

schlizbäda

B a u a n l e i t u n g

... und ein Erfahrungsbericht zur Phoniebox

Phoniebox – die open-source-Alternative zur Toniebox



GNU Free Documentation License v1.3
© 2020 by schlizbäda

Datum:
29.08.2020



Dieses Dokument wurde mit dem Textsatzsystem L^AT_EX erstellt

Inhaltsverzeichnis

Abbildungsverzeichnis	4
Tabellenverzeichnis	5
1 Einleitung	6
1.1 Rechtliche Hinweise	6
1.2 Hinweis zur neumodernen „Genderei“ landauf, landab	7
1.3 Danksagung	8
1.4 Kurzbeschreibung	8
2 Technischer Aufbau von schlizbädas Phoniebox	10
2.1 Verwendete Komponenten / Stückliste	10
2.1.1 Gehäuse	11
2.1.2 Raspberry Pi 3B	11
2.1.3 HifiBerry MiniAmp	11
2.1.4 Bedientaster	12
2.1.5 Neuftech USB-RFID-Reader	12
2.1.6 Stromversorgung	12
2.1.7 OLED-Display	13
2.2 Selbstgebaute Teile	13
2.2.1 Lochrasterplatine	14
2.2.2 Relais für OnOff SHIM	17
2.2.3 Beschreibung des „Turms“	19
2.2.4 Verwendete Anschlussbuchsen	20
2.3 Holzarbeiten	21
2.3.1 Wahl des Gehäuses	21
2.3.2 Einbau des RFID-Readers	22
2.3.3 Sichtöffnung für Ladeanzeige der Powerbank	22
2.3.4 Sichtscheibe für OLED-Display	23
2.3.5 Halterung für die Powerbank	23
3 Installation der Software	25
3.1 <i>Raspbian Buster Lite</i> auf dem Raspberry Pi installieren	26
3.1.1 Erstellen einer SD-Karte mit <i>Raspbian Buster Lite</i>	26
3.1.2 Anmeldung am Raspberry Pi über <i>ssh</i>	27

3.1.3	<i>Raspbian</i> konfigurieren	28
3.1.4	WLAN einrichten	28
3.2	OnOff SHIM	30
3.2.1	Installation der Shellskripte für den OnOff SHIM	31
3.2.2	Einträge in den aufrufenden Dateien vornehmen	31
3.2.3	Funktionsweise	32
3.3	HifiBerry MiniAmp einrichten	33
3.4	<i>Music Player Daemon (mpd)</i> einrichten	35
3.4.1	Standardpaket des <i>mpd</i> aus <i>Raspbian</i> installieren	36
3.4.2	Herunterladen und Kompilieren des <i>mpd</i>	36
3.4.3	Installieren des <i>mpd</i>	38
3.4.4	Resümee der Kompilierung von <i>mpd</i> – es wäre einfacher gewesen... .	39
3.4.5	Konfiguration des <i>mpd</i>	40
3.5	Manuelle Installation der Phoniebox-Software	42
3.5.1	benötigte Programmpakete und -bibliotheken installieren	42
3.5.2	Projekt RPi-Jukebox-RFID von <i>github</i> klonen und installieren	43
3.5.3	Einbinden des RFID-Lesers als Python(?)-Event	45
3.5.4	Webserver <i>lighttpd</i> für die Web-App der Phoniebox einrichten	45
3.6	RFID-Leser installieren	49
3.6.1	RFID-Leser überprüfen	49
3.6.2	RFID-Leser in der Phoniebox-Software registrieren	49
3.6.3	RFID-Konfigurationsdatei der Phoniebox-Software anlegen	50
3.7	Konfigurationsdateien im Verzeichnis <i>RPi-Jukebox-RFID/settings</i>	50
3.7.1	Audioeinstellungen	50
3.7.2	automatische Abschaltung bei Nichtbenutzung	51
3.7.3	Was soll beim wiederholten Auflegen einer RFID-Karte passieren? .	51
3.8	Die Phoniebox mit Tastern an der GPIO-Leiste steuern	52
3.9	Zuordnung von Hörspielen zu RFID-Karten	53
3.9.1	RFID-Karte manuell zuordnen	54
3.9.2	RFID-Karte über die Webanwendung registrieren	55
3.9.3	Testen der Taster an den GPIO-Pins	57
3.10	<i>systemd</i> : Autostart der Phoniebox-Software einrichten	58
3.11	OLED-Display einrichten	59
3.12	Bootzeit reduzieren	61
3.13	Software-FAQ – oder Hinweise zu „beliebten“ Fehlern	62

Abbildungsverzeichnis

2.1	HifiBerry MiniAmp mit aufgelötetem Pimoroni OnOff SHIM	13
2.2	GPIO-Aufbau mit Lochrasterplatine	14
2.3	GPIO-Belegung – © Raspberry Pi Foundation (CC BY-SA 4.0)	14
2.4	Schaltung auf der Lochrasterplatine	16
2.5	Ansichten der Lochrasterplatine	17
2.6	Konzept zur gesteuerten Trennung des Tasters vom OnOff SHIM	18
2.7	Der Raspberry Pi-Turm der Phoniebox in seiner vollen Pracht	19
2.8	Der Raspberry Pi-Turm schematisch dargestellt	20
2.9	RFID-Reader vernünftig montiert	22
2.10	Ladeanzeige der Powerbank an der Unterseite der Phoniebox	22
2.11	OLED-Display für Statusanzeige in der Frontplatte der Phoniebox	23
2.12	Montage der Powerbank	24
3.1	erster Login über <i>ssh</i>	27
3.2	Lautstärkeeinstellung mit dem <i>alsamixer</i>	35
3.3	Homepage des <i>mpd</i> -Projektes mit Versionsangabe (0.21.25)	37
3.4	Abfrage bei der Installation von <i>Samba</i> mit <i>apt-get</i>	43
3.5	Aufruf des <i>lighttpd</i> -Webservers	48
3.6	Einstellung des Verhaltens bei wiederholtem Auflegen derselben RFID-Karte über die WebApp	52
3.7	Audiodaten (Musikalben) auf die Phoniebox kopieren	53
3.8	RFID-Karte über Webservice registrieren	56
3.9	splitti79s one line installer	60

Tabellenverzeichnis

2.1	Verwendete Komponenten bei schlizbädas Phoniebox	10
2.2	Belegung der GPIO-Leiste bei schlizbädas Phoniebox	15
2.3	Verwendete Anschlussbuchsen bei schlizbädas Phoniebox	21
3.1	Konfigurationsdateien von schlizbädas Phoniebox	25
3.2	Skripte für den OnOff SHIM	31
3.3	Verhalten bei wiederholtem Auflegen derselben RFID-Karte	51
3.4	Taster an der GPIO-Leiste bei schlizbädas Phoniebox	57

1 Einleitung

Vielen Dank für Ihr Interesse an schlizbädas Phoniebox.

Hierbei handelt es sich um ein kindgerechtes Abspielgerät auf Basis des Raspberry Pi für Audiodateien aller Art, insbesondere für Kinderhörspiele. Die Bedienung wurde bewusst einfach und für Kinder verständlich gehalten. Sie erfolgt über Taster für Ein/Aus, Play/Pause, vor/zurück und zur Lautstärkeregelung. Die Auswahl des gewünschten Hörspiels erfolgt über entsprechend beschriftete bzw. bebilderte RFID-Karten.

Die Software der Phoniebox basiert auf dem MIT-licenzierten Werk **RPi-Jukebox-RFID** von **MiczFlor**, die auf *github* unter folgender URL bereitgestellt wird:

<https://github.com/MiczFlor/RPi-Jukebox-RFID>

schlizbäda, der Autor dieser Dokumentation ist **nicht** der geistige Urheber der Phoniebox! Er hat lediglich ein Exemplar für sich gebaut und gibt mit diesem Dokument seine dabei gemachten Erfahrungen und die aufgetretenen Probleme weiter.

1.1 Rechtliche Hinweise

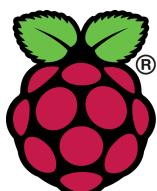
Diese Dokumentation wurde von schlizbäda unter der *GNU Free Documentation License v1.3* veröffentlicht. Sie enthält keine unveränderlichen Abschnitte.

Marken

Einige Bezeichnungen in dieser Schrift können Marken sein, deren Benutzung durch Dritte für deren Zwecke die Rechte der Inhaber verletzen können.

Das in dieser Dokumentation verwendete Raspberry Pi-Logo ist ein registriertes Warenzeichen der Raspberry Pi Foundation und darf nur zusammen mit der Kennzeichnung ® verwendet werden. Details finden sich im Internetauftritt der Raspberry Pi Foundation unter

<https://www.raspberrypi.org/trademark-rules/>



1 Einleitung

Links

In dieser Dokumentation sind Links zu externen Seiten im Internet enthalten. Diese Inhalte macht sich der Verfasser schlizbäda trotz Verlinkung nicht zu eigen, da sie nicht in seinem Einflussbereich stehen! Zum Zeitpunkt der Verlinkung waren keine rechtswidrigen Inhalte erkennbar. Eine ständige Überprüfung auf etwaige rechtsverstoßende Änderungen ist dem Verfasser nach geltendem Recht nicht zuzumuten.

Sollten aktuelle oder künftige Inhalte jedoch rechtswidrig sein, so kann der Autor darüber per e-mail an <mailto:himself@schlizbaeda.de> informiert werden. Es werden dann entsprechende Maßnahmen zur Beseitigung des/der betroffenen Links ergriffen.

Bildrechte

Alle inhaltlich relevanten Fotos und technischen Abbildungen in diesem Dokument stammen vom Verfasser schlizbäda selbst und werden hiermit von ihm unter der *Creative-Commons-Lizenz CC BY-SA 3.0* veröffentlicht. Sie dürfen daher von jedem bei Namensnennung des Urhebers in unveränderter oder auch in veränderter Form unter den gleichen Bedingungen weitergegeben werden:



Nicht vom Autor stammende Bilder sind mit einem entsprechenden Copyright-Vermerk versehen.

Die von der Raspberry Pi Foundation stammende Dokumentation samt Bildern ist in der Regel gemäß **CC BY-SA 4.0** lizenziert: <https://www.raspberrypi.org/documentation/>

1.2 Hinweis zur neumodernen „Genderei“ landauf, landab

Der Autor schlizbäda kann gar nicht oft genug betonen, dass ihm eine Gleichbehandlung **aller** Geschlechter (mittlerweile mehr als zwei) äußerst wichtig ist. Er verurteilt eine Diskriminierung von Menschen nur aufgrund ihres Geschlechtes oder anderer Nebensächlichkeiten wie ihrer Herkunft etc. aufs Schärfste!

Dennoch – oder gerade deswegen – lehnt er die derzeit grassierende Unart des sogenannten „Genderns“ mit sprachlichen Auswüchsen wie „AnwenderInnen“ oder gar „Benutzer*innen“ (jetzt neu mit Stern!) zugunsten einer klaren und verständlichen Ausdrucksweise ab. In diesem Trend sind selbst Wörter wie „Abiturienten“ verpönt: Soll man hier wirklich gender-gerecht „Abiturierende“ schreiben? 😊

Im Übrigen hat der grammatischen Genus mit dem biologischen Sexus nichts zu tun.

1.3 Danksagung

schlizbäda möchte folgenden Personen und Einrichtungen danken:

MiczFlor, dem Schöpfer der Phoniebox:

Homepage: <http://phoniebox.de/>, Softwaredownload: <https://github.com/MiczFlor>

dem **deutschen Raspberry Pi Forum** (<https://forum-raspberrypi.de>) für die unzähligen Beiträge zum Thema Phoniebox

sowie folgenden Mitgliedern des deutschen Raspberry Pi Forums:

@splitti79 (<https://forum-raspberrypi.de/user/54710-splitti79/>),

der auch einen eigenen Internetblog zum Thema betreibt:

<https://splittscheid.de/selfmade-phoniebox/>

<https://splittscheid.de/phoniebox-2/>

@hailogugo (<https://forum-raspberrypi.de/user/51824-hailogugo/>)

1.4 Kurzbeschreibung

Die Phoniebox ist ein intuitiv über RFID-Karten zu bedienender Audioplayer für Kinder. Als kommerzielle Pendants existieren die Toniebox (<https://tonies.de/>) oder der vom Konzept her bezüglich der Software freier konzipierte Hörbert (<https://www.hoerbert.com/>).

Von der Phoniebox existieren zwei Softwarevarianten: Die „einfache“ Variante, die nur auf Audiodateien lokal auf dem Raspberry Pi oder im freigegebenen Heimnetzwerk zugreifen kann. Sie verwendet den **Music Player Daemon (MPD)** von GNU/Linux:

<https://www.musicpd.org/>

Die andere Variante ermöglicht zudem den Zugriff auf Spotify. Sie basiert allerdings auf **Mopidy** anstelle von **MPD**. In dieser Dokumentation wird jedoch nicht auf die Spotify-Variante eingegangen.

Die Hardware einer Phoniebox besteht mindestens aus folgenden Komponenten:

- Raspberry Pi
- Audioverstärker und Lautsprecher
- RFID-Leser und einem Satz passender RFID-Karten
- *optional*: Taster für grundlegende Bedienung
- *optional*: USB-Powerbank als Stromversorgung
- *optional*: Ein-/Ausschalter
- *optional*: LAN- oder WLAN-Modul, falls nicht Bestandteil des verwendeten Raspberry Pi
- *optional*: Statusdisplay

1 Einleitung

Für eine **minimalistische aber funktionale Phoniebox** sind nur der Raspberry Pi als zentrale Einheit, der RFID-Leser für die Umsetzung des Bedienkonzeptes und ein kleiner Audioverstärker erforderlich. Da die Bedienung aller Playerfunktionen (play, pause, prev, next, volume) auch über speziell konfigurierte RFID-Karten erfolgen kann, sind zusätzliche Bedientaster nicht zwingend notwendig.

Ebenso muss die Box nicht unbedingt batteriebetrieben sein, so dass ein passendes Netzteil zur Spannungsversorgung des RPi ausreicht. Dann können auch die Powerbank und der Ein-/Ausschalter weggelassen werden.

Wenn die Phoniebox nur Audiodateien von der lokalen SD-Karte oder von einem USB-Speicherstick abspielen soll, kann auf ein LAN- oder WLAN-Modul verzichtet werden. Die Raspberry Pis der Bauform B haben ohnehin eine Ethernetbuchse und ab dem Raspberry Pi 3B ist ein WLAN-Modul verbaut.

Grundsätzlich optional ist ein Statusdisplay. Es erleichtert aber die Erkennung des Betriebszustandes enorm.

Im deutschen Raspberry Pi Forum gibt es einige Themen zur Phoniebox, die dort ursprünglich als *jukebox4kids* oder auch als *RPi-Jukebox-RFID* bezeichnet wird. Der Begriff *Phoniebox* wurde erst später geprägt.

<https://forum-raspberrypi.de/forum/board/152-musikboxen-webradios-musikplayer/> – Phoniebox-Unterforum

<https://forum-raspberrypi.de/forum/thread/13144-projekt-jukebox4kids-jukebox-fuer-kinder/> – Monsterthread!

<https://forum-raspberrypi.de/forum/thread/45571-splittis-phoniebox-1-0-aka-ghostbox-aka-die-drei-kids-box/>

2 Technischer Aufbau von schlizbädas Phoniebox

Die Phoniebox ist ein open-source-Projekt. Deshalb unterscheidet sich beinahe jede gebaute Phoniebox von allen anderen. Dies gilt nicht nur hinsichtlich der Gestaltung des Gehäuses, sondern auch für die technische Umsetzung. Impressionen von den Phonieboxen der letzten Jahre gibt es auf den Kalendern von *MiczFlor* für 2019 und 2020:

<https://github.com/MiczFlor/RPi-Jukebox-RFID#phoniebox-the-rpi-jukebox-rfid>

In diesem Kapitel wird der hardwareseitige Aufbau des konkreten Exemplars von schlizbäda beschrieben. Es werden auch Hinweise gegeben, warum eine bestimmte Umsetzung gewählt wurde. Dabei muss es sich nicht unbedingt um die optimale Lösung handeln!

2.1 Verwendete Komponenten / Stückliste

Anz.	Artikel	Bestellnummer	Lieferant	Einzel-preis
1x	Holzkiste 33cm*24cm	VARIERA 21386	IKEA	14,99
1x	Sperrholzplatte Kiefer 33cm*24cm	Holzzuschnitt	OBI	2,40
1x	Raspberry Pi 3B	RASPBERRY PI 3	reichelt	35,80
1x	SD-Karte 32GB: Sandisk Extreme A1 32GB	SDSQXAF032GGN6MA	reichelt	11,50
1x	HifiBerry MiniAmp	RPI HB MINI AMP	reichelt	19,50
2x	Lautsprecher Visaton F 8SC-8	VIS F 8SC-8	reichelt	10,25
2x	Schutzwand Visaton 8 ES	VIS 4634	reichelt	6,60
1x	LED-beleuchteter Taster, rot	RAFI 1.15.106.501/1300	reichelt	6,50
1x	LED-beleuchteter Taster, grün	RAFI 1.15.106.502/1500	reichelt	9,10
2x	LED-beleuchteter Taster, gelb	RAFI 1.15.106.503/1400	reichelt	7,99
2x	LED-beleuchteter Taster, blau	RAFI 1.15.106.504/1600	reichelt	10,80
1x	USB RFID-Reader 125kHz	Neuftech	amazon	10,89
100x	RFID-Karte 125kHz	–	amazon	21,28
1x	26000mAh Powerbank 4,8A	EasyAcc	amazon	48,99
1x	Pimoroni OnOff SHIM	Pimoroni	amazon	9,50
1x	1,3inch OLED I2C-Display 128x64 Pixel	AZDelivery	amazon	8,49

Tabelle 2.1: Verwendete Komponenten bei schlizbädas Phoniebox

Bei der Auswahl der Komponenten spielte ein möglichst niedriger Preis eine eher untergeordnete Rolle. Vielmehr wurden bereits vorhandene Teile verwendet. Bei Neubeschaffungen wurde auf Artikel gesetzt, mit denen bereits gute Erfahrungen gemacht wurden.

2.1.1 Gehäuse

Die IKEA-Holzkiste war eigentlich für ein anderes Projekt gedacht. Sie ist zwar formschön, aber die schrägen Wände ohne rechte Winkel machen den Einbau nicht einfacher! Gott sei Dank ist das Ding nun verbaut!

<https://www.ikea.com/de/de/p/variera-kasten-mit-griff-bambus-70226053/>

2.1.2 Raspberry Pi 3B

Ein Raspberry Pi 3B ist für diese Anwendung performant genug und er bietet eine ausreichende *Schwuppdigität*. Außerdem sind darauf bereits onboard ein WLAN- und ein Bluetooth-Modul verbaut. Ein Raspberry Pi 4 wäre zwar noch etwas schneller, verbraucht aber mehr Strom und entwickelt deshalb mehr Wärme. Neben anderer daraus resultierender Nachteile und der erforderlichen Gegenmaßnahmen reduziert sich dadurch auch die Akkulaufzeit.

<https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>

2.1.3 HifiBerry MiniAmp

<https://www.hifiberry.com/shop/boards/miniamp/>

Ein kleiner Class-D-Verstärker mit 2x3W Maximalleistung an 4Ω-Lautsprechern, der speziell für den Raspberry Pi entwickelt wurde. In Kombination mit den Visaton-Lautsprechern wird damit ein durchaus passabler Klang erzeugt, der für kleinere Kinder mehr als ausreichend, aber noch nicht zu laut ist.

In manchen Anleitungen wird die Verwendung von kleinen PC-Aktivboxen empfohlen und bisweilen auch noch, als Audioquelle dafür die originale vierpolige Klinkenbuchse des Raspberry Pi zu verwenden. Dies hat in meinen Augen zwei gravierende Nachteile:

- Die Audioqualität an der originalen Klinkenbuchse des Raspberry Pi ist eher schlecht, da die Audiosignale über eine windig umgesetzte Pulsweitenmodulation erzeugt werden.
- Die in diversen Foren und Tutorials empfohlenen Trust-Leto Aktivboxen sind billigste Chinaware.

Im Raspberry Pi Forum habe ich dies auch schon mit drastischen Worten entsprechend kundgetan: <https://forum-raspberrypi.de/forum/thread/13144-projekt-jukebox4kids-jukebox-fuer-kinder/?postID=384062#post384062>

2.1.4 Bedientaster

Welche Taster hierfür verwendet werden, ist reine Geschmacksache. Da ich derjenige bin, der bei vermeintlichen Schnäppchen immer die fehlerhaften bzw. ganz kaputten Teile erwischt, konnte ich mir die Arcade-Buttons vom freundlich grinsenden Chinesen verkneifen und griff zu den Tastern eines namhaften Herstellers (Rafi). **Ein echter Vorteil dieser Taster ist, dass die verbauten LEDs so hell sind, dass bereits 1-2mA für ausreichende Leuchtintensität sorgen und somit nur mit einem Vorwiderstand direkt an den GPIOs des Raspberry Pi angeschlossen werden können.**

<https://components.rafi.de/DE/DE/produktstruktur/taster/sonstige/leuchtdrucktaster/leuchtdrucktaster-9-1-mm/leuchtdrucktaster-9-1-mm-mit-eingebauter-led>

2.1.5 Neuftech USB-RFID-Reader

Mangels jeglichen Vorwissens auf diesem Gebiet beschloss ich, mich an die Empfehlungen der diversen Tutorials zu halten. Es ist ein chinesisches Qualitätsprodukt, das tut, was es soll: Es liest die Karten problemlos durch die ca. 8mm starke Holzwand der IKEA-Kiste. Sowohl die beigefügte Dokumentation als auch die im Internet geht jedoch gegen Null!

<https://www.amazon.de/Neuftech-Reader-Kartenleseger%C3%A4t-Kartenleser-Kontaktlos/dp/B0180Y0R3E>

2.1.6 Stromversorgung

Die *EasyAcc*-Powerbank zeichnet sich dadurch aus, dass sie zum Aufladen jederzeit an die Stromversorgung angeschlossen werden kann, ohne den Raspberry Pi dabei auch nur kurzzeitig stromlos zu schalten. Dadurch wird der Ärger durch unkontrolliert ausgeschaltete Raspberry Pis von vorne herein vermieden. Es mag günstigere Powerbänke geben, die dies auch können, aber der Forenkonsens geht eindeutig zu den *EasyAcc*-Produkten.

<https://www.easyacc.com/de/670-easyacc-26000mah-power-bank.html>

Von *EasyAcc* gibt es eine deutlich günstigere Variante mit „nur“ 20000mAh Ladekapazität, die in den meisten Phonieboxen verwendet wird. Aber der schlizbäda gönnte sich getreu dem Motto „Klotzen, nicht kleckern!“ die größere Variante mit 26000mAh und auch die funktioniert problemlos.

Zum Ein- und Ausschalten der Phoniebox wird der *OnOff SHIM* von Pimoroni verwendet. Allerdings musste der *gepimpt* werden, siehe Abschnitt 2.2.2

<https://shop.pimoroni.com/search?type=product&q=onoff+shim>

2.1.7 OLED-Display

Hier wurde 1:1 die Erweiterung von **@splitti79** aus dem Raspberry Pi Forum nachgebaut:

<https://forum-raspberrypi.de/forum/thread/41465-oled-display-fuer-die-phoniebox/>

https://github.com/splitti/oled_phoniebox

Artikel bei amazon: <https://www.amazon.de/gp/product/B078J78R45>

2.2 Selbstgebaute Teile

Die Phoniebox, das Wochenend-Projekt – <http://phoniebox.de/>

Ganz so schnell ging's bei mir leider nicht. Auf dem Weg zu meiner Box gab es diverse Fallstricke, die ich im Raspberry Pi Forum im Thread

<https://forum-raspberrypi.de/forum/thread/45820-phoniebox-2-0-rc7-mpd-spielt-ueber-rfid-karte-gewahlte-musik-nicht-ab/>

schilderte. Dort bekam ich viele Hilfestellungen zur Lösung der einzelnen Punkte. Mit dem heutigen Wissen würde ich einem Phoniebox-Neuling folgende Tipps mitgeben:

<https://forum-raspberrypi.de/forum/thread/46231-hilfe-erste-phoniebox-totaler-anfaenger/?postID=415862#post415862>

Für eine gut funktionierende Phoniebox ist eine feste, stabile und wirklich haltende Verkabelung unabdingbar! Alle Verbindungen, die beim fliegenden Testaufbau auf einem Steckbrett mit Jumperkabeln (sogenannten DuPont-Kabeln) und Ähnlichem aufgebaut wurden, müssen beim Einbau in die Box sauber verlötet oder gecrimpt werden. Es lohnt sich auch, gute Steckverbinder einzubauen, um die Box zu Wartungszwecken später wieder leichter zerlegen zu können. Selbst die von mir verwendeten Stift- und Buchsenleisten im 2,54mm-Raster sind eigentlich noch nicht zuverlässig genug und insbesondere bei Einzelbuchsen zu wackelig.

Die gesamte Hardware an der GPIO-Leiste des Raspberry Pi wurde gestapelt: Direkt auf den RPi wird der HifiBerry MiniAmp gesteckt. Da der MiniAmp oben keine Stifitleiste hat, auf der weitere RPi-HATs gesteckt werden könnten, habe ich selbst auf der Oberseite eine 40-polige Stifitleiste aufgelötet. Auf dieser Stifitleiste habe ich gleich den Pimoroni OnOff SHIM verlötet, siehe Abbildung 2.1.

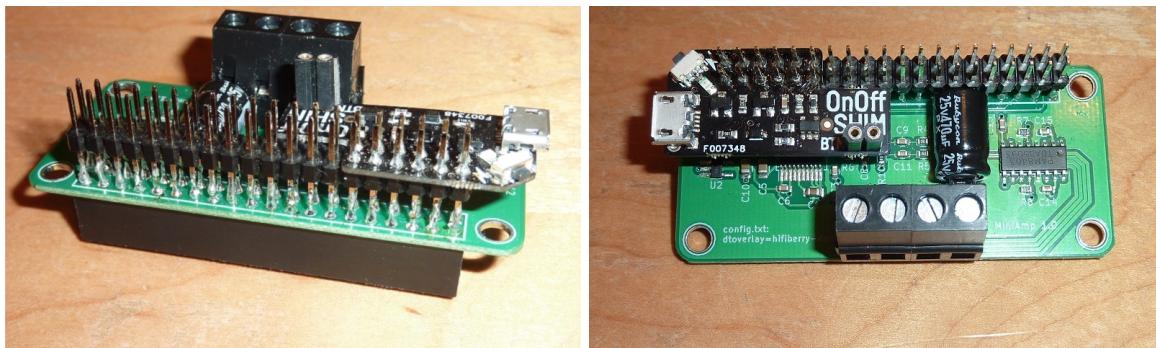


Abbildung 2.1: HifiBerry MiniAmp mit aufgelötetem Pimoroni OnOff SHIM

2 Technischer Aufbau von schlizbädas Phoniebox

Ganz oben auf diesen *Turm* wird schließlich eine Lochrasterplatine gesteckt, die die Anschlüsse für die Bedientaster, die LEDs und das OLED-Display enthält, siehe Abbildung 2.2.

Ich habe die Lochrasterplatine deshalb ganz oben aufgesteckt, dass die Bauteile auf der Platine freier platziert werden können und die Stifteleisten beim Anstecken der Taster und LEDs besser zugänglich sind.

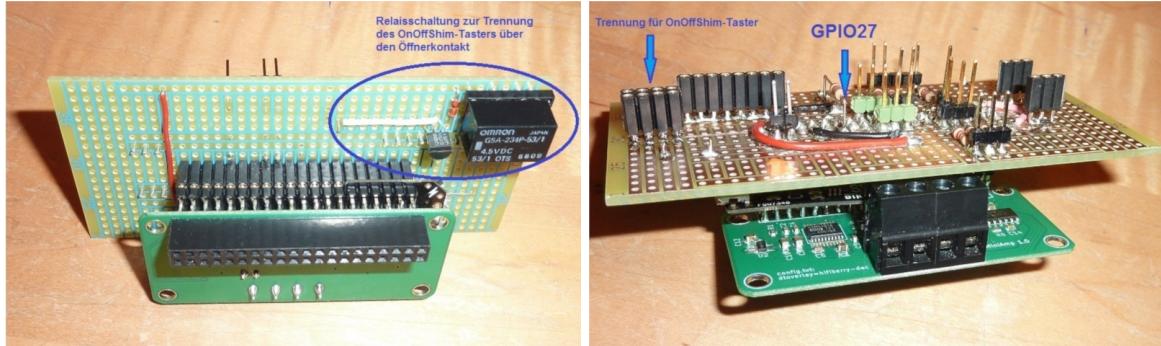


Abbildung 2.2: GPIO-Aufbau mit Lochrasterplatine

2.2.1 Lochrasterplatine

Vor dem Aufbau der Lochrasterplatine wurde die Verwendung der GPIO-Pins überprüft und unter Berücksichtigung von Abbildung 2.3 für schlizbädas Phoniebox gemäß Tabelle 2.2 festgelegt. In Kapitel 3.8 wird die Software an die hier beschriebene Pinbelegung der Lochrasterplatine angepasst.

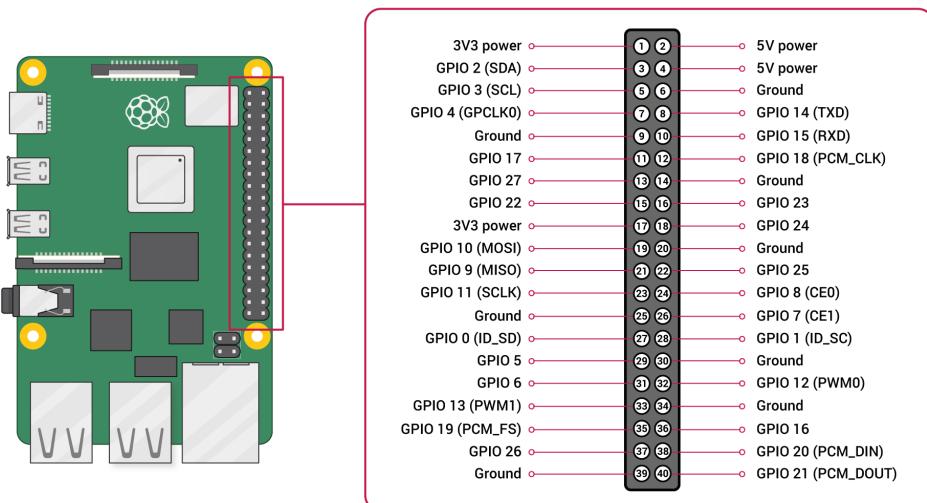


Abbildung 2.3: GPIO-Belegung – © Raspberry Pi Foundation (CC BY-SA 4.0)

Verwendung	Funktion	Pin		Pin	Funktion	Verwendung
	3V3	1		2	5V	
OLED-Display: SDA	GPIO2 (SDA)	3		4	5V	
OLED-Display: SCL	GPIO3 (SCL)	5		6	GND	
OnOff SHIM: W 0=ausschalten	GPIO4	7		8	GPIO14	–
	GND	9		10	GPIO15	–
OnOff SHIM: R Status Ein-/Aus-Taster W Anzeige Power-LED	GPIO17	11		12	GPIO18	MiniAmp: PCM CLK I2S BCLK (bit clock)
OnOff SHIM: W 1=Relais ansteuern	GPIO27	13		14	GND	
–	GPIO22	15		16	GPIO23	–
	3V3	17		18	GPIO24	LED vol. down
LED prev	GPIO10	19		20	GND	
LED next	GPIO9	21		22	GPIO25	LED vol. up
LED play/pause	GPIO11	23		24	GPIO8	Taster prev
	GND	25		26	GPIO7	Taster next
–	ID SD	27		28	ID SC	–
Taster play/pause	GPIO5	29		30	GND	
Taster vol. down	GPIO6	31		32	GPIO12	Taster vol. up
Taster vol. mute	GPIO13	33		34	GND	
MiniAmp: PCM FS I2S SYNC (left/right clk)	GPIO19	35		36	GPIO16	MiniAmp: PAM8403 W 0=MUTE
MiniAmp: PAM8403 W 0=SHDN (shutdown)	GPIO26	37		38	GPIO20	MiniAmp: PCM DIN I2S data in (<i>not used</i>)
	GND	39		40	GPIO21	MiniAmp: PCM DOUT I2S data out

Tabelle 2.2: Belegung der GPIO-Leiste bei schlzbädas Phoniebox

Auf der Lochrasterplatine befinden sich folgende Komponenten:

- GPIO-Pins als Eingänge für die Taster mit PullUp-Widerständen
- GPIO-Pins als Ausgänge zur direkten Ansteuerung von LowCurrent-LEDs $\leq 3\text{mA}$
- I2C-Bus für die Ansteuerung des OLED-Displays
- Relaiserweiterung für den OnOff SHIM

Um die Bedientaster zuverlässig einzulesen, empfehle ich die Verwendung von **externen** PullUp-Widerständen mit $10\text{k}\Omega$. Es ist zwar möglich, die internen PullUp-Widerstände des SoC BCM2837 zu aktivieren, doch bei so mancher Phoniebox war dies dann letztlich der

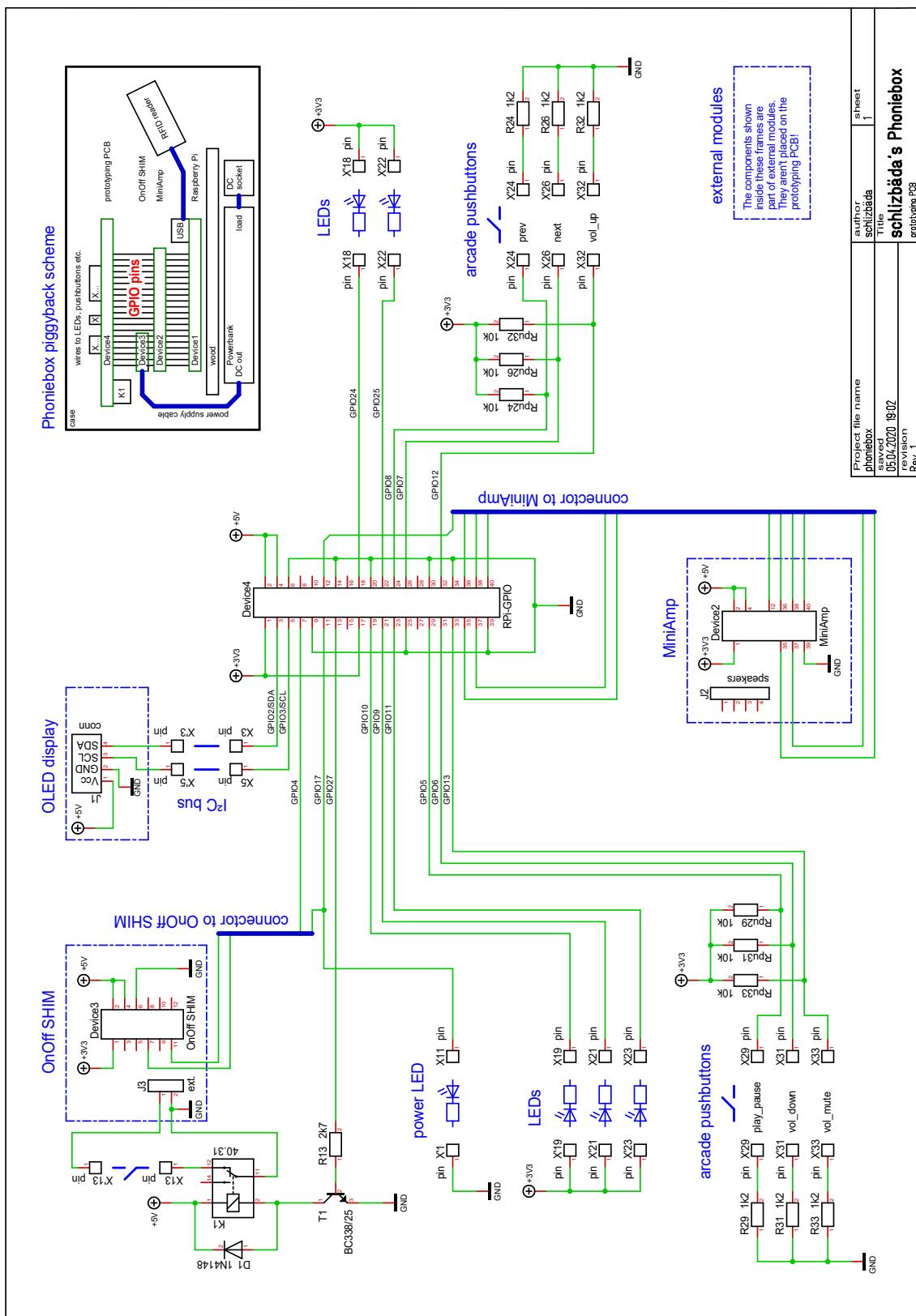


Abbildung 2.4: Schaltung auf der Lochrasterplatine

2 Technischer Aufbau von schlzbädas Phoniebox

Grund, dass die Taster nicht wie erwartet funktioniert hatten.

Um den Pegel bei gedrücktem Taster auf *low* zu ziehen, aber den Stromfluss zu begrenzen, werden mit GND verbundene 1,2k Ω -Widerstände zugeschaltet.

Einfacher ist die Ausführung der GPIO-Pins, an denen die LEDs der Taster angeschlossen werden. Da in den verwendeten Tastern LowCurrent-LEDs eingebaut sind, die bereits unter 3mA Stromfluss ausreichend hell leuchten, sind keine Transistortreiberschaltungen notwendig. Der bei LEDs obligatorische Vorwiderstand wurde für einen 3,3V-Betrieb ausgerechnet und direkt an die LEDs angelötet. Eine visuelle Kontrolle zeigte, dass für die gleiche Helligkeit bei den verschiedenen Farben unterschiedliche Widerstandswerte erforderlich sind:

LED-Farbe Vorwiderstand

rot	1,2k Ω
gelb	1,2k Ω
grün	4,7k Ω
blau	1,5k Ω
weiß	1,5k Ω

Auf der Lochrasterplatine befinden sich daher nur die Steckverbinder für den Anschluss der Taster-LEDs.

Das OLED-Display wird über den I2C-Bus des Raspberry Pi angesteuert, der an den GPIO-Pins 3 und 5 der Stiftleiste aufgelegt ist. Da dieses Modul bereits 10k Ω -PullUp-Widerstände enthält, wurden auf der Lochrasterplatine keine zusätzlichen PullUps verbaut.

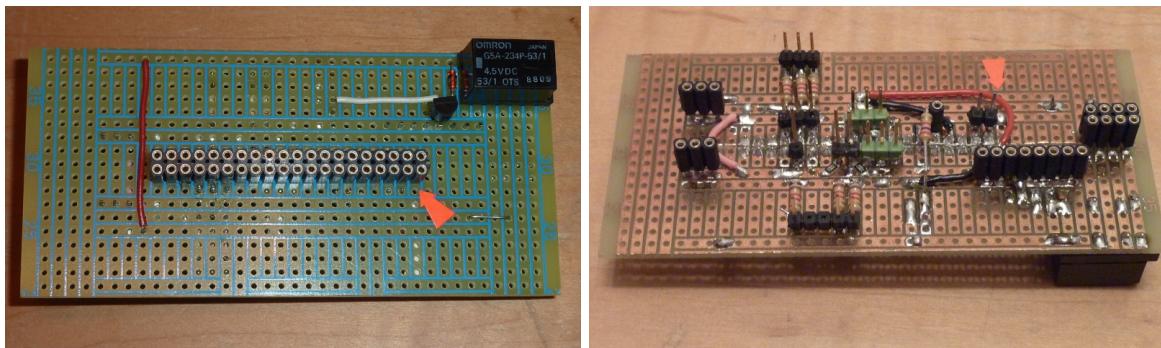


Abbildung 2.5: Ansichten der Lochrasterplatine

2.2.2 Relais für OnOff SHIM

Zum Ein- und Ausschalten der Phoniebox wird der OnOff SHIM von Pimoroni verwendet. Beim Testen stellte sich heraus, dass, wenn der Ein-/Aus-Taster des OnOff SHIMs (oder ein dazu parallel angeschlossener Taster) bis über den Abschaltzeitpunkt hinaus gedrückt gehalten wird, der Raspberry Pi zwar herunterfahren, aber die Versorgungsspannung nicht getrennt wird! Die rote Power-LED des Raspberry Pi leuchtet weiter und es fließt ein deutslicher Strom von ca. 80-90mA, der die Powerbank *leersäuft*.

2 Technischer Aufbau von schlizbädas Phoniebox

Mittlerweile weiß ich, dass es sich hierbei eher um ein generelles Problem beim OnOff SHIM handelt. Es ist nicht bloß wieder eine Macke meines Exemplars! Siehe <https://forum-raspberrypi.de/forum/thread/42364-phoniebox-on-off-shim/?postID=425619#post425619>

Aber da gerade Kinder kein Gefühl dafür haben, wie lange sie die Taste drücken dürfen/sollen, musste eine vernünftige Lösung her!

Letztlich konnte dieses Problem wie in Abbildung 2.6 durch den Einbau eines als Öffner geschalteten Relaiskontakte in Reihe zum Ein-/Aus-Taster des OnOff SHIM gelöst werden. Beim Herunterfahren wird dabei über ein Softwareskript das Relais angesteuert, so dass ein evtl. immer noch geschlossener Taster vom OnOff SHIM getrennt wird. Weil *nach* dem Abschalten auch das Relais wieder abfällt, führt betätigter Taster somit zu einem erneuten Hochfahren der Phoniebox. Der Ein-/Aus-Taster hat jetzt drei Funktionalitäten:

- im ausgeschalteten Zustand kurz betätigen: Phoniebox fährt hoch
- etwas länger betätigen zum Herunterfahren der Phoniebox
- richtig lange betätigen für einen Reboot (Neustart) der Phoniebox

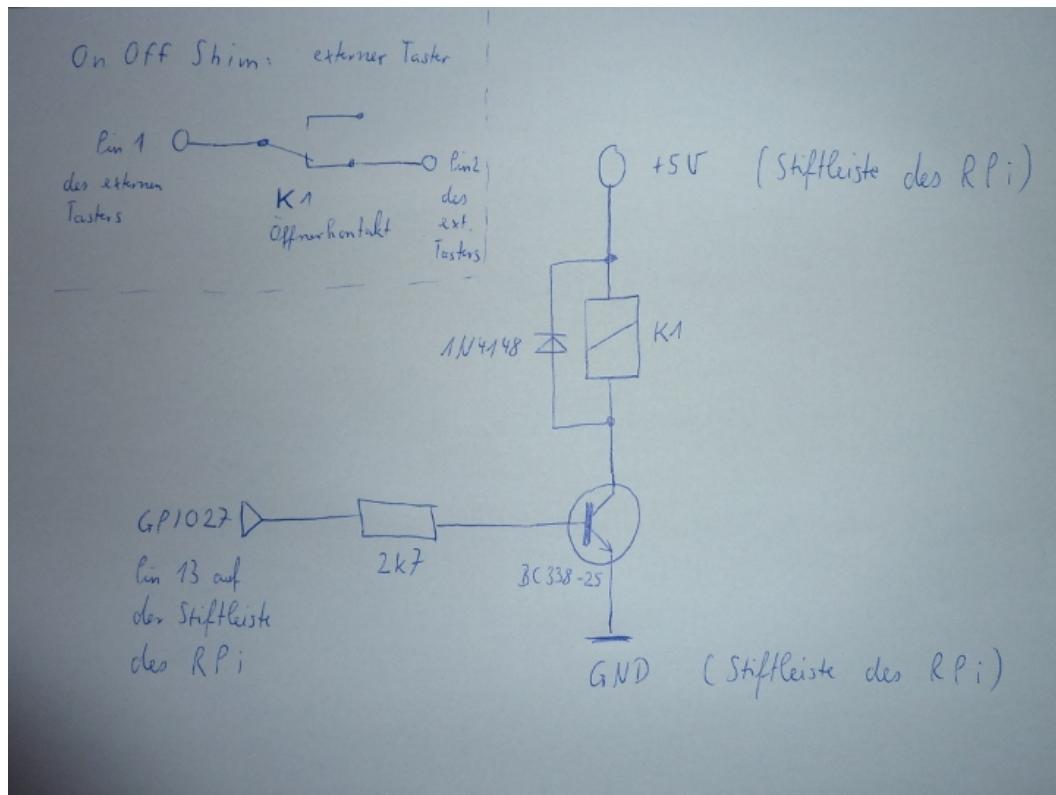


Abbildung 2.6: Konzept zur gesteuerten Trennung des Tasters vom OnOff SHIM

Dieser Schaltungsteil wurde nachträglich auf der Lochrasterplatine ergänzt. Die Installation der Software für den OnOff SHIM wird in Kapitel 3.2 beschrieben.

2.2.3 Beschreibung des „Turms“

Das Erweiterungskonzept beim Raspberry Pi besteht darin, eine zusätzliche Platine *huckepack* als sogenannte piggyback-Platine oder im Slang der Raspberry Pi Foundation als Pi-HAT („hardware attached on top“) auf die 40-polige GPIO-Leiste aufzustecken. Bei mehreren Erweiterungsplatinen entsteht so schnell ein ganzer Turm von Zusatzplatinen 😊

Auch bei der Phoniebox sind mehrere Aufsatzplatten im Einsatz, wie in Abbildung 2.7 zu bewundern ist.

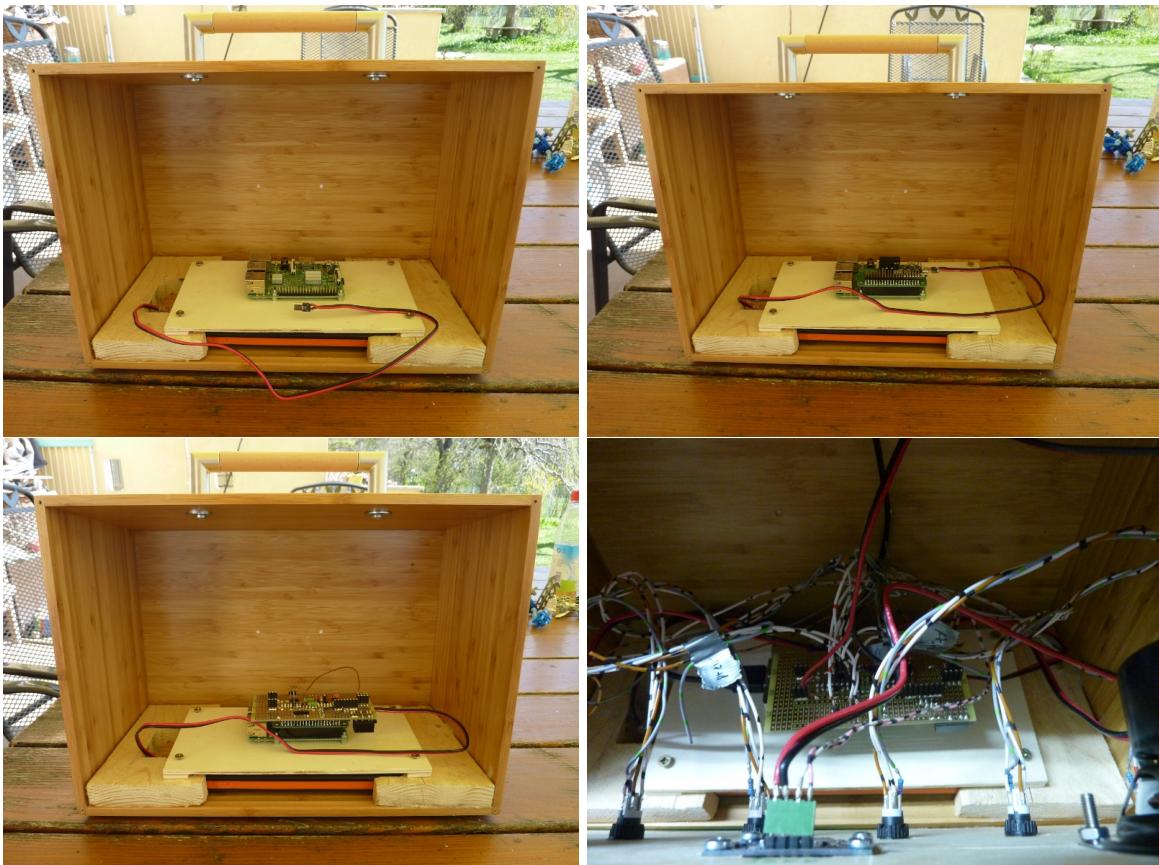


Abbildung 2.7: Der Raspberry Pi-Turm der Phoniebox in seiner vollen Pracht

Ganz unten in der Box befindet sich alleine schon aus Gründen der Schwerpunktverlagerung die Powerbank, die über ein Holzgestell fixiert wird. Darüber ist auf einer dünnen Sperrholzplatte der Raspberry Pi montiert, auf den der Hifiberry MiniAmp aufgesteckt wurde. Auf den MiniAmp wurde eine 40-polige Stiftleiste aufgelötet, auf der auch der OnOff SHIM verlötet wurde. Um die Verkabelung beim Zusammenbau etwas leichter durchführen zu können, wurde die selbstgebaute Lochrasterplatine ganz bewusst oben aufgesetzt. Der Drahtverhau im Inneren der Box vom Turm zu den Tastern, zum OLED-Display und zu den Lautsprechern ist immer noch groß genug!

Schematisch ist es nochmals etwas übersichtlicher in Abbildung 2.8 dargestellt.

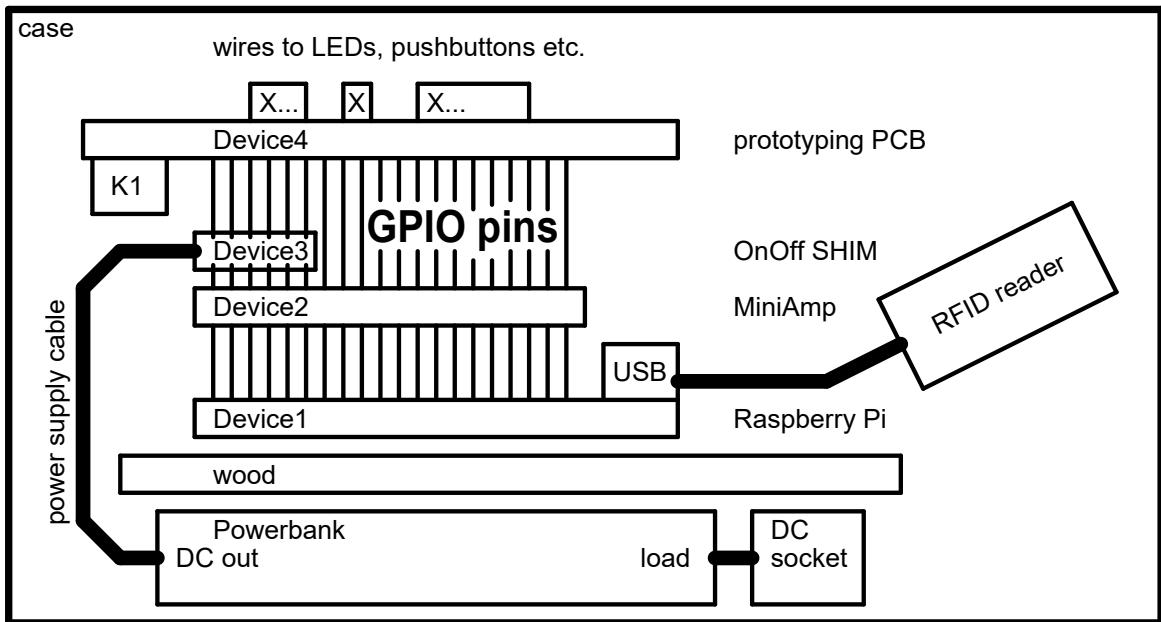


Abbildung 2.8: Der Raspberry Pi-Turm schematisch dargestellt

2.2.4 Verwendete Anschlussbuchsen

Bei der Stromversorgung wurde darauf geachtet, die auftretenden Stromstärken verlustfrei zu übertragen. So wurden nur Kupferlitzen mit einem Querschnitt $\geq 0,25\text{mm}^2$ verwendet, um den Spannungsabfall niedrig genug zu halten. Dieser Querschnitt ist bei mehradrigen Leitungen laut https://www.lappkabel.de/fileadmin/catalog/2010_de_pdf/T12_Belastbarkeit_Grundtabelle_Reduktionstabellen.pdf für bis zu 4A ausgelegt.

Neben ausreichend dicken Kabelquerschnitten müssen auch die verwendeten Steckverbinder dafür ausgelegt werden. In die Gehäuserückwand der Phoniebox wurde als Ladebuchse für die Powerbank das Produkt NEB-1 (lumberg) eingebaut. Diese Buchse ist für Stromstärken bis 3A ausgelegt. Da die Powerbank bei Anschluss an eine Micro-USB-Ladebuchse bis zu 2A zieht und bei Anschluss an beiden USB-Buchsen bis zu 4A, darf hier nur eine USB-Buchse verwendet werden!

Das Gegenstück, der Hohlstecker NES-1 GR (lumberg) wurde an einem 5V-Steckernetzteil verbaut.

Für die USB-Anschlüsse wurden die lötbaren USB-Stecker BKL-10120098 (USB-A) und BKL-10120262 (micro USB-B) verwendet. Siehe Tabelle 2.3.

Anz.	Abb.	Artikel	Bestellnummer	Lieferant	Einzelpreis
1x		Hohlstecker	LUM NES1 GR	reichelt	0,83
1x		Einbaukupplung	LUM NEB 1	reichelt	1,20
2x		BKL 10120262	USB MICRO ST	reichelt	0,85
1x		BKL 10120100	BKL 10120100	reichelt	0,79
1x		BKL 10120098	BKL 10120098	reichelt	1,40

Tabelle 2.3: Verwendete Anschlussbuchsen bei schlizbädas Phoniebox

Hinweis

Der Einbau der Lumberg-Buchse NEB-1 war in der 8mm starken Rückwand des verwendeten Holzgehäuses etwas diffizil: Um die Buchse mit ihrer seitlich angebrachten Kontaktfeder überhaupt durch die Holzplatte zu bringen, ohne das Loch so groß zu bohren, dass es durch den Befestigungsflansch nicht mehr überdeckt wurde, musste am Loch in der Gehäuseinnenseite mehrmals vorsichtig *herumgeknabbert* werden! Diese Buchse ist wohl eher nur für dünne Metallwände ($\leq 2\text{mm}$?) geeignet.

2.3 Holzarbeiten

Zu den Holzarbeiten möchte ich mich als Laie auf diesem Gebiet gar nicht zu weit aus dem Fenster lehnen. Das können andere besser, siehe hier:

<https://github.com/MiczFlor/RPi-Jukebox-RFID/blob/develop/docs/2020-Phoniebox-Calendar.jpg>

Aber auf einige Punkte möchte ich dennoch hinweisen:

2.3.1 Wahl des Gehäuses

Ich habe ein Gehäuse verwendet, das sich durch schräg verbaute Wände und dadurch wenige rechte Winkel auszeichnet. Für echte Holzwürmer kein Problem, aber mir machte es den Einbau nicht leichter. Nächstes Mal werde ich die typischen Baumarkt-Kolzkisten verwenden. Da sind die Außenwände dicker und der Einbau ist entsprechend einfacher 😊
Umso mehr freut es mich, dass ich das Gehäuse nicht verpfuscht habe (yeah!)

2.3.2 Einbau des RFID-Readers

In meiner Faulheit beschloss ich zunächst, den RFID-Reader mit einem doppelseitigen Klebeband (eigentlich ein gutes 3M-Band, aber mittlerweile auch ca. 10 Jahre alt) innen an der Oberseite der Box festzukleben. Eines Tages reagierte die Box nicht mehr auf aufgelegte RFID-Karten und nicht einmal das Pieperäusch des Readers ertönte mehr. Frontplatte runter und was sah ich? Der Reader lag mitten in der Kiste...

Auch zusätzliche Unterstützung durch sogenanntes *Panzertape* half nichts, so dass ich den Reader letztlich mit vier Holzschrauben vernünftig montierte: Dazu wurden die kleinen Gehäuseschrauben unter den Gummifüßen entfernt, die Schraublöcher im Gehäuse durchgebohrt und vergrößert und der Reader mit vier Holzschrauben innen stabil an der Oberseite der Phoniebox verschraubt.



Abbildung 2.9: RFID-Reader vernünftig montiert

2.3.3 Sichtöffnung für Ladeanzeige der Powerbank



Abbildung 2.10: Ladeanzeige der Powerbank an der Unterseite der Phoniebox

Die *EasyAcc*-Powerbank ist mit vier kleinen LEDs ausgestattet, die über die Anzahl der leuchtenden LEDs den prozentualen Ladezustand der Powerbank anzeigen. Wenn das Netzteil angesteckt ist und die Powerbank geladen wird, blinkt die höchstwertige LED.

Der Einfachheit halber habe ich die Powerbank im Gehäuse mit der Ladeanzeige nach unten montiert und im Gehäuse an der entsprechenden Stelle ein kleines Sichtfenster herausgesägt, siehe Abbildung 2.10.

2.3.4 Sichtscheibe für OLED-Display

Auch in schlizbädas Phoniebox wurde ein kleines 1,3-Zoll OLED-Display eingebaut. Dies ist gerade zu Beginn u. a. für eine schnelle Fehlerdiagnose äußerst sinnvoll. Allerdings sollte man vor das Display eine kleine Plexiglasscheibe oder Ähnliches einsetzen, um das Display vor groben Kinderfingern zu schützen.

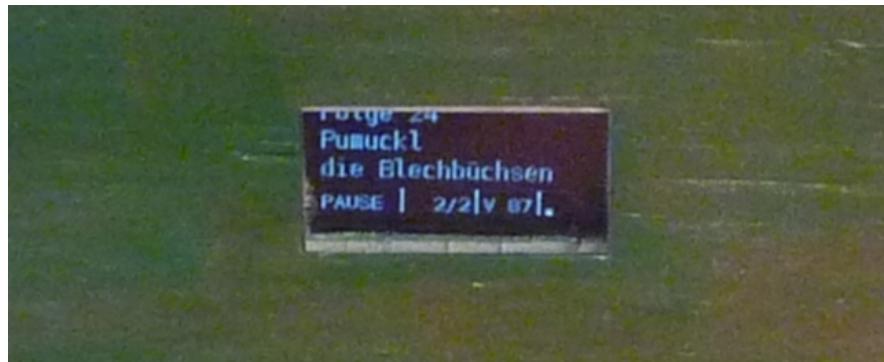


Abbildung 2.11: OLED-Display für Statusanzeige in der Frontplatte der Phoniebox

2.3.5 Halterung für die Powerbank

Die Powerbank wird über verleimte Holzhalterungen im Gehäuse fixiert. Da die *vorhandene* Holzplatte für die Halterungen etwas dünner war als die Powerbank selbst, wurde die Breite mit zwei Sperrholzstreifen „aufgedoppelt“. Damit die Powerbank fest eingepresst wird, wurde ein ESD-Schaumstoff dazwischen gelegt. Mit einer Sperrholzplatte wird die Powerbank gehalten. Später wird darauf der Raspberry Pi montiert.

2 Technischer Aufbau von schlzbädas Phoniebox



Abbildung 2.12: Montage der Powerbank

3 Installation der Software

Eigentlich ist die Installation der Software *babyleicht* und man muss (Achtung: Unwort!) *nur* den **one-line-installer** von *MiczFlor* auf einem jungfräulichen **Raspbian** starten...

<https://github.com/MiczFlor/RPi-Jukebox-RFID#installation>

Aber:

Aufgrund vieler Anfragen der mittlerweile relativ großen Community rund um die Phoniebox und der daraus resultierenden zahlreichen *Push Requests* auf github und auch wegen vieler anderer guter Gründe schleichen sich in die an sich großartige Arbeit von *MiczFlor* naturbedingt immer wieder kleinere Fehler ein, die insbesondere für unbedarfte Neulinge bisweilen schwer zu finden und zu beseitigen sind.

Für die Installation der Software müssen einige Konfigurationsdateien angepasst werden. Im git-Repository unter https://github.com/schlizbaeda/schlizbaedas_Phoniebox gibt es das Unterverzeichnis *files*, in dem diese Dateien in dem Stand enthalten sind, wie sie auf schlizbädas Phoniebox produktiv eingesetzt werden. TODO:

/absoluter Pfad/Dateiname	Verwendung	wo?
<i>/boot/wpa-supplicant.conf</i>	t.b.d.	
<i>/boot/config.txt</i>	allgemeine Konfigurationsdatei des Raspberry Pi	link
<i>/etc/asound.conf</i>	t.b.d.	
<i>/home/pi/RPi-Jukebox-RFID/requirements.txt</i>	t.b.d.	
<i>/etc/lighttpd/lighttpd.conf</i>	Web-App der Phoniebox	
<i>/etc/sudoers</i>	t.b.d.	
<i>/etc/lighttpd/conf-available/15-fastcgi-php.conf</i>		
<i>/etc/mpd.conf</i>	t.b.d.	
<i>/home/pi/RPi-Jukebox-RFID/scripts/gpio-button.sh</i>		
<i>rfid_trigger_play.conf</i>	In dieser Datei ist die Syntax erklärt	
<i>/etc/rc.local</i>	Autostart: Starten von <i>onoffshim_switch.sh</i>	
<i>/home/pi/onoffshim_switch.sh</i>	OnOff SHIM cyberghost Teil 1	
<i>.../onoffshim_gpio-shutoff.sh</i>	OnOff SHIM cyberghost Teil 2	

Tabelle 3.1: Konfigurationsdateien von schlizbädas Phoniebox

Tabelle
vervoll-
ständigen

3.1 Raspbian Buster Lite auf dem Raspberry Pi installieren

Einige Tutorials im Internet empfehlen zwar, die Software für die Phoniebox unter **Raspbian Buster with desktop** zu installieren, aber da es sich bei der Phoniebox um ein einfaches standalone-System ohne Bildschirm handelt, ist es in meinen Augen völlig ausreichend, die Installation auf dem wesentlich kleineren **Raspbian Lite** vorzunehmen. Ein grafischer X11-Desktop ist schlichtweg nicht erforderlich. Außerdem reicht für **Raspbian Buster Lite** zur Not sogar eine SD-Karte mit einer Speicherkapazität von 4GB.

3.1.1 Erstellen einer SD-Karte mit **Raspbian Buster Lite**

Zunächst wird die aktuelle Version von **Raspbian Buster Lite** von der Homepage der Raspberry Pi Foundation (<https://www.raspberrypi.org/downloads/raspbian/>, ca. 450MiB) heruntergeladen und entpackt (1,8GiB). Das entpackte Image wird mit einem dafür vorgesehenen Programm wie **win32diskimager** oder am GNU/Linux-PC mit dem systemeigenen Kommando `dd` auf die SD-Karte *geflasht*, um das Image 1:1 zu übertragen.

Achtung

Es reicht nicht, die Imagedatei einfach auf eine bereits formatierte SD-Karte zu kopieren!

Die Details des *Flashens* von Betriebssystem-Images auf eine SD-Karte werden von der Raspberry Pi Foundation unter <https://www.raspberrypi.org/documentation/installation/installing-images/README.md> ausführlich beschrieben und in diesem Dokument als bekannt vorausgesetzt.

Nach dem Flashen enthält die SD-Karte zwei Partitionen:

- die FAT32-formatierte `/boot`-Partition
- die ext4-formatierte Rootpartition von **Raspbian**

Hinweis

Beim (erneuten) Anstecken der SD-Karte am PC werden diese beiden Partitionen im Dateimanager angezeigt. Sollte auf dem PC jedoch das Betriebssystem **Windows** verwendet werden, so wird nur die FAT32-formatierte `/boot`-Partition erkannt.

3.1.2 Anmeldung am Raspberry Pi über ssh

Hinweis

Für die ersten Schritte muss die Netzwerkverbindung zum Raspberry Pi über ein verkabeltes LAN aufgebaut werden, da die WLAN-Verbindung noch nicht eingerichtet ist! Dies geschieht erst in Abschnitt 3.1.4.

```

pi@raspberrypi: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
peter@tuxedo-lmde:~$ ssh pi@raspberrypi
The authenticity of host 'raspberrypi (192.168.178.99)' can't be established.
ECDSA key fingerprint is SHA256:WrU6VX2U2KMfFKUMb/0cRX0Wq7MqKOU+g9fS7Bddpls.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'raspberrypi,192.168.178.99' (ECDSA) to the list of known hosts.
pi@raspberrypi's password:
Linux raspberrypi 4.19.97+ #1294 Thu Jan 30 13:10:54 GMT 2020 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Wed Apr  8 22:28:05 2020 from 192.168.178.155

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $ 

```

Abbildung 3.1: erster Login über ssh

Da an den Raspberry Pi für die Phoniebox weder Bildschirm noch Tastatur angeschlossen werden, muss der Zugriff auf den Raspberry Pi von Anfang an über **ssh** erfolgen. Dazu muss auf der Partition **/boot** der SD-Karte die leere Datei **ssh** angelegt werden.

Nun wird die SD-Karte in den Raspberry Pi gesteckt, der Raspberry Pi über **Ethernet ans LAN** gehängt und eingeschaltet. Nach spätestens einer Minute sollte **Raspbian Buster Lite** vollständig gebootet und der Raspberry Pi im LAN bekannt sein. Auf dem PC wird in einem Terminalfenster (unter **Windows** in der Eingabeaufforderung **cmd.exe**) die Software **ssh** mit folgendem Kommando gestartet:

```
PC $ ssh pi@raspberrypi # Das Passwort lautet raspberry
```

Die Bildschirmausgabe sieht in etwa wie in Abbildung 3.1 aus.

3.1.3 Raspbian konfigurieren

Nach erfolgreichem *ssh*-Login muss das System jetzt auf einen aktuellen und sicheren Stand gebracht werden:

```
pi $ sudo apt update && sudo apt upgrade # System auf den neuesten Stand bringen
pi $ sudo raspi-config

1 Change User Password                                # Passwort unbedingt ändern!
2 Network Options          -> N1 Hostname           # z.B. phoniebox1
3 Boot Options            -> B1 Desktop / CLI      -> B2 Console Autologin
4 Localisation Options   -> I2 Change Timezone     # Zeitzone anpassen
                           -> I4 Change Wi-fi Country  # diese Anpassung ist ganz wichtig!
5 Interfacing Options    -> P2 SSH
```

Beim Verlassen von *raspi-config* sollte die Abfrage *Would you like to reboot now?* mit [yes] beantwortet werden. Der Raspberry Pi wird neu hochgefahren. Vom PC aus kann nach ca. einer Minute Wartezeit ein neuer *ssh*-Login auf den neuen Hostnamen mit dem (hoffentlich) geänderten Passwort erfolgen:

```
PC $ ssh pi@phoniebox1
```

3.1.4 WLAN einrichten

Da die Phoniebox als tragbares Gerät für's Kinderzimmer konzipiert ist, sollte ein WLAN-Zugang eingerichtet werden, um als Eltern später z. B. per Laptop oder Smartphone jederzeit einen *bequemen* Zugriff auf die Box zu haben. Wie häufig unter *GNU/Linux* gibt es auch für die WLAN-Konfiguration mehrere zielführende Wege, wie sie z. B. im Elektronik-Kompendium unter <https://www.elektronik-kompendium.de/sites/raspberry-pi/1912221.htm> beschrieben werden. Für schlizbädas Phoniebox wird die dort beschriebene „Variante 2“ mittels *wpa_supplicant* verwendet.

Auf einem frisch installierten *Raspbian Lite* wird der WLAN-Zugriff in der *ssh*-Konsole über die Datei */etc/wpa_supplicant/wpa_supplicant.conf* eingerichtet. Die drei ersten Zeilen sollten in der Datei bereits vorhanden sein. Für jedes gewünschte WLAN muss ein Eintrag *network{...}* nach folgendem Schema eingefügt werden. Dazu wird der auch über *ssh* funktionierende Texteditor *nano* verwendet:

```
pi $ sudo nano /etc/wpa_supplicant/wpa_supplicant.conf
:
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=DE

network={ # Erweiterungen von schlizbäda
```

3 Installation der Software

```

ssid=<wlan-name> # Klartext-Bezeichnung des Netzwerkes
psk=<passphrase> # Passwort für WLAN-Zugriff
key_mgmt=WPA-PSK
}
:
```

Um diese Einstellungen tatsächlich zu aktivieren, müssen im Anschluss folgende Kommandos abgesetzt werden:

```

pi $ sudo systemctl restart dhcpcd
Warning: The unit file, source configuration file or drop-ins of dhcpcd.service changed on disk.
Run 'systemctl daemon-reload' to reload units.
pi $ sudo systemctl daemon-reload # gemäß obiger Warnung
pi $ ip addr
1: lo: <LOOPBACK LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: BROADCASTMULTICASTUPLOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b8:27:eb:36:6f:a7 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.153/24 brd 192.168.178.255 scope global dynamic noprefixroute eth0
        valid_lft 859570sec preferred_lft 751570sec
    inet6 fe80::3a9f:9d8c:1f44:e940/64 scope link
        valid_lft forever preferred_lft forever
3: wlan0: <BROADCASTMULTICASTUPLOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether b8:27:eb:63:3a:f2 brd ff:ff:ff:ff:ff:ff
    inet 192.168.178.154/24 brd 192.168.178.255 scope global dynamic noprefixroute wlan0
        valid_lft 863877sec preferred_lft 755877sec
    inet6 fe80::b04c:315b:a4c9:3c9d/64 scope link
        valid_lft forever preferred_lft forever
```

Damit ist die Phoniebox für den künftigen WLAN-Zugriff freigeschaltet. Der bis jetzt verwendete LAN-Zugriff über das Ethernetkabel ist ab sofort nicht mehr notwendig. Zum Test wird die laufende *ssh*-Sitzung beendet, das Ethernetkabel abgesteckt und eine neue *ssh*-Sitzung gestartet:

```

pi $ exit # Schließen der bestehenden ssh-Sitzung über LAN
Jetzt das Ethernetkabel abstecken!
PC $ ssh pi@phoniebox1 # neue ssh-Sitzung über WLAN öffnen
```

3 Installation der Software

Hinweis

Entgegen vieler Tutorials wurde bei schlizbädas Phoniebox auf die Zuteilung einer statischen (festen) IP-Adresse verzichtet! Stattdessen wird von der Phoniebox beim Bootvorgang über DHCP eine dynamische IP-Adresse angefordert, um den Netzwerkzugriff möglichst flexibel zu halten. Die konkret vergebene IP-Adresse kann mit dem Kommando `ip addr` ermittelt werden und lautet im obigen Beispiel 192.168.178.154.

3.2 OnOff SHIM

Der OnOff SHIM ist ein kleines Erweiterungsmodul, das ein komfortables Ein- und Ausschalten inklusive sauberem Herunterfahren des Raspberry Pi ermöglicht:

<https://shop.pimoroni.com/search?type=product&q=onoffshim>

Prinzipiell ist dieses Modul vom Konzept her gut durchdacht und für batteriebetriebene Raspberry Pi-Projekte gut geeignet. Allerdings führte bei mir ein zu langes Betätigen des On-/Off-Tasters dazu, dass der OnOff SHIM die Spannungsversorgung nicht vollständig ausschaltete, siehe Abschnitt 2.2.2.

Achtung

Die hier beschriebene Installation der Software verwendet **nicht** die Originalsoftware des Herstellers *Pimoroni*, sondern eine Variante, die unter <https://retropie.org.uk/forum/topic/15727/tutorial-onoff-shim-exposed-neat-powerswitch-from-pimoroni/> beschrieben wird!

Vom Hersteller *Pimoroni* wird zwar auch eine Software mitgeliefert, die als GNU/Linux-Daemon (Hintergrunddienst) arbeitet und mit folgendem Kommando aus dem Internet heruntergeladen und installiert werden könnte:

`pi $ curl https://get.pimoroni.com/onoffshim | bash # wurde so nicht durchgeführt!`

Diese Software war mir aber zu undurchsichtig, so dass ich lieber das einfachere und (für mich) durchschaubare Verfahren des Anwenders *cyberghost* aus dem Retropie-Forum (<https://retropie.org.uk/>) verwendete. Mir ist an dieser Stelle wichtig zu betonen, dass auch die Software von *Pimoroni* einwandfrei funktioniert und ebenso verwendet werden könnte.

Grundsätzlich ist der Einschaltvorgang beim OnOff SHIM ein rein hardwaremäßig umgesetztes Konzept, das keinerlei zusätzliche Software benötigt. Beim Ausschalten ist eine Software erforderlich, die die Betätigung des Tasters erkennt und das Betriebssystem vor dem Abschalten der Versorgungsspannung sauber beendet. Die Variante aus dem Retropie-Forum besteht aus lediglich zwei Shellskripten:

3 Installation der Software

Skript	Aufruf durch	Speicherort
<i>onoffshim_switch.sh</i>	<i>/etc/rc.local</i>	<i>/home/pi</i>
<i>onoffshim_gpio-shutoff.sh</i>	<i>systemd</i>	<i>/lib/systemd/system-shutdown</i>

Tabelle 3.2: Skripte für den OnOff SHIM

3.2.1 Installation der Shellskripte für den OnOff SHIM

Die beiden Shellskripte aus Tabelle 3.2 können vom PC aus dem Downloadverzeichnis dieser Dokumentation mit dem Kommando

```
PC $ scp ./files/onoffshim/*.sh pi@phoniebox1:/home/pi
```

in das Homeverzeichnis des Raspberry Pi kopiert werden.

Das Skript *onoffshim_switch.sh* muss auf dem Raspberry Pi beim Hochfahren gestartet werden. Dazu kann es im Homeverzeichnis des Benutzers verbleiben, muss aber mit Ausführrechten versehen werden:

```
pi $ chmod 755 onoffshim_switch.sh
```

Das Skript *onoffshim_gpio-shutoff.sh* dagegen soll beim Herunterfahren des Raspberry Pi über *systemd* aufgerufen werden. Dazu muss es mit Rootrechten in das Verzeichnis */lib/systemd/system-shutdown* kopiert werden, wo es von *systemd* gesucht und ausgeführt wird:

```
pi $ sudo mv onoffshim_gpio-shutoff.sh /lib/systemd/system-shutdown
pi $ sudo chown root:root /lib/systemd/system-shutdown/onoffshim_gpio-shutoff.sh
pi $ sudo chmod 755 /lib/systemd/system-shutdown/onoffshim_gpio-shutoff.sh
```

Nicht zwingend erforderlich, aber zur Bewahrung der Übersicht lohnt es sich, im Homeverzeichnis einen symbolischen Link zum neuen Speicherort des Skriptes *onoffshim_gpio-shutoff.sh* anzulegen:

```
pi $ ln -s /lib/systemd/system-shutdown/onoffshim_gpio-shutoff.sh
```

3.2.2 Einträge in den aufrufenden Dateien vornehmen

Damit das Skript *onoffshim_switch.sh* beim Hochfahren des Raspberry Pi gestartet wird, muss es in die Datei */etc/rc.local* eingetragen werden:

```
pi $ sudo nano /etc/rc.local
:
#!/bin/sh -e
#
# rc.local
```

3 Installation der Software

```

#
# 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.

# Print the IP address
:

# added by schlizbäda:
/home/pi/onoffshim_switch.sh & # do it in the background!

exit 0
:
```

3.2.3 Funktionsweise

Das Skript `onoffshim_switch.sh` wird beim Hochfahren über den Eintrag in `/etc/rc.local` im Hintergrund gestartet und läuft solange in einer Schleife, bis der Taster des OnOff SHIM an GPIO17 (Pin 11 der Stiftleiste) betätigt wird. Damit wird die Schleife verlassen und das Herunterfahren des Raspberry Pi durch Aufruf des Kommandos `poweroff` veranlasst.

`systemd` fährt den Raspberry Pi herunter. Im Laufe dieses Prozesses werden alle ausführbaren Dateien im Verzeichnis `/lib/systemd/system-shutdown` gestartet, in dem sich unser Shellskript `onoffshim_gpio-shutoff.sh` befindet:

Zunächst wird über den GPIO27 (Pin 13, Variable `$cut_pin`) das Relais angesteuert, um den Taster des OnOff SHIM künstlich zu trennen (siehe Kapitel 2.2.2). Der GPIO17 (Pin 11, Variable `$led_pin`) wird auf Ausgang umkonfiguriert, um zunächst in einer for-Schleife die rote LED auf dem OnOff SHIM dreimal blinken zu lassen. Schließlich wird der GPIO4 (Pin 7, Variable `$poweroff_pin`) als Ausgang konfiguriert und auf low geschaltet, damit der OnOff SHIM die Versorgungsspannung abschaltet.

Siehe auch:

<https://forum-raspberrypi.de/forum/thread/45820-phoniebox-2-0-rc7-mpd-spielt-ueber-rfid-karte-gewahlte-musik-nicht-ab/?postID=413400#post413400>

3.3 HifiBerry MiniAmp einrichten

Die Community ist sich uneins darüber, wie hochwertig die Audioqualität einer Phoniebox sein sollte. Meine persönliche Meinung ist, dass gerade Kinder noch ein wesentlich besseres Gehör haben als wir Erwachsenen, sowohl hinsichtlich des Frequenzspektrums bei Hochtönen als auch bei der Empfindlichkeit. Daher sollten mal 10 Euro Mehrausgaben für die Kleinen nicht **das** Entscheidungskriterium sein. Die Freude bei Groß und Klein wird sich bei einer qualitativ guten Umsetzung länger halten. Allerdings gibt es auch Gründe, gerade für noch recht kleine Kinder etwas leichtgewichtigere Komponenten in Form kleinerer Lautsprecher zu verwenden.

So ist auch der MiniAmp zusammen mit den verwendeten 8cm Visaton-Lautsprechern (siehe Stückliste in Tabelle 2.1) weit weg von HiFi-technischem *high end*, aber in meinen Augen ein guter Kompromiss.

Der MiniAmp muss in **Raspbian** bekanntgegeben und konfiguriert werden. Dazu sind Anpassungen in zwei Dateien erforderlich:

```
/boot/config.txt anpassen
pi $ sudo nano /boot/config.txt
:
# Enable audio (loads snd_bcm2835)
##dtparam=audio=on # Diese Zeile auskommentieren!
dtoverlay=hifiberry-dac
:

/etc/asound.conf anpassen
pi $ sudo nano /etc/asound.conf
:
pcm.hifiberryMiniAmp {
    type softvol
    slave.pcm "plughw:0"
    control.name "Master"
    control.card 0
}
pcm.!default {
    type plug
    slave.pcm "hifiberryMiniAmp"
}
:
```

Nach Änderungen an */boot/config.txt* muss der Raspberry Pi neu gebootet werden, damit die dort vorgenommenen Änderungen greifen:

3 Installation der Software

```
pi $ sudo reboot
PC $ ssh pi@phoniebox1 # über ssh neu auf dem RPi anmelden
```

Anschließend kann die Audioausgabe über den MiniAmp überprüft werden:

```
pi $ aplay -l # sollte folgende Ausgabe liefern:
```

```
**** List of PLAYBACK Hardware Devices ****
card 0: sndrpihifiberry [snd_rpi_hifiberry_dac], device 0: HifiBerry DAC HiFi pcm5102a-hifi-0 [HifiBerry
DAC HiFi pcm5102a-hifi-0]
Subdevices: 1/1
Subdevice #0: subdevice #0
```

Mit dem folgenden Kommando sollte an beiden Lautsprechern abwechselnd ein sogenanntes *Rosa Rauschen* ertönen. Dieser Test kann durch die Tastatureingabe **Ctrl-C** unterbrochen werden:

```
pi $ speaker-test -D hifiberryMiniAmp -c 2
speaker-test 1.1.8
```

```
Playback device is hifiberryMiniAmp
Stream parameters are 48000Hz, S16_LE, 2 channels
Using 16 octaves of pink noise
Rate set to 48000Hz (requested 48000Hz)
Buffer size range from 128 to 131072
Period size range from 64 to 65536
Using max buffer size 131072
Periods = 4
was set period_size = 32768
was set buffer_size = 131072
0 - Front Left
1 - Front Right
Time per period = 2.754591
0 - Front Left
1 - Front Right
```

```
pi $ alsamixer # Einstellung der Lautstärke
```

3 Installation der Software

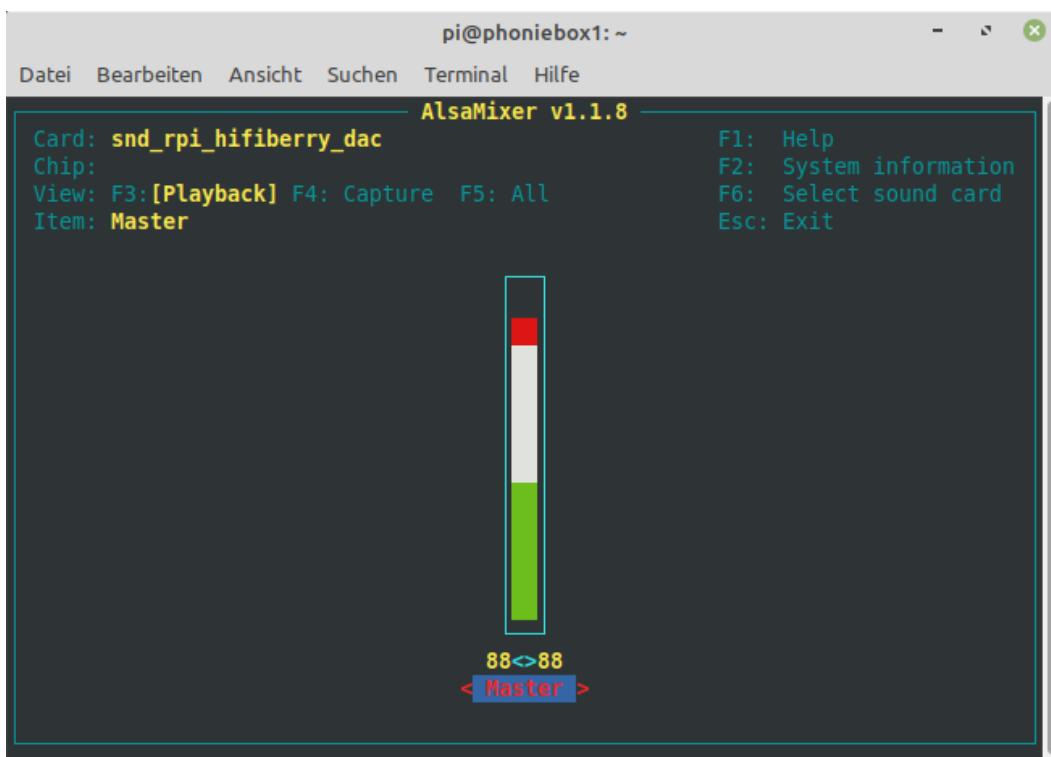


Abbildung 3.2: Lautstärkeeinstellung mit dem *alsamixer*

3.4 Music Player Daemon (*mpd*) einrichten

Der **Music Player Daemon (*mpd*)** ist eine Server-Anwendung zum Abspielen von Musikdateien in den gängigen Audiocodecs wie *mp3*, *Ogg Vorbis*, *FLAC*, *AAC*, *Mod* oder *WAV*. Clientsoftware kann den ***mpd*** über dessen Protokoll sowohl lokal als auch über das Netzwerk steuern. Damit ist er ideal für die Audiomeldung auf der Phoniebox geeignet.

Ein relativer Sprung innerhalb der gesamten Playlist für einen schnellen Vor- und Rücklauf kann über den ***mpd*-Standardclient *mpc*** mit dem Kommando `mpc seekthrough <sec>` abgesetzt werden. Ein Vorlauf wird über positive Werte *mit führendem Pluszeichen* abgesetzt, ein Rücklauf mit negativen Werten.

Hinweis

Nur leider ist der mit `apt-get install` aus der **Raspbian**-Distribution installierte ***mpd*** fehlerhaft: Negative Sprünge für den Rücklauf werden nicht ausgeführt! 😞
Es tritt nur ein kurzer Aussetzer in der aktuell wiedergegebenen Audiodatei auf.

Dieses Problem wurde im Raspberry Pi Forum bereits diskutiert:

<https://forum-raspberrypi.de/forum/thread/45381-schneller-vor-ruecklauf-durch-when-held-auf-next-prev-buttons-buttons-b?postID=435383#post435383>

Letztendlich kann der Fehler behoben werden, indem der Quellcode von der Projektseite <https://www.musicpd.org/> heruntergeladen und auf dem Raspberry Pi kompiliert wird.

3 Installation der Software

Danach muss der ***mpd*** richtig in ***systemd*** eingebunden werden, so dass er beim Booten von ***Raspbian*** gestartet wird.

3.4.1 Standardpaket des ***mpd*** aus ***Raspbian*** installieren

Zunächst wird der ***mpd*** direkt aus der Distribution installiert:

```
pi $ sudo apt install mpd mpc # inklusive dem Client mpc
```

Damit wird der ***mpd*** inklusive der erforderlichen Konfigurationsdateien installiert.

Standardmäßig wird der ***mpd*** im Verzeichnis ***/usr/bin*** installiert. Die Programmdatei wird aussagekräftig umbenannt und ein symbolischer Link namens ***mpd*** darauf erstellt:

```
pi $ mpd --version | head # Version ermitteln
```

```
Music Player Daemon 0.21.5 (0.21.5)
```

```
:
```

```
pi $ cd /usr/bin
```

```
pi $ sudo mv mpd mpd_0.21.5_raspbian
```

```
pi $ sudo ln -s /usr/bin/mpd_0.21.5_raspbian mpd
```

Hinweis

Es ist natürlich möglich, dass der *seek*-Fehler im ***mpd*** eines Tages behoben wird. Dann ist die Kompilierung nicht mehr erforderlich und es geht in Abschnitt [3.4.5](#) weiter.

Ebenso kann man den folgenden zeitaufwändigen und für den Anfänger durchaus lästigen Teil zunächst überspringen, um die Phoniebox an sich einigermaßen zügig zum Laufen zu bringen. Die Kompilierung des ***mpd*** kann dann bei Bedarf immer noch nachgeholt werden.

3.4.2 Herunterladen und Kompilieren des ***mpd***

Auf der Homepage des ***mpd***-Projektes ist die aktuell veröffentlichte Programmversion („stable“) direkt auf der Hauptseite unter <https://www.musicpd.org/> angegeben, siehe Abbildung [3.3](#).

Diese Versionsnummer (in diesem Beispiel **0.21.25**) ist wichtig für den Download des Quellcodes:

```
pi $ cd /home/pi # Download im Home-Verzeichnis durchführen!
```

```
pi $ wget https://www.musicpd.org/download/mpd/0.21/mpd-0.21.25.tar.xz
```

```
pi $ tar xf mpd-0.21.25.tar.xz # Quellcode entpacken
```

```
pi $ cd mpd-0.21.25
```

3 Installation der Software

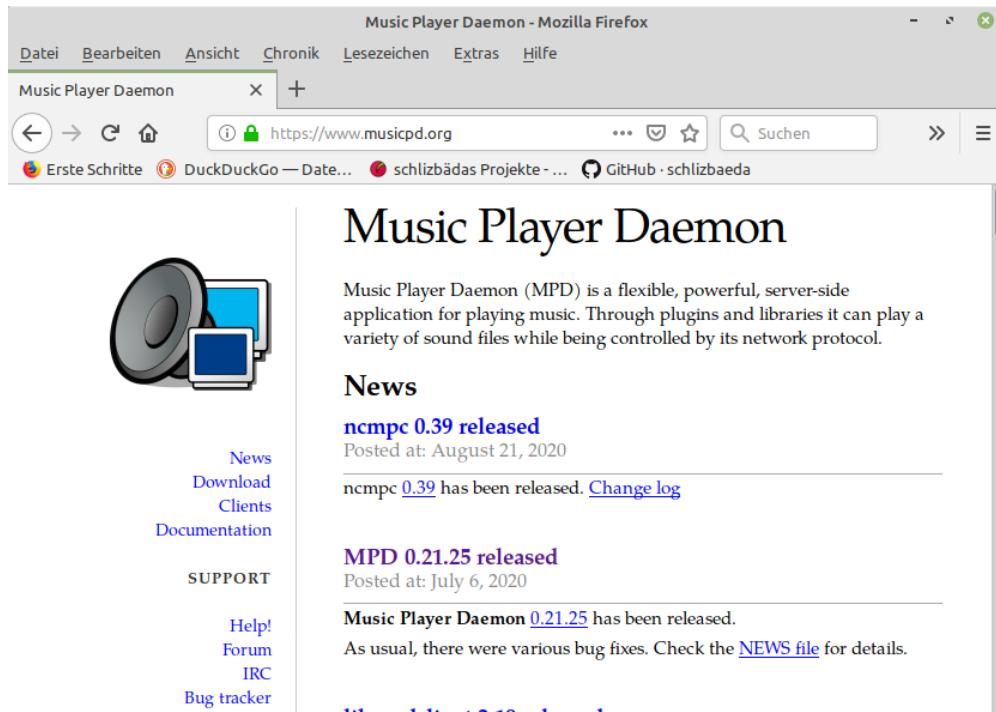


Abbildung 3.3: Homepage des *mpd*-Projektes mit Versionsangabe (0.21.25)

Unter <https://www.musicpd.org/doc/html/user.html#compiling-from-source> ist der eigentliche Kompiliervorgang beschrieben. Hier nur die notwendigen Kommandos:

```
pi $ echo apt install meson g++ \
libpcre3-dev libmad0-dev libmpg123-dev libid3tag0-dev \
libflac-dev libvorbis-dev libopus-dev libogg-dev \
libadplug-dev libaudiofile-dev libsndfile1-dev libfaad-dev \
libfluidsynth-dev libgme-dev libmikmod-dev libmodplug-dev \
libmpcdec-dev libwavpack-dev libwildmidi-dev \
libsidplay2-dev libsidutils-dev libresid-builder-dev \
libavcodec-dev libavformat-dev \
libmp3lame-dev libtwolame-dev libshine-dev \
libsamplerate0-dev libsoxr-dev libbz2-dev \
libcdio-paranoia-dev libiso9660-dev libmms-dev \
libzip-dev libcurl4-gnutls-dev libyajl-dev libexpat-dev \
libasound2-dev libao-dev libjack-jackd2-dev libopenal-dev \
libpulse-dev libshout3-dev libsndio-dev \
libmpdclient-dev libnfs-dev libsmbclient-dev \
libupnp-dev libavahi-client-dev libsqlite3-dev \
libsystemd-dev libgtest-dev libboost-dev libicu-dev \
libchromaprint-dev libgcrypt20-dev # alle benötigten Bibliotheken installieren
```

3 Installation der Software

Konfiguration des Quellcode-Projektes für den Kompilationsvorgang:

```
pi $ meson . output/release --buildtype=debugoptimized -Db_ndebug=true
pi $ meson configure output/release # Anzeige der Compileroptionen
pi $ ninja -C output/release # Kompilierung starten
```

Die Kompilierung besteht aus ca. 632 einzelnen Modulen. Entsprechend dauert dieser Vorgang auf einem Raspberry Pi 3B ca. 15 – 20 Minuten. Er läuft aber erstaunlich unproblematisch durch... 😊

Bewusst weggelassen wird das folgende Kommando, das den soeben kompilierten **mpd** unter **/usr/local/bin** installieren würde. Der schlizbäda hat keine Ahnung, wie er den neuen **mpd** an diesem neuen Pfad (**/usr/local/bin** statt **/usr/bin**) so in das Betriebssystem einbinden muss, dass die restlichen Programme gar nicht „merken“, dass hier etwas *gefaked* wurde. 🤪

```
pi $ # sudo ninja -C output/release install # nicht ausgeführt!
```

3.4.3 Installieren des **mpd**

Nun muss der neu kompilierte **mpd** so installiert werden, dass er beim Booten vom Betriebssystem geladen und gestartet wird. Dies geschieht mittlerweile in den meisten GNU/Linux-Distributionen, so auch in **Raspbian** mittels des Init-Systems **systemd**.

Zunächst wird das neu erstellte Kompilat nach **/usr/bin** kopiert und der bereits bestehende symbolische Link aus Kapitel 3.4.1 auf die neue Binary „umgebogen“:

```
pi $ cd /usr/bin
pi $ sudo cp /home/pi/mpd-0.21.25/output/release/mpd /usr/bin/mpd-0.21.25_compiled_by_schlizbaeda
pi $ sudo ln -s /usr/bin/mpd-0.21.25_compiled_by_schlizbaeda mpd
```



Achtung

Leider reicht das bloße Rüberkopieren der neuen Version von **mpd** nach **/usr/bin** nicht aus, um die alte Variante 1:1 durch die neue Variante zu ersetzen.

Der Hintergrunddienst **mpd** wird beim Hochfahren der Phoniebox automatisch über **systemd** gestartet. Allerdings schlägt der Start des neu kompilierten mpd fehl. Nun wäre interessant, warum es nicht funktioniert. Dies findet man mit folgendem Kommando heraus:

```
pi $ systemctl status mpd.service
# mpd.service - Music Player Daemon
_ Loaded: loaded (/etc/systemd/system/mpd.service; enabled; vendor preset: enabled)
_ Active: failed (Result: exit-code) since Mon 2020-08-24 21:02:39 CEST; 13s ago
_     Docs: man:mpd(1)
           man:mpd.conf(5)
           file:///usr/share/doc/mpd/user-manual.html
```

3 Installation der Software

```
_ Process: 6483 ExecStart=/usr/bin/mpd -no-daemon $MPDCONF (code=exited, status=1/FAILURE)
_Main PID: 6483 (code=exited, status=1/FAILURE)

Aug 24 21:02:38 phoniebox1 systemd[1]: Starting Music Player Daemon...
Aug 24 21:02:39 phoniebox1 mpd[6483]: exception: No configuration file found
Aug 24 21:02:39 phoniebox1 systemd[1]: mpd.service: Main process exited, code=exited, status=1/FAILURE
Aug 24 21:02:39 phoniebox1 systemd[1]: mpd.service: Failed with result 'exit-code'.
Aug 24 21:02:39 phoniebox1 systemd[1]: Failed to start Music Player Daemon.
```

systemd erkannte, dass das Aufrufkommando `/usr/bin/mpd -no-daemon $MPDCONF` mit einem fehlerhaften Exitcode ungleich 0 beendet wurde. Im neu kompilierten *mpd* trat also ein Fehler auf. Letzlich liegt es offenbar daran, dass mit dem neuen *mpd* auf die Shellvariable `$MPDCONF` irgendwie nicht richtig zugegriffen werden kann.

💡 Jamileckstamårsch!

| Das muss der Linux-Dauernoob schlizbäda nicht verstehen!

Der *quick+dirty*-Ansatz vom schlizbäda besteht darin, die Servicedatei von *systemd* so anzupassen, dass der Pfad der Konfigdatei `/etc/mpd.conf` nicht über die ominöse Shellvariable `$MPDCONF`, sondern direkt angegeben wird. Dazu wird die Servicedatei `/etc/systemd/system/mpd.service` angepasst:

```
pi $ sudo systemctl edit -full mpd.service # offizielle Bearbeitung der systemd-Servicedatei
:
[Service]
Type=notify
EnvironmentFile=/etc/default/mpd
#ExecStart=/usr/bin/mpd -no-daemon $MPDCONF # auskommentiert!
# adjusted by schlizbäda at 2020-08-16:
ExecStart=/usr/bin/mpd -no-daemon /etc/mpd.conf
:
pi $ sudo systemctl restart mpd.service # mpd neu starten
```

3.4.4 Resümee der Kompilierung von *mpd* – es wäre einfacher gewesen...

Das im *mpd*-Projekt enthaltene *make*-Installationsskript

```
sudo ninja -C output/release install
```

installiert die kompilierte Binärdatei nach `/usr/local/bin` und nicht nach `/usr/bin`. Dies ist aber – entgegen meiner ersten Einschätzung – kein Problem, da der Start des *mpd* über das Init-System *systemd* erfolgt. Alle Programmaufrufe, die über *systemd* gestartet werden, sind mit absolutem Pfad in den *systemd*-Steuerdateien, den sogenannten Units hinterlegt.

3 Installation der Software

Dort muss man „nur“ in der für ***mpd*** zuständigen Unit `/etc/systemd/system/mpd.service` den Aufrufpfad `/usr/local/bin` im Eintrag `ExecStart=...` setzen. Das im vorigen Kapitel beschriebene Umkopieren der Originalversion samt symbolischen Link auf die neu kompilierte Version hätte es *vermutlich* gar nicht gebraucht.

Leider habe ich das erst später gemerkt! Aber ich musste dann ohnehin an die Unit ran, denn meine neu kompilierte Variante des ***mpd*** kann nicht auf die Shellvariable `$MPDCONF` zugreifen.

Diesen Sachverhalt habe ich im deutschen Raspberry Pi Forum beschrieben unter

<https://forum-raspberrypi.de/forum/thread/45381-schneller-vor-ruecklauf-durch-when-held-auf-next-prev-buttons-buttons-b?postID=446062#post446062>

Das Init-System ***systemd*** ist unter <https://ubuntuusers.de/> recht gut beschrieben:

Allgemeine Erklärung: <https://wiki.ubuntuusers.de/systemd/>

Steuerdateien (Units): <https://wiki.ubuntuusers.de/systemd/Units>

Konfiguration mit `systemctl`: <https://wiki.ubuntuusers.de/systemd/systemctl/>



Hinweis zu einem echten Problem von FLOSS

Dies ist ein Beispiel für ein markantes und ernstzunehmendes Problem von FLOSS (Free/Libre Open Source Software): So behaupten viele Befürworter von FLOSS (auch der schlizbäda), dass nur mit quelloffenem Code verhindert werden könne, fehlerhaften Code oder gar Schadcode auf ein System einzuspielen. Oder zumindest, dass dies nachvollziehbar sei.

Allerdings tut die aus dem Raspbian-Repository installierte Programmversion von ***mpd*** nicht das, was im heruntergeladenen Quellcode drinsteht. Der ***mpd*** führt keinen Rücklauf aus. Aber wenn der heruntergeladene Code kompiliert wird, tut er es plötzlich doch. Es stellt sich unweigerlich die Frage, welcher Code denn nun für die Binärdatei im Raspbian-Repository verwendet wurde und was der mit dem von mir heruntergeladenen Code zu tun hat (wenngleich dies beim ***mpd*** keine Absicht ist).

Die obige Argumentation, dass ein System genau das macht, was in irgendwelchen veröffentlichten Quellcodedateien drinsteht, gilt eigentlich nur dann, wenn man das System auch wirklich selbst kompiliert!

FLOSS: https://de.wikipedia.org/wiki/Free_Libre_Open_Source_Software

3.4.5 Konfiguration des ***mpd***

Die gesamte Konfiguration des ***mpd*** wird in der Datei `/etc/mpd.conf` vorgenommen. Diese Datei besteht aus Einträgen in der Form *Schlüsselwort Wert* und in der Standardauslieferung aus vielen Kommentarzeilen. Letztlich müssen folgende Einträge ergänzt oder angepasst werden:

3 Installation der Software

```
pi $ sudo nano /etc/mpd.conf
:
music_directory "/home/pi/RPi-Jukebox-RFID/shared/audiofolders"
playlist_directory "/home/pi/RPi-Jukebox-RFID/playlists"
user "root"
auto_update "yes"
auto_update_depth "10"
mixer_control "Master"
:
```

Folgendes Kommando dient dazu, die Bezeichnung für den Eintrag `mixer_control` herauszufinden:

```
pi $ amixer scontrols # liefert z.B. bei MiniAmp-Installation
```

Simple mixer control 'Master',0 **Der `mpd`** wird mit der neuen Konfiguration mit folgendem Kommando gestartet:

```
pi $ mpc update
```

Test von `mpd`

Jetzt ist der Zeitpunkt gekommen, einen geschmeidigen Audiotest mit **`mpd`** durchzuführen. Dazu zunächst vom PC eine Audiodatei auf den Raspberry Pi in das oben konfigurierte Audioverzeichnis `/home/pi/RPi-Jukebox-RFID/shared/audiofolders` kopieren und mit dem Client **`mpc`** abspielen:

```
PC $ scp /Pfad/Musik.flac pi@phoniebox1:/home/pi/RPi-Jukebox-RFID/shared/audiofolders
```

```
pi $ mpc add /home/pi/RPi-Jukebox-RFID/shared/audiofolders/Musik.flac
```

```
pi $ mpc play
```

Sollte keine Musik hörbar sein, mit dem **`alsamixer`** die Lautstärke überprüfen und anpassen:

```
pi $ alsamixer
```

(siehe Abbildung 3.2)

3.5 Manuelle Installation der Phoniebox-Software

Hinweis

Eigentlich sollte die Installation mit Hilfe des sogenannten *one line installers* funktionieren. Mit etwas Glück läuft die Installation problemlos durch, weil alle gemeldeten Fehler durch die Projekt-Maintainer ausgemerzt wurden. Insbesondere Neulinge sollten zuerst den *one line installer* ausprobieren, bevor sie sich an den folgenden Einzelkommandos austoben 😊

<https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/INSTALL-stretch#one-line-install-command>

Bei Erfolg kann dieses Kapitel ggf. komplett übersprungen werden. 😊

Beim schlizbäda lief es natürlich anders und er führte dann unter **Raspbian Buster Lite** Anfang Januar 2020 die Installation der Phoniebox-Software manuell durch. Dabei traten diverse Probleme auf, die bei den einzelnen Schritten beschrieben werden. Es kann natürlich jederzeit sein, dass aufgrund von Aktualisierungen im github-Repository sowohl der *one line installer* als auch die Einzelkommandos wieder funktionieren...

Dieses Kapitel orientiert sich dabei an der Installationsanleitung auf github ab dem Abschnitt *install required packages and the Phoniebox code*:

<https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/INSTALL-stretch#install-required-packages-and-the-phoniebox-code>

3.5.1 benötigte Programmpakete und -bibliotheken installieren

Für den Betrieb der Phoniebox-Software sind etliche Programmpakete erforderlich, die über apt-get installiert werden. Vor der Installation empfiehlt es sich, zunächst das Kommando-paar

`pi $ sudo apt-get update && sudo apt-get upgrade`

abzusetzen, da sich gerade bei **Raspbian** in den Repositorys ständig viel ändert!

```
pi $ sudo apt-get install apt-transport-https
pi $ sudo apt-get install samba samba-common-bin # siehe Abfrage aus Abbildung 3.4
pi $ sudo apt-get install python-dev python-pip
pi $ sudo apt-get install python3-dev python3-pip # Python3 ist hier kein Fehler!
pi $ sudo apt-get install gcc raspberrypi-kernel-headers lighttpd
pi $ sudo apt-get install php7.3-common php7.3-cgi php7.3 php7.3-fpm
pi $ sudo apt-get install at mpg123
pi $ sudo apt-get install mpd mpc # geschah schon in Kapitel 3.4
pi $ sudo apt-get install git ffmpeg python-mutagen
pi $ sudo apt-get install python3-mutagen # Python3 schadet auch hier auf keinen Fall!
```

3 Installation der Software

Hinweis

In Kapitel 3.10 hat mich dann auch das Fehlen von Python3 eingeholt, weil in der Originalanleitung in Abschnitt 3.5.3 nur die Python2-Version installiert wurde... 

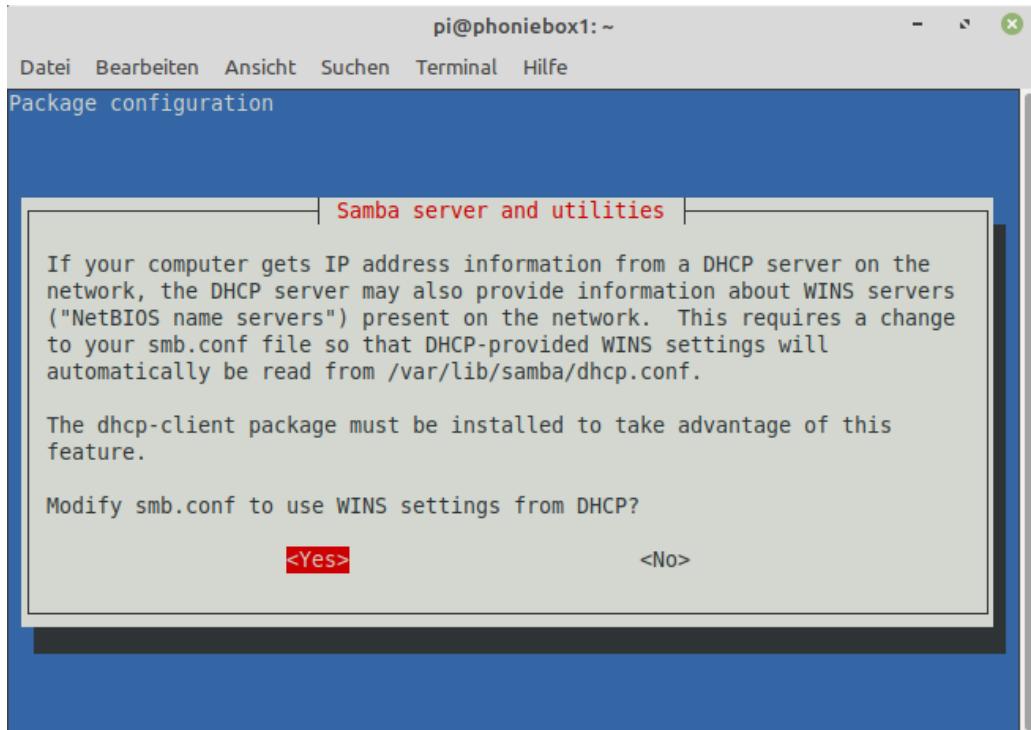


Abbildung 3.4: Abfrage bei der Installation von **Samba** mit *apt-get*

Bei der Installation von **Samba** erscheint die Abfrage aus Abbildung 3.4, die wie abgebildet mit Yes beantwortet wird.

3.5.2 Projekt RPi-Jukebox-RFID von *github* klonen und installieren

```
pi $ git clone https://github.com/MiczFlor/RPi-Jukebox-RFID.git
pi $ cd RPi-Jukebox-RFID
```

In der Datei requirements.txt ist (vermutlich zu einem alten Stand herührend) das Paket spidev auskommentiert.

```
pi $ nano requirements.txt
:
# Library dependencies for the python code. You need to install these with
# 'sudo pip install -r requirements.txt' before you can run this.

#### ESSENTIAL LIBRARIES FOR MAIN FUNCTIONALITY ####
```

3 Installation der Software

```
# related libraries.
evdev==0.7.0
git+git://github.com/lthiery/SPI-Py.git#egg=spi-py
youtube_dl
pyserial
# spidev - currently installed via apt-get # eben nicht!
RPi.GPIO
pi-rc522
:
```



Achtung

Im Gegensatz zum Kommentar in der roten Zeile wird das Paket *spidev* nicht installiert. Dies führt zu folgender Fehlermeldung beim Kommando:

```
pi $ pip install -r requirements.txt
```

```
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting evdev==0.7.0 (from -r requirements.txt (line 7))
  Downloading https://files.pythonhosted.org/packages/67/15/eac376f3e1fc1960a54439c21459b2582e68340001aff83b...
Collecting spi-py from git+git://github.com/lthiery/SPI-Py.git#egg=spi-py (from -r requirements.txt (line 8))
  Cloning git://github.com/lthiery/SPI-Py.git to /tmp/pip-install-uDNQOf/spi-py
Collecting youtube_dl (from -r requirements.txt (line 9))
  Downloading https://files.pythonhosted.org/packages/d0/b3/c3d42f6bbf91da104c272950d30923c222061d7323aa43dc...
    100% |xxxxxxxxxxxxxxxxxxxxxxxxxxxxx| 1.8MB 129kB/s
Collecting pyserial (from -r requirements.txt (line 10))
  Downloading https://files.pythonhosted.org/packages/0d/e4/2a744dd9e3be04a0c0907414e2a01a7c88bb3915cbe3c8cc...
    100% |xxxxxxxxxxxxxxxxxxxxxxxxxxxxx| 194kB 608kB/s
Requirement already satisfied: RPi.GPIO in /usr/lib/python2.7/dist-packages (from -r requirements.txt (line 1))
Collecting pi-rc522 (from -r requirements.txt (line 13))
  Downloading https://files.pythonhosted.org/packages/84/f2/e3d02257949e9caa9bff26044e0185e743ff45d3e0e099a93...
    Complete output from command python setup.py egg_info:
    Traceback (most recent call last):
      File "<string>", line 1, in <module>
      File "/tmp/pip-install-uDNQOf/pi-rc522/setup.py", line 11, in <module>
        from pirc522 import __version__ # flake8: noqa
      File "pirc522/__init__.py", line 4, in <module>
        from .rfid import RFID
      File "pirc522/rfid.py", line 2, in <module>
        import spidev
    ImportError: No module named spidev
```

Command "python setup.py egg_info" failed with error code 1 in /tmp/pip-install-uDNQOf/pi-rc522/

Deshalb die Python-Software so installieren:

```
pi $ echo spidev >spidev.txt
```

3 Installation der Software

```
pi $ pip install -r spidev.txt  
pi $ pip install -r requirements.txt
```

Hiermit läuft die Python-Installation fehlerfrei durch!

3.5.3 Einbinden des RFID-Lesers als Python(?)-Event

```
pi $ pip install evdev
```

```
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Requirement already satisfied: evdev in /home/pi/.local/lib/python2.7/site-packages (0.7.0)
```

Hinweis

Aufgrund der obigen Meldung scheint das Python2-Paket `evdev` bereits aktuell zu sein.

Viel wichtiger ist aber, dass diese Bibliothek anschließend auch für **Python3** installiert wird!

```
pi $ pip3 install evdev
```

```
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple  
Collecting evdev  
  Downloading https://www.piwheels.org/simple/evdev/evdev-1.3.0-cp37-cp37m-linux_armv7l.whl (98kB)  
    100% |xxxxxxxxxxxxxxxxxxxxxxxxxxxxx| 102kB 628kB/s  
Installing collected packages: evdev  
Successfully installed evdev-1.3.0
```

3.5.4 Webserver *lighttpd* für die Web-App der Phoniebox einrichten

Auch bei der Web-App läuft's ganz und gar nicht wie geschmiert. 😊

Datei `/etc/lighttpd/lighttpd.conf` bearbeiten

In der Datei `/etc/lighttpd/lighttpd.conf` müssen folgende Änderungen (rot) ergänzt werden:

```
pi $ sudo nano /etc/lighttpd/lighttpd.conf  
:  
# adjusted by schlizbäda due to https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/INSTALL-stretch  
server.document-root = "/home/pi/RPi-Jukebox-RFID/htdocs"  
#server.document-root = "/var/www/html"  
server.upload-dirs = ( "/var/cache/lighttpd/uploads")  
server.errorlog = "/var/log/lighttpd/error.log"  
server.pid-file = "/var/run/lighttpd.pid"  
server.username = "www-data"
```

3 Installation der Software

```
server.groupname = "www-data"
server.port = 80
:
```

Datei */etc/sudoers* bearbeiten

Hier muss hinten der rot eingefärbte Teil angehängt werden:

```
:
#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#
Defaults env_reset
Defaults mail_badpass
Defaults secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
:
#
# added by schlzbäda:
www-data ALL=(ALL) NOPASSWD: ALL
:
```

Modul fastcgi für den Webserver *lighttpd* konfigurieren

Damit der Webserver die bei der Phoniebox zum Einsatz kommenden HP-Scripte ausführen kann, muss das Modul fastcgi mit den folgenden Kommandos einmalig aktiviert werden:

```
pi $ sudo lighttpd-enable-mod fastcgi
Enabling fastcgi: ok
Run "service lighttpd force-reload" to enable changes
pi $ sudo lighttpd-enable-mod fastcgi-php
Enabling fastcgi-php: ok
Run "service lighttpd force-reload" to enable changes
```

Die neue Konfiguration des *lighttpd*-Webservers neu laden:

```
pi $ sudo service lighttpd force-reload
```

Upload von Audiodateien über den Webservice ermöglichen

Um den Webservice für den Upload von Audiodateien auf die Phoniebox zu ertüchtigen, muss die Datei */etc/php/7.3/fpm/php.ini* angelegt werden. Dazu ist im github-Repository der Phoniebox eine Beispieldatei vorhanden:

```
pi $ sudo cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/php.ini.stretch-default.sample /etc/php/7.3/fpm/php.ini
```

3 Installation der Software

```
pi $ sudo chown root:root /etc/php/7.3/fpm/php.ini  
pi $ sudo chmod 644 /etc/php/7.3/fpm/php.ini
```

In der Datei `/etc/php/7.3/fpm/php.ini` sind folgende Einträge zu kontrollieren und gegebenenfalls zu ergänzen:

```
pi $ sudo nano /etc/php/7.3/fpm/php.ini  
:  
file_uploads = On  
upload_max_filesize = 0  
max_file_uploads = 20  
post_max_size = 0  
:  
pi $ sudo service php7.3-fpm restart
```

Konfigurationsdatei für die Web-App aus der Beispieldatei kopieren

```
pi $ cp /home/pi/RPi-Jukebox-RFID/htdocs/config.php.sample /home/pi/RPi-Jukebox-RFID/config.php
```

Zugriff auf Verzeichnisse `shared` und `htdocs` für den Webservice ermöglichen

Der Zugriff erfolgt unter dem Benutzer `www-data`, daher muss der Zugriff auf die beiden Verzeichnisse für diesen Benutzer über Gruppenrechte ermöglicht werden:

```
pi $ sudo chown -R pi:www-data /home/pi/RPi-Jukebox-RFID/shared  
pi $ sudo chmod -R 775 /home/pi/RPi-Jukebox-RFID/shared  
pi $ sudo chown -R pi:www-data /home/pi/RPi-Jukebox-RFID/htdocs  
pi $ sudo chmod -R 775 /home/pi/RPi-Jukebox-RFID/htdocs
```

3 Installation der Software

Kontrolle im Browser:

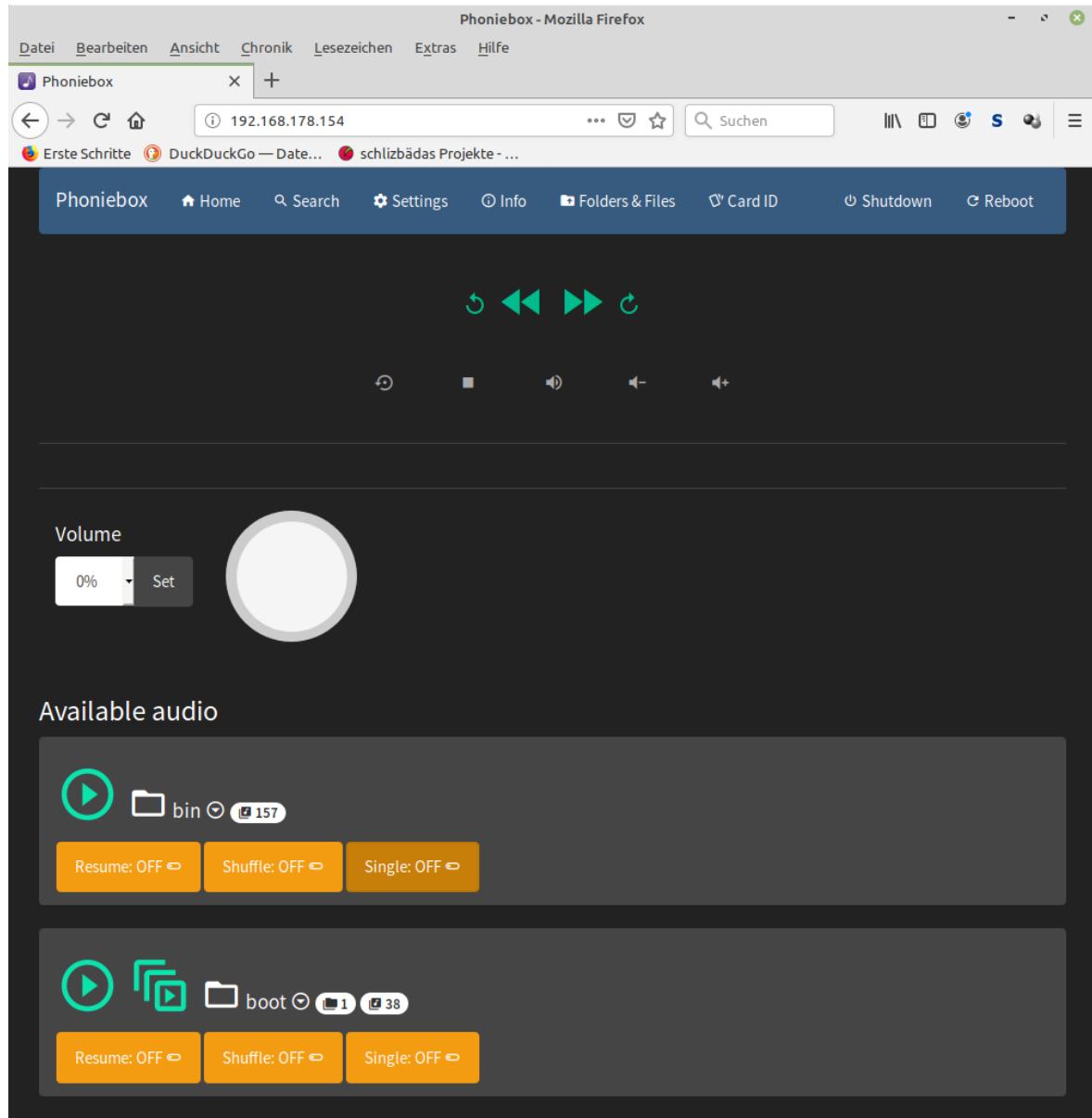


Abbildung 3.5: Aufruf des *lighttpd*-Webservers

Hintergrunddetails zur Einrichtung des Webservers siehe:

<https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/INSTALL-stretch#lighttpd-web-server-for-web-app>

3 Installation der Software

 **Achtung**

Ab hier bezieht sich diese Dokumentation auf die Seite

<https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/CONFIGURE-stretch>
von MiczFlor!

3.6 RFID-Leser installieren

In diesem Abschnitt wird die Installation des *Neuftech USB RFID-Reader 125kHz* beschrieben, der in schlizbädas Phoniebox verbaut wurde und bei amazon erhältlich ist:

<https://www.amazon.de/Neuftech-Reader-Kartenleseger%C3%A4t-Kartenleser-Kontaktlos/dp/B0180Y0R3E>

Dieses Gerät emuliert am USB-Anschluss eine Tastatur (ein Gerät aus der USB-Klasse *Human Interface Device*, https://de.wikipedia.org/wiki/Human_Interface_Device).

Die Phoniebox-Software fängt Ereignisse vom RFID-Leser jedoch über *udev* ab:

<https://wiki.ubuntuusers.de/udev/>

3.6.1 RFID-Leser überprüfen

Einen unter **Raspbian** korrekt installierten RFID-Leser erkennt man durch Kontrolle folgender Punkte

- Die LED des RFID-Lesers leuchtet – im eingebauten Zustand evtl. nicht erkennbar
- Piepton beim Hochfahren
- Piepton beim Lesen einer RFID-Karte
- Überprüfen, ob das Softwareevent für den RFID-Leser ordnungsgemäß eingerichtet ist:

```
pi $ ls -la /dev/input/by-id
total 0
drwxr-xr-x 2 root root 60 Jan 11 15:36 .
drwxr-xr-x 4 root root 120 Jan 11 15:36 ..
lrwxrwxrwx 1 root root 9 Jan 11 15:36 usb-HXGCoLtd_27db-event-kbd -> ../event0
```

3.6.2 RFID-Leser in der Phoniebox-Software registrieren

```
pi $ cd /home/pi/RPi-Jukebox-RFID/scripts
pi $ python3 RegisterDevice.py # liefert z.B. :
Choose the reader from list
0 HID 046a:0011
1 HXGCoLtd Keyboard
```

3 Installation der Software

Device Number: 1

Unter GNU/Linux wird der Chipsatz des RFID-Lesers als *HXGCoLtd Keyboard* erkannt. Daher ist bei der Abfrage der Device Number in diesem Beispiel der Wert 1 einzugeben.

Ob die Registrierung des RFID-Lesers erfolgreich war, kann mit der Datei */home/pi/RPi-Jukebox-RFID/scripts/deviceName.txt* geprüft werden. Diese Datei enthält den oben gewählten Gerätenamen:

```
pi $ cat deviceName.txt
HXGCoLtd Keyboard
```

3.6.3 RFID-Konfigurationsdatei der Phoniebox-Software anlegen

```
pi $ cd /home/pi/RPi-Jukebox-RFID/settings
pi $ cp rfid_trigger_play.conf.sample rfid_trigger_play.conf
pi $ sudo chown pi:pi rfid_trigger_play.conf
pi $ sudo chmod 665 rfid_trigger_play.conf
```

3.7 Konfigurationsdateien im Verzeichnis

RPi-Jukebox-RFID/settings

Das Unterverzeichnis filenam/*/home/pi/RPi-Jukebox-RFID/settings* enthält einige Konfigurationsdateien, in denen das Verhalten der Phoniebox eingestellt und angepasst werden kann. Der Dateinhalt besteht dabei nur aus dem gewünschten Parameter. Die Funktion wird durch den Dateinamen bestimmt und ist in den jeweiligen Python- bzw. Shellskripten fest hinterlegt, siehe

<https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/CONFIGURE-stretch#create-settings-for-audio-layout>

3.7.1 Audioeinstellungen

Die folgenden Kommandos legen im Verzeichnis */home/pi/RPi-Jukebox-RFID/settings* kleine Textdateien an, die das Verhalten der Taster *volume up* und *volume down* konfigurieren. Ferner kann die maximale Lautstärke begrenzt werden:

```
pi $ echo "Master" > /home/pi/RPi-Jukebox-RFID/settings/Audio_iFace_Name
pi $ echo "3" > /home/pi/RPi-Jukebox-RFID/settings/Audio_Volume_Change_Step
pi $ echo "100" > /home/pi/RPi-Jukebox-RFID/settings/Max_Volume_Limit
```

MP3-Dateien für Startup und Shutdown kopieren:

```
pi $ cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/startupsound.mp3.sample
/home/pi/RPi-Jukebox-RFID/shared/startupsound.mp3
```

3 Installation der Software

```
pi $ cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/shutdownsound.mp3.sample
/home/pi/RPi-Jukebox-RFID/shared/shutdownsound.mp3
```

3.7.2 automatische Abschaltung bei Nichtbenutzung

Gerade bei Kindern ist es sinnvoll, die Phoniebox nach einiger Zeit der Nichtverwendung automatisch abzuschalten, um die Akkulaufzeit zu verlängern. Der angegebene Wert ist inaktive Zeit in Minuten *ab dem Beenden der mpd-Playlist*, bevor sich die Box automatisch abschaltet. Der Wert 0 deaktiviert diese Funktion.

```
pi $ echo "0" > /home/pi/RPi-Jukebox-RFID/settings/Idle_Time_Before_Shutdown
```

Ein automatisches Abschalten der Phoniebox nach 15 Minuten Nichtbenutzung erreicht man alternativ durch:

```
pi $ echo "15" > /home/pi/RPi-Jukebox-RFID/settings/Idle_Time_Before_Shutdown
```

3.7.3 Was soll beim wiederholten Auflegen einer RFID-Karte passieren?

Quelle:

<https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/MANUAL#second-swipe>

In der Datei */home/pi/RPi-Jukebox-RFID/settings/Second_Swipe* wird festgelegt, was beim wiederholten Auflegen derselben RFID-Karte passieren soll. Dazu wird in diese Datei das entsprechende Schlüsselwort aus folgender Liste eingetragen:

Schlüsselwort	Funktion
RESTART	Neustart der aktiven Playlist von Anfang an
PAUSE	Umschalten zwischen <i>Play/Pause</i>
SKIPNEXT	Zum nächsten Titel aus der Playlist springen
NOAUDIOPLAY	Keine Beeinflussung der Audiomeldung

Tabelle 3.3: Verhalten bei wiederholtem Auflegen derselben RFID-Karte

Bei meiner Box habe ich mich für den Eintrag PAUSE entschieden.

```
pi $ nano /home/pi/RPi-Jukebox-RFID/settings/Second_Swipe
```

```
PAUSE
```



Hinweis

| In der Webapp ist die Einstellung sicherer...

3 Installation der Software

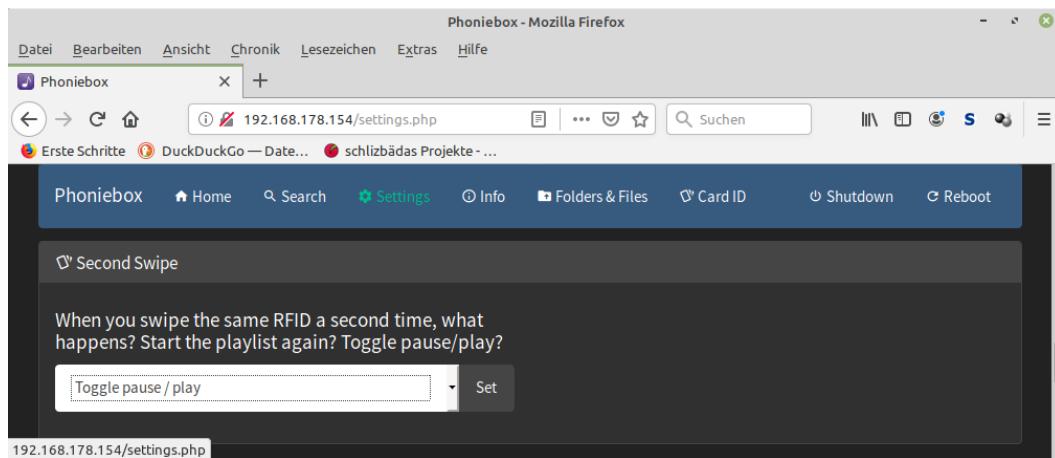


Abbildung 3.6: Einstellung des Verhaltens bei wiederholtem Auflegen derselben RFID-Karte über die WebApp

3.8 Die Phoniebox mit Tastern an der GPIO-Leiste steuern

Ursprünglich war vorgesehen, bestimmte RFID-Karten mit Steuerungsfunktionen zu belegen. Dieses Konzept ist aber gerade bei Kindern ungünstig, da (zumindest meine) Kinder solche Sachen in den Tiefen ihrer Zimmer sehr gut verstecken und die Phoniebox damit zwischenzeitlich(?) unbedienbar wird.

Daher hat MiczFlor in seinem github-Repository das Kapitel <https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/Using-GPIO-hardware-buttons> eingefügt, in dem die Inbetriebnahme der Steuerung über Taster an der GPIO-Leiste beschrieben wird.

Achtung

Die Pinbelegung in der Anleitung von MiczFlor ordnet den Tastern Pins zu, die für die digitale Audioübertragung über I2S zum HifiBerry MiniAmp benötigt werden! Daher muss das Python Skript für die Abfrage der Taster stark angepasst werden!

Für den Zugriff auf die GPIOs wird die Python-Bibliothek **gpiozero** verwendet, von der es jeweils eine eigene Variante für Python2 und Python 3 gibt. Installation der Bibliothek über folgendes Kommando:

```
pi $ sudo apt-get install python3-gpiozero python-gpiozero # für Python2 und 3!
```

Zur Abfrage der Taster wird das Python Skript **/home/pi/RPi-Jukebox-RFID/scripts/gpio-buttons.py** verwendet. Die Originalvorlage könnte mit folgendem Kommando installiert werden:

```
pi $ sudo cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/gpio-buttons.py.sample
/home/pi/RPi-Jukebox-RFID/scripts/gpio-buttons.py # funktioniert nicht mit dem MiniAmp!
Allerdings kollidiert die Pinvergabe mit der Pinzuordnung des HifiBerry MiniAmps!
```

3 Installation der Software

Bei Einbau des MiniAmps muss die von schlizbäda angepasste Datei verwendet werden! Diese Datei entspricht der Pinbelegung der Lochrasterplatine aus Kapitel 2.2.1:

```
PC $ scp ./files/GPIO/gpio-buttons.py pi@phoniebox1:/home/pi/RPi-Jukebox-RFID/scripts
pi $ nano /home/pi/RPi-Jukebox-RFID/scripts/gpio-buttons.py
#shut = Button(3,hold_time=2) # --> Pin 5 # no longer necessary due to OnOffShim
vol0 = Button(13,pull_up=True) # --> Pin 33
volU = Button(12,pull_up=True,hold_time=0.3,hold_repeat=True) # --> Pin 32
volD = Button(6,pull_up=True,hold_time=0.3,hold_repeat=True) # --> Pin 31
next = Button(7,pull_up=True) # --> Pin 26
prev = Button(8,pull_up=True) # --> Pin 24
halt = Button(5,pull_up=True) # --> Pin 29
#reco = Button(6, pull_up=True) # Choose GPIO to fit your hardware
#play = Button(12,pull_up=True) # Choose GPIO to fit your hardware
```

siehe auch Tabelle 2.2

3.9 Zuordnung von Hörspielen zu RFID-Karten



Hinweis
Die funktionalen Elemente der Phoniebox sind nun installiert. Jetzt ist es an der Zeit, einzelnen Hörspielen oder Musikstücken RFID-Karten zuzuordnen und einen *Soundcheck* zu machen, um die grundlegende Funktionalität der Box zu überprüfen:



Abbildung 3.7: Audiodaten (Musikalben) auf die Phoniebox kopieren

Zunächst werden *rekursiv* zwei Unterverzeichnisse mit den Musikdateien zweier Musikalben vom PC in den Audioordner auf der Phoniebox kopiert:

3 Installation der Software

```
PC $ scp -r Helloween pi@phoniebox1:/home/pi/RPi-Jukebox-RFID/shared/audiofolders
PC $ scp -r Metallica pi@phoniebox1:/home/pi/RPi-Jukebox-RFID/shared/audiofolders

Kontrolle mit ls -lR auf der Phoniebox:
pi $ ls -lR /home/pi/RPi-Jukebox-RFID/shared/audiofolders

total 26728
drwxr-xr-x 2 pi pi          4096 May 17 15:50 Helloween
drwxr-xr-x 2 pi pi          4096 May 17 15:56 Metallica

./Helloween:
total 104140
-rw xr-xr-x 1 pi pi 37314544 May 17 15:47 01_I_Want_Out.flac
-rw xr-xr-x 1 pi pi 38112711 May 17 15:48 02_Dr._Stein.flac
-rw xr-xr-x 1 pi pi 31203347 May 17 15:49 03_Future_World.flac

./Metallica:
total 151828
-rw xr-xr-x 1 pi pi 38978523 May 17 15:52 01_Enter_Sandman.flac
-rw xr-xr-x 1 pi pi 41647915 May 17 15:54 02_Sad_But_True.flac
-rw xr-xr-x 1 pi pi 29319722 May 17 15:54 03_Holier_Than_Thou.flac
-rw xr-xr-x 1 pi pi 45518619 May 17 15:56 04_The_Unforgiven.flac
```

3.9.1 RFID-Karte manuell zuordnen

Um nun diesen beiden Alben zwei neue RFID-Karten zuzuordnen, sind folgende Schritte erforderlich:

- Die einzelnen Audiodateien eines Hörspiels oder Musikalbums ins Verzeichnis `/home/pi/RPi-Jukebox-RFID/shared/audiofolders/ALBUM` kopieren.
- m3u-Datei `ALBUM.m3u` mit *absoluten Pfadangaben*(!) im Verzeichnis `/home/pi/RPi-Jukebox-RFID/playlists` anlegen.
- RFID-Karte dem Album im Verzeichnis `/home/pi/RPi-Jukebox-RFID/shared/shortcuts` zuordnen.

Im folgenden wird das Album „Helloween“ nur durch Verwendung von Shellkommandos eingerichtet und einer RFID-Karte zugeordnet:

Der erste Schritt, die Audiodateien nach `/home/pi/RPi-Jukebox-RFID/shared/audiofolders` zu kopieren, wurde bereits im vorherigen Abschnitt erläutert.

Zweiter Schritt: Da aber in den einzelnen Unterverzeichnissen (Alben) in `/home/pi/RPi-Jukebox-RFID/shared/audiofolders` neben den eigentlichen Audiodateien auch noch beliebige andere Dateien abgespeichert sein können, muss es eine Datei geben, in der die Playlist abgelegt ist. Dies ist eine im Unterverzeichnis `/home/pi/RPi-Jukebox-RFID/playlists`

3 Installation der Software

abgelegte m3u-Datei, deren Dateiname gleich lauten muss wie der Name des Unterverzeichnisses mit den Audiodateien in `/home/pi/RPi-Jukebox-RFID/shared/audiofolders`, also allgemein gesprochen ***ALBUM.m3u*** oder konkret in unserem Beispiel ***Helloween.m3u***. Beim Inhalt der m3u-Datei muss darauf geachtet werden, dass für jede einzelne Audiodatei der vollständige absolute Pfad angegeben wird:

```
pi $ nano /home/pi/RPi-Jukebox-RFID/playlists/Helloween.m3u
/home/pi/RPi-Jukebox-RFID/shared/audiofolders/Helloween/01_I_Want_Out.flac
/home/pi/RPi-Jukebox-RFID/shared/audiofolders/Helloween/02_Dr._Stein.flac
/home/pi/RPi-Jukebox-RFID/shared/audiofolders/Helloween/03_Future_World.flac
```

Weitere Informationen zum Playlist-Format *m3u* findet man unter <https://de.wikipedia.org/wiki/M3U>.

Schließlich muss im dritten Schritt das Album „Helloween“ auf unserer Phoniebox noch einer RFID-Karte zugeordnet werden:

Um die ID herauszufinden, wird eine neue RFID-Karte über den RFID-Leser eingelesen. Dabei wird im Unterverzeichnis `/home/pi/RPi-Jukebox-RFID/shared/shortcuts` eine Datei angelegt, die den Namen der Karten-ID hat, z. B. 0009563230. Der Inhalt dieser Datei muss auf den Verzeichnisnamen des Albums oder Hörbuchs unter

`/home/pi/RPi-Jukebox-RFID/shared/audiofolders` angepasst werden, z. B. auf „Helloween“.

```
pi $ nano /home/pi/RPi-Jukebox-RFID/shared/shortcuts/0009563230
Helloween
```

Nach erneutem Auflegen der RFID-Karte mit der ID 0009563230 werden die in der Playlist `/home/pi/RPi-Jukebox-RFID/playlists/Helloween.m3u` hinterlegten Musiktitel abgespielt.

3.9.2 RFID-Karte über die Webanwendung registrieren

Alternativ kann eine RFID-Karte auch mit Hilfe des Webservices zugeordnet werden. Das Vorgehen wird in der folgenden Bilderstrecke beispielhaft für den zweiten Ordner „Metallica“ dargestellt:

Am PC wird über einen Internetbrowser (Firefox) eine Verbindung zur Phoniebox hergestellt, z.B. durch Eingabe der IP-Adresse (hier 192.168.178.154). Durch Klick auf **"Settings"** in der oberen Menüleiste erscheint das Bild links oben. Hier muss auf die Schaltfläche **[Register new card ID]** geklickt werden. Nun erscheint das Bild rechts oben. Jetzt die neue RFID-Karte über den RFID-Leser einlesen. Im dazugehörigen Textfeld wird die ID aktualisiert (hier 0009593627).

In der Zeile **"Audio Folder"** werden im Auswahlfeld (ComboBox) alle Unterverzeichnisse von `/home/pi/RPi-Jukebox-RFID/shared/audiofolders` angezeigt. Hier den gewünschten

PHONIEBOX – DIE OPEN-SOURCE-ALTERNATIVE ZUR TONIEBOX

3 Installation der Software

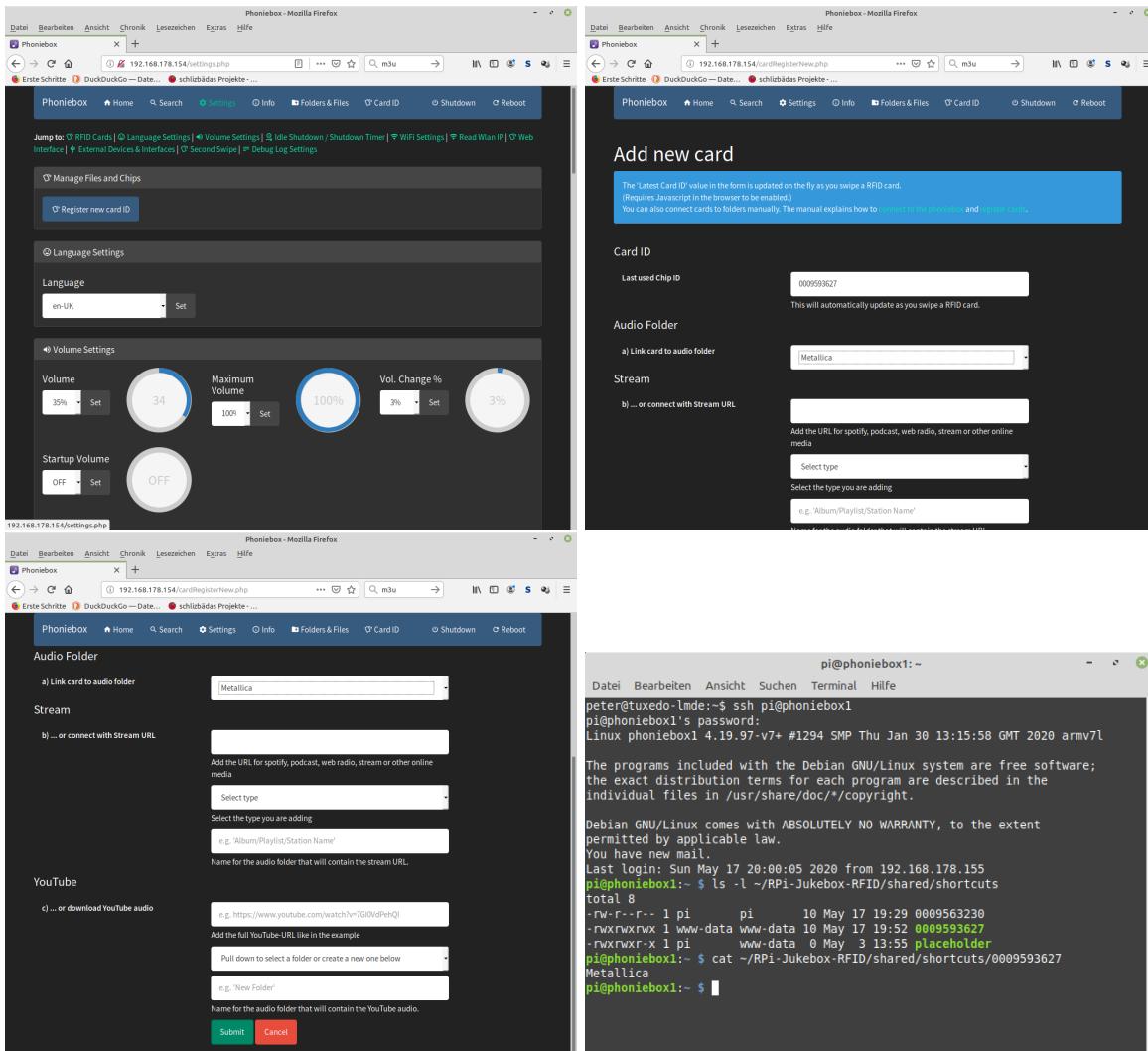


Abbildung 3.8: RFID-Karte über Webservice registrieren

Ordner „Metallica“ auswählen. Zum Speichern der gewählten Einstellungen im Browser nach unten scrollen und die grüne Schaltfläche [Submit] anklicken.

Eine Überprüfung im ssh-Terminal zeigt die Zuordnung der neuen RFID-Karte zum Album „Metallica“:

```
pi $ cat /RPi-Jukebox-RFID/shared/shortcuts/0009593627
Metallica
```

Auch hier bewirkt das erneute Auflegen dieser Karte den Start der Wiedergabe der Audiodateien.

3.9.3 Testen der Taster an den GPIO-Pins

In der Konfiguration von schlizbädas Phoniebox sind in der Datei `/home/pi/RPi-Jukebox-RFID/scripts/gpio-buttons.py` an folgenden GPIOs Taster definiert, siehe auch Kapitel 2.2.1 sowie Tabelle 2.2:

GPIO	Pin	Funktion	Beschreibung	Anmerkung
13	33	<i>vol0</i>	stumm schalten	(nicht verwendet)
12	32	<i>volU</i>	Lautstärke erhöhen	
6	31	<i>volD</i>	Lautstärke verringern	
7	26	<i>next</i>	nächster Titel	<i>fast forward</i>
8	24	<i>prev</i>	vorheriger Titel	<i>rewind</i>
5	29	<i>halt</i>	Play/Pause	

Tabelle 3.4: Taster an der GPIO-Leiste bei schlizbädas Phoniebox

Die Taster können bei dieser Gelegenheit auch gleich auf Funktion überprüft werden 😊

3.10 **systemd**: Autostart der Phoniebox-Software einrichten

Quelle:

<https://github.com/MiczFlor/RPi-Jukebox-RFID/wiki/CONFIGURE-stretch#auto-start-the-phoniebox>

Im folgenden Abschnitt wird die Phoniebox so eingerichtet, dass die gesamte Software beim Hochfahren gestartet wird. Dies geschieht über den Systemdienst **systemd**

Hinweis

Die in den folgenden Kommandos angegebenen Dateien wurden zwar für **Raspbian Stretch** vorgesehen, aber auch dort kam schon **systemd** zum Einsatz. Daher ist es unproblematisch, dass hier die **Stretch**-Dateien für **Raspbian Buster** verwendet werden. Wichtig ist aber, dass in Abschnitt 3.5.1 auch die entsprechenden Pakete für **Python3** installiert wurden!

```
pi $ sudo cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/phoniebox-rfid-reader.service.stretch-default.sample
/etc/systemd/system/phoniebox-rfid-reader.service
pi $ sudo cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/phoniebox-startup-sound.service.stretch-default.sample
/etc/systemd/system/phoniebox-startup-sound.service
pi $ sudo cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/phoniebox-gpio-buttons.service.stretch-default.sample
/etc/systemd/system/phoniebox-gpio-buttons.service
pi $ sudo cp /home/pi/RPi-Jukebox-RFID/misc/sampleconfigs/phoniebox-idle-watchdog.service.sample
/etc/systemd/system/phoniebox-idle-watchdog.service
```

Neue Dienste bekannt geben:

```
pi $ sudo systemctl daemon-reload
```

Dienste aktivieren:

```
pi $ sudo systemctl enable phoniebox-idle-watchdog
pi $ sudo systemctl enable phoniebox-rfid-reader
pi $ sudo systemctl enable phoniebox-startup-sound
pi $ sudo systemctl enable phoniebox-gpio-buttons
```

```
Created symlink /etc/systemd/system/multi-user.target.wants/<Name>.service -> /etc/systemd/system/<Name>.service
```

Dienste starten (anstelle eines Reboots):

```
pi $ sudo systemctl start phoniebox-idle-watchdog
pi $ sudo systemctl start phoniebox-rfid-reader
pi $ sudo systemctl start phoniebox-startup-sound
pi $ sudo systemctl start phoniebox-gpio-buttons
```

Status der Dienste anzeigen:

```
pi $ sudo systemctl status phoniebox-idle-watchdog
pi $ sudo systemctl status phoniebox-rfid-reader
```

3 Installation der Software

```
pi $ sudo systemctl status phoniebox-startup-sound
pi $ sudo systemctl status phoniebox-gpio-buttons
```



Hinweis

Sollten hier Fehler auftreten, so liegt die Ursache meist darin, dass nicht von allen Modulen die jeweils aktuelle Version heruntergeladen und installiert wurde. Gerade die Umstellung auf Python3 ist noch nicht in allen Teilbeschreibungen berücksichtigt und es wurden nur die Python2-Module installiert

```
pi $ sudo reboot
```

Jetzt sollten alle erforderlichen Softwarepakete auch im *Autostart* richtig hinterlegt sein. Beim Hochfahren ertönt ein Startupsound. Danach sollte sowohl der RFID-Leser auf aufgelegte RFID-Karten reagieren, als auch die Funktion aller GPIO-Taster gegeben sein. Die Original-Software der Phoniebox sollte weitestgehend richtig und sauber installiert sein. Es fehlt nur noch die aus einem anderen Repository stammende externe Software zur Ansteuerung des OLED-Displays.

3.11 OLED-Display einrichten

Die Phoniebox funktioniert zwar bereits vollständig, nur das optionale OLED-Display zeigt noch nichts an. Die Anzeige im Display wird über die I2C-Schnittstelle übertragen.

Quellen:

https://github.com/splitti/oled_phoniebox
https://splittscheid.de/selfmade-phoniebox/#5_3

Auch dieses Projekt kann über einen sogenannten *one line installer* installiert werden. Im Gegensatz zum Installer der Phoniebox-Software aus Kapitel ist der für die OLED-Software eher noobtauglich 🌱



Hinweis

Aus Gründen des Seitenlayouts wird der *one line installer* in seine Einzelbefehle zerlegt.

```
pi $ cd;
pi $ rm o4p_installer.sh;
pi $ wget https://raw.githubusercontent.com/splitti/oled_phoniebox/master/scripts/install/o4p_installer.sh;
pi $ chmod +x o4p_installer.sh;
pi $ ./o4p_installer.sh
```

Auch hier zeigt eine Bilderstrecke mehr als viele Worte:

3 Installation der Software



Abbildung 3.9: *splitti79s one line installer*

3.12 Bootzeit reduzieren

TODO! _____

Quelle:

[https://forum-raspberrypi.de/forum/thread/45581-dauer-des-bootvorgaenge-bei-eurer-phoniebox
?postID=414508#post414508](https://forum-raspberrypi.de/forum/thread/45581-dauer-des-bootvorgaenge-bei-eurer-phoniebox?postID=414508#post414508)

Maßnahmen
zur Re-
duzierung
der
Bootzeit
beschrei-
ben

3.13 Software-FAQ – oder Hinweise zu „beliebten“ Fehlern

In diesem Abschnitt werden nochmals stichpunktartig häufig gestellte Fragen samt ihrer Lösungsmöglichkeiten aufgezählt. Hinweise zur Erweiterung dieses Kapitels werden gerne per e-mail unter <mailto:himself@schlizbaeda.de> entgegengenommen, am liebsten gleich mit den dazugehörigen Antworten ☺

Speicherkapazität der SD-Karte

Eigentlich reichen 4GB zur Installation des Betriebssystems **Raspbian Lite**, aber den *mpd* darf man dann nicht mehr kompilieren müssen... (siehe Kapitel 3.4.2)

8 – 16GB: Diese Kapazität ist ausreichend für das Betriebssystem. Viele Erweiterungen können *problemlos* nachinstalliert werden. Zudem ist noch genügend Speicher frei für die Hörspiele der Kinder.

Hinweis

Wichtiger als die Speicherkapazität ist aber die Zugriffsgeschwindigkeit auf die Karte. Die sollte mindestens Class 10 betragen. Aber auch hier kommt es bei der gleichen Angabe zu großen messbaren Geschwindigkeitsunterschieden.
Der schlizbäda empfiehlt hier tatsächlich, zu den gängigen Markenprodukten zu greifen.

Raspbian Lite oder Raspbian Desktop?

blabla

Quelldateien von schlizbädas Phoniebox

../files

framps Raspbiback prüfen

todo!

testen!

Startupsound

Startupsound XOR mpd-Playlist!