# JOY-PI

Experimentier- und Education-Koffer

JOY-IT

## 1. INHALTSVERZEICHNIS



Die Anmeldedaten sind:
Username : **pi**
Passwort : **12345**

## 2. ALLGEMEINE INFORMATIONEN

Sehr geehrter Kunde,

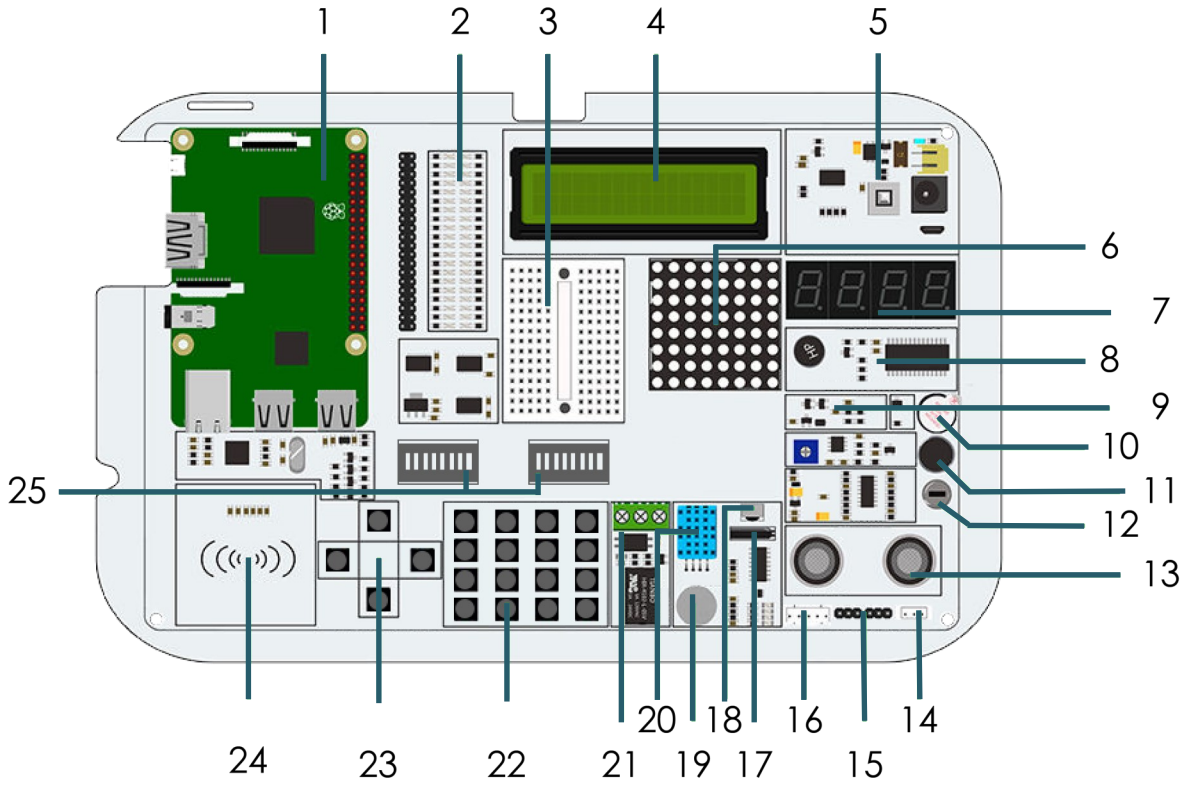
vielen Dank, dass Sie sich für unser Produkt entschieden haben.
Im Folgenden zeigen wir Ihnen, was bei der Inbetriebnahme und der Verwendung zu beachten ist. Sollten Sie während der Verwendung unerwartet auf Probleme stoßen, so können Sie uns selbstverständlich gerne kontaktieren.

Die folgenden Lektionen sind so konzipiert, dass Sie, unabhängig davon wie viel Vorwissen Sie bereits haben, alle Lektionen ohne Probleme ausführen können. Für die verschiedenen Lektionen müssen Sie Beispieldateien herunterladen und auf dem Joy-Pi ausführen. Wie Sie dies durchführen, können Sie ebenfalls dieser Anleitung entnehmen.

Doch diese Tutorials sind nur der Anfang, denn Ihrer Kreativität sind keine Grenzen gesetzt. Sie können unseren Joy-Pi für die verschiedensten Projekte verwenden und auch Projekte erstellen, wo der Joy-Pi im Fokus steht.

Wir freuen uns darauf zu sehen, was Sie mit unserem Joy-Pi machen werden.

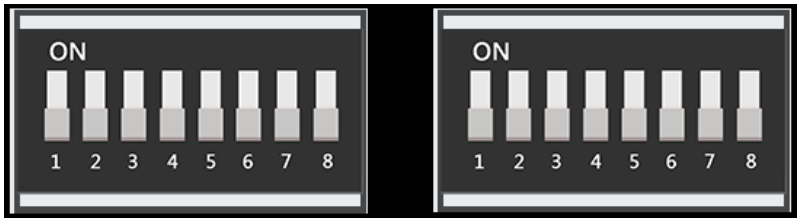
## 3. DETAILS



| | |
|---|---|
| 1 | Raspberry Pi |
| 2 | GPIO LED Anzeige |
| 3 | Breadboard |
| 4 | 16 x 2 LCD Modul (MCP23008) |
| 5 | Stromversorgung |
| 6 | 8 x 8 LED Matrix (MAX7219) |
| 7 | 7 Segment LED Anzeige (HT16K33) |
| 8 | Vibrationsmodul |
| 9 | Lichtsensor (BH1750) |
| 10 | Buzzer |
| 11 | Schallsensor |
| 12 | Bewegungssensor (LH1778) |
| 13 | Ultraschallabstandssensor |
| 14 / 15 | Servo - Schnittstellen |
| 16 | Schrittmotor - Schnittstelle |
| 17 | Neigungssensor (SW-200D) |
| 18 | Infrarotsensor |
| 19 | Berührungssensor |
| 20 | DHT11 Temperatur - und Luftfeuchtigkeitssensor |
| 21 | Relais |
| 22 | Taster - Matrix |
| 23 | Unabhängige Taster |
| 24 | RFID - Modul (MFRC522) |
| 25 | Schalter |
| 26 | Lüfteranschluss |
| 27 | Stromversorgung microUSB |

# 4. MODULWECHSEL UND VERWENDUNG DER GPIOS

## 4.1 Modulwechsel




Auf der Joy-Pi Platine befinden sich zwei Schaltereinheiten à 8 Schalter. Die Schalter ermöglichen es, zwischen verschiedenen Sensoren und Modulen zu wechseln. Da der Raspberry Pi nur eine begrenzte Anzahl an GPIO - Pins hat, werden diese Schalter benötigt um mehr Sensoren und Module zu verwenden zu können als GPIO - Pins vorhanden sind.

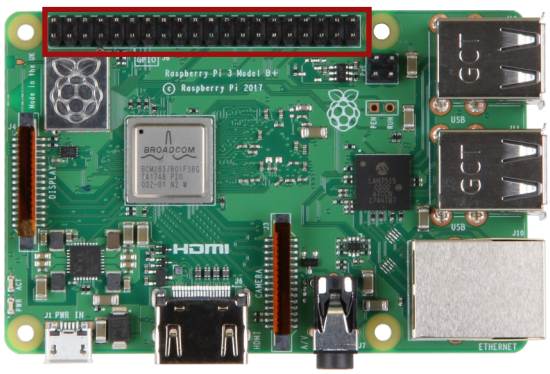
Die Verwendung dieser Schalter ist ziemlich einfach und wird in einigen der folgenden Lektionen benötigt.

In der Tabelle sehen Sie, welcher Schalter welchen Sensor bzw. welches Modul schaltet,

| Sensoren / Module | Schalteinheit | Schalter |
|---|---|---|
| Taster - Matrix | Links | 1 - 8 |
| unabhängige Taster | Links | 5 - 8 |
| Vibrationsmodul | Rechts | 1 |
| Neigungssensor | Rechts | 2 |
| Schrittmotor | Rechts | 3, 4, 5, 6 |
| Servomotor | Rechts | 7, 8 |

## 4.2 Verwendung der GPIOs

Im Folgenden werden wir Ihnen genauer erläutern, was GPIO-Pins sind, wie diese funktionieren und wie diese gesteuert werden.



GPIO steht für: *General - purpose input / output (Universal Eingang / Ausgang)*.
GPIO - Pins haben keine festgelegte Funktion. Es kann konfiguriert werden, ob die GPIO - Pins als digitaler Eingang oder als digitaler Ausgang dienen.

**Beispiel Eingangs - Pin:** Taster
Wenn der Taster gedrückt wird, gelangt das Signal zur weiteren Verarbeitung über den Eingang - Pin zum Raspberry Pi.

**Beispiel Ausgangs - Pin:** Buzzer
Über den Ausgangs - Pin wird ein Signal in Richtung Buzzer gesendet, um diesen zu steuern.

Wenn Sie den geöffneten Joy - Pi von vorne betrachten, befinden sich die GPIO - Pins auf der rechten Seite des Raspberry Pis.

Es gibt 2 mögliche Raspberry Pi GPIO Schemata: **GPIO - BOARD** und **GPIO - BCM**.
Die GPIO - BOARD Option gibt an, dass Sie sich auf die Pins anhand der Nummer des Pins beziehen. Das heißt es werden die unten voranstehenden Pin - Nummern verwendet.

Die Option GPIO - BCM bedeutet, dass Sie sich auf die Pins des *Broadcom SOC Channel* beziehen. Dies sind die Zahlen nach GPIO:

| | | | | |
|---|---|---|---|---|
| 1 | 3.3 V DC | | 2 | 5 V DC |
| 3 | GPIO 2 (SDA1, I2C) | | 4 | 5 V DC |
| 5 | GPIO 3 (SCL1, I2C) | | 6 | Ground |
| 7 | GPIO 4 | | 8 | GPIO 14 (TXD0) |
| 9 | Ground | | 10 | GPIO 15 (RXD0) |
| 11 | GPIO 17 | | 12 | GPIO 18 |
| 13 | GPIO 27 | | 14 | Ground |
| 15 | GPIO 22 | | 16 | GPIO 23 |
| 17 | 3.3 V | | 18 | GPIO 24 |
| 19 | GPIO 10 (SPI, MOSI) | | 20 | Ground |
| 21 | GPIO 9 (SPI, MISO) | | 22 | GPIO 25 |
| 23 | GPIO 11 (SPI, CLK) | | 24 | GPIO 8 (SPI) |
| 25 | Ground | | 26 | GPIO 7 (SPI) |
| 27 | ID_SD (I2C, EEPROM) | | 28 | ID_SC |
| 29 | GPIO 5 | | 30 | Ground |
| 31 | GPIO 6 | | 32 | GPIO 12 |
| 33 | GPIO 13 | | 34 | Ground |
| 35 | GPIO 19 | | 36 | GPIO 16 |
| 37 | GPIO 26 | | 38 | GPIO 20 |
| 39 | Ground | | 40 | GPIO 21 |

| GPIO - BOARD | Sensoren und Module |
|---|---|
| 1 | 3.3 V |
| 2 | 5.0 V |
| 3 | I2C, SDA1 (Lichtsensor, LCD Display, 7 Segment Display) |
| 4 | 5.0 V |
| 5 | I2C. SCL1 (Lichtsensor, LCD Display, 7 Segment Display) |
| 6 | Ground |
| 7 | DHT11 Sensor |
| 8 | TXD0 |
| 9 | Ground |
| 10 | RXD0 |
| 11 | Touchsensor |
| 12 | Buzzer |
| 13 | Schaltflächenmatrix (ROW1), Vibrationsmotor |
| 14 | Ground |
| 15 | Schaltflächenmatrix (ROW2), Neigungssensor |
| 16 | Bewegungssensor |
| 17 | 3.3 V |
| 18 | Schallsensor |
| 19 | SPI |
| 20 | Ground |
| 21 | SPI |
| 22 | Servo2, Schaltflächenmatrix (COL1), Left Button |
| 23 | SPI |
| 24 | RFID - Modul |
| 25 | Ground |
| 26 | LED - Matrix |
| 27 | IS_SD (I2C, EEPROM (Electrically Erasable Programmable Read - only Memory)) |
| 28 | ID_SC |
| 29 | Schrittmotor (STEP1), Schaltflächenmatrix (ROW3) |
| 30 | Ground |
| 31 | Schrittmotor (STEP2), Schaltflächenmatrix (ROW4) |
| 32 | Ultraschallsensor (Echo) |
| 33 | Schrittmotor (STEP3), Schaltflächenmatrix (COL4), |
| 34 | Ground |
| 35 | Schrittmotor (STEP4), Schaltflächenmatrix (COL3), |
| 36 | Ultraschallsensor (TRIG) |
| 37 | Servo1, Schaltflächenmatrix (COL2), Up Button |
| 38 | Infrarotsensor |
| 39 | Ground |
| 40 | Relais |

In unseren Beispiel verwenden wir die Programmiersprache *Python*, um die GPIO-Pins zu steuern. In Python gibt es eine Bibliothek namens *RPi.GPIO.* Dies ist eine Bibliothek, die dabei hilft, die Pins mit Python zu steuern.

Das folgende Beispiel und die Kommentare im Code sollen Ihnen helfen das Programm zu verstehen.

Als Erstes müssen die benötigten Bibliothek mit Hilfe des **import** Befehls importiert werden. Die Variablen **TOUCH** und **BUZZER** verweisen auf die Pins des Touchsensors und des Buzzers. Im Anschluss wird mit *GPIO.setmode(GPIO.BOARD)* das verwendete GPIO Schema definiert. Als Nächstes werden die zuvor festgelegten Variablen mit Hilfe des Befehls *GPIO.setup()* als Ein- bzw. Ausgang konfiguriert. Pin 11 (TOUCH) wird als Eingang und Pin 12 (BUZZER) wird als Ausgang festgelegt.
Die Funktion **main** fragt ab, ob es eine Berührung des Touchsensors gibt. Ist dies der Fall wird die Funktion **do_smt** ausgeführt.
Die Funktion **do_smt** druckt den Text *Touch wurde erkannt*, setzt dann den Buzzer-Pin ***HIGH*** und eine Sekunde später wieder ***LOW*** (der Buzzer summt eine Sekunde).

```
import RPi.GPIO as GPIO
import time #importieren der Bibliotheken
import signal
TOUCH = 11 #Deklaration der Pins
BUZZER = 12
def setup_gpio(): #Definition der Ein- und Ausgaenge
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(TOUCH, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.setup(BUZZER, GPIO.OUT)
def do_smt(channel): #Funktion zur Ausgabe das Touch erkannt wurde
    print("Touch wurde erkannt") #und Ausgabe das Touch erkannt wurde
    GPIO.output(BUZZER, GPIO.HIGH) #Signalausgabe
    time.sleep (1) #1 Sekunde warten
    GPIO.output(BUZZER, GPIO.LOW) #Signalausgabe stoppen
def main():
    setup_gpio()
    try: #Pruefung ob ein Touch erkannt wurde
        GPIO.add_event_detect(TOUCH,GPIO.FALLING,callback=do_smt,bouncetime=200)
    except KeyboardInterrupt: #STRG + C beendet das Programm
        pass
    finally:
        GPIO.cleanup()
if __name__=='__main__':
    main()
```

Um noch mehr über den Zweck und die Verwendung von GPIOs zu erfahren, empfehlen wir Ihnen sich die offizielle Dokumentation der Raspberry Pi Foundation zum Thema GPIO anzuschauen.

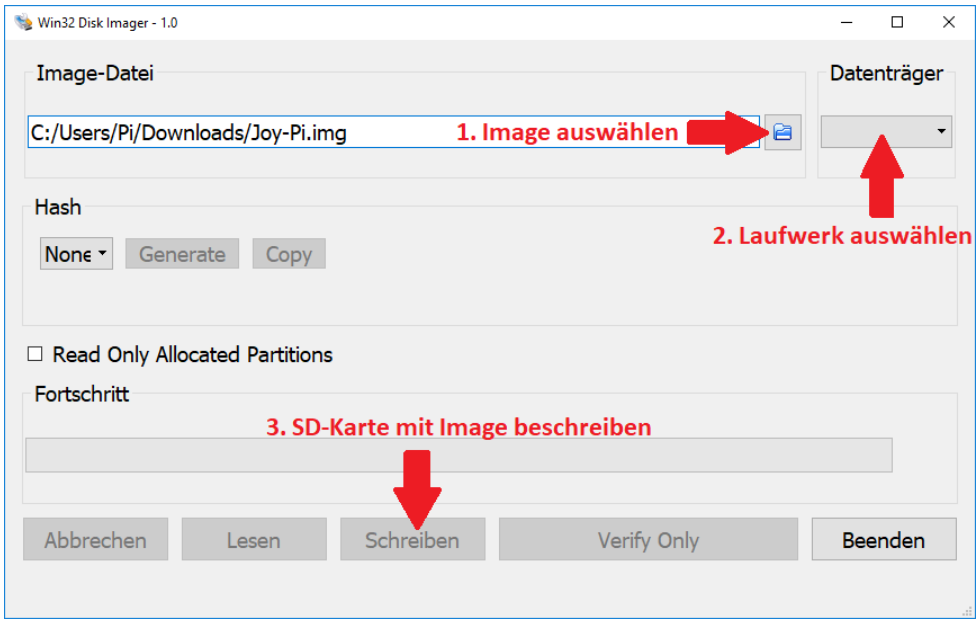
https://www.raspberrypi.org/documentation/usage/gpio/

## 4.3 Softwareinstallation für den Joy-Pi

Auf der mitgelieferten microSD-Karte ist schon ein vorinstalliertes Betriebssystem aufgespielt. Wenn Sie die Karte neu beschreiben möchten, können Sie dies wie folgt durchführen:

Als erstes sollten Sie die aktuellste Image-Datei für den Joy-Pi von unserer Website www.joy-pi.net herunterladen.

1. Laden Sie die Image-Datei (.zip Format) auf Ihren PC. Nach dem Entpacken der Datei erhalten Sie eine Datei mit der Endung .img.
2. Schließen Sie eine microSD-Karte an Ihren PC an und formatieren Sie diese mit dem Programm SD Formatter. Ein microSD-Kartenlesegerät ist im Lieferumfang enthalten.
3. Starten Sie das Programm Win32-Disk-Imager und wählen Sie
   ① die heruntergeladene Image-Datei aus.
   ② das zu beschreibende Laufwerk aus.
4. Nun ist die Karte mit dem Betriebssystem beschrieben und Sie können diese in den microSD-Kartenslot des Raspberry Pi stecken.



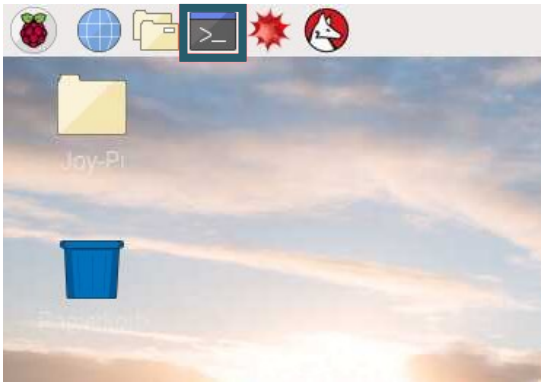

5. Zum Schluss müssen Sie das Image noch auf die Größe Ihrer SD-Karte anpassen. Starten Sie dafür Ihren Raspberry Pi, öffnen Sie das Terminal und geben **`sudo raspi-config`** ein. Gehen Sie nun auf ***Advanced Options*** und dann auf ***Expand Filesystem***. Nach einem Neustart wird die Image-Größe auf Ihre SD-Karte angepasst.

# 5. VERWENDUNG VON PYTHON UND LINUX

Dieser Schritt ist optional, macht es jedoch einfacher Skripte auszuführen, ohne sie einzeln erstellen zu müssen. Auf der mitgelieferten microSD-Karte befinden sich Skripte auf dem Desktop.

Die in der Anleitung verwendeten Skripte können direkt in einem Paket heruntergeladen werden.
Folgen Sie dazu einfach den folgenden Anweisungen:

1. Öffnen Sie das **Terminal**. Dies benutzen wir, um die meisten unserer Python Skripte auszuführen und Erweiterungen und Skripte herunterzuladen.



2. Nachdem wir das Terminal erfolgreich geöffnet haben, müssen wir bei Bedarf das Skriptarchiv mit dem folgenden Befehl auf den Desktop herunterladen (gegebenenfalls auf dem Image enthalten):

```
cd Desktop/
wget https://www.joy-it.net/files/files/Produkte/RB-JoyPi/Joy-Pi.zip
```

3. Drücken Sie **Enter** auf Ihrer Tastatur. Nun muss das Archiv nur noch entpackt werden:

```
unzip Joy-Pi.zip
```

4. Drücken Sie **Enter** auf Ihrer Tastatur und warten Sie bis der Vorgang abgeschlossen ist.
5. Mit dem Befehl **cd** wechseln Sie in das richtige Verzeichnis damit die Skripte verwendet werden können, welche sich dort befinden:

```
cd Joy-Pi/Python3
```

**Achtung!** Jedes Mal, wenn Sie Ihren Joy-Pi ausschalten, müssen Sie die Schritte des Verzeichniswechsels wiederholen.

Die Anmeldedaten sind:
Username : **pi**
Passwort : **12345**

## Ausführen von Python Skripts

Nachdem die Skripte erfolgreich von unserer Website heruntergeladen wurden, möchten wir es jetzt ausführen. Öffnen Sie erneut das Terminal und befolgen Sie die folgende Anweisung, um das Skript auszuführen:

1. Geben Sie den Befehl **`sudo python3 <script name>`** ein, um ein Python-Skript auszuführen, wie zum Beispiel:

```
sudo python3 buzzer.py
```

Der Befehl setzt sich aus 3 Teilen zusammen. Durch den Befehl **`sudo`** wird der folgende Teil der Befehlszeile mit root-Berechtigungen (Admin-Berechtigungen ausgeführt. **`python3`** ist der Befehl der gleichnamigen Programmiersprache, in der die Skripte geschrieben sind. Am Ende steht der Name des Skripts. Hierbei ist zu beachten, dass man sich entweder in dem Ordner befindet, in dem das jeweilige Skript gespeichert ist oder den Pfad (z.B. **`~/Joy-Pi/buzzer.py`**) angibt.

## Lektion 1 : Verwenden des Buzzers für Warntöne

In der vorherigen Erklärung haben wir gelernt, wie man den GPIO-Pin sowohl als Ausgabe als auch als Eingabe verwendet. Um dies nun zu testen, gehen wir mit einem realen Beispiel voran und wenden unser Wissen aus der vorherigen Lektionen an. Das Modul, welches wir verwenden werden, ist der Buzzer.

Wir werden den GPIO-Ausgang verwenden, um ein Signal an den Buzzer zu senden und die Schaltung zu schließen, um ein lautes Summen zu Erzeugen. Dann werden wir ein weiteres Signal senden, um es auszuschalten.

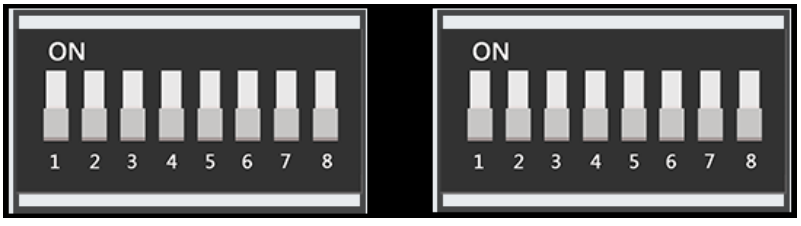

Der Buzzer befindet sich auf der rechten Seite des Joy-Pi-Boards und ist durch das laute Geräusch, dass er bei Aktivierung macht leicht zu erkennen. Wenn Sie Ihren Raspberry Pi zum ersten Mal verwenden, ist der Buzzer möglicherweise mit einem Schutzaufkleber versehen. Stellen Sie sicher, dass dieser Aufkleber vor Gebrauch des Buzzers entfernt wurde.

Genau wie im vorherigen Beispiel haben wir ein spezielles Skript mit detaillierten Kommentaren vorbereitet, die erklären werden, wie der gesamte Buzzer-Prozess funktioniert und wie wir den Buzzer mit den GPIOs steuern können.

Zuerst importieren wir die **RPi.GPIO-Bibliothek** und die **time-Bibliothek**. Dann konfigurieren wir den Buzzer. An **Pin 12** richten wir den GPIO-Modus auf **GPIO BOARD** und den Pin als **OUTPUT** ein.

Wir geben ein Signal für 0,5 Sekunden aus und schalten dieses dann aus.

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und rechten Schalteinheit auf OFF stellen.



```
#!/usr/bin/python

import RPI.GPIO as GPIO   #importieren der benoetigten Bibliotheken
import time

buzzer_pin = 12           #buzzer_pin wird definiert

GPIO.setmode(GPIO.BOARD)
GPIO.setup(buzzer_pin, GPIO.OUT)

GPIO.output(buzzer_pin, GPIO.HIGH)     #gebe Geraeusch aus

time.sleep(0.5)           #warte eine halbe Sekunde

GPIO.output(buzzer_pin, GPIO.LOW) #stoppe Geraeuschausgabe

GPIO.cleanup()
```

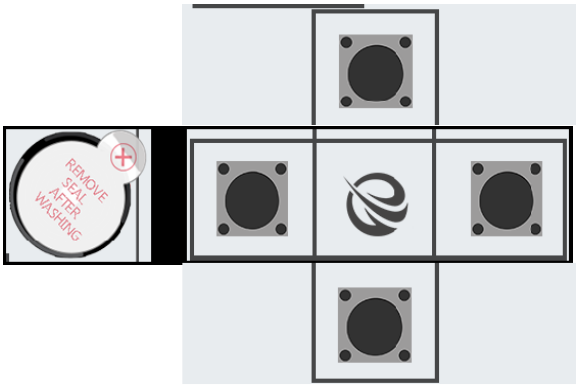
Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 buzzer.py
```

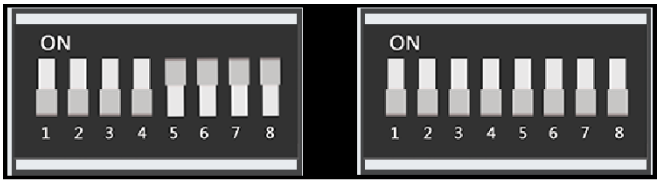
## Lektion 2 : Buzzer mit Taster steuern

Nachdem Sie nun wissen, wie man den Buzzer ein- und ausschaltet, ist es Zeit, die Ding ein wenig aufregender zu gestalten. In dieser Lektion wird ein Taster mit dem Buzzer kombiniert, sodass der Buzzer nur durch Drücken des Tasters eingeschaltet wird.

Dieses Mal werden 2 GPIO-Setups verwendet. Einer wird der GPIO.INPUT sein, der den Taster als Eingabemöglichkeit einstellt, ein anderer wird der GPIO.OUTPUT sein, der ein Signal an den Buzzer sendet um ein Geräusch auszugeben.




**Achtung!** Für dieses Beispiel müssen Sie zwischen den Modulen wechseln. Stellen Sie die Schalter Nummer 5, 6, 7 und 8 der linken Schalteinheit auf **ON**. Alle anderen Schalter sollten auf **OFF** stehen.



In diesem Beispiel wird der Obere der 4 Taster auf der unteren linken Seite verwendet. Theoretisch kann jedoch jeder der 4 Taster verwendet werden. Wenn Sie trotzdem einen anderen Taster verwenden möchten, müssen Sie die Pinbelegung dementsprechend ändern.

| | |
|---|---|
| GPIO37 | Oberer Taster |
| GPIO33 | Unterer Taster |
| GPIO22 | Linker Taster |
| GPIO35 | Rechter Taster |

Für diesen Teil des Tutorials müssen 2 GPIO-Einstellungen verwendet werden: eine Eingabe und eine Ausgabe. Der GPIO-Eingang wird verwendet um zu bestimmen, wann ein Taster gedrückt wurde und die GPIO-Ausgabe wird verwendet um den Buzzer zu aktivieren, sobald dieser Taster gedrückt wird.

Wenn Sie den Taster auf Ihrem Joy-Pi drücken, ertönt der Buzzer! Lassen Sie den Taster los und der Buzzer verstummt. Das Programm läuft solange bis **STRG + C** gedrückt wird.

Beispielcode:

```
#!/usr/bin/python

import RPI.GPIO as GPIO   #importieren der benoetigten Bibliotheken
import time

#definiere Pins
button_pin = 37
buzzer_pin = 12

#setze Board Modus zu GPIO.BOARD
GPIO.setmode(GPIO.BOARD)

#lege button_pin als Eingang und buzzer_pin als Ausgang fest
GPIO.setup(button_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(buzzer_pin, GPIO.OUT)

try:
    while True:
        #ueberpruefe ob Knopf gedrueckt wird
        if (GPIO.input(button_pin) == 0):
            #Buzzer ein
            GPIO.output(buzzer_pin, GPIO.LOW)
        else:
            #Buzzer aus
            GPIO.output(buzzer_pin, GPIO.LOW)

except KeyboardInterrupt:
    GPIO.cleanup()
```

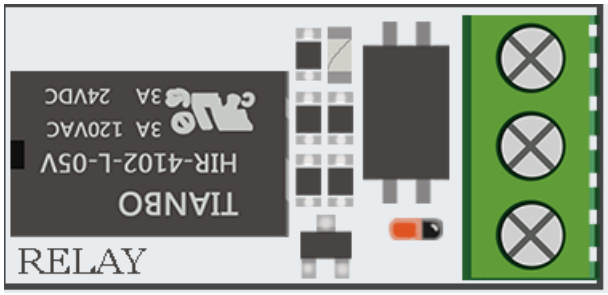
Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 button_buzzer.py
```

## Lektion 3 : Wie ein Relais funktioniert und wie man es steuert

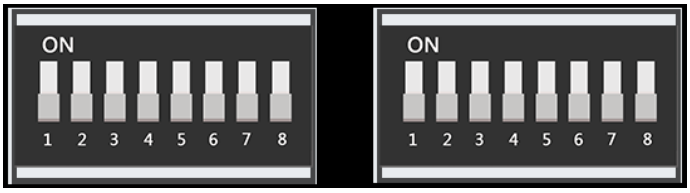
Nachdem wir jetzt alles Nötige über den Buzzer wissen, ist es Zeit für die nächste Lektion. Nun lernen wir, wie das Relais zu verwenden ist, welche Funktion das Relais hat und wie man es steuert.

Ein Relais ist vereinfacht ein Schalter, dem (unteranderem) mit Hilfe von GPIO-Pins ein- und ausschalten kann. Relais werden verwendet, um eine Schaltung durch ein separates Niederleistungssignal zu steuern oder wenn mehrere Schaltungen durch ein Signal gesteuert werden müssen. In unserem Beispiel zeigen wir, wie ein GPIO-Signal gesendet wird um das Relais zu schließen, um eine benutzerdefinierte Schaltung zu aktivieren und wie man ein weiteres Signal sendet, um das Relais zu öffnen und die Schaltung zu deaktivieren.




Das Relais befindet sich im mittleren, unteren Teil der Platine, direkt neben der Taster-Matrix. Es hat drei Anschlüsse, von denen wir 2 in diesem Beispiel benutzen werden. **NC** steht für *normally closed*, **NO** steht für *normally open* und **COM** steht für *common*. *Common* steht in dem Fall für die **gemeinsame** Masse.
Wenn ein Stromkreis an **NC** und **COM** angeschlossen wird, ist der Stromkreis geschlossen, wenn der Steuerstromkreis spannungslos ist (GPIO.LOW). Wird der Steuerstromkreis unter Spannung gesetzt (GPIO.HIGH), öffnet das Relais die Verbindung im Arbeitsstromkreis und der Stromfluss wird unterbrochen.
Bei der Nutzung von **NO** und **COM** verhält sich genau umgekehrt. Ist der Steuerstromkreis spannungslos (GPIO.LOW), ist das Relais geöffnet und der Arbeitsstromkreis unterbrochen. Wird der Steuerstromkreis mit Spannung versorgt (GPIO.HIGH), schließt das Relais den Arbeitsstromkreis und der Strom kann fließen.

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen




**Achtung!** Es ist sehr wichtig nicht zu versuchen Hochspannungsgeräte an das Relais anzuschließen (z.B. Tischlampe, Kaffeemaschine usw.). Dies könnte zu Stromschlägen und schweren Verletzungen führen.

```
#!/usr/bin/python

import RPI.GPIO as GPIO
import time

#definiere Relais Pin
relay_pin = 40

#Board Modus  GPIO.BOARD
GPIO.setmode(GPIO.BOARD)
#relay_pin als Ausgang
GPIO.setup(relay_pin, GPIO.OUT)

#Oeffne Relais
GPIO.output(relay_pin, GPIO.LOW)
#warte eine halbe Sekunde
time.sleep(0.5)
#schliesse Relais
GPIO.output(relay_pin, GPIO.HIGH)
GPIO.cleanup()
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 relay.py
```

## Lektion 4 : Senden eines Vibrationssignals

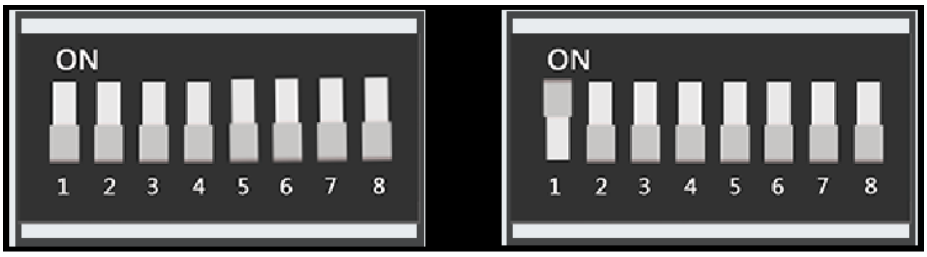
Haben Sie sich schon einmal gefragt, wie Ihr Telefon vibriert, wenn Sie jemand anruft oder wenn Sie eine Nachricht erhalten? Wir haben das gleiche Modul in unserem Joy-Pi verbaut und nun werden Sie lernen, wie man es verwendet.

Das Vibrationsmodul befindet sich auf der rechten Seite der LED-Matrix und unterhalb der Segment-LED. Wenn es eingeschaltet ist, ist es schwierig zu erkennen, woher die Vibration kommt, da es sich so anfühlt, als vibriere das ganze Joy-Pi Board.

Das Vibrationsmodul verwendet ein **GPIO.OUTPUT**-Signal, genau wie der Buzzer und andere Module zuvor. Durch Senden eines Ausgangssignals vibriert das Vibrationsmodul durch Stoppen des Signals mit **GPIO.LOW** hört die Vibration auf.

Mit verschiedenen ***time.sleep()*** Intervallen kann man die Dauer der Vibration regeln. Versuchen Sie es selbst und schauen Sie , wie Sie das Beispiel erweitern können.

**Achtung!** Für dieses Beispiel müssen Sie zwischen den Modulen wechseln. Stellen Sie Schalter Nummer 1 der rechten Schalteinheit auf **ON**. Alle anderen Schalter sollten auf **OFF** stehen.




Beispielcode:

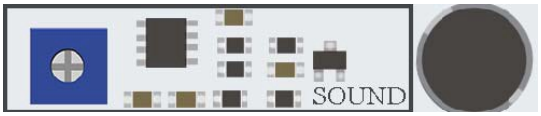
```
#!/usr/bin/python

import RPI.GPIO as GPIO
import time

#definiere des Vibrationspins
vibration_pin = 13

#setze Board Modus zu GPIO.BOARD
GPIO.setmode(GPIO.BOARD)

#lege Vibrationspin als Ausgang fest
GPIO.setup(vibration_pin, GPIO.OUT)

#schalte Vibration ein
GPIO.output(vibration_pin, GPIO.HIGH)
#warte eine Sekunde
time.sleep(1)
#schalte Vibration aus
GPIO.output(vibration_pin, GPIO.LOW)

GPIO.cleanup()
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 vibration.py
```

# Lektion 5 : Geräusche mit dem Schallsensor erkennen

In dieser Lektion lernen wir, wie man über den Schallsensor Eingaben tätigt, laute Geräusche erkennt und entsprechend reagiert. So können Sie Ihr eigenes Alarmsystem aufbauen, das laute Geräusche erkennt oder eine LED durch Klatschen einschalten!




Der aus 2 Teilen aufgebaute Schallsensor besteht aus einem blauen Potentiometer, dass für die Regulierung der Empfindlichkeit zuständig ist und dem Sensor selbst, der Geräusche erkennt. Der Schallsensor ist gut am blauen Potentiometer zu erkennen und der Sensor selbst befindet sich rechts unter dem Buzzer.

Mit Hilfe des Potentiometers können wir die Empfindlichkeit des Sensors regulieren.
Damit unser Skript funktioniert, müssen wir zuerst lernen, wie man die Empfindlichkeit steuert. Um die Empfindlichkeit zu regulieren, müssen Sie die kleine Schraube am Potentiometer mit einem Schraubendreher nach links oder rechts drehen. Wenn sie gegen den Uhrzeigersinn drehen, steigt die Empfindlichkeit und mit dem Uhrzeigersinn verringert sich diese. Der beste Weg die Empfindlichkeit zu testen ist das Skript auszuführen. Klatschen Sie in die Hände und schauen Sie, ob das Gerät ein Signal empfängt. Wenn kein Signal empfangen wird bedeutet dies, dass die Empfindlichkeit des Sensors nicht hoch genug eingestellt ist. Dies können Sie durch drehen des Potentiometers einfach beheben.

```
#!/usr/bin/python

import RPI.GPIO as GPIO
import time

#sound_pin wird definiert
sound_pin = 18

#GPIO mode wird auf GPIO.BOARD gesetzt
GPIO.setmode(GPIO.BOARD)

#sound_pin wird als Eingang festgelegt
GPIO.setup(sound_pin, GPIO.IN, pull_up_down=GPIO.PUD_UP)

try:
    while True:
        #ueberpruefe ob ein Geraeusch erkannt wird
        if(GPIO.input(sound_pin)==GPIO.LOW):
            print('Sound erkannt')
            time.sleep(0.1)
except KeyboardInterrupt:
    #STRG+C beendet das Programm
    GPIO.cleanup()
```

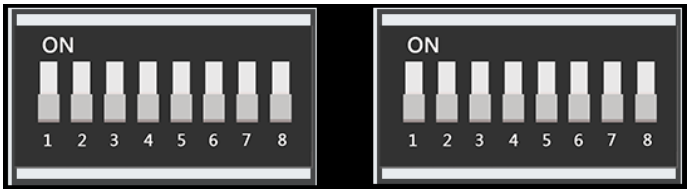
Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 sound.py
```

Wir definieren zuerst unseren Pin, **GPIO18**. Dann setzen wir eine **while-Schleife**, um dieses Skript dauerhaft laufen zu lassen. Wir prüfen, ob wir vom Schallsensor eine Eingabe erhalten haben, die anzeigt, dass laute Geräusche erkannt wurden und dann drucken wir *Sound Detected*.

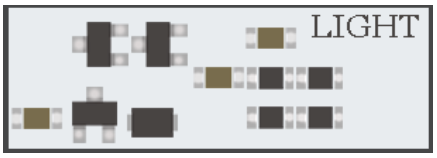
Wenn **STRG + C** gedrückt wird, wird das Programm beendet.

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen




## Lektion 6 : Erkennen der Helligkeit mit dem Lichtsensor

Der Lichtsensor ist einer unserer Lieblinge. Er ist in vielen Projekten und Situationen äußerst nützlich, zum Beispiel bei Lampen, die automatisch angehen, sobald es dunkel wird. Mit dem Lichtsensor kann man erkennen, wie hell die Moduloberfläche ist.



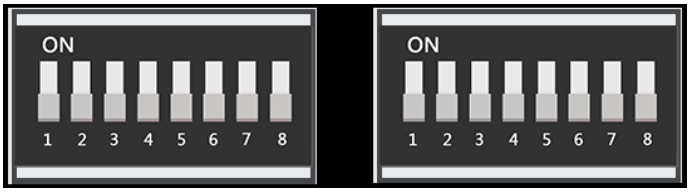

Der Lichtsensor ist schwer zu erkennen, da er aus sehr kleinen Teilen besteht. Der Sensor liegt links neben dem Buzzer. Wenn Sie ihn mit Ihrem Finger verdecken, sollte die Ausgabe des Lichtsensors nahe null gehen, da ihn kein Licht mehr erreichen kann.

Nun ist es Zeit, den Sensor zu testen und zu sehen, wie er funktioniert. Jedoch ist der Lichtsensor ein wenig anders, als andere Sensoren, da er mit I2C funktioniert und nicht mit den normalen GPIOs, wie Sie es in den Lektionen zuvor gelernt haben.

In diesem Skript wird eine Funktion verwendet um mit dem Lichtsensor zu kommunizieren und die gewünschte Ausgabe mit der Helligkeit zu erhalten. Je höher die ausgegebene Zahl ist, desto heller ist die Umgebung.

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author: Matt Hawkins
# Author's Git: https://bitbucket.org/MattHawkinsUK/
# Author's website: https://www.raspberrypi-spy.co.uk
import RPi.GPIO as GPIO
import smbus
import time

if(GPIO.RPI_REVISION == 1):
    bus = smbus.SMBus(0)
else:
    bus = smbus.SMBus(1)

class LightSensor():
    def __init__(self):
        # Definiere Konstante vom Datenblatt
        self.DEVICE = 0x5c # Standard I2C Geräteadresse
        self.POWER_DOWN = 0x00 # Kein aktiver zustand
        self.POWER_ON = 0x01 # Betriebsbereit
        self.RESET = 0x07 # Reset des Data registers
        # Starte Messungen ab 4 Lux.
        self.CONTINUOUS_LOW_RES_MODE = 0x13
        # Starte Messungen ab 1 Lux.
        self.CONTINUOUS_HIGH_RES_MODE_1 = 0x10
        # Starte Messungen ab 0.5 Lux.
        self.CONTINUOUS_HIGH_RES_MODE_2 = 0x11
        # Starte Messungen ab 1 Lux.
        # Nach Messung wird Gerät in einen inaktiven Zustand gesetzt.
        self.ONE_TIME_HIGH_RES_MODE_1 = 0x20
        # Starte Messungen ab 0.5 Lux.
        # Nach Messung wird Gerät in einen inaktiven Zustand gesetzt.
        self.ONE_TIME_HIGH_RES_MODE_2 = 0x21
        # Starte Messungen ab 4 Lux.
        # Nach Messung wird Gerät in einen inaktiven Zustand gesetzt.
        self.ONE_TIME_LOW_RES_MODE = 0x23

    def convertToNumber(self, data):
        # Einfache Funktion um 2 Bytes Daten
        # in eine Dezimalzahl umzuwandeln
        return ((data[1] + (256 * data[0])) / 1.2)

    def readLight(self):
        data = bus.read_i2c_block_data(self.DEVICE,self.ONE_TIME_HIGH_RES_MODE_1)
        return self.convertToNumber(data)

def main():
    sensor = LightSensor()
    try:
        while True:
            print("Light Level : " + str(sensor.readLight()) + " lx")
            time.sleep(0.5)
    except KeyboardInterrupt:
        pass

if __name__ == "__main__":
    main()
```

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen

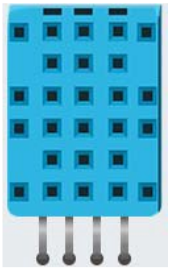



Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 light_sensor.py
```

## Lektion 7 : Erkennen der Temperatur und Luftfeuchtigkeit

Der DHT11 ist ein sehr interessanter Sensor, da er nicht nur eine Funktion hat, sondern zwei! Er enthält sowohl einen Feuchtigkeitssensor als auch einen Temperatursensor, welche beide sehr genau sind. Ideal für jedes Wetterstationsprojekt, oder wenn Sie die Temperatur und Luftfeuchtigkeit im Raum überprüfen möchten!



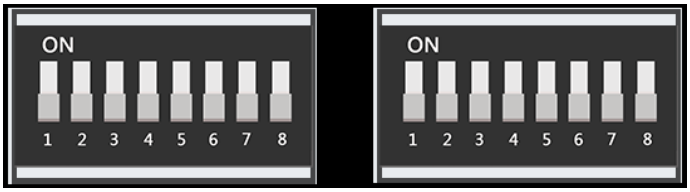
Der DHT11-Sensor ist sehr einfach zu erkennen. Ein kleiner blauer Sensor mit vielen kleinen Löchern. Er liegt rechts neben dem Relais und oberhalb des Berührungssensors. Als besonders zugänglich hat sich die Python DHT Sensor Library erwiesen, die auf https://github.com/coding-world/Python_DHT unter der [MIT-Lizenz] veröffentlicht wurde.
Die Bibliothek wird verwendet, um Temperatur und Feuchtigkeit als Werte auszugeben ohne dabei komplizierte mathematische Berechnungen durchführen zu müssen.

```
import Python_DHT

sensor = Python_DHT.DHT11
pin = 4

feuchtigkeit, temperatur = Python_DHT.read_retry(sensor, pin)
print("Temperatur = "+str(temperatur)+ "C Feuchtigkeit = "+str( feuchtigkeit)+"%")
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 dht11.py
```

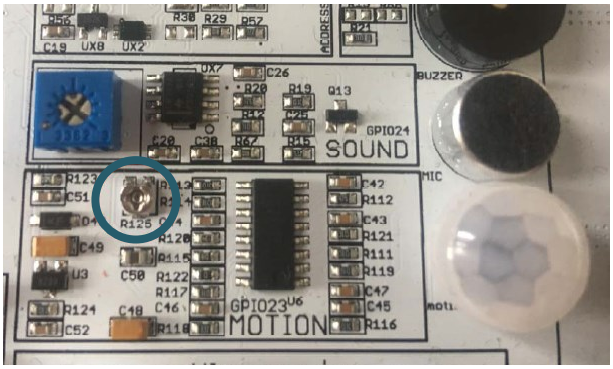
**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen




## Lektion 8 : Bewegungen erkennen

Der Bewegungssensor ist einer der nützlichsten und am häufigsten benutzten Sensoren. Man kann mit ihm zum Beispiel eine Alarmanlage bauen. Wenn der Sensor eine Bewegung erkennt, kann er ein Signal an den Buzzer senden, der dann einen lauten Alarmton von sich gibt.


Der Bewegungssensor befindet sich direkt unter dem Schallsensor und wird von einer kleinen, transparenten Kappe abgedeckt. Die Kappe hilft dem Sensor mehr Bewegungen zu erkennen, indem sie das Infrarotlicht der Umgebung bricht. Die Empfindlichkeit des Bewegungssensors wird, wie die des Schallsensors mit einem Potentiometer geregelt. Dieser befindet sich unterhalb des Potentiometers des Schallsensors, ist jedoch deutlich kleiner. Mit Hilfe eines Kreuzschraubendrehers können Sie einstellen, über welche Entfernungen der Bewegungssensor auslösen soll. Wenn Sie gegen den Uhrzeigersinn drehen, steigt die Empfindlichkeit, mit dem Uhrzeigersinn verringert sie sich.



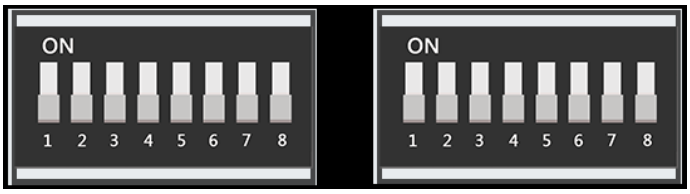

Der Bewegungssensor wird durch die GPIO-Pins gesteuert. Wenn eine Bewegung erkannt wird, wird der Bewegungssensor ein Signal senden. Dieses hält einige Zeit an und hört dann wieder auf bis der Sensor die nächste Bewegung erkennt.

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import RPi.GPIO as GPIO
import time #Importieren der Bibliotheken

motion_pin = 16 #Den Pin des Bewegungssensors einer Variable zuweisen.

GPIO.setmode(GPIO.BOARD) #Die GPIO Boardkonfiguration benutzen.
GPIO.setup(motion_pin, GPIO.IN) #Der Pin der Deklarierten Variable wird als Input gesetzt

try:      # Beginn einer Schleife
     while True:
          if(GPIO.input(motion_pin) == 0): # Wenn der Sensor Input = 0 ist
               print("Keine Bewegung ...") # Wird der print Befehl ausgeführt
          elif(GPIO.input(motion_pin) == 1): # Wenn der Sensor Input = 1 ist
               print("Bewegung Erkannt!") # Wird der print Befehl ausgeführt
          time.sleep(0.1) # 0,1 Sekunde Warten
except KeyboardInterrupt:
     GPIO.cleanup() # Gibt GPIO Ports wieder frei.
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

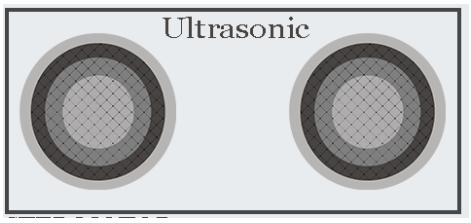
```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 motion.py
```

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen




## Lektion 9 : Entfernungen mit dem Ultraschallsensor messen

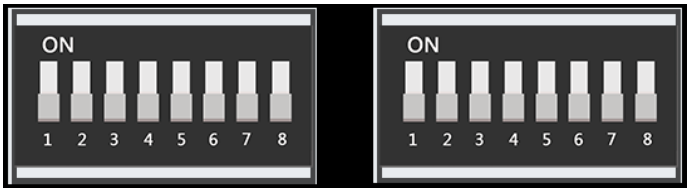
Nun werden wir lernen, wie wir den Ultraschallsensor verwenden um Entfernungen zu messen und auf dem Joy-Pi Bildschirm auszugeben.
Autos verwenden übrigens die gleiche Methode um Abstände zu messen.



Der Ultraschallsensor befindet sich rechts unten auf der Joy Pi Platine, direkt über der Schrittmotor- und der Servo-Schnittstellen. Er ist leicht an den zwei großen Kreisen zu erkennen. Wir werden unsere Hände über dem Entfernungssensor bewegen um die Entfernung zwischen unseren Händen und dem Joy Pi zu messen.

Der Abstandssensor arbeitet mit **GPIO.INPUT**, doch es ist etwas anders als das, was wir in unseren vorherigen Lektionen verwendet haben. Der Sensor benötigt ein gewisses Intervall um in der Lage zu sein, die Entfernung auf genaue Art und Weise zu erfassen. Er sendet ein Ultraschallsignal und empfängt mit einem eingebauten Sensor das Echo, das von einem Hindernis zurückgeworfen wird. Aus dem zeitlichen Abstand zwischen dem Senden des Signals und dem Empfangen des Echos, wird die Distanz berechnet.

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen




```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : www.modmypi.com
# Link: https://www.modmypi.com/blog/hc-sr04-ultrasonic-range-sensor-on-the-raspberry-pi

import RPi.GPIO as GPIO
import time

GPIO.setmode(GPIO.BOARD) # Setze die GPIO Boardkonfiguration ein.

TRIG = 36    # Variablendeklaration
ECHO = 32    # Variablendeklaration

print ("Entfernung wird ermittelt.") # Ausgabe von Text in der Konsole

GPIO.setup(TRIG,GPIO.OUT) # Variable TRIG als Output festlegen.
GPIO.setup(ECHO,GPIO.IN)  # Variable ECHO als Input festlegen.

GPIO.output(TRIG, False)
print ("Warte auf den Sensor.")
time.sleep(2) # 2 Sekunden Wartezeit.

GPIO.output(TRIG, True)  # Sendet ein Ultraschallsignal
time.sleep(0.00001)      # Wartet 0,00001 Sekunden
GPIO.output(TRIG, False) # Beendet das senden des Ultraschallsignals

while GPIO.input(ECHO)==0:
  pulse_start = time.time()

while GPIO.input(ECHO)==1:
  pulse_end = time.time()
```

```
pulse_duration = pulse_end - pulse_start # Berechnung für die Dauer Des Pulses

distance = pulse_duration * 17150  # Berechnung zur Bestimmung der Entfernung.

distance = round(distance, 2)      # Ergebnis wird auf 2 Nachkommastellen gerundet.

print ("Distance:",distance,"cm")    # Konsolenausgabe der Distanz in cm.

GPIO.cleanup() # Gibt GPIO Ports wieder frei.
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

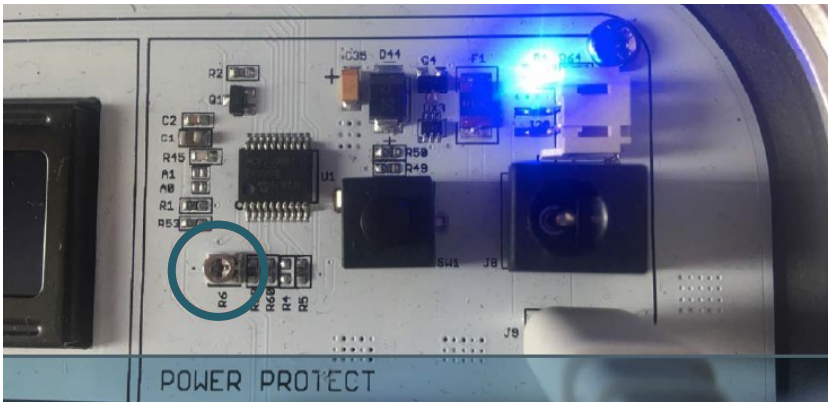
```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 distance.py
```

## Lektion 10 : Steuern des LCDs

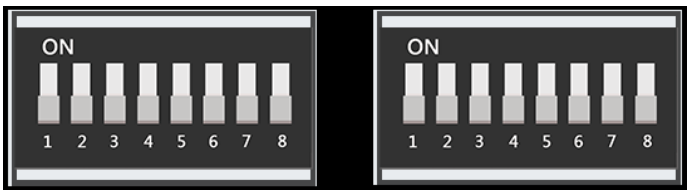
Mit dem Joy Pi können Sie Daten auf dem LCD anzeigen lassen, die Sie mit Ihren Sensoren sammeln und aktualisieren Sie in Echtzeit, abhängig von den Änderungen, die die Module durchlaufen. Zum Beispiel in Verbindung mit dem Temperatursensor - lassen Sie immer die aktuelle Temperatur und Luftfeuchtigkeit auf dem LCD anzeigen.

Der LCD-Bildschirm nimmt einen großen Teil des Joy Pi Boards ein. Er befindet sich oben in der Mitte des Joy Pis, rechts von der GPIO LED-Anzeige. Sobald das Demo-Skript und die Beispiele ausgeführt werden, schaltet sich das Display ein. Dank der integrierten Hintergrundbeleuchtung kann man auch in völliger Dunkelheit Daten von dem Display ablesen.

Wie bereits der Schall- und der Bewegungssensor, hat auch das LCD ein zugehöriges Potentiometer. Mit diesem Potentiometer lässt sich der Kontrast des Displays einstellen.
Wenn Sie gegen den Uhrzeigersinn drehen, wird der Kontrast höher und mit dem Uhrzeigersinn schwächer.



**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen




```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import time
import board
import busio
import adafruit_character_lcd.character_lcd_i2c as character_lcd

# Definiere LCD Zeilen und Spaltenanzahl.
lcd_columns = 16
lcd_rows    = 2

# Initialisierung I2C Bus
i2c = busio.I2C(board.SCL, board.SDA)

# Festlegen des LCDs in die Variable LCD
lcd = character_lcd.Character_LCD_I2C(i2c, lcd_columns, lcd_rows)

try:
    # Hintergrundbeleuchtung einschalten
    lcd.backlight = True

    # Zwei Worte mit Zeilenumbruch werden ausgegeben
    lcd.message = "Hallo\nWelt!"

    # 5 Sekunden warten
    time.sleep(5.0)

    # Cursor anzeigen lassen.
    lcd.clear()
    lcd.cursor = True
    lcd.message = "Show Cursor!"

    # 5 Sekunden warten
    time.sleep(5.0)

    # Cursor blinken lassen
    lcd.clear()
    lcd.blink = True
    lcd.message = "Blinky Cursor!"

    # 5 Sekunden warten, den blinkenden Cursor stoppen und Cursor ausblenden
    time.sleep(5)
    lcd.blink = False
    lcd.clear()
```

```
    # Nachricht von Rechts/Links scrollen lassen.
    lcd.clear()
    scroll_msg = "<-- Scroll -->"
    lcd.message = scroll_msg
    for i in range(len(scroll_msg)):
        time.sleep(0.5)
        lcd.move_right()
    for i in range(len(scroll_msg)):
        time.sleep(0.5)
        lcd.move_left()

    # Hintergrundbeleuchtung an und ausschalten.
    lcd.clear()
    lcd.message = "Flash backlight\nin 5 seconds..."
    time.sleep(5.0)
    # Hintergrundbeleuchtung ausschalten.
    lcd.backlight = False
    time.sleep(1.0)
    lcd.backlight = True
    time.sleep(1.0)
    lcd.backlight = False
    # Nachricht ändern.
    lcd.clear()
    lcd.message = "Goodbye"
    # Hintergrundbeleuchtung einschalten.
    lcd.backlight = True
    # Hintergrundbeleuchtung ausschalten.
    time.sleep(2.0)
    lcd.clear()
    lcd.backlight = False

except KeyboardInterrupt:
    # LCD ausschalten.
    lcd.clear()
    lcd.backlight = False
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
python3 lcd.py
```

# Lektion 11 : Lesen und Schreiben von RFID - Karten

In dieser Lektion werden Sie lernen, wie man das RFID-Modul steuert. Das RFID-Modul ist ein sehr interessantes und nützliches Modul. Es wird weltweit in einer Vielzahl von Lösungen eingesetzt, wie z.B. intelligente Türschlösser, mitarbeiterausweise, Visitenkarten und sogar Hundehalsbänder.


Das RFID-Modul befindet sich direkt unter dem Raspberry Pi und sieht wie ein kleines WiFi-Symbol aus. Dieses Symbol bedeutet drahtlose Konnektivität. Um es zu benutzen, müssen wir den Chip oder die Karte nehmen, die mit dem Joy Pi geliefert wird und die über den Joy Pi RFID-Chip Bereich halten. Es muss nah genug für unser Skript sein, damit es erkannt wird. 2 - 4 cm sollten nah genug sein. Probieren Sie es einfach aus!

Für das RFID RC522 Shield verwenden wir den SPI-Bus. Damit der Kernel beim Starten geladen wird, bearbeiten wir die config.txt Datei indem wir folgenden Befehl eingeben:

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

Folgender Inhalt wird an das Ende der Datei hinzugefügt:

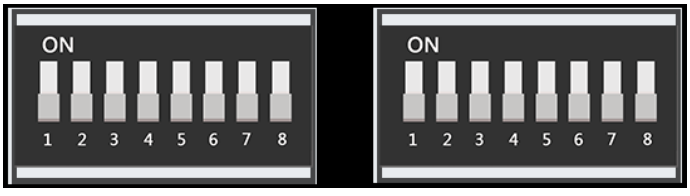
```
device_tree_param=spi=on
dtoverlay=spi-bcm2708
```

Gespeichert und beendet wird mit **STRG + O** und **STRG + X**.
Danach aktivieren Sie noch SPI:

```
sudo raspi-config
```

Unter ***Interfacing Options*** → ***SPI*** aktivieren und anschließend den Raspberry Pi neu starten.

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen



Um in den Ordner für die Skripte des RFID Lesers zu gelangen, geben Sie folgenden Befehl ein:

```
cd /home/pi/Desktop/Joy-Pi/Python3/MFRC522-python
```

Wenn Sie nun die RFID Karte oder den Chip beschreiben wollen, benutzen Sie folgenden Befehl:

```
sudo python3 Write.py
```

Um die Dateien zu bearbeiten die Sie auf dem Chip oder der Karte speichern, müssen Sie den Quellcode bearbeiten:

```
        # Select the scanned tag
        MIFAREReader.MFRC522_SelectTag(uid)

        # Authenticate
        status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
        print "\n"

        # Check if authenticated
        if status == MIFAREReader.MI_OK:

            # Variable for the data to write
            data = [99, 11, 55, 66, 44, 111, 222, 210, 125, 153, 136, 199, 144, 177, 166, 188]

            # Fill the data with 0xFF
            for x in range(0,16):
                data.append(0xFF)
```

Um die Dateien die gespeichert werden zu verändern, müssen Sie nun die Zahlen in der eckigen Klammer verändern, allerdings können diese Zahlen nicht kleiner als 0 und nicht größer als 255 sein.

Wenn Sie anschließend die Zahlenfolge auslesen möchten, benutzen Sie den folgenden Befehl:

```
sudo python Read.py
```

Wenn Sie nun die Karte oder den Chip auf den Leser legen wird Ihnen die auf der Karte gespeicherte Zahlenfolge angezeigt.

```
Card detected
Card read UID: 107,144,78,115
Size: 8
Sector 8 [99, 11, 55, 66, 44, 111, 222, 210, 125, 153, 136, 199, 144, 177, 166,
188]
```

Beispielcode RFID-Read:

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
#
#    Copyright 2014,2018 Mario Gomez <mario.gomez@teubi.co>
#    This file is part of MFRC522-Python
#    MFRC522-Python is a simple Python implementation for
#    the MFRC522 NFC Card Reader for the Raspberry Pi.
#    MFRC522-Python is free software: you can redistribute it and/or modify
#    it under the terms of the GNU Lesser General Public License as published by
#    the Free Software Foundation, either version 3 of the License, or
#    (at your option) any later version.
#    MFRC522-Python is distributed in the hope that it will be useful,
#    but WITHOUT ANY WARRANTY; without even the implied warranty of
#    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
#    GNU Lesser General Public License for more details.
#    You should have received a copy of the GNU Lesser General Public License
#    along with MFRC522-Python.  If not, see <http://www.gnu.org/licenses/>.
import RPi.GPIO as GPIO
import MFRC522
import signal

continue_reading = True

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    print("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()

# Welcome message
print("Welcome to the MFRC522 data read example")
print("Press Ctrl-C to stop.")

# This loop keeps checking for chips. If one is near it will get the UID and authenti-
cate
while continue_reading:

    # Scan for cards
    (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # If a card is found
    if status == MIFAREReader.MI_OK:
        print("Card detected")

    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()
```

```
        # Print UID
        print("Card read UID: %s,%s,%s,%s".format(uid[0], uid[1], uid[2], uid[3]))

        # This is the default key for authentication
        key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

        # Select the scanned tag
        MIFAREReader.MFRC522_SelectTag(uid)

        # Authenticate
        status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)

        # Check if authenticated
        if status == MIFAREReader.MI_OK:
            MIFAREReader.MFRC522_Read(8)
            MIFAREReader.MFRC522_StopCrypto1()
        else:
            print("Authentication error")
```

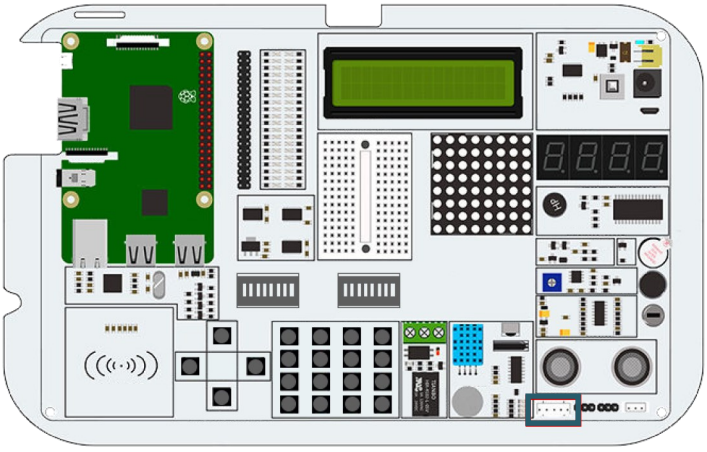
Beispielcode RFID-Write:

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
import RPi.GPIO as GPIO
import MFRC522
import signal

continue_reading = True

# Funktion um cleanup Funktionen durchzuführen wenn das Script abgebrochen wird.
def end_read(signal,frame):
    global continue_reading
    print ("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

signal.signal(signal.SIGINT, end_read)

# Erstelle ein Objekt aus der Klasse MFRC522
MIFAREReader = MFRC522.MFRC522()

# Diese Schleife Sucht dauerhaft nach Chips oder Karten. Wenn eine nah ist bezieht er
die UID und identifiziert sich.
while continue_reading:

    # SUcht Karten
    (status,TagType) = MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # Wenn Karte gefunden
    if status == MIFAREReader.MI_OK:
        print ("Card detected")
    # UID der Karte erhalten
    (status,uid) = MIFAREReader.MFRC522_Anticoll()
```

```
    # Wenn UID erhalten, fortfahren
    if status == MIFAREReader.MI_OK:

        # UID in Konsole ausgeben
        print ("Card read UID: %s,%s,%s,%s" % (uid[0], uid[1], uid[2], uid[3]))

        # Standard Schlüssel für Authentifizierungen
        key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]


        MIFAREReader.MFRC522_SelectTag(uid)

        # Authentifizieren
        status = MIFAREReader.MFRC522_Auth(MIFAREReader.PICC_AUTHENT1A, 8, key, uid)
        print ("\n")

        # Prüfen ob authentifiziert
        if status == MIFAREReader.MI_OK:

            # Variablen der Werte die auf der Karte gespeichert werden sollen.
            data = [99, 11, 55, 66, 44, 111, 222, 210, 125, 153, 136, 199, 144, 177, 166, 188]

            for x in range(0,16):
                data.append(0xFF)

            print ("Sector 8 looked like this:")
            # Block 8 lesen
            MIFAREReader.MFRC522_Read(8)
            print ("\n")

            print ("Sector 8 will now be filled with 0xFF:")
            # Dateien Schreiben
            MIFAREReader.MFRC522_Write(8, data)
            print ("\n")

            print ("It now looks like this:")
            # Überprüfen ob beschrieben wurde
            MIFAREReader.MFRC522_Read(8)
            print ("\n")

            MIFAREReader.MFRC522_StopCrypto1()

            # Sicherstellen das, das Kartenlesen eingestellt wird.
            continue_reading = False
        else:
            print ("Authentification error")
```

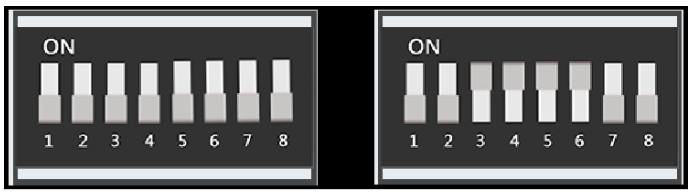
# Lektion 12 : Schrittmotoren verwenden

Der Schrittmotor ist ein unabhängiges Modul, welches Sie mit dem Board verbinden müssen.
Verbinden Sie den Schrittmotor einfach am folgenden Anschluss des Joy Pi Boards:



Der Schrittmotor kann sich während der Verwendung erwärmen. Dies ist technisch bedingt und nicht ungewöhnlich.
Der Schrittmotor ist mit 4 GPIO-Pins verbunden, welche schnell hintereinander eingeschaltet werden. Dies bewirkt, dass der Schrittmotor vorwärts "schiebt" und einen Schritt macht. Mit der Funktion **turnSteps** kann eine beliebige Anzahl von Schritten ausgeführt werden. Die Funktion **turnDegrees** lässt den Motor um einen bestimmten Winkel drehen.

**Achtung!** Für dieses Beispiel müssen Sie zwischen den Modulen wechseln. Stellen Sie die Schalter Nummer 3, 4, 5 und 6 der rechten Schalteinheit auf **ON**. Alle anderen Schalter sollten auf **OFF** stehen.



Beispielcode Schrittmotor

```
                                                                    #!/usr/bin/
python
# -*- coding: utf-8 -*-
# Author : Original author ludwigschuster
# Original Author Github: https://github.com/ludwigschuster/RasPi-GPIO-Stepmotor

import time
import RPi.GPIO as GPIO
import math

class Stepmotor:

    def __init__(self):
        # GPIO modus Festlegen
        GPIO.setmode(GPIO.BOARD)
        # Das Sind die Pins Ihres RasperryPis die benutzt werden.
        self.pin_A = 29
        self.pin_B = 31
        self.pin_C = 33
        self.pin_D = 35
        self.interval = 0.010

        # Pins als Output Deklarieren
        GPIO.setup(self.pin_A,GPIO.OUT)
        GPIO.setup(self.pin_B,GPIO.OUT)
        GPIO.setup(self.pin_C,GPIO.OUT)
        GPIO.setup(self.pin_D,GPIO.OUT)
        GPIO.output(self.pin_A, False)
        GPIO.output(self.pin_B, False)
        GPIO.output(self.pin_C, False)
        GPIO.output(self.pin_D, False)

    def Step1(self):

        GPIO.output(self.pin_D, True)
        time.sleep(self.interval)
        GPIO.output(self.pin_D, False)

    def Step2(self):

        GPIO.output(self.pin_D, True)
        GPIO.output(self.pin_C, True)
        time.sleep(self.interval)
        GPIO.output(self.pin_D, False)
        GPIO.output(self.pin_C, False)

    def Step3(self):

        GPIO.output(self.pin_C, True)
        time.sleep(self.interval)
        GPIO.output(self.pin_C, False)

    def Step4(self):
        GPIO.output(self.pin_B, True)
        GPIO.output(self.pin_C, True)
        time.sleep(self.interval)
        GPIO.output(self.pin_B, False)
        GPIO.output(self.pin_C, False)

    def Step5(self):
        GPIO.output(self.pin_B, True)
        time.sleep(self.interval)
```

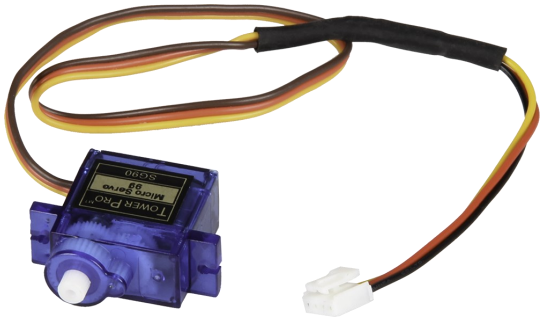
```
    def Step6(self):

        GPIO.output(self.pin_A, True)
        GPIO.output(self.pin_B, True)
        time.sleep(self.interval)
        GPIO.output(self.pin_A, False)
        GPIO.output(self.pin_B, False)

    def Step7(self):

        GPIO.output(self.pin_A, True)
        time.sleep(self.interval)
        GPIO.output(self.pin_A, False)

    def Step8(self):

        GPIO.output(self.pin_D, True)
        GPIO.output(self.pin_A, True)
        time.sleep(self.interval)
        GPIO.output(self.pin_D, False)
        GPIO.output(self.pin_A, False)

    def turn(self,count):
        for i in range (int(count)):
            self.Step1()
            self.Step2()
            self.Step3()
            self.Step4()
            self.Step5()
            self.Step6()
            self.Step7()
            self.Step8()

    def close(self):
        # Die GPIO Pins fuer andere Aktivitaeten freigeben.
        GPIO.cleanup()

    def turnSteps(self, count):
        # Bewegen um n schritte
        # (n wird von Ihnen festgelegt.)
        for i in range (count):
            self.turn(1)

    def turnDegrees(self, count):
        # Bewegen um n Grad (Kleine Werte koennen zu Ungenauigkeit fuehren.)
        # (Gradnummer die gedreht werden soll angeben.)
        self.turn(round(count*512/360,0))

    def turnDistance(self, dist, rad):
        self.turn(round(512*dist/(2*math.pi*rad),0))

def main():

    print("Bewegung gestartet.")
    motor = Stepmotor()
    print("Ein Schritt")
    motor.turnSteps(1)
    time.sleep(0.5)
    print("20 Schritte")
    motor.turnSteps(20)
    time.sleep(0.5)
```

```
    print("Viertel Umdrehung")
    motor.turnDegrees(90)
    print("Bewegung gestoppt.")
    motor.close()

if __name__ == "__main__":
    main()
```

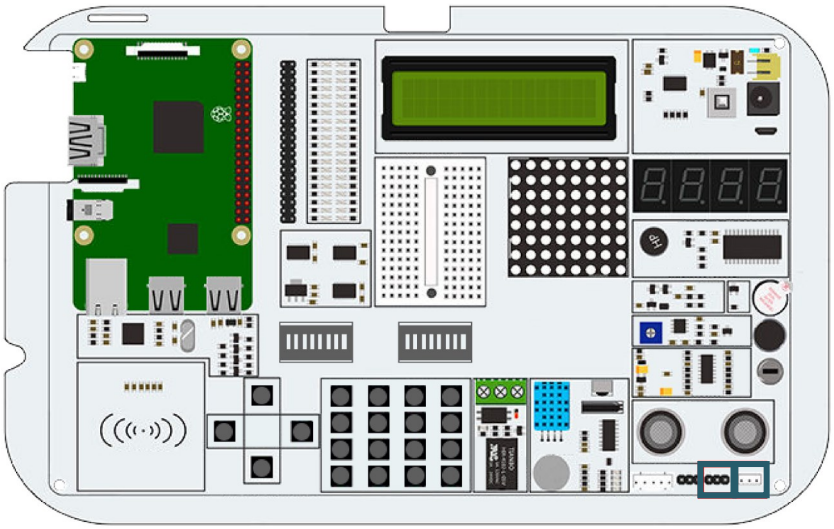
Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 stepmotor.py
```

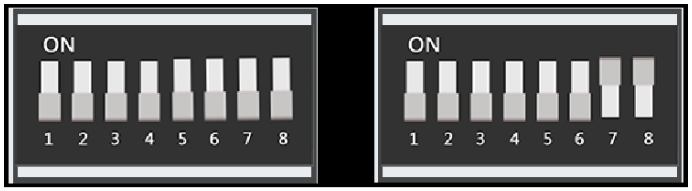
## Lektion 13 : Steuerung von Servomotoren



Mit Hilfe des Servomotors lassen sich Geräte mechanisch steuern und Teile bewegen. So können beispielsweise intelligente Abfalleimer, eine Schachtel mit intelligenter, öffnender / schließender Tür und viele andere interessante Projekte erstellt werden.
Der Joy Pi besitzt zwei Servo-Schnittstellen, die beide zur Steuerung von Servomotoren genutzt werden können. In diesem Tutorial verwenden wir die Schnittstelle Nummer zwei, die als *Servo2* markiert ist. Sie können natürlich auch die anderen Servo-Schnittstellen verwenden, dazu müssen Sie jedoch das Skript auf die richtigen GPIOs anpassen.
Der Servomotor benötigt drei Pins: Positiv, Negativ und den Daten-Pin. Der positive Pin ist das rote Kabel, der negative Pin das schwarze Kabel (auch Masse genannt) und das Datenkabel ist Gelb.



**Achtung!** Für dieses Beispiel müssen Sie zwischen den Modulen wechseln. Stellen Sie die Schalter Nummer 7 und 8 der rechten Schalteinheit auf **ON**. Alle anderen Schalter sollten auf **OFF** stehen.




| Kabel | Pin |
|---|---|
| Rot | Mittlerer Pin von Servo2 |
| Schwarz | Rechter Pin von Servo2 |
| Bunt | Linker Pin von Servo2 |

Schauen Sie sich den Beispielcode an, um ihn besser zu verstehen:

Der Servo benutzt die GPIO.Board Nummer 22. Jedes Mal wird das Skript die Richtung vom Servomotor festlegen, um zu drehen. Wir können positive Gradzahlen verwenden, um links herumzudrehen und negative, um rechts herumzudrehen. Verändern Sie einfach die Gradzahlen und schauen Sie, wie sich die Drehung des Motors verändert.

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 servo.py
```

Beispielcode Servomotor:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : Original author WindVoiceVox
# Original Author Github: https://github.com/WindVoiceVox/Raspi_SG90
import RPi.GPIO as GPIO
import time
import sys

class sg90:

  def __init__( self, pin, direction ):
    GPIO.setmode( GPIO.BOARD )
    GPIO.setup( pin, GPIO.OUT )
    self.pin = int( pin )
    self.direction = int( direction )
    self.servo = GPIO.PWM( self.pin, 50 )
    self.servo.start(0.0)

  def cleanup( self ): #Funktion zum Stoppen und GPIO Pins Freigeben
    self.servo.ChangeDutyCycle(self._henkan(0))
    time.sleep(0.3)
    self.servo.stop()
    GPIO.cleanup()
```

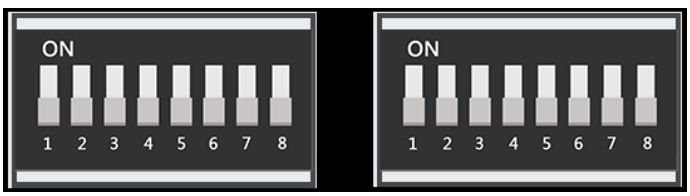
Beispielcode fortgeführt:

```
  def currentdirection( self ): #Funktion die die Momentane Position feststellt.
    return self.direction

  def _henkan( self, value ):
    return 0.05 * value + 7.0

  def setdirection( self, direction, speed ): #Funktion um die Richtung anzugeben
    for d in range( self.direction, direction, int(speed) ):
      self.servo.ChangeDutyCycle( self._henkan( d ) )
      self.direction = d
      time.sleep(0.1)
    self.servo.ChangeDutyCycle( self._henkan( direction ) )
    self.direction = direction

def main():
    servo_pin = 22
    s = sg90(servo_pin,0) #Deklaration von Pin und Motor
    try:
        while True:
            print ("Turn left ...")
            s.setdirection( 100, 10 )   #Links Herum drehen
            time.sleep(0.5)             #0,5 Sekunden warten
            print ("Turn right ...")
            s.setdirection( -100, -10 ) #Rechts Herum drehen
            time.sleep(0.5)             #0,5 Sekunden warten
    except KeyboardInterrupt:
        s.cleanup()

if __name__ == "__main__":
    main()
```

## Lektion 14 : Steuern der 8 x 8 LED-Matrix

Die LED-Matrix spielt in vielen blinkenden LED-Projekten eine wichtige Rolle. Selbst wenn Sie es auf den ersten Blick nicht sehen, kann die LED-Matrix viel mehr als nur rot blinken. Sie kann verwendet werden, um kleine Symbole anzuzeigen und sogar um ein Spiel wie Snake zu spielen. Das LED-Matrixmodul ist ein großes quadratisches Modul, das sich auf der linken Seite der Segment-LED und direkt unter dem LCD befindet. Es kann leicht durch die kleinen weißen Punkte, welche die LEDs sind, erkannt werden.
In diesem Beispiel wird ein kurzer Text auf der LED-Matrix angezeigt. Im Skript wird eine Zeichenfolge mit einer Nachricht erstellt und mit der Funktion **show_message()**, auf dem Matrix-Display angezeigt.
Sie können Eigenschaften, wie z.B. Verzögerungen steuern, die den Nachrichtenfluss etwas verlangsamt. Die LED-Matrix verwendet im Gegensatz zu anderen Modulen eine SPI-Schnittstelle von der die Matrix aus gesteuert werden kann. Probieren Sie das Beispiel aus und ändern Sie den Code, um zu sehen, was passiert.

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen



Beispielcode LED-Matrix

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# Copyright (c) 2017-18 Richard Hull and contributors
# License: https://github.com/rm-hull/luma.led_matrix/blob/master/LICENSE.rst
# Github link: https://github.com/rm-hull/luma.led_matrix/

# Alle benötigten Module importieren
import re
import time
from luma.led_matrix.device import max7219
from luma.core.interface.serial import spi, noop
from luma.core.render import canvas
from luma.core.virtual import viewport
from luma.core.legacy import text, show_message
from luma.core.legacy.font import proportional, CP437_FONT, TINY_FONT, SINCLAIR_FONT, LCD_FONT


def main(cascaded, block_orientation, rotate):

    # Matrix Gerät festlegen und erstellen.
    serial = spi(port=0, device=1, gpio=noop())
    device = max7219(serial, cascaded=cascaded or 1, block_orientation=block_orientation,
    rotate=rotate or 0)
    # Matrix Initialisierung in der Konsole anzeigen
    print("[-] Matrix initialized")

    # Hallo Welt in der Matrix anzeigen
    msg = "Hallo Welt"
    # Ausgegebenen Text in der Konsole Anzeigen
    print("[-] Printing: %s" % msg)
    show_message(device, msg, fill="white", font=proportional(CP437_FONT), scroll_delay=0.1)


if __name__ == "__main__":

    # cascaded = Anzahl von MAX7219 LED Matrixen, standart=1
    # block_orientation = choices 0, 90, -90, standart=0
    # rotate = choices 0, 1, 2, 3, Rotate display 0=0°, 1=90°, 2=180°, 3=270°, standart=0

    try:
        main(cascaded=1, block_orientation=90, rotate=0)
    except KeyboardInterrupt:
        pass
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 matrix_demo.py
```

# Lektion 15 : Steuern des 7 - Segment - Displays

Die Segment-Anzeige ist ein sehr nützliches Display, wenn es um Zahlen geht. Es kann uns die Zeit zeigen oder zählen, wie oft wir bestimmte Dinge getan haben. Ihrer Fantasie sind keine Grenzen gesetzt. Die Segment-Anzeige wird außerdem in vielen industriellen Lösungen, wie z.B. in Aufzügen verwendet.

Die Segment-Anzeige befindet sich direkt über dem Vibrationssensor und neben der LED-Matrix. Wenn sie ausgeschaltet ist, sind 4 Achten zu erkennen. Sobald Sie das Segment-Display-Modul verwenden, wird die dunkle Farbe zu glänzendem, hellem Rot.

In unserem Beispiel demonstrieren wir eine Uhr. Wir werden die Uhrzeit- und Datumsmodule verwenden, um die Systemzeit des Raspberry Pi zu erhalten, welche wir mit dem Befehl **segment.write_display()** auf der Anzeige ausgeben. Der Befehl **set_digit()** in Kombination mit den Nummern 0, 1, 2 und 3 legt die Position auf dem Display fest, an der die Nummer angezeigt werden soll.

Da in diesem Beispiel die aktuelle Systemzeit abgerufen wird, ist es notwendig den Raspberry Pi zuerst auf die richtige Zeitzone zu konfigurieren. Öffnen Sie dazu ein Terminalfenster und geben Sie den folgenden Befehl ein:

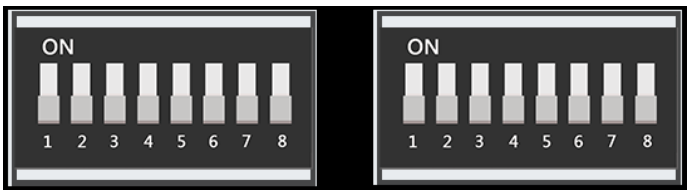
```
sudo dpkg-reconfigure tzdata
```

Es öffnet sich ein Fenster in dem Sie Ihre aktuelle Zeitzone auswählen können. Nachdem Sie die richtige Zeitzone ausgewählt haben, bestätigen Sie mit **OK** und drücken Sie noch einmal **Enter** zur Bestätigung.

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 segment.py
```

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen



Beispielcode Segment-Anzeige:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

import time
import datetime
from Adafruit_LED_Backpack import SevenSegment


segment = SevenSegment.SevenSegment(address=0x70)
#segment der I2C Adresse 0x70 und die Displaydefinition zuweisen

segment.begin()
# Initialisierung des Displays. Muss einmal ausgeführt werden bevor das Display benutzt wird.

print ("STRG+C Druecken zum beenden.") #print Befehl für Ausgabe zum beenden des Scriptes

#Schleife welche dauerhaft die Zeit updated und sie auf dem Display anzeigt.
try:
  while(True):
    now = datetime.datetime.now()
    hour = now.hour
    minute = now.minute
    second = now.second

    segment.clear()
    # Anzeige für die Stunden.
    segment.set_digit(0, int(hour / 10))     # Zehnerzahlen
    segment.set_digit(1, hour % 10)          # Einerzahlen
    # Anzeige für die Minuten.
    segment.set_digit(2, int(minute / 10))   # Zehnerzahlen
    segment.set_digit(3, minute % 10)        # Einerzahlen

    segment.set_colon(second % 2)

    segment.write_display() # Wird benötigt um die Display LEDs zu updaten.

    time.sleep(1) # Warte eine Sekunde
except KeyboardInterrupt:
    segment.clear()
    segment.write_display()
```

# Lektion 16 : Berührungen erkennen

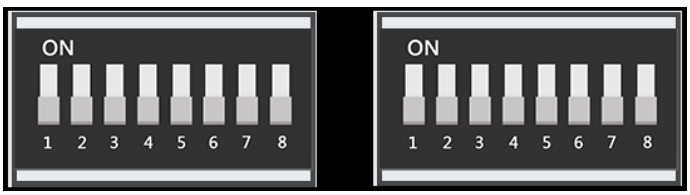
Der Touch-Sensor ist sehr nützlich, wenn es um Tastenfunktionen geht. Viele Produkte auf dem Markt verwenden Touch anstelle eines Knopfdrucks, zum Beispiel Smartphones und Tablets. Der Berührungssensor befindet sich direkt unter dem DHT11-Sensor und rechts neben dem Relais. Die leicht zugängliche Positionierung auf dem Joy Pi ermöglicht eine einfache Bedienung.

Der Berührungssensor funktioniert wie jedes andere Tastenmodul. Der einzige Unterschied besteht darin, dass er nur berührt werden muss anstatt gedrückt zu werden. Durch Berühren des Berührungssensors schließt das Modul eine Schaltung, da der Computer erkennt, dass der Sensor berührt wurde. Der Berührungssensor benutzt GPIO Board Pin 11.

Beispielcode Touchsensor:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

from RPi import GPIO      # Bibliotheken einfügen
import signal

TOUCH = 11                      # TOUCH pin 11 zuweisen (Variablendeklaration).

def setup_gpio():         # Funktion setup_gpio erstellen
    GPIO.setmode(GPIO.BOARD)  # Benutze GPIO Pins nach GPIO Board Schema.
    GPIO.setup(TOUCH, GPIO.IN, pull_up_down=GPIO.PUD_UP)

def do_smt(channel):
    print("Touch wurde erkannt")

def main():
    setup_gpio()
    try:
        GPIO.add_event_detect(TOUCH, GPIO.FALLING, callback=do_smt, bouncetime=200)
        signal.pause()
    except KeyboardInterrupt:
        pass
    finally:
        GPIO.cleanup()

if __name__ == '__main__':
    main()
```

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen



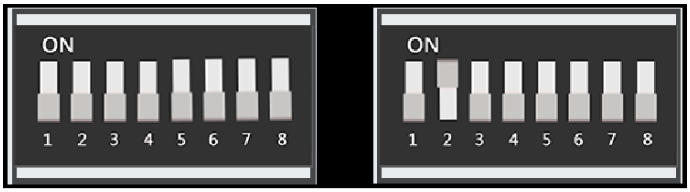
Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 touch.py
```

## Lektion 17 : Neigungen mit dem Neigungssensor erkennen

Der Neigungssensor ermöglicht es eine Neigung nach rechts oder links zu erkennen. Er wird in der Robotik und anderen Industrien verwendet um sicherzustellen, dass Dinge gerade gehalten werden. Er ist ein kleiner, länglicher, schwarzer Sensor, der zwischen dem DHT11 und dem Ultraschallsensor liegt. Man kann ihn leicht an dem Klang erkennen, den er macht, wenn Sie das Board zur Seite neigen.
Wenn der Neigungssensor nach links geneigt ist, wird die Schaltung aktiviert und ein *GPIO HIGH* -Signal wird gesendet. Bei einer Neigung nach rechts wird die Schaltung deaktiviert und ein *GPIO LOW* -Signal gesendet.

**Achtung!** Für dieses Beispiel müssen Sie zwischen den Modulen wechseln. Stellen Sie den Schalter Nummer 2 der rechten Schalteinheit auf **ON**. Alle anderen Schalter sollten auf **OFF** stehen.



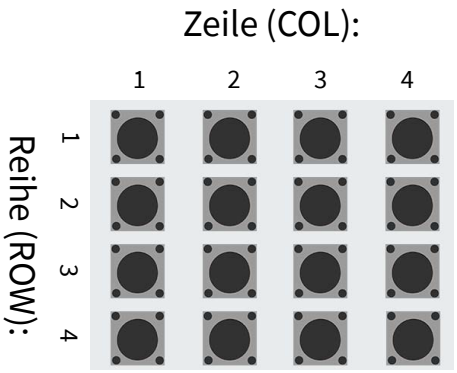

Beispielcode Neigungssensor:

```
#!/usr/bin/python

import time
import RPi.GPIO as GPIO

#tilt_pin wird definiert
tilt_pin = 15

#GPIO Modus wird auf GPIO.BOARD gesetzt
GPIO.setmode(GPIO.BOARD)

# Pin wird als Eingang festgelegt
GPIO.setup(tilt_pin, GPIO.IN)

try:
     while True:
          #positiv ist nach rechts, negativ ist nach links geneigt
          if GPIO.input(tilt_pin):
               print ("[-] Left Tilt")
          else:
               print ("[-] Right Tilt")
          time.sleep(1)
except KeyboardInterrupt:
     #Strg+c beendet das Programm
     GPIO.cleanup()
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 tilt.py
```

## Lektion 18: Verwenden der Taster-Matrix



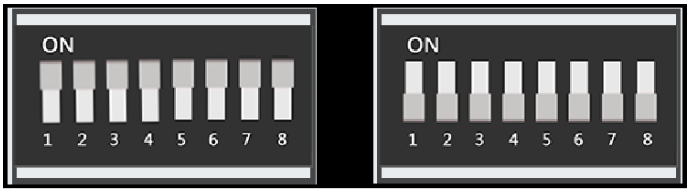

Die Taster-Matrix ist ein Modul mit 16 unabhängigen Tastern, die für viele Projekte benutzt werden können wie z.B. eine Tastatur oder ein Memory-Spiel.

Die Taster-Matrix befindet sich unten in der Mitte des Joy Pi Boards, rechts neben dem Relais. Sie ist leicht an den 16 einzelnen tasten zu erkennen. Die hervorragende Positionierung auf dem Board ermöglicht eine einfache Bedienung der Taster, während man trotzdem noch einen guten Überblick über alle anderen Sensoren hat.

Die Taster-Matrix besteht aus vier Spalten und Zeilen. Die Zeilen und Spalten werden mit ihren GPIO-Pins konfiguriert und das Objekt **ButtonMatrix()** wird als Schaltflächenvariable initialisiert.
Danach können Sie jeden Taster der Matrix drücken und feststellen, welcher Taster gedrückt wurde.

In diesem Beispiel wird die Funktion **activateButton()**, nachdem ein Tasterdruck erkannt wurde, aktiviert wodurch die Nummer des gedrückten Tasters angezeigt wird. Sie können diesen Code selbstverständlich nach Ihren eigenen Vorstellungen bearbeiten.

**Achtung!** Für dieses Beispiel müssen Sie zwischen den Modulen wechseln. Stellen Sie **ALLE** Schalter der linken Schalteinheit auf **ON**. Alle anderen Schalter sollten auf **OFF** stehen.



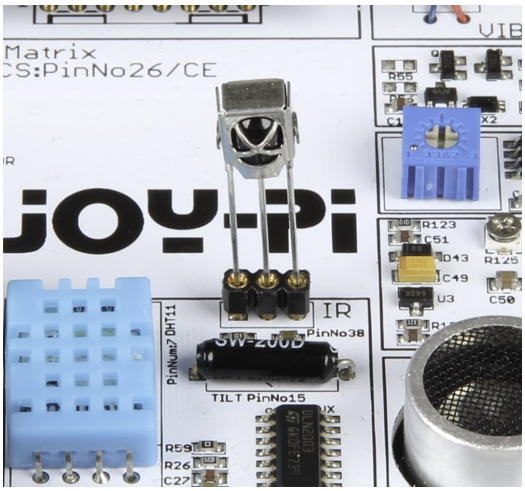
Beispielcode Tastermatrix:

```
#!/usr/bin/python
# -*- coding: utf-8 -*-
# Author : original author stenobot
# Original Author Github: https://github.com/stenobot/SoundMatrixPi

import RPi.GPIO as GPIO
import time

class ButtonMatrix():

    def __init__(self):

        GPIO.setmode(GPIO.BOARD)

        # Die IDs der Buttons werden festgelegt
        self.buttonIDs = [[4,3,2,1],[8,7,6,5],[12,11,10,9],[16,15,14,13]]
        # GPIO Pins für die Zeilen werden deklariert.
        self.rowPins = [13,15,29,31]
        # GPIO Pins für die Spalte werden deklariert.
        self.columnPins = [33,35,37,22]

        # Definiere Vier Inputs mit pull up Widerständen.
        for i in range(len(self.rowPins)):
            GPIO.setup(self.rowPins[i], GPIO.IN, pull_up_down = GPIO.PUD_UP)

        # Definiere Vier Outputs und setze sie auf high.
        for j in range(len(self.columnPins)):
            GPIO.setup(self.columnPins[j], GPIO.OUT)
            GPIO.output(self.columnPins[j], 1)

    def activateButton(self, rowPin, colPin):
        # Erhalte die Button Nummer
        btnIndex = self.buttonIDs[rowPin][colPin] - 1
        print("button " + str(btnIndex + 1) + " pressed")
        # Verhindert mehrere Knopfdrücke in zu kurzer zeit
        time.sleep(.3)

    def buttonHeldDown(self,pin):
        if(GPIO.input(self.rowPins[pin]) == 0):
            return True
        return False

def main():

    # Initialisierung der Button Matrix
    buttons = ButtonMatrix()
    try:
        while(True):
            for j in range(len(buttons.columnPins)):
                # Jeder Output Pin wird auf low gesetzt.
                GPIO.output(buttons.columnPins[j],0)
                for i in range(len(buttons.rowPins)):
                    if GPIO.input(buttons.rowPins[i]) == 0:
                        buttons.activateButton(i,j)
                        # Nichts tun solange der Button gedrückt gehalten wird.
                        while(buttons.buttonHeldDown(i)):
                            pass
                GPIO.output(buttons.columnPins[j],1)
    except KeyboardInterrupt:
        GPIO.cleanup()

if __name__ == "__main__":
    main()
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 button_matrix.py
```

## Lektion 19: Steuern und Verwenden des IR-Sensors




In dieser Lektion lernen Sie, wie man den Infrarotempfänger benutzt um IR-Codes von einer Fernbedienung zu erkennen. Die Verwendung dieser Methode ist äußerst nützlich, da wir verschieden Aktionen für verschiedene Tasten festlegen können. Mit einer Fernbedienung können Sie z.B. bei jedem Tastendruck verschiedene LEDs einschalten oder den Servomotor steuern.

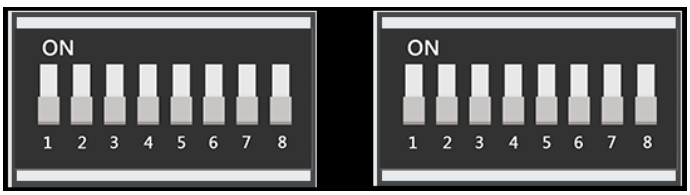
Der Sensor ist im Lieferumfang enthalten, ist jedoch nicht bereits in dem Koffer verbaut, sondern wird separat verpackt.
Der Steckplatz des IR-Sensors befindet sich rechts neben dem DHT11-Sensor und über dem Neigungssensor. Stecken Sie den IR-Sensor wie im Bild oben zusehen in den Steckplatz. Zusätzlich benötigen Sie die IR-Fernbedienung, die ebenfalls im Lieferumfang des Joy Pi Kits enthalten ist.

Der IR-Empfänger verwendet eine Bibliothek namens LIRC und Python-LIRC um die Codes, die wir mit der IR-Fernbedienung senden, zu verstehen. Die Out-Variable enthält die Taste, die wir gedrückt haben. Mit Hilfe von if-Abfragen können wir überprüfen, ob bestimmte Tasten gedrückt wurden. Anhand dieser Informationen können wir entsprechende Befehle ausführen.

**Hinweis!** Der IR-Sensor kann nur mit Python 2 Skripten ausgeführt werden und nicht mit Python 3.

**Wichtig!** Ziehen Sie den IR-Sensor aus dem Steckplatz heraus bevor Sie den Koffer schließen.

**Achtung!** Für dieses Beispiel müssen Sie alle Schalter der linken und der rechten Schalteinheit auf **OFF** stellen



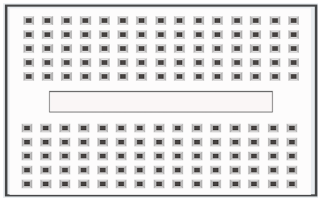
Beispielcode IR-Receiver:

```
#!/usr/bin/env python
import socket, signal
import lirc, time, sys
import RPi.GPIO as GPIO
from  array import array

GPIO.setmode(11)
#PORT = 42001
#HOST = "localhost"
Socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

Lirc = lirc.init("keys")
#lirc.set_blocking(False, Lirc)         # Un-Comment to stop nextcode() from
waiting for a signal ( will return empty array when no key is pressed )

def handler(signal, frame):
     Socket.close()
     GPIO.cleanup()
     exit(0)

signal.signal(signal.SIGTSTP, handler)

def sendCmd(cmd):
    n = len(cmd)
    a = array('c')
    a.append(chr((n >> 24) & 0xFF))
    a.append(chr((n >> 16) & 0xFF))
    a.append(chr((n >> 8) & 0xFF))
    a.append(chr(n & 0xFF))
    Socket.send(a.tostring() + cmd)

while True:

     Out = lirc.nextcode()
     print Out[0]
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

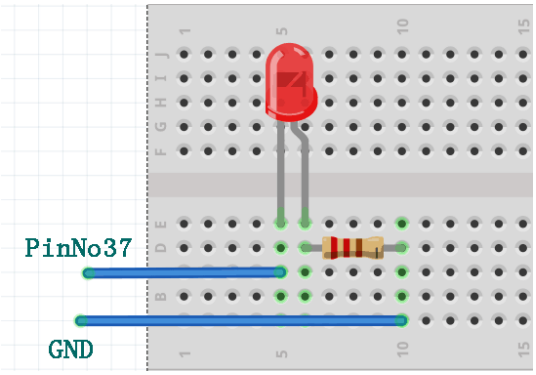
```
cd /home/pi/Desktop/Joy-Pi/Python2
sudo python IR.py
```

## Lektion 20: Eigene Schaltungen mit dem Breadboard



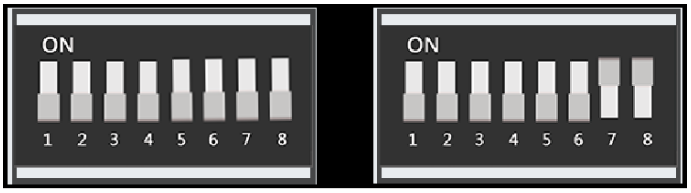
Das Breadboard ist ein äußerst nützliches Teil im Joy Pi, dass es uns ermöglicht eigene Schaltungen und Funktionen zu erstellen. Nachdem wir gelernt haben wie man all die Sensoren benutzt, ist es nun an der Zeit unseren eigenen zu erstellen. In dieser Lektion erstellen Sie Ihre erste benutzerdefinierte Schaltung anhand eines blinkenden LED Beispiels. Das Breadboard ist eine sogenannte Steckplatine und befindet sich in der Mitte des Joy Pis.

In diesem Beispiel wird eine benutzerdefinierte Schaltung erstellt mit der Funktion, eine LED blinken zu lassen.
Um dies zu tun, müssen Sie einen GPIO-Pin als Ausgabe wie Sie es bereits in früheren Lektionen getan haben und einen GND-Pin verwenden. Dazu wird die **Servo-Schnittstelle** (SERVO1-Schnittstelle) an **GPIO 37** verwendet.

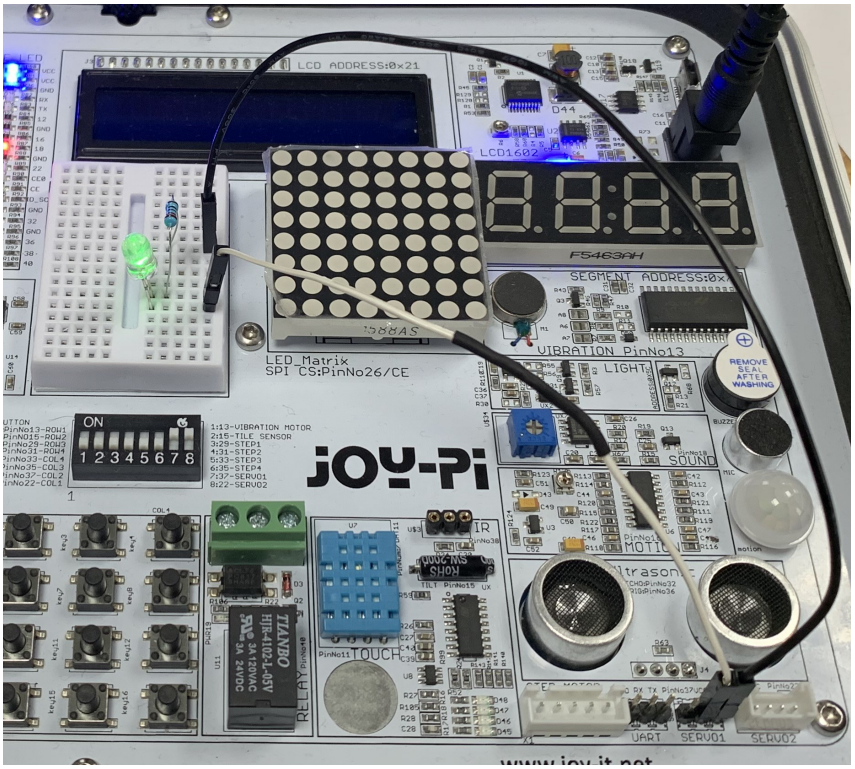



Sie können sich an diesem Bild orientieren um Ihre Schaltung auf dem Steckbrett zu erstellen. Vergessen Sie nicht, dass sich Pin Nummer 37 am GPIO-Port und Masse am GND-Port der SERVO1-Schnittstelle befinden.

**Achtung!** Für dieses Beispiel müssen Sie zwischen den Modulen wechseln, da die Servo-Pins verwendet werden. Stellen Sie dafür die Schalter **7** und **8** der rechten Schalteinheit auf **ON**. Alle anderen Schalter sollten auf **OFF** stehen.




Sie müssen einen Widerstand verwenden und ihn an der negativen Seite der LED anschließen (die negative Seite der LED ist die mit dem kürzeren Bein). Die andere Seite des Widerstands müssen Sie direkt mit einem Kabel mit dem GND-Pin der SERVO1-Schnittstelle verbinden. Die positive Seite der LED verbinden Sie mit dem GPIO37 Pin der SERVO1-Schnittstellle.




Nachdem Sie die Schaltung erfolgreich aufgebaut haben, ist es Zeit den Code zu schreiben um die LED zu kontrollieren. Das Signal des SERVO1 GPIO-Pin wird durch **GPIO.HIGH** high gesetzt und nach 0,2 Sekunden durch **GPIO.LOW** wieder heruntergesetzt. Dies hat zur Folge, dass die Leuchtdiode angeht und nach 0,2 Sekunden sich wieder abschaltet. Dadurch wird die LED anfangen zu blinken bis das Programm mit **STRG + C** beendet wird.

**Wichtig!** Die LED, der Widerstand und die Kabel sind nicht im Lieferumfang enthalten.

Beispielcode:

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

#definiere LED Pin
led_pin = 37
#setze GPIO Modus auf GPIO.BOARD
GPIO.setmode(GPIO.BOARD)
#lege Pin als Ausgang fest
GPIO.setup(led_pin, GPIO.OUT)

try:
     while True:
          #LED an
          GPIO.output(led_pin, GPIO.HIGH)
          #warte 0,2 Sekunden
          time.sleep(0.2)
          #LED aus
          GPIO.output(led_pin, GPIO.LOW)
          #warte 0,2 Sekunden
          time.sleep(0.2)

except KeyboardInterrupt:
     #STRG+C zum Beenden des Programms
     GPIO.cleanup()
```

Führen Sie die folgenden Befehle aus und versuchen Sie es selbst:

```
cd /home/pi/Desktop/Joy-Pi/Python3
sudo python3 blinking_led.py
```

## Lektion 21: Fotografieren mit der Joy Pi Kamera

Die Joy Pi Kamera ist äußerst nützlich und kann für eine Vielzahl an Projekten verwendet werden. Beispielsweise für Sicherheitskameras, Gesichtserkennung und vieles mehr. In der folgenden Lektion werden Ihnen die Grundlagen zur Verwendung der Joy Pi Kamera näher gebracht.

Die Kamera befindet sich mittig über dem Bildschirm des Joy Pis und ist mit einem USB-Kabel direkt mit dem Raspberry Pi verbunden.

Als nächstes, nachdem Sie sichergestellt haben, dass die Kamera verbunden ist, müssen Sie das *fswebcam* Paket mit folgendem Befehl installieren (das Paket ist in dem vorbereiteten Image bereits installiert):

```
sudo apt-get install fswebcam
```

Geben Sie den Befehl *fswebcam* gefolgt von einem Dateinamen ein. Mit der Webcam wird ein Bild aufgenommen und unter dem angegebenen Dateinamen gespeichert:

```
fswebcam image.jpg
```

So können Sie ein Bild in der Auflösung 1280 x 1024 aufnehmen:

```
fswebcam -r 1280-1024 image2.jpg
```

Wenn Sie jetzt den Befehl *--no-banner* hinzufügen entfernen Sie den Zeit- und Datumsstempel:

```
fswebcam -r 1280-1024 --no-banner image3.jpg
```

Um ein Video aufzunehmen, benutzen Sie den folgenden Befehl, wobei die Auflösung variabel ist:

```
ffmpeg -f v4l2 -r 25 -s 780x480 -i /dev/video0 Beispiel.avi
```

## 7. SONSTIGE INFORMATIONEN

### Unsere Informations- und Rücknahmepflichten nach dem Elektrogesetz (ElektroG)

**Symbol auf Elektro- und Elektronikgeräten:**

Diese durchgestrichene Mülltonne bedeutet, dass Elektro- und Elektronikgeräte **nicht** in den Hausmüll gehören. Sie müssen die Altgeräte an einer Erfassungsstelle abgeben. Vor der Abgabe haben Sie Altbatterien und Altakkumulatoren, die nicht vom Altgerät umschlossen sind, von diesem zu trennen.

**Rückgabemöglichkeiten:**
Als Endnutzer können Sie beim Kauf eines neuen Gerätes, Ihr Altgerät (das im Wesentlichen die gleiche Funktion wie das bei uns erworbene neue erfüllt) kostenlos zur Entsorgung abgeben. Kleingeräte bei denen keine äußere Abmessungen größer als 25 cm sind können unabhängig vom Kauf eines Neugerätes in Haushaltsüblichen Mengen abgeben werden.

**Möglichkeit Rückgabe an unserem Firmenstandort während der Öffnungszeiten:**
Simac GmbH, Pascalstr. 8, D-47506 Neukirchen-Vluyn

**Möglichkeit Rückgabe in Ihrer Nähe:**
Wir senden Ihnen eine Paketmarke zu mit der Sie das Gerät kostenlos an uns zurücksenden können. Hierzu wenden Sie sich bitte per E-Mail an Service@joy-it.net oder per Telefon an uns.

**Informationen zur Verpackung:**
Verpacken Sie Ihr Altgerät bitte transportsicher, sollten Sie kein geeignetes Verpackungsmaterial haben oder kein eigenes nutzen möchten kontaktieren Sie uns, wir lassen Ihnen dann eine geeignete Verpackung zukommen.

## 8. RECHTLICHES

Dieses Produkt enthält Software, welche ganz oder teilweise als freie Software den Lizenzbedingungen der GNU General Public License, Version 2 (GPL) oder X11 License (auch MIT License genannt) unterliegt. Die vollständigen Lizenztexte ersehen Sie nachfolgend. Näheres über die Lizenzbedingungen erfahren Sie unter http://www.gnu.org/licenses/old-licenses/gpl-2.0 und https://www.gnu.org/licenses/license-list.html. Da es sich um freie Software handelt, besteht keinerlei Gewährleistung, soweit gesetzlich zulässig. Details hierzu finden Sie in der GNU General Public License und der X11 License. Bitte beachten Sie, dass die Gewährleistung für die Hardware davon natürlich nicht betroffen ist und in vollem Umfang besteht.

Ferner verpflichten wir uns für 3 Jahre, den Quellcode auf Anfrage jeden in maschinenlesbarer Form zur Verfügung zu stellen, berechnet werden lediglich die Herstellungskosten des Mediums. Die Anfrage ist zu senden an service@joy-it.net.

Weitere Fragen beantworten wir Ihnen gerne unter service@joy-it.net.

# GNU GENERAL PUBLIC LICENSE

Version 2, June 1991



51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA



## Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

## Terms and conditions for copying, distribution and modification

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each l icensee is addressed as "you".
Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.
You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

    a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

    b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

    c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

    These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.
In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.
   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the

original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program. If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.
   It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
   This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
   Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**END OF TERMS AND CONDITIONS**

**How to Apply These Terms to Your New Programs**

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" lineand a pointer to where the full notice is found.

*one line to give the program's name and an idea of what it does.*
*Copyright (C) yyyy name of author*

*This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.*

*This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.*

*You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301, USA.*

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

*Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type `show w'. This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.*

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

*Yoyodyne, Inc., hereby disclaims all copyright interest in the program `Gnomovision'(which makes passes at compilers) written by James Hacker.*
*signature of Ty Coon, 1 April 1989*
*Ty Coon, President of Vice*

## GNU GENERAL PUBLIC LICENSE

Version 3, 29 June 2007 Copyright © 2007 Free Software Foundation, Inc. <https://fsf.org/> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

### Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works. The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too. When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things. To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others. For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights. Developers that use the GNU GPL protect your rights with two steps: (1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions. Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.
Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free. The precise terms and conditions for copying, distribution and modification follow.

## TERMS AND CONDITIONS

0. **Definitions.**

"This License" refers to version 3 of the GNU General Public License.
"Copyright" also means copyright-like laws that apply to other kinds of works, such as semiconductor masks. "The Program" refers to any copyrightable work licensed under this License. Each licensee is addressed as "you". "Licensees" and "recipients" may be individuals or organizations.
To "modify" a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a "modified version" of the earlier work or a work "based on" the earlier work.
A "covered work" means either the unmodified Program or a work based on the Program.
To "propagate" a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.
To "convey" a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.
An interactive user interface displays "Appropriate Legal Notices" to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. **Source Code.**

The "source code" for a work means the preferred form of the work for making modifications to it. "Object code" means any non-source form of a work. A "Standard Interface" means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.
The "System Libraries" of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A "Major Component", in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.
The "Corresponding Source" for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work's System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work

is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source. The Corresponding Source for a work in source code form is that same work.

2. **Basic Permissions.**
All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law. You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you. Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. **Protecting Users' Legal Rights From Anti-Circumvention Law.**
No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures. When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. **Conveying Verbatim Copies.**
You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program. You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions. You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

    a. The work must carry prominent notices stating that you modified it, and giving a relevant date.

    b. The work must carry prominent notices stating that it is released under this License and any conditions added under section 7. This requirement modifies the requirement in

section 4 to “keep intact all notices”.

c. You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d. If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation's users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. **Conveying Non-Source Forms.**
You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b. Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c. Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d. Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same

place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e. Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work. A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product. “Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made. If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM). The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of

the network or violates the rules and protocols for communication across the network. Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. **Additional Terms.**

"Additional permissions" are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions. When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a. Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b. Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c. Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d. Limiting the use for publicity purposes of names of licensors or authors of the material; or

e. Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f. Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying

under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying. If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms. Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination. You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11). However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation. Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice. Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies. You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients. Each time you convey a covered work, the recipient automatically receives a license from the original licensors, to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License. An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party's predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding

Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts. You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. **Patents.**

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor's “contributor version”. A contributor's “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License. Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor's essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version. In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party. If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient's use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid. If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it. A patent license is “discriminatory” if it does not include within the scope of

its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007. Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. **No Surrender of Others' Freedom.**
If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the
conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. **Use with the GNU Affero General Public License.**
Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. **Revised Versions of this License.**
The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.
If the Program specifies that a proxy can decide which future

versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program. Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. **Disclaimer of Warranty.**
THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. **Limitation of Liability.**
IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. **Interpretation of Sections 15 and 16.**
If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

## How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms. To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

*<one line to give the program's name and a brief idea of what it does.>*
*Copyright (C) <year> <name of author>*
*This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.*
*This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.*
*You should have received a copy of the GNU General Public License along with this program. If not, see <https://www.gnu.org/licenses/>.*

Also add information on how to contact you by electronic and paper mail. If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

*<program> Copyright (C) <year> <name of author>*
*This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.*
*This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.*

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an “about box”. You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary. For more information on this, and how to apply and follow the GNU GPL, see <https://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <https://www.gnu.org/licenses/why-not-lgpl.html>.

## X11 License (MIT License)

Copyright (C) 1996 X Consortium

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE X CONSORTIUM BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Except as contained in this notice, the name of the X Consortium shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Software without prior written authorization from the X Consortium.

X Window System is a trademark of X Consortium, Inc.

## Adafruit Python DHT Sensor Library

Python library to read the DHT series of humidity and temperature sensors on a Raspberry Pi or Beaglebone Black. Designed specifically to work with the Adafruit DHT series sensors ----> https://www.adafruit.com/products/385 Currently the library is tested with Python 2.6, 2.7, 3.3 and 3.4. It should work with Python greater than 3.4, too.

### Installing

Dependencies

For all platforms (Raspberry Pi and Beaglebone Black) make sure your system is able to compile and download Python extensions with pip:
On Raspbian or Beaglebone Black's Debian/Ubuntu image you can ensure your system is ready by running one or two of the following sets of commands:

Python 2:

```
sudo apt-get update
sudo apt-get install python-pip
sudo python -m pip install --upgrade pip setuptools wheel
```

Python 3:

```
sudo apt-get update
sudo apt-get install python3-pip
sudo python3 -m pip install --upgrade pip setuptools wheel
```

Install with pip

Use `pip` to install from PyPI.

Python 2:

```
sudo pip install Adafruit_DHT
```

Python 3:

```
sudo pip3 install Adafruit_DHT
```

Compile and install from the repository

First download the library source code from the GitHub releases page, unzipping the archive, and execute:

Python 2:

```
cd Adafruit_Python_DHT
sudo python setup.py install
```

Python 3:

```
cd Adafruit_Python_DHT
sudo python3 setup.py install
```

You may also git clone the repository if you want to test an unreleased version:

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

Usage

See example of usage in the example folder.

Author

Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!

Written by Tony DiCola for Adafruit Industries.

MIT license, all text above must be included in any redistribution

## DEPRECATED LIBRARY.Adafruit Python CharLCD

This library has been deprecated! We are leaving this up for historical and research purposes but archiving the repository.
We are now only supporting the use of our CircuitPython libraries for use with Python.
Check out this guide for info on using character LCDs with the CircuitPython library:
https://learn.adafruit.com/character-lcds/python-circuitpython

### Adafruit_Python_CharLCD

Python library for accessing Adafruit character LCDs from a Raspberry Pi or BeagleBone Black. Designed specifically to work with the Adafruit character LCDs ----> https://learn.adafruit.com/character-lcds/overview
For all platforms (Raspberry Pi and Beaglebone Black) make sure you have the following dependencies:

```
sudo apt-get update
```

For a Raspberry Pi make sure you have the RPi.GPIO library by executing:

```
sudo pip install RPi.GPIO
```

For a BeagleBone Black make sure you have the Adafruit_BBIO library by executing:

```
sudo pip install Adafruit_BBIO
```

Install the library by downloading with the download link on the right, unzipping the archive, navigating inside the library's directory and executing:

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

See example of usage in the examples folder.

Adafruit invests time and resources providing this open source code, please support Adafruit and open-source hardware by purchasing products from Adafruit!

Written by Tony DiCola for Adafruit Industries.
MIT license, all text above must be included in any redistribution

## 9. SUPPORT

Wir sind auch nach dem Kauf für Sie da. Sollten noch Fragen offen bleiben oder Probleme auftauchen stehen wir Ihnen auch per E-Mail, Telefon und Ticket-Supportsystem zur Seite.

E-Mail: service@joy-it.net
Ticket-System: http://support.joy-it.net
Telefon: +49 (0)2845 98469 – 66 (10 - 17 Uhr)

Für weitere Informationen besuchen Sie unsere Website:
**www.joy-it.net**

Veröffentlicht: 29.01.2020