Music Machine notes

Prototype to completed project:

ARM-Pi4+AudioHW based
ARM-PiCM4+IO-Board+AudioHW
ARM-PiCM4+Zynia(custom Board)
RISC-V(VisionFiveSBC)- [look up]-[find out] if the JH7110 audio hw is ‘exposed’ on the SBC
RISC-V(LM4 module)
RISC-V(jh7110) (maybe)
RISC-V+FPGA for Faust-Fast support

Common Features:
TouchScreen
encoders-switches
Provision for ‘music keyboard’ - both traditional and ‘isomorphic’.

Three ‘level’s (versions, models, etc) - NOT a time sequence
‘Accordion’ level-like organelle
‘Synth’ level
‘Studio’ level

Think of it as a General Purpose multiprocessor ‘utility’ platform, not just for music.

Faust fast risc-v
Also - ominous indications of ARM raising prices in anticipation of their IPO.

Our ‘roles’
Packager
Curator
Pioneer

Once the V5 was announced, the ‘combined single CPU test board’ seemed obsolete-[no point] - if we want to make it, just use that(V5) with any changes we want.

I THINK this is where the build-ci script is:
https://github.com/zynthian/ZynthianOS

I checked the Zynthian HW repo, 2023-05-13,no signs of V5 yet.
https://github.com/zynthian/zynthian-hw

A Possible first contribution to Zynthian which would become a pull request - adding A master or overall setting to the repositories screen in Zynthian web conf so in addition to changing the individual settings you could with one action change all of them - I think this would be useful because most people want to change them all in sync anyway.
Part of it is fixing what I would consider a ‘defect’ - if you try to change the individual sections with the “stable” box from ‘non-advanced’ view unchecked it happily lets you change them, then loses your selections later.

Another defect that is potentially confusing and much time-wasting and hopefully ‘pretty easily fixed’ is the “Zynthian needs to reboot to apply your changes!” in the red bar coming up way too early when you change repos.

WebPD
https://sebpiq.github.io/WebPd_website

‘New’ modularity - 3 above ‘keyboard bay’
A reset button on the outside of case.

Ssh zynthian.local
Home pi zynthian plugins lv2
zynthian/zynthian-ui/zyngine

To get to the ‘NOVNC’ functionality
http://zynthian.local:6081/vnc.html

Features of Zynia1 - ‘a better Zynthian’

Hardware UI:
4 x 4 of ‘existing buttons’ that handle the ‘long and intermediate press’ issue
New ‘macro’ buttons
Macro controls
Built in speaker/amp
7inch? Touchscreen
Tape-Recorder Transport controls
Bang! buttons
Vial - a qmk fork for more 'dynamic'-[run time] keyboard definition - includes midi support - ‘basic’ and ‘advanced’.
the link below seems to just be a regular web page, despite the slightly scary 'get' prefix:
https://get.vial.today/

Pure data as a plugin lv2 - allow multiple
Name in mixer
Dynamic patch? -may already exist
Lib of pd patches and automatically wrapping it in Camomile
Multi-core multi-processor support - just multiple patches - minimal communication

Rnbo - ?

Path - Sequence - Phase Progression

Prototype with existing hardware
RPi cluster board
Custom Board - should we do it with RPi or RiscV? Single or Multi-CPU?
Ultimate - when USA Fabs are built and a multi-Project wafer is available for ‘ordinary users’ - custom chip.

Equivalence of DAW-Ardour-etc and Zynthian-Zynthbox-Zynia-etc as ‘environment’-[operating system] for plugins.

Architecture to allow splitting to take advantage of parallelism-[additional CPUs-cores].

Heat management is an important aspect.

Possible approaches-paths:
Hardware:
None - just use Zynthian and focus on Software
A Zynthian clone - Possible changes - Easier assembly, Better availability in the US, a musical keyboard, larger-better screen, more controls, CM4
Two CPUs - Synth and DAW
Multiple CPUs >2 - Control - Generator(s) - DAW
One Variation on Multiple CPUs - a Commercial ‘cluster’ board ie: Turing Pi 2 Cluster Computer (currently on kickstarter 5/31/2022)
Note - it is NOT limited to RPi ‘architecture’ - we could even make a ‘Music Processor’ board for it with an array of Risc V processors.
With a [big enough]-right heat sink, we might avoid a fan.

Software:
Separate out DAW from Synth - or just go with Zynthian combined function and further refine it.
Improve sequencer - allow real time and step
Allow editing of the virtual-jack connection (does the current Zynthian really not allow you to change or delete and reassign a layer?!)
Come up with a better organizing principle than midi channel for layers

Improve PureData - ‘more’ integration, a LV2 plugin?

A ‘use-case’ for Zynia:
A portable musical instruments that can be played in a live environment without requiring large screen or other "non-musical" aspects. For sound design a screen can be attached.
Another use case is ‘one man band’ - Drew pretty much said this was what he wanted.

TO do

Install Ardour on a RPi under PatchBox or ZynthOS.

Some notes on dacs-adcs-codecs ‘of interest’ and to be considered from a ‘footprint’ perspective.
Daisy Seed rev5 (with the WM8731 codec)
AKM CoDec AK4556
Zynthian TI
Third alternative supported by libDaisy is PCM3060PW that is used by Patch SM, it also needs I2C.
PCM3060, which is used for Patch SM and future Daisy hardware.
Full high definition 24-bit 192kHz Texas Instruments TAS5756M digital audio codec (DAC) - used in RPi ‘official’
Cirrus Logic CS4270-CZZ - another option - they have at JLCPCB! - at least it's listed.

Xvnc browser

Even with the plethora of musical gadgets and software that is available due to the magnificence of capitalism there are ‘deficiencies’ that persist. there are gaps in product lines and a lack of openness, interoperability, expandability, price issues, Etc

The goal(s) of the project:

To create an ‘architecture’ for experimenting-doing-playing Computer Music-Visual production. The architecture should be flexible enough to allow a range of uses from a portable music synth or gadget at the low end to a ‘flagship’ synth at the high. More details below, first we’ll take a little detour for ideas and prototypes and plans.

Some key ideas of the ‘Music Machine’ are:

Open Source - both Hardware and Software
Use of ‘small’ computers and combining them to get the processing power needed. Particularly at the ‘layer’ which ‘computes’ audio samples. This layer, ‘core’ is made out of ‘compute elements’

Currently we have ‘prototypes’ for experimentation using three Raspberry Pi, (GUI, core, DAC). If more power is needed, add more core CPUs. The core processors can be replaced by more tailored to use ‘compute elements’. Something like a Daisy Seed, but having the necessary ports (Ethernet? I2S?). And of course, aspirationally-eventually, all Arm processors would ideally be replaced by Risc V processors to align with the open source nature of the project and allow multiple compute elements on a chip and easy customization at the chip-IC level.

A next prototype would bring the three processors together more neatly, perhaps via a PC Board.

Back to goals:

To create a ‘flagship’ ‘open’ ‘affordable’ synthesizer.

‘flagship’ meaning high end, powerful - eg Yamaha Montage (and its ‘affordable’ sibling MoDX), Roland Fantom, Korg Kronos etc. This doesn’t have to restrict the result to ‘large’-’expensive’-’heavy’ products. In the same way that the DIY Korg NTS-1 utilizes the technology of the ‘logue’ series synths, a more portable Music Machine might have only 1, or 2, or 8 processors, tradeoff complexity for reduced polyphony. And it doesn’t have to ‘look like’ or ‘be like’ traditional flagship synths in terms of form factor, ‘availability of the Operating System’ etc. ‘availability of the Operating System’ -means the user-musician or musician-user or muser can get to the OS if it will help.

‘open’ - Openness - of source code, hardware design, processor architecture, etc. Openness to interoperability. to supporting new devices and old. Ironically Montage is based on open source software, yet it appears to the musician-user as a mostly closed system.

‘affordable’ - short answer <$1000 - ‘Reasonableness’ of price - there's no need to make the music machine ‘cheap’ and in quality versus cost issues quality wins, but there's also no need to have a high price-huge profit which I think is a problem with Flagship synthesizers such as Montage and Fantom. Perhaps even modx the ‘affordable’ version of Montage is not affordable in our frame of reference. Or at least it’s not a ‘casual’-easy purchase - you have to think about it. The lowest price MoDX is $1400 US.

Another goal, which is related and to some degree solved by the above but might involve simpler-(more specific) hardware is to ‘play’ or ‘auralize’ various ‘natural’ data meaning DNA, ‘atomic’-’electron’ structure etc.

An important point-’piece of the puzzle’ is the ‘abstraction’ or ‘abstraction layer’ which the ‘musician’ or ‘electronic-computer musician’ uses to specify their ‘score’-’musical desires-intentions’. This will largely use what Miller Puckett called ‘the Max Paradigm’ and we will initially use Pure Data as the Graphical-Mathematical language.

An interim-[short term] goal would be a ‘technology-architecture’ demonstrator-prototype-breadboard-etc. This would be a board with holes for mounting and ‘areas’ for Seed(s), an Arduino for ‘control’-HMI-etc, and a Pi for ‘development-[sound design]. Each ‘area’ is positioned so its ports are accessible externally as needed and connected ‘in-the-box’ or ‘on the board’.. It would run software in the Max paradigm on the ‘workload’ split into HMI, Sound Design, Score-Event Processing, Amplification (I know that sounds odd-analog but we stay digital including the power amp(s) - and it is important to allow ‘feedback’ from that layer. I’m [starting to think]-considering that DaisyDuino eliminates the need for a separate Arduino board.

An even shorter term goal, which has already been done (May 2021) by Frank is to prototype the above by just hooking up 3 Raspberry Pi via Ethernet.

A control panel, including watchdog timer indicator, perhaps even display the digits, or just ‘log’ it.

Just as early IC Designers Had to unlearn the design wisdom of their time (to minimize the number of active components) we have to unlearn the wisdom of minimizing the number of processors. hopefully this will get less ugly when those processors are all a common open source architecture IE risc V. But for now the splitting of the workload means at least 2, perhaps 3 different ‘boardtypes’ The ‘perhaps 3’ is whether Arduino really gets you easier DIY interfaces.

The different board types need to be connected in 1 or 2 ‘ways’ depending on their function. A ‘control link’ handles control info, which is some fraction (perhaps 10% or even 1%) of the ‘sample’-’audio’ rate, which is of course the actual audio samples. The control link uses MIDI 2, which is NOT the same as MIDI 1 or just MIDI. The sample link uses raw ‘pcm’ data. It is possible that an Ethernet based link could ‘unify’-’combine’ the 2 link types and add a ‘broadcast’ capability.

Other possible features-differences for Leaf-[BoardX-PodX] would be:
Ethernet Connection - Internally between pi and leaf, future growth to 2 Pis and a power of 2 Leafs.
Designed for and-or includes a case
Access to all processor pins with a ‘doable’ effort.
Breakout all processor-interesting pins to Pads-Holes-[test points]
Breakable links to a shadow Seed
LQFP instead of BGA package
Mounting space for Pi by headers
A decent amp and speaker(s)
Feedback from class D amp
Full Size (¼ inch) audio jacks and Midi connectors
Some measures to reduce noise - allow or place a resistor between agnd and dgnd
Provide a jtag connector outside the case

Provide a reset button
Include stlink on board like some of st's development boards
Communication via USB, EtherNet, I2S

Dear reader - here's more on each of the above 'features'-[design decisions] - pros and cons - etc
Ethernet Connection - Internally between pi and leaf, future growth to 2 Pis and a power of 2 Leafs.
Designed for and-or-maybe includes a case - or just the 3D print file is available - pots and switches accessible and easily mounted..
Access to all processor pins with a 'doable' effort. A 'pad' available for each pin-of-interest
Breakout all processor-interesting pins to Pads-Holes-[test points]
Breakable links to a shadow Seed
LQFP instead of BGA package - This one's interesting, and it's not necessarily a 'feature', more a way of achieving some of the above goals. And also making the board 'manufacturable' in small quantities.
Correct the 'issue' with USB connector(s)-Midi
Mounting space for Pi by headers
A decent amp and speaker(s)
Feedback from class D amp
Full Size (¼ inch) audio jacks and Midi connectors
Some measures to reduce noise - allow or place a resistor between agnd and dgnd - see the thread: Daisy Field and 4051 multiplexer crosstalk? In the forum.
Provide a jtag connector outside the case
Provide a reset button
Include stlink on board like some of st's development boards
Communication via USB, EtherNet, I2S

.
The equivalent to the Daisy Processor STM32H750IBK6 in a UFBGA package is STM32H750IBT6 which is a 176 pin LQFP. We might be able to use a smaller LQFP but we'd lose some pins. However, I checked on availability - STM32H750IBT6 176 pin LQFP is out of stock at distributors and ST. STM32H750IBK6 UFBGA is available at DigiKey - 973 in stock. STM32H750VBT6, 100pin LQFP is also in stock at DK, 170 in stock, and two European distributors.

The 'host' and 'mixer' functions could be Raspberry Pi, Beagle Boards, ST's 'microprocessor' class machine (2 fast cores, one slow), and even a PC for the initial implementation.

Terminology

Root processor - this is what's responsible for interpreting incoming midi signals and distributing them and the ‘score’ to Leaf processors. We will also assume that it's responsible for summing the final audio signal before it goes out to an amplifier and speakers. At some point this might become a separate ‘distribution’ or ‘summing’ processor.

Leaf Processor - this is what's responsible for turning the notes of the score-performance into audio samples – It's the ‘guts’ of the synthesizer. For at least some versions-concepts of the music machine it's where LibPD runs. There's a question of whether it would be a stripped down version of LibPD or the full version.

Choices: we could use one processor for everything however I've read about issues with the organelle having trouble maintaining The screen and user interface and the music generation. we could either not have the user interface code active while the music generation is happening or we can have two processors for that purpose, although it seems like as we generate a lot of notes or do cpu-intensive synthesis that could create a bottleneck. that brings us to the scheme we've been considering which is a ‘root’ processor to handle the user interface and dispatching notes from the score to a set of leaf processors that do the actual sound generation. that still leaves a question of how the leaf processors output is combined, for now I'm thinking the root processor can handle that too.

In a sense the project could be looked at as an extension or a layer for the daisy platform and architecture.With the possible replacement of official Daisy boards with our own processor boards probably arm-ST but maybe risc V.

Frank's comment about the leaf root terminology implying a certain structure is a good one and brings up several points-questions about the way the nodes will communicate. There are several conceptual ‘links’ necessary which might or might not be physically combined. the conceptual links: score distribution link could be implemented as a one-to-many broadcast path either through a bus or a daisy chain or a protocol like ethernet. A digital audio or sample link which also could be broadcast or specific to a processor in this case there might be performance reasons why you would want to make it separate so that when you have a large number or I should say a handful to a large number of processors perhaps 3 or 6 for a very small polyphonic synthesizer and hundreds for what you would call an orchestra. There is a standard for a commercially available audio oriented version of ethernet I'm not sure who defines it if there is some sort of non-proprietary aspect to the definition of a standard basically I'll check on that.

Do list

Figure out what libPD requires to run and what it means to run a pure data patch without the pure data graphical environment in other words identify in the pure data code what remains of the idea for the past implementation of a separate DSP processor. Or perhaps the Approach

that Electro Smith seems to be taking to compile the patch directly into C plus plus code to make a binary module to run on the daisy would make more sense.

Gather relevant documentation - literature - propaganda from manufacturers into a folder things like ST-AKM Doc on the chip(s) in Daisy pure data documentation Etc

Look at the Daisy Seed to see what is available for ‘inter-node’ communication - particularly is Ethernet available - I know it’s on the ST chip, but don’t know if they brought the pins out.

Pure Data notes

About two thirds towards the bottom of the changelog are notes presumably from Miller Puckett on conventions and the way things are set up in the source code ‘Source style and organization" as he puts it

There’s an ‘experimental version’ Version of PD available on Miller's software page that says it's been ported to Max I don't understand what that means as a concept it's kind of like if you said I have an experimental version of VisiCalc that's been ported to Lotus 1-2-3. I looked just binarymsg - hmmm, interesting.

Guide to writing external “objects” in C that can be loaded into PD. Helpful to understanding some of the internals of PD. https://github.com/pure-data/externals-howto

Pure data patch file format: https://puredata.info/docs/developer/PdFileFormat

Pure Data: Recent Progress - Miller Puckettehttp://msp.ucsd.edu › Publications › icmf97
This ‘is’ a paper which, despite the ‘recent’ in the title ‘is’ from 1997! Despite that it was very interesting and to me at least provided insight on Gem, and netsend-receive.
It is a PostScript file rather than PDF, which make the behavior of the Google result link very strange for my Chrome browser if I just clicked on it or tried to ‘open in new tab’. I could download and open with qpdfview.

Libpd:
https://wwws.uni-weimar.de/kunst-und-gestaltung/wiki/images/Embedding_Pure_Data_with_libpd.pdf
https://github.com/libpd/libpd

Interesting examples of pd patches:
https://puredata.info/community/member-downloads/patches

This is the Pure Data chapter of a commercial book ‘Designing Sound’ by Andy Farnell:
http://aspress.co.uk/ds/pdf/pd_intro.pdf
It’s a good intro to Pure Data

CSound notes

Would there by a ‘good’ way of providing the ‘timing’ capabilities of CSound in the ‘Max paradigm’-Pure Data environment?

The Canonical Csound Reference Manual - cSounds.comwww.csounds.com › manual › html
Includes A list of opcodes.

Blog
There is another article written by Miller Puckette http://msp.ucsd.edu/Publications/dartmouth-reprint.dir/. Called “Max at 17“. It even talks about the “note card”. He doesn't include music n when she is listing influences I think because it had so much influence he couldn't even see The possibility of not understanding that.

It leads to a series of letters about Max and flaws in Max from computer music Journal (http://www.computermusicjournal.org/cmjlib/editors-notes/17-2.html) which also sheds light on the big questions that we are sometimes struggling with mostly trying to avoid.

The concept that I was talking about in our last call I'm struggling with is that CSound in a very direct sense is written in itself the CSound compiler written in CSound in much the same sense that the C compiler is written in C. The ‘tigoto’ Is a good example if you look at the key tigoto CSound file it's defining the parameters and how they work in CSound.

Other related-adjacent systems:

This section will contain references and pointers to other music or audio systems and how they relate to music machine.

Axoloti - is a small open source system that's very close to what we want to do the only-some thing(s) that's missing is the distributed multiprocessor aspect it might be a good source for information on components and where to get them such as the right angle midi connectors Etc It uses the STM32F427. There is a comment on a YT video https://www.youtube.com/watch?v=so66RHSh4V4 ‘new Axoloti (Akso by ZRNA) same chipset as the daisy’ - ‘new’ may mean the one referred to above or it might be newer.

Bela - is a cap for a Beagle Bones board - seems like the primary goal is low latency, partly for research purposes.

PureData Personal Synth
This is a system on hackaday which is essentially an open source clone of the organelle https://hackaday.io/project/160754-puredata-portable-synth

Raspberry Pi - Linux

Interesting article-website on audio latency, how to lower it, and Jack
https://wiki.linuxaudio.org/wiki/raspberrypi

There is an adapter board available to put a pi 4 compute element into a DIMM (SODIMM?) socket -
https://www.gumstix.com/manufacturer/raspberry-pi/cm4-uprev.html
Gumstix Raspberry Pi CM4 Uprev
BACKORDERED PKG900000001454
$30.00

JACK (Jack Audio Connection Kit) and ALSA (Advanced Linux Sound Architecture)

Both under one ‘bolded heading’ because they are closely related but separate.
Good intro at:
https://sandsoftwaresound.net/get-started-alsa-jack/

SuperCollider

Programming language - SuperCollider is an environment and programming language originally released in 1996 by James McCartney for real-time audio synthesis and algorithmic composition. Since then it has been evolving into a system used and further developed by both scientists and artists working with sound. Wikipedia
Comes on the desktop in PatchBox
Operating system: FreeBSD, Linux, macOS, Windows
Written in: C++

Teensy - similar to Daisy, less powerful. Not specifically audio.

Qun Synth - Open Source? Software Synth built on an Esperif ESP32 Lyra development board with hardware (schematics published-available) front panel consisting of switches, a potentiometer which acts as a rotary encoder through software,

SamplerBox
https://www.samplerbox.org/
A Raspberry Pi based DIY project to play SD Card samples on a keyboard
Source-plans:
https://github.com/josephernest/SamplerBox

pduino - use pure data to control arduino boards - https://github.com/pd-externals/pduino

Monome Norns - Hard to describe - technobear mentions this in Musical Hacks #1 - Intro and hardware platform overview at https://www.youtube.com/watch?v=fczLNwFR3is&list=PL-kso3G9NXujt3HeE81KiMSTC9uo2pR2K

SuperCollider - I read this mentioned as an alternative to ORAC I think.

Daisy - there's a category for Daisy cos it's probably part of the music machine. Here is their road map:
https://trello.com/b/bPzhN0dc/daisy-roadmap
Chips in-on daisy:
Stm32h750 'value' family, or H7 family, specifically STM32H750IBK6
32-bit Arm Cortex-M7 480MHz MCUs, 128 Kbyte Flash 1 Mbyte RAM
AKM CoDec AK4556
$4 at DigiKey
https://www.akm.com/content/dam/documents/products/audio/audio-codec/ak4556vt/ak4556vt-en-datasheet.pdf
SDRAM
IS25LP Flash Memory 8MByte
? qspi memory - currently (2021-01-dd) only partially supported within Daisy-LibSP-DaisySP

I have a Field and several pods. I've loaded several sample 'patches' with the pod. The only thing I've done with the Field is run the program that comes loaded in it, Kind of a simple hardware 'exerciser'.

Next steps re pd2dsy - this uses the inactive 'heavy' compiler
Add support for Daisy Field.
Clean up board support to be metadata based, and a bit easier to add new boards
SDRAM support for larger buffers/delays.
Add support different types of controls (i.e. switches are currently only bangs, but should also be able to be toggles).
Double check licensing requirements, etc. and bundle a pre-compiled binaries for the arm toolchain into the project for built-in compilation
Add a GUI? (Web app / built in tester, similar to OWL platform?)
-From GitHub - electro-smith/pd2dsy: Utility for converting Pure Data (Vanilla) patches to Daisy projects.

Thetechnobear aka Mark Harris is on the Daisy forum.
As of 2021-02-07 there are 492 users on the Daisy forum

Some Notes on Zadig for Daisy USB issues

If you are on Windows you might need to use Zadig to select a different USB driver.

To begin, download Zadig. 21 Use the most recent version.

Launch Zadig and then put the Daisy into bootloader mode by holding BOOT and pressing RESET.
Click the Options menu in Zadig and select “List All Devices.”

In the drop down menu inside the main window, select “DFU in FS mode,” followed by selecting WinUSB in the target driver box.

Click “Replace Driver.” Zadig will update to the appropriate driver to flash Daisy.
These instructions are also in the Windows Toolchain Install Wiki page 8 on GitHub with screenshots.

edit: To clarify, Zadig is not only for command line programming but can be used for the Web Progammer as well on Windows. It should fix the connectivity problem if the USB driver is to blame.

List of synthesis techniques:

subtractive synthesis

additive (Fourier) synthesis

FM (analog, and (mainly) digital.

Sampling - many variations

Granular - some variation

Here are interesting links:

Here’s John Chowning talking about the ‘discovery’ of Frequency Modulation, with a good helping of general computer music development thrown in.
Dr. John Chowning | Knobcon 2019 - YouTube
https://www.youtube.com/watch?v=tpysRrYXxg4

Here’s the ‘1-bit’ link I was looking for:
https://phd.protodome.com/
Look at 4000AD on the right hand side

Other 1-bit pages:
1-bit stuff
https://pbat.ch/wiki/1bit/

I ran across a full, apparently real Open Source DAW called Muse (2021-02-2d). Haven't tried using it yet.

Pins for Ethernet?
Pin loc, name, alt function
A2 1 PE2 ETH_MII_TXD3
E3 12 PI10 ETH_MII_RX note-not present on LQFP100 & 144, present on UFBGA176 and LQFP176
M3 33 PC1 ETH_MDC
M4 34 PC2_C ETH_MII_TXD2
Only on TFBGA240+25 package - ETH_MII_TX_CLK, ETH_MII_CRS and other ETH
P2 42 PA2 ETH_MDIO
F4 43 PH2 ETH_MII_CRS
G4 44 PH3 ETH_MII_COL
R2 47 PA3 ETH_MII_COL
R3 53 PA7 ETH_MII_RX_DV/ETH_RMII_CRS_DV
N5 54 PC4 ETH_MII_RX_D0/ETH_RMII_RXD0
P5 55 PC5 ETH_MII_RX_D1/ETH_RMII_RXD1
R5 56 PB0 ETH_MII_RX_D2

Here's an ST training doc
STM32L4 SAI - Serial Audio Interface_Rev 3-1_27pages_en.STM32L4_Peripheral_SAI.pdf
https://www.st.com/content/ccc/resource/training/technical/product_training/0c/16/3b/b4/76/8a/47/51/STM32L4_Peripheral_SAI.pdf/files/STM32L4_Peripheral_SAI.pdf/jcr:content/translations/en.STM32L4_Peripheral_SAI.pdf

Google Electronic Music 'exhibition'

Ideas for interprocessor communication:

Use I2C and FUDI for the content - so we don't have to send audio at all.

Ethernet! (needs hardware, might not be allowed by seed pins)

Feather Specification from Adafruit - Feather is not audio-music oriented but is very similar in size and general idea to Daisy Seed
https://learn.adafruit.com/adafruit-feather/feather-specification

Microtonal Systems
Microtonal Encyclopedia
https://microtonal.miraheze.org/wiki/Main_Page
https://microtonal.miraheze.org/wiki/Early_developments_of_keyboards_with_more_than_12_notes_per_octave

Orthotonophonium

Zynthian - open source hardware/software "kit" built around the RPi. Looks like a nice case plus lots and lots of software integration work added to the stock RPi. Fairly active and recent user community (as per their forum). https://zynthian.org/

Update 2021-11-18 - after much discussion and building a Zynthian kit:
Possible Zynthian concepts-projects:
Porting ORAC or 'essence of ORAC' as a Zynthian 'synth'.
An organelle style synth - portable, battery operated, self contained
A 'high end' synth eg-ala Yamaha Montage, Roland Fantom - form factor largely determined by keybed
An 'Americun' Zynthian - changes - available pre-built, not a kit, definitions for many popular synths, include amp and speaker(s), see below for other possible changes. Could be a kickstarter-esque.

Possible areas of 'Contributions' for Zynthian
A file format-standard for 'voices'-banks that would include examples of use for an 'audition' function, along with the audio info for the instrument.
'Pedals' based on Zynthian - would the encoder's push button switches be robust enough?
Midi panic 'pull switch'
Multiprocessor, to include DAW, permit more instruments.
Support to use cores - does Zynthian already have?
Group buy of parts, including a keybed? Sell as kits to avoid regulations-testing,etc.
Use CM4 - any advantages?
Base it on something other than RPi - this is a biggie, but might offer some real advantages - consider, PC, other ARM variants, RiscV
A 'test' layer - unit diagnostics, MIDI-OSC scope, generate test signals
Built units
Include CV inputs and outputs (they-Zynthian do offer this already as an addon - there are traces on their Zynaptik PCB for it that are unpopulated)
Improve PureData integration (Obviously we have to check out the current implementation to know what-whether to improve.) I'm thinking let the patch use the Zynthian UI. Maybe implement the 'modular' concept of ORAC with a better UI.

More than 16 layers? 256? >1000?
Kickstarter - alternatives
Unfolded-spread out version
Gem or other video support
‘Linear’ (or circular) leds and buttons for a sequencer.
Improve build instructions - make them for latest version.
Change the midi leds so that the function of the circuit doesn’t depend on them.
A ‘macro’ knob(s) that can be configured through a hierarchical-flexible scheme to do what the musician wants.
‘Morphing’ functionality - between layers and maybe within a synth-layer as well, but that would depend on the layer.
Add an FPGA to handle ‘events’ more quickly than you can in software,
Add a Daisy class processor(s) for sound generation.
Write new-additional ‘synths’ - Sampling, FM (more operators than DX7-Dexed), Granular, etc.

An interesting Zynthian implementation:
https://www.programmersought.com/article/51275065199/

Zynadac parts:
+ DAC PCM5242 (Stereo Balanced Output, 32 bit, 384 KHz, 114-dB)
    Expected 2/1/2023 - mouser
    3/5/2023 - digikey
    3 3rd party dist have stock - RFQ for price
+ ADC PCM1862/63 (Stereo Balanced Input, 192 KHz, 102/110-dB)
    Expected 5/5/2023 - mouser
+ 3 x LDO NCP163 (ultra-low noise 6.5 mVRMS)
+ 2 X Oscillators DSC60 (high stability ±25 ppm)

Other Things to Check Out

https://github.com/konsumer/zerostomp-ideas
Cheap & easily-programmable effects pedal / synth that's easy to build.

https://github.com/t3db0t/PureeData
PuréeData is a browser-based GUI interface for a remote PureData server,

https://nagasm.org/ASL/icmc2003/closed/CR1010.PDF
Graphical Interface Editing Tool and Run-time Environment for Pure Data - written by someone at UCSD, presumably some connection with Miller Puckett. Interesting ‘Future Improvements’ paragraph at the end - including distributing midi among multiple machines.

https://ossia.io/ & https://ossia.io/score/about.html
‘allows to sequence OSC, MIDI, DMX, sound files and more, between multiple software and hardware. Its novel interactive timeline enables scores written with it to depend on external events and controls through a simple visual language.’

Check out: https://github.com/pierreguillot/Camomile
Camomile appears to be a VST plug-in that embeds PureData into the context of any DAW that accepts VST plug-ins.

https://youtu.be/Rt5IflidLDQ
Granular synthesis demo with PureData. Ends with playing ‘Soon May the Wellerman Come’ using 4 voices, which is all this ‘worst possible implementation’ of granular synthesis can achieve.

https://www.youtube.com/watch?v=vkQQ_fZPFGg
Pure Data book: Electronic Music and Sound Design: Harmonic Beats
Very cool demo!

Opensource MIDI controller
Dart controllers are designed to be replicable in any digital manufacturing laboratory. They are based on Arduino and their hardware / software is documented with simple examples, making this system ideal for educational workshops and DIY enthusiasts. Dart is offered as a complete unit in various models and as a DIY kit.
https://dartmobo.com/

Bizzarro Pure Data patch with embedded oscilloscope displays
pure data algorithmic beat
https://www.youtube.com/watch?v=DGiT55likdo

This is an interesting artform - live typing into Pure Data - music starts about 2 min in - reminds me of Sonic Pi, a language designed to be modified live.
Pure Data Livecoding
https://www.youtube.com/watch?v=jADuLtsFqkk

Pretty advanced music from biomolecular-physical stuff such as spider webs
Materiomusic: Making a Virus Sing
https://www.youtube.com/watch?v=UZkFhgDjcpI

https://github.com/zhdk/tpf-client Multisite music performance client written in PD

Potentially Interesting PureData externals

HCS (grabbag of misc utilities)

Cyclone

Ggee

GrIPD - maybe more than just an 'external' - A Graphical Interface Editing Tool and Run-time Environment for Pure Data

http://msp.ucsd.edu/tools/quacktrip/

ELSE - 401 Objects

'Mr. Peach' - a 'collection of general purpose objects' - but might be abandoned-[connected with Pd-extended].

Ext13

https://github.com/zhdk/tpf-client. Multisite music performance client written in PD

Automatonism is a modular synthesiser that runs in the open source programming language Pure Data. This could be a good app to try running on TMM. There is a nice little video on this intro page:
https://www.automatonism.com/the-software
5/5/2021-I tried out Automatonism a bit on a Linux PC, in preparation for trying it on a '[3 Raspberry Pi music machine prototype]'. It does have a 'midi in', (and out) module available. I did NOT get any sound out yet. I didn't see any obvious way to just 'turn on' sound. I don't think it will be a problem to put a 'dac' on to hear it. And it looks like it has a very nice set of modules, help files and 'tricks' put together.

Synthberry Pi, A Standalone Pure Data Synth Based On Raspberry Pi
https://www.synthtopia.com/content/2020/02/01/synthberry-pi-a-standalone-pure-data-synth-based-on-raspberry-pi/

Audiolab on deken and https://github.com/solipd/AudioLab "a focus on electroacoustic composition, live electronics and sound design"

Organelle Software - I read somewhere that it was 'open source'. In combination with **ORAC** it would be a strong contender for residence on TMM.
Found: https://github.com/critterandguitari/Organelle_OS
The 'license' file from January 22, 2016 is BSD 3

The latest update I see is from 15 months ago, the latest update to main.cpp is 16 months ago (as of 5/6/2021)

Here's an interesting summary from technobear responding to a question about whether Organelle is 'open source'. I'm fairly certain that he is talking about the original Organelle here, NOT the 2nd generation based on a Raspberry Pie module:

Sep '17
@dbetz
ok, so pure data is what I would call the 'patchers interface' into organelle…
but given your talking about C++/open source, I'm guessing your more a developer, so perhaps want the 'less simplified view' :wink:

to understand the software layer, we first need to talk about hardware layers

pots, encoders ,keyboard switches and lcd are all connected to an MCU (STM32F0? , need to check that).
this MCU is then connected to a SOC (solid run , see other threads) via a serial link.

so the MCU encodes/decodes serial messages to communicate with the hardware and the SOC.
this software is open source, Organelle_Controller, but you would need to open the organelle to re-program it… this is no issues, since its very generic, so is not needed.

the SOC, has the Arm CPU, codec, usb etc… and runs an appropriate arch linux build for the SOC.
also on start up its runs the 'mother host' (aka Organelle_UI) written in C++,
its listens for OSC messages about the LCD, and coverts these, and sends then via serial line to the MCU. it also listens on the serial lines for msgs re pots/keys etc, and broadcasts these via OSC.
mother hosts, also provides the 'user menu' the users sees, and so also launches pure data when the user selects a patch.

on the pure data side, as Patrick said, there is a mother.pd patch loaded alongside the user patch which converts to/from OSC messages to pure data 'send/recieve'

(for audio , the SOC provides a standard audio interface which is interfaced directly via alsa)

phews… so back to the original question.
...snip…
I think organelle is a great platform… I think it excels, when things are written specifically for it… which can be optimised for its hardware. (rather than ports of things designed for PCs - which will tend to be a compromise)

some might read the above and think, all seems simple enough, they could implement this on a rPI, and they would be right , theoretically, if they have the skill set :wink: .
but are also missing some key elements

its quite a bit of work, to actually create the above, and make it work seamlessly together
a common form factor (# encoders, lcd) etc, whilst provides limits, it also provides a target platform, so patchers can target this… i.e. we can share patches
without this commonality, everyone has to write their own ‘instrument’, which is exactly what you see on the rPI… a few hackers build great things, but not something a community can be built around.
sorry, that went probably a bit off-topic.
I’m an enthusiastic developer in this area, I also build things with Axoloti, Bela and rPI (each has its pros/cons)

From oweno:
Most of this info is kicking around, but basically the hardware is:

hardware (keys, knobs, screen) --> STM32F051 MCU --> i.MX6 SOM (via serial) --> SGTL5000 Audio Codec (via I2S) --> audio jacks
USB / SD / HDMI wired to SOM

ORAC - Originally for Organelle, but ORAC 2.0 also supports Raspberry Pi and EuroRack module.

Orac 2.0 Overview and new features
https://www.youtube.com/watch?v=mywEOeth40Y

Orac 2.0 for Raspberry PI
https://www.youtube.com/watch?v=eunF5yBVrkM

Orac 2.0 - Getting Started Guide
https://www.youtube.com/watch?v=0XVzwFN4iLc

Here are 2 videos by Mark Harris, ‘The TechnoBear’ on creating-converting modules:

Orac: Creating modules
https://www.youtube.com/watch?v=ilHzy4mNu3w

Orac : Converting patches to modules
https://www.youtube.com/watch?v=w_a11cry1f8

Click Tracker
http://jmmmp.github.io/clicktracker/

"Metronome" app for conducters, performers, composers.

tanh~: soft clipping / distortion

Hcs - The 'hcs' library is a random grabbag of objects that are experiments that sometimes lead to full-fledged libraries. (fairly widely used)

Soundtouch~: pitch shifting library
Olli Parviainen's pages for the SoundTouch library, it's method, and pitch shifting in general:
http://www.surina.net/soundtouch/
http://www.surina.net/article/time-and-pitch-scaling.html
My pages on pitch shifting, and on soundtouch~ for Pure Data:
http://www.katjaas.nl/pitchshift/pitchshift.html
http://www.katjaas.nl/pitchshift/soundtouch%7E.html

Ekext

This library is a collection of objects for analyzing audio to get musical information, like spectrum and peak information, to generate sound based on analysis, like Linear-Predictive Coding, and for working with polyphony.

Raspberry Pi Audio

Interesting article-website on audio latency, how to lower it, and Jack
Includes partial over-clocking mainly to avoid CHANGING cpu freq, disabling services - also found how to support Pi built in audio with Jack via MMAP support
https://wiki.linuxaudio.org/wiki/raspberrypi

PInstrument

https://github.com/TricksterSam/PInstrument

https://www.youtube.com/watch?v=4OKeFUdwkBM&list=PLgyLwxm56Dvqr4tt8lrNM1o7uOgnfDGMi

I've shared my code for my PInstrument, a set of modules for audio synthesis built in Pure Data designed to run on a small touch screen. Specifically, the interface is designed to work on a 320x240 2.8 inch TFT Capacitive Touchscreen but it will work on any larger screen. The PInstrument is also designed to run on a low-power CPU like those in a Raspberry Pi 3 or 4. It was also designed with a 4in/8out DC-coupled

interface to be used with a Eurorack synthesizer. As of 2021, Expert Sleepers produces both the ES-8 and ES-9 that will solve these purposes.

I use it as a rack-mounted touch-screen in conjunction with an ES-8 to add functionality to my Eurorack setup without needing a big laptop on the desk.

In it you'll find a number of modules:
Relabi Waves - a four channel chaotic oscillator
Turing Machine
Phase Sequence
Wave Folder
VCA Panner - panner and four channel mixer
Quad Cosines
Clocks - a four channel clock-divider and clock-synced wave generator
Drums - a quick and dirty set of synth drum sounds
Pitch Tracker - three channels of pitch tracking for V/O control
You can certainly run this wherever and however you want. You might even find some of the modules and abstractions useful. Using Eurorack has fundamentally shifted some of my approaches to Pure Data coding and I wanted to see what I could accomplish here. You can run it with Eurorack with Expert Sleepers or just run it stand alone on your computer.

I believe it's all vanilla Pd. So there shouldn't be any compatibility issues.

Feel free to fork or get in touch if you want to contribute. This is an evolving project, but I think it's reached a level of maturity that others might benefit. If you want help setting it up to boot from startup on a Raspberry Pi, get in touch.

3 Net Objects

I'm presenting 3 net objects:

[nicinfo]
- prints to the console the associated ip(s) of your network card(s).
also prints your outside ip if you have internet.

[webserver]
- ipv4 http server (threaded).

[websocketserver]
- a multi-threaded web-socket server. (not ready for macOS yet [1]).

You can get them from Deken.

Github repositories:

https://github.com/Lucarda/pd-nicinfo

https://github.com/Lucarda/pd-webserver

https://github.com/Lucarda/pd-websocketserver

earplug~cRealtime binaural filter (HRTF) based on KEMAR impulse measurement

earplug~ is a binaural filter based on KEMAR impulse measurement which allows you to spatialize a sound in realtime. It basically takes the KEMAR dummy head data set, and interpolates 366 locations where HRTF measurement exists in a spherical surface. You get azimuth control 0 - 360 degrees and elevation -40 - 90 Degrees.

https://github.com/pd-externals/earplug/tree/main

Scheme for PureData
https://github.com/iainctduncan/scheme-for-pd
Scheme for Pd (s4pd) is an open-source external for live-coding and scripting Pd with an embedded s7 Scheme Lisp interpreter. It is a port of most of Scheme for Max, by the same author, for Max/MSP.

Music Machine Control Surface Keyboard Notes-Ideas:

Some ideas relate to Pure Data:
Several Color Coded bang buttons.
Edit-Play mode switch
Volume Control - Stereo?
A ‘setup’ button that invokes preferences-options?

Some would relate to DAW:
Transport Control
Record Arm and Record ala Reaper

Some would relate to multi-threading, multi-cpu, ‘networking’

The above ‘color coding’
A screen-display - big questions on what would drive it and what ‘level’ it applies to - presumably the ‘patch’ could write to it. Another ‘level’ would be the Pure Data GUI - this requires a ‘large’-’expensive’ screen

Linux Audio
http://www.tedfelix.com/linux/linux-midi.html

Get started: Linux ALSA and JACK
https://sandsoftwaresound.net/get-started-alsa-jack/

Notes on Block Diagram

Question on 4 USB A Connectors.

Could not File, Download as ODP.

We could start with just the Ethernet on the CM4s, and only add the additional port if needed.

We can start with one processor, use techniques that would also work for multiple.

Vision5 Notes:

Audio is on schematic page 11

Notes on URLs-Links-etc. For VisionFive Linux Ubuntu Debian

cristicc/visionfive-debos
Experimental *Debian* based OS and application for *VisionFive* SBC.
https://github.com/search?q=visionfive%20debian&type=repositories

RISCV Full System

This document provides instructions to create a riscv disk image, a riscv boot loader (berkeley bootloader (bbl)) and also points to the associated gem5 scripts to run riscv Linux full system simulations. The boot loader bbl is compiled with a Linux kernel and a device tree as well.

The used disk image is based on busybox and UCanLinux. It is built using the instructions, mostly from here.
here=https://github.com/UCanLinux/riscv64-sample
https://gem5.googlesource.com/public/gem5-resources/+/HEAD/src/riscv-fs/README.md

https://rvspace.org/en/project/VisionFive2_Debian_Wiki_202303_Release
Contains a link:
https://debian.starfivetech.com/
Which offers you a choice of
VisionFive2 Debian

Baidu Cloud Disk - Chinese only
https://pan.baidu.com/s/146_K7BNT0cBfMUeTpvb3uA?pwd=csx2
Google Cloud Disk
https://drive.google.com/drive/folders/1saKcvhEMv7_ncX8VM7JWidCE2crdJgXA