

# DQMusicBox: Design


17 October 2017

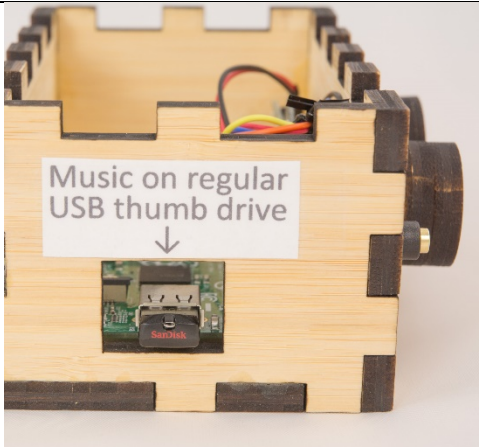
## 1 About this document

This document specifies the design for a dementia-friendly music player called DQMusicBox. It is an open source project. This document exists so other people can modify or improve on DQMusicBox. If you want to make a DQMusicBox from the existing design, you don't need this document – you can follow the existing build instructions (readme.pdf).

## 2 Discussion

Let's start where we left off in the requirements document – with some of the challenging use-cases. I have added a few key sentences about the design.

Use case	Challenge
<b>[End-user] Be familiar</b>	<p>The device should immediately seem familiar to the user. For instance, by resembling a vintage radio or a vintage phonograph (you may have better ideas than these).</p> <p><b>Design:</b> Vintage radio.</p> 
<b>[Caregiver] Update the set of music</b>	<p>The caregiver may be a non-technical elderly spouse. Thus, mucking with micro-SD cards may not be the best choice.</p> <p><b>Design:</b> An externally accessible normal USB thumb drive holds the music and only the music (the operating system and programs are stored elsewhere).</p>

	
<b>[System] Play key music formats</b>	<p>It's helpful for the caregiver if the system supports a variety of music formats e.g. MP3, AAC/iTunes, FLAC.</p> <p><b>Design:</b> <a href="#">VLC</a> (specifically VLC-NOX) handles the playback, so all of the above formats are supported.</p>
<b>[System] Create playlist</b>	<p>Create a playlist in some sensible order e.g. alphabetical by album/folder name.</p> <p><b>Design:</b> My code scans the USB thumb drive, orders found music by album/folder name, and then hands this off as a single playlist to VLC-NOX.</p>
<b>[System] Have good audio quality</b>	<p>The sound quality should at least be good. It doesn't have to be amazing, but it shouldn't be aggravating.</p> <p><b>Design:</b> The system uses a <a href="#">firmware update</a> from the Raspberry Pi Foundation that dramatically improves audio quality, details below.</p>
<b>[System] support sudden shutdown</b>	<p>The device should behave like a consumer electronics device e.g. you can unplug it with no issues.</p> <p><b>Design:</b> The system uses a read-only micro-SD card. More specifically, a <a href="#">TMP WRITE PROTECT</a> micro-SD card.</p>

### 3 Key hardware components

#### 3.1 List of hardware components

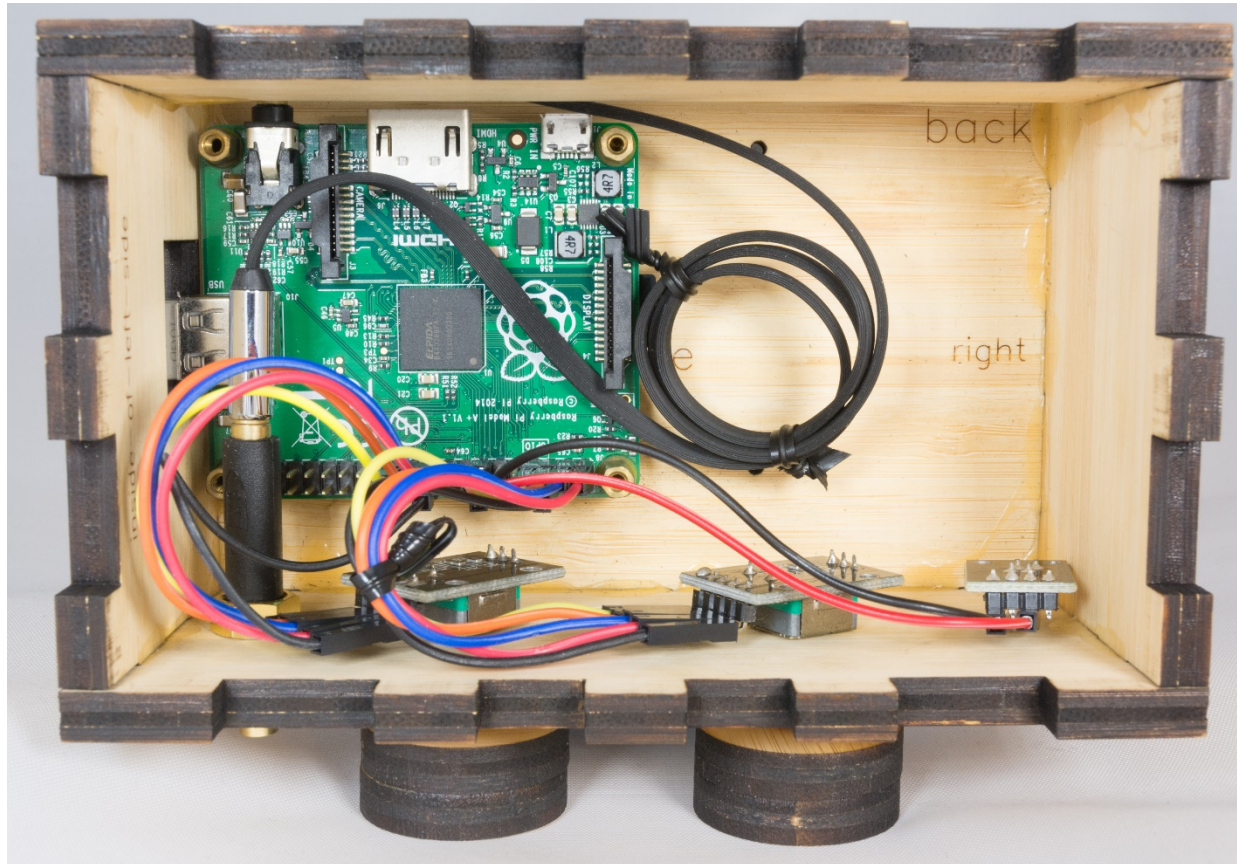
For a full list of hardware components, see the build instructions (readme.pdf). Below are a few of the key hardware components.

### 3.2 How did you choose key components?

When my wife asks me a question, sometimes I say, “It’s a long nerdy answer, do you really want to know?”. Then she usually leaves the room. Do you want to know the long answer? If so, read on. The choice was largely driven by satisfying the above challenging use-cases. And by cost. And I used the decision grid table below. The decision grid is 1.5 years out of date, but gives you a sense.

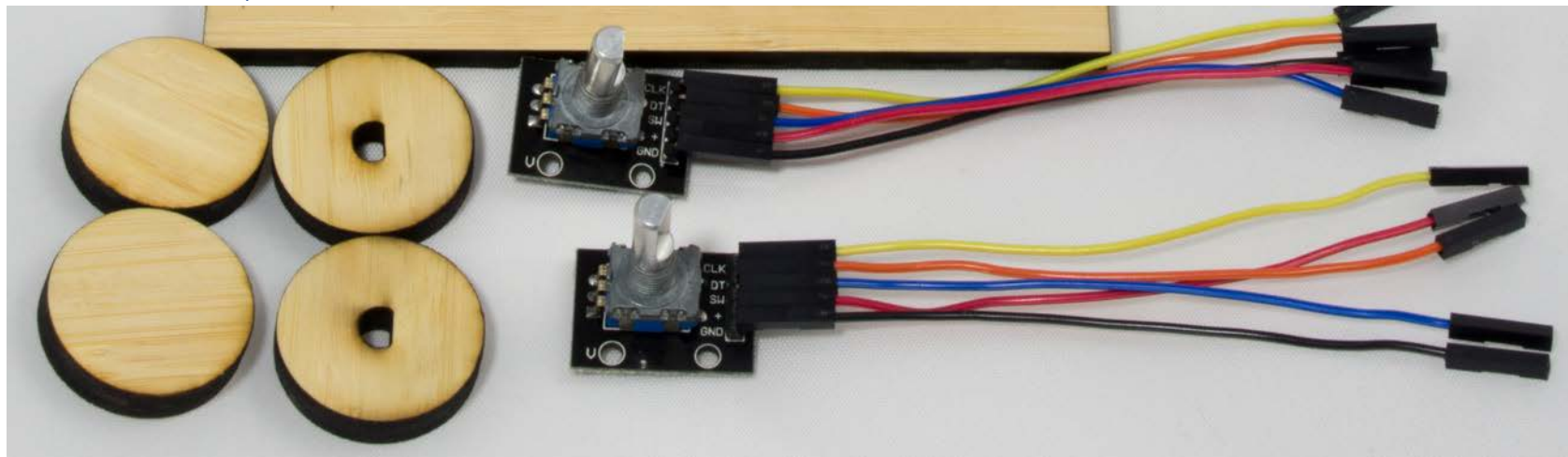
	Parts cost including case	Ease of loading music	Unpluggable?	Quick boot?	Software effort	Hardware design effort	Soldering	Support	audio quality	Total	Notes
Score range	3 = \$50 or less 2 = \$65 or less 1 = \$80 or less 0 = \$95 or less	2 = USB flash 1 = ext SD 0 = int SD	2 = safe 1 = mostly safe 0 = crashola	1 = instant 0 = <1 min	2 = no coding 1 = some coding 0 = all coding	2 = none 1 = some 0 = all	2 = little 1 = some 0 = lots	2 = great 1 = OK 0 = none			
Score weight	2	1	1	1	1	2	1	1			
<a href="#">Pi 2</a>	1	2	1	0	2	2	2	2	G	15	
<a href="#">Pi A+</a>	2	1	1	0	2	2	2	2	G	16	
<a href="#">Pi Zero</a>	3	1	1	0	2	2	0	2	G	16	
<a href="#">Banana Pi</a>	1	2	1	0	1	2	2	1	G	13	
<a href="#">Arduino + Adafruit MP3 shield</a>	1	1	2	1	0	2	0	2	G	12	MP3 shield is \$30.
<a href="#">Arduino + Sparkfun MP3 shield</a>	1	1	2	1	0	2	0	2	G	12	MP3 shield is \$35.
<a href="#">Arduino + Elechouse MP3 shield</a>	2	2	2	1	0	2	2	1	G	16	MP3 shield is \$20. Poor playlisting.
<a href="#">Sparkfun Lilypad</a>	0	0	2	1	0	2	1	2	G	10	Lilypad is \$50.
<a href="#">Custom standalone circuit</a>	3	2	2	1	2	0	1	1	G	15	
<a href="#">MP3 hardware player</a>											Can't find good one
<a href="#">Pine64</a>	3	2	1	0	1	2	2	1	?	16	Has headphone jack.
<a href="#">C.H.I.P.</a>	3	2	1	0	1	2	2	1	?	16	\$9. Has headphone jack.
<a href="#">Orange Pi PC</a>	2	2	1	0	1	2	2	0	N	14	OK built-in audio
<a href="#">Odroid C1+</a>											No headphone jack.

### 3.3 Raspberry Pi 1 Model A+



- **Why Pi?** A Pi satisfied the challenging use-cases above.
- **Why Pi A+?** Cheapest Pi model that has a headphone jack and populated USB headers.

### 3.4 KY-040 rotary encoder



- Why?
  - No soldering required! Just jumper wires.
  - Performs well.

### 3.5 Making the micro-SD card reliable -- TMP\_WRITE\_PROTECT

The micro-SD card is the weak link in Raspberry Pi based solutions. For instance, micro-CD cards can get corrupted when power is lost during a write. But what if we didn't need to write? That is the case here – the DQMusicBox doesn't need to change. DQMusicBox is a fixed function device – it doesn't have to evolve. So the micro-SD card can be read-only. Raspbian doesn't really want to be read-only. But I don't care if Raspbian is unhappy. I only care if it works.

The answer is the TMP\_WRITE\_PROTECT feature that is part of the SD card spec. It is fiendish. It accepts write requests (like Raspbian wants), but it puts all such writes in a special place. And those writes are thrown away when power is lost or when the device is rebooted. So no worries, if power is lost during a write. And if you want to factory reset a DQMusicBox? Just unplug it. References:

- [SD Card Write Protection](#)
- [Build the SD Locker and Make Your SD Cards More Secure](#)

I know of two methods to enable TMP\_WRITE\_PROTECTION:

1. Building the [SD locker project on Hackaday](#).

2. Follow Appendix 1 in this project's build instructions. The essence of it is to use a Pi 3, booted from USB, using the [SD Card Write Protection](#) too. This is the method that I used.

## 4 Software stack (component diagram)

I like to think of myself as intelligently lazy. And I think that I achieved that here. The project makes good use of existing code. In particular, VLC does a lot of the work. I wrote a shell script and ~300 lines of Python. Thanks to [Bob Rathbone](#) for his excellent [tutorial on rotary encoders and for his GPL'ed code](#).

dqmusicbox.sh (starts service)	dqmusicbox.py (finds music, setups VLC, handles knob events)
Bob Rathone's decoder (decodes rotary encoder (knob) signals)	
VLC-NOX (music playback and most things music related)	
DietPi ("extremely lightweight Debian")	
Firmware update for audio quality	

## 5 Use-case implementation

I've already covered the challenging use-cases. Here are the others.

### 5.1 End-user

Name	Description
Start song	Turn a knob, any knob
Change song	Turn the songs knob – clockwise for next, anti-clockwise for previous
Change volume	Turn the volume knob

## 5.2 Caregiver

Name	Description
Pause music	Tap either knob.
Update the set of music	Use a PC or Mac to add or remove albums/folders from the USB thumb drive.

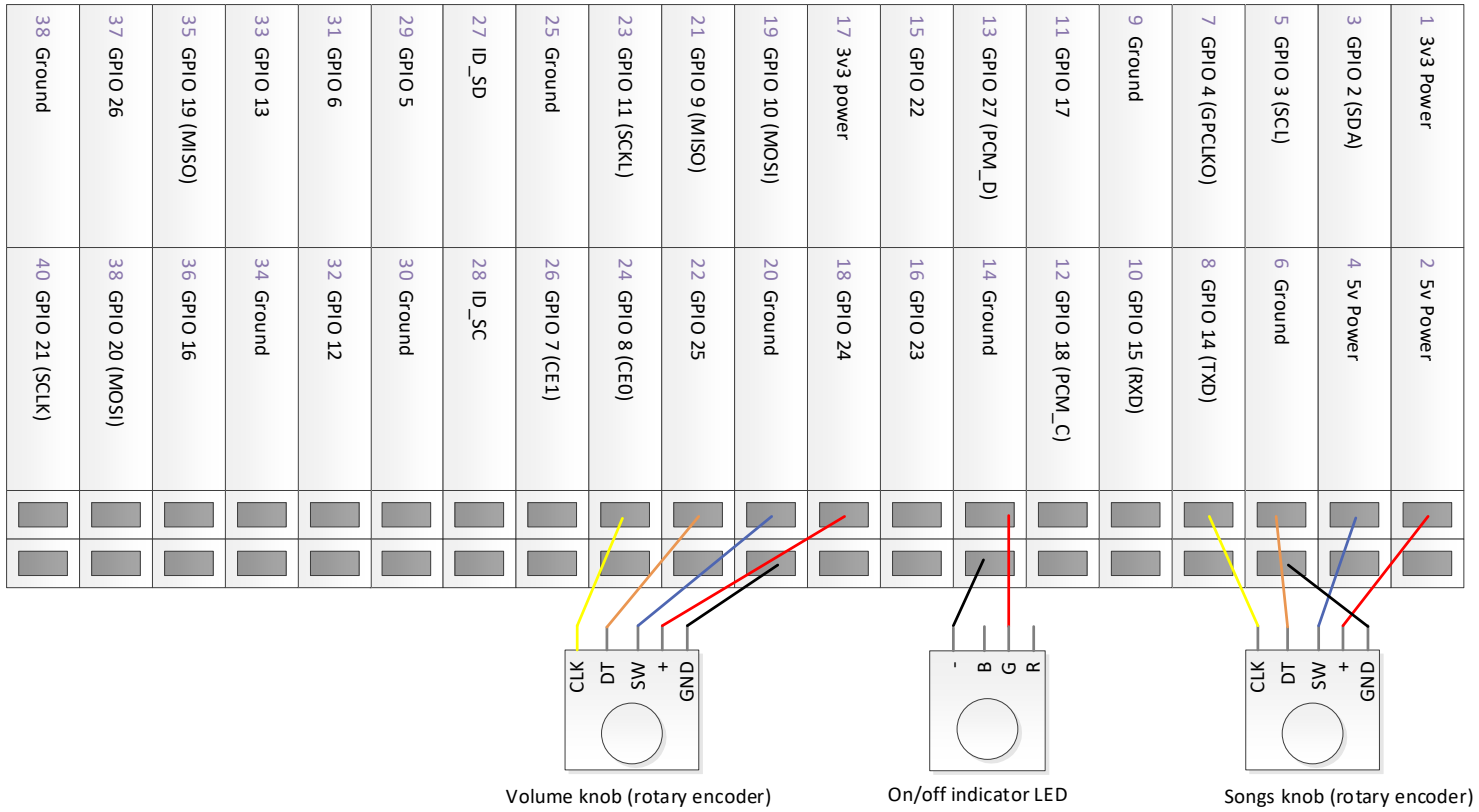
## 5.3 Builder

Name	Description
Support builders that are not woodworkers	We don't need no stinking jigsaws, we have lasers! The case is laser cut, pieces just need to be glued together. You can send the design to a company that does the laser cutting for you for \$28.
Support builders that are not good at soldering	No soldering required. The key was finding rotary encoders that don't require soldering.

## 5.4 System

Name	Description
Be secure	This is not an Internet of Things device. This is a thing. It never has to be online, it never has to be exposed to nasty things like hackers and reality television.

## 6 Wiring diagram



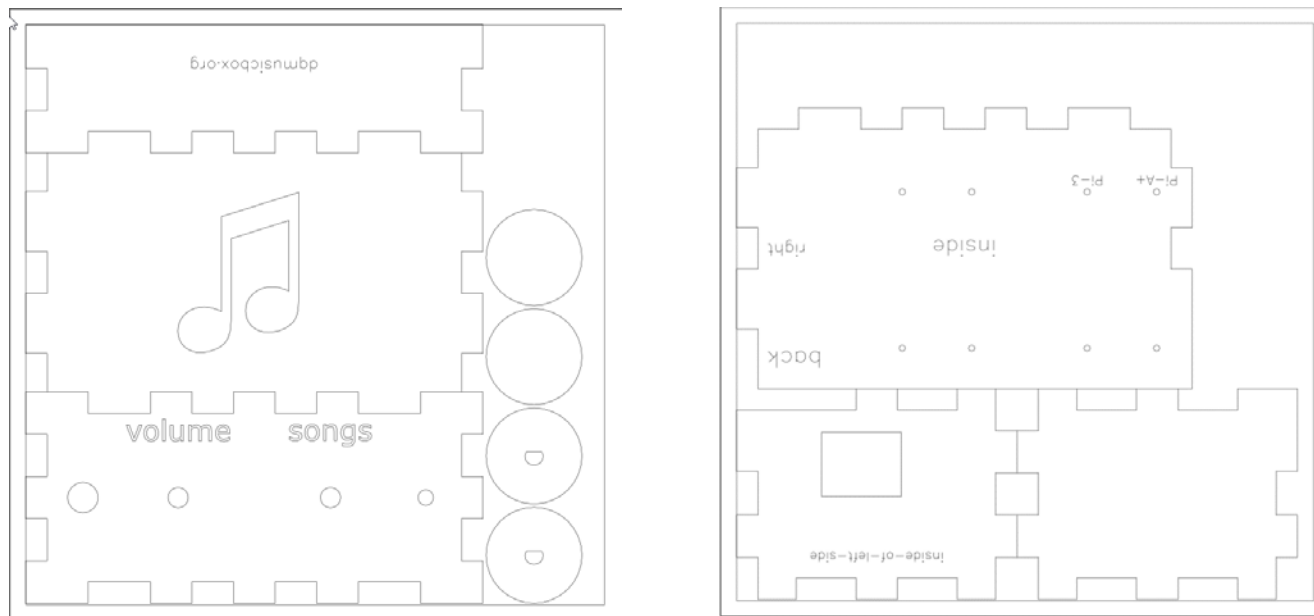


## 7 Case design

The requirements and key elements of the design for the case:

- Keep with the theme of vintage radio
- Use a laser-cutter approach, so no woodworking skills are required
- Make it possible to send off the design to a company (e.g. Ponoko) for laser cutting (not everyone has access to a laser cutter)
- No staining, painting, or other finishing required
- Include engraving (“volume”, “songs”, and the musical note)
- Include knobs
- Wood thicker than the screw threads of the standoffs (the screws that hold the Pi in place)
- Not too expensive

The result is a design intended to be laser-cut from two pieces of [181x181mm 6.7mm thick amber bamboo](#):



The [actual case design](#) can be found in [github](#).