RNBO – ein Primer

Pascal Zurek – mail@pascalzurek.de – pz019 – Stand 2023/10

Dieses Dokument dient als grundsätzlicher, sehr knapp gefasster erster Startpunkt für die Entwicklung von Projekten mit Max RNBO.

Zuerst wird auf die Installation und auf die Entwicklung von VST-Plugins für DAWs eingegangen, danach werden einige spezifischere Themen erläutert, um schließlich noch das Vorgehen mit RNBO auf Raspberry Pi-Computern zu zeigen. Das macht dieses Dokument **nicht** zu einer vollständigen Anleitung, aber zu einer Sammlung von Links und Informationen, die helfen können, nicht dieselben Fehler wie der Autor dieser Zeilen zu machen und schnell zu ersten spannenden Ergebnissen zu kommen. Die Reihenfolge ist bewusst so gewählt, dass sie die meisten Entwicklungen eines RNBO-Anfängers nachvollzieht. Ein Lesen des **gesamten** Dokuments **vor** dem ersten Öffnen eines RNBO-Patchers wird liebevoll empfohlen.

1. Grundsätzliche Ressourcen zu RNBO

- Dokumentation:
  https://rnbo.cycling74.com/
- Synth Building Blocks und RNBO Pedals
  https://docs.cycling74.com/max8/vignettes/rnbo_resources

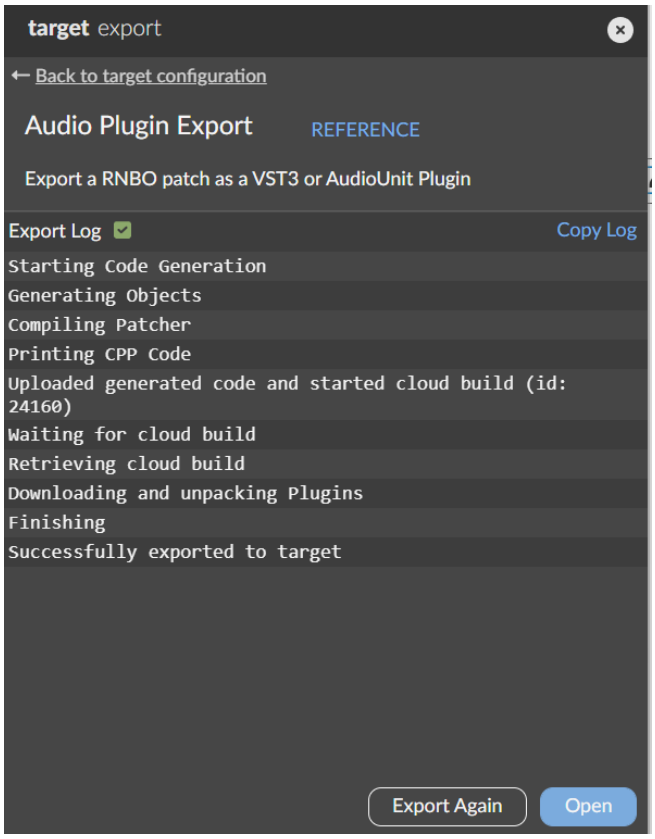
2. Wann sollte man RNBO benutzen und wann nicht?

- Ja:
  - Max-Patches als VST oder auf den RasPi oder auf eine Webpage exportieren (das kann nur RNBO)
  - Samplegenaues statt blockweises Arbeiten (kann allerdings auch Gen)
  - Gen auf anderen Geräten benutzen (*alle* Gen-Befehle klappen in RNBO!)
  - „unumständliche“ Polyphony
  - (Laut Manual soll Granularsynthese aus irgendwelchen Gründen besser funktionieren – konnte bislang nicht direkt bestätigt oder widerlegt werden)
- Nein:
  - während des Entwickelns eines großen Patches (Kompilierzeiten skalieren schlecht – lieber erst das Projekt in „normalem“ Max schreiben und dann auf RNBO umziehen)
  - bei Rapid Prototyping (denn der Kompiliervorgang braucht Zeit)
  - bei Bedarf für bpatcher – sollten die für das Projekt nötig sein, ist RNBO nicht das richtige, denn es gibt in RNBO keine
  - zum Ausprobieren von neuer, unbekannter Hardware, denn es gibt keine Debug-Ausgaben in VST oder auf dem RasPi, d. h. schon die CC-Ausgabe eine MIDI-Controllers kann nicht sichtbar gemacht werden. (Die Debug-Ausgabe von Max funktioniert natürlich)

- Wenn man VSTs programmieren will, aber keinen Internetzugang hat (kann ja vorkommen)
- Presets funktionieren zwar, aber nur über Snapshots

3. Installation

- In Max im Package Manager: RNBO und evtl. auch RNBO Guitar Pedals installieren. Das war's eigentlich!
- Falls für VST in DAWs gearbeitet wird: In der DAW ein VST-Verzeichnis definieren, in das man dann die Max RNBO-VST-Dateien exportiert.
- Installation eines Raspberry Pi: ggf. s. u.!
- Zum Funktionscheck ggf. einfach mal ein Beispiel laden (indem man ein rnbo~-Objekt erstellt, mit F1 die Hilfe aufruft und dann eins der Projekte lädt). Rechts im Inspector findet man das Kompiliersymbol. Die nächsten Schritte sind selbsterklärend. Nach erfolgreichem Kompilieren sollte man in etwa dieses Fenster sehen:



4. VST-Plugins mit RNBO generieren

Patch 0: Ein Delay als „Hallo Welt“

siehe Verzeichnis „0 – Hallo Welt“

Der Patch nimmt nur das (Mono-)Eingangssignal entgegen, spielt es unverzögert links und um delay_smp verzögert rechts ab. Man kann einerseits den Max-Patch laden und ihn als Blaupause für ein simpelstmögliches RNBO-Patch nehmen, andererseits das VST-Plugin direkt in eine DAW importieren und auf einer Spur ausprobieren.

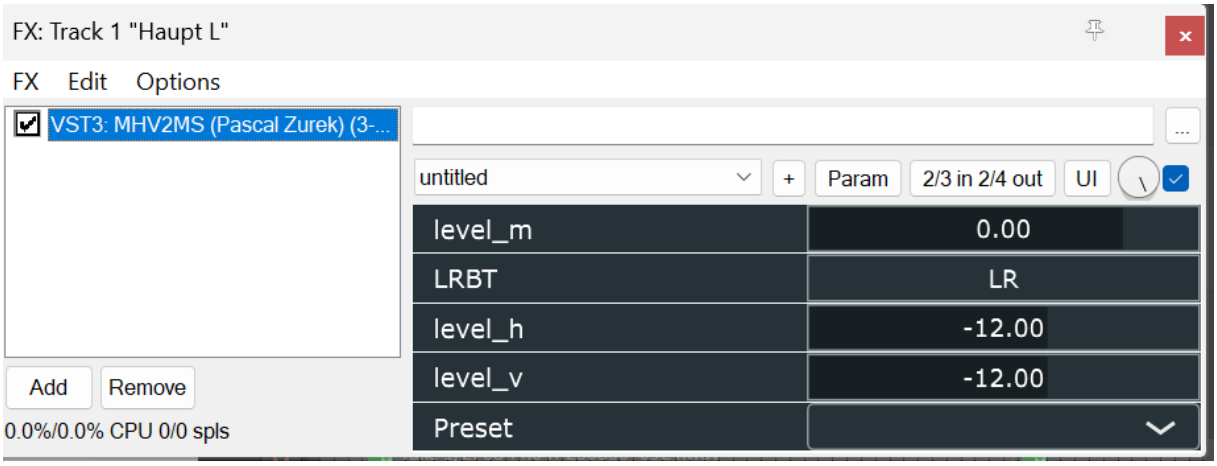
Patch 1: MHV zu Doppel-MS

siehe Verzeichnis „1 – MHV2MS“

Erläuterung:
Das Plugin „dekodiert“ ein Mitten- ein Höhen- und ein Vertikalsignal ähnlich, wie man MS in LR dekodieren kann. Dabei gilt:
Horizontal (wie MS):
$L = M + H$, $R = M - H$
Vertikal:
$B = M + V$, $T = M - V$
Im Patch ist (zum Testen) auswählbar, ob man auf Kanal 1+2 das L/R- oder das B/T-Signal haben will. (Auf Kanal 3/4 ist immer das B/T-Signal.) - Das VST nimmt dB als Werte entgegen, dazu den “enum”-Auswähler für LR/BT-Wahl.

4.1 UI/JUCE

- Die UI für RNBO-VST-Plugins sieht gewöhnungsbedürftig karg und sehr generisch aus. Tatsächlich entscheidet die DAW selbst, wie sie die UI-Elemente, die RNBO exportiert, darstellt; sie sehen also z. B. in Sequoia anders aus als in Reaper.



- Ist eine bessere UI für ein VST-Plugin gewünscht, muss man sie mit JUCE bauen. Das ist nicht ganz trivial. Infos gibt es hier:
  - https://github.com/Cycling74/rnbo.example.juce
  - https://github.com/Cycling74/rnbo.example.juce/blob/main/CUSTOM_UI.md
  - Fazit: Viel Voraussetzungen, um das bauen zu können. Das Tool ProJucer kann man sich sicherlich dennoch mal anschauen.

4.2 FFT in RNBO

- https://rnbo.cycling74.com/learn/using-the-fft
- Das Windowing ist vereinfacht; es gibt preset Window-Funktionen, man kann aber auch die alten Max-Patches per Copy/Paste verwenden – aber der default ist, dass kein Windowing angestellt ist! Man muss also prinzipiell weniger das Windowing von Hand machen, muss sich aber natürlich dennoch überlegen, was man da gerade tut, um nicht Artefakte zu generieren.

4.3 Messages und Ports

- Lernressource: https://rnbo.cycling74.com/learn/messages-and-ports
- Man muss sich erstmal dran gewöhnen, wie RNBO Nachrichten empfängt und sendet: Die in~ und inport~-Eingänge muss man sich genau angucken und schauen, welches Scope sie jeweils haben. Das ist eine neue Denkweise gegenüber normalen Patches.
- Wichtiger „paradigm change" hier auch: Für das Prototyping auf einem normalen Computer strickt man oft einen Max-Patch mit ezdac~, ezadc~ usw. um ein RNBO-Objekt. All das fliegt aber raus, wenn man das RNBO-Objekt dann kompiliert und an die entsprechende Stelle (z. B. auf den RasPi) bringt. Man muss also immer genau überlegen, welche Infrastruktur in den RNBO-Patcher soll und welche Infrastruktur außen herum gebraucht wird.
- Der Outport ist die einzige Debug-Möglichkeit. Man kann dann mit "rnbo.remote ipadresse" aus dem linken Inlet die Nachrichten empfangen. Wenn der Outport z. B. "eingangskanal" heißt, kann ich links an rnbo.remote ein "route eingangskanal" und ein "print" hängen und mir das auf meiner lokalen Konsole ausgeben lassen.
- Die Nachrichten, die ich über die Ports in RNBO hineinbringe, kann ich bearbeiten:
  - @fromnormalized expression => linear scaling
  - @max, @min, @enum sind auch offensichtlich wichtige Parameter
  - Für das RasPi-Webinterface ist noch @displayorder interessant (s. u.)

4.4 MIDI und OSC

- midiin und ctlin sind die Blöcke, die wir brauchen.
  - Wenn im Max-Patcher draußen herum kein midiin um den rnbo~ ist, funktioniert beim Prototyping das midiin im RNBO-Subpatcher auch nicht. **Man muss also in Max an zwei Stellen ein midiin einbauen!**
  - Dem midiin im Hauptpatcher gibt man von midiinfo (message: Controller) das richtige control device.
  - Folgerichtig erscheint dann im rnbo~ objekt ein zweites inlet für MIDI. Sonst nicht!
- Um z. B. einen RasPi zu steueren, ohne das Web-Interface zu benutzen, ist die einfachste Lösung, dass man einen Max-Patch zur Kontrolle baut mit OSC-Buttons:
  - Alle Parameter findet man über http://adresse_des_raspi:5678/rnbo/inst/0/ heraus
  - Dann baut man entsprechende Messages in Max (oder anderer OSC-Software) und sendet sie via "udpsend ipaddr 1234" an die OSC-Adresse. Das war's schon.

5. RNBO auf dem Raspberry Pi

5.1 Installation & erste Schritte

- RNBO läuft laut Cycling74 nur auf RasPi 3 und 4, nicht auf älteren!
- Externes Audio-Interface wird benötigt, z. B. das übliche Billig-Behringer UCA oder ein USB-Mikro, das class compatible ist. (Das interne Audio-Interface ist als Standard disabled.)
- Guter Überblick und Startpunkt:
  - https://rnbo.cycling74.com/learn/raspberry-pi-target-overview
- Bester Anfang: Raspberry Pi Imager downloaden und RNBO-Image auf den Pi ziehen
  - Mustergültiges Lernvideo: https://www.youtube.com/watch?v=oYBGYqhbRR4
  - Raspberry Pi Imager: https://www.raspberrypi.com/software/
  - Image für den RasPi: https://rnbo.cycling74.com/resources#raspberry-pi-images
- Das schönste Feature, direkt „out of the box“ nach der Installation, ist sicher das Web-Interface zum Einstellen der Parameter
  - Auch am Handy bedienbar! QR-Code mit URL ist direkt dabei (in Max).
  - Achtung, Fallstrick: Das Webinterface kriegt nur initial Daten vom Pi, sendet aber immer, wenn man etwas verstellt. Das hat die kuriose Auswirkung, dass man z. B. per MIDI oder OSC etwas auf dem Pi verstellen kann, im Webinterface sieht man es aber nicht.
- Grundlegendes Setup, falls noch nichts klappt: http://ip_address:5678/rnbo/inst/0 - dort sind die OSC-Endpoints, mit denen man alles einstellen kann. Siehe unten.
- Veraltete Doku: https://rnbo.cycling74.com/learn/configuring-audio-on-the-raspberry-pi
  - Da sollte eigentlich ein entsprecheder Audio-Config-Button sein, der kommt aber nicht mehr (Stand Max 8.5.x).
  - Das meiste ist stattdessen jetzt direkt rechts beim Export einstellbar (Samplerate, Audiointerface, Vektorgröße)
  - Das (nicht verlinkte) Script „rpi-configure-audio“ hilft auch sehr und liegt im Max-Ordner unter „resources/packages/“
- Einstellung per OSC
  - https://rnbo.cycling74.com/learn/configuring-audio-on-the-raspberry-pi#using-the-osc-interface
  - Man kann z. B. via OSC alles einstellen;
    - https://github.com/yoggy/sendosc ist ein nützlicher Helfer
    - Als Befehl dann:
      ```
      sendosc hostname 5678 /rnbo/jack/config/card s hw:0
      ```
  - Aus der Doku: „The RNBO Runner exposes an OSC-based interface to JACK, which you can use to select your active soundcard, choose a sample rate, and restart JACK. You can use this OSC interface directly, but it’s often easier to use the special audio configuration Max patch included with the RNBO package.“

5.2 Fallstricke und Wissenswertes

- Der Username muss "pi" bleiben!
  - Dazu: https://rnbo.cycling74.com/learn/working-with-the-raspberry-pi-target
- Falls nichts passiert: Per SSH und htop nachschauen, ob die Datei /usr/bin/rnbooscquery läuft
- Wo liegen die Dateien der Patches eigentlich?
  - In /home/pi/documents/rnbo. Den Ordner sollte man backuppen.
  - Nice to know: Das Kompilieren findet dann nativ auf dem RasPi statt, d. h. man sollte dafür sorgen, dass auf dem RasPi keine Last ist, wenn man schnell exporten will.

5.3 Potentiometer und andere GPIO-Dinge (Buttons etc.) einbauen

- Leider geht das nicht direkt in Max bzw. RNBO, aber noch immer sehr einfach: nämlich, indem man in Python ein Script schreibt, das die GPIOs ausliest und OSC-Nachrichten baut.
- Beispiel mit gpiozero und liblo (OSC-Sende-Package):
  https://www.youtube.com/watch?v=LVrVWhJBOL4
  Nötig dafür unter einer üblichen Ubuntu-Installation:
  "sudo apt install liblo-dev" und "pip install pyliblo3"

5.4 LCD-Displays und Encoder

- Nicht getestet. Brauchbare Anleitung aber hier:
  - https://www.youtube.com/watch?v=TZ-4zqEvpgo

6. RNBO als Plugin auf Webseiten

Das hier sind nur Links, um damit zu starten – ich habe selbst keine Web-Plugins gebaut.

- Grundsätzlich ist ein Webserver nötig, z. B. der in VisualStudioCode eingebaute oder ein lighttpd oder ein nginx… Was auch immer man zur Hand hat. Es reicht nicht, einfach die HTML-Seite doppelzuklicken ;-)
- HowTo: https://www.youtube.com/watch?v=Q3S3BD1LnCY
- Quelle für das „drumrum": https://github.com/Cycling74/rnbo.example.webpage

7. Common Mistakes/Stolpersteine/Unerwartetes/FAQ

Achtung: Das hier ist bewusst eine lose Sammlung an Stolpersteinen ohne Ordnung. Die meisten beziehen sich auf das Erstellen von VST-Plugins, viele sind aber allgemeiner Natur und tauchen auf, sobald man sich mit RNBO beschäftigt und Patches bauen möchte, die viele komplexere Max-Funktionen beinhalten – also auch, wenn man danach für Webpages oder Raspberry Pi arbeitet.

- Achtung bei kleinen Bildschirmen: Der RNBO-Export-Button ist ganz unten ganz rechts!
- Ohne Internetzugang ist RNBO schwer bis gar nicht zu gebrauchen; bei langsamem Internetzugang hängt Max regelmäßig. Das kann auch im Zug, bei Netzsperren in Hotels o.ä. problematisch sein.
- Beim VST bauen: Vor Recompile auf jeden Fall das VST in der DAW unloaden. Danach dran denken, dass man einen Neuscan des VST-Ordners machen muss. (Gibt es einen "einfacheren" VST-Host, damit man nicht immer eine "große" DAW manövrieren muss? Sequoia muss man z. B. manchmal komplett neu starten.)
- RNBO rechnet nur mit floats (intern)
- RNBO ist im Input blockbasiert. Die Befehle sind dennoch samplebasiert. Das hat Folgen: Ähnlich wie history~ in Gen ("gib mir das Sample n-1") kann man feedback~ in RNBO benutzen und kriegt ein um einen Block verschobenes Signal. Das muss man u. a. dann machen, wenn man eine Feedbackschleife bauen will. RNBO lässt das nicht zu; man muss den Output eben mit feedback~ um ein Sample verzögern.
- RNBO ist nicht Gen: Man kann Scope etc. benutzen, was in Gen nicht geht (weil Gen nur samplebasiert arbeitet)
- z.*-Objekte heißen hier list.*
- **Buffer**
    - Als Alternative zu buffer gibt es data-Objekte (64-bit Buffer). Das zieht einiges nach sich: - "groove~ @buffername buff1 buff2 buff3 @loop 1 @end 1500"
    - "waveform~" kann "buffer" darstellen, aber keine "data". Es gibt also keine "waveform"-Preview bei "data"-Objekten
    - Scope von Objekten kann modifiziert werden: "buffer local:foo 2048" - Buffer können auch urls als Source kriegen! Das ist wunderschön, weil man so Internet-Ressourcen laden kann (z. B. auf dem Pi headless Audiodateien abspielen)
    - Auch noch eine Menge anderer Änderungen in dem Buffer-Objekt. Weiteres: https://rnbo.cycling74.com/learn/using-buffers
- **Sample-Accurate Patching**
    - RNBO benutzt nicht den normalen Scheduler („wann werden Signale verarbeitet") von Max.
    - Man sollte also niemals ein RNBO in einem größeren Max-Patcher verwenden und dann z. B. ein uzi darauf anwenden.

    - Wann RNBO eine Message (also nicht: ein Signal!) verarbeitet, ist ungewiss. Wenn man will, dass etwas pünktlich passiert, sollte man Signale draus machen:
    - sig~ wandelt Zahlen in Signale um, damit kann ich wieder sample-akkurate Verarbeitung auch von Zahlenwerten nutzen.
    - Achtung, das frisst natürlich Ressourcen.
    - Weiteres: https://rnbo.cycling74.com/learn/intro-to-sample-accurate-patching
- Immer wieder: Man muss sich WIRKLICH daran gewöhnen, dass alles länger braucht.

8. FAQ

- Gibt es Text-Ausgabemöglichkeiten in die UI (z. B. in der VST)?
  - Nein. Es gibt praktisch keine Ausgabemöglichkeiten außer dem "outport", der eigentlich Ausgaben ins Webinterface schreiben soll, wenn man z. B. auf den RasPi exportiert. (Funktioniert aber dort nicht.)
  - Debuggen ist damit sehr, sehr schwer.
  - Der Outport ist die einzige Debug-Möglichkeit. Man kann sich dann lokal einen Max-Patch bauen, dort mit "rnbo.remote ipadresse" aus dem linken Inlet die Nachrichten empfangen.
  - Wenn der Outport z. B. "eingangskanal" heißt, kann ich links an rnbo.remote ein "route eingangskanal" und ein "print" hängen und mir das auf meiner lokalen Konsole ausgeben lassen.
- Wie debugge ich mein Plugin in der DAW?
  - Leider gar nicht: Erst das Plugin 100% fertig machen, dann in die DAW exportieren.
- Wie viel kann man ohne Internetzugang erreichen?
  - Beim VST-Programmieren: fast gar nichts. Schon beim Starten verbindet sich Max mit dem RNBO-Server, und wenn das nicht klappt, wird die Arbeit nichts.
  - Beim RasPi-Programmieren: Alles in Ordnung, denn der Code wird auf dem Pi kompiliert.
- Ist das nur bei mir so, oder ist Max auf einmal langsam und stürzt oft ab?
  - Du bist nicht alleine. Stand heute (2023-10) stockt Max beim Einsatz von RNBO erstaunlich häufig (zumindest unter Windows) oder crasht ganz trotz sauberer Windows-Installation und aktuellem Notebook. In der Erfahrung des Autors passiert das vor allem bei schlechtem Internetzugang: RNBO sendet ständig Kompilierungsanfragen. Man kann dieses Verhalten in der Fußzeile des Editors ausstellen und sollte das auch tun, solange man nicht nahe vor Vollendung des Patches ist.