1. SamplerBox2 features

1.1. RPI3 support

Works perfect on RSPI3, although there are some errors during startup and wifi does not work.

1.2. Accurate velocity (bug Fix)

In the original python script the sample is always played at the default velocity (if a sample is played in a velocity range it is always played that the velocity level of the sample, now the level is according the actual played velocity).
So suppose in the original version there are 4 velocity levels samples:

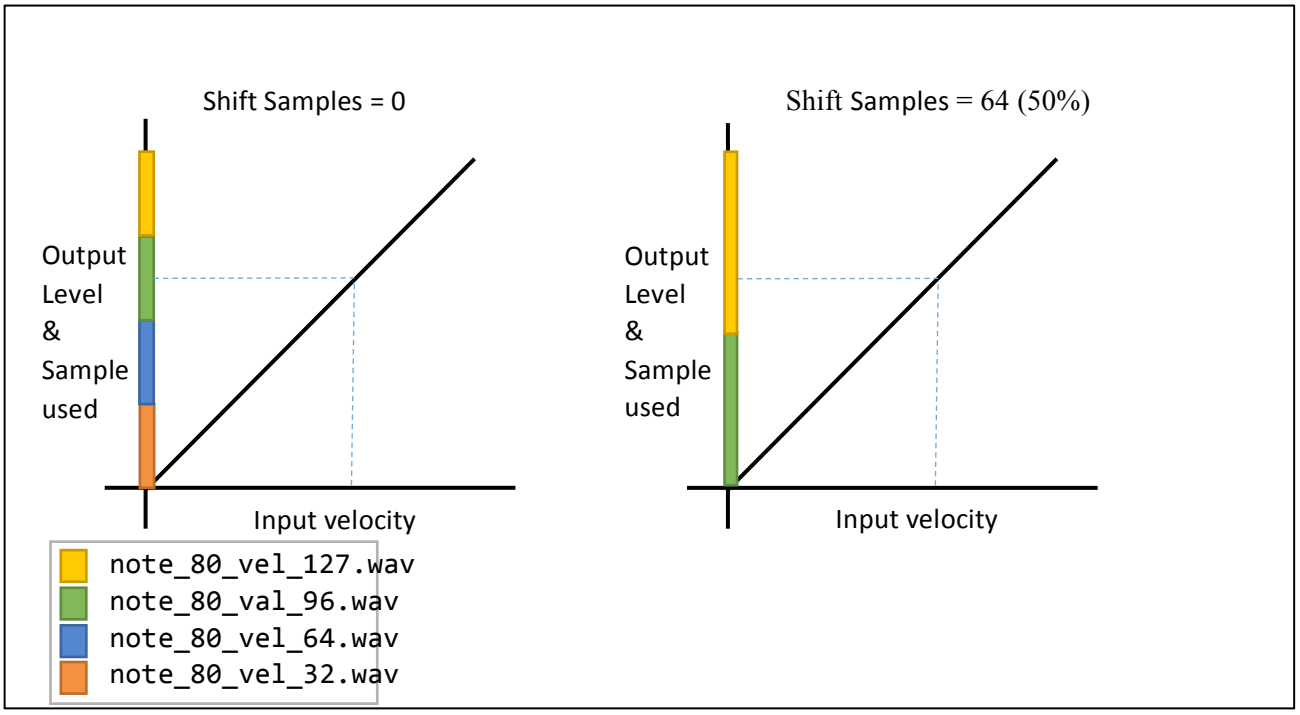
- note_80_vel_32.wav,
- note_80_vel_64.wav,
- note_80_vel_96.wav,
- note_80_vel_127.wav

In the original version when you play with velocity 1, the sample note_80_vel_32.wav is played at level 32. When you play 50, the note_80_vel_64.wav sample is played at level 64.

In my correction I assume all samples are normalized. At velocity 1 the sample note_80_vel_32.wav is played at level 1, Velocity 50 will play sample note_80_vel_64.wav at level 50

1.3. Shift samples

Added a parameter that can be used to shift the played samples in the multiple velocities, while maintaining the velocity level. Suppose there are 4 levels of velocity in a piano, adding this value to 64 (50%), the top 2 samples are played over the full velocity of 127 levels. As the top samples are typically brighter, this make the sound much brighter. For a piano this works great to make it more cutting through but keeping it dynamic. Lowering will make it more authentic (ballads, solo) while increasing will make it more suitable for a band context.



1.4. Freeverb

Full implementation of freeverb (in C++) is working, All parameters can be freely assigned to midi CC. I'm pretty critical to reverb, but I'm astound by the quality of the verb. Very useful!!! Full floating implementation, hardly any grain in the tail.

1.5. Added 2x2 channel playback of backing tracks

Using a 4 channel Audio interface, I can play a stereo backingtrack and a stereo clicktrack together with the sample. In the python code it is easy to use any midi parameter for selecting the backingtrack and any midi parameter to start/stop the backing track.
I use a Maya 44 USB+ audio interface (4x input, 4x output) For some reason the 44.1 KHz frequency is not stable (some clicks), after changing all samples to 48KHz it is super stable.
Location of backingtracks: ./samples
Naming convention:

```
1b Superstitious.wav
1c Superstitious.wav
2b I wish.wav
2c I wish.wav
```

- 1 is the number
- b is the backingtrack audio
- c is the backingtrack click

When CHANNELS is set to 2, the click-track is not played.
Only 48KHz wav files can be used, as 44.1 gives some clicks in my implementation. I aspect that it has to do with the used Audio Interface, but did not spend that much time in investigating that.
For 2 channel audio and other audio interfaces you can change the sample rate to 44.1KHz again.

1.6. Limited the polyphony for sustain pedal actions.

In the original code the notes where added for every note press of the same key when the sustain pedal is pressed, this creates too many notes.

1.7. Added tonecontrol (disabled)

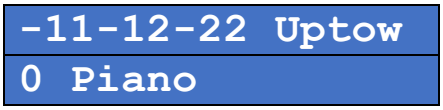
There is now an option to add all kinds of filters. This can be used to add a treble or bass boost. Although they work great, many parameters can be changed and it is a big CPU drain. When you assign filter parameters to sliders, this works, but adds clicks during modification.
For now I disabled this, but perhaps you need it, you can increase the buffer, lower the polyphony or remove freeverb to get some extra CPU cycles and enable this. The RPI4 might be much faster and this can be enabled.

2. Options

Option	Values	Description
CHANNELS	2,4	2: stereo interface. Select a audio interface with 2 channels the backingtrack audio will also play (the 1b name.wav files), but no click track 4: 4 channel interface, only tested with Maya 44 USB+ on 48 KHz
USE_FREEVERB	True, False	Default On
USE_TONECONTROL	True, False	Default Off
BUFFERSIZE	32,64,128,256	Default 128, 512 is around 5ms latency @ 48KHz 256 is around 2ms latency @ 48KHz
POLYPHONY	1-128	Default: 80

2.1. LCD display

I use a 2x16 char LCD display. It didn't work initially, but after updating the GPIO module for Python it started to work. For connections see, at the end of this document.



Top line:

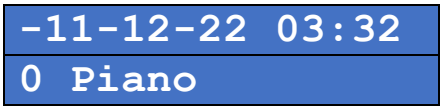
- volumes for Sample engine in dB
- Backing Track Audio Volume in dB
- Backing Track Click Volume in dB
- Backing track info:
    - Name when loaded and not running
    - Remaining Time when running
    - Percentage when loading

Bottom Line:

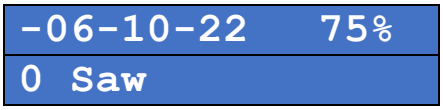
- Name of Sample Engine Preset
- When loading the percentage is shown on the right side

- When a parameter is being changed of messages are needed they are displayed on this line for 3 seconds, after that the Preset name is shown again

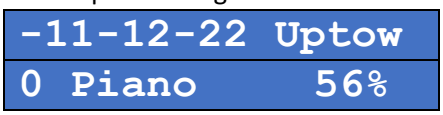
Backing Track Playing:



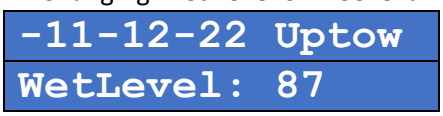
Backing Track Loading:



Sample Loading:



Changing Wet Level of Freeverb:



3. Create Image

3.1. Build image

Download SamplerBox image from SamplerBox website
Download Update Package from my website

Update Package contains:

File	From	Description
Fixup.dat	Pre-May Jessie version (May and later versions do not work, no clue why)	To update SamplerBox image
Startup.elf	See above	See above

Save to SD card (OSX):

- get disk name via disk utility -> info (could be something like disk2 or disk5)
- unmount using disk utility

Copy image to SD:

```
sudo dd bs=1m if=samplerbox_20150618.img of=/dev/rdisk2
```

Copy 2 files

- Open SD card on PC or Apple
- Copy **fixup.dat** and **start.elf** to SD

Change location of samplerbox.py:
Boot
Login with root, root

Make SD read/write:

```
Mount –o remount,rw /
```

in /Samplerbox there is samplerbox.sh

```
cd SamplerBox
nano samplerbox.sh
```

Change to:

```
sleep 5
cd /media/SamplerBox2
python samplerbox2.py
```

Set output level
Set alsamixer to 100 for Maya 44 USB+

- Alsamixer
- F6, select MAYA
- Arrow up to set to 100

Optional:
install scipy for Filters

download .deb file for armhf

- decorator: https://packages.debian.org/jessie/python-decorator
- scipy: https://packages.debian.org/jessie/python-scipy

5 filters: LP, HP do not have Gain, just use Peak filters with Q of 0.5-1
Or Low/High Shelving with 80Hz and 10-12 Khz

Install newer version of GPIO to get IO ports working (only if you need it, like for the LCD Panel):

- Get GPIO from **https://pypi.python.org/pypi/RPi.GPIO**
- Copy RPI.GPIO-0.6.2 folder to USB drive
- Go to directory on USB stick from Raspberry PI
- Mount as read/write
- Install:

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

3.2. Manually resizing the SD card on Raspberry Pi

You can also resize the partitions of the SD card that your Pi is running on.
First you need to change the partition table with fdisk. You need to remove the existing partition entries and then create a single new partition than takes the whole free space of the disk. This will only change the partition table, not the partitions data on disk. **The start of the new partition needs to be aligned with the old partition!**
Start fdisk:

```
fdisk /dev/mmcblk0
```

Then delete partitions with *d* and create a new with *n*. You can view the existing table with *p*.

- *p* to see the current start of the main partition
- *d*, *3* to delete the swap partition
- *d*, *2* to delete the main partition
- *n p 2* to create a new primary partition, next you need to enter the start of the old main partition and then the size (*enter* for complete SD card). The main partition on the Debian image from 2012-04-19 starts at 157696, but the start of your partition might be different. Check the *p* output!
- *w* write the new partition table

Now you need to reboot:

```
shutdown -r now
```

After the reboot you need to resize the filesystem on the partition. The `resize2fs` command will resize your filesystem to the new size from the changed partition table.

```
sudo resize2fs /dev/mmcblk0p2
```

This will take a few minutes, depending on the size and speed of your SD card.
When it is done, you can check the new size with:

```
df -h
```

4. Create USB stick

Download SamplerBox2.zip from my website
Copy full SamplerBox2 zip file to root of USB stick

5. General things to know:

Some things I learned and my take to modifications:

5.1. LOGIN

First of all: the login is: **root**
Password: **root**

5.2. Open read/write access (when image is ready, this is not needed anymore):

```
mount -o remount,rw /
```

5.3. Modify Python code

Login with root, root
kill running Python script:

```
top
```

Note down process number for Python: typical something like 258

```
kill 258
```

goto working directory

```
cd /media/SamplerBox2
```

Change the Script:

```
nano samplerbox2.py
```

quit and save with ctrl-x, y, enter

Test the script:

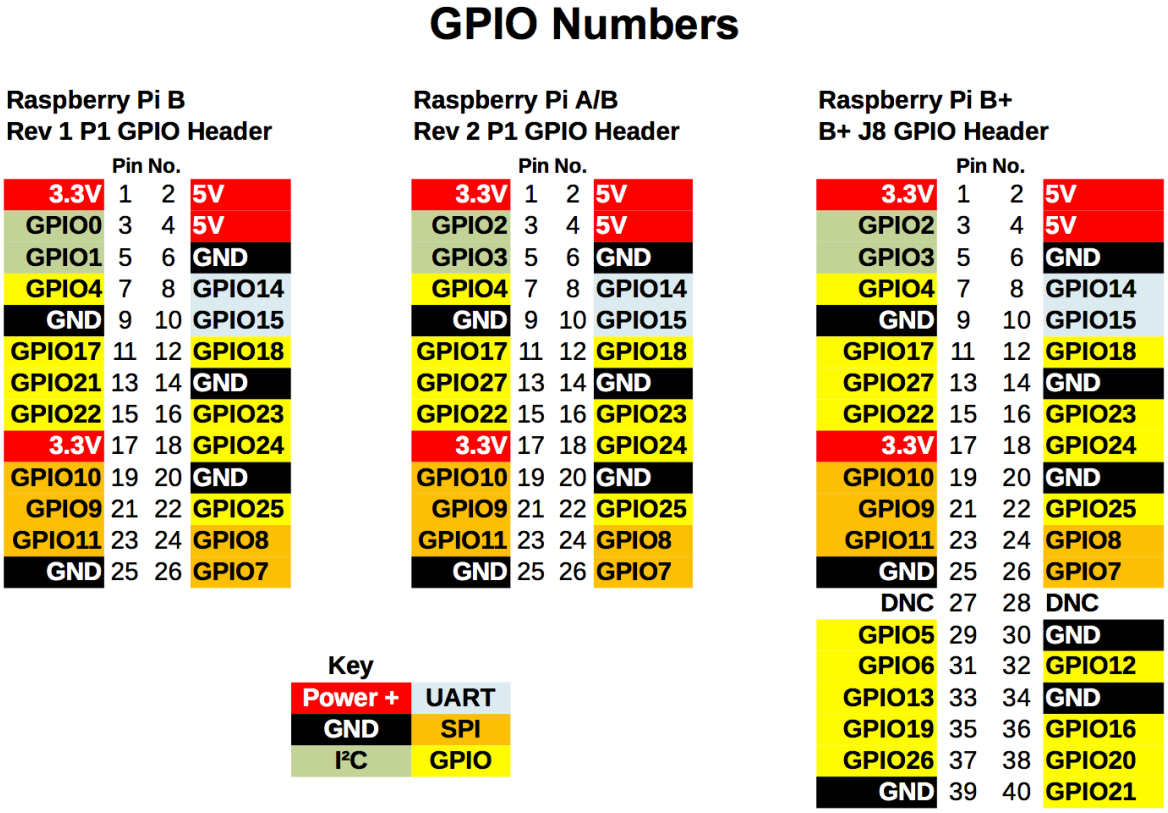
```
./run.sh
```

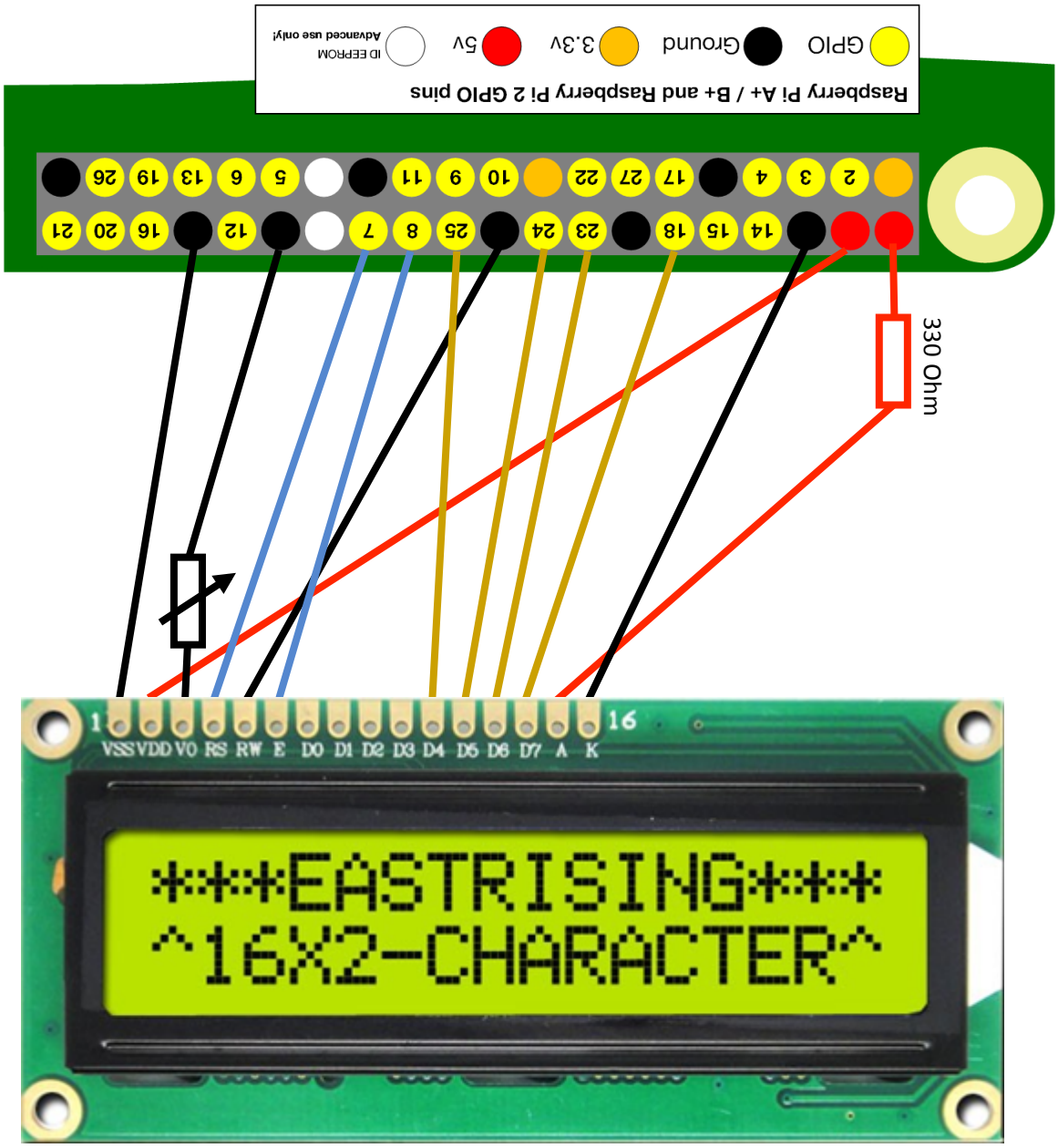
For final test:

```
reboot
```

This will make the SD card Read-only and will start the script in about 10 seconds

5.4. LCD Connections




Raspberry Pi A+ / B+ and Raspberry Pi 2 GPIO pins
GPIO
Ground
3.3v
5v
ID EEPROM
Advanced use only!
330 Ohm
VSS VDD VO RS RW E D0 D1 D2 D3 D4 D5 D6 D7 A K
EASTRISING
^16X2-CHARACTER^