

Table of contents





















Table of Contents

Table of contents.....	1
1. Introduction	2
1.1 GPIO Verification	3
2.1 433MHz Wireless modules (transmitter + receiver)	5
2.2 DHT11 Temperature & Humidity.....	9
2.3 Laser module.....	10
2.4 Soil Moisture and Ground Humidity.....	11
2.5 Rain sensor	14
2.6 Motion Detector (PIR = Passive Infrared)	17
2.7 Light depending Resistor LDR.....	19
2.8 Shock sensor.....	20
2.9 Position sensor	21
2.10 Noise sensor	22
2.11 Proximity sensor	23
2.12 Flame sensor	24
2.13 Proximity sensor 2	25
2.14 Ultrasonic sensor HC-SR04.....	26
2.15 MQ-2 Gas sensor	29
2.16 Real-time clock	31

This Raspberry Pi eBook will assist you with the installation of the operating system and the preparation of the Raspberry.



1.1 GPIO Verification

Raspberry Pi 3 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. 2
29/02/2016

www.element14.com/RaspberryPi



Safety instructions



Important to know when working with the GPIO:

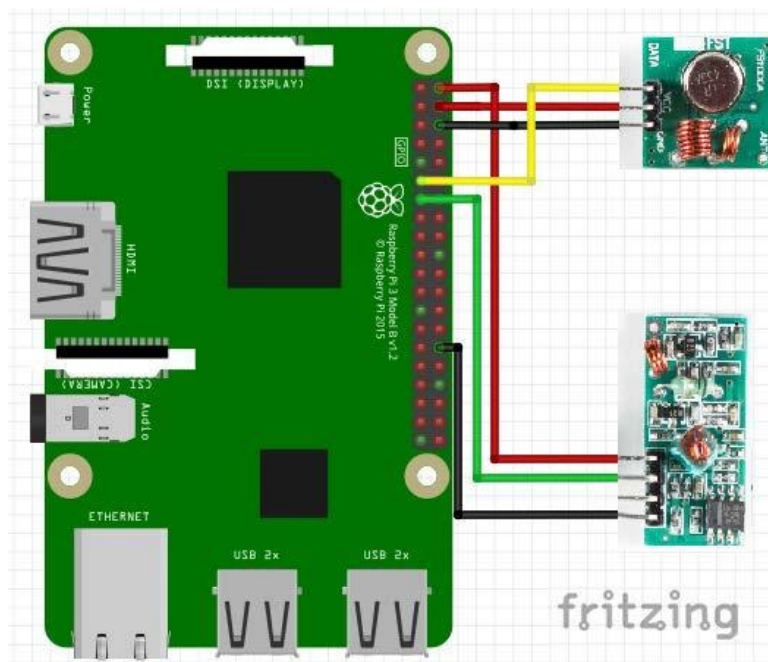
- **Short-circuit and improper wiring of the pins can be damaging to the Raspberry Pi!**
- **When changing the circuit or wiring of the board, always switch off the Raspberry Pi.**
- **The maximum voltage on the GPIO pins is 3.3 volts. Never connect 5 volts to the GPIO.**

2.1 433MHz Wireless modules (transmitter + receiver)



The transmitter (XY-FST) and receiver (XY-MK) wireless module set communicate at 433.92 MHz. With the module, for example, wireless sockets can be controlled, which will be further explained here in this eBook.

Wiring the module with the Raspberry Pi:



The modules have only 3 ports each, VCC, GND, and Data.

Az-Delivery

There are 4 pins attached to the receiver module with the middle 2 (DATA) being bridged.

Transmitter module (3 pins):

VCC is connected to **PIN 2 (5V)** on the Raspberry
GND is connected to **PIN 6 (GND)**
DATA is connected to **PIN 11 (GPIO 17)**

Red wire
Black wire
Yellow wire

Receiver module (4 pins):

VCC is connected to **PIN 4 (5V)** on the Raspberry
GND is connected to **PIN 30 (GND)**
DATA is connected to **PIN 13 (GPIO 27)**

Red wire
Black wire
Green wire

Antennas

To begin, we should attach an antenna to each module.
We quickly calculate this:

The module transmits on **433,92 MHz**.

The wavelength λ is calculated as follows: $\frac{\text{Speed of light}}{\text{Frequency}}$

$$\lambda = \frac{299.792.458 \text{ m/s}}{433.920.000 \text{ 1/s}} = 0,69 \text{ m}$$

The antenna should be $\frac{1}{4} \lambda$, that is $69\text{cm} / 4 \Rightarrow 17,25 \text{ cm}$
So we connect a wire, to the antenna, with a length of 17 cm.

After everything has been wired, the Raspberry Pi can be started.

Information: This manual is based on the Raspberry Pi Image from 29.11.2017 (Stretch - Lite) - Updates may require slight modifications to the instructions.

As an alternative to the above-mentioned pins used on the Raspberry here in this eBook, any ground pin can be used, as well as other GPIO pins. If the GPIO pins are changed, the sample software must be also adapted and recompiled.



„Programming“ the Raspberry Pi:

Before you install on the Raspberry Pi software, the Raspberry Pi should be brought up to date:

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

Do you want to continue? [Y/n] -> **y** (enter *Y* and confirm with *Enter*)

Now that the Raspberry Pi is updated, we can install software.

```
sudo apt-get install git-core
```

git-core: Software for downloading software from GIT

Do you want to continue? [Y/n] -> **y**

When *git* is completely installed, we load (`git clone`) from *git* the library from Ninjablocks and wiringPi. Then we compile the packages (`./build` or `make all`):

```
git clone git://git.drogon.net/wiringPi
cd ~/wiringPi
./build
cd ~
```

```
git clone --recursive https://github.com/ninjablocks/433Utils.git cd
~/433Utils/RPi_utils
make all
```

After the compilation, these packages can be found in the following folder:
433Utils/RPi_utils

```
pi@raspberrypi:~/433Utils/RPi_utils $
```

If this is not the case, then simply enter the following command:

```
cd ~/433Utils/RPi_utils
```

Let us first test the receiver and read the code of our remote control:

```
sudo ./RFSniffer
```

Now, when switched on and off the wireless socket, the Raspberry should receive something:

Az-Delivery

Received 263505
Received 263508

In my case, the Raspberry received 263505 when powered on and 263508 when shut down.

With the keyboard combination CTRL + C we end the RFSniffer.

In order to be able to switch a socket, enter *codesend* in the program. We give this code at startup and this then sends this code to 433.92 MHz.

```
sudo ./codesend 263505
```

The wireless socket is now switched on.

And shut down is as follows:

```
sudo ./codesend 263508
```

Great, now we can control our wireless socket. But what can we do if we do not have a remote control, which we can read from?



O N	O N		O N	O N	O N				
		O F F				O F F	O F F	O F F	O F F
1	2	3	4	5	A	B	C	D	E

For that, there is the command *send*. We issue this command to the system code (the first five switches on the outlet), followed by the outlet number (1, 2, 3, ...) and 0 (off) or 1(on).

In my case, it is as follows:

```
sudo ./send 11011 1 0
```

 OFF

```
sudo ./send 11011 1 1
```

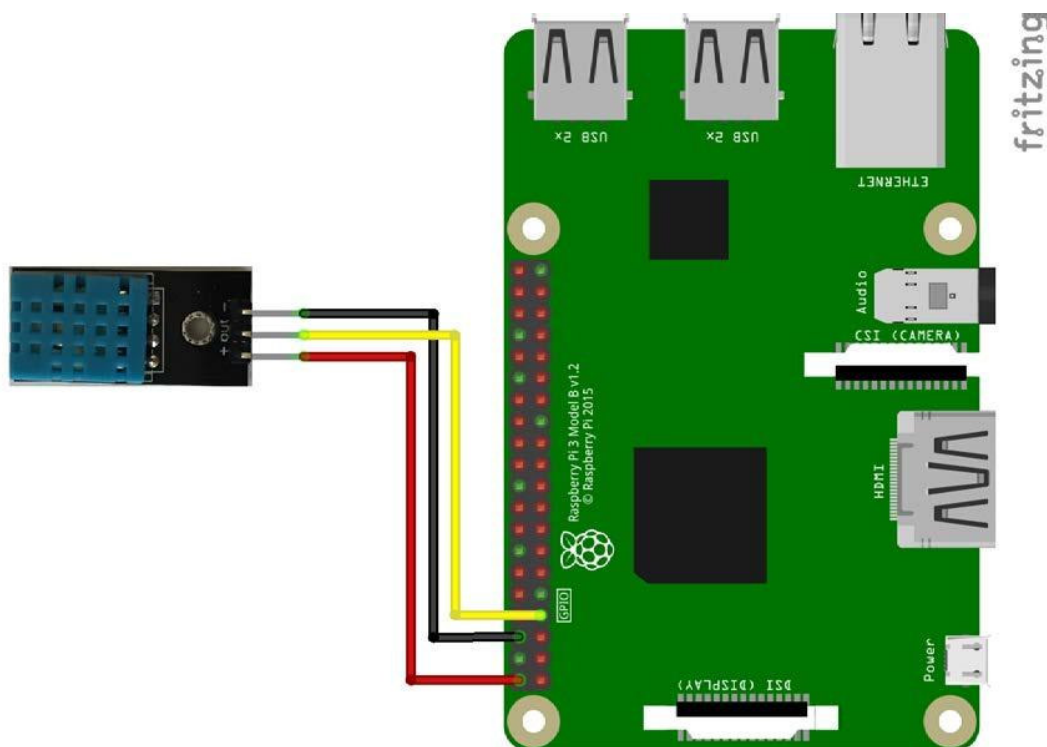
 ON

2.2 DHT11 Temperature & Humidity

Now we use the temperature and humidity sensor DHT11.

This is controlled by 1-wire. The 1-wire protocol is supported on the Raspberry only on the GPIO4. Therefore, the 1-wire connection of the temperature sensor (middle connection) must be connected here.

Let us build the circuit.



If everything is correctly and accordingly wired, we can begin with the installation of the required libraries:

```
sudo apt-get install build-essential python-dev python-openssl git
```

```
git clone https://github.com/adafruit/Adafruit_Python_DHT.git && cd  
Adafruit_Python_DHT
```

```
sudo python setup.py install
```

If everything had worked out well, we would be able to already read the temperature and

humidity. The easiest way to do that is with the demo files:

```
cd examples
```

```
sudo ./AdafruitDHT.py 11 4
```

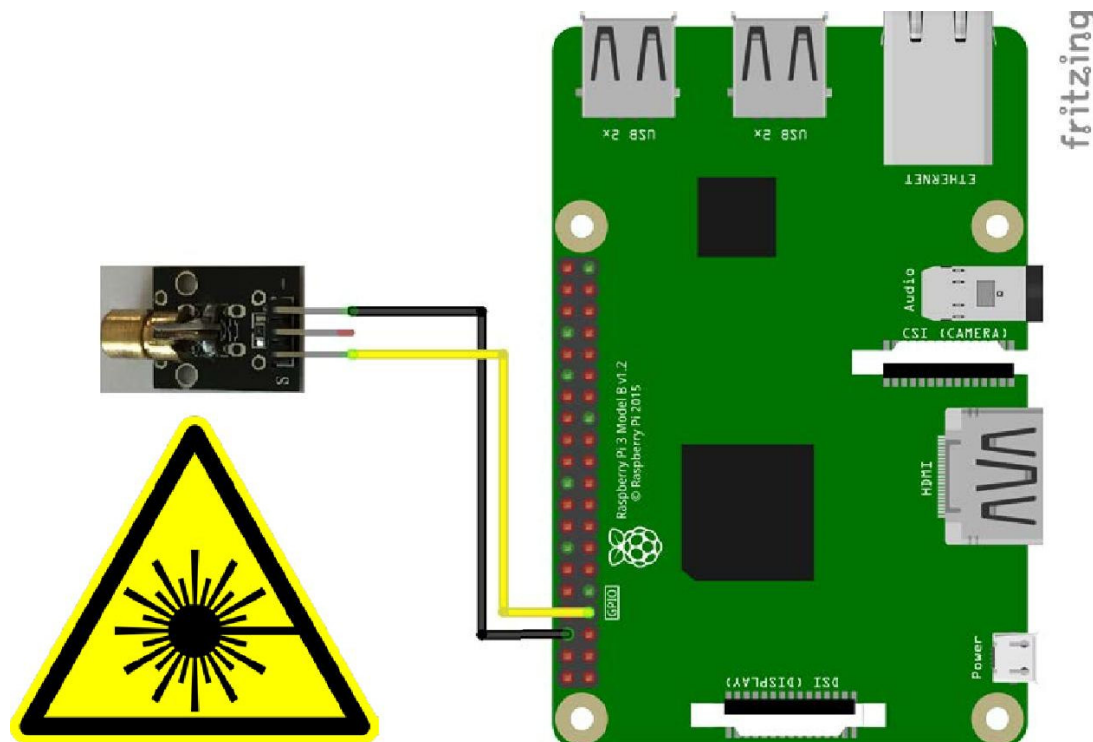
The first parameter (11) indicates which sensor had been used (22 for the DHT22) and the second, to which GPIO it is connected (not the pin number, but the GPIO number). We receive the following information:

Temp=24.0* Humidity=41.0%

Attention: The sensors are ready only every two seconds. A query to the sensor takes about two seconds. During this time a second query is not possible and should not be pursued.

2.3 Laser module

The laser module is connected directly to a GPIO. When the corresponding GPIO is switched on, the laser is also switched on.



In order to control the GPIO we do not need any special software, we simply use the existing system:

Az-Delivery

```
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	Write a 4 in the file xx (initialize the GPIO 4)
echo "out" > xx	Write out in the file xx (define GPIO 4 as output)
chmod	Change permissions
echo "1" >	Write a 1 in the file xx (turn GPIO 4 on)

With the last command (echo "1"> / sys / class / gpio / gpio4 / value) we write in the file the output value of our GPIO4. If we write a 0, we turn off the pin (laser).

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

Note: After a restart of the Raspberry Pi, the initialization must be carried out again!

2.4 Soil Moisture and Ground Humidity

The Soil Moisture Module provides a digital output and an analog value. Unfortunately, the Raspberry Pi only has digital inputs, so we can only use this one.

The digital output sends a signal as soon as a threshold value is exceeded, but the value cannot be determined exactly. The threshold value can be changed by a potentiometer (if the value has been exceeded, the green light will be turned on).

To read the GPIO we use the GPIO library *wiringPi*.

```
sudo apt-get -y install git-core
```

```
cd ~
```

```
git clone git://git.drogon.net/wiringPi
```

```
cd wiringPi
```

```
./build
```

Az-Delivery

After installing *wiringPi* we can test it right away:

```
gpio -v
```

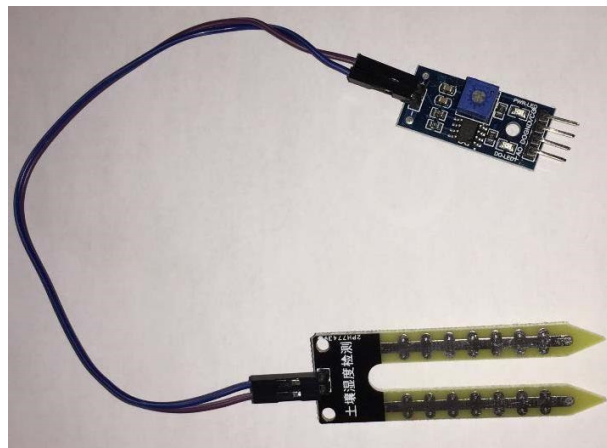
```
gpio readall
```

We receive the following information:

```
pi@raspberrypi3p:~/wiringPi $ gpio readall
```

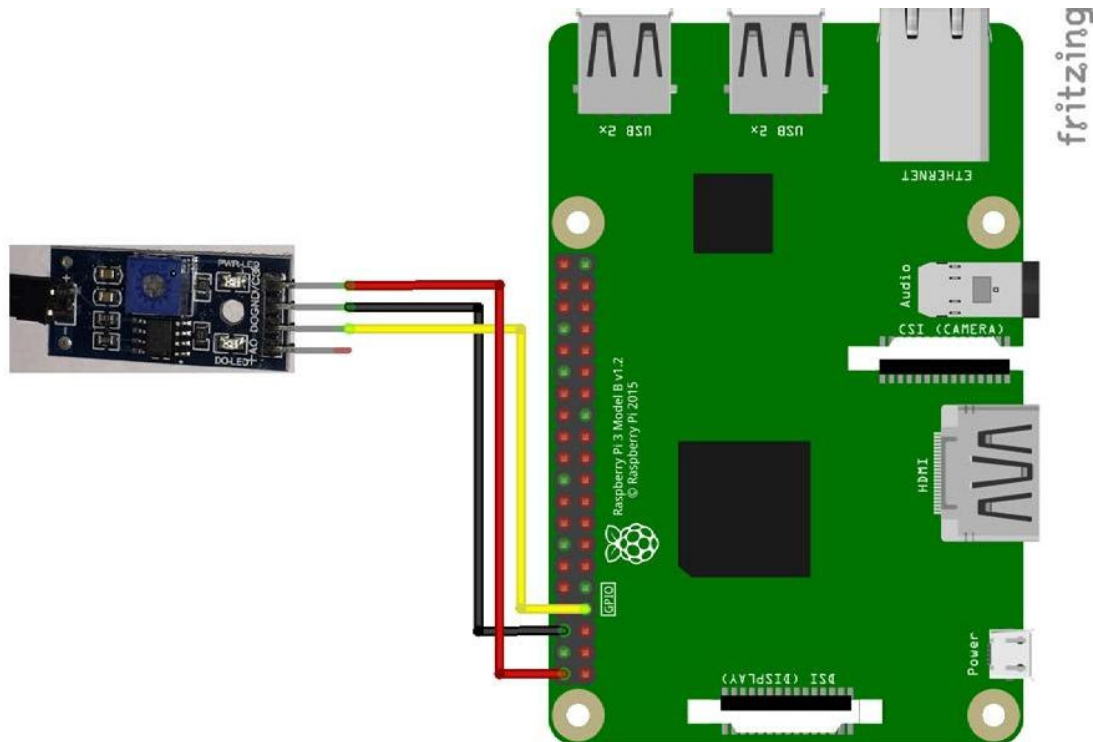
--Pi 3--											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	IN	1	3	4		5v			
3	9	SCL.1	IN	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	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--											

Now we can pin the humidity sensor on the sensor board:



Az-Delivery

And connect to the Raspberry:



Now we program the few lines to the query:

```
cd ~
```

```
touch feuchte.sh
```

```
nano feuchte.sh
```

```
#!/bin/sh
gpio -g mode 4 in
while true
do
result="$( gpio -g read 4 )"
if [ "$result" = "0" ]; then
echo "Boden ist Feucht"
fi
if [ "$result" = "1" ]; then
echo "Boden ist Trocken"
fi
sleep 2
done
```

```
bash /home/pi/feuchte.sh
```

Az-Delivery

We now get an output every 2 seconds if the soil is drier or wetter than the set threshold.

```
pi@raspberrypi3p:~ $ bash /home/pi/feuchte.sh
Boden ist Trocken
Boden ist Trocken
Boden ist Trocken
Boden ist Trocken
Boden ist Feucht
Boden ist Feucht
Boden ist Trocken
Boden ist Trocken
```

With CTRL + C, the program will be terminated.

2.5 Rain sensor

The rain sensor provides a digital output and an analog value. Unfortunately, the Raspberry Pi only has digital inputs, so we can only use this one.

The digital output sends a signal as soon as a threshold value is exceeded, but the value cannot be determined exactly. The threshold can be changed by the potentiometer (if it has been reached, the green light will be turned on).

To read the GPIO we use the GPIO library *wiringPi*.

```
sudo apt-get -y install git-core
```

```
cd ~
```

```
git clone git://git.drogon.net/wiringPi
```

```
cd wiringPi
```

```
./build
```

After installing *wiringPi* we can test it right away:

```
gpio -v
```

```
gpio readall
```

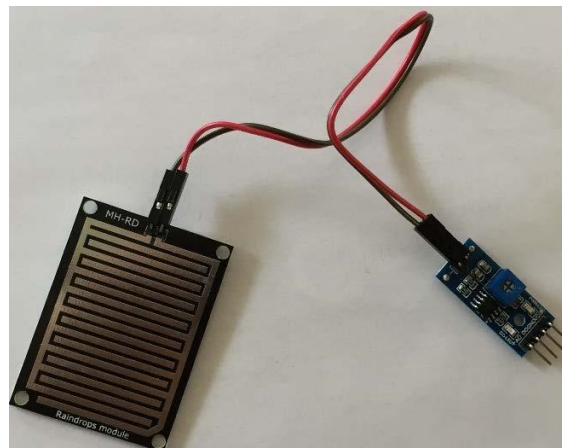

Az-Delivery

We receive the following information:

```
pi@raspberrypi3p:~/wiringPi $ gpio readall
```

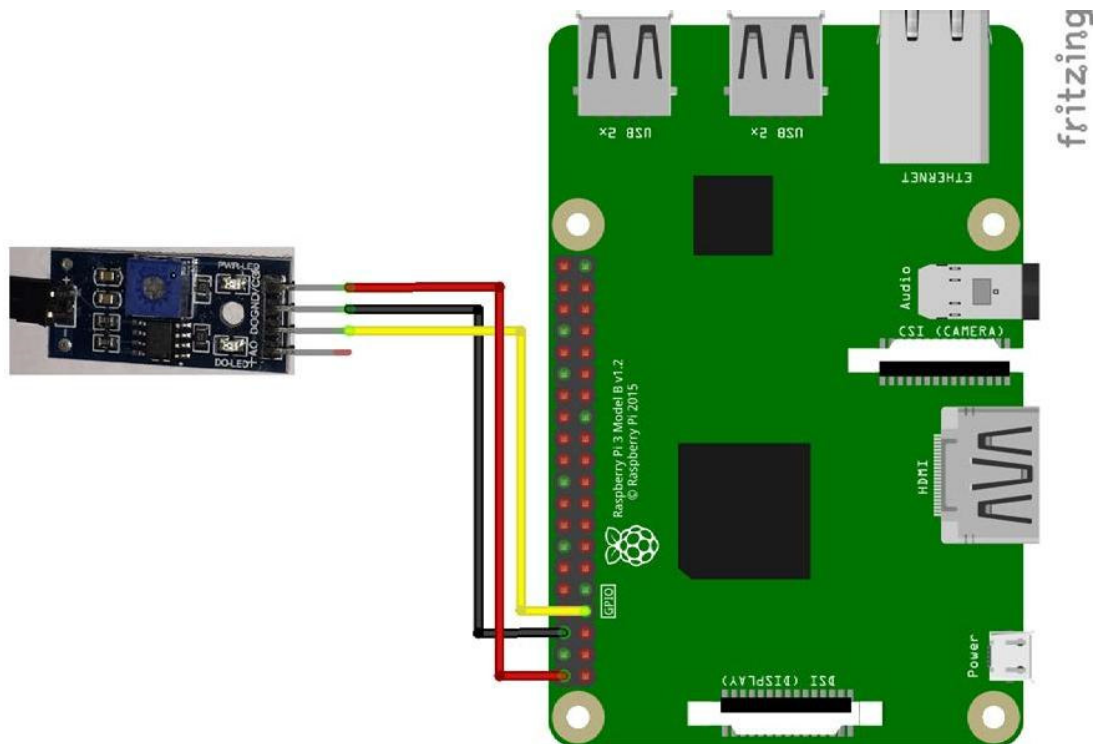
Pi 3+											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	IN	1	3	4		5v			
3	9	SCL.1	IN	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	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+											

Now we can pin the rain sensor on the sensor board:



Az-Delivery

And now we connect to the Raspberry:



Now we program the few lines to the query:

```
cd ~
```

```
touch regen.sh
```

```
nano regen.sh
```

```
#!/bin/sh
gpio -g mode 4 in
while true
do
result="$( gpio -g read 4 )"
if [ "$result" = "0" ]; then
echo "Es regenet"
fi
if [ "$result" = "1" ]; then
echo "Kein Regen"
fi
sleep 2
done
```

```
bash /home/pi/regen.sh
```

We now receive every 2 seconds information on whether it is raining or not, or whether the set threshold has been reached or not.

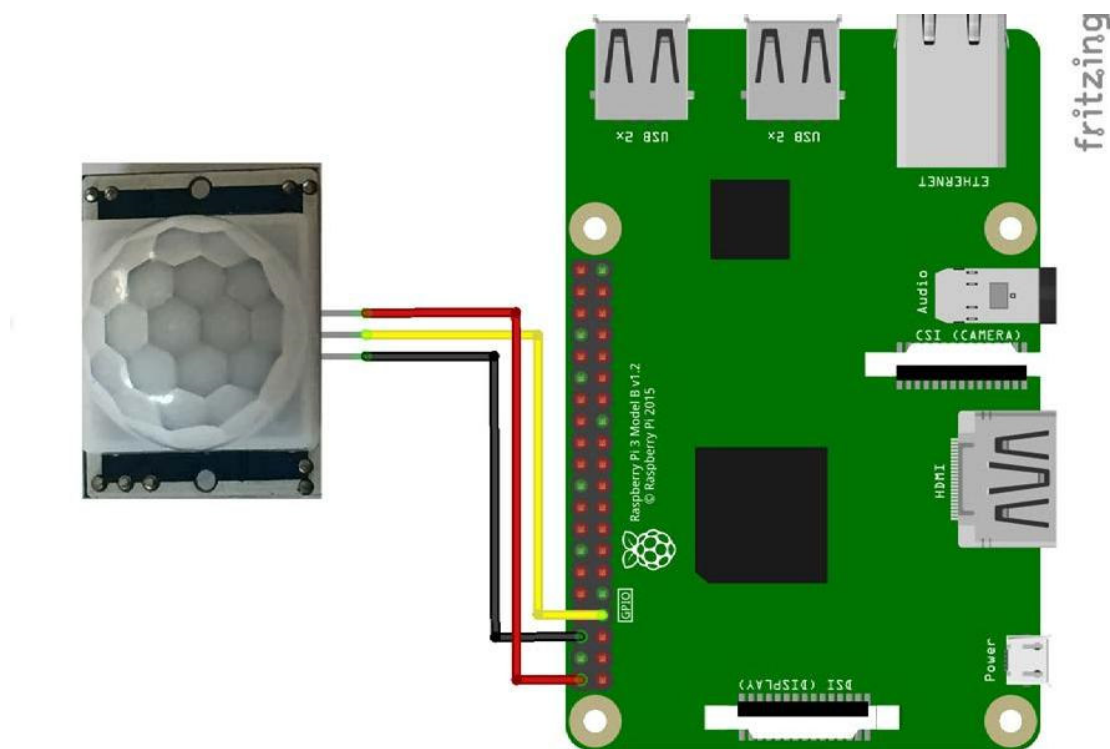
Az-Delivery

```
pi@raspberrypi3p:~ $ bash /home/pi/regen.sh
Kein Regen
Kein Regen
Es regenet
Es regenet
Kein Regen
Kein Regen
```

With CTRL + C, the program is terminated.

2.6 Motion Detector (PIR = Passive Infrared)

The motion detector provides a signal when the detected infrared field changes.



We can also query this sensor with a *shell program*, but in this case, we will use a Python program.

Az-Delivery

```
cd ~
```

```
touch pir.py
```

```
nano pir.py
```

```
import RPi.GPIO as
GPIO import time

SENSOR_PIN = 4

GPIO.setmode(GPIO.BCM)
GPIO.setup(SENSOR_PIN, GPIO.IN)

def bewegung(channel):
    print('Es gab eine Bewegung!')

try:
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING, callback=bewegung)
    while True:
        time.sleep(100)
except KeyboardInterrupt:
    print "Programm wurde
beendet." GPIO.cleanup()
```

```
python pir.py
```

Jedes Mal wenn der Bewegungsmelder etwas erkennt wird bei dem Python Programm ein Interrupt ausgeführt und „Es gabe eine Bewegung!“ ausgegeben.

Each time the motion detector detects something, the Python program executes an interrupt and displays "There was a movement!".

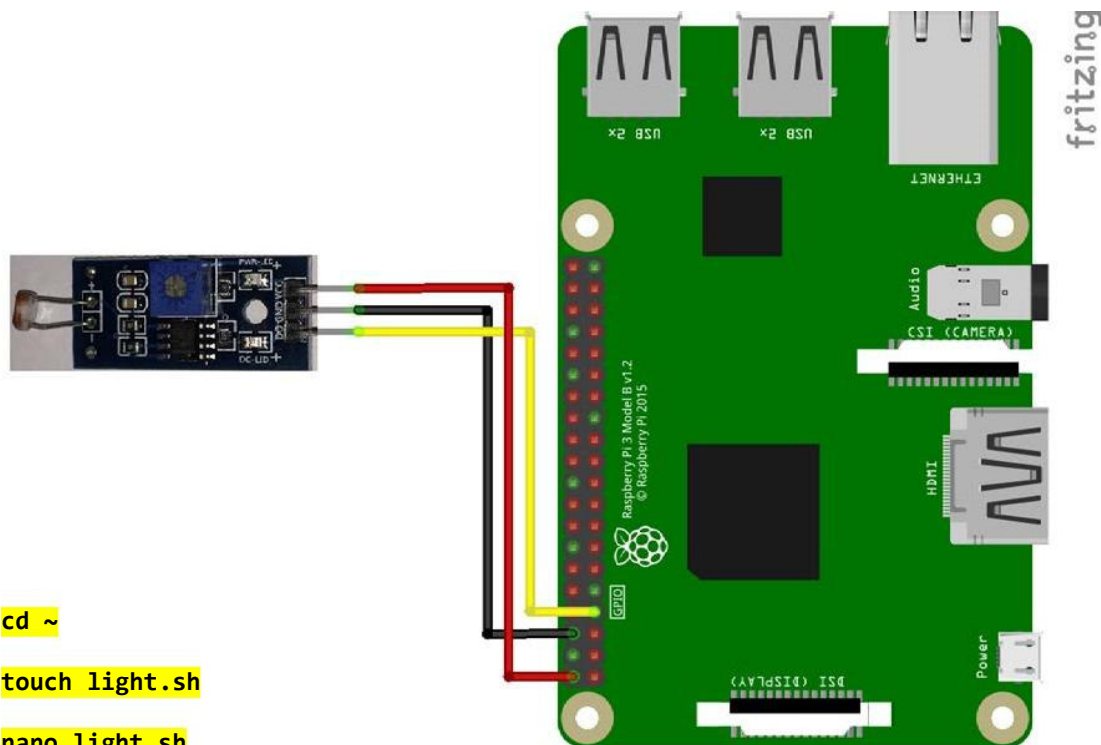
You would possibly have to turn on the 2 potentiometers and adjust the sensors.

```
pi@raspberrypi3p:~ $ python pir.py
Es gab eine Bewegung!
Es gab eine Bewegung!
Es gab eine Bewegung!
Es gab eine Bewegung!
```

2.7 Light depending Resistor LDR

The light depending resistor changes its resistance with the light intensity. As soon as the setpoint value on the potentiometer has been exceeded, the output is switched.

In section 2.4 "Soil Moisture and Ground Humidity" it is described how to install *wiringPi*.



```
cd ~
```

```
touch light.sh
```

```
nano light.sh
```

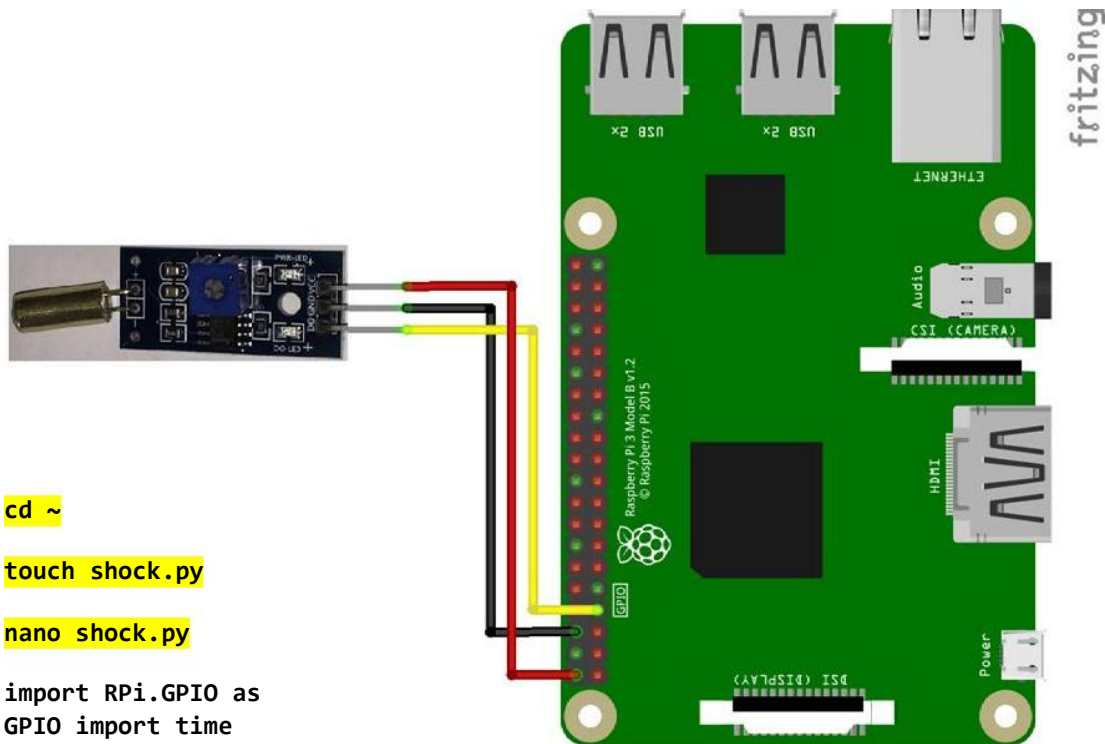
```
#!/bin/sh
gpio -g mode 4 in
while true
do
result="$( gpio -g read 4 )"
if [ "$result" = "0" ]; then
echo "Es ist hell"
fi
if [ "$result" = "1" ]; then
echo "Es ist dunkel"
fi
sleep 2
done
```

```
bash /home/pi/licht.sh
```

```
pi@raspberrypi3p:~ $ bash /home/pi/licht.sh
Es ist hell
Es ist dunkel
Es ist dunkel
Es ist hell
Es ist hell
Es ist hell
```

2.8 Shock sensor

The shock sensor gives a signal when a shake, tremor or a vibration above a set threshold has been detected.



```
cd ~
```

```
touch shock.py
```

```
nano shock.py
```

```
import RPi.GPIO as  
GPIO import time
```

```
SENSOR_PIN = 4
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setup(SENSOR_PIN, GPIO.IN)
```

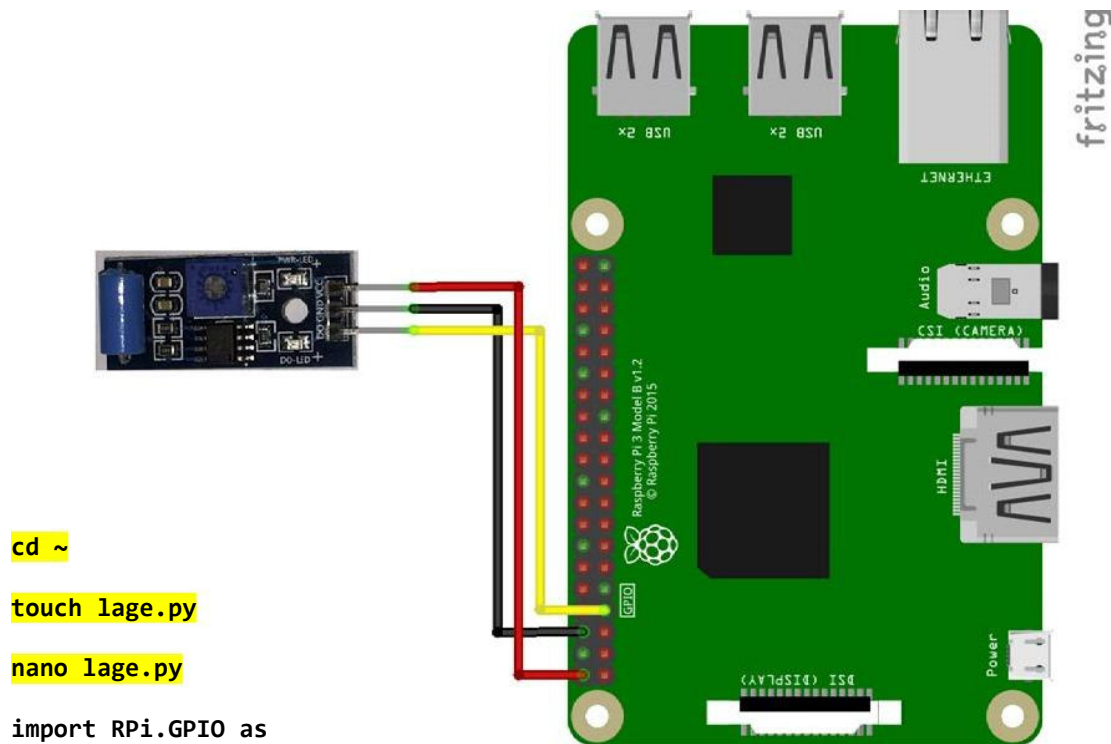
```
def shock(channel):  
    print('Es gab eine Erschuetterung!')  
try:  
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING,  
        callback=shock) while True:  
        time.sleep(100)  
except KeyboardInterrupt:  
    print "Programm wurde  
beendet." GPIO.cleanup()
```

```
python shock.py
```

```
pi@raspberrypi3p:~ $ python shock.py  
Es gab eine Erschuetterung!  
Es gab eine Erschuetterung!  
Es gab eine Erschuetterung!  
Es gab eine Erschuetterung!
```

2.9 Position sensor

The position sensor gives a signal when a position has been detected above a certain threshold. The position sensor is in fact also a vibration and tremor sensor.



```
cd ~
```

```
touch lage.py
```

```
nano lage.py
```

```
import RPi.GPIO as  
GPIO import time
```

```
SENSOR_PIN = 4
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setup(SENSOR_PIN, GPIO.IN)
```

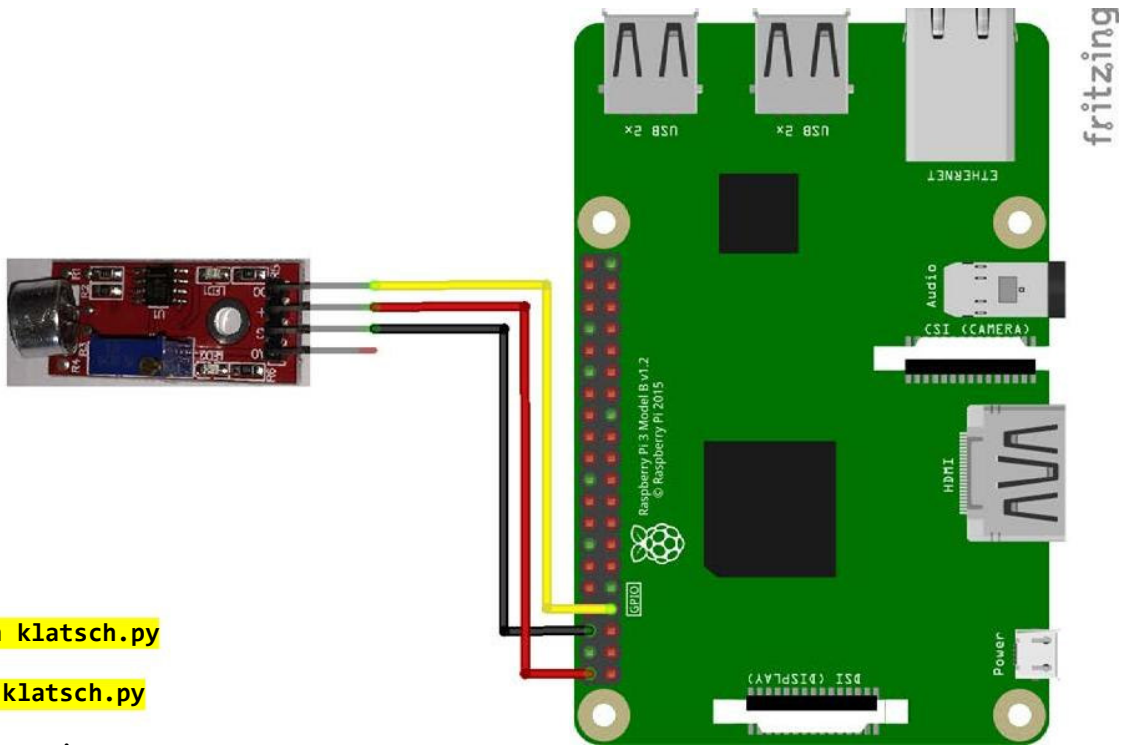
```
def lage(channel):  
    print('Lageaenderung erkannt')  
try:  
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING, callback=lage)  
    while True:  
        time.sleep(100)  
except KeyboardInterrupt:  
    print "Programm wurde  
beendet." GPIO.cleanup()
```

```
python lage.py
```

```
pi@raspberrypi3p:~$ python lage.py  
Lageaenderung erkannt  
Lageaenderung erkannt  
Lageaenderung erkannt
```

2.10 Noise sensor

The noise sensor gives a signal when a loud sound above a certain threshold has been detected.



```
cd ~
```

```
touch klatsch.py
```

```
nano klatsch.py
```

```
import RPi.GPIO as
GPIO import time

SENSOR_PIN = 4

GPIO.setmode(GPIO.BCM)
GPIO.setup(SENSOR_PIN, GPIO.IN)

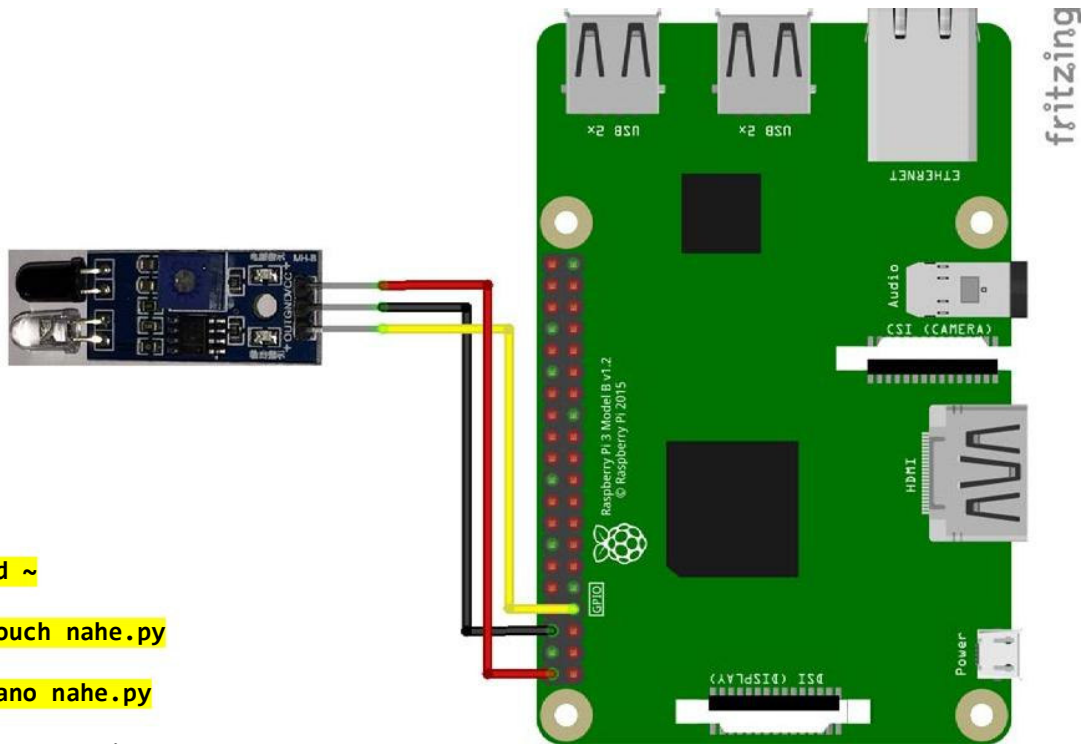
def klatsch(channel):
    print('klatschen erkannt')
try:
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING,
        callback=klatsch) while True:
        time.sleep(100)
except KeyboardInterrupt:
    print "Programm wurde
beendet." GPIO.cleanup()
```

```
python klatsch.py
```

```
pi@raspberrypi3p:~$ python klatsch.py
klatschen erkannt
klatschen erkannt
klatschen erkannt
klatschen erkannt
```

2.11 Proximity sensor

The proximity sensor emits an IR signal and when the IR receiving diode detects a reflection, the digital output is switched on. The distance can be adjusted with the potentiometer.



```
cd ~
```

```
touch nahe.py
```

```
nano nahe.py
```

```
import RPi.GPIO as  
GPIO import time
```

```
SENSOR_PIN = 4
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setup(SENSOR_PIN, GPIO.IN)
```

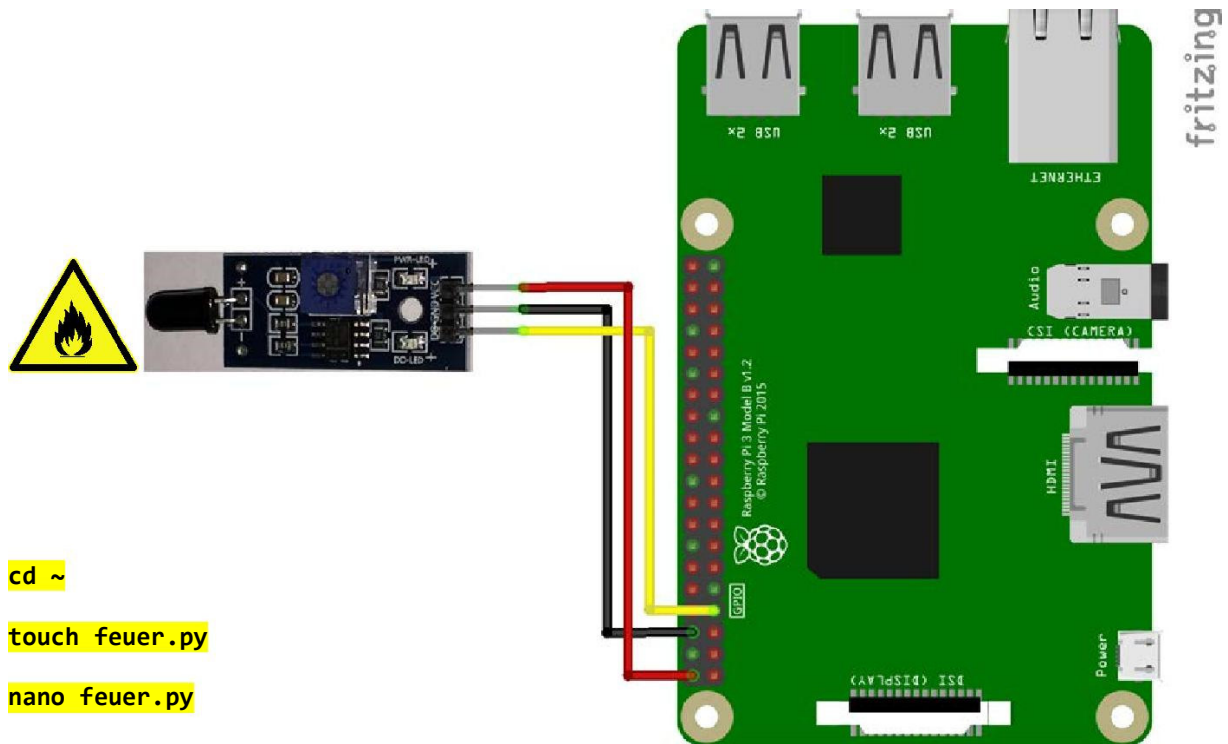
```
def nahe(channel):  
    print('Ein Gegenstand ist nahe dem Sensor')  
try:  
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING, callback=nahe)  
    while True:  
        time.sleep(100)  
except KeyboardInterrupt:  
    print "Programm wurde  
beendet." GPIO.cleanup()
```

```
python nahe.py
```

```
pi@raspberrypi3p:~ $ python nahe.py  
Ein Gegenstand ist nahe dem Sensor  
Ein Gegenstand ist nahe dem Sensor  
Ein Gegenstand ist nahe dem Sensor  
Ein Gegenstand ist nahe dem Sensor
```


2.12 Flame sensor

The flame sensor detects fire. If a flame of a fire (lighter, etc.) is detected, then the output is switched on.



```
cd ~
```

```
touch feuer.py
```

```
nano feuer.py
```

```
import RPi.GPIO as
GPIO import time

SENSOR_PIN = 4

GPIO.setmode(GPIO.BCM)
GPIO.setup(SENSOR_PIN, GPIO.IN)

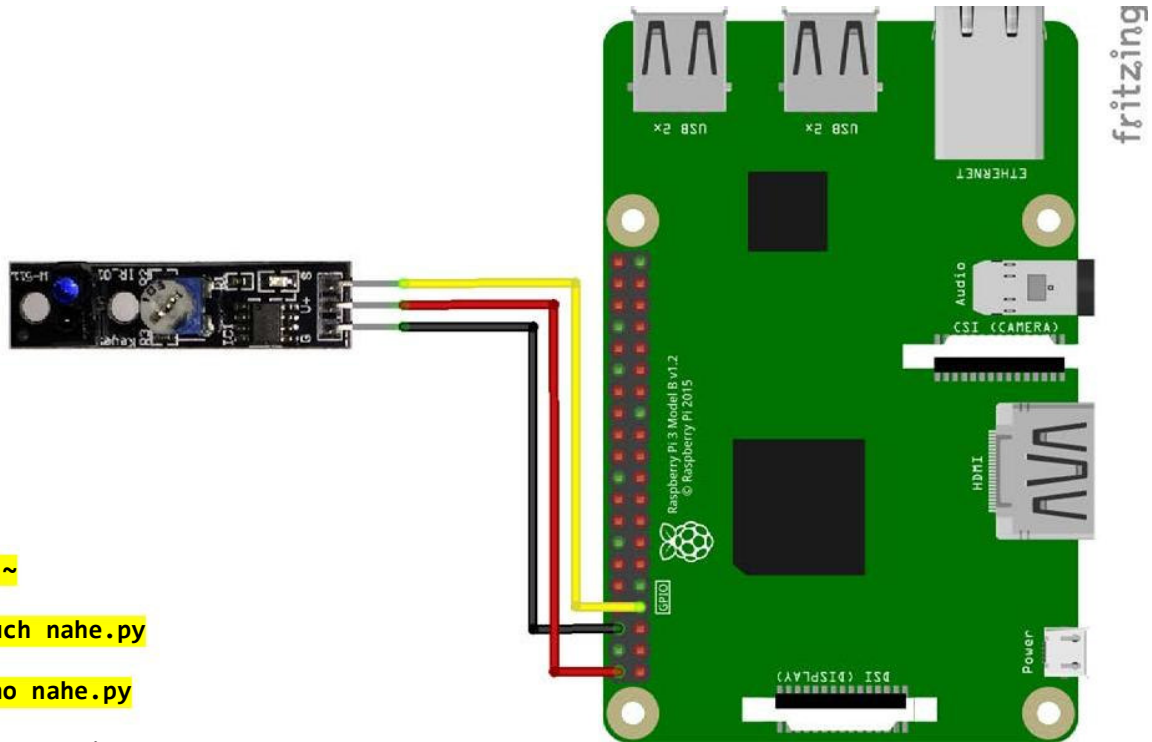
def feuer(channel):
    print('Es brennt eine Flamme!')
try:
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING,
        callback=feuer) while True:
        time.sleep(100)
except KeyboardInterrupt:
    print "Programm wurde
beendet." GPIO.cleanup()
```

```
python feuer.py
```

```
pi@raspberrypi3p:~ $ python feuer.py
Es brennt eine Flamme!
Es brennt eine Flamme!
```


2.13 Proximity sensor 2

The proximity sensor emits an IR signal and when the IR receiving diode detects a reflection, the digital output is switched on. The distance can be adjusted with the potentiometer.



```
cd ~
```

```
touch nahe.py
```

```
nano nahe.py
```

```
import RPi.GPIO as  
GPIO import time
```

```
SENSOR_PIN = 4
```

```
GPIO.setmode(GPIO.BCM)  
GPIO.setup(SENSOR_PIN, GPIO.IN)
```

```
def nahe(channel):  
    print('Ein Gegenstand ist nahe dem Sensor')  
try:  
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING, callback=nahe)  
    while True:  
        time.sleep(100)  
except KeyboardInterrupt:  
    print "Programm wurde beendet."  
GPIO.cleanup()
```

```
python nahe.py
```

```
pi@raspberrypi3p:~ $ python nahe.py  
Ein Gegenstand ist nahe dem Sensor  
Ein Gegenstand ist nahe dem Sensor  
Ein Gegenstand ist nahe dem Sensor  
Ein Gegenstand ist nahe dem Sensor
```

2.14 Ultrasonic sensor HC-SR04

With the ultrasonic sensor, the distance between the sensor and an object can be measured by the means of an echo.

The measuring process is commenced with a flank at the trigger input. The previous high signal must be at least 10µs.

Subsequently, a high signal will be present at the echo output for the duration of the echo. We now have to measure this time of the high signal and convert it into distance.

The conversion works on the basis of physics. The sound returns about 330 meters in one second. This value depends on the temperature. The propagation velocity can be calculated. At room temperature 20 °C, this is calculated as follows:

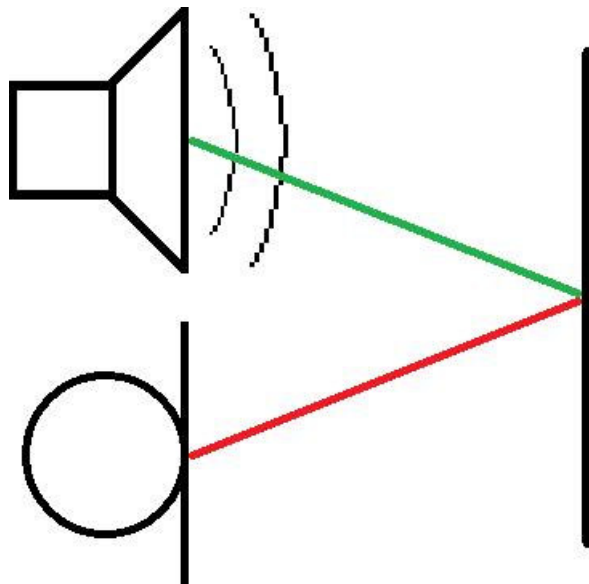
$$\text{Distance} = 331,5\text{m} + (0,6 * \text{Temperature}) = 331,5\text{m/s} + (0,6 * 20) = 343,5 \text{ m/s}$$

At room temperature, the speed of sound is thus:

$$343,5 \text{ m/s} \Rightarrow 0,03435\text{cm}/\mu\text{s}$$

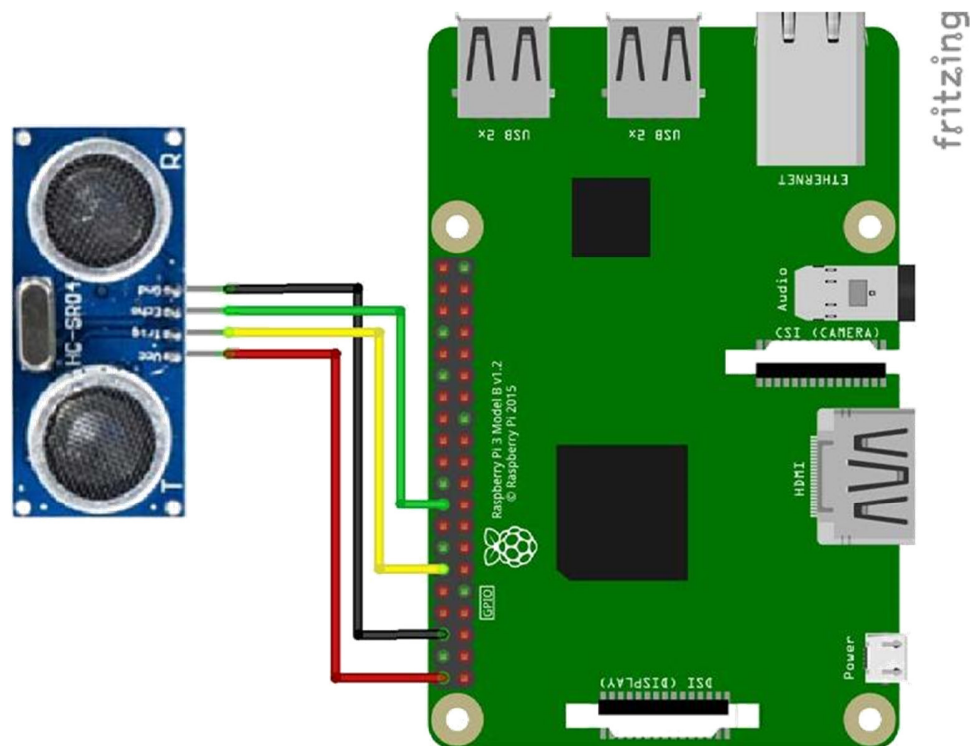
Now we can divide the duration of the sound by 2 because the sound is sent out and then reflected again, the path must be covered twice. But we only want the distance, not the path the sound needs.

$$\text{Distance} = (\text{Time} / 2) * \text{Speed of sound (20°)}$$



In addition, you have to provide a free from interference measurement, so interrupts are disabled during the measurement.

Az-Delivery



If you now execute the program from the next page you get the following information.

```
pi@raspberrypi3p:~ $ python us.py
Entfernung = 104.2 cm
Entfernung = 106.4 cm
Entfernung = 106.5 cm
Entfernung = 16.9 cm
Entfernung = 15.6 cm
Entfernung = 16.2 cm
Entfernung = 14.0 cm
Entfernung = 104.1 cm
Entfernung = 108.2 cm
```

Every second, a new measuring process is commenced and given.

Az-Delivery

```
cd ~
```

```
touch us.py
```

```
nano us.py
```

```
import RPi.GPIO as
GPIO import time

GPIO.setmode(GPIO.BCM)

GPIO_TRIGGER = 18
GPIO_ECHO = 24

GPIO.setup(GPIO_TRIGGER, GPIO.OUT)
GPIO.setup(GPIO_ECHO, GPIO.IN)

def distanz():
    GPIO.output(GPIO_TRIGGER, True)
    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)
    StartZeit = time.time()
    StopZeit = time.time()
    while GPIO.input(GPIO_ECHO) ==
        0: StartZeit = time.time()
    while GPIO.input(GPIO_ECHO) ==
        1: StopZeit = time.time()
    TimeElapsed = StopZeit - StartZeit
    distanz = (TimeElapsed * 34300) / 2
    return distanz

if __name__ ==
    '__main__': try:
        while True:
            abstand = distanz()
            print ("Entfernung = %.1f cm" % abstand)
            time.sleep(1)

    except KeyboardInterrupt:
        print("Messung gestoppt")
        GPIO.cleanup()
```

```
python us.py
```

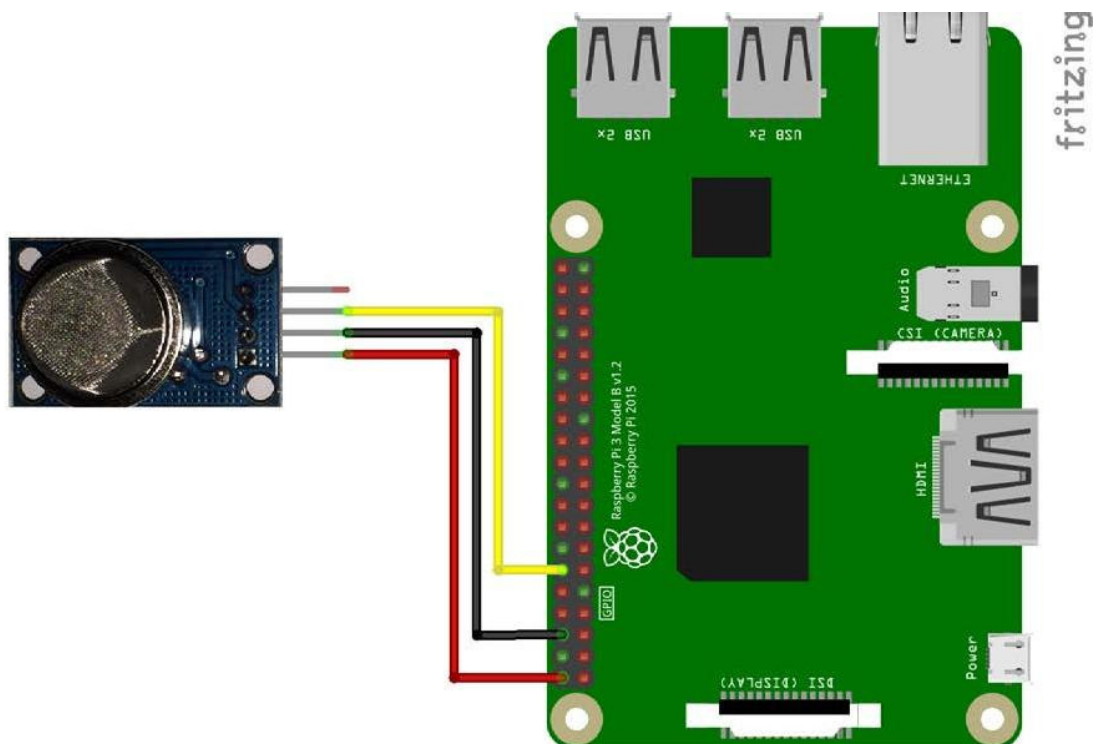
2.15 MQ-2 Gas sensor

The gas sensor can be used to measure the concentration of LPG, i-butane, propane, methane, alcohol, hydrogen and smoke in the air. The MQ-2 measures a gas concentration of 100 to 10000ppm and is ideal for detecting a gas leak or as a gas alarm.

The MQ-2 sensor uses a small heating element with an electronic-chemical sensor. They are sensitive to various gases and are only suitable for indoor use. ATTENTION: The heating element warms up the sensor!!

The sensor has a high sensitivity and fast response time but takes a few minutes to issue accurate readings, as the sensor has to first heat up.

Unfortunately, the Raspberry Pi has no analog input, so we can only monitor one threshold.



Az-Delivery

```
cd ~
```

```
touch gas.py
```

```
nano gas.py
```

```
import RPi.GPIO as
GPIO import time

SENSOR_PIN = 4

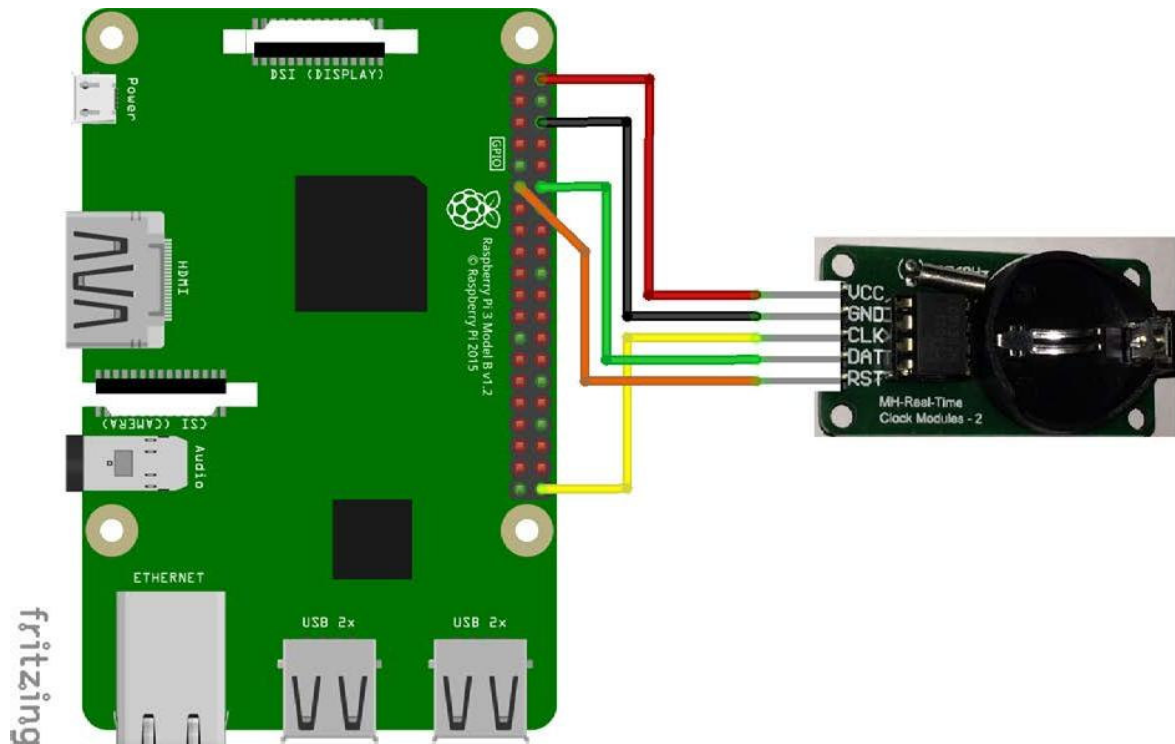
GPIO.setmode(GPIO.BCM)
GPIO.setup(SENSOR_PIN, GPIO.IN)

def gas(channel):
    print('Die Gaskonzentration ist zu hoch!')
try:
    GPIO.add_event_detect(SENSOR_PIN , GPIO.RISING,
        callback=gas) while True:
        time.sleep(100)
except KeyboardInterrupt:
    print "Programm wurde
beendet." GPIO.cleanup()
```

```
python gas.py
```

2.16 Real-time clock

With the real-time clock, the Raspberry Pi, even without Internet, gives the correct time after a reboot because the clock keeps running in the RTC module. For the module, you need a CR2032 battery!



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

```
wget https://www.dropbox.com/s/6ugi82e00a7x12j/rtc.c
```

```
sudo cc rtc.c -o rtc
```

```
sudo chmod +x rtc
```

```
sudo ./rtc
```

If the time of your Raspberry Pi is correct, we can write it directly to the RTC:

```
sudo ./rtc `date +"%Y%m%d%H%M%S"``
```

Alternatively, we can also set it manually:

```
sudo ./rtc YYYYMMDDHHMMSS
```

```
20180519164500    >>    19. Mai 2018 16:45:00
```

With the command `sudo ./rtc`, we can read the current time from the RTC module and also automatically correct the system time.

You made it! You can now actualize many new projects with the Raspberry Pi!

Now you can experiment.

And for more hardware, our online store is always at your disposal:

<https://az-delivery.de>

Enjoy!
Imprint

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