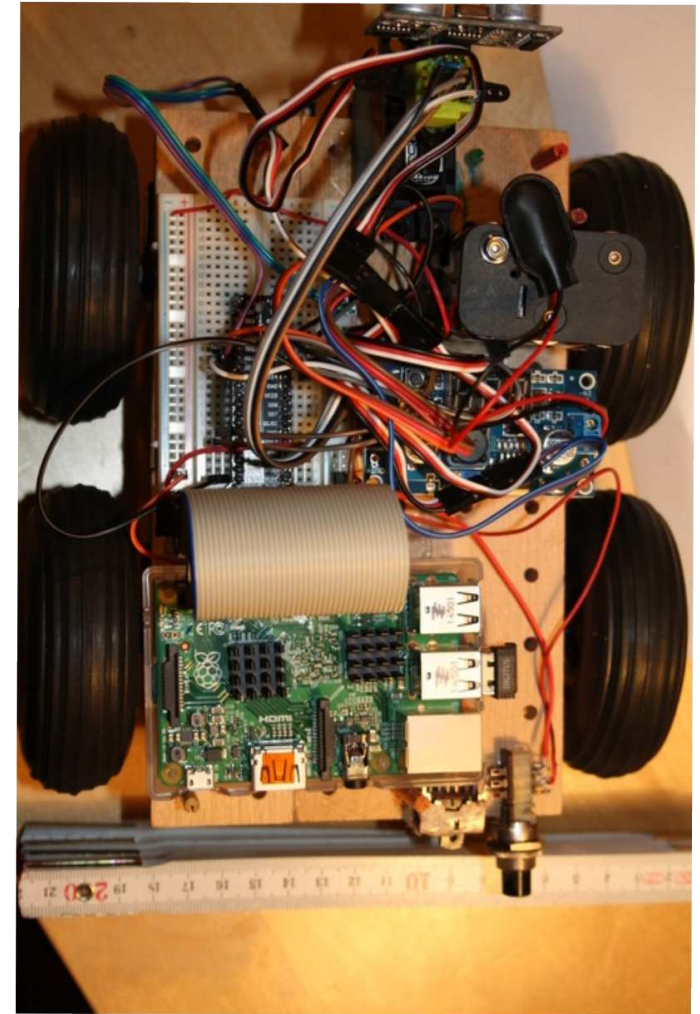
Wie steuert man mit dem PI DC- Motoren oder Servos?

Wie kann man sich mit dem Handy verbinden?

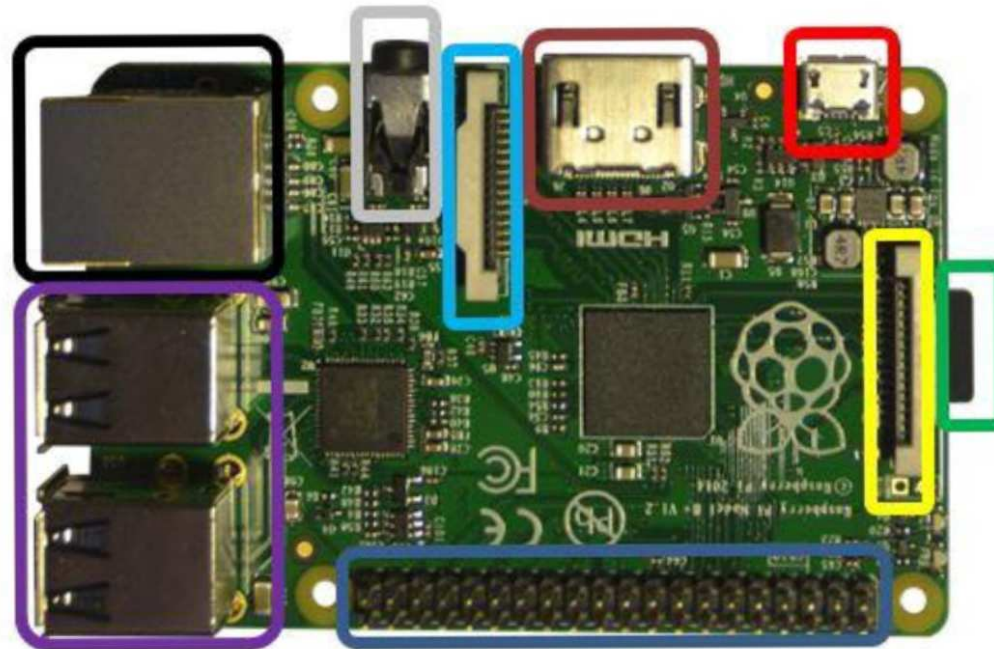
Wie kann der ROBO seine Umwelt wahrnehmen?

Welches Wissen ist über Hardware notwendig?

Welche Sensoren sind leicht zu handhaben?

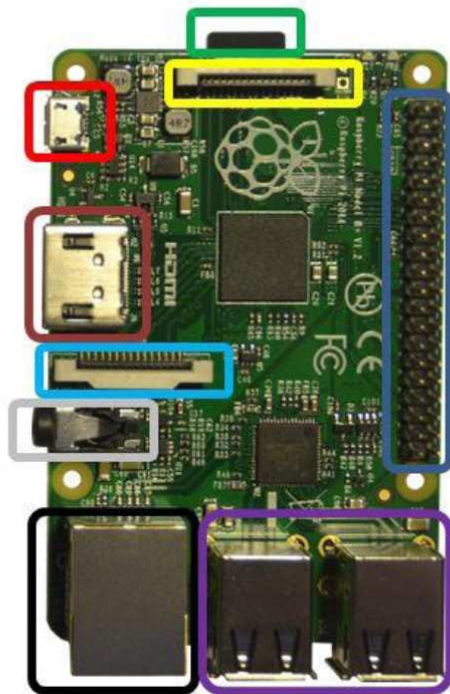


Raspberry Pi 2



- | | | |
|---|--|---|
|  SD-Karte |  HDMI |  Audio/Video |
|  Micro-USB (Power) |  Camera | |
|  Display |  RJ-45 (Netzwerk) | |
|  GPIO |  USB | |

GPIO



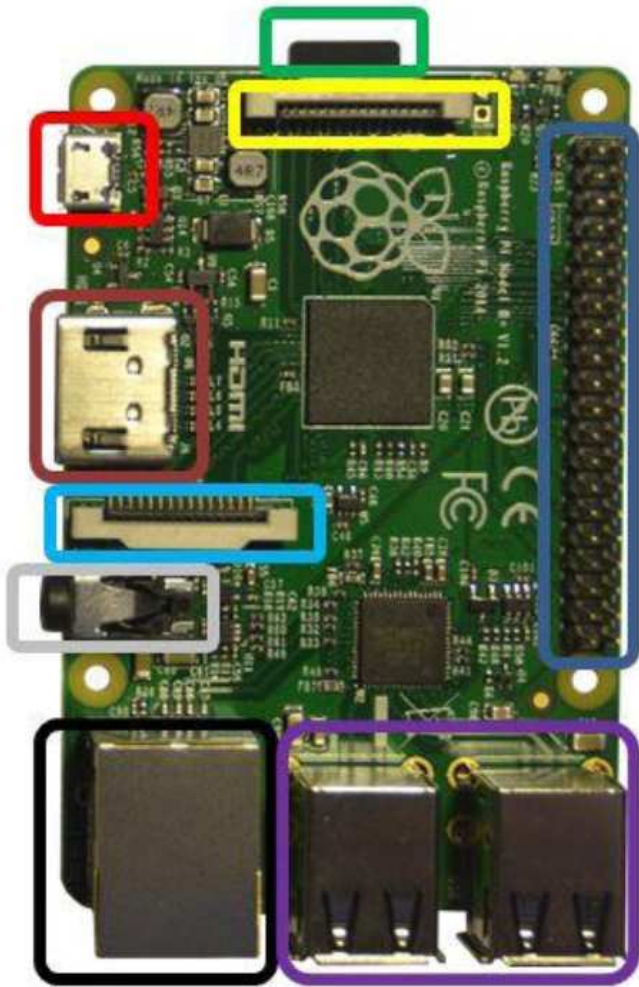
Raspberry Pi2 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 ; I ² C)	DC Power 5v	04
05	GPIO03 (SCL1 ; I ² C)	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 (I ² C ID EEPROM)	(I ² C 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>

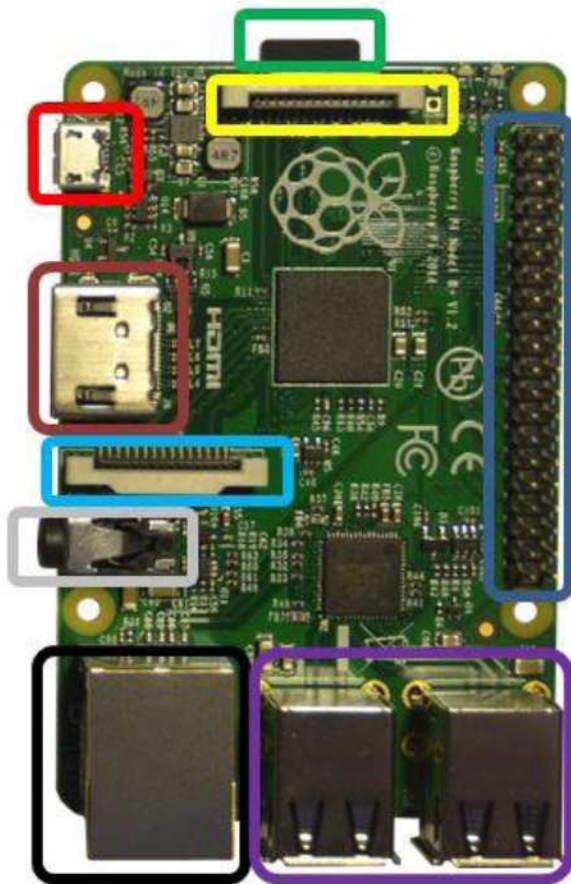
RASPBERRY PI & TERMINAL



```
user@debian: ~/Desktop
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
user@debian:~/Desktop$ ssh 192.168.2.215
```



RASPBERRY PI & TERMINAL

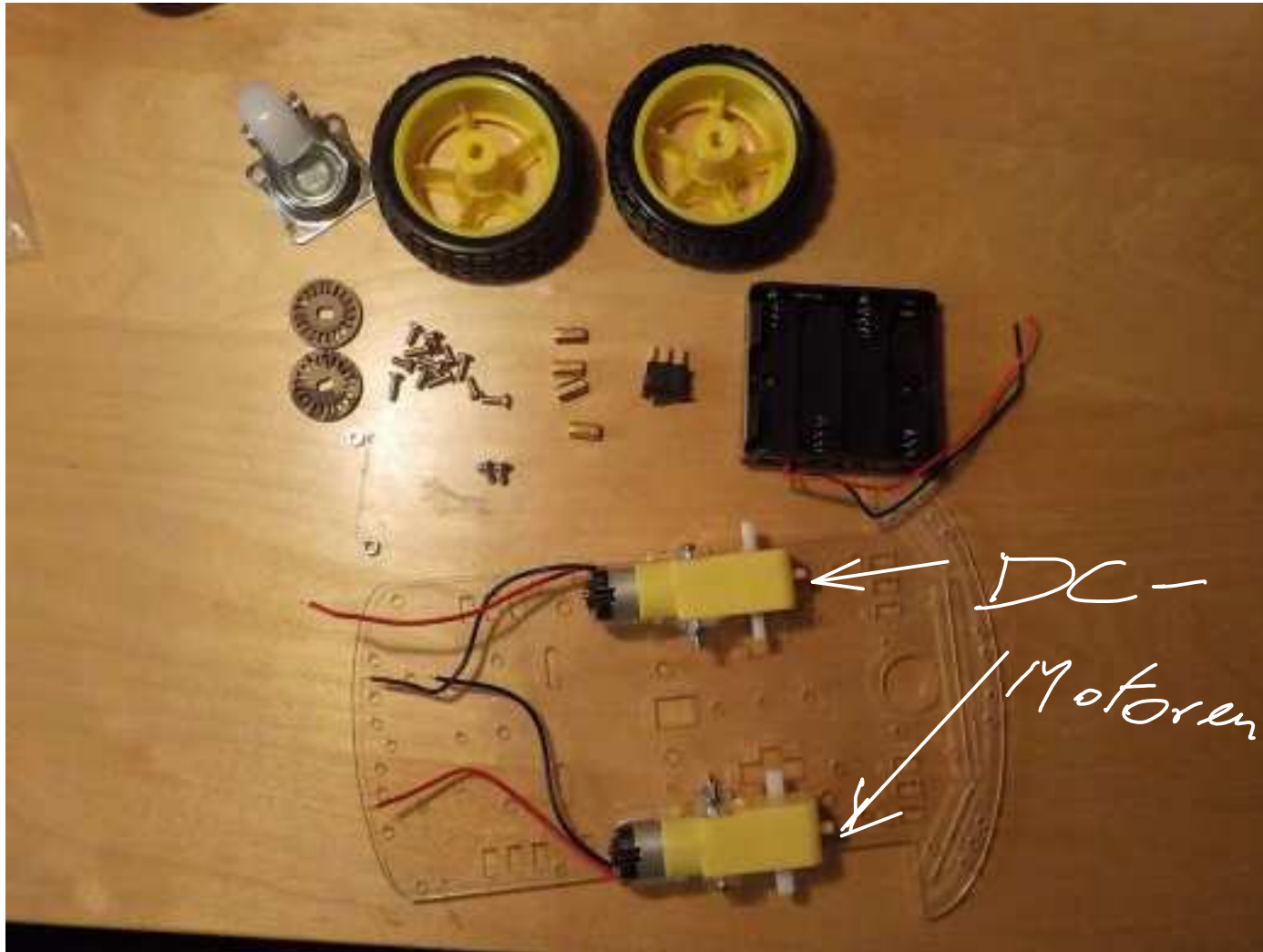


In der Datei :
`/etc/wpa_supplicant/wpa_supplicant.conf` :

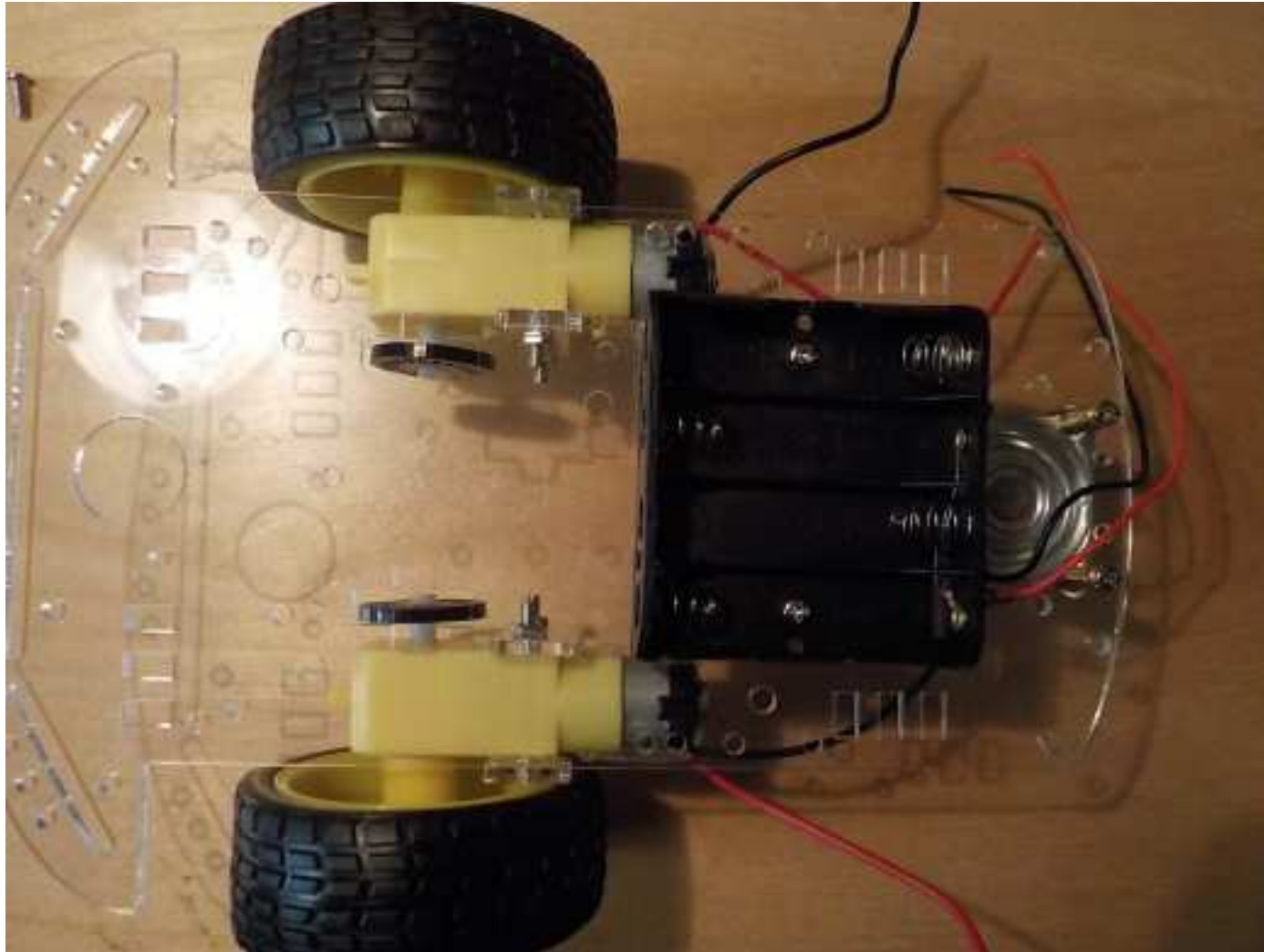
```
ctrl_interface=DIR=/var/run/wpa_supplicant  
group=netdev  
update_config=1
```

```
network={  
    ssid="SCHULE"  
    psk="xxxxxx"  
    key_mgmt=WPA-PSK  
}
```

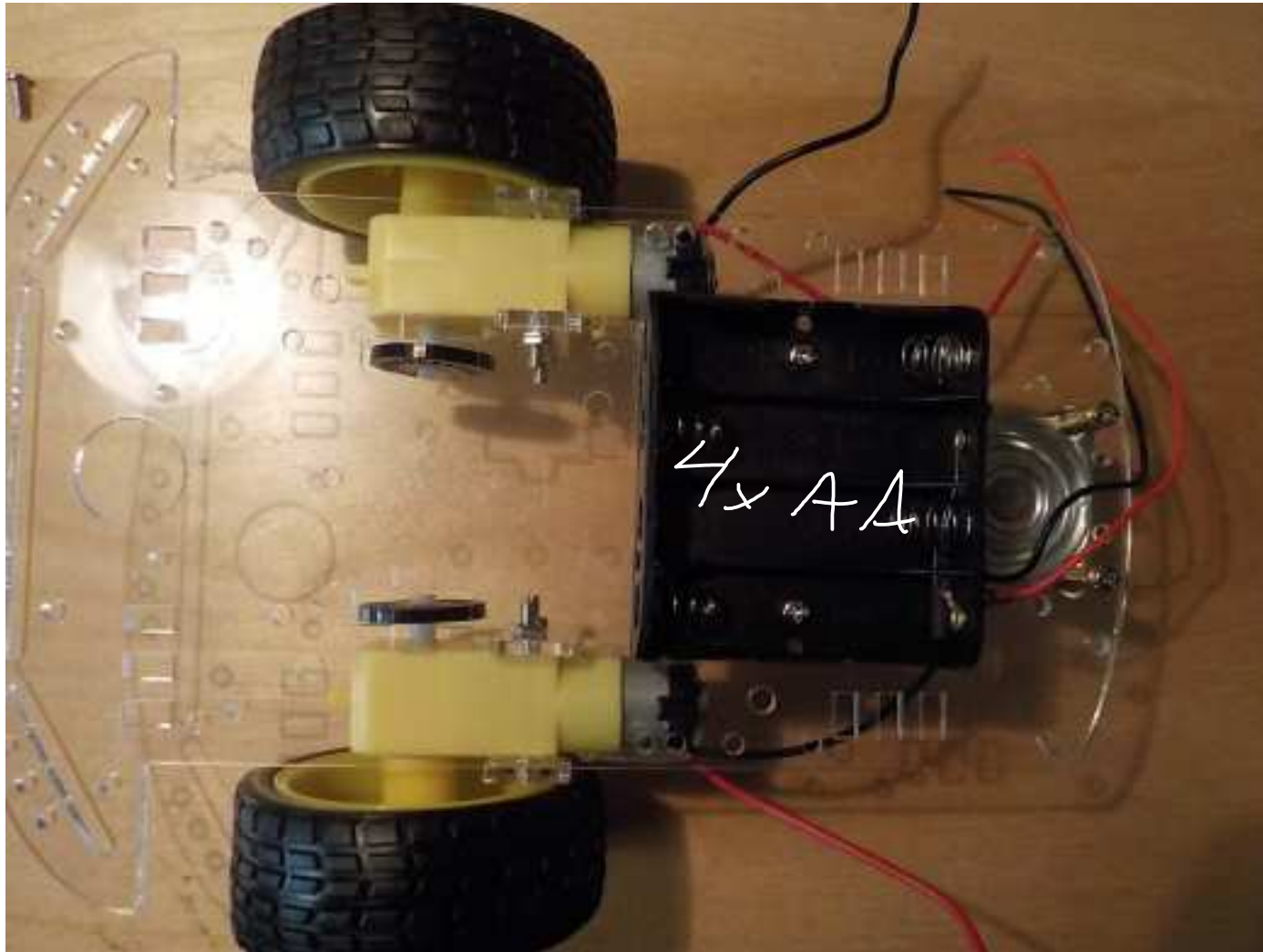
Chassis



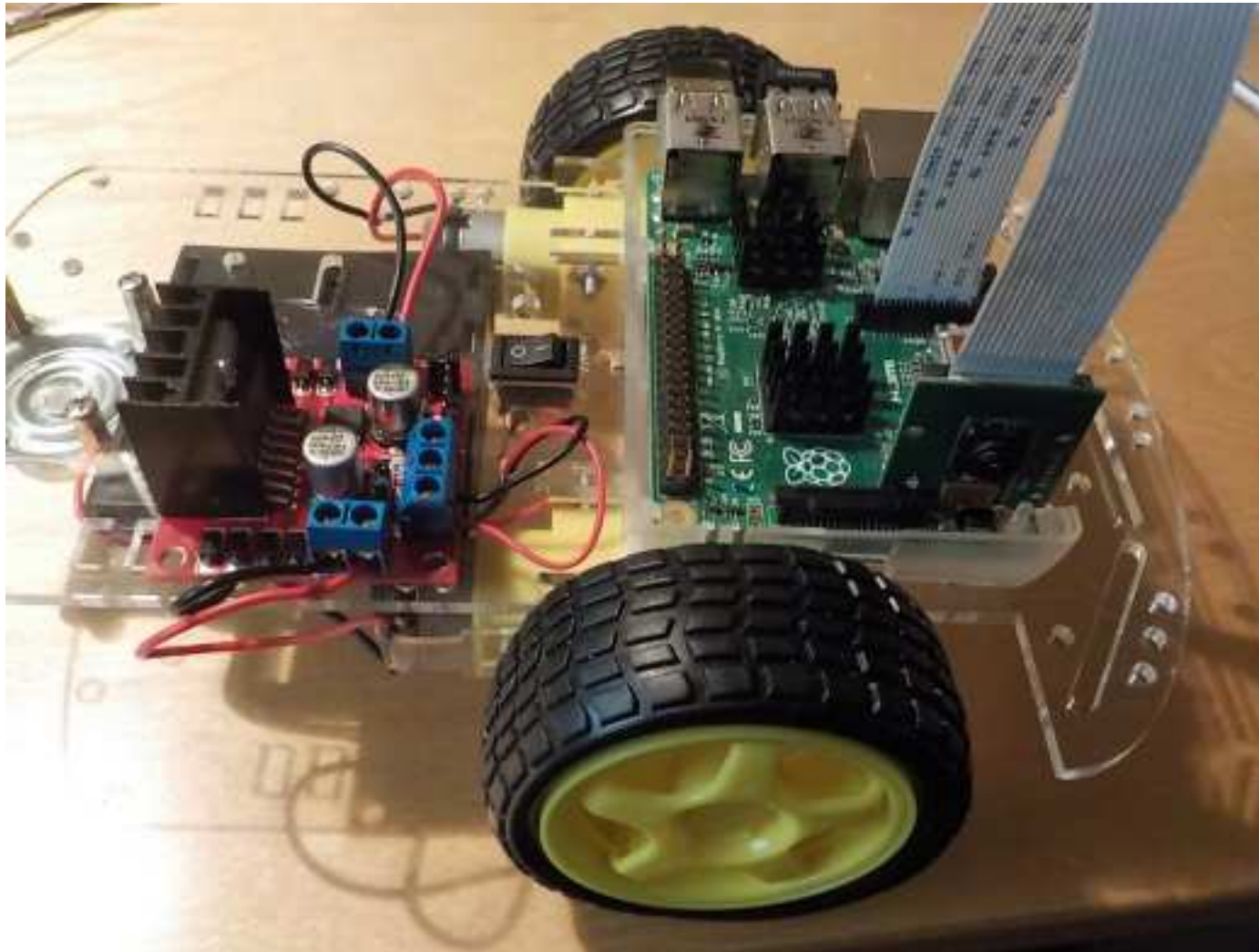
Chassis



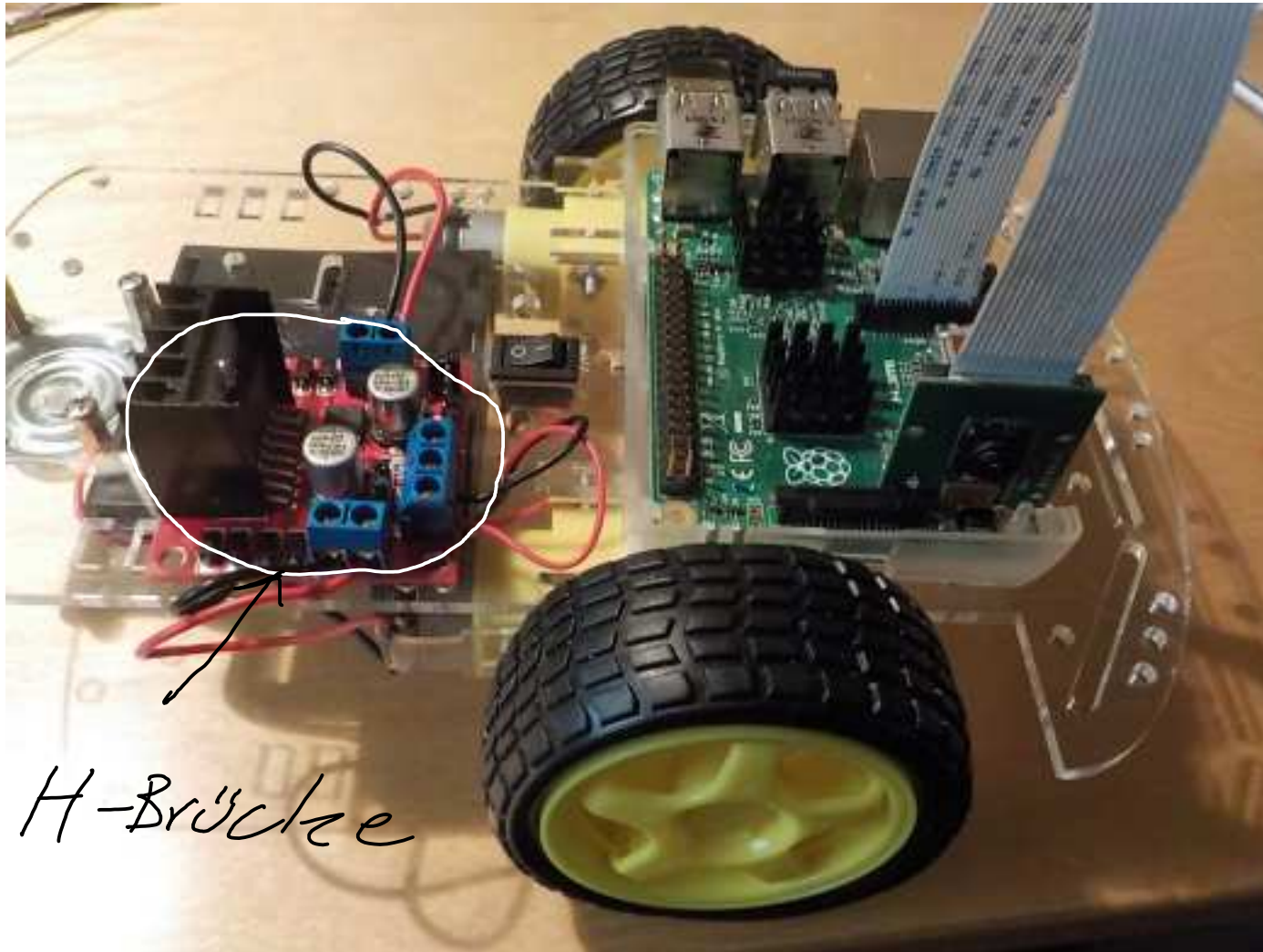
Chassis



Chassis

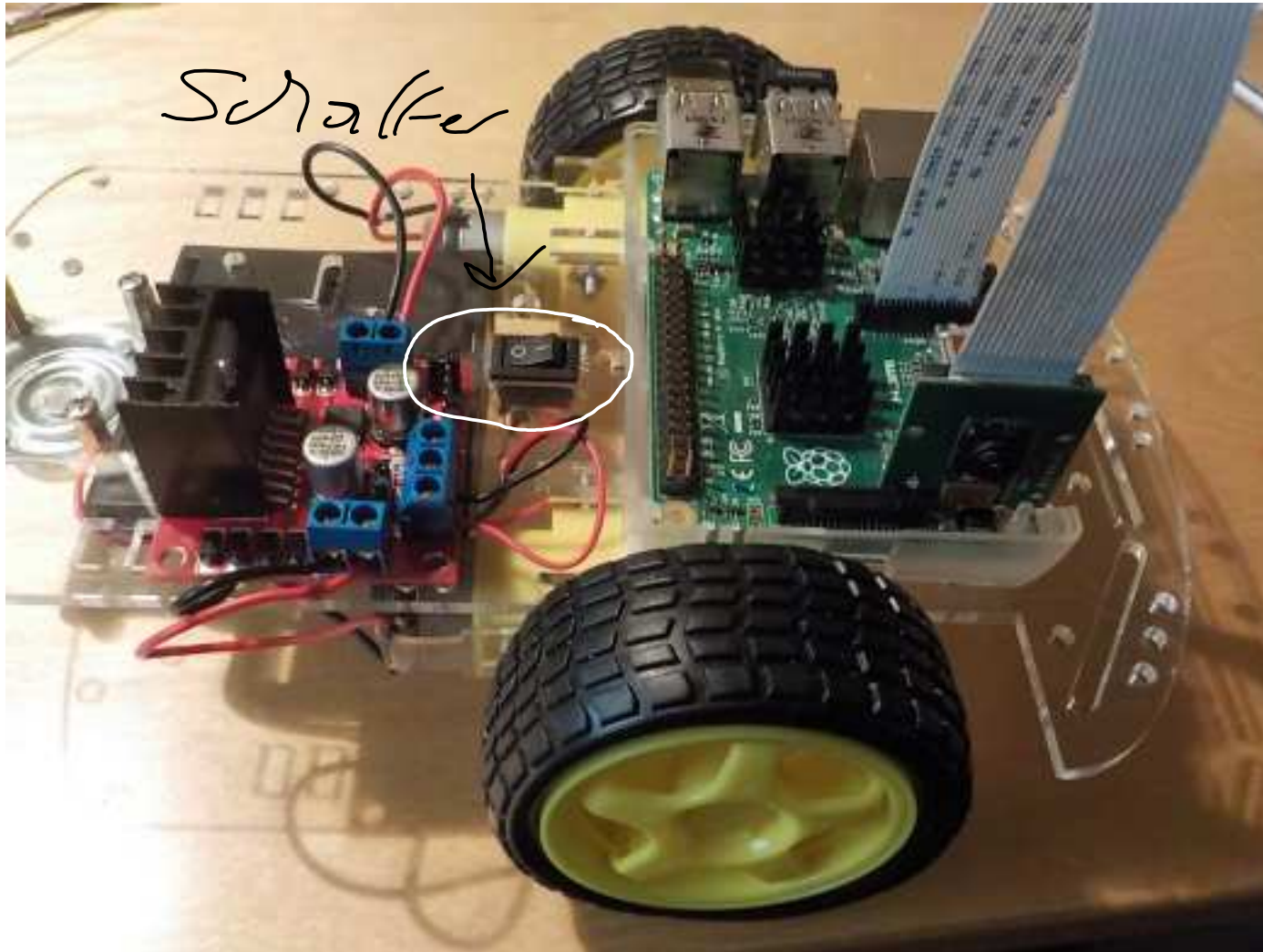


Chassis

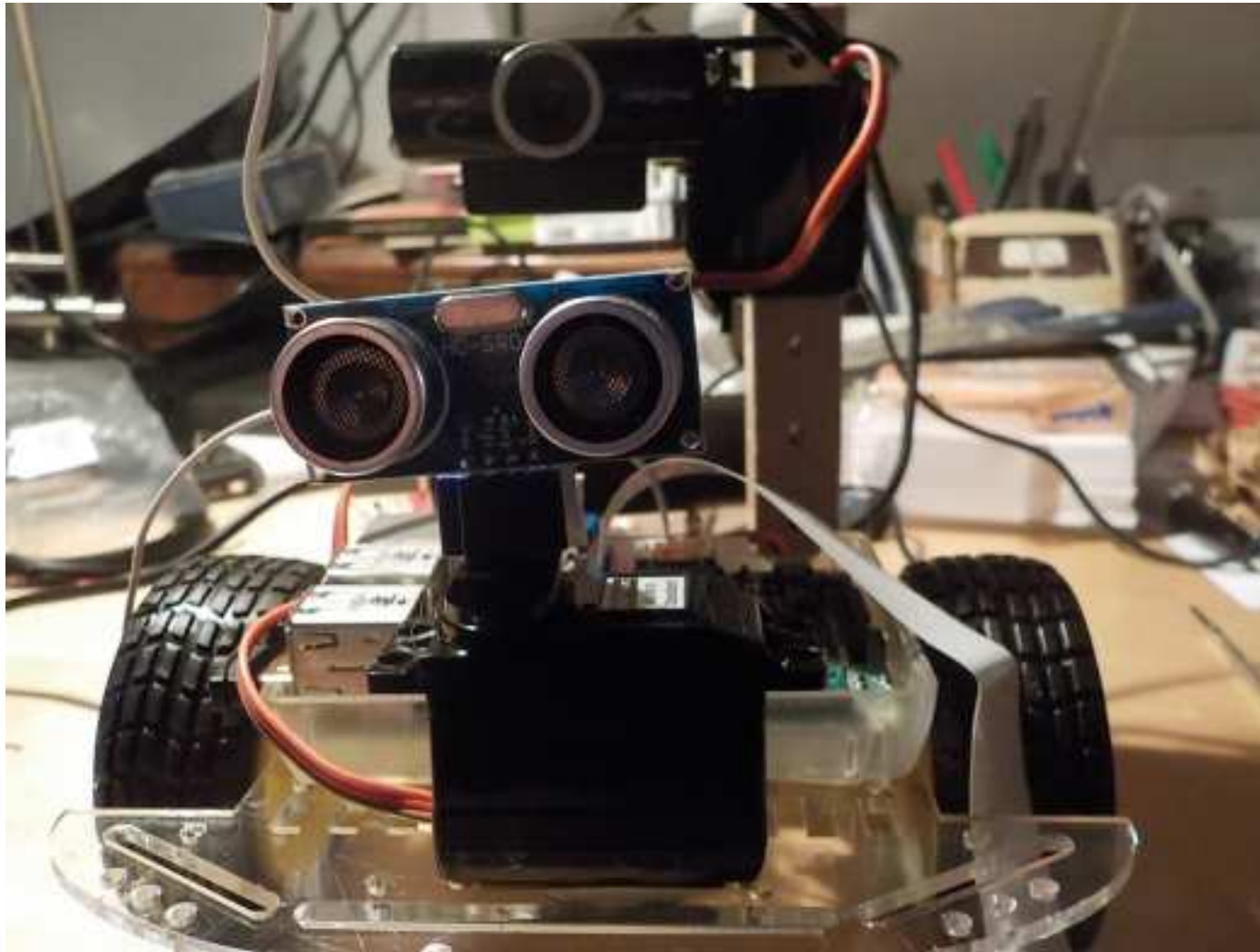


H-Brücke

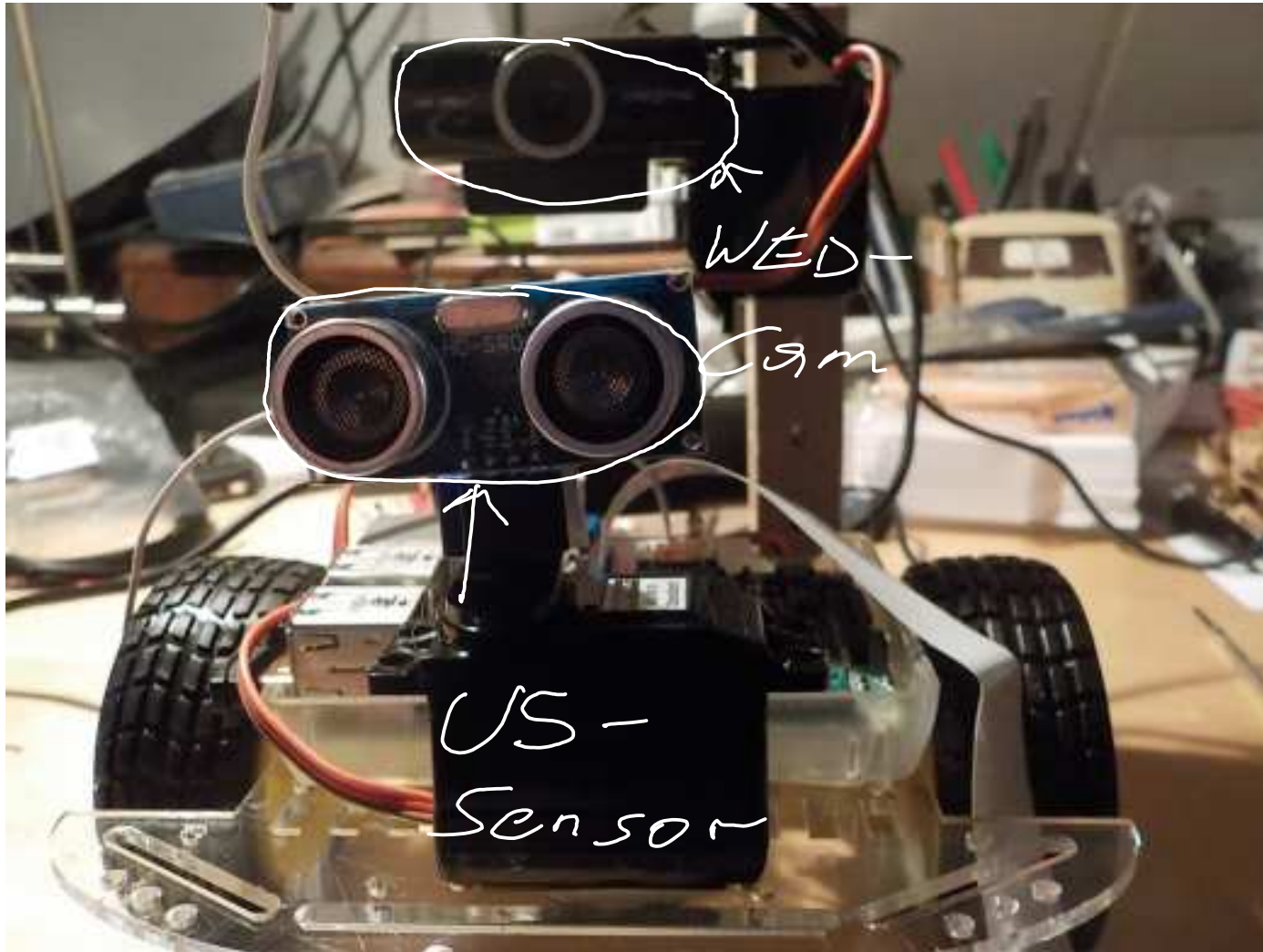
Chassis



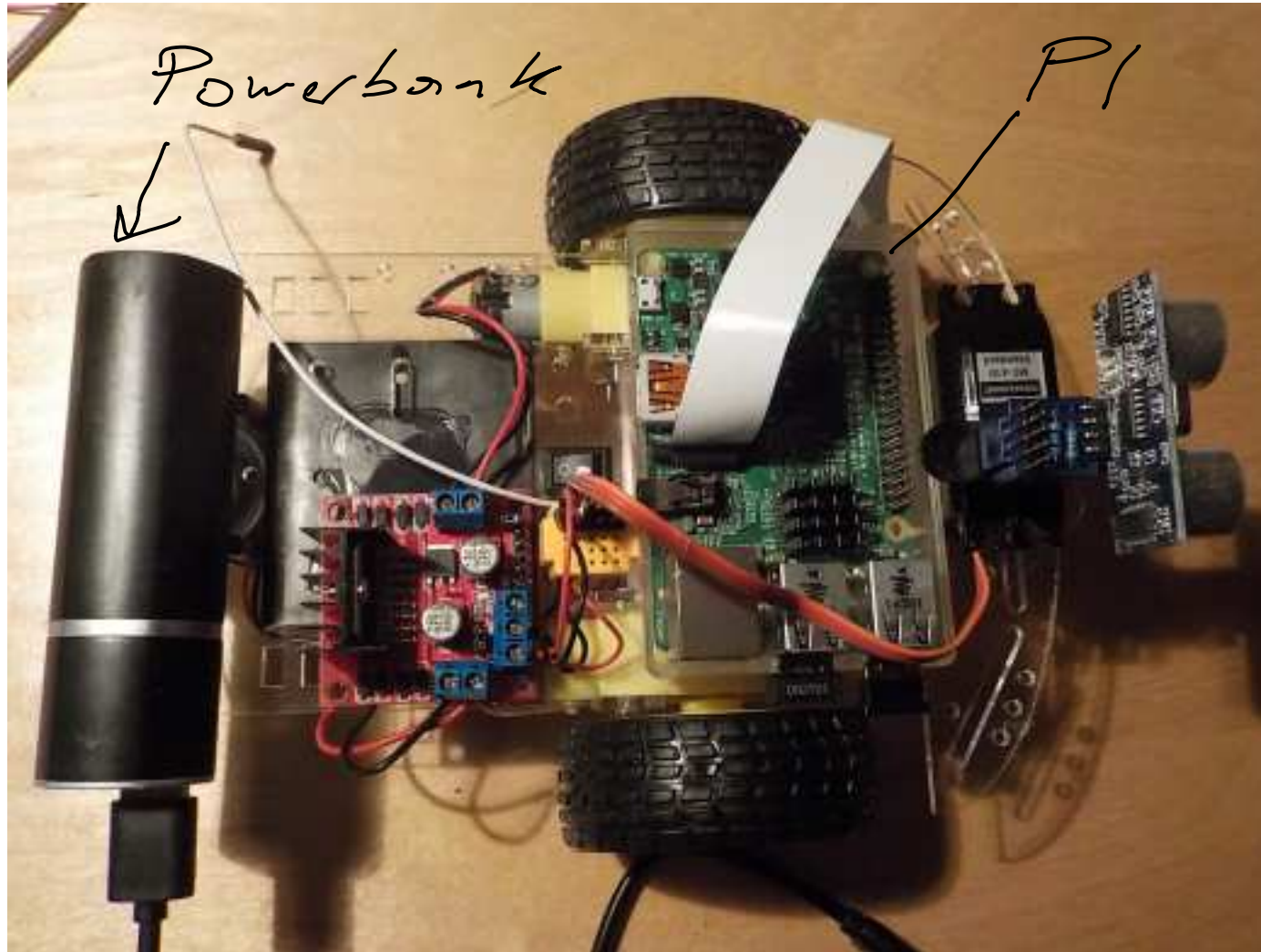
Chassis



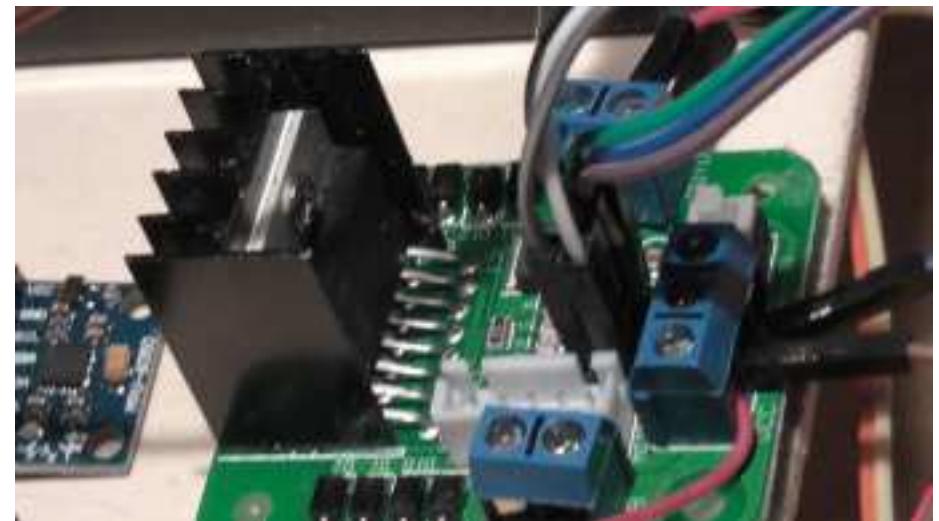
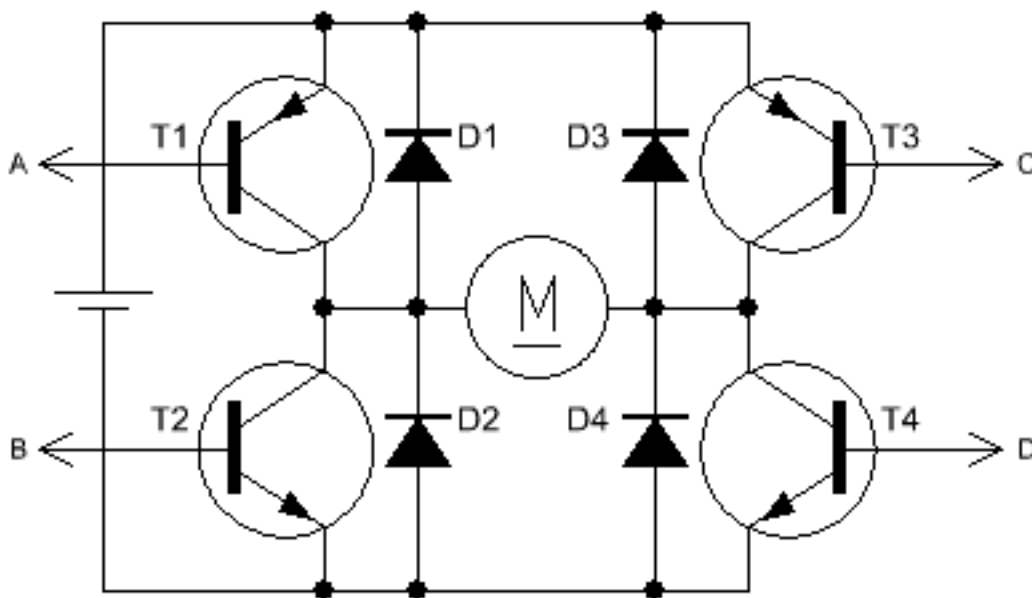
Chassis



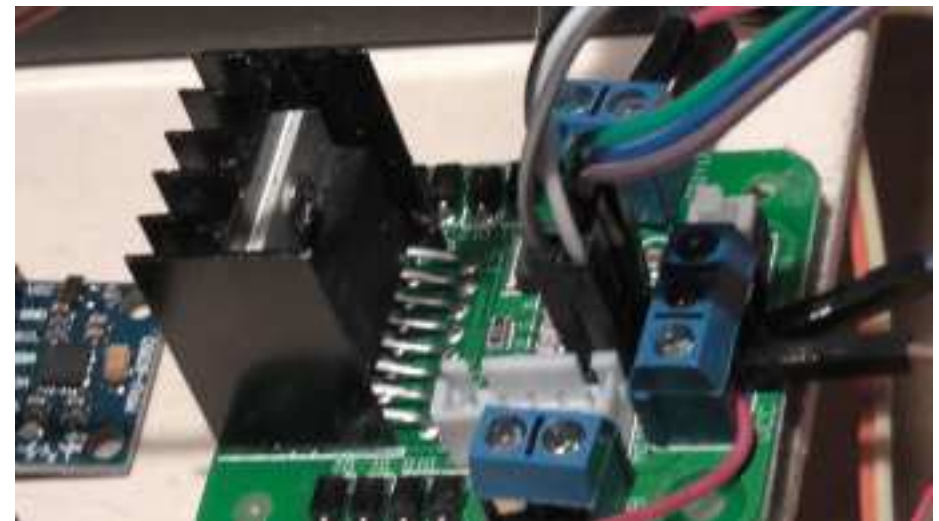
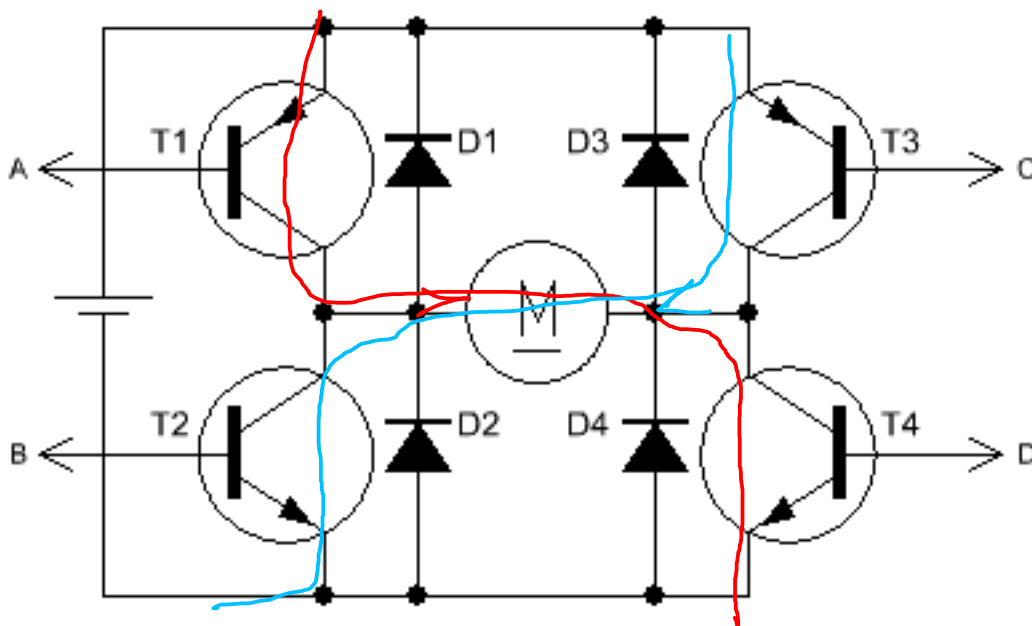
Chassis



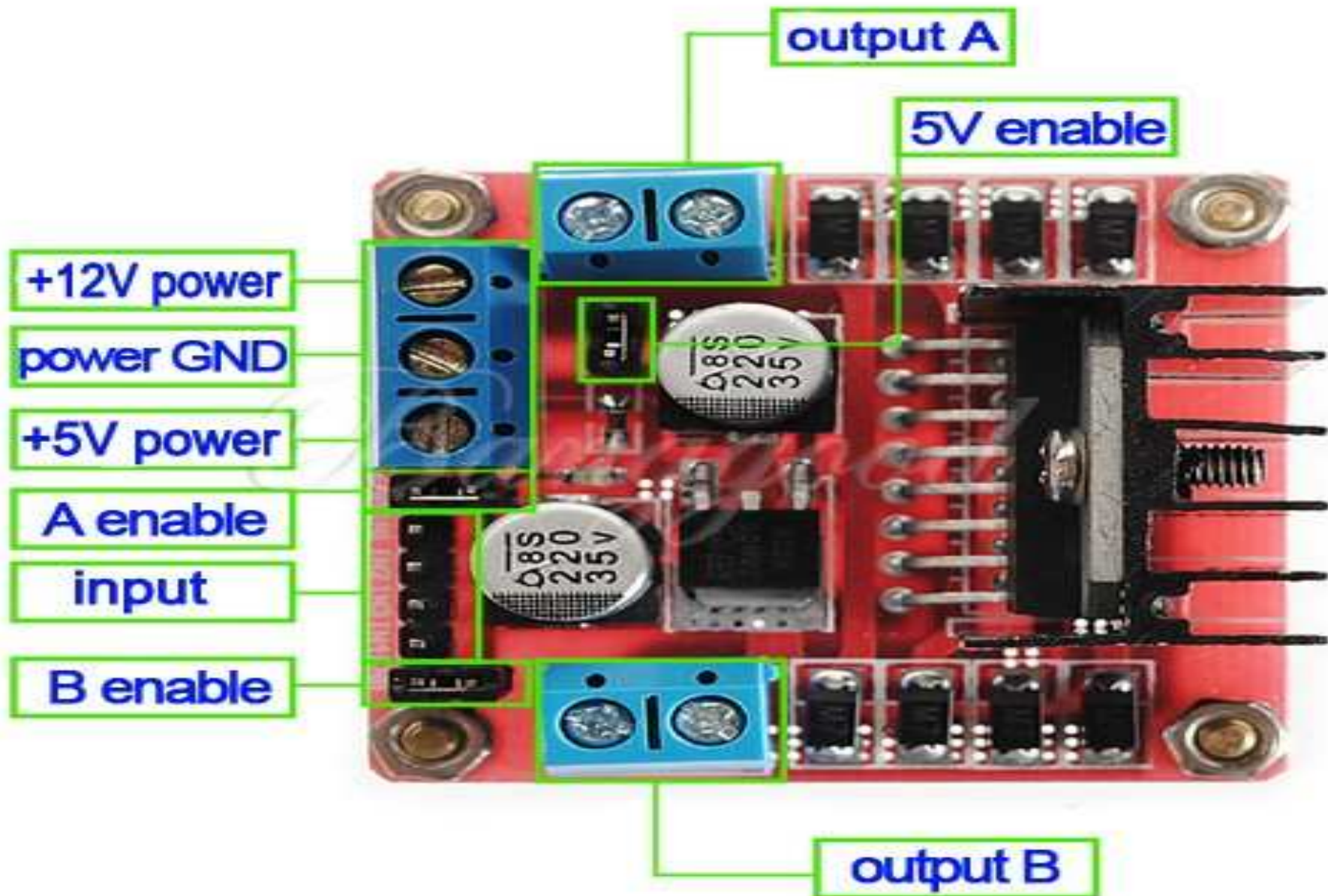
DC-MOTOREN & H-BRÜCKE



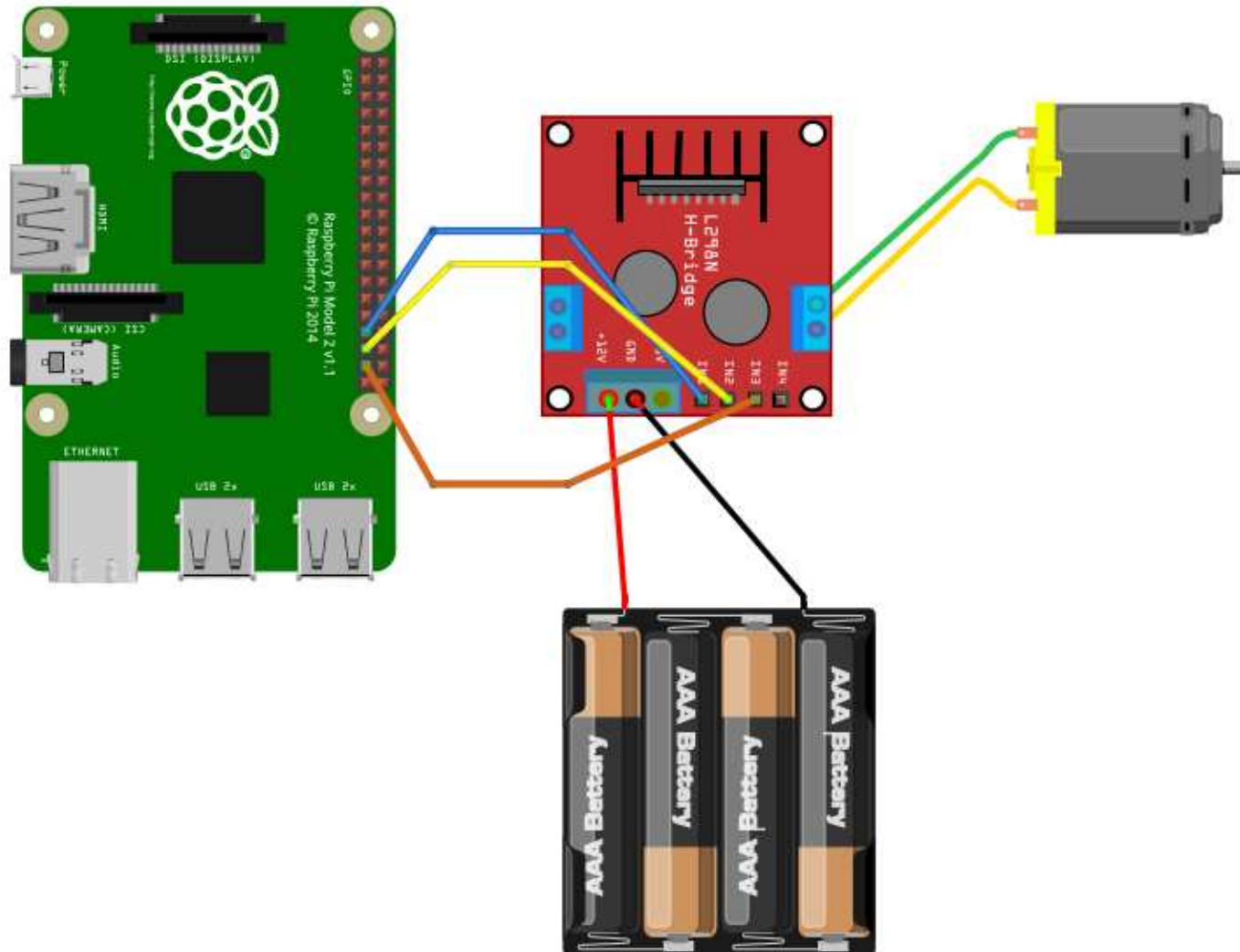
DC-MOTOREN & H-BRÜCKE



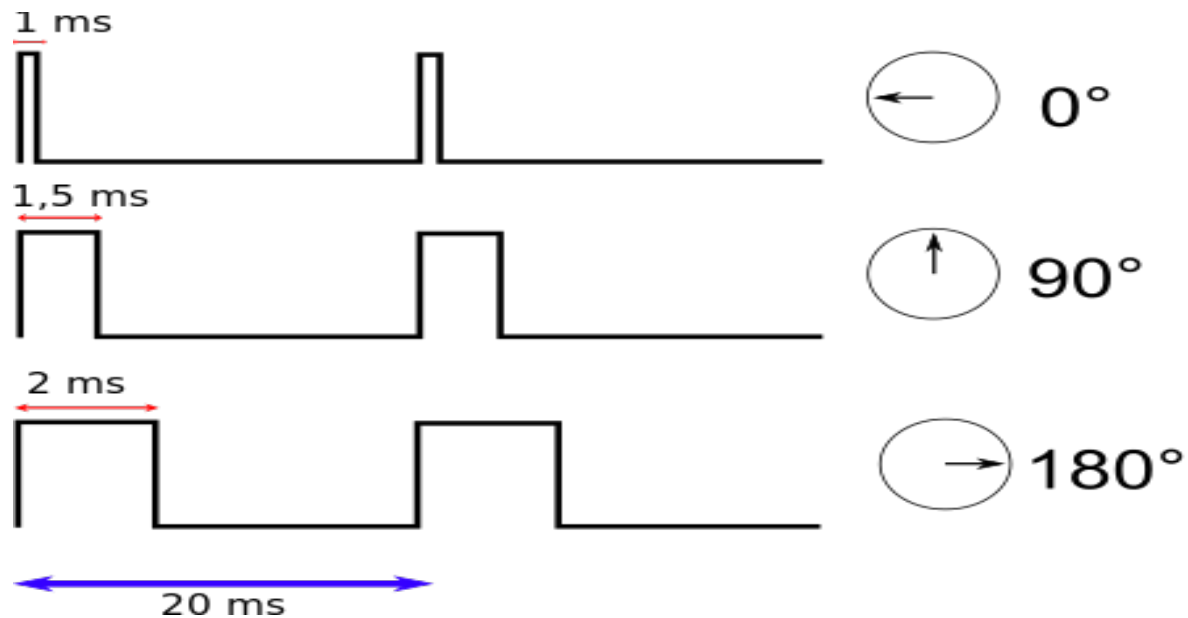
H-BRÜCKE



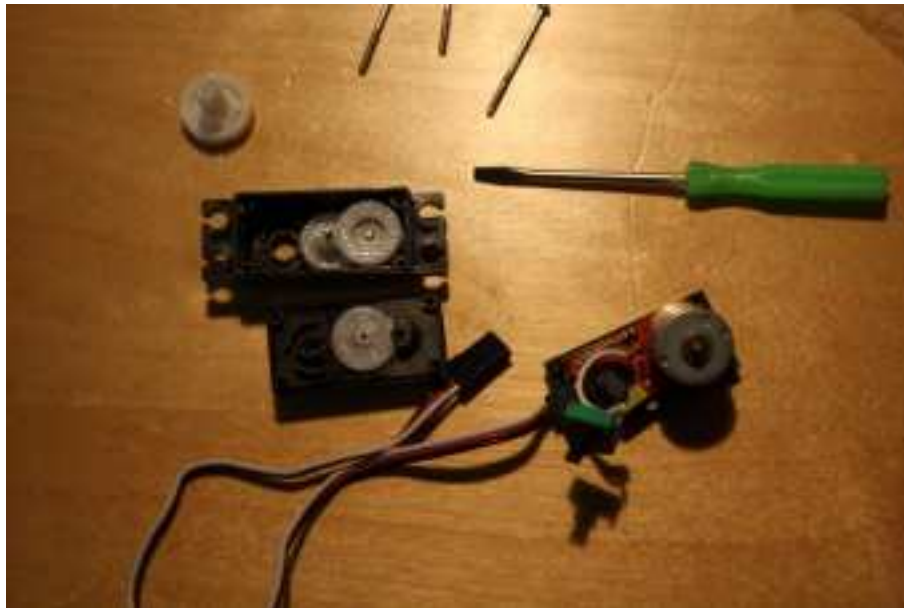
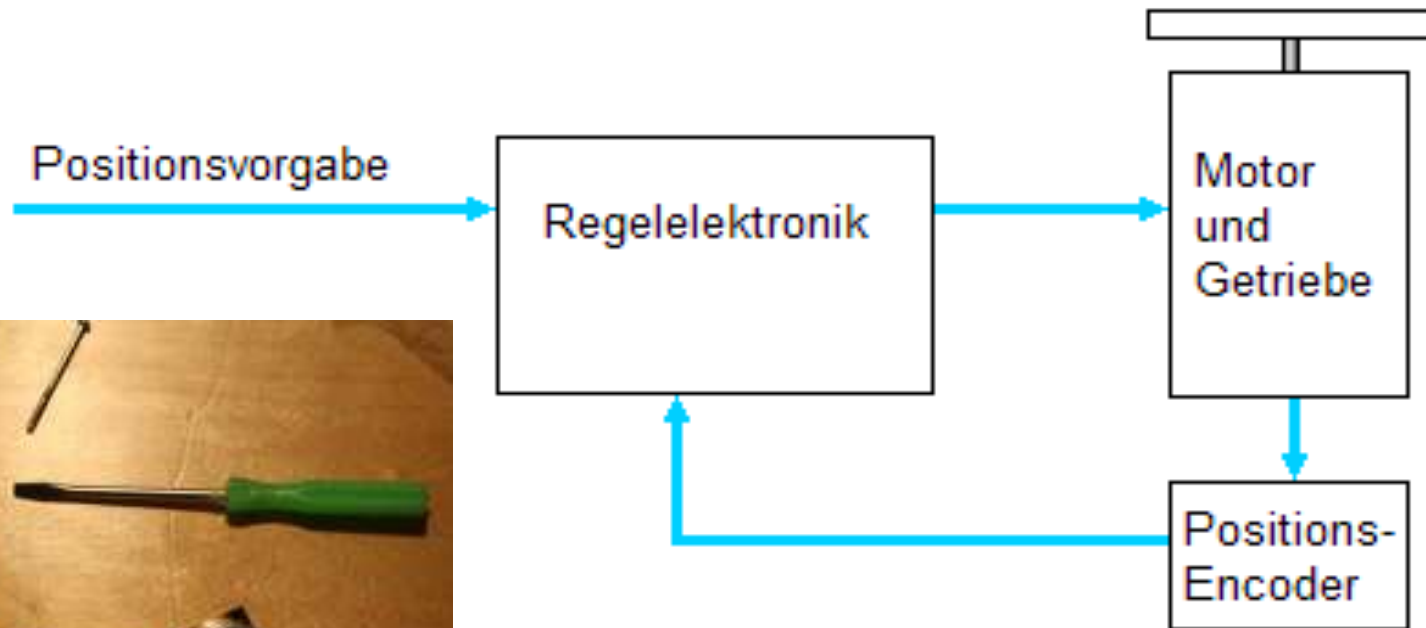
Fritzing-Software



SERVO & PWM-SIGNAL



SERVO & UMBAU



ENERGIEVERSORGUNG



L298N -Dual- H-Brücke

EnableA	MotorA1	MotorA2	Wirkung
0	0	0	Motor läuft aus
1	1	1	Motor stoppt
1	1	0	Motor dreht nach links
1	0	1	Motor dreht nach rechts
1	0	0	Motor stoppt



Code: PIN-Nummern

```
#!/usr/bin/python

import RPi.GPIO as GPIO

MotorEnableA    = 37
Motor1A         = 35
Motor2A         = 33

MotorEnableB    = 40
Motor2B         = 38
Motor1B         = 36
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
```

Raspberry Pi2 GPIO Header

Pin#	NAME	NAME	Pin#
01	3.3v DC Power	DC Power 5v	02
03	GPIO02 (SDA1 , I ² C)	DC Power 5v	04
05	GPIO03 (SCL1 , I ² C)	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 (I ² C ID EEPROM)	(I ² C 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>

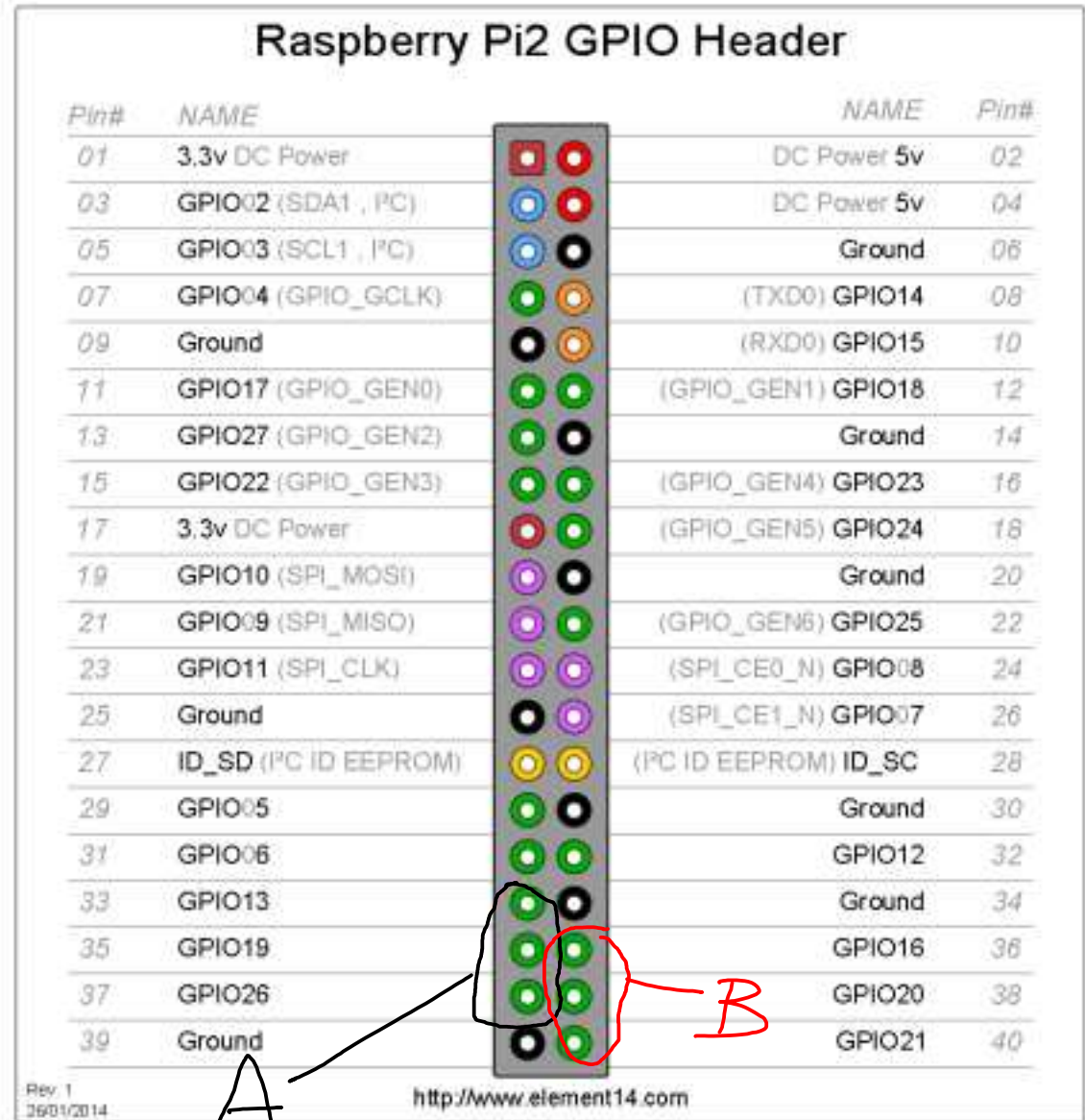
Code: PIN-Nummern

```
#!/usr/bin/python

import RPi.GPIO as GPIO

MotorEnableA = 37
Motor1A = 35
Motor2A = 33

MotorEnableB = 40
Motor2B = 38
Motor1B = 36
GPIO.setmode(GPIO.BOARD)
GPIO.setwarnings(False)
```



Code: PIN-Setup

```
GPIO.setup(MotorEnableA,GPIO.OUT)
GPIO.setup(Motor1A,GPIO.OUT)
GPIO.setup(Motor2A,GPIO.OUT)
```

```
GPIO.setup(MotorEnableB,GPIO.OUT)
GPIO.setup(Motor1B,GPIO.OUT)
GPIO.setup(Motor2B,GPIO.OUT)
```

```
pulsA=GPIO.PWM(MotorEnableA,100)
pulsB=GPIO.PWM(MotorEnableB,100)
```

```
GeschwindigkeitA=100
GeschwindigkeitB=100
geschwindigkeitA_langsam=100
geschwindigkeitB_langsam=100
```



Code: DC-Ansteuerung

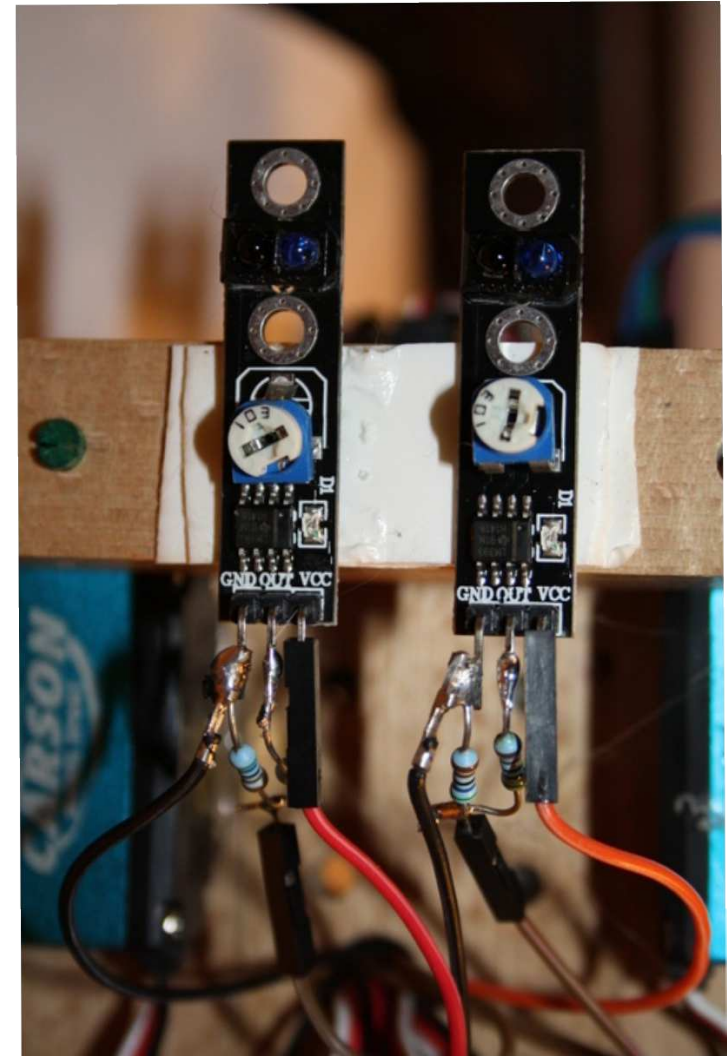
```

def all_off() :
    GPIO.output(Motor1A,GPIO.LOW)
    GPIO.output(Motor2A,GPIO.LOW)
    GPIO.output(Motor1B,GPIO.LOW)
    GPIO.output(Motor2B,GPIO.LOW)
    GPIO.output(MotorEnableA,GPIO.LOW)
    GPIO.output(MotorEnableB,GPIO.LOW)

def motorA(move):
    if move == 0 :
        GPIO.output(Motor1A,GPIO.LOW)
        GPIO.output(Motor2A,GPIO.LOW)

    #MotorA vorwaerts
    if move == 1 :
        GPIO.output(Motor1A,GPIO.LOW)
        GPIO.output(Motor2A,GPIO.HIGH)

    #MotorA rueckwärts
    if move == 2 :
        GPIO.output(Motor1A,GPIO.HIGH)
        GPIO.output(Motor2A,GPIO.LOW)
    
```



Code: DC-Funktionen



```
def forward():  
    pulsA.start(geschwindigkeitA)  
    pulsB.start(geschwindigkeitB)  
    motorA(1)  
    motorB(1)
```

```
def back():  
    pulsA.start(geschwindigkeitA)  
    pulsB.start(geschwindigkeitB)  
    motorA(2)  
    motorB(2)
```

```
def stop():  
    #pulsA.start(geschwindigkeitA)  
    #pulsB.start(geschwindigkeitB)  
    motorA(0)  
    motorB(0)
```

```
def right():  
    pulsA.start(geschwindigkeitA_langsam)  
    pulsB.start(geschwindigkeitB_langsam)  
    motorA(2)  
    motorB(1)
```

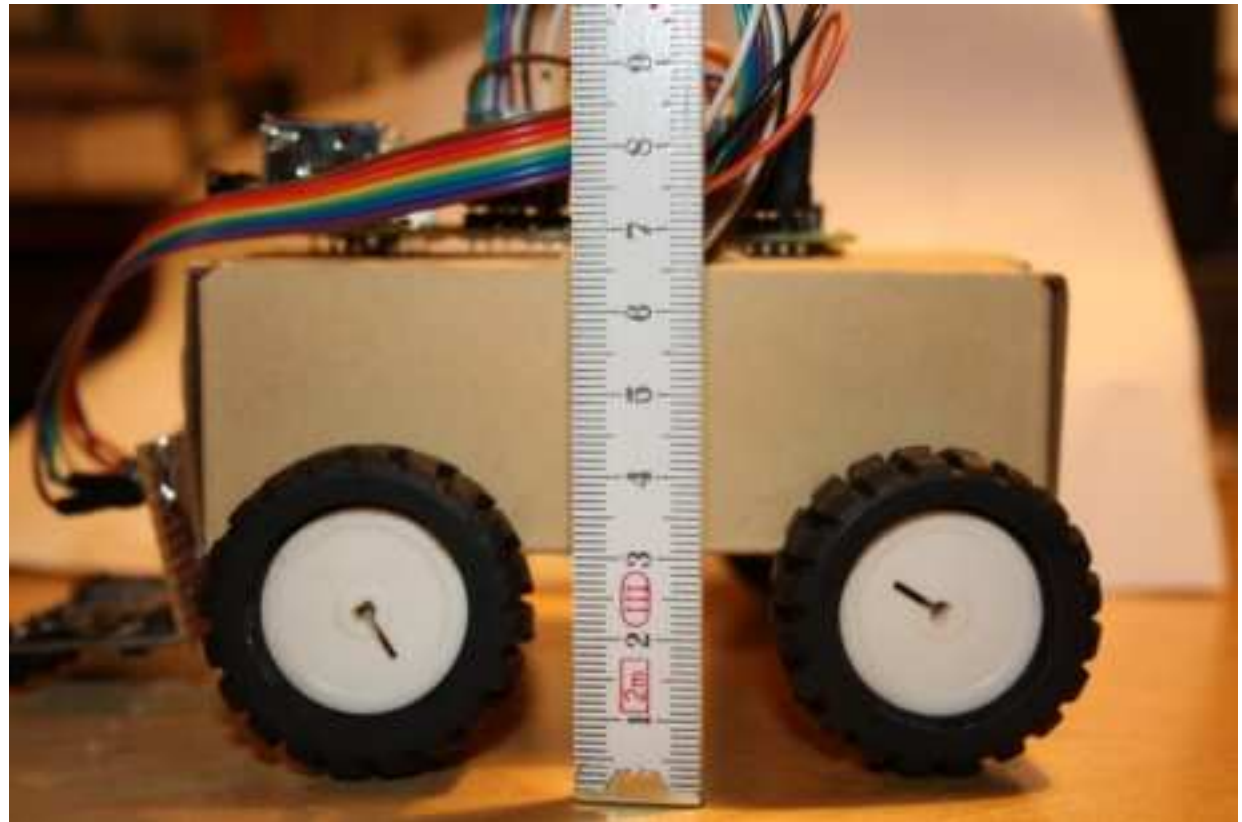
```
def left():  
    pulsA.start(geschwindigkeitA_langsam)  
    pulsB.start(geschwindigkeitB_langsam)  
    motorA(1)  
    motorB(2)
```

Code: Funktionen testen

```
#main
```

```
forward()
time.sleep(2)
stop()
Back()
time.sleep(2)
stop()
left()
time.sleep(2)
stop()
right()
time.sleep(2)
stop()
```

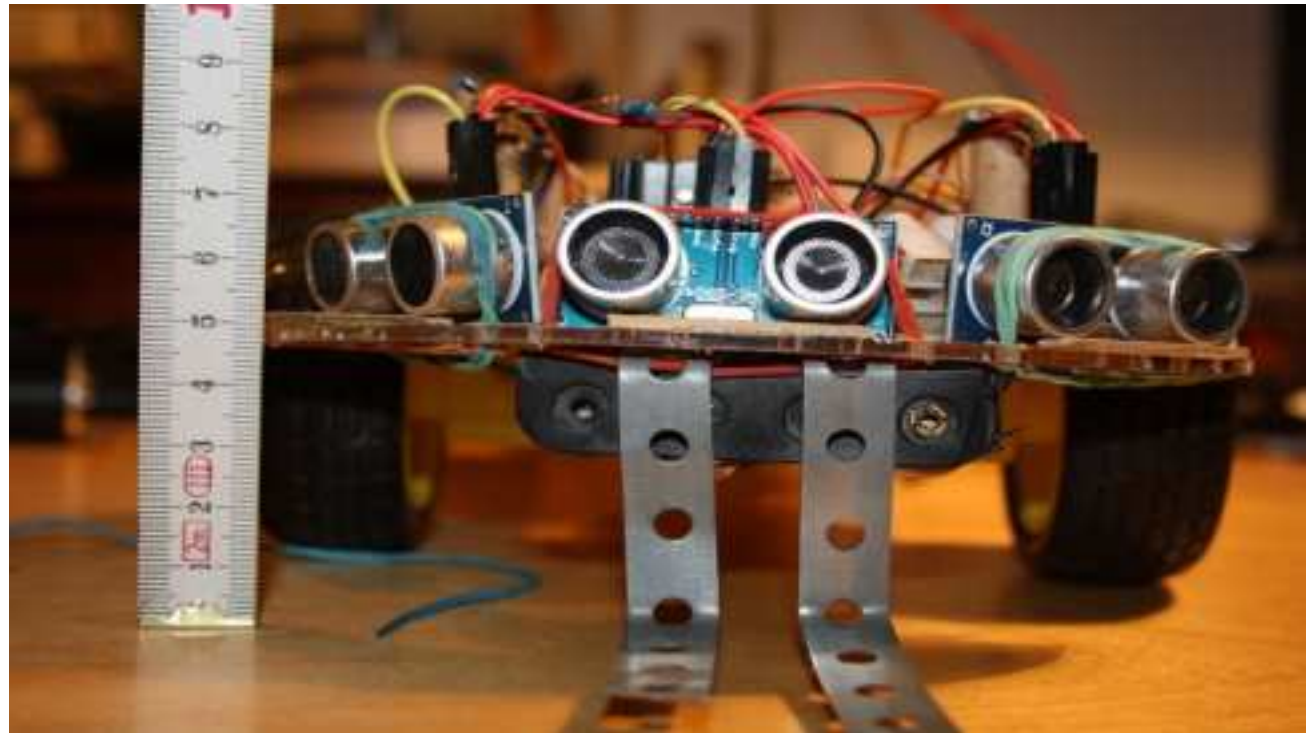
```
GPIO.cleanup()
```



Cursor-Tasten

```
import curses

stdscr = curses.initscr()
while 1:
    c = stdscr.getch()
    if c == ord('q'):
        stop()
        break
    if c == 65:
        #print "Vorwaerts"
        forward()
    if c == 66:
        #print "Rueckwaerts"
        back()
    if c == 67:
        #print "Rechts"
        right()
    if c == 68:
        #print "Links"
        left()
curses.endwin()
```



Code: Command Web

```
import sys

if sys.argv[1] == "forward" :
    kmmotor.forward()
if sys.argv[1] == "back" :
    kmmotor.back()
if sys.argv[1] == "left" :
    kmmotor.left()
if sys.argv[1] == "right" :
    kmmotor.right()

if sys.argv[1] == "stop" :
    kmmotor.stop()
```

```
#GPIO.cleanup()
```



Ultraschallsensor

Betriebsspannung:

$$U=5V$$

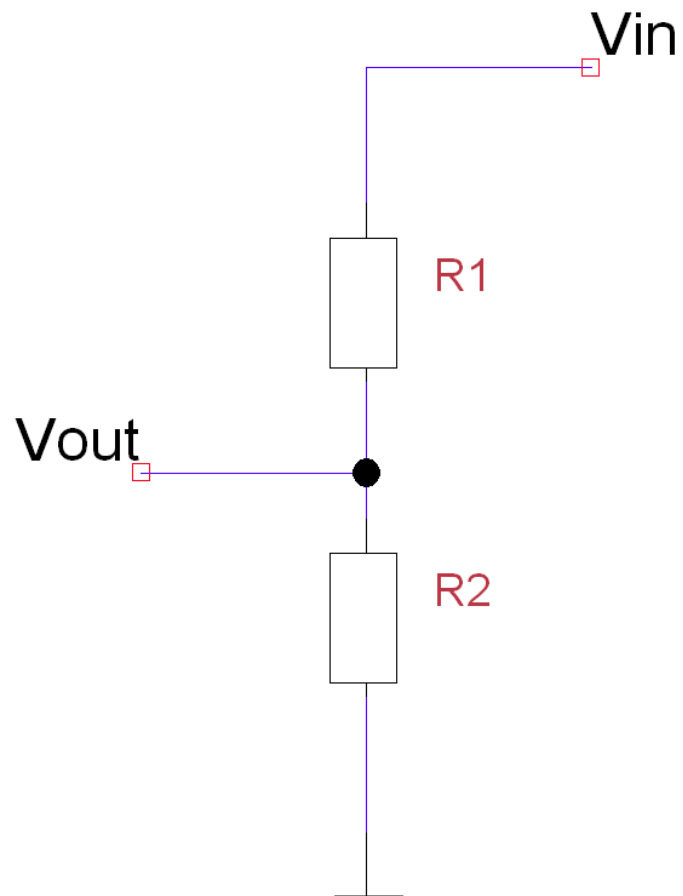
Echo-Signal:

Ebenfalls 5V, aber zu groß für den Raspberry PI!!! (max. 3,2V!!!)

→ Spannungsteiler



Spannungsteiler



$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

$$\frac{V_{out}}{V_{in}} = \frac{R_2}{R_1 + R_2}$$

Spannungsteiler

$$\frac{3.3}{5} = \frac{R2}{1000 + R2}$$

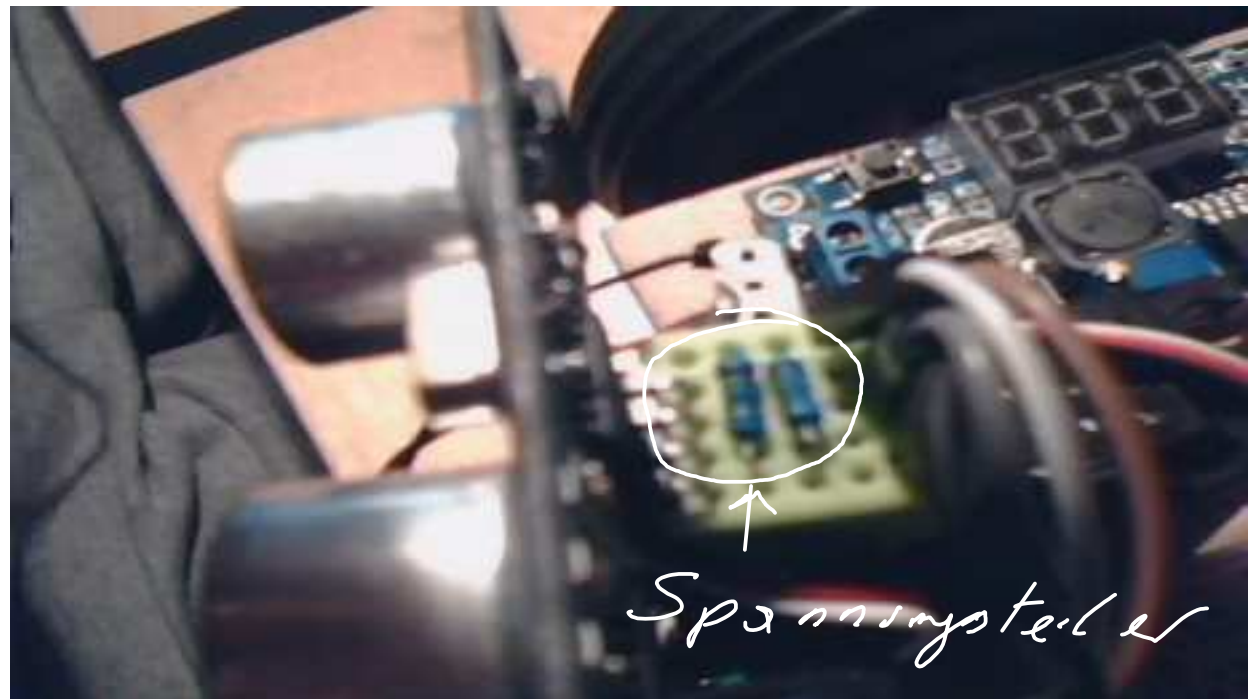
$$0.66 = \frac{R2}{1000 + R2}$$

$$0.66(1000 + R2) = R2$$

$$660 + 0.66R2 = R2$$

$$660 = 0.34R2$$

$$1941 = R2$$



Code: Ultraschallsensor

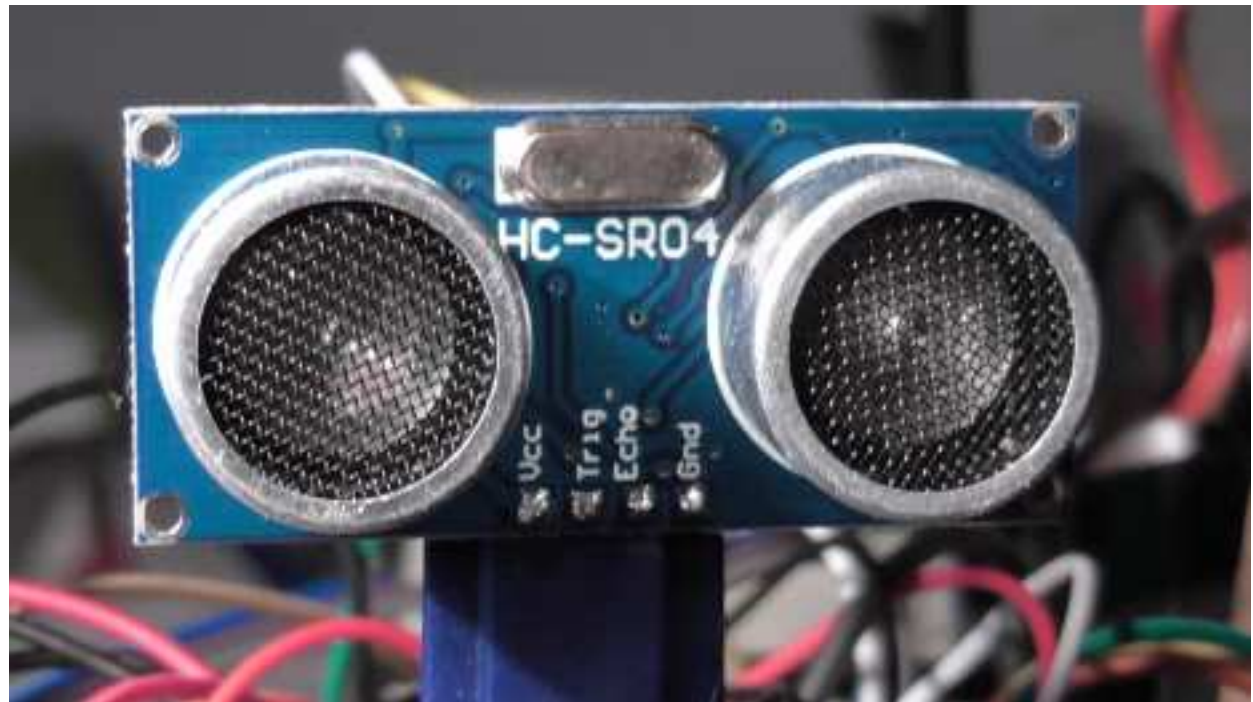
```

import RPi.GPIO as GPIO
import time
import os

GPIO.setmode(GPIO.BOARD)
TRIG = 29 #PIN 29
ECHO = 31 #PIN 31

GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
GPIO.output(TRIG,0)

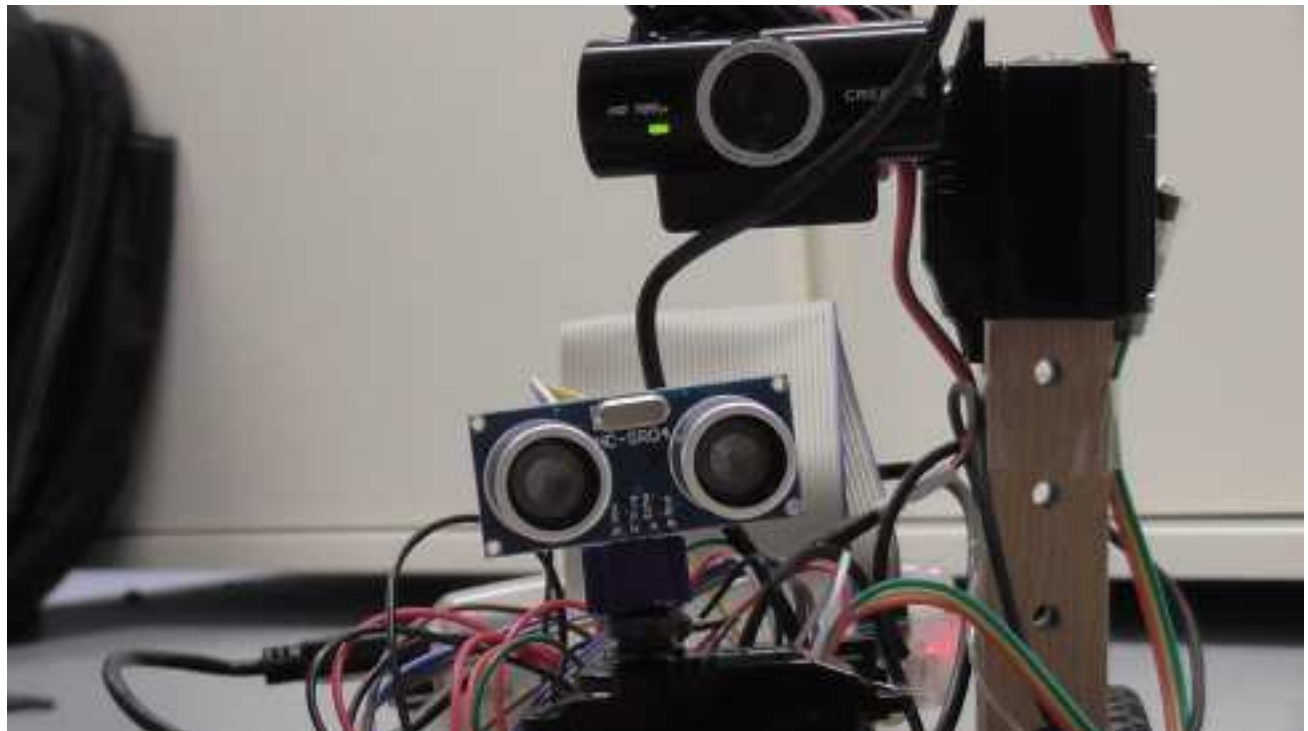
maxabstand=50
ruhezeit=0.2
maxzeit=1
    
```



Code: Ultraschallsensor

```
def distance():
    abstand = 0
    zeit = time.time()
    start = time.time()
    #GPIO.output(TRIG,0)
    #time.sleep(0.1)
    GPIO.output(TRIG,1)
    time.sleep(0.000001)
    GPIO.output(TRIG,0)
    #start = time.time()
    while (GPIO.input(ECHO) ==
0):
        start = time.time()
    while (GPIO.input(ECHO) ==
1):
        stop = time.time()
    abstand=(stop-start)*17000
    time.sleep(ruhezeit)
    print(int(abstand))
    return int(abstand)
```

```
#while True:
# test=distance()
# print(test)
#GPIO.cleanup
```



Code: Ultraschallsensor

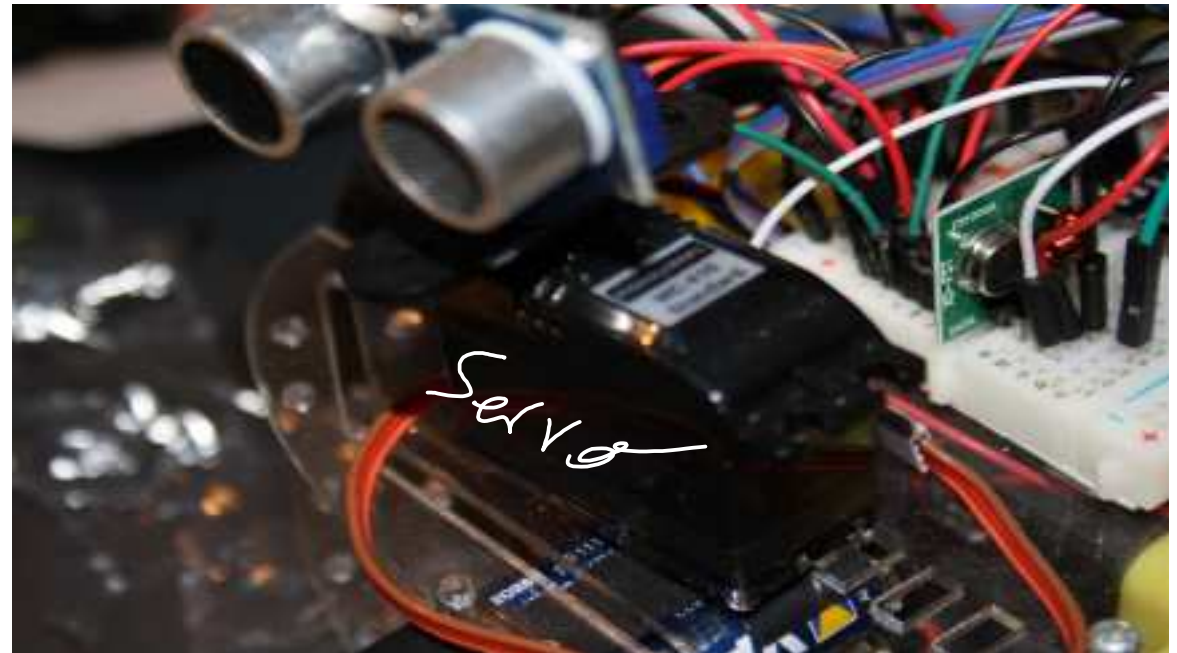
```

import RPi.GPIO as GPIO

GPIO.setup(servoC,GPIO.OUT)
pwmC=GPIO.PWM(servoC,50)

neutral=7.5
max=12
min=4
spanne=4.8
winkel=90.0
Stellzeit=0.2

ef position(pos):
    #pwmC.start(pos)
    wert=neutral+pos/winkel*spanne
    pwmC.ChangeDutyCycle(wert)
    time.sleep(stellzeit)
    pwmC.start(0)
    time.sleep(0.2)
    
```



WEBCAM & MJPG-STREAMER



```
apt-get install subversion  
apt-get install libv4l-dev  
apt-get install libjpeg8-dev  
apt-get install imagemagick
```



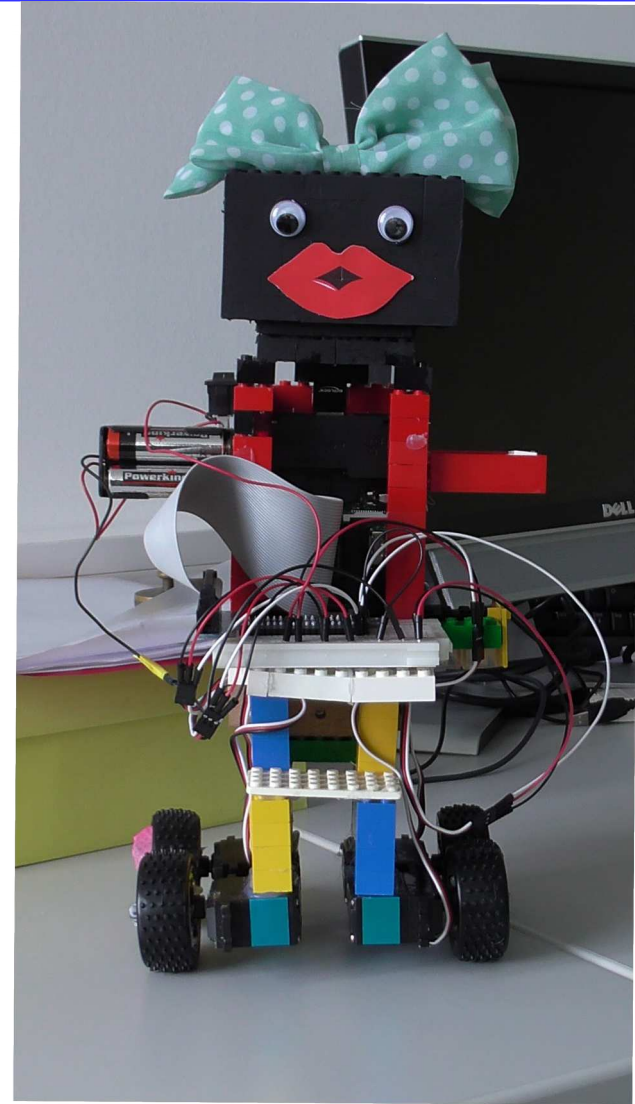
MJPEG-STREAMER & Compilieren

```
svn co https://svn.code.sf.net/p/mjpg-streamer/code/
```

```
cd mjpg-streamer/mjpg-streamer
```

```
make USE_LIBV4L2=true clean all
```

```
make DESTDIR=/usr install
```



Bilder Gallerie



```
sudo nano /usr/sbin/webcam.sh
```

```
mjpg_streamer -i "/usr/lib/input_uvc.so -d /dev/video0 -r 640x480 -f 1" -o "/usr/lib/output_http.so -p 8090 -w /var/www/mjpg_streamer"
```

Save the file and give it exec permission

```
sudo chmod 755 /usr/sbin/webcam.sh
```

Create a link to global applications folder so you can run it from any folder

```
sudo ln -s /usr/sbin/webcam.sh /etc/init.d/webcam.sh
```

Make sure it gets executed during boot

```
update-rc.d webcam.sh defaults 94 6
```



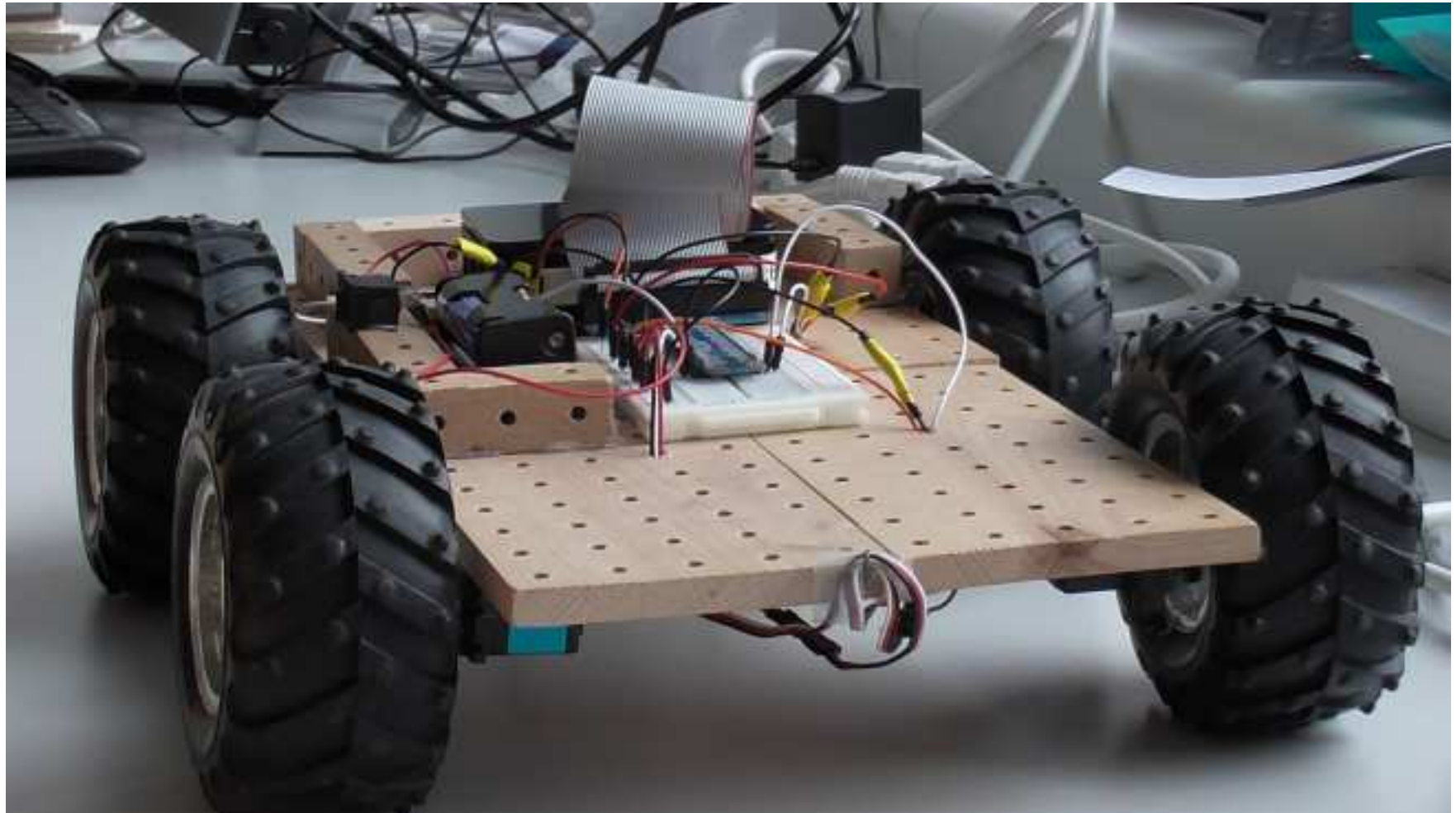
Webpage-Example

```

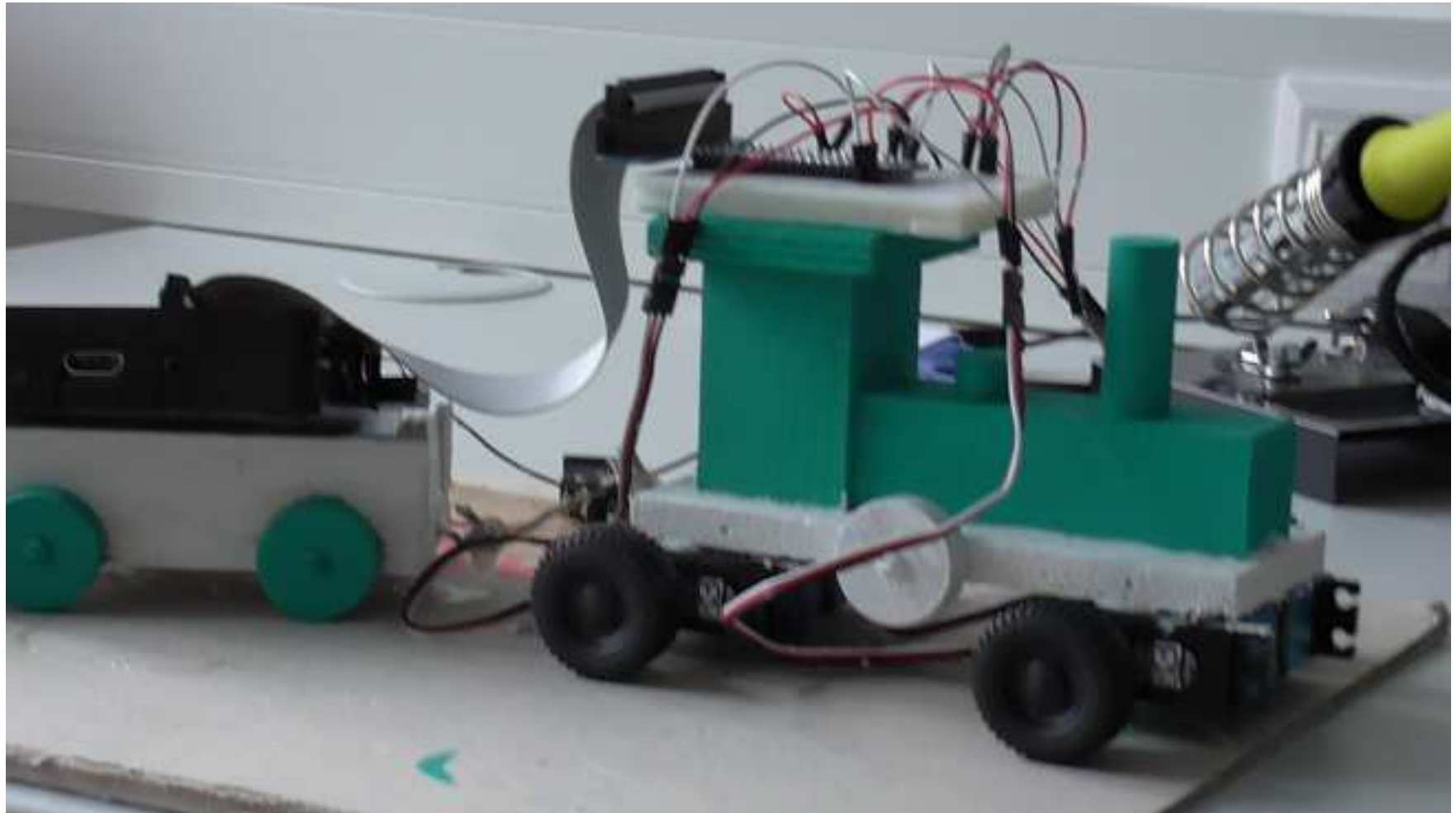
sudo nano /var/www/video.html
<html>
  <head><title>MJPG Streamer</title></head>
  <body>
    <h1>MJPG video of ELMTREE Lounge Room</h1>
    
    <br><a href="http://10.0.0.38:8090/?action=stream">View</a>
  </body>
</html>

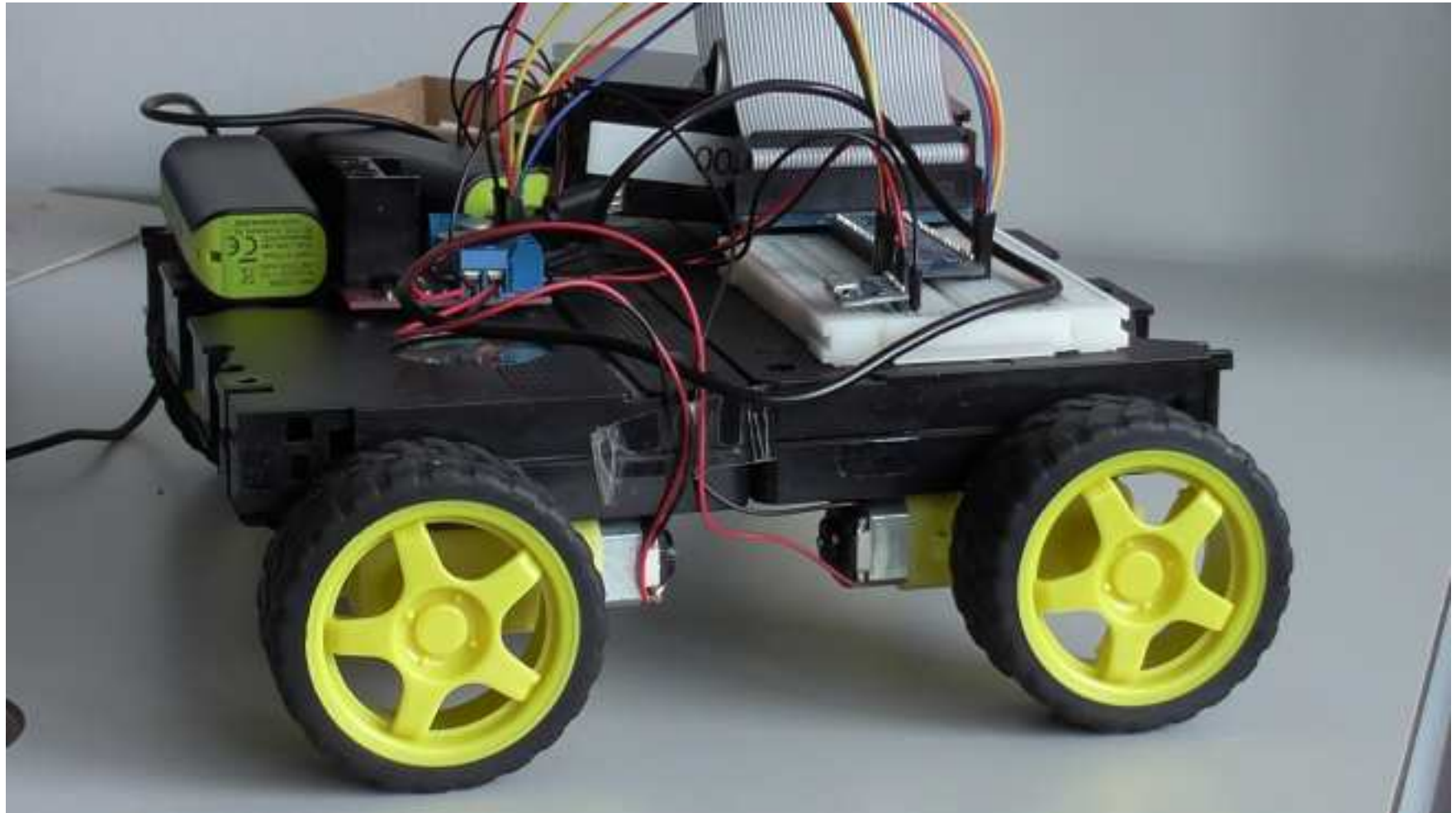
```







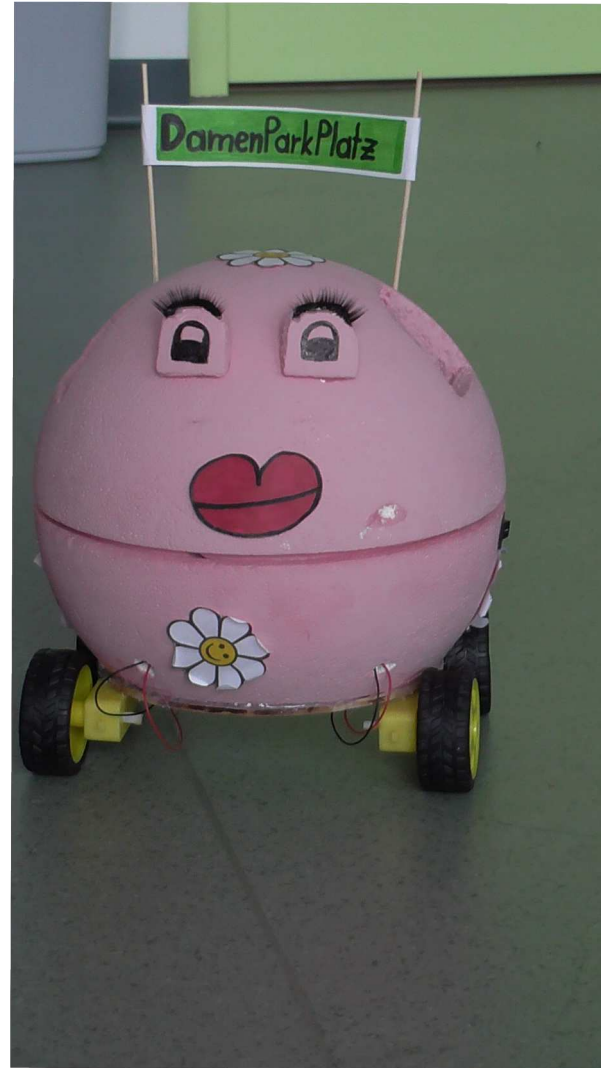
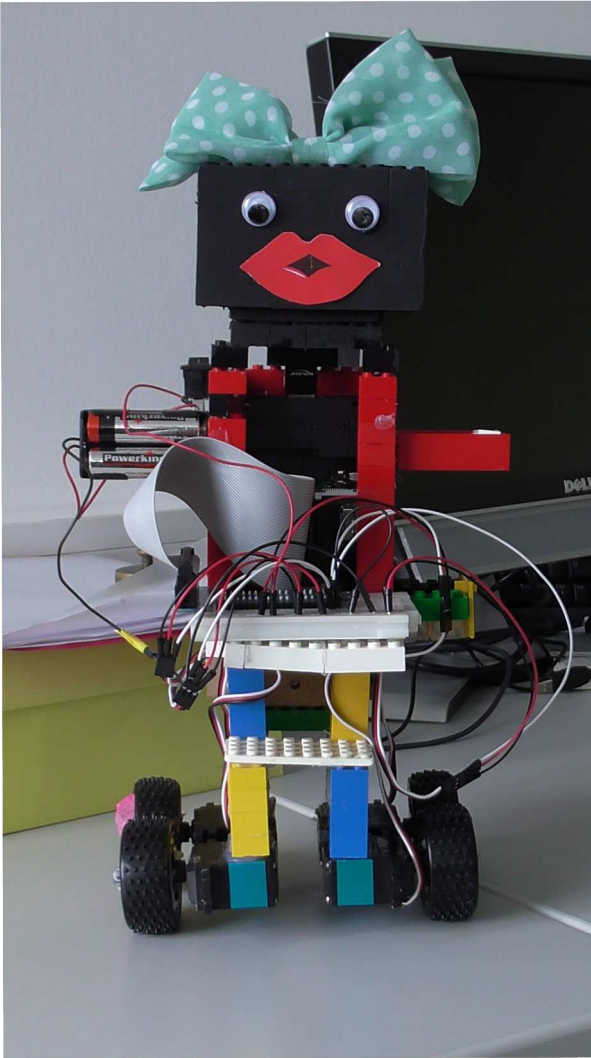




Bilder Gallerie



Bilder Gallerie



Self-Balancing Robot



Bilder Gallerie



A photograph of a white robot chassis inside a clear plastic storage bin. Two compact discs (CDs) are placed on the surface in front of the robot. The text 'Viel Erfolg bei der Robotik' is overlaid in white on the image.

**Viel Erfolg bei der
Robotik**

**Workshops:
klausmisof@gmx.at**