Projektbeschreibung Light Painting

Beschreibung des Raspberry-Projekts „Light Painting“ mit dem Raspberry Pi.

Weitere Fotos auf meiner Foto-Website

Inhaltsverzeichnis

1. Raspberry Pi Projekt

Der Raspberry Pi (genannt Pi) ist ein kreditkarten-kleiner und preisgünstiger (45 CHF) Linux Computer der sich seit seiner Einführung vor bald 4 Jahren grosser Beliebtheit erfreut und der bereits mehrere Millionen Mal verkauft wurde. Im Internet finden sich unzählige Projektbeschreibungen; alles Mögliche und Unmögliche wird mit diesem Micro-Computer gebaut. Der Initiant Eben Upton ist Engländer und in folgendem Video auf YouTube erklärt die Geschichte des Pi.

Eben Upton on YouTube

Der Pi hat 4 USB-Anschlüsse, einen Lan-Anschluss, ein HDMI-Anschluss für Monitor und einen Audio-Video-Ausgang. Ebenfalls auf dem Board ist ein Anschluss für die Pi-Camera - eine HD-fähige Kleinst-Camera für Fotos und Video. Was den Pi auszeichnet, ist die 40-polige Steckerleiste. Viele dieser Anschlüsse sind General Purpose Input/Output Pins, welche in Programmen angesteuert/gelesen werden können. Es gibt in der Zwischenzeit vielfältige Expansion Boards von unzähligen Anbietern. Der Pi konsumiert bloss 1 Watt Leistung: 5V und 200mA Strom.

Nach den Projekten **Funksteckdosen** (Switcher) und **Christmas TV** ist Light Painting bereits das dritte Pi Projekt.

Der Raspberry PI Model B+

2. Idee Pixelstab

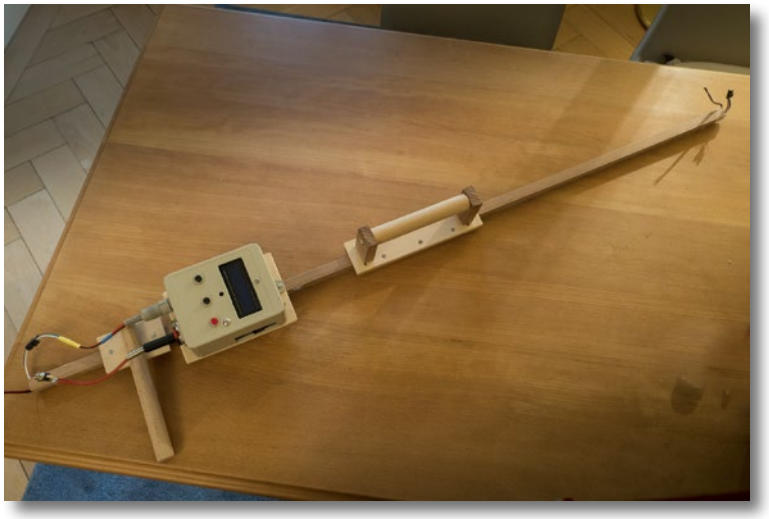
Light Painting, also Malen mit Licht, ist eine Ausdrucksform, die verschiedene Ausprägungen hat. Meist handelt es sich um photographische Aufnahmen mit langer Belichtungszeit (3-30 Sekunden). Auf der einen Seite gibt es Fotografen, die nächtliche, unbewegte Szenerien mit Lampen beleuchten. Andere verwenden bewegte Lichtquellen, um damit zu ‚zeichnen'. Absicht ist es immer, mit nur einer Belichtung (also keine Doppelbelichtung und kein Photoshopen), künsterlische Werke zu schaffen.

Seit einiger Zeit sind Led-Strips erhältlich, die bis zu 144 individuell ansteuerbare RGB-Led's pro Meter haben. Damit sind höher aufgelöste ‚Zeichnungen' möglich und dies ist die Essenz dieses Projektes. Verwendet werden die als **NeoPixel** bezeichneten Produkte der Firma Adafruit aus New York.

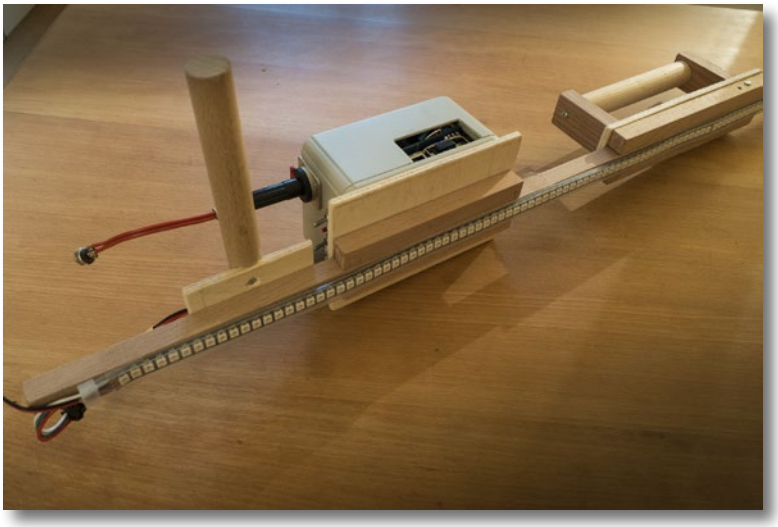
Ziel: Es soll ein Lichtstab mit einem Meter Länge - also 144 RGB-Pixel - gebaut werden und die Steuerung soll mittels Raspberry Pi erfolgen.

3. Fertige Lösung

Der fertige Pixelstab sieht so aus



Pixelstab mit Griffen und Steuerungs-Box

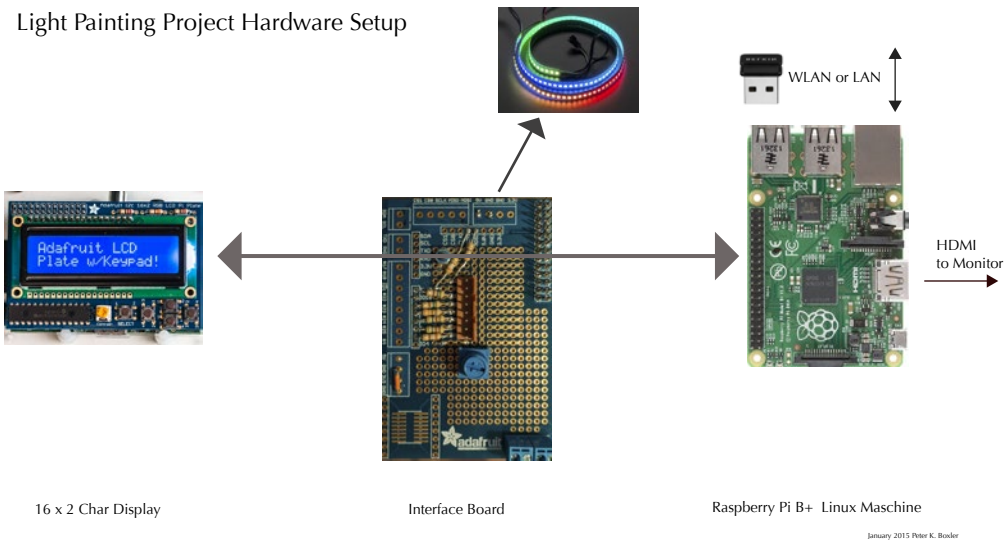


144 RGB-Pixel auf einem Meter Länge

Pixelstab mit Regenbogenfarben

4. Hardware Aufbau

4.1 System Aufbau



Anordnung der Komponenten

4.2 Lichstab

Der NeoPixel-Streifen mit 144 Pixeln und einem Meter Länge ist auf einem 15x15mm Buchenstab mon-

tiert. Die Control Box und Handgriffe sind ebenfalls auf dem Stab befestigt.

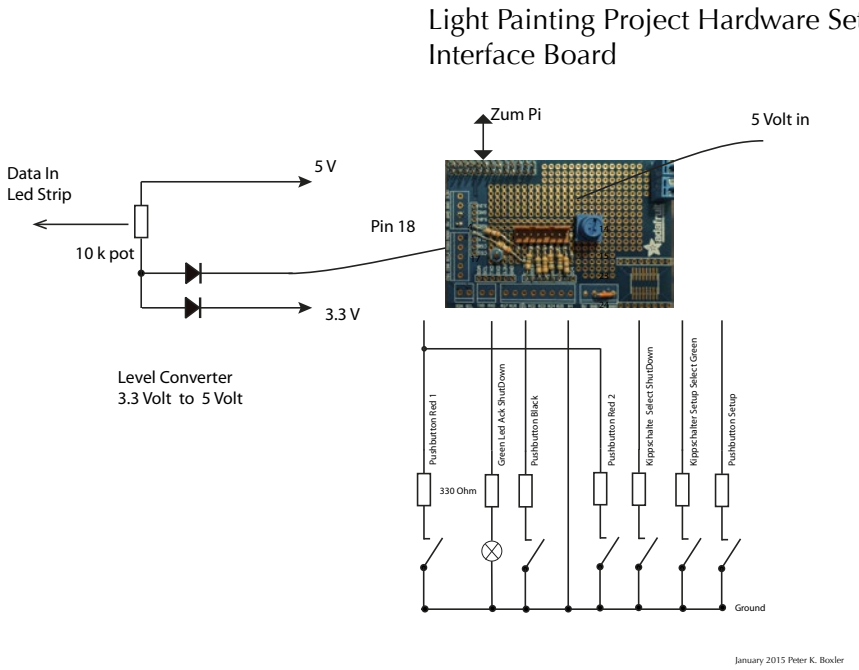
4.3 Control Box

Die Control Box beinhaltet die Steuer-Elektronik, bestehend aus dem Raspberry Pi, dem Interface Board und dem 16x2 Char Display-Board. Diese 3 Komponenten sind stacked - also alle aufeinander gesteckt. Die Box hat links und rechts Öffnungen für USB und LAN-Anschluss und für Wechsel der Micro-SD Karte mit dem Linux OS.

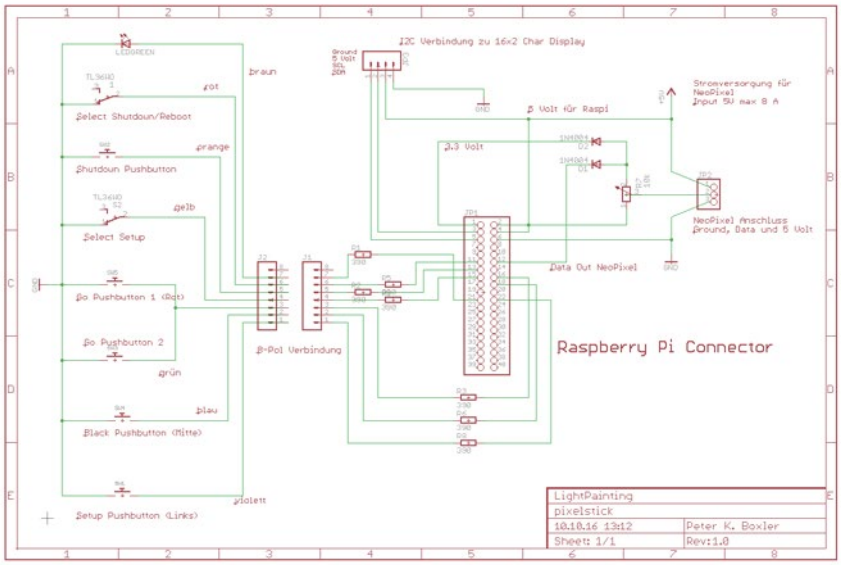
Das User Interface besteht aus dem 16x2 Char Display sowie 5 Drucktasten (Momentankontakt) und 2 Kippschaltern. Die 5 V Stromversorgung wird via 2-poligen Stecker angeschlossen. Die 3 Leitungen zum Led-Pixelstick (5 Volt, Ground, Data In) werden über einen 3-poligen DIN-Stecker an die Box angeschlossen.

Die GPIO-Pins des Raspberry Pi haben einen Pegel von 3.3 Volt; der Led-Pixelstrip braucht für die serielle Ansteuerung (DATA-IN) einen Pegel von 5 Volt. Die Pegelwandlung wird durch 2 Dioden und ein Potentiometer erledigt. GPIO Pin 18 wird dafür benötigt. Pin-Belegung des Pi siehe Tabelle weiter hinten. Beschreibung des User Interface siehe weiter hinten.

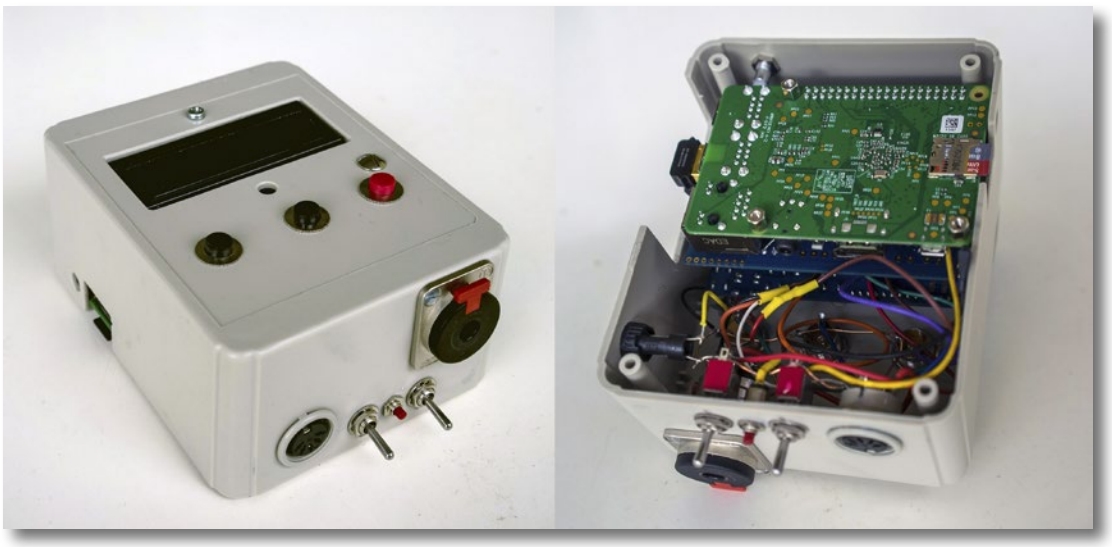
Die Box sitzt auf dem 15x15mm Buchenstab.



Uebersicht Interface Board



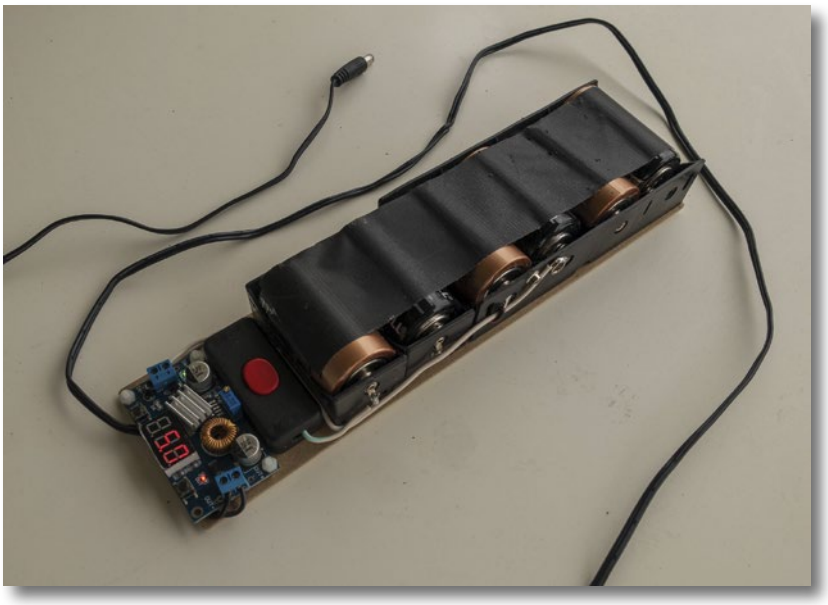
Schaltschema der Raspi-externen Komponenten (siehe sparates pdf)



Control Box

4.4 Stromversorgung

Ein einzelner Pixel benötigt bei der Farbe Weiss (alle 3 Led's leuchten) max. 60 mA Strom bei 5 Volt. Für den ganzen Streifen von 144 Pixeln ist demnach eine Stromversorgung mit ca. 8 Ampère Maximalstrom bereitzustellen. Für den portablen Betrieb dienen 6 Batterien vom Typ D (6 x 1.5 Volt) und ein Spannungsregler für den nötigen Saft.



Portable 5 Volt Stromversorgung

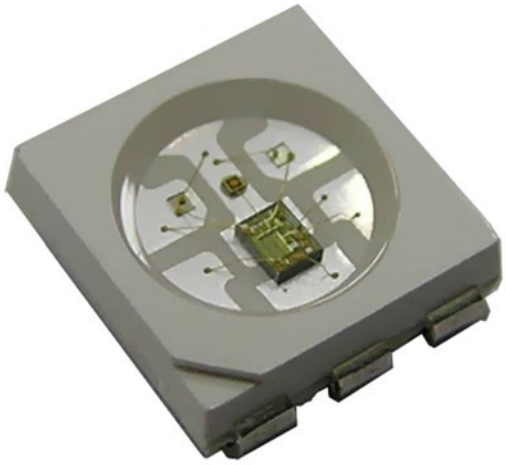
4.5 Pin Belegung Raspberry Pi

Der verwendete Raspberry Pi Modell B+ hat 40-Pins, es werden jedoch nur die ersten 26 Pins verwendet. In der Software werden folgende Pins verwendet.

Pin	GPIO	Richtung	Verwendung	Kabel Farbe (Foto)
11	17	IN	SoftShut Down Kippschalter, für reboot oder halt	rot
12	18	OUT	Serielle Ansteuerung Led Strip	weiss
13	27	IN	Soft Shut Down Pushbutton	orange
16	23	IN	Roter Pushbutton (rechts)	grün
18	24	IN	Schwarzer Pushbutton (mitte)	blau
22	25	IN	Setup Pushbutton (links)	violett
21	9	OUT	Grüne LED	braun
15	22	IN	Setup Kippschalter Welcher Setup soll gemacht werden	gelb

4.6 RGB Led‘s

Die verwendeten Led sind vom Typ WS2812 Intelligent control LED integrated light source. Jeder einzelne Pixel (des Streifens) besteht aus einem Element WS2812. Ein solches beinhaltet 3 Led (Rot, Grün und Blau) sowie einen kleinen Controller, der die Led‘s steuert. Grösse ca. 5x5mm.
Weitere Info siehe Datenblatt .



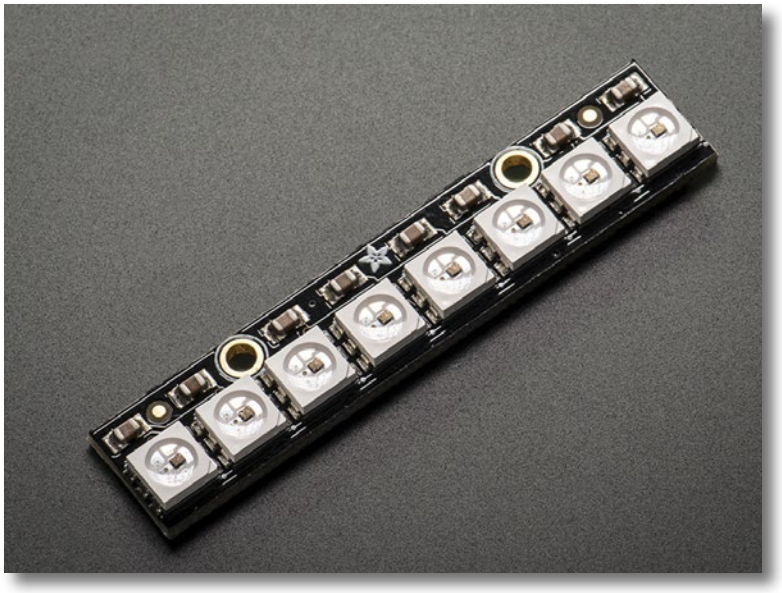
Einzelnes Element WS2812

Für einen Streifen mit 144 solcher Pixel-Elemente werden nur 3 Anschlüsse benötigt: Ground, 5 Volt und Steuerleitung (Data In). Es sind Farbwerte von 0-255 je für Rot, Grün und Blau möglich; total also 16 Mio Farben.

Die Ansteuerung eines Streifens mit 144 RGB-Led's erfolgt also seriell asynchron (ohne clock) über einen einzigen Draht. Das Timing der Pulse ist relativ streng definiert (im Bereich Microsekunden) und ein Linux-Computer ist als Nicht-Real-Time-Machine dafür wenig geeignet. Kluge Köpfe haben jedoch für dieses Problem eine Lösung gefunden: Adafruit bietet eine Library an (rpi_ws281x), mit welcher sog. Neopixel-Strips aus Python-Scripten in einem Pi angesteuert werden können. Diese Library ist die Grundlage der vorliegenden Lösung.

Hier Info zu den NeoPixel Produkten der Firma Adafruit.

Der **NeoPixel Ueberguide** von Adafruit gibt sehr gute Einführung in diese genialen RGB-Led's. Siehe Links im Anhang.



Streifen mit 8 RGB-Led WS2812

Led Strip mit 144 Pixeln - Adafruit Product 1507

5. Software

5.1 Voraussetzungen

Die im folgenden beschriebenen LP-Scripts setzen folgendes voraus

- der Adafruit Code für 16x2 Char Plate muss im aktuellen Folder vorhanden sein.
- die Adafruit Neopixel Library (rpi_ws281x) muss installiert sein (Ansteuerung der Pixel).
- Die Python Image Library Pillow muss installiert sein.

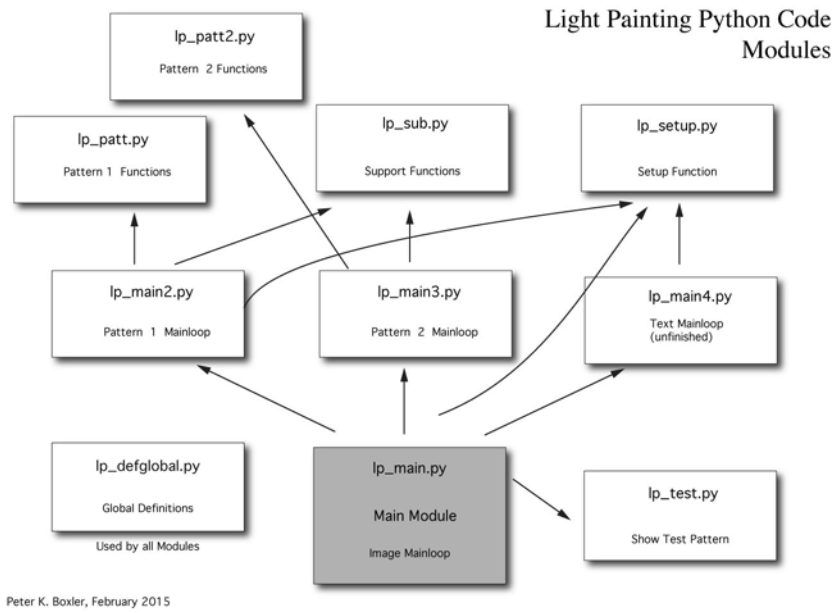
5.2 Light Painting Programm

Die verwendete Script-Sprache ist Python. Das Hauptprogramm (Script) heisst **lp_main.py**. Es ist in /etc/rc.local eingetragen und wird nach den Boot gestartet.

Beim Start des Programms können folgenden Commandline-Parameter angegeben werden:

```
-d          kleiner debug, Statusmeldungen werden ausgegeben (stdout)
-D          grosser debug, weitere Statusmeldungen
-g wert     Gamma Wert
```

Die Programm-Struktur der verwendeten Programm-Teile sieht so aus. Erläuterung zu den einzelnen Funktionen siehe weiter hinten.



Module Struktur LIght Painting

Das Light Painting-Projekt besteht aus ca. 3600 Zeilen Python Code, der in folgende Module strukturiert ist:

Script Name	**Funktion**
lp_main.py	Hauptprogramm Light Paint Beinhaltet Main Loop für Draw Pictures Wird gestartet in /etc/rc.local
lp_defglobal.py	Include mit Definition globaler Variablen
lp_main2.py	Main-Module für Draw Pattern 1
lp_main3.py	Main-Module für Draw Pattern 2
lp_main4.py	Main-Module für Draw Text
lp_patt.py	Generierung Pattern 1
lp_defpattern.py	Definitionen Pattern 1
lp_patt2.py	Generierung Pattern 2
lp_defpattern2.py	Definition Pattern 2
lp_sub.py	Allgemeine Functions
lp_test.py	Testpattern LED, nach Start aufgerufen
lp_setup.py	Setup Funtionen).
lp_softshut.py	Soft Shut Down Script, gestartet in /etc/rc.local
clear.py	Programm zu löschen aller Led's.
coltest1.py	Testprogramm für Farbtest
coltest2.py	Testprogramm für Farbtest
coltest3.py	Testprogramm für Farbtest
coltest4.py	Testprogramm für Farbtest
coltest5.py	Testprogramm für Farbtest

coltest6.py	Testprogramm für Farbtest
usb_read.py	Testprogramm für Lesen Dateien ab USB-Stick Aufrufen mit debug option sudo python usb_read.py -d

Der gesamte Code kann ab meiner Dropbox geladen werden (t.b.d.)

6. Shutdown Pi

Für den ordentlichen Shutdown des Pi sorgt das Script **lp_softshut.py**. Es ist ebenfalls in /etc/rc.local eingetragen und wird damit nach dem Boot gestartet. Dieses Script prüft continuierlich, ob Pin GPIO27 auf Null geht - bei Tastendruck auf die kleine rote Taste an der Unterseite der Control Box. Die Led an GPIO Pin 9 blinkt dann dreimal (ack).

In Abhängigkeit der Stellung des Kippschalters 2 (GPIO 17) wird entweder:

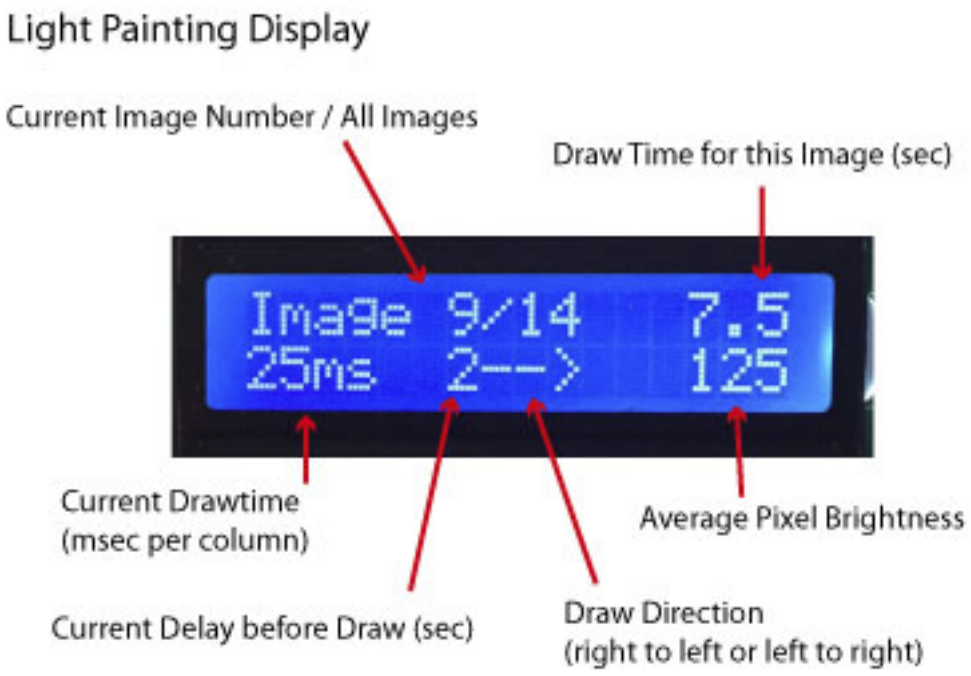
- Der Pi wird heruntergefahren mit dem Befehlt halt.
- Es wird ein Reboot ausgeführt, Befehl reboot

7. User Interface

Das User-Interface des Lichtstabs besteht aus einen Adafruit 16x2 Character LCD 2x16 (Product Number 1115) Auswahl der verschiedenen Funktionen werden über 5 Pushbuttons und 2 Kippschalter vorgenommen.

Weitere Statusmeldungen werden über eine grüne LED kommuniziert.

Die Anzeige auf dem 16x2 Display sieht etwa so aus. Gezeigt ist die Variante 1 (Draw Images):



User Interface Anzeige

Note:

Average Brightness ist nicht implementiert.

Eingaben und Selektionen des Benutzers werden durch Pushbuttons und Kippschalter vorgenommen:

Schalter	**Funktion**
Kippschalter 1	Steuert Funktion des Setup Pushbuttons: Setup Parameter oder Choose Drawing Type.
Kippschalter 2	Steuert Funktion des Soft Shut Down Pushbuttons: System Halt oder System Reboot.
Grüner Pushbutton	Aufruf des Setup. Abhängig von Kippschalter 1 wird aufgerufen: - Auswahl der Hauptfunktion: Draw Images, Draw Pattern 1, Draw Pattern 2. - Setup diverse Parameter
Schwarzer Pushbutton	Diverse Funktionen, ja nach System Status
Roter Pushbutton 1	Diverse Funktionen, je nach System Status. Haupt-funktion: Start Drawing.
Roter Pushbutton 2	Identisch mit roten Pushbutton 1 aber an der Seite der Box angeordnet.
Roter Pushbutton klein	Soft Shut Down, abhängig von Kippschalter 2 wird entweder: - System angehalten sudo halt - System reboot
Grüne LED	Blinkt 3 mal, wenn Soft Shut Down erkannt wird

8. Funktionen des Lichstabs

8.1 Draw Image

Diese Funktion ‚zeichnet' Bilder. Diese Bilder müssen am Computer vorbereitet werden und müssen eine Höhe von 144 Pixeln haben (Lichtstab hat 144 Led's). Die Breite ist variabel, sinnvoll sind 200 bis 1000 Pixel. Die Bilder müssen auf einem USB Stick im Ordner images gespeichert werden; ebenfalls möglich ist, die Fotos im Linux Filesystem zu abzulegen. Möglich sind jpg, png oder Gif.

Diese Bilder werden ‚gezeichnet' indem die Kolonnen (1 Kolonne entspricht einem Pixel der Bildbreite) sukzessive auf die 144 Pixel (Bildhöhe) ausgegeben werden. Das Intervall ist dabei 20 ms - dies ist aber im Setup konfigurierbar von 5ms bis 50ms. Mit anderen Worten: eine Kolonne ist sichtbar für 20 ms.

Beispiel: ein Bild der Grösse 500 Pixel Breite und 144 Pixel Höhe wird also in 500 x 20ms = 10 Sekunden ‚gezeichnet'. Je nach Laufgeschwindigkeit kann das Bild breiter/schmäler werden.

Das User Interface erlaubt die Weiterschaltung von Bild zu Bild (round robin), dabei wird für jedes Bild die ‚Schreibzeit' in Sekunden angezeigt. Dies ist wichtig für die Wahl der Verschlusszeit beim Fotografieren.

8.2 Funktion Setup

Diese Funktion erlaubt die Änderung folgender Parameter:

- Sekunden Wartezeit vor Beginn Zeichnen (nach Druck auf roten Pushbutton)
- Millisekunden Intervall (Leuchtzeit einer Kolonne)
- Helligkeit der Led‘s, Wert zwischen 50 und 255 möglich
- Richtung des ‚Zeichnens‘ der Bilder; links nach rechts oder rechts nach links
- Anzahl Iterationen beim Zeichnen von Pattern (nicht relevant bei Bildern)
- Gamma Wert für Korrektur der Helligkeit (Led‘s haben ziemlich lineare Transferfunktion, das Auge hat nichtlineare Rezeptionsfähigkeit.

8.3 Draw a Pattern 1

Diese Funktion erlaubt das ‚Zeichnen‘ von verschiedenen Mustern, die alle in Real Time durch Programmcode erzeugt werden. Es gibt verschiedene Arten von Regenbogen-Mustern, es gibt Lauflichter, und und und.

8.4 Draw a Pattern 2

Diese Funktion erlaubt das Zeichnen von vordefinierten Mustern, die durch Programmcode interpretiert und gezeichnet werden. Es stehen dabei mehrere verschiedene Farbdefinitionen zur Verfügung - diese sind ebenfalls definiert und erlauben Farbvariation clockwise und anticlockwise auf dem Farbrad. Innerhalb eines Musters kann eine Farbe clockwise verändert werden, einen andere Farbe anticlockwise.

8.5 Draw Text

Diese Funktion ist noch nicht voll realisiert - sie funktioniert im Testmodus, wo im Compi eine Tastatur vorhanden ist: sie ‚zeichnet‘ einen eingegebenen Text mit einem wählbaren Font. Wird der Lichtstab portable verwendet, so ist zur Zeit keine Tastatur möglich. Braucht noch weitere Abklärungen.

8.6 Generierung von Farben

Zur Generierung von RGB-Farbwerten ist eine Funktion color_wheel () implementiert, die auf dem Farbrad 1536 verschiedene Farben des Regenbogens erzeugen kann. Obwohl mit je 0-255 Werten für R,G und B maximal 16 Mio. Farben möglich sind, sieht man nach einiger Überlegung schnell, dass bei Variation (Einerschritte) aller 3 Farben R,G und B nur maximal 1036 kontinuierliche Farben des Regenbogens erzeugt werden können. Alle weiteren anderen Farben sind selbstverständlich möglich, befinden sich jedoch auf anderen Farbkreisen (mit anderen Luminanzen).

Code der Funktion siehe Anhang.

Diese Funktion wird in den übrigen Modulen (Generierung von Pattern) sehr oft verwendet.

9. Ansteuerung der RGB Led‘s

Mittels der Library **rpi_ws281x**, welche von Adafruit zur Verfügung gestellt wird, ist das Ansteuern der einzelnen Pixel sehr einfach. Das Application-Program-Interface (API) kennt nur wenige Funktionen. Als Beispiel hier der Pseudo Code zum Setzen aller 144 Pixel des Stabes auf Farbe Rot (255,0,0). Man sieht, dass der Code sehr simpel ist.

(strip ist dabei das vorgängig allozierte Objekt, das den Strip kennzeichnet).

```
colo = Color ( 255 , 0 , 0 )            # Funktion Color erstellt 24 bit Farbwert
       for z in range(144)  {           # läuft von 0 bis 143, Anzahl Pixel auf dem Stab
       strip.setPixelColor ( z , colo ) # set Pixel z auf gewünschte Farbe)
       }
       strip.show()                     # Abschluss, Led Controller schalten Led‘s ein
```

10. Beispiele Light Painting Aufnahmen

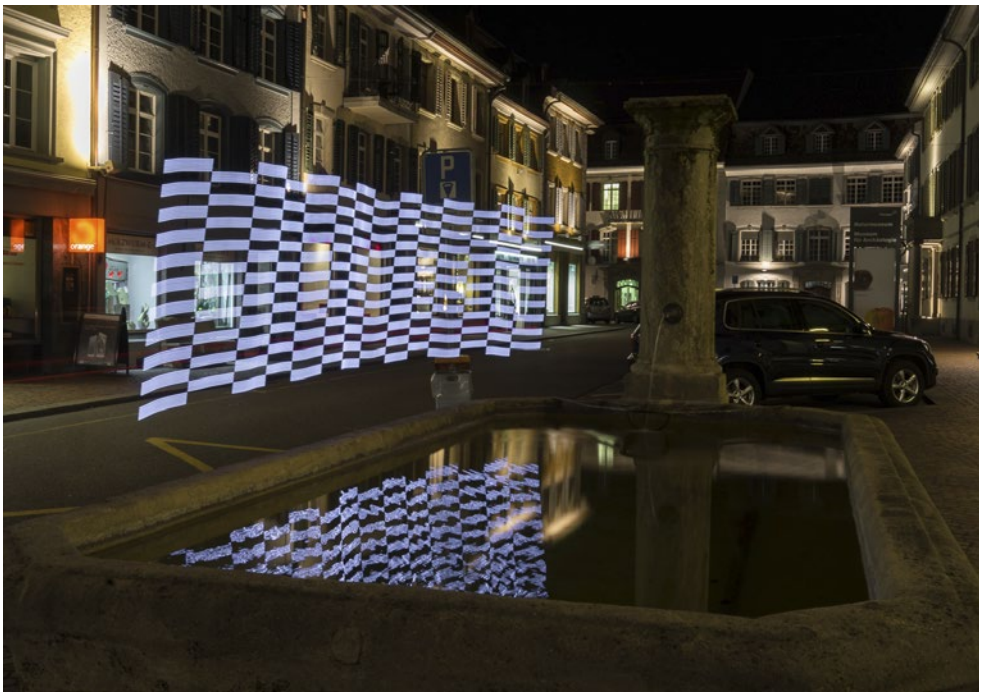
Bei allen Aufnahmen wurde eine Belichtungszeit von 20 Sekunden verwendet, Blende 16 bei ISO 100. Kamera Sony A7, 35 mm Objektiv.

Bild 144 x 400 Pixel, weisse Schrift auf schwarzem Hintergrund

Muster, durch Programm-Code interpretiert

Regenbogen 1536 Farben

Light Paint



Muster, in Real-Time erzeugt

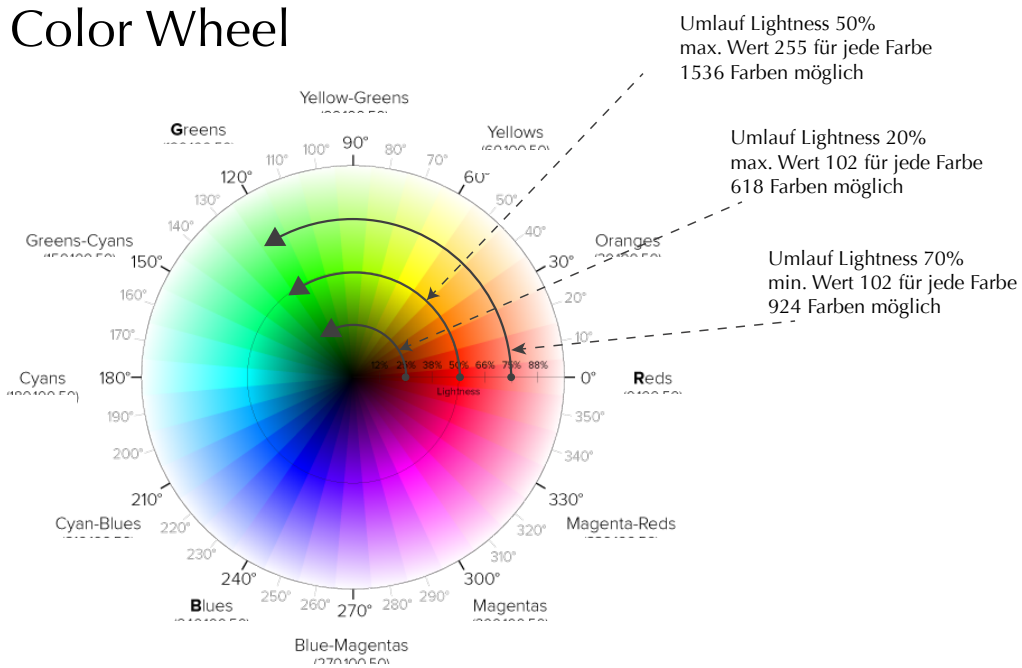
11. Anhang

11.1 Funktion color_wheel ()

Die Funktion color_wheel implementiert folgende Tabelle - diese zeigt den Verlauf von R, G und B beim Lauf um das Farbrad. Die Tabelle zeigt gesättigte Farben - eine der drei Farben hat den Wert 255.

Grad	**Position**	**Sechstel**	**Rot-Wert**	**Grün-Wert**	**Blau-Wert**	**Farbe Anfang/Ende**
0-59	0 bis 255	erster	fix 255	0 bis 255	fix 0	rot / gelb
6 bis 119	256 bis 511	zweiter	255 bis 0	fix 255	fix 0	gelb / grün
120 bis 179	512 bis 767	dritter	fix 0	fix 255	0 bis 255	grün/mangenta
180 bis 239	768 bis 1023	vierter	fix 0	255 bis 0	fix 255	mangenta/blau
240 bis 299	1024 bis 1279	fünfter	0 bis 255	fix 0	fix 255	blau/violett
300 bis 360	1280 bis 1536	sechster	fix 255	fix 0	255 bis 0	violett/rot

Dieses Bild zeigt die Grundlage



Die Funktion **color_wheel (pos , luminance)** akzeptiert 2 Parameter:

- Position auf dem Farbrad, Werte von -1536 bis _1536 möglich
- Luminanz, 10% - 90% in Zehnerschritten. Defaultwert ist 50%, dabei ist die maximale Anzahl Farben (1536) möglich.

Rückgabe sind RGB-Werte im Bereich 0-255.

```
#--------------------------------------------------------------------
# Generate 1536 rgb Colors around the colorwheel
# Starting with RED
#--------------------------------------------------------------------
#   parameters: position on colorwheel, luminace 10-90 %
#   loop needs to be outside the function (see example)
#   pos indicates postion on the colorwheel
#   luminance (lightness) can be 10,20,30,40,50,60,70,80,90 %
#   luminance 50% gives max. number of colors : 1536
#   the loopcounter has to set like this:
#   while True
#
#   returns -99 if max. number of colors reached or wrong luminance
#
#   Peter K. Boxler, February 2015
# --------------------------------------------------------------------
def color_wheel_r (posin , lum=50):

    colval_min=[255,0,0,0,0, 0 , 51,102,153,204,255]      # rbg values max and min
    colval_max=[0   , 51,102,153,204,255,255,255,255,255,255]      # rbg values max and min

    count=[0,312,618,924,1230,1536,1230,924,618,312,0]  # loop counter limit based on lum
    retu=0

    if lum<10 or lum>90:        # must be between 10 and 90
        return (-98,[0,0,0])
```

```
pos=posin                    # keep colorwheel pos input
if pos >= count[lum/10]:     # signal that we reached max number of colors for this lumin
    pos=pos-count[lum/10]
    retu=-99                 # signal overflow to caller (but continue after correction)
if pos < 0:
    pos=count[lum/10]-abs(pos)         # ???????????????
    retu=-99
start=colval_min[lum/10]     # set min values for r,g and b  based on luminace
max=colval_max[lum/10]       # set max values for r,g and b
z=max-start+1

if pos < z:                  # first sixth of wheel
    return(retu,[max, pos+start, start])          # red=max, green increases, blue=0

elif pos < (2 * z):          # second sixth of wheel
    pos -= z
    return(retu,[max-pos, max, start])           # red decreases, green max, blue=0

elif pos < (3 * z):           # third sixth of wheel
    pos -= 2 * z
    return(retu,[start, max, pos+start])        # red=0, green max, blue=increases

elif pos < (4 * z):           # fourth sixth of wheel
    pos -= 3 * z
    return(retu,[start, max-pos, max])          # red=0, green=decreases, blue=max

elif pos < (5 * z):            # fifth sixth of wheel
    pos -= 4 * z
    return(retu,[pos+start, start, max])        # red increases, green=0, blue=max

else:                           # last sixth of wheel
    pos -= (5 * z)
    return(retu,[max, start, max-pos])          # red=max, green=0, blue decreases
```

11.2 Betrieb des Lightpainting

Die Controlbox hat:

- 1 Led
- 5 Drucktasten
- 2 Kippschalter
- Ein 16x2 Char Display (Adafruit CharPlate Product 1115)

Led

Die Led (lp_defglobal.led_green, GPIO Pin 9) wird im Hauptscript verwendet. Sie zeigt an, dass Bild-Vorbereitung abgeschlossen ist, gleichzeitig wird auf dem Display ready gezeigt: also ready to draw. Nach Ende des Init blinkt diese Led zudem dreimal: Init fertig.

Sie wird ebenso im Script lp_shoftshut.py verwendet: sie blinkt dreimal beim Start des scripts und dreimal, wenn die Shutdown- Taste gedrückt wird.

Siehe Kapitel Start des Pi

Folgendes kann in der Datei lp_defglobal.py eingestellt werden:

Variable painting_type: definiert, mit welchem Typ Lightpainting gestartet werden soll. Es gibt diese 4

Typen:

- type= 1 Zeichen Bilder, die auf USB oder im lokalen Folder gefunden werden, wird ind er Funktion main_loop_type1() ausgeführt (ist im Script lp_main.py)
- type=2 Zeichne Muster, es wird die Funktion main_loop_type2() aufgerufen (ist im Script lp_main2.py)
- type=3 Zeichne Muster, es wird die Funktion main_loop_type3() aufgerufen (ist im Script lp_main3.py)
- type=4 noch nicht fertig codiert, ist vorgesehen für Text-Ausgabe (aber da keine Tastatur angeschlossen ist, geht dies vorläufig nicht.

Drucktasten

Die Drucktasten haben folgende Funktion (GPIO-Pin in lp_defglobal definiert):

- lp_defglobal.red_button: hat verschiedene Funktionen
- lp_defglobal.black_button: hat verschiedene Funktionen
- lp_defglobal.setup_button: Setup wird aufgerufen (ja nach Stellung des Kippschalters 1)

Kippschalter

Die 2 Kippschalter haben folgende Funktion

- Wahl des Setup: lp_defglobal.switch_type_pin (GPIO-Pin in lp_defglobal.py definiert): wird in Script lp_sub.py verwendet, dieser Schalter definiert, was passieren soll, wenn die Drucktaste lp_defglobal.setup_button gedrückt wird: in der Funktion do_setup(GPIO) in Script lp_setup.py) wird entweder ein neuert Lightpainting Typ ausgewählt oder der Setup für das Zeichnen der Bilder wird ausgeführt (wartezeit vor zeichnen, Millsec für eine Kollonne). Die Stellung dieses Kippschalters definiert also den Typ des Setups, der ausgeführt wird, wenn die Setup-Drucktaste gedrückt wird. In einer neuen Version täte ich 2 Drucktasten dafür vorsehen.
- Wahl des Shutdowns: Der andere Kippschalter wird im Script lp_softshut.py verwendet (Variable switch_type_pin), dieser definiert, ob bei Druck auf den Shutdown Button (ebefalls in diesem Script) eine halt oder ein reboot ausgelöst wird.

Funktionen der Drucktasten:

- Rote Drucktaste (es gibt 2 davon, sind parallel geschaltet, eine Oben und eine auf der Seite). Variable lp_defglobal.red_button.
  - Nach dem Start: bestätigen der Init-Anzeige
  - Wählen (nach dem Start) ob Test-Muster angezeigt werden sollen, rot: ja, schwarz: nein
  - Beim Bilder zeichnen: angezeigte Bildnummer vorbereiten, es wird ready angezeigt, bei nochmligen Druck wird gezeichnet
  - Beim Muster zeichnen: starte Muster-Display
- Schwarze Drucktaste , Variable lp_defglobal.black_button
  - Beim Bilder zeichnen: wählt nächstes Bild
  - Beim Muster zeichnen: wählt nächstes Muster
- Grüne Drucktaste (ist auch schwarz, da ich keine grüne verfügbar hatte): wählt den Setup, gemäss obiger Beschreibung

11.3 Start des Pi

Die Datei /etc/rc.local sieh so aus, es wird hier das Script lp_softshut.py und auch das Main-Script des Lightpainting gestaret (sudo nano /etc/rc.local)

```
#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will „exit 0“ on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Softshut down
python /home/pi/bin/lp_softshut.py &
#
python /home/pi/lp/lp_main.py &
#
# Print the IP address
_IP=$(hostname -I) || true
if [ „$_IP“ ]; then
  printf „My IP address is %s\n“ „$_IP“
fi

exit 0
```

11.4 Ablauf nach Start des Pi

Nach Einstecken des Netzteils wird der Pi gebootet. Durch die Einträge in /etc/rc.local werden folgende Scripts gestartet:

- lp_main.py Hauptprogramm Lightpainting
- lp_softshut. py Shutdown Script

Im Wesentlichen wird im Init des Main-Scripts zuerst versucht, Bild-Dateien zu finden: erst auf einem ev. vorhandenen USB-Stick (Folder images) oder im lokalen Ordner /home/pi/lp/images.

Dann erfolgt die erste Ausgabe auf dem Display:

Erste Anzeige nach Start: Anzahl Bilder gefunden und wo

Ein Druck auf die rote Taste (GPIO Pin 23) lässt das Script weiterlaufen und es erscheint die nächste Ausgabe:

Test-Muster gewünscht: Schwarze Taste NO, Rote Taste YES

Abhängig der Auswahl wird nun entweder direkt weitergefahren im Script oder es werden die Test-Muster auf dem Led-Stick ausgegeben - dient der Funktionskontrolle.

Normalerweise verzweigt das Main-Script dann zur Funktion ‚Zeichne Bilder'. Dies kann aber mit der Variable painting_type (in lp_defglobal.py) : definiert werden, normaler Wert ist 1.

Dann erscheint der normale Display für das Zeichnen von Bildern

Ausgabe beim Zeichnen von Bildern

- Druck auf rote Taste bereitet Bild vor, dann erscheint ‚ready' und erneuter Druck auf rote Taste startet das Zeichnen des aktuellen Bildes (nach einer Verzögerung von (im Beispiel) 2 sec. Das Bild wird von links nach rechts gezeichnet.
- Druck auf schwarze Taste bringt immer das nächste Bild

- Druck auf Setup-Taste bringt entweder Setup-Menu 1 oder Setup-Menu 2 - je nach Stellung des Kippschalters 1

11.5 Setup des Raspi OS für Lightpainting

Im Dezember 2015 wurde eine neue SD-Karte erstellt, basierend auf den aktuellen Raspian Wheezy vom Mai 2015.

Downloads der aktuellen Raspi-OS Software hier:

Raspian Downloads

Folgende Schritte wurden dabei durchführt. Nach Abschluss liefen die LP-Scripte problemlos.

Nachdem neuestes Image auf Karte geschrieben wurde, werden mit raspi-config die wichtigsten Einstellungen angepasst.

Achtung: für neuere Versionen des OS können andere Schritte notwendig sein !!

Danach die notwendigen Packages laden, folgende Liste zeigt die Aktivitäten

```
sudo apt-get update
ev. auch
sudo apt-get upgrade    (aber bei ganz neuem image nicht nötig)

sudo apt-get install python-smbus

Installieren der Python Image Library

sudo apt-get install python-dev
sudo apt-get install python-pip
sudo pip install Pillow                         gibt Fehler, deshalb zuerst dies
sudo apt-get install libjpeg-dev
sudo pip install -I pillow                      reinstall Pillow


Adafruit Neopixel Library installieren

sudo apt-get install build-essential python-dev git scons swig

git clone https://github.com/jgarff/rpi_ws281x.git
cd rpi_ws281x
scons

cd python
sudo python setup.py install



USB-Stick einbinden
_______________

Zuerst Mount Point für USB Stick erstellen:

Ordner im Verzeichnis /media anlegen, in den das USB-Speichermedium später eingebunden wird
(Mountpoint genannt). Name soll sein usb1


sudo mkdir /media/usb1
```

```
sudo chown pi:pi /media/usb1


dann noch fstab erweitert, damit USb-Stick bei jedem Start des Pi eingebunden wird, diese
Zeile eingefügt:

/dev/sda1       /media/usb1     vfat    rw,defaults     0       0

Manueller mount für Fat32 Filesystem, wäre so:

sudo mount -t vfat -o uid=pi,gid=pi /dev/sda1 /media/usb1/

oder für Dateinamen mit Umlauten

sudo mount -t vfat -o utf8,uid=pi,gid=pi,noatime /dev/sda1 /media/ubs1

Unmout des USB-Stick

sudo umount /media/usb1


Prüfen mit
df -h
oder
sudo blkid -o list -w /dev/null


Automatischer Start von Scripts hier einbauen
sudo nano /etc/rc.local

i2c enablen mit
sudo nano /etc/modules

Diese zeile anfügen
i2c-dev



Alle Packages anzeigen

dpkg --get-selections
oder
dpkg -l



VNC-Server für Desktop Zugriff

sudo apt-get install tightvncserver


Stoppen kann man den TightVNC VNC-Server mit:
vncserver -kill :1
Starten
thightvncserver

Eintrag in /etc/rc.local machen für Start des VNC_Servers
# thightvncserver
/usr/bin/vncserver

Notwendige Directories für Lightpainting im Home erstellen
mkdir lp
```

```
mkdir bin

Ins Directory lp alle scripts für LP kopieren
Ins Directory bin das shutdown Script kopieren

Zudem ins Directory lp noch den Adafruit Code für CharPlate kopieren, dies sind die Dateien
Adafruit_CharLCDPlate.py
Adafruit_I2C.py
```

Quelle dazu

https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code

Danach testen und wenn ok ein Backup der Karte erstellen.

Auf dem Mac-Terminal dies eingeben (Notwendiges anpassen!):

Zuerst Device feststellen

```
diskutil list
sudo dd if=/dev/rdisk1 bs=1m | gzip > /Users/admin/public/lp-3.gz
```

Voila, ist keine Hexerei.

11.6 Bonjour für Raspi installieren

Will man nicht bei jedem Login in den Raspi die IP-Adresse angeben, so wird Bonjour zusätzlich installiert.

Siehe hier

```
Bonjour installieren, damit Raspi mit logischen Namen errreichbar ist (keine genaue IP-Adresse notwendig für Login)

sudo apt-get update
sudo apt-get upgrade

sudo apt-get install avahi-daemon

Für automatischen Start des daemons
sudo update-rc.d avahi-daemon defaults

Hostnamen des Raspi definieren

sudo raspi-config

Dort gehen wir auf Advanced Options, dann auf Hostname. geben dort den gewünschten Hostnamen ein und beenden wieder raspi-config.
Für Lightpainting ist der Name: lpaint1

dann Bonjour installiern, amit Raspi mit logischen Namen errreichbar ist (keine genaue IP-Adresse notwendig für Login)

sudo apt-get update
sudo apt-get upgrade

sudo apt-get install avahi-daemon

Für automatischen Start des daemons
```

```
sudo update-rc.d avahi-daemon defaults

Hostnamen des Raspi definieren

sudo raspi-config

Dort gehen wir auf Advanced Options, dann auf Hostname. geben dort den gewünschten Hostnamen
ein und beenden wieder raspi-config.
Für Lightpainting ist der Name: lpaint1

dann login zum Raspi mit (.local nie vergessen !!)

ssh pi@lpaint.local
```

Beim Mac geht dies dann von selbst, bei Windows muss man noch was machen (ein Gebastel, wie halt immer bei MS):
Dies funktioniert unter Windows nur, wenn man den Bonjour Dienst von Apple installiert hat. Bei Apple Produkten ist diese Funktion von Haus aus verbaut.

11.7 Datenblatt WS2812

Datasheet WS2812

11.8 Links

Meine Raspberry Projekte
Projekt Xmas TV, anderes Raspeberry Projekt:
Projekt Xmas TV

Switcher Projekt, Beschreibung und Code auf meiner Dropbox:
Projekt Switcher

Light Painting Projekt, Bau eines Pixelsticks, Beschreibung und Code siehe Dropbox
Projekt Light Paint

Gute Einführung in NeoPixel
Adafruit Ueberguide

NeoPixel ansteuern mit dem Pi
Adafruit NeoPixels on Raspberry Pi

Adafruit Online Shop Led Strips

Timing bei der Ansteuerung von WS2812
Ansteuerung WS2812

ColorWheel in Web

Colorwheel

Adobe ColorWheel

Spannungsanpassung Pi Pin18 nach Neopixel 5 Volt

TIP #7 3.3V → 5V Using a Diode Offset

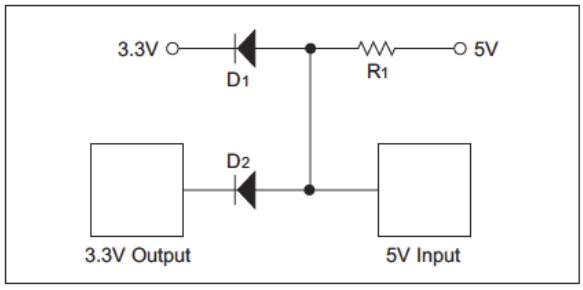
The inputs voltage thresholds for 5V CMOS and the output drive voltage for 3.3V LVTTL and LVCMOS are listed in Table 7-1.

Table 7-1: Input/Output Thresholds

	5V CMOS Input	3.3V LVTTL Output	3.3V LVCMOS Output
High Threshold	> 3.5V	> 2.4V	> 3.0V
Low Threshold	< 1.5V	< 0.4V	< 0.5V

Note that both the high and low threshold input voltages for the 5V CMOS inputs are about a volt higher than the 3.3V outputs. So, even if the output from the 3.3V system could be offset, there would be little or no margin for noise or component tolerance. What is needed is a circuit that offsets the outputs and increases the difference between the high and low output voltages.

Figure 7-1: Diode Offset



When output voltage specifications are determined, it is done assuming that the output is driving a load between the output and ground for the high output, and a load between 3.3V and the output for the low output. If the load for the high threshold is actually between the output and 3.3V, then the output voltage is actually much higher as the load resistor is the mechanism that is pulling the output up, instead of the output transistor.

If we create a diode offset circuit (see Figure 7-1), the output low voltage is increased by the forward voltage of the diode D1, typically 0.7V, creating a low voltage at the 5V CMOS input of 1.1V to 1.2V. This is well within the low threshold input voltage for the 5V CMOS input. The output high voltage is set by the pull-up resistor and diode D2, tied to the 3.3V supply. This puts the output high voltage at approximately 0.7V above the 3.3V supply, or 4.0 to 4.1V, which is well above the 3.5V threshold for the 5V CMOS input.

Note: For the circuit to work properly, the pull-up resistor must be significantly smaller than the input resistance of the 5V CMOS input, to prevent a reduction in the output voltage due to a resistor divider effect at the input. The pull-up resistor must also be large enough to keep the output current loading on the 3.3V output within the specification of the device.

11.9 Liste der Bauteile

Die Tabelle listet alle Bauteile und gibt Bezugsquelle an

Bauteil	Anzahl	Quelle
Raspberry Pi 2 Modell B	1	Reichelt, Play-Zone
16x2 Char Display mit I2C Back-pack	1	Note: das von mir verbaute Produkt kann direkt auf den Pi gesteckt werden. Dann wird I2C Verbindung ohne Kabel möglich. https://www.adafruit.com/products/1115 Aber auch anders möglich: einfacheres Display, wie folgt: Das Display wird mit 4 Drähten verbunden (ground, 5Volt, SCL, Data). I2C Backpack ist an den Display zu löten - einfach. Adafruit Produkte https://www.adafruit.com/products/181 https://www.adafruit.com/products/292
Kippschalter	2	Einfache Kippschalter 1xein, überall erhältlich
Led mit Einbau Fassung	1	überall erhältlich
Drucktasten	4	überall erhältlich
Protoytpe Board für Interface	1	Verschiedene Produkte möglich, Board sollte direkt auf dn Pi gesteckt werden können. https://www.adafruit.com/products/801 https://www.adafruit.com/products/2310
Widerstand 390 Ohm	7	überall erhältlich
Dioden 1N4004 oä	2	überall erhältlich
Kleines TrimmPotentiometer	10K	überall erhältlich
NeoPixel Strip 144 Pixel/Meter		Adafruit Produkt, verschiedene Produkte möglich: https://www.adafruit.com/products/1506 https://www.adafruit.com/products/2847 https://www.adafruit.com/products/2969
Stromversorgung 5 Volt ca. 6-8 Ampere	1	Selbstbau
Stecker	diverse	überall erhältlich
Gehäuse, Holzkonstruktion		
Kabel, Löteinrichtung		

Peter K. Boxler, Februar 2015
Update Oktober 2016